triton consulting

Db2 HADR Automation with Pacemaker

By Damir Wilder

Db2 HADR Automation with Pacemaker

Table of Contents

Introduction

Pacemaker is an open-source, high availability cluster manager software integrated with Db2 Advanced Edition and Db2 Standard Edition on Linux (no support announced for AIX/Windows). It provides high availability and disaster recovery capabilities for on-premises deployments and non-container cloud environments, such as Amazon Web Service (AWS). And it is meant as the replacement for the old TSAMP technology.

Those who have ever tried to automate the HADR failovers in the past, using TSAMP as the cluster manager, know how cumbersome this technology is and how difficult it has been working with it. From past experiences, it got stuck or corrupted many times so thoroughly that the only way to fix the problem was to completely remove the cluster (i.e. drop the TSAMP Domain) and recreate it from a scratch. This is hardly a practice suitable for production environments, the native ground for TSAMP. In some cases, it was actually easier to drop TSAMP altogether and live on without the automated failovers.

Therefore, there are high hopes for Pacemaker in the Db2 community, in that it will offer us a much simplified and easier way of managing HADR clusters and automating the failovers between the HADR nodes.

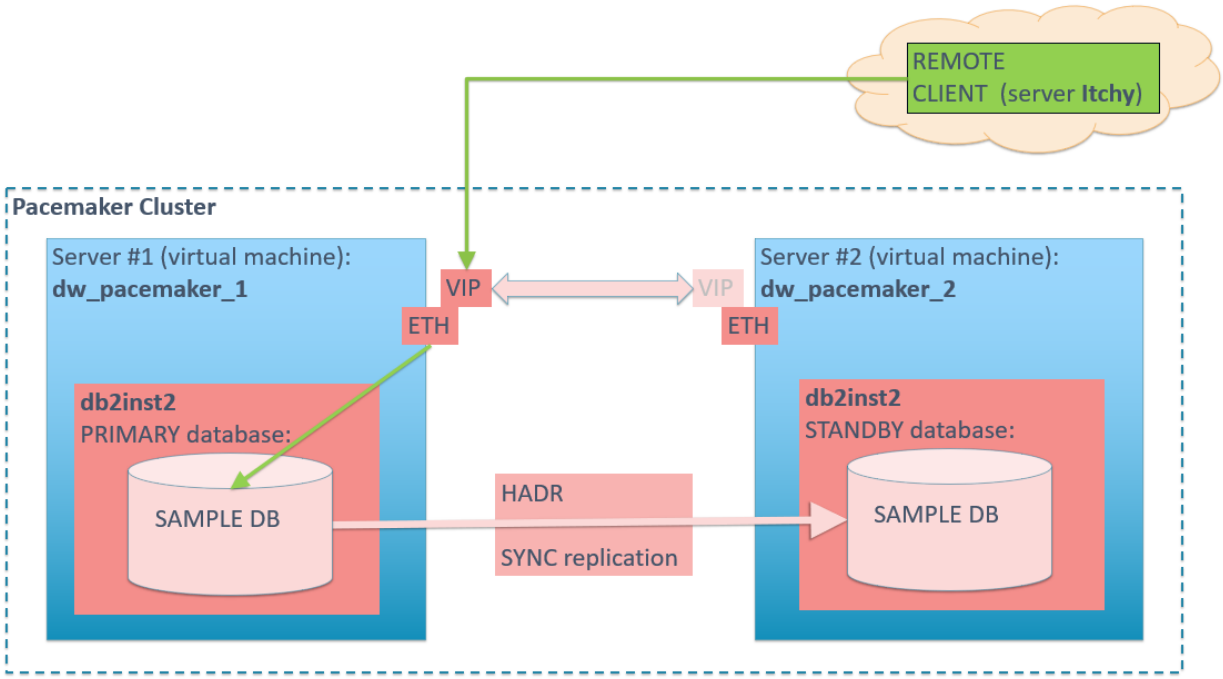
In this article Damir Wilder shares his experience on working with Pacemaker starting with prerequisites for the Pacemaker installation and the installation procedure itself. This is followed by configuration of a simple clustered environment consisting of two nodes and setup of a HADR Db2 database within it. To illustrate the workings of Pacemaker and HADR, a series of tests will then be described with or without active clients on the database to find out what happens with the servers, the database and the client connections during and after a failover.

Environment Setup

In our setup, we use two hosts to set up the Pacemaker HADR cluster (we will not use a third host as the QDevice quorum, since that is not strictly required as explained below). In reality these hosts are virtual machines residing on the same hardware server, but that's irrelevant for this test as we will not pay any attention to the cluster performance.
The hosts that will be used are:

dw_pacemaker_1 IP=192.168.56.107
dw_pacemaker_2 IP=192.168.56.108

We will use another server called **Itchy** to act as our remote client.



PACEMAKER prerequisites

Important: In Version 11.5 Mod Pack 4, using Pacemaker as a cluster manager in an automated failover to HADR standby is a Technical Preview. This means it should be restricted to development, test, and proof of concept environments only. Any output and error messages from the new db2cm utility may change in the final version of this feature.

The integrated Pacemaker high availability (HA) solution is available on the following Linux distributions:

- Red Hat Enterprise Linux 8.1 (Intel Linux and Linux on IBM Z)
- SuSE Linux Enterprise Server 15 SP1 (Intel Linux and Linux on IBM Z)

QDevice quorum mechanism

This is the recommended quorum mechanism for a production system and a best practice for Db2.
This requires a third host to install the corosync-qnetd software (available in the public IBM site) to act as the arbitrator. The host itself is not required to be part of the cluster and does not require the **Db2** server to be installed.

The host used must be accessible via TCP/IP to the other two hosts in the cluster.
The IP address specified in the setup of the QNetd server must be the same one that the Pacemaker/Corosync used to communicate.

All clusters using the QNetd server must have unique cluster names.

NOTE: QDevice set up on a separate host is not strictly required and when not, one of the nodes in the cluster will assume the role of the arbitrator.

Db2 high availability disaster recovery (HADR)

For the HADR functionality, perform the following tasks:

- Ensure that both HADR databases exist on different systems.
- Ensure all HADR databases are started in their respective primary and standby database roles, and that all HADR primary-standby database pairs are in peer state.
- Ensure that users are using one of the following HADR synchronization modes: **SYNC** or **NEARSYNC**.
- Configure **hadr_peer_window** for all HADR databases to a value of at least 120 seconds.
- Disable the **Db2** fault monitor.

PACEMAKER installation

The installation instructions were taken from the following article: https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG_11.5.0/com.ibm.db2.luw.admin.ha.doc/doc/install_pacemaker_cluster.html?pos=2)

Pre-setup checklist

1. Instance user ID and group ID are setup (**db2inst2:db2iadm2**)
2. /etc/hosts are setup with both hosts in it following the format listed in the prerequisite section
3. Both hosts have TCP/IP connectivity between their Ethernet network interfaces
4. Both root and instance user ID can use ssh between the two hosts, using both long and short host names.

NOTE: for this to work, generate the SSH key pairs (**ssh-keygen -t rsa**) and exchange public keys between the servers (**ssh-copy-id**) for both root and db2inst1 users.

NOTE2: if db2inst1 home directory is not under /home, then some wizardry must be performed in order for the SSH key exchange to work:

```
semanage fcontext -a -t ssh_home_t '/db2inst1/.ssh/authorized_keys'
restorecon -v '/db2inst1/.ssh/authorized_keys'
```

5. The Pacemaker cluster software has been downloaded to both hosts.

Installation

All steps are done on both hosts:

1. Extract the tar file in the /tmp folder

```
cd /tmp
tar -zxf Db2_v11.5.4.0_Pacemaker_20200615_RHEL8.1_x86_64.tar.gz
```

This creates the directory `Db2_v11.5.4.0_Pacemaker_20200615_RHEL8.1_x86_64` with the following subdirectory tree:

```
Db2/
Db2agents/
RPMS/
RPMS/<architecture>
RPMS/noarch
SRPMS/
```

2. For RHEL 8.1 (as is the case here) install the epel-release, followed by the RPMs in the untarred Pacemaker directory:

```
cd /tmp/Db2_v11.5.4.0_Pacemaker_20200615_RHEL8.1_x86_64/RPMS
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-
latest-8.noarch.rpm
dnf install */*.rpm
```

3. Verify that the following packages are installed. The output may vary slightly for different architectures and Linux distributions. All packages should include the **db2pcmk** text in the output. For example:

```
[root@...]# rpm -q corosync
corosync-3.0.3-1.db2pcmk.el8.x86_64
[root@...]# rpm -q pacemaker
pacemaker-2.0.2-1.db2pcmk.el8.x86_64
[root@...]# rpm -q crmsh
crmsh-4.1.0-0.db2pcmk.el8.noarch
```

4. Copy the **db2cm** utility from the cluster software directory into the instance sqllib/adm directory:

```
cp /tmp/Db2_v11.5.4.0_Pacemaker_20200615_RHEL8.1_x86_64/Db2/db2cm
/home/db2inst2/sqllib/adm
chmod 755 /home/db2inst2/sqllib/adm/db2cm
```

5. Copy the resource agent scripts (**db2hadr, db2inst, db2ethmon**) from /tmp/Db2agents into /usr/lib/ocf/resource.d/heartbeat/ on both hosts:

```
/home/db2inst2/sqllib/adm/db2cm -copy_resources
/tmp/Db2_v11.5.4.0_Pacemaker_20200615_RHEL8.1_x86_64/Db2agents -host
<hostname>
```

Check the tools (**db2ethmon, db2hadr, db2inst**) have been copied OK:

```
ls -alL /usr/lib/ocf/resource.d/heartbeat/db2*
-rwxr-xr-x. root 25484 .. /usr/lib/ocf/resource.d/heartbeat/db2ethmon
-rwxr-xr-x. root root 36129 .. /usr/lib/ocf/resource.d/heartbeat/db2hadr
-rwxr-xr-x. root root 22968 .. /usr/lib/ocf/resource.d/heartbeat/db2inst
```

Disable the Db2 Fault Monitor

If the Db2 Fault Monitor is running (and it is), it should be disabled now:

```
[root@dw_pacemaker_1]/root# ps -ef | grep db2fmcd
```

```
root   855   1   0 Aug07 ?   00:00:38   /opt/ibm/db2/V11.5.4_fp0/bin/db2fmcd
```

To disable the Db2 Fault Monitor for the entire server, navigate to the Db2 **bin** directory and run (as **root**):

```
cd /opt/ibm/db2/V11.5.4_fp0/bin
db2fmcu -d
```

Check that there are no more **db2fmcd** processes running:

```
[root@dw_pacemaker_1]/root# ps -ef | grep db2fmcd
  [no processes shown]
```

Repeat this procedure on all nodes in the Pacemaker cluster.

Configure a Clustered Environment

The following steps need to be executed just once on any one of the hosts **by the root user**.

