Stay Ahead With Pacemaker

Session 2187 / 2373

Lab Exercise Guide

Andreas Christian
Information Architecture Technical Specialist
achristian@de.ibm.com

Stefan Hummel
Information Architecture Technical Specialist
stefan.hummel@de.ibm.com

Think 2021

IBM

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.
Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information.

This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided. The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled,
isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers (Continued)

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.** The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

Table of Contents

1 Introduction

The combination of Pacemaker and HADR enables high availability and disaster recovery for DB2 databases. It is currently supported for Db2 HADR clusters on Linux for on-prem deployments and is installed manually on top of Db2. Pacemaker is planned to become the future cluster solution for all types of Db2 deployments including pureScale, DPF, and containerized Db2 deployments in the cloud. The Pacemaker software installation will be part of the Db2 installation in the future. One important difference to TSA is the usage of a quorum device that is supposed to run a separate server.

HADR on its own provides mainly Disaster Recovery, by maintaining one or more synchronised copies of a DB2 database. If the primary DB2 server fails, there is a database copy (a standby database) that can be used instead of the primary database. The failover process (switching from a primary to standby database) must be initiated *manually* by a database administrator.

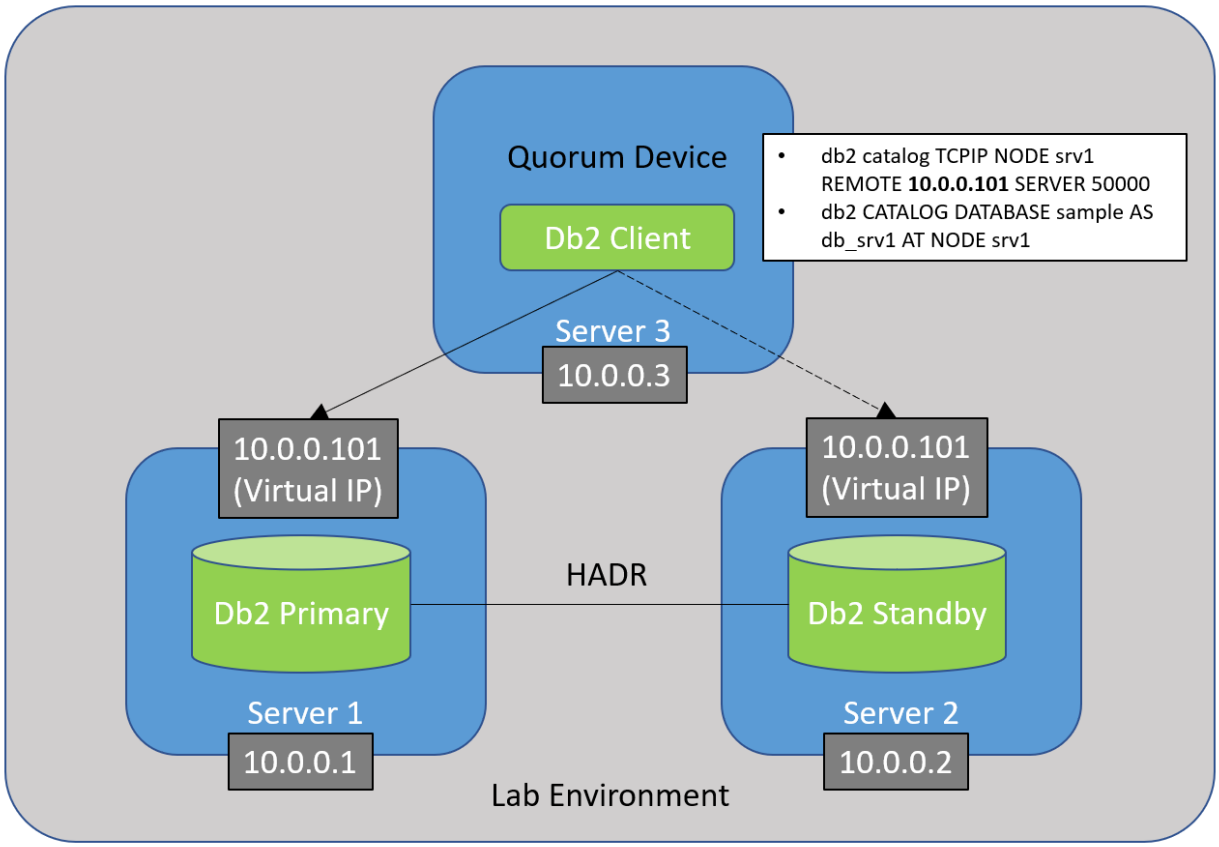
Pacemaker *automates the failover process* and helps to minimize the database downtime. It actively monitors the health of the databases and their host servers and in case of a failure automatically performs the switch to the standby server.