Create the Pacemaker cluster and the public network resources

Run the command as root:

```
/home/db2inst2/sqllib/adm/db2cm -create -cluster -domain HadrDomain
-host dw_pacemaker_1 -publicEthernet enp0s8 -host dw_pacemaker_2 -
publicEthernet enp0s8
Created db2_dw_pacemaker_1_enp0s8 resource.
Created db2_dw_pacemaker_2_enp0s8 resource.
Cluster created successfully.
```

Create the instance resource model

Run the following commands as root:

```
/home/db2inst2/sqllib/adm/db2cm -create -instance db2inst2 -host
dw_pacemaker_1
Created db2_dw_pacemaker_1_db2inst1_0 resource.
Instance resource for db2inst1 on dw_pacemaker_1 created
successfully.

/home/db2inst2/sqllib/adm/db2cm -create -instance db2inst2 -host
dw_pacemaker_2
Created db2_dw_pacemaker_2_db2inst1_0 resource.
Instance resource for db2inst1 on dw_pacemaker_2 created
successfully.
```

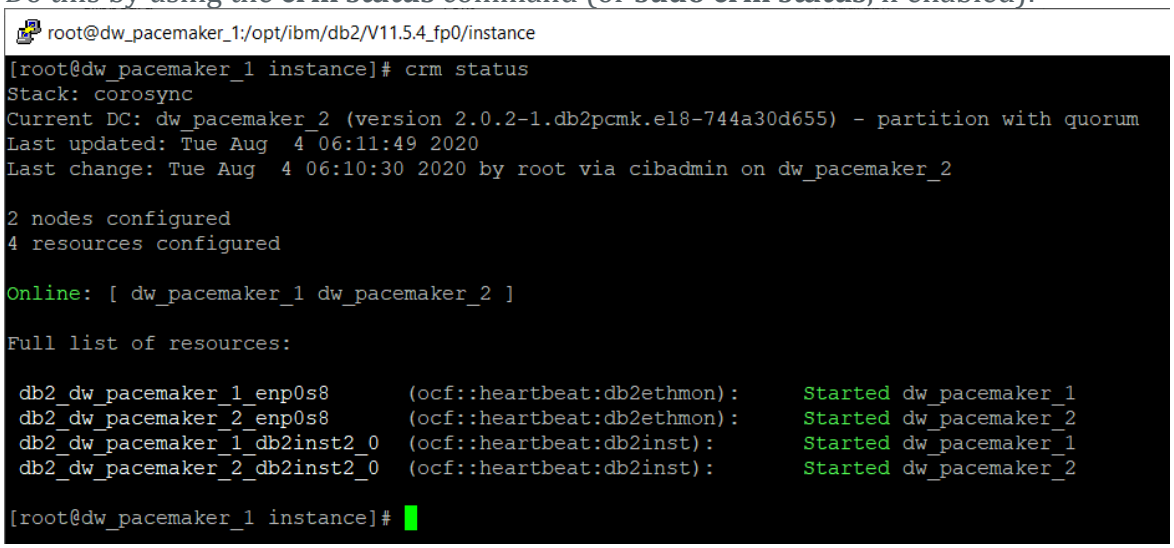
NOTE: if any of the commands fail, su to the inst.owner and start the Db2 instance beforehand!

Check (on both hosts) that the DBM CFG has been updated:

```
db2 get dbm cfg | grep -i cluster
 Cluster manager                                      = PACEMAKER
```

Check the cluster status

Do this by using the **crm status** command (or **sudo crm status**, if enabled):



NOTE: The Online parameter needs to include both hosts.

Create a new database

Run the **db2sampl** command on the server **dw_pacemaker_1** only:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2sampl

  Creating database "SAMPLE"...
  Connecting to database "SAMPLE"...
  Creating tables and data in schema "DB2INST2"...
  Creating tables with XML columns and XML data in schema
"DB2INST2"...

  'db2sampl' processing complete.
```

Configure HADR for the new database

Set up the log archiving for the SAMPLE database on **dw_pacemaker_1**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>mkdir db2inst2/logarchive
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 update db cfg for SAMPLE
using LOGARCHMETH1 'DISK:/home/db2inst2/db2inst2/logarchive/'
DB20000I  The UPDATE DATABASE CONFIGURATION command completed
successfully.
```

A bit more config.setup:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 update db cfg for SAMPLE
using LOGINDEXBUILD ON immediate
DB20000I  The UPDATE DATABASE CONFIGURATION command completed
successfully.
```

Backup the SAMPLE database to activate the log archiving:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 backup db SAMPLE

Backup successful. The timestamp for this backup image is :
    20200804084331
```

Update the HADR configuration for SAMPLE@**dw_pacemaker_1**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 update db cfg for SAMPLE
using HADR_LOCAL_HOST 192.168.56.107 HADR_LOCAL_SVC 51100
HADR_REMOTE_HOST 192.168.56.108 HADR_REMOTE_SVC 51100
HADR_REMOTE_INST db2inst2 HADR_TARGET_LIST '192.168.56.108:51100'
HADR_SYNCMODE SYNC HADR_PEER_WINDOW 120
DB20000I  The UPDATE DATABASE CONFIGURATION command completed
successfully.
```

Overview of the HADR parameters for the SAMPLE database at **dw_pacemaker_1**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 get db cfg for SAMPLE |
grep HADR
HADR database role                                    = STANDARD
 HADR local host name                (HADR_LOCAL_HOST) = 192.168.56.107
 HADR local service name              (HADR_LOCAL_SVC) = 51100
 HADR remote host name               (HADR_REMOTE_HOST) = 192.168.56.108
 HADR remote service name             (HADR_REMOTE_SVC) = 51100
 HADR inst.name of remote server(HADR_REMOTE_INST) = db2inst2
 HADR timeout value                      (HADR_TIMEOUT) = 120
 HADR target list               (HADR_TARGET_LIST) = 192.168.56.108:51100
 HADR log write synchro. mode       (HADR_SYNCMODE) = SYNC
 HADR spool log data limit (4KB) HADR_SPOOL_LIMIT) = AUTOMATIC(0)
 HADR log replay delay (sec.)   (HADR_REPLAY_DELAY) = 0
 HADR peer window duration(sec.)(HADR_PEER_WINDOW) = 120
 HADR SSL certificate label         (HADR_SSL_LABEL) =
```

Copy the backup image to **dw_pacemaker_2**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>scp
SAMPLE.0.db2inst2.DBPART000.20200804084331.001
192.168.56.108:/home/db2inst2
  SAMPLE.0.db2inst2.DBPART000.20200804084331.001     100%  169MB  55.1MB/s   00:03
```

Restore the database SAMPLE on **dw_pacemaker_2** from the backup image:

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>db2 restore db SAMPLE from .
taken at 20200804084331
DB20000I  The RESTORE DATABASE command completed successfully.
```

Update the HADR configuration for SAMPLE@**dw_pacemaker_2**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 update db cfg for SAMPLE
using HADR_LOCAL_HOST 192.168.56.108 HADR_LOCAL_SVC 51100
HADR_REMOTE_HOST 192.168.56.107 HADR_REMOTE_SVC 51100
HADR_REMOTE_INST db2inst2 HADR_TARGET_LIST '192.168.56.107:51100'
HADR_SYNCMODE SYNC HADR_PEER_WINDOW 120
```

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed
successfully.
```

Overview of the HADR parameters for the SAMPLE database at **dw_pacemaker_2**:

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>db2 get db cfg for SAMPLE |
grep HADR
 HADR database role                                      = STANDARD
 HADR local host name                  (HADR_LOCAL_HOST) = 192.168.56.108
 HADR local service name                (HADR_LOCAL_SVC) = 51100
 HADR remote host name                (HADR_REMOTE_HOST) = 192.168.56.107
 HADR remote service name              (HADR_REMOTE_SVC) = 51100
 HADR inst.name of remote server(HADR_REMOTE_INST) = db2inst2
 HADR timeout value                       (HADR_TIMEOUT) = 120
 HADR target list              (HADR_TARGET_LIST) = 192.168.56.107:51100
 HADR log write synchro. mode      (HADR_SYNCMODE) = SYNC
 HADR spool log data limit (4KB)(HADR_SPOOL_LIMIT) = AUTOMATIC(0)
 HADR log replay delay (sec.)   (HADR_REPLAY_DELAY) = 0
 HADR peer window duration(sec.)(HADR_PEER_WINDOW) = 120
 HADR SSL certificate label        (HADR_SSL_LABEL) =
```

Open the HADR port(s) between the two servers
(NOTE: this must be done as the **root** user):

```
[root@dw_pacemaker_1 instance]# firewall-cmd --zone=public --
permanent --add-port 51100/tcp
  success
[root@dw_pacemaker_1 instance]# firewall-cmd --reload
  success
```

...and:

```
[root@dw_pacemaker_2 instance]# firewall-cmd --zone=public --
permanent --add-port 51100/tcp
  success
[root@dw_pacemaker_2 instance]# firewall-cmd --reload
  success
```

Check that the port(s) are reachable:

```
[root@dw_pacemaker_1 instance]# ncat -zv 192.168.56.108 51100
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.56.108:51100.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

(NOTE: port 51100 is already open at this point on **dw_pacemaker_2**!)

```
[root@dw_pacemaker_2 instance]# ncat -zv 192.168.56.107 51100
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connection refused.
```

(NOTE: port 51100 is not yet open at this point on **dw_pacemaker_1**, but the error message "**Connection refused**" proves the host/port is reachable, as otherwise the error message "**No route to host.**" would have been returned!)

The servers should now be ready to join the HADR cluster!

Start HADR

First, start HADR on **dw_pacemaker_2** as standby:

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>date; db2 start hadr on db
SAMPLE as standby; date
Tue  4 Aug 08:59:11 EDT 2020
DB20000I  The START HADR ON DATABASE command completed successfully.
Tue  4 Aug 08:59:12 EDT 2020
```

Next, start HADR on **dw_pacemaker_1** as primary:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>date; db2 start hadr on db
SAMPLE as primary; date
Tue  4 Aug 09:34:43 EDT 2020
DB20000I  The START HADR ON DATABASE command completed successfully.
Tue  4 Aug 09:34:45 EDT 2020
```

Check the HADR status:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>hadrgap
                 HADR_ROLE = PRIMARY
               HADR_STATE = PEER
               HADR_FLAGS = TCP_PROTOCOL
       STANDBY_MEMBER_HOST = 192.168.56.108
       HADR_CONNECT_STATUS = CONNECTED
  HADR_CONNECT_STATUS_TIME = 04/08/2020 09:34:44.559679 (1596548084)
 PRIMARY_LOG_FILE,PAGE,POS = S0000000.LOG, 14, 44893726
 STANDBY_LOG_FILE,PAGE,POS = S0000000.LOG, 14, 44893726
       HADR_LOG_GAP(bytes) = 0
STBY_RPLY_LOG_FIL,PAGE,POS = S0000000.LOG, 14, 44893726
STBY_RCV_REPLAY_GAP(bytes) = 0
          PRIMARY_LOG_TIME = 04/08/2020 09:34:46.000000 (1596548086)
          STANDBY_LOG_TIME = 04/08/2020 09:34:46.000000 (1596548086)
   STANDBY_REPLAY_LOG_TIME = 04/08/2020 09:34:46.000000 (1596548086)
```

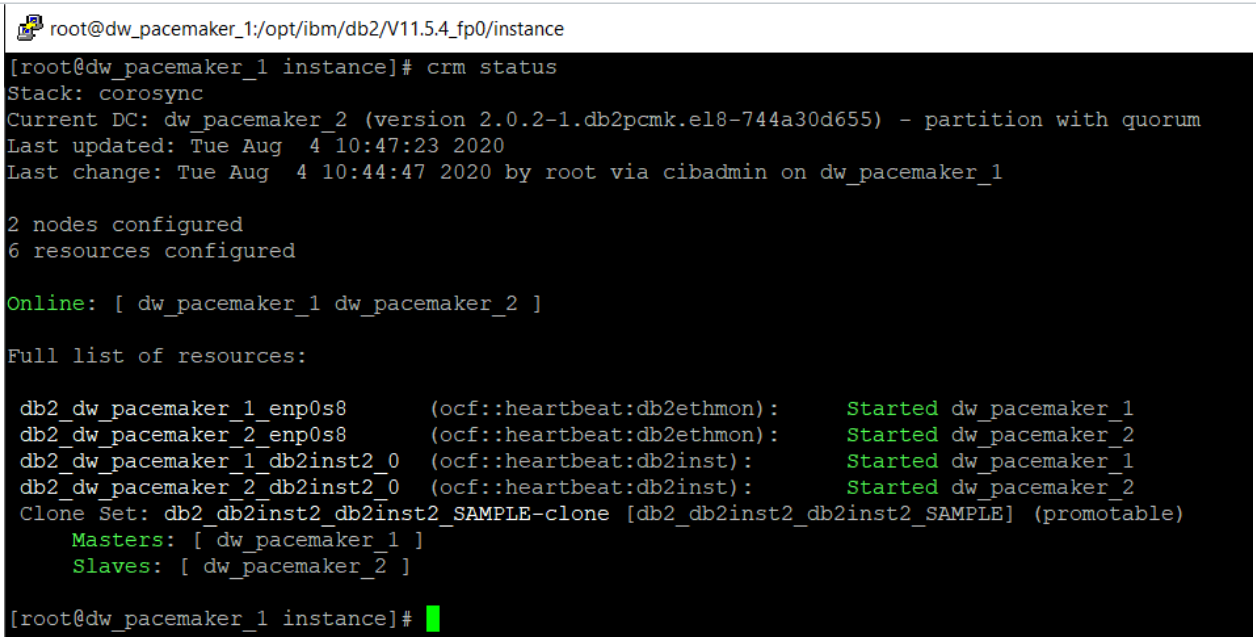
Create the HADR database resources

Run the following command as **root**:

```
[root@dw_pacemaker_1 instance]# /home/db2inst2/sqllib/adm/db2cm -
create -db SAMPLE -instance db2inst2
Database resource for SAMPLE created successfully.
```

Check the cluster status again

Do this by using the **crm status** command (or **sudo crm status**, if enabled):



NOTE:

- The **Online** parameter needs to include both hosts.
- **db2_<hostname>_enp0s8** is the public Ethernet resource in the resource model. There should be one on each host, and both should be labelled with the **Started** state
- **db2_<host_name>_<instance_name>_0** is the instance resource. There should be one for every instance.
- The database resource has **Master** and **Slaves** started on the respective hosts.

Alternatively, the cluster status can be shown using the **db2cm** command:

```
[db2inst2@dw_pacemaker_1]~>sudo ~/sqllib/adm/db2cm -list

      Cluster Status

Domain information:
Domain name                 = HadrDomain
Pacemaker version           = 2.0.2-1.db2pcmk.el8
Corosync version            = 3.0.3
Current domain leader       = dw_pacemaker_1
Number of nodes             = 2
Number of resources         = 6

Node information:
Name name           State
----------------    --------
dw_pacemaker_1      Online
dw_pacemaker_2      Online

Resource Information:

Resource Name               = db2_db2inst2_db2inst2_SAMPLE
  Resource Type                   = HADR
    DB Name                       = SAMPLE
    Managed                       = true
    HADR Primary Instance         = db2inst2
    HADR Primary Node             = dw_pacemaker_1
    HADR Primary State            = Online
```

```
    HADR Standby Instance       = db2inst2
    HADR Standby Node           = dw_pacemaker_2
    HADR Standby State          = Online

Resource Name              = db2_dw_pacemaker_1_db2inst2_0
  State                        = Online
  Managed                      = true
  Resource Type                = Instance
    Node                       = dw_pacemaker_1
    Instance Name              = db2inst2

Resource Name              = db2_dw_pacemaker_1_enp0s8
  State                        = Online
  Managed                      = true
  Resource Type                = Network Interface
    Node                       = dw_pacemaker_1
    Interface Name             = enp0s8

Resource Name              = db2_dw_pacemaker_2_db2inst2_0
  State                        = Online
  Managed                      = true
  Resource Type                = Instance
    Node                       = dw_pacemaker_2
    Instance Name              = db2inst2

Resource Name              = db2_dw_pacemaker_2_enp0s8
  State                        = Online
  Managed                      = true
  Resource Type                = Network Interface
    Node                       = dw_pacemaker_2
    Interface Name             = enp0s8

Fencing Information:
  Not Configured
Quorum Information:
  Two-node quorum
```

Disable the Pacemaker Cluster Resource Management

To disable the automatic management of the configured cluster resources, run the following command:

```
[db2inst2@dw_pacemaker_1]~>sudo ~/sqllib/adm/db2cm -disable
  Cluster manager is disabled.
```

This action does not stop the running Db2 resources (i.e. HADR cluster), both the primary and the standby database are still up and in connected/peer state:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>hadrgap
                            HADR_ROLE = PRIMARY
                           HADR_STATE = PEER
                           HADR_FLAGS = TCP_PROTOCOL
                  STANDBY_MEMBER_HOST = 192.168.56.108
                  HADR_CONNECT_STATUS = CONNECTED
             HADR_CONNECT_STATUS_TIME = 06/08/2020 06:22:17.660602 (1596709337)
            PRIMARY_LOG_FILE,PAGE,POS = S0000007.LOG, 102, 73786763
            STANDBY_LOG_FILE,PAGE,POS = S0000007.LOG, 102, 73786763
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000007.LOG, 102, 73786763
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 06/08/2020 08:00:57.000000 (1596715257)
                    STANDBY_LOG_TIME = 06/08/2020 08:00:57.000000 (1596715257)
               STANDBY_REPLAY_LOG_TIME = 06/08/2020 08:00:57.000000 (1596715257)
[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

Even though all resources are still shown in green colour (**Online**, **Started**), note the extra information on the High Availability in the Pacemaker Cluster Status:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_1 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Thu Aug  6 09:35:31 2020
Last change: Thu Aug  6 09:30:15 2020 by root via cibadmin on dw_pacemaker_1

2 nodes configured
6 resources configured

             *** Resource management is DISABLED ***
  The cluster will not attempt to start, stop or recover services

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1 (unmanaged)
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2 (unmanaged)
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1 (unmanaged)
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2 (unmanaged)
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable) (unmanaged)
     db2_db2inst2_db2inst2_SAMPLE       (ocf::heartbeat:db2hadr):       Slave dw_pacemaker_2 (unmanaged)
     db2_db2inst2_db2inst2_SAMPLE       (ocf::heartbeat:db2hadr):       Master dw_pacemaker_1 (unmanaged)

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

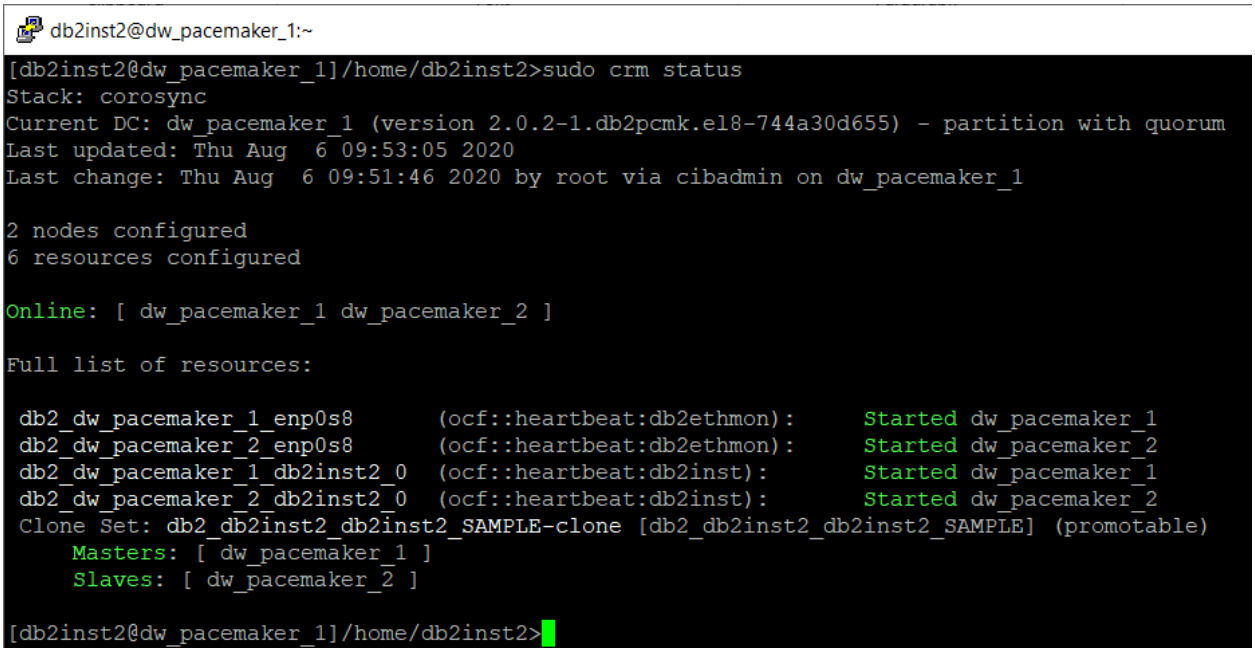
In conclusion, disabling the Pacemaker cluster resource manager takes away the "HA" out of the Db2 HADR!

Enable the Pacemaker Cluster Resource Management

To enable the automatic management of the configured cluster resources, run the following command:

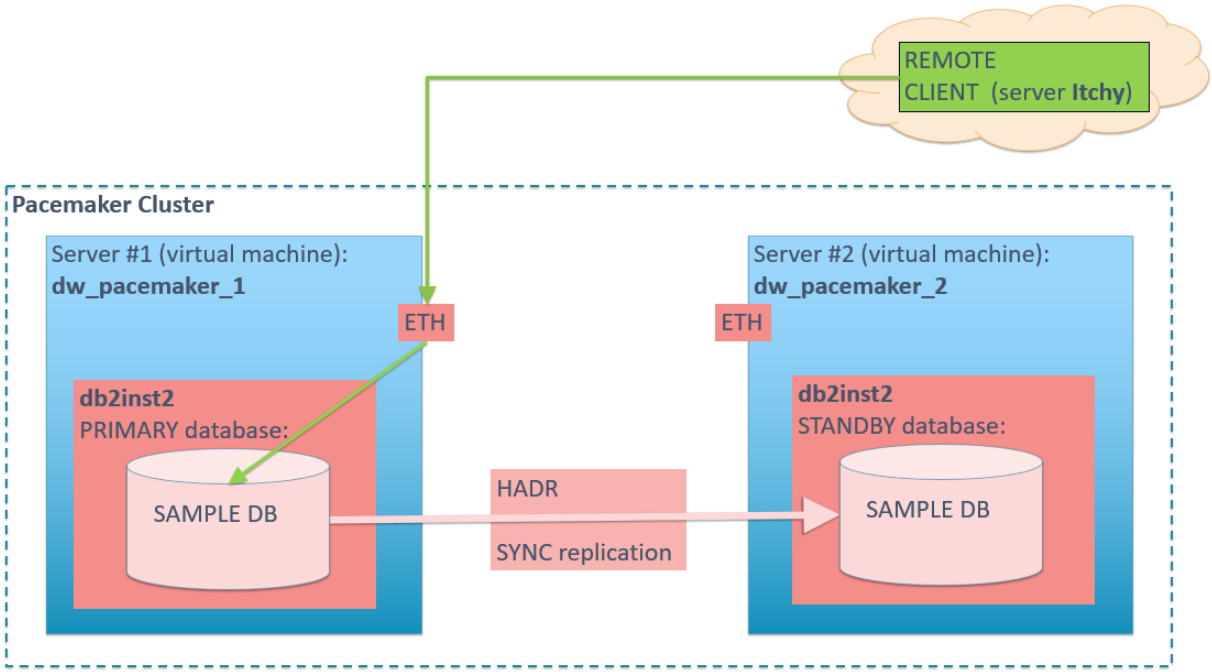
```
[db2inst2@dw_pacemaker_1]~>sudo ~/sqllib/adm/db2cm -enable
  Cluster manager is enabled.
```

The Db2 primary and standby databases are still up and running, but now they are returned under the HA control of the Pacemaker Cluster (the "DISABLED" info is not there anymore):


db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_1 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Thu Aug 6 09:53:05 2020
Last change: Thu Aug 6 09:51:46 2020 by root via cibadmin on dw_pacemaker_1
2 nodes configured
6 resources configured
Online: [dw_pacemaker_1 dw_pacemaker_2]
Full list of resources:
db2_dw_pacemaker_1_enp0s8 (ocf::heartbeat:db2ethmon): Started dw_pacemaker_1
db2_dw_pacemaker_2_enp0s8 (ocf::heartbeat:db2ethmon): Started dw_pacemaker_2
db2_dw_pacemaker_1_db2inst2_0 (ocf::heartbeat:db2inst): Started dw_pacemaker_1
db2_dw_pacemaker_2_db2inst2_0 (ocf::heartbeat:db2inst): Started dw_pacemaker_2
Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
Masters: [dw_pacemaker_1]
Slaves: [dw_pacemaker_2]
[db2inst2@dw_pacemaker_1]/home/db2inst2>

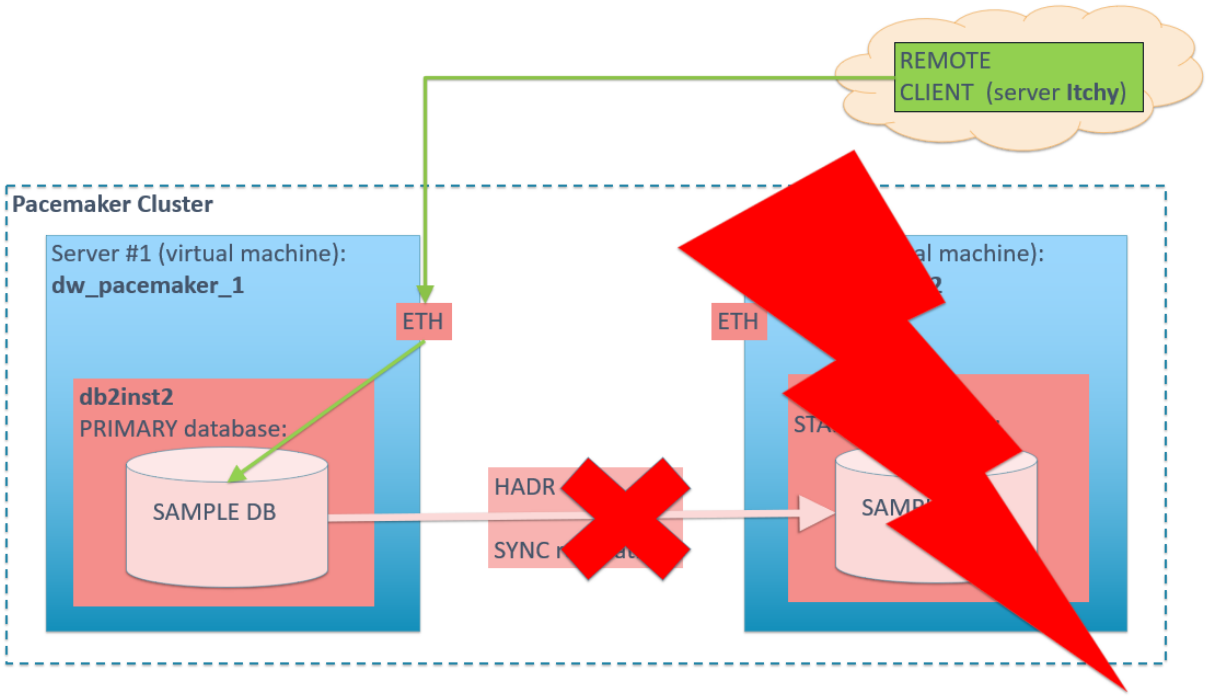
Testing the HADR Cluster

Having completed the configuration of the Pacemaker/HADR cluster resources, it is now time to run some tests to see how the cluster reacts:



TEST 1: Shut down the standby

In this test we start with both VMs up and running, the primary on **dw_pacemaker_1**, the standby on **dw_pacemaker_2**, HADR in Connected/Peer state; then we shut down the standby; observe what happens with the cluster; restart the standby; again observe what happens with the cluster:



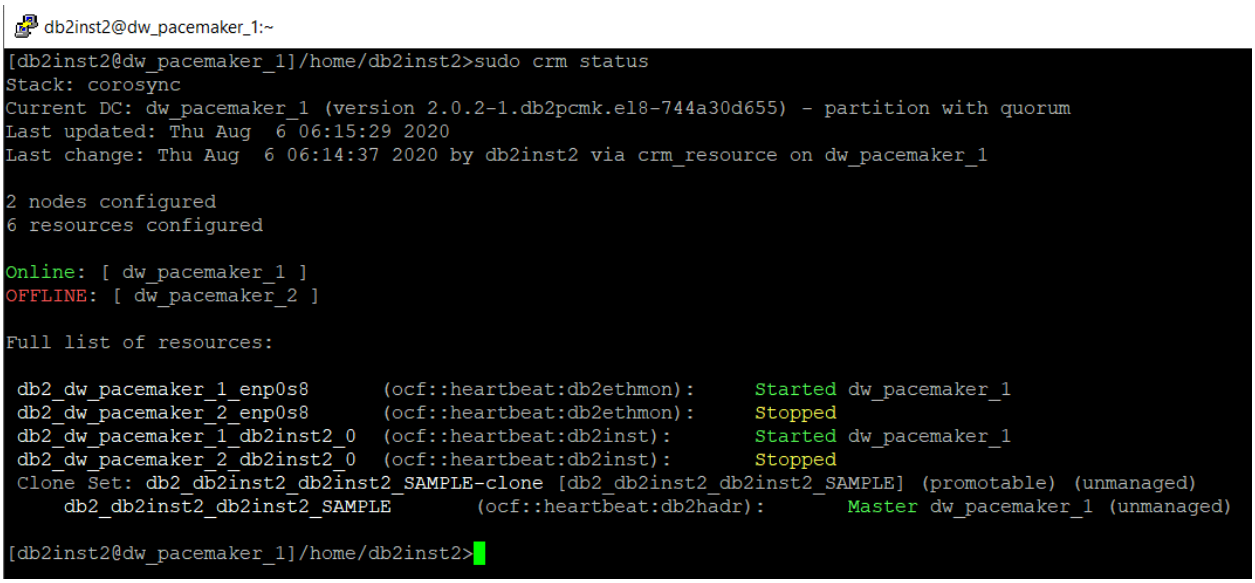
The current cluster status shows all resources to be on-line:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_1 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Thu Aug  6 06:12:33 2020
Last change: Thu Aug  6 05:54:54 2020 by root via crm_attribute on dw_pacemaker_2

2 nodes configured
6 resources configured

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_1 ]
     Slaves: [ dw_pacemaker_2 ]

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

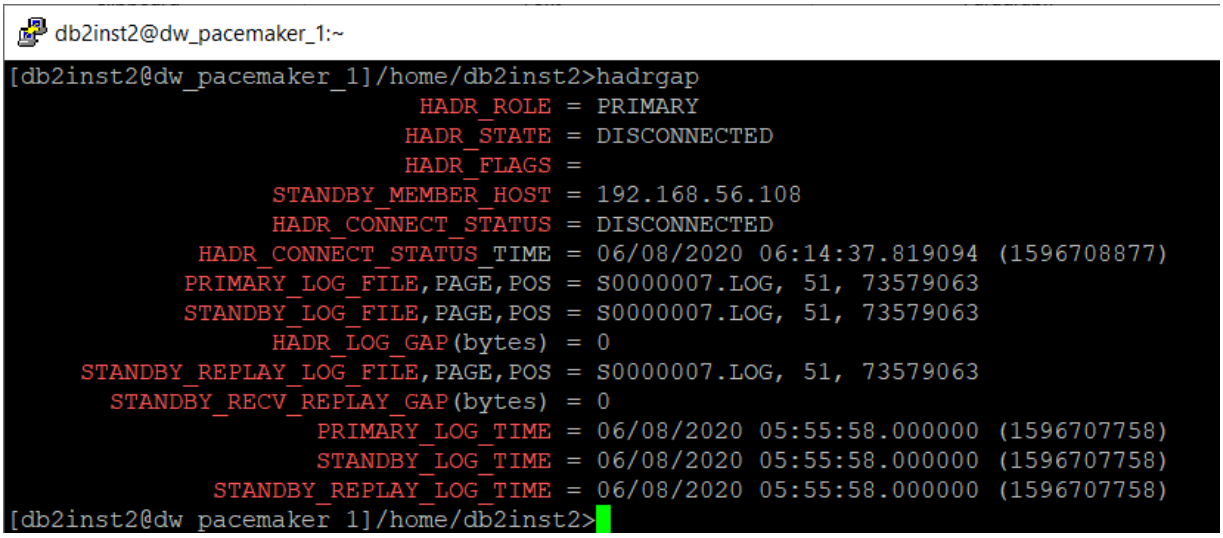
To stop the standby node **dw_pacemaker_2**, run the **shutdown** command as **root**:

```
[root@dw_pacemaker_2 instance]# shutdown
Shutdown scheduled for Thu 2020-08-06 06:14:45 EDT, use 'shutdown -
c' to cancel.
...
[root@dw_pacemaker_2 ~]# Connection to dw_pacemaker_2 closed by
remote host.
Connection to dw_pacemaker_2 closed.
```

The cluster status now shows **dw_pacemaker_2** to be offline, as expected:



HADR status on **dw_pacemaker_1** shows the primary is in the **disconnected** state:



Next, we restart the **dw_pacemaker_2** VM by executing:

```
vboxmanage startvm dw_pacemaker_2 --type headless
  Waiting for VM "dw_pacemaker_2" to power on...
  VM "dw_pacemaker_2" has been successfully started.
```

As soon as the VM is restarted, the cluster status changes back to "all green":


db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_1 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Thu Aug 6 06:23:48 2020
Last change: Thu Aug 6 06:22:18 2020 by db2inst2 via crm_resource on dw_pacemaker_1
2 nodes configured
6 resources configured
Online: [dw_pacemaker_1 dw_pacemaker_2]
Full list of resources:
db2_dw_pacemaker_1_enp0s8 (ocf::heartbeat:db2ethmon): Started dw_pacemaker_1
db2_dw_pacemaker_2_enp0s8 (ocf::heartbeat:db2ethmon): Started dw_pacemaker_2
db2_dw_pacemaker_1_db2inst2_0 (ocf::heartbeat:db2inst): Started dw_pacemaker_1
db2_dw_pacemaker_2_db2inst2_0 (ocf::heartbeat:db2inst): Started dw_pacemaker_2
Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
Masters: [dw_pacemaker_1]
Slaves: [dw_pacemaker_2]
[db2inst2@dw_pacemaker_1]/home/db2inst2>

…

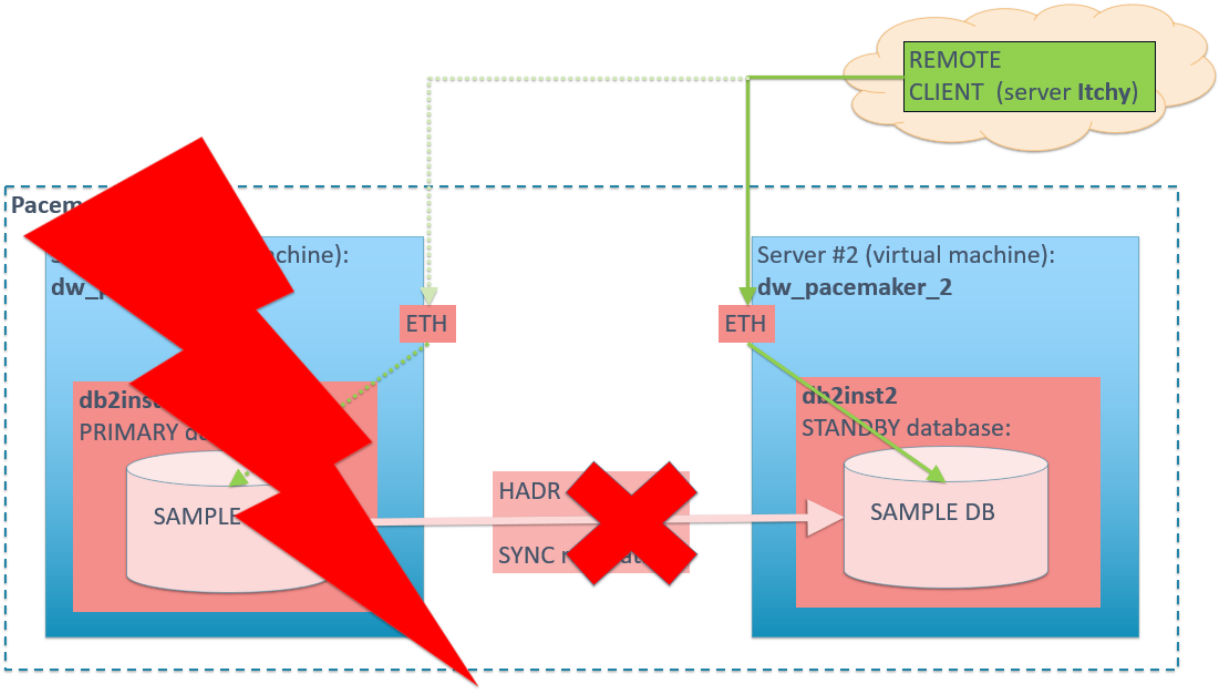
...and the standby node is automatically restarted and synced to the primary (no user intervention here!):

db2inst2@dw_pacemaker_2:~

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>hadrgap
                            HADR_ROLE = STANDBY
                           HADR_STATE = PEER
                           HADR_FLAGS = TCP_PROTOCOL
                  STANDBY_MEMBER_HOST = 192.168.56.108
                  HADR_CONNECT_STATUS = CONNECTED
             HADR_CONNECT_STATUS_TIME = 06/08/2020 06:22:17.659795 (1596709337)
            PRIMARY_LOG_FILE,PAGE,POS = S0000007.LOG, 51, 73579063
            STANDBY_LOG_FILE,PAGE,POS = S0000007.LOG, 51, 73579063
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000007.LOG, 51, 73579063
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 06/08/2020 05:55:58.000000 (1596707758)
                    STANDBY_LOG_TIME = 06/08/2020 05:55:58.000000 (1596707758)
               STANDBY_REPLAY_LOG_TIME = 06/08/2020 05:55:58.000000 (1596707758)
[db2inst2@dw_pacemaker_2]/home/db2inst2>
```

TEST 2: Shut down the primary

In this test we start with both VMs up and running, the primary on **dw_pacemaker_1**, the standby on **dw_pacemaker_2**, HADR in Connected/Peer state; then we shut down **dw_pacemaker_1** (the current primary); observe what happens with the cluster; restart **dw_pacemaker_1**; again observe what happens with the cluster:



The current cluster status shows all resources to be on-line:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_1 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Fri Aug  7 05:39:55 2020
Last change: Thu Aug  6 09:51:46 2020 by root via cibadmin on dw_pacemaker_1

2 nodes configured
6 resources configured

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_1 ]
     Slaves: [ dw_pacemaker_2 ]

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

HADR status on **dw_pacemaker_1** shows it to be the primary node and the cluster is in the **connected/peer** state:

db2inst2@dw_pacemaker_1:~

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>hadrgap
                            HADR_ROLE = PRIMARY
                           HADR_STATE = PEER
                           HADR_FLAGS = TCP_PROTOCOL
                  STANDBY_MEMBER_HOST = 192.168.56.108
                  HADR_CONNECT_STATUS = CONNECTED
             HADR_CONNECT_STATUS_TIME = 06/08/2020 06:22:17.660602 (1596709337)
           PRIMARY_LOG_FILE,PAGE,POS = S0000007.LOG, 844, 76810604
           STANDBY_LOG_FILE,PAGE,POS = S0000007.LOG, 844, 76810604
                 HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000007.LOG, 844, 76810604
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 07/08/2020 04:50:57.000000 (1596790257)
                    STANDBY_LOG_TIME = 07/08/2020 04:50:57.000000 (1596790257)
               STANDBY_REPLAY_LOG_TIME = 07/08/2020 04:50:57.000000 (1596790257)
[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

To stop the current primary node **dw_pacemaker_1**, run the **shutdown** command as **root**:

```
[root@dw_pacemaker_1 ~]# shutdown
Shutdown scheduled for Fri 2020-08-07 05:47:14 EDT, use 'shutdown -
c' to cancel.
[root@dw_pacemaker_1 ~]# Connection to dw_pacemaker_1 closed by
remote host.
Connection to dw_pacemaker_1 closed.
```

As soon as **dw_pacemaker_1** has stopped, the Pacemaker cluster status shows it to be **OFFLINE** and the associated resources to be **Stopped**:

db2inst2@dw_pacemaker_2:~

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_2 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Fri Aug  7 05:47:42 2020
Last change: Fri Aug  7 05:47:24 2020 by root via crm_attribute on dw_pacemaker_2

2 nodes configured
6 resources configured

Online: [ dw_pacemaker_2 ]
OFFLINE: [ dw_pacemaker_1 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Stopped
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Stopped
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_2 ]

[db2inst2@dw_pacemaker_2]/home/db2inst2>
```

The Db2 instance on **dw_pacemaker_2** has taken over the primary role and is expectedly in Disconnected state:

db2inst2@dw_pacemaker_2:~

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>hadrgap
                            HADR_ROLE = PRIMARY
                           HADR_STATE = DISCONNECTED
                           HADR_FLAGS =
                  STANDBY_MEMBER_HOST = 192.168.56.107
                  HADR_CONNECT_STATUS = DISCONNECTED
             HADR_CONNECT_STATUS_TIME = 07/08/2020 05:47:20.032709 (1596793640)
            PRIMARY_LOG_FILE,PAGE,POS = S0000008.LOG, 0, 77444046
            STANDBY_LOG_FILE,PAGE,POS = S0000000.LOG, 0, 0
                  HADR_LOG_GAP(bytes) = 0
     STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000000.LOG, 0, 0
       STANDBY_RECV_REPLAY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 07/08/2020 04:50:57.000000 (1596790257)
                    STANDBY_LOG_TIME = NULL
               STANDBY_REPLAY_LOG_TIME = NULL
[db2inst2@dw_pacemaker_2]/home/db2inst2>
```

Next, we restart the **dw_pacemaker_1** VM by running the following command:

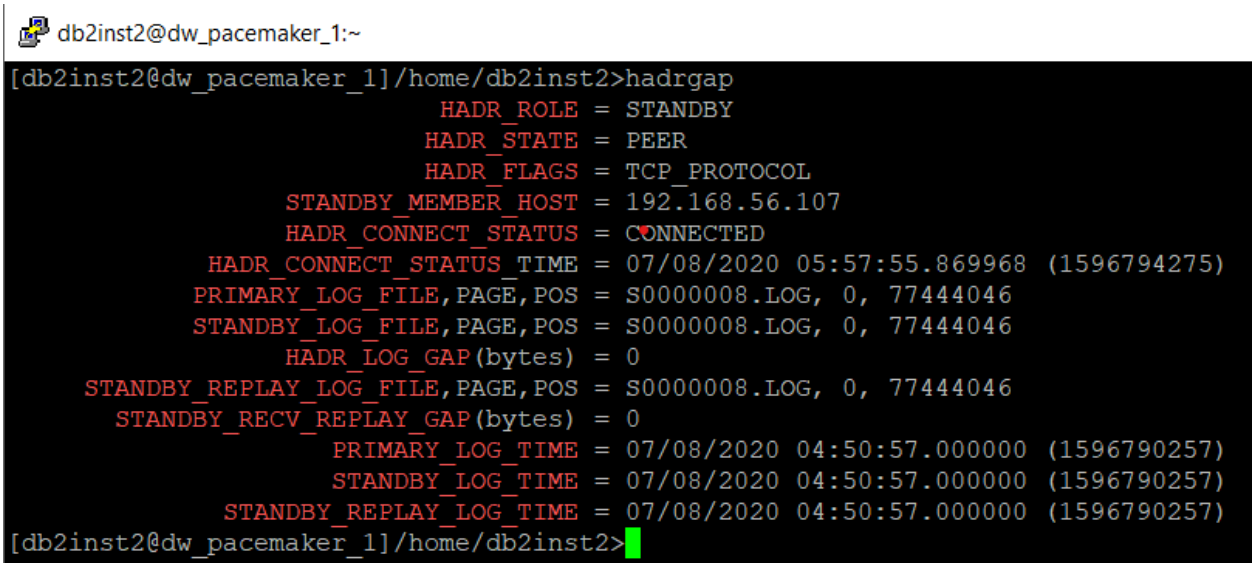
```
vboxmanage startvm dw_pacemaker_1 --type headless
  Waiting for VM "dw_pacemaker_1" to power on...
  VM "dw_pacemaker_1" has been successfully started.
```

When the VM is restarted, the Pacemaker cluster status changes to "all green":

db2inst2@dw_pacemaker_1:~

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_2 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Fri Aug  7 05:59:15 2020
Last change: Fri Aug  7 05:57:55 2020 by root via crm_attribute on dw_pacemaker_1

2 nodes configured
6 resources configured

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_2 ]
     Slaves: [ dw_pacemaker_1 ]

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

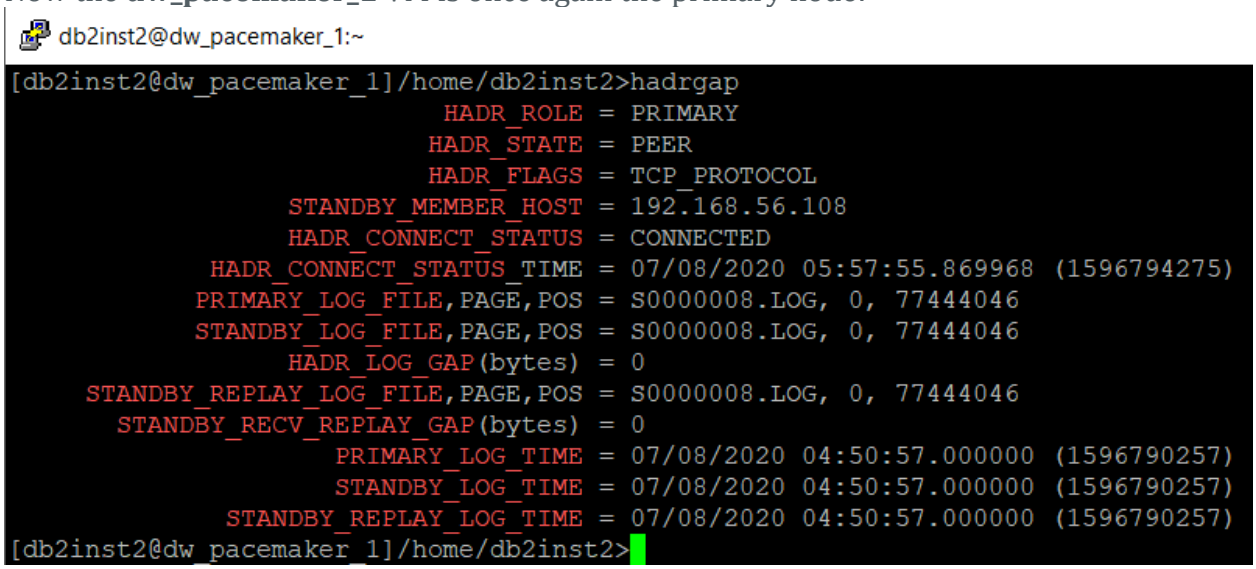
Checking the HADR status, we can see that **dw_pacemaker_1** has now assumed the standby role:



So, in order to bring the cluster back to its original state, we must execute a HADR takeover by hand:

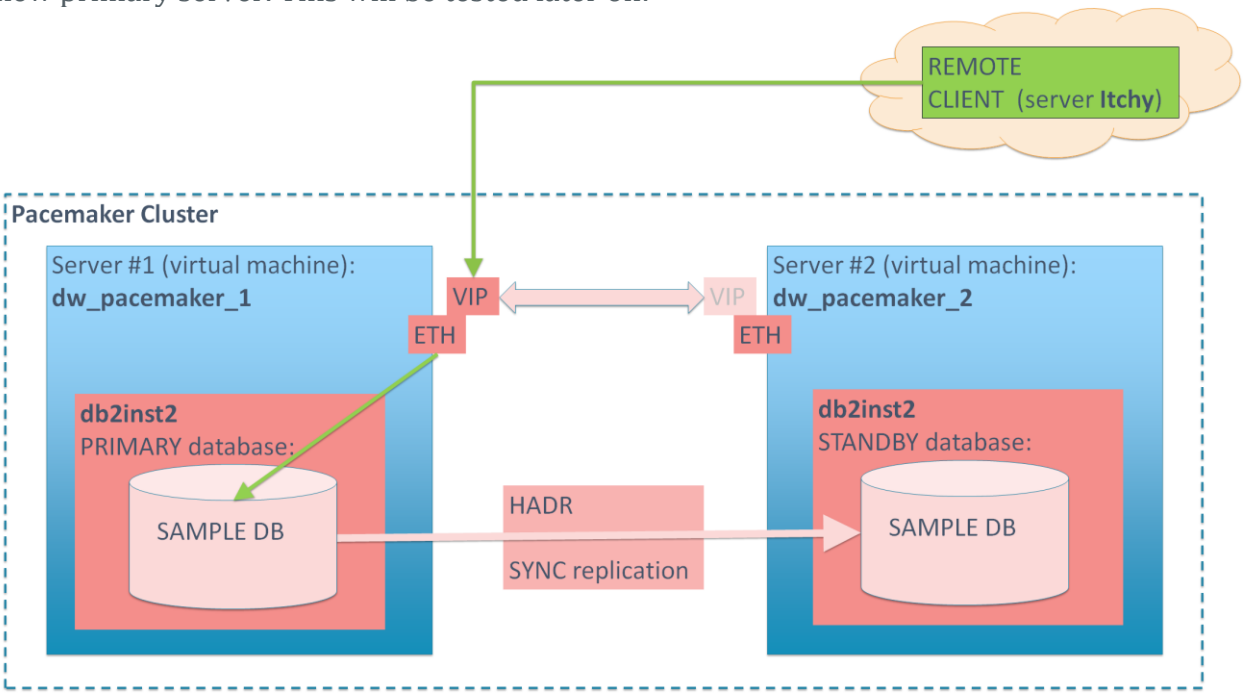
```
[db2inst2@dw_pacemaker_1]~>date; db2 takeover hadr on db SAMPLE; date
Fri  7 Aug 06:02:24 EDT 2020
DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.
Fri  7 Aug 06:02:39 EDT 2020
```

Now the **dw_pacemaker_1** VM is once again the primary node:



Create the VIP resource

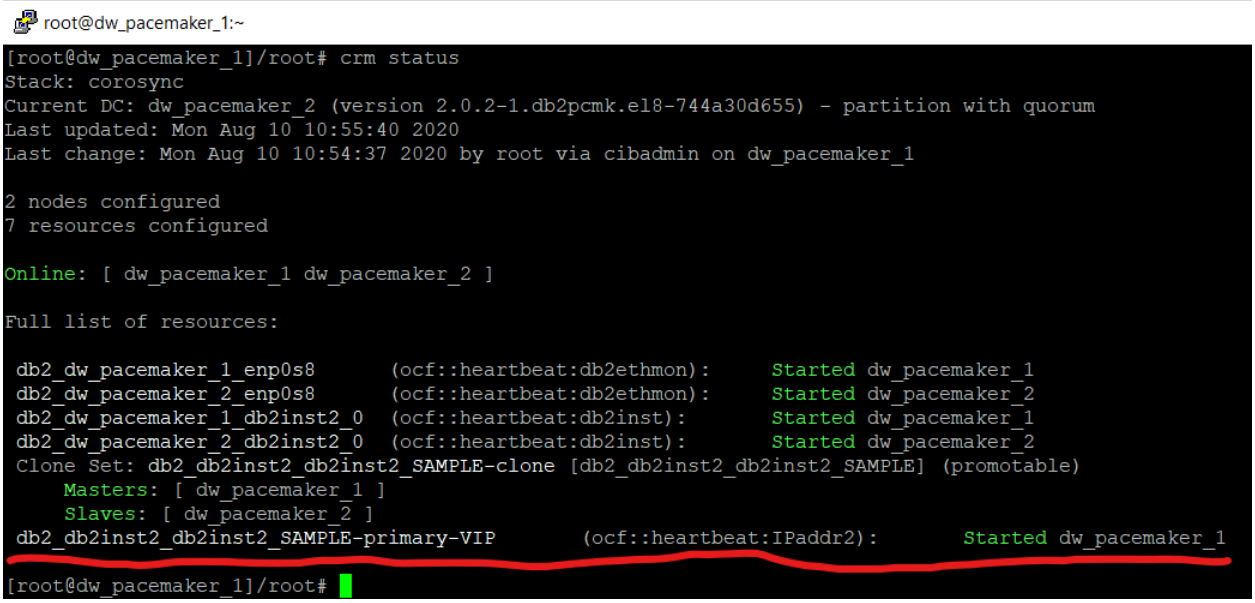
The "VIP resource" is a single TCP/IP address that will be shared between the nodes in the Pacemaker HADR cluster and remote clients will (should) be able to connect to the HA database using this address only, regardless of which node in the cluster is the current primary.
In the case of a failover, Pacemaker will (should) automatically transfer the VIP address to the new primary server. This will be tested later on!



Find an unused TCP/IP address (**192.168.56.249** in this case) and configure it as the VIP resource for the Pacemaker cluster:

```
[root@dw_pacemaker_1]/root# /home/db2inst2/sqllib/adm/db2cm -create
-primaryVIP 192.168.56.249 -db SAMPLE -instance db2inst2
Primary VIP resource created successfully.
```

Verify the VIP address has been added to the Pacemaker cluster:



Testing Remote Client Connection

Open the TCP/IP ports

To enable the remote access, first enable the Db2 TCP/IP port(s) for external access:
dw_pacemaker_1:

```
[root@dw_pacemaker_1 instance]# firewall-cmd --zone=public --
permanent --add-port 51000/tcp
  success
[root@dw_pacemaker_1 instance]# firewall-cmd --reload
  success
```

dw_pacemaker_2:

```
[root@dw_pacemaker_2 ~]# firewall-cmd --zone=public --permanent --
add-port 51000/tcp
success
[root@dw_pacemaker_2 ~]# firewall-cmd --reload
Success
```

Catalogue the remote database

Log on to Itchy (this is our test server that will assume the role of a remote client), under the account **db2dwtst** and catalogue the remote SAMPLE database that's running on **dw_pacemaker_1** and **dw_pacemaker_2**.
This will be done using the previously configured **VIP resource**, i.e. TCP/IP address **192.168.56.249**
Remote node:

```
[db2dwtst@itchy:~]$ db2 catalog tcpip node DWPM1 remote
192.168.56.249 server 51000
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
```

Remote database:

```
[db2dwtst@itchy:~]$ db2 catalog database SAMPLE as DWPMSAMP at node
DWPM1
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
```

Connect to the remote database

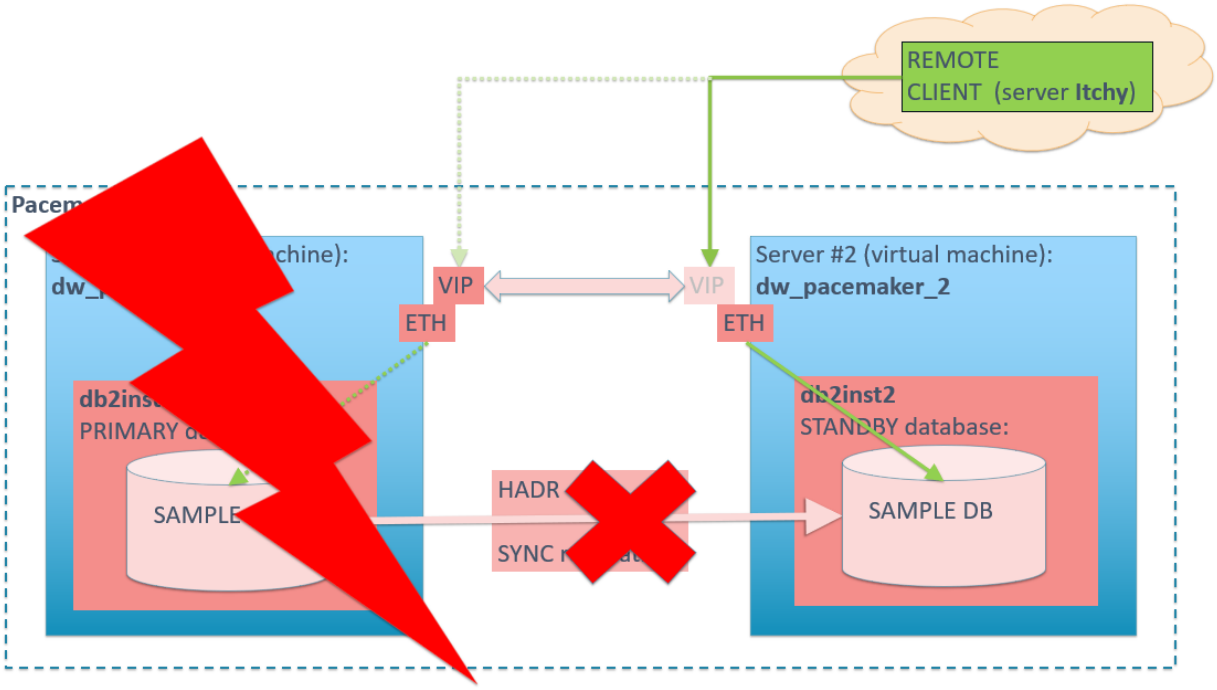
When the TCP/IP ports are open and the database in the local catalogue, verify that the remote client can connect to the database:

```
[db2dwtst@itchy:~]$ db2 connect to dwpmsamp user db2inst2
db2Enter current password for db2inst2:

   Database Connection Information

 Database server        = Db2/LINUXX8664 11.5.4.0
 SQL authorization ID   = DB2INST2
 Local database alias   = DWPMSAMP
```

All good!