1.1 High Level Architecture

This tutorial takes you through the steps of configuring a Db2 HADR cluster with Pacemaker.

The lab environment consists of three servers:

- server1 hosts the Db2 HADR primary database
- server2 hosts the Db2 HADR standby database
- server3 hosts the Pacemaker quorum device, a Db2 client, and some scripts to simulate an application that generates some database workload.



To save you some time, we have already installed the Db2 software and a Db2 instance on all three servers. We also completed some other prerequisit steps (see details further down). The hands-on lab consists of the following parts:

1. You start by configuring the HADR cluster (using predefined scripts). This requires a Db2 backup on server1, to restore the backup on server2, configure the Db2 HADR parameters on both servers, and finally to start both databases in their respective roles (standby and primary).
2. Next, you will perform a planned take-over. As part of this lab section you also configure the Db2 automatic client reroute (ACR) feature.
3. Then you will setup the Pacemaker cluster configuration.
4. In the final section, you will simulate an outage and monitor how the system peforms an automatic failover.

Note: The virtual IP (10.0.0.101) for the Pacemaker cluster will be created during the Pacemaker cluster configuration (by the db2cm command).

1.2 User IDs, Passwords, Directory Structure, Tools

User IDs:

- root / 4711think_root
- db2inst1 / 4711think_db2inst1

To switch between the servers the following aliases are configured on all three servers:
alias **s1**='ssh -X db2inst1@server1'
alias **s2**='ssh -X db2inst1@server2'
alias **s3**='ssh -X db2inst1@server3'

Location of the Pacemaker software:
/tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64

The tar ball with the Db2 installation files was extracted to the following directory:
/root/db2Image

Installation directory of the Db2 software:
/opt/ibm/db2/V11.5

Location of the Db2 diagnostic file:
/home/db2inst1/sqllib/db2dump/DIAG0000/db2diag.log

The following lab directory will be created during the preparational steps (see below):
/home/db2inst1/Lab_HADR_Pacemaker

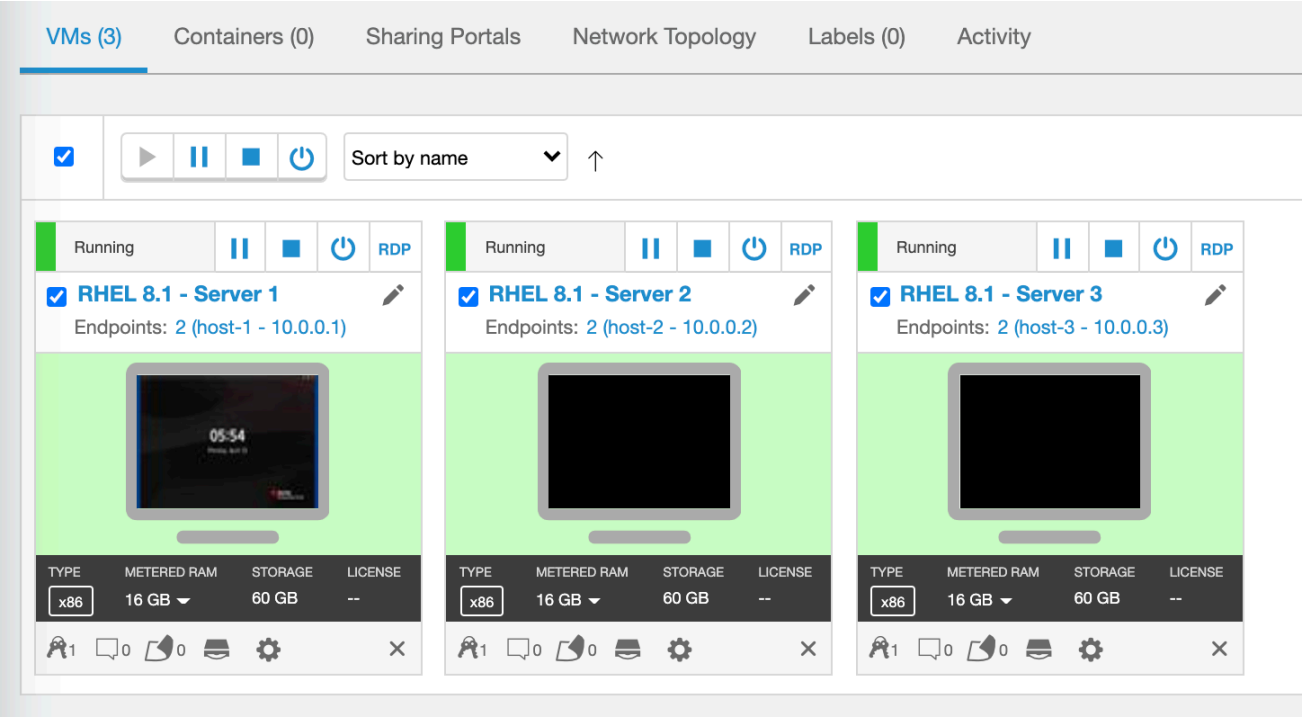
Directory containing the Db2 cluster manager (db2cm):
/home/db2inst1/sqllib/adm

The following tools are available on the servers:

- nmon: Monitor CPU, memory, network and disk as well as processes.
- db2pd: Monitor and troubleshoot db2 databases.
- db2top: Db2 interactive snapshot monitor to monitor dynamic SQL statements, sessions, HADR and much more.

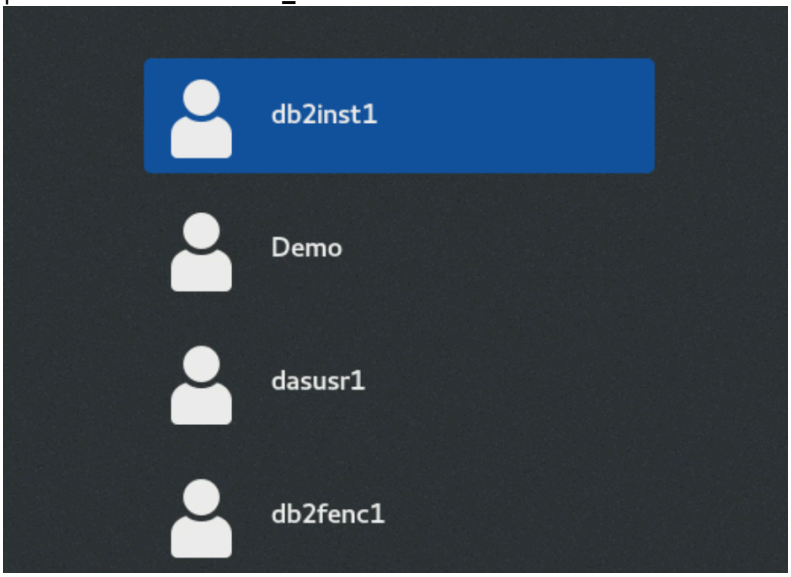
2 Getting Started

To access the servers described in the previous chapter open the cloud environment with the URL provided in the invitation. You will see three virtual machines.

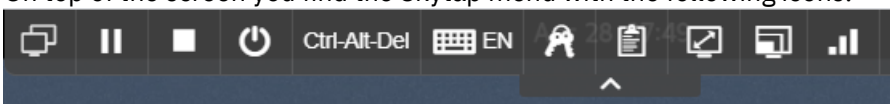


The green background indicates that a virtual machine is running. If the background is grey then start the virtual machine with the “play” symbol.

Access **server3** by clicking on the very left symbol “*RHEL 8.1 – Server3*” and login as user *db2inst1*. The password is **4711think_db2inst1**.