Test the Client Connection in a Failover (with VIP, no ACR)

In this test, the remote client (**db2dwtst@itchy**) will connect to the **SAMPLE** database (currently running on **dw_pacemaker_1** as the primary) and start a batch job which will insert one row into the **TEST** table every ~10 seconds.
At some point the primary node **dw_pacemaker_1** will be brought down in order to find out what will happen with the running batch job (the Pacemaker will automatically failover the database **SAMPLE** to the other node in the cluster: **dw_pacemaker**_2):



On **dw_pacemaker**_1, check the Pacemaker cluster status:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_2 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Tue Aug 11 04:59:18 2020
Last change: Mon Aug 10 10:54:37 2020 by root via cibadmin on dw_pacemaker_1

2 nodes configured
7 resources configured

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_1 ]
     Slaves: [ dw_pacemaker_2 ]
 db2_db2inst2_db2inst2_SAMPLE-primary-VIP       (ocf::heartbeat:IPaddr2):       Started dw_pacemaker_1

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

From the remote client, connect to the database and create the test table:

```
[db2dwtst@itchy:~]$ db2 connect to dwpmsamp user db2inst2
db2Enter current password for db2inst2:

   Database Connection Information

 Database server        = Db2/LINUXX8664 11.5.4.0
 SQL authorization ID   = DB2INST2
 Local database alias   = DWPMSAMP

[db2dwtst@itchy:~]$ db2 "create table DB2INST2.TEST (ID integer not
null, TS timestamp not null with default current timestamp)"
DB20000I  The SQL command completed successfully.
```

Check the remote client connection is visible on **dw_pacemaker_1**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 list applications
Auth Id  Application    Appl.      Application Id                 DB       # of
         Name           Handle                                    Name     Agents
-------- -------------- ---------- ------------------------------ -------- -----
DB2INST2 db2bp          7054       192.168.56.1.54490.200811085242 SAMPLE   1
```

Start the batch job on the remote client:

```
[db2dwtst@itchy:~]$ typeset I=1;
>     date "+%Y-%m-%d_%T Inserting ID=$I" | tee -a PM_Test_01.LOG;
>     db2 -v "insert into db2inst2.test values ($I, current
timestamp)" | tee -a PM_Test_01.LOG;
>     sleep 10;
>     I=$((I+1));
> done

2020-08-11_10:01:42 Inserting ID=1
insert into db2inst2.test values (1, current timestamp)
DB20000I  The SQL command completed successfully.

2020-08-11_10:01:52 Inserting ID=2
insert into db2inst2.test values (2, current timestamp)
DB20000I  The SQL command completed successfully.

...
```

Shutdown **dw_pacemaker_1**:

```
[root@dw_pacemaker_1]/root# date; shutdown now
Tue 11 Aug 10:02:28 BST 2020
Connection to dw_pacemaker_1 closed by remote host.
Connection to dw_pacemaker_1 closed.
```

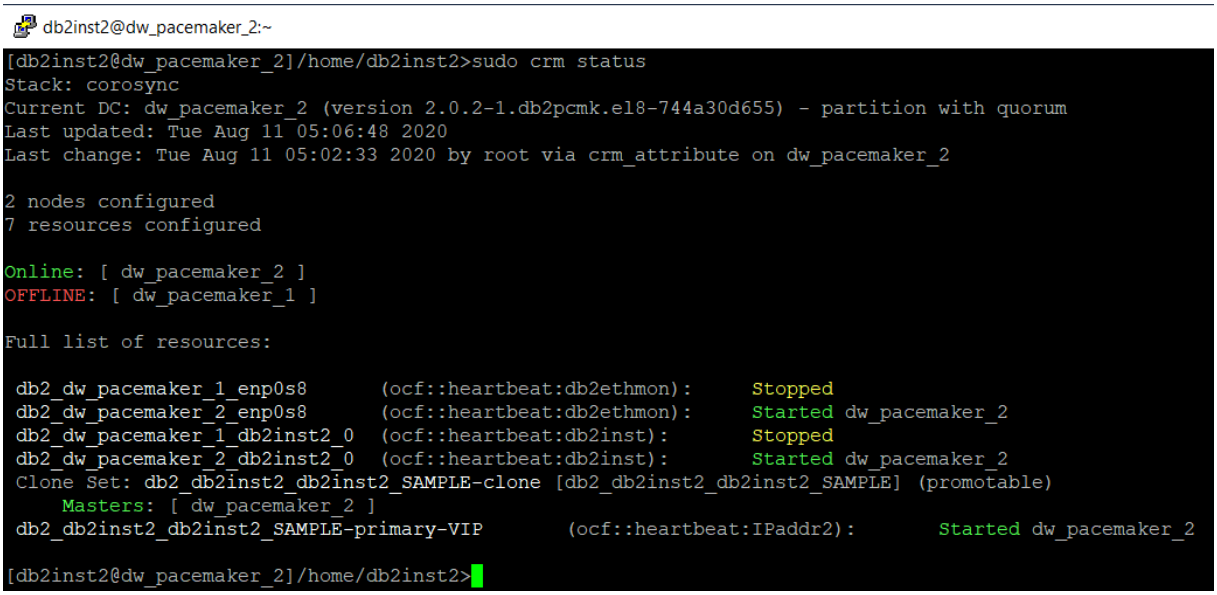
Check what happened with the batch job on the remote client (**db2dwtst@itchy**):

```
...

2020-08-11_10:02:22 Inserting ID=5
insert into db2inst2.test values (5, current timestamp)
DB20000I  The SQL command completed successfully.
```

```
2020-08-11_10:02:32 Inserting ID=6
insert into db2inst2.test values (6, current timestamp)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL30081N  A communication error has been detected. Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".  Location
where the error was detected: "192.168.56.249".  Communication function
detecting the error: "send".  Protocol specific error code(s): "32", "*",
"0".
SQLSTATE=08001

2020-08-11_10:02:42 Inserting ID=7
insert into db2inst2.test values (7, current timestamp)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL1024N  A database connection does not exist.  SQLSTATE=08003

2020-08-11_10:02:52 Inserting ID=8
insert into db2inst2.test values (8, current timestamp)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL1024N  A database connection does not exist.  SQLSTATE=08003
```

The batch job failed on insert with ID=6, when the shutdown happened, and all subsequent inserts failed because the client lost the connection to the database!

Looking at the Pacemaker cluster status on the other node, **dw_pacemaker_2**, we can see that it has assumed the role of "Master" (i.e. it is the current HADR primary node), while **dw_pacemaker_1** is **OFFLINE** (as expected):



Try reconnecting the remote client to the database:

```
[db2dwtst@itchy:~]$ db2 connect to DWPMSAMP user db2inst2
Enter current password for db2inst2:

   Database Connection Information

 Database server        = Db2/LINUXX8664 11.5.4.0
```

```
 SQL authorization ID   = DB2INST2
 Local database alias   = DWPMSAMP
```

That worked fine, check the data in the TEST table:

```
[db2dwtst@itchy:~]$ db2 "select * from db2inst2.test"
ID          TS
----------- --------------------------
          1 2020-08-11-05.01.42.674554
          2 2020-08-11-05.01.52.716484
          3 2020-08-11-05.02.02.755486
          4 2020-08-11-05.02.12.793455
          5 2020-08-11-05.02.22.830999
  5 record(s) selected.
```

Check the remote client connections on **dw_pacemaker_2**:

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>db2 list applications
Auth Id  Application    Appl.      Application Id                 DB       # of
         Name           Handle                                    Name     Agents
-------- -------------- ---------- ------------------------------ -------- -----
DB2INST2 db2bp          3168       192.168.56.1.54578.200811091119 SAMPLE   1
```

The connection to the new primary server (**dw_pacemaker**_2) is open so the remote client is now in a position to resume the interrupted batch job!

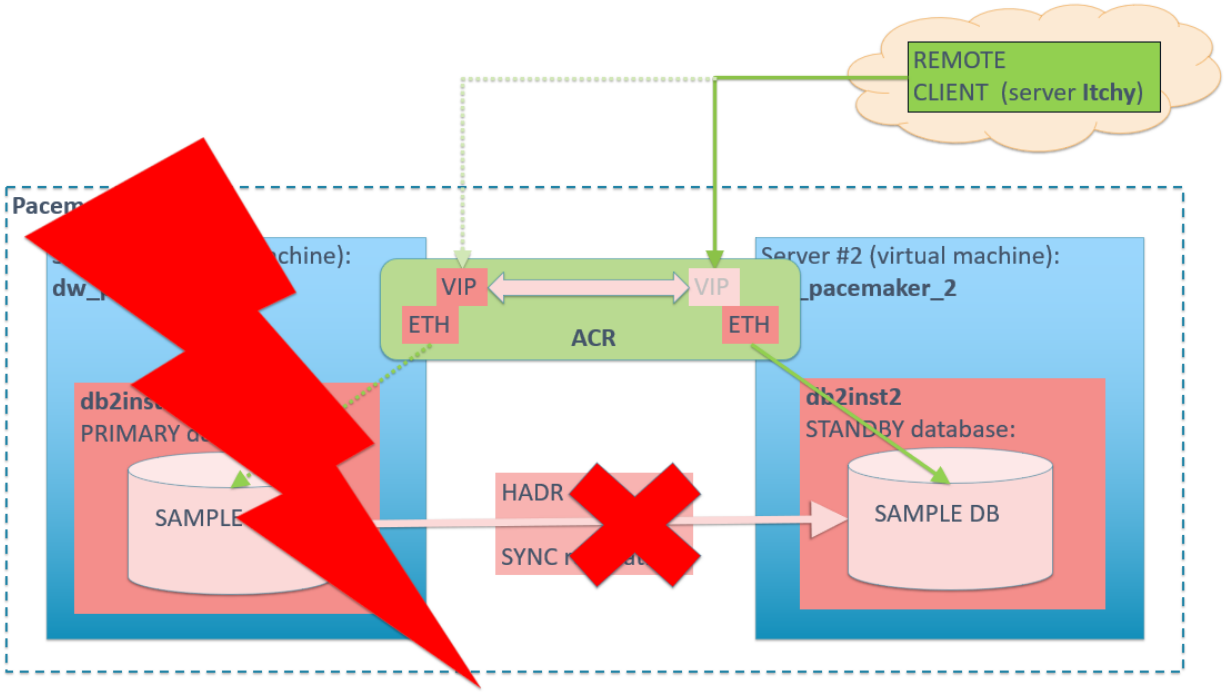
Test Summary

This test proved that the HA database **SAMPLE** survived the primary node failure by failing over to the standby node, but the remote client connections were lost and remote clients had to explicitly reconnect to the database in order to resume the processing.

In the next test we will additionally configure ACR (Automatic Client Reroute) and repeat this test to find out what will happen with the remote connections in that environment.

Test the Client Connection in a Failover (with VIP and ACR)

In this test first the ACR (Automatic Client Reroute) configuration is updated on all nodes in the Pacemaker HADR cluster first (using the previously configured **VIP resource**).
Then the remote client will connect to the **SAMPLE** database (currently running on **dw_pacemaker_1** as the primary) and start a batch job which will insert one row into the **TEST** table every ~10 seconds.
At some point the primary node **dw_pacemaker_1** will be brought down in order to find out what will happen with the running batch job (the Pacemaker will automatically failover the database **SAMPLE** to the other node in the cluster: **dw_pacemaker_2**).
The expected result is that the remote client will automatically reconnect to the **SAMPLE** database (now running on **dw_pacemaker_2**) and resume its active batch job (any in-flight transactions at the time of the failover will be lost though):



Update ACR configuration

On **dw_pacemaker_1**, add the VIP address as the ACR:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 update alternate server
for database SAMPLE using hostname 192.168.56.249 port 51000
DB20000I  The UPDATE ALTERNATE SERVER FOR DATABASE command completed
successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
```

On **dw_pacemaker_2**, add the VIP address as the ACR:

```
[db2inst2@dw_pacemaker_2]/home/db2inst2>db2 update alternate server
for database SAMPLE using hostname 192.168.56.249 port 51000
DB20000I  The UPDATE ALTERNATE SERVER FOR DATABASE command completed
successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
```

Check the ACR update is visible on the remote client:

```
[db2dwtst@itchy:~]$ db2 connect to dwpmsamp user db2inst2
Enter current password for db2inst2:

   Database Connection Information
 Database server        = Db2/LINUXX8664 11.5.4.0
 SQL authorization ID   = DB2INST2
 Local database alias   = DWPMSAMP

[db2dwtst@itchy:~]$ db2 list db directory
...
Database 2 entry:
 Database alias                       = DWPMSAMP
 Database name                        = SAMPLE
 Node name                            = DWPM1
 Database release level               = 14.00
 Directory entry type                 = Remote
 Alternate server hostname            = 192.168.56.249
 Alternate server port number         = 51000
```

Looking good!

Resume the test

With that done, the test can start:
On **dw_pacemaker_1**, check the Pacemaker cluster status:

```
db2inst2@dw_pacemaker_1:~
[db2inst2@dw_pacemaker_1]/home/db2inst2>sudo crm status
Stack: corosync
Current DC: dw_pacemaker_2 (version 2.0.2-1.db2pcmk.el8-744a30d655) - partition with quorum
Last updated: Tue Aug 11 05:43:32 2020
Last change: Tue Aug 11 05:29:16 2020 by root via crm_attribute on dw_pacemaker_1

2 nodes configured
7 resources configured

Online: [ dw_pacemaker_1 dw_pacemaker_2 ]

Full list of resources:

 db2_dw_pacemaker_1_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_1
 db2_dw_pacemaker_2_enp0s8      (ocf::heartbeat:db2ethmon):     Started dw_pacemaker_2
 db2_dw_pacemaker_1_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_1
 db2_dw_pacemaker_2_db2inst2_0  (ocf::heartbeat:db2inst):       Started dw_pacemaker_2
 Clone Set: db2_db2inst2_db2inst2_SAMPLE-clone [db2_db2inst2_db2inst2_SAMPLE] (promotable)
     Masters: [ dw_pacemaker_1 ]
     Slaves: [ dw_pacemaker_2 ]
 db2_db2inst2_db2inst2_SAMPLE-primary-VIP       (ocf::heartbeat:IPaddr2):       Started dw_pacemaker_1

[db2inst2@dw_pacemaker_1]/home/db2inst2>
```

From the remote client, connect to the database:

```
[db2dwtst@itchy:~]$ db2 connect to dwpmsamp user db2inst2
db2Enter current password for db2inst2:
   Database Connection Information
 Database server        = Db2/LINUXX8664 11.5.4.0
 SQL authorization ID   = DB2INST2
 Local database alias   = DWPMSAMP
```

Clean up the TEST table:

```
[db2dwtst@itchy:~]$ db2 "delete from db2inst2.test"
DB20000I  The SQL command completed successfully.
```

Check the remote client connection is visible on **dw_pacemaker_1**:

```
[db2inst2@dw_pacemaker_1]/home/db2inst2>db2 list applications
Auth Id  Application    Appl.      Application Id                 DB       # of
         Name           Handle                                    Name     Agents
-------- -------------- ---------- ------------------------------ -------- -----
DB2INST2 db2bp          7054       192.168.56.1.58580.200811101112 SAMPLE   1
```

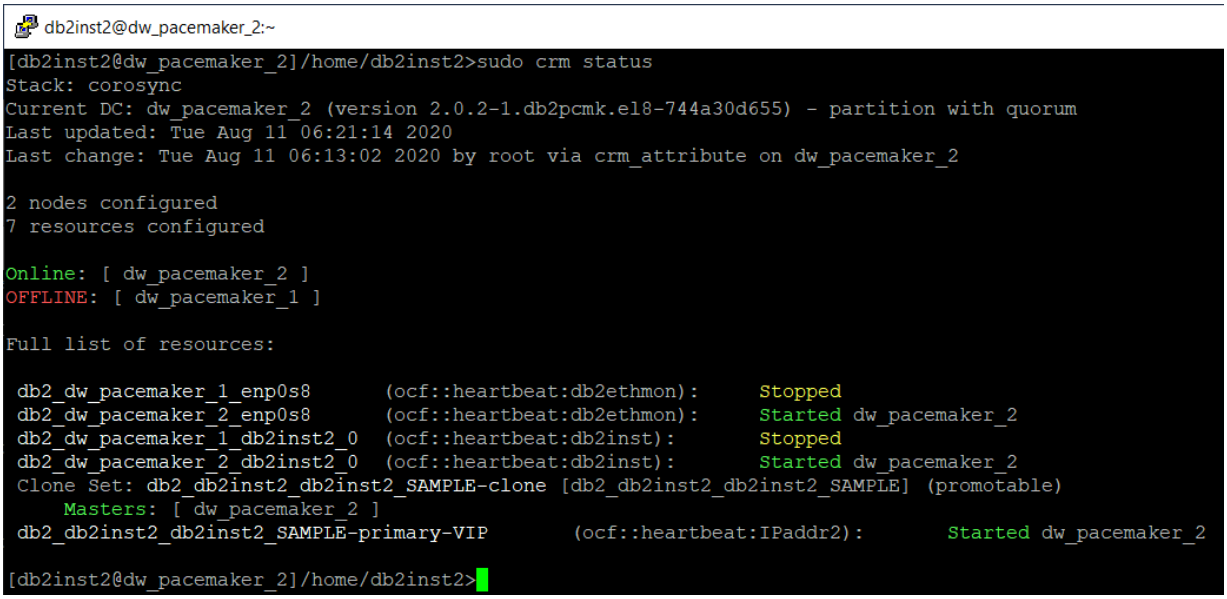
Start the batch job on the remote client:

```
[db2dwtst@itchy:~]$ typeset I=1;
>     date "+%Y-%m-%d_%T Inserting ID=$I" | tee -a PM_Test_01.LOG;
>     db2 -v "insert into db2inst2.test values ($I, current
timestamp)" | tee -a PM_Test_01.LOG;
>     sleep 10;
>     I=$((I+1));
> done

...
```

Shutdown **dw_pacemaker_1**:

```
[root@dw_pacemaker_1]/root# date; shutdown now
Tue 11 Aug 11:12:53 BST 2020
Connection to dw_pacemaker_1 closed by remote host.
Connection to dw_pacemaker_1 closed.
```

Check the Pacemaker cluster status (from **dw_pacemaker_2** which should now be the HADR primary):



dw_pacemaker_2 is indeed the new primary, with **dw_pacemaker_2 OFFLINE** as expected.
Check what happened with the batch job on the remote client (**db2dwtst@itchy**):

```
2020-08-11_11:12:39 Inserting ID=1
insert into db2inst2.test values (1, current timestamp)
DB20000I  The SQL command completed successfully.

2020-08-11_11:12:49 Inserting ID=2
insert into db2inst2.test values (2, current timestamp)
DB20000I  The SQL command completed successfully.

2020-08-11_11:12:59 Inserting ID=3
insert into db2inst2.test values (3, current timestamp)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL30108N  A connection failed in an automatic client reroute environment.
The transaction was rolled back. Host name or IP address: "192.168.56.249".
Service name or port number: "51000". Reason code: "1". Connection failure
code: "2". Underlying error: "*".  SQLSTATE=08506

2020-08-11_11:14:18 Inserting ID=4
insert into db2inst2.test values (4, current timestamp)
DB20000I  The SQL command completed successfully.

2020-08-11_11:14:28 Inserting ID=5
insert into db2inst2.test values (5, current timestamp)
DB20000I  The SQL command completed successfully.

2020-08-11_11:14:38 Inserting ID=6
insert into db2inst2.test values (6, current timestamp)
DB20000I  The SQL command completed successfully.
```

[batch job is interrupted at this point!]

The result is that the batch job continued its processing (inserts into the TEST table) as soon as the failover was done, with one (in-flight) transaction rolled back due to the failed connection (temporary condition during a failover).

Test Summary

This test showed that the HA database **SAMPLE** survived the primary node failure by failing over to the standby node.
And even more, by configuring the ACR (Automatic Client Reroute) the active remote client connections were able to automatically resume their processing shortly after the failover, without any user intervention!
However, any in-flight transactions that were active at the time of the failover were lost.

Grand Conclusion

From all experiences and activities carried out with the Pacemaker so far, the following can be concluded:

1. Pacemaker installation is a simple matter
2. Pacemaker adds the High Availability functionality to Db2 HADR clusters, and it does that well
3. Coupled with ACR, Pacemaker enables uninterrupted client connections to Db2 HADR clusters

What remains to be tested is the addition of a quorum device to a Pacemaker cluster and also trying to alter the configuration of active Pacemaker clusters (a big headache for the TSA/MP clusters).

About Triton

Triton Consulting are experts in Hybrid Data Management and Digital Transformation. The company's team of consultants represent some of the most highly experienced and qualified in the industry, and are able to advise on a range of Data Management solutions including Db2 for z/OS, Db2 for LUW plus data related infrastructure and modernisation services.

As well as expert consultancy in all areas of Db2, Triton Consulting also cover a wider spectrum of high-level consultancy including senior project management, technical planning, technical architecture, performance tuning and systems programming.

Triton Consulting has been providing consultancy services for over 24 years. Triton are internationally recognised for their Db2 expertise with three IBM Gold Consultants and seven IBM Champions.

Find out more about Triton Consulting: www.triton.co.uk

About the Author

Damir Wilder is a multi-talented Db2 expert and has worked with Db2 for LUW since V5.0. With significant experience as a Database Administrator and Application Developer. Damir has developed deep technical skills with database design, maintenance and performance tuning, as well as migrations both on-premise and in the cloud.

As part of our Db2 Midrange team Damir works on both Consultancy projects and Remote Db2 support for a wide range of clients.

Contact

Damir Wilder
Senior Consultant
damir.wilder@triton.co.uk