On top of the screen you find the Skytap menu with the following icons:

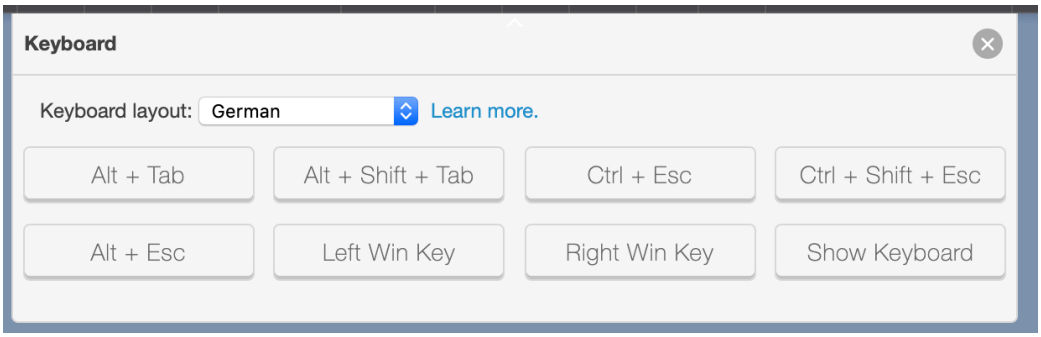


Now let's customize the screen and keyboard.

Click the following icon to expand the desktop:

This expands the screen of the virtual machine to the size of your browser window.
Next, click the keyboard icon:

You get this pop-up window:

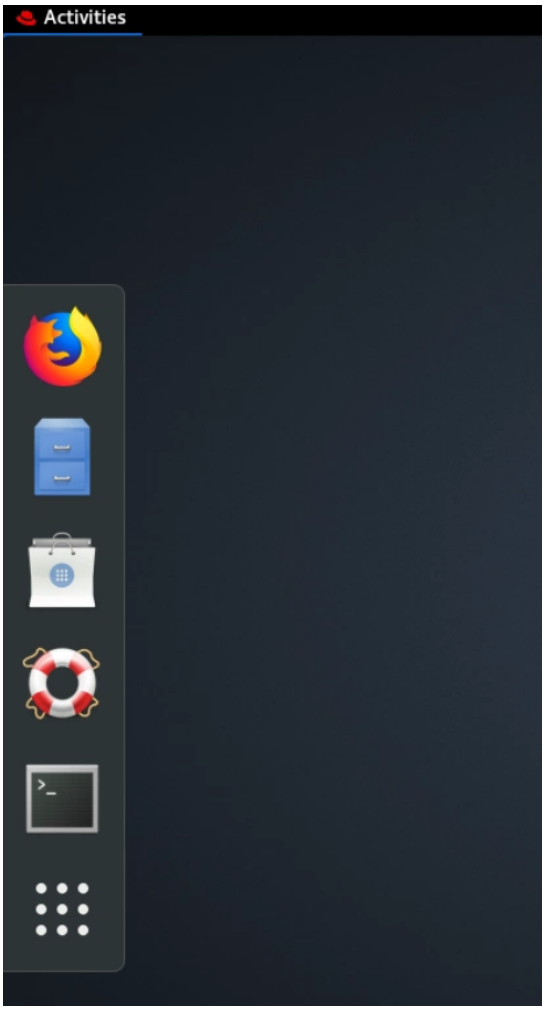


At "Keyboard layout" select the type of your keyboard and close this window.

If you are using a keyboard layout other than English-US, then you have to set the keyboard used at the OS level as well. For this click the "en" icon in the upper right corner of your screen:

You can use the following icon in the Skytap toolbar to copy text between your local machine and the Skytap virtual machines.In the Red Hat Linux desktop open a terminal as follows:

- Click **Activities** on the top left of the desktop
- Click the **Terminal** icon



In the terminal window you can use auto text completion for files and directory names by typing the first part of a name and then pressing the tabulator key. For example, type in the following string in the window (do not press the enter key):

```
[db2inst1@server3 ~]$ cd sql
```

Now, press the tabulator key to use auto completion. The directory name will be automatically completed as follows:

```
[db2inst1@server3 ~]$ cd sqllib/
```

2.1 Get the Latest Lab Description and Scripts

Open a new terminal window and switch to **server1**.

```
[db2inst1@server3 ~]$ s1
```

In this terminal window change to the home directory of the db2 instance owner:

```
[db2inst1@server1 ~]$ cd
```

Get the latest copy of this directory by typing the command below.
Please note: Please make sure that no special characters or spaces occur when copying the URL.

Check if directory Lab_HADR_Pacemaker already exists and rename it:

```
[db2inst1@server1 Lab]$ mv Lab_HADR_Pacemaker Lab_HADR_Pacemaker.old
[db2inst1@server1 Lab]$ git clone https://github.com/stefanhummel/Lab_HADR_Pacemaker
Cloning into 'Lab_HADR_Pacemaker'...

[db2inst1@server1 Lab]$
```

Make the scripts executable:

```
[db2inst1@server1 ]$ cd Lab_HADR_Pacemaker
[db2inst1@server1 ]$ chmod +x *.sh
```

Repeat the above steps on **server2** and **server3**.

All the scripts you need in the lab sections are now located in the following directory on each server:
/home/db2inst1/Lab_HADR_Pacemaker

3 Lab Exercises

The following list provides an overview of the prerequisites that are in place on each server. Before you start with the labs, please take a quick review of the already completed prerequisite steps below:

- The Db2 DB2 v11.5.5.0 was installed.
- The following user IDs (and corresponding group IDs) were created manually after the Db2 installation: db2inst1, db2fenc1, dasusr1
- A Db2 instance was already created (/opt/ibm/db2/V11.5/instance/db2icrt -a SERVER -u db2fenc1 db2inst1)
- The Pacemaker cluster software has been downloaded to the /tmp directory.
- The Db2 software has been downloaded and unpacked to the /root/db2Image directory.
- The `/etc/hosts` file is setup as follows:

```
[db2inst1@server1 scripts]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

# DB2 Server
10.0.0.1    server1.compute.internal                    server1
10.0.0.2    server2.compute.internal                    server2
10.0.0.3    server3.compute.internal                    server3
```

- All hosts have TCP/IP connectivity between their Ethernet network interfaces
- Both root and instance user ID can use ssh between the hosts, using both long and short host names.
- The Db2 fault monitor was disabled (db2greg -updinstrec instancename=db2inst1 startatboot=0)

3.1 HADR Setup

Perform the following tasks to create the *sample* database and to configure an HADR cluster for the database.

server1:

- Change to the script directory:

```
[db2inst1@server1 ]$ su - db2inst1
[db2inst1@server1 ]$ cd /home/db2inst1/Lab_HADR_Pacemaker
```

- Run the following script to create the sample database:

```
[db2inst1@server1 ]$ ./01_create_db.sh
```

- Run the following script to enable log archiving to directory /home/db2inst1/sample_arch1/ and to setup the HADR parameters of the sample database:

```
[db2inst1@server1 ]$ ./02_setup_hadr_primary.sh
```

- Run the following script to perform a database backup to directory /home/db2inst1/sample_backup:

```
[db2inst1@server1 ]$ ./03_create_copy_backup.sh
```

server2:

- Change to the script directory:

```
[db2inst1@server2 ]$ su - db2inst1
[db2inst1@server2 ]$ cd /home/db2inst1/Lab_HADR_Pacemaker
```

- Run the following script to restore the backup of the *sample* database on server2:

```
[db2inst1@server2 ]$ ./04_restore_db.sh
```

- Run the following script to configure the HADR parameters for the restored *sample* database (HADR standby database):

```
[db2inst1@server2 ]$ ./05_setup_hadr_standby.sh
```

- Run the following script to start the *sample* database on server2 as HADR standby database:

```
[db2inst1@server2 ]$ ./06_start_hadr_standby.sh
```

server1:

- Change to the script directory:

```
[db2inst1@server1 ]$ su - db2inst1
[db2inst1@server1 ]$ cd /home/db2inst1/Lab_HADR_Pacemaker
```

- Run the following script to start the *sample* database on server1 as HADR primary database:

```
[db2inst1@server1 ]$ ./07_start_hadr_primary.sh
```

- Run the following script to verify that HADR is running and that both servers are in PEER state:

```
[db2inst1@server1 ]$ ./08_check_hadr.sh

Database Member 0 -- Database SAMPLE -- Active -- Up 3 days 21:50:48 -- Date 2021-04-20-
04.06.42.048173

                           HADR_ROLE = PRIMARY
                       REPLAY_TYPE = PHYSICAL
                     HADR_SYNCMODE = NEARSYNC
                        STANDBY_ID = 1
                     LOG_STREAM_ID = 0
                        HADR_STATE = PEER
                        HADR_FLAGS = TCP_PROTOCOL
               PRIMARY_MEMBER_HOST = server1
                  PRIMARY_INSTANCE = db2inst1
                    PRIMARY_MEMBER = 0
               STANDBY_MEMBER_HOST = server2
                  STANDBY_INSTANCE = db2inst1
                    STANDBY_MEMBER = 0
               HADR_CONNECT_STATUS = CONNECTED
                                     ...
```

server1:

- Run the following commands to connect to the *sample* database and to retrieve the table names:

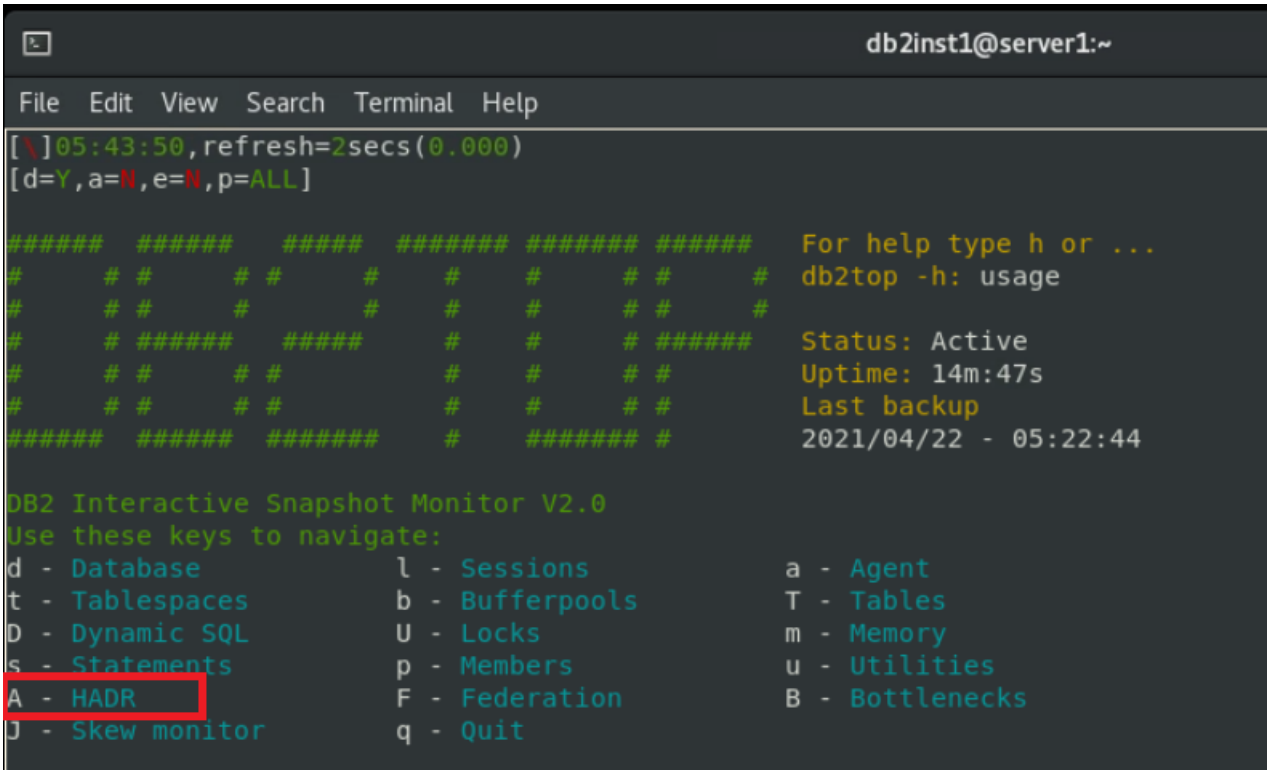
```
[db2inst1@server1 ]$ db2 connect to sample
[db2inst1@server1 ]$ db2 list tables
```

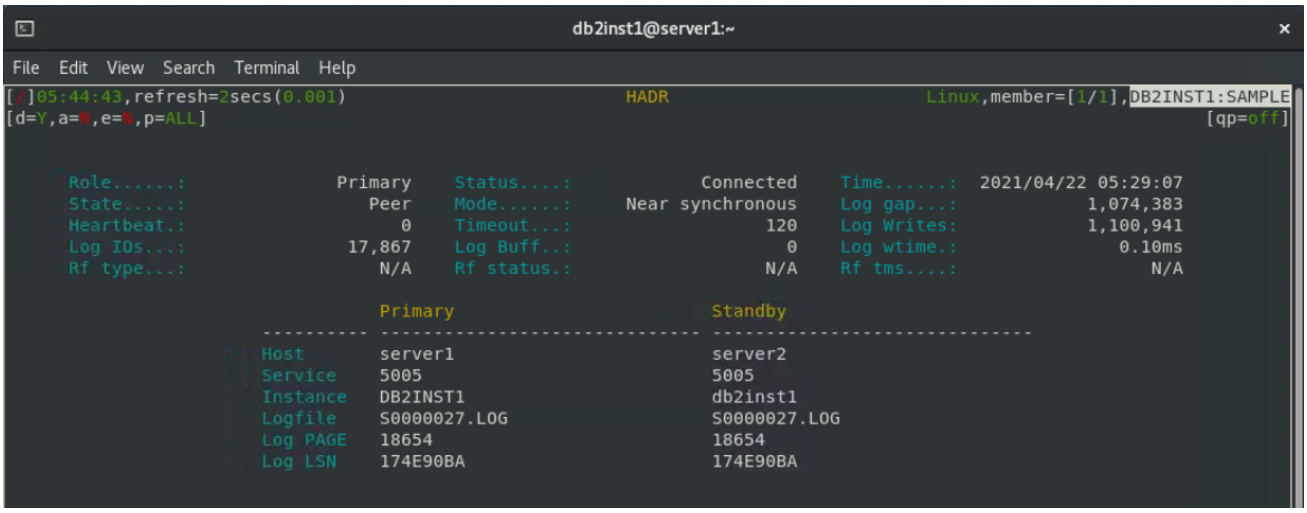
- Simulate a database workload by executing the following script:

```
[db2inst1@server1 ]$ db2 -td@ -vf gen_workload.sql
```

- Open another terminal window and use db2top to monitor the database HADR:

```
[db2inst1@server1 ]$ db2top -d sample
```





- Alternatively, you can use dsmtop to monitor the database:

```
[db2inst1@server1 ]$ dsmtop -d sample -j 4 -u db2inst1 -p 4711think_db2inst1
```

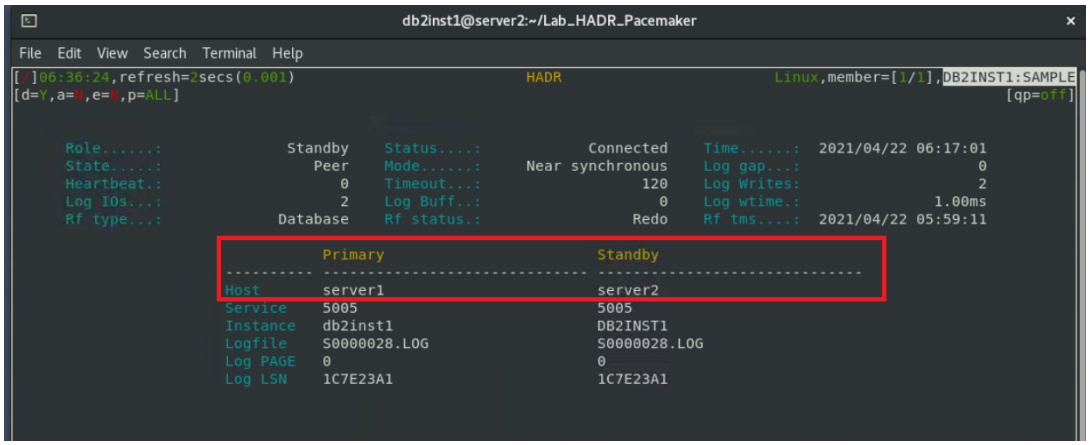
3.2 Planned take-over

- Check current STANDBY_MEMBER_HOST by running the following command on **server1**.

```
[db2inst1@server1 ]$ db2pd -hadr -db sample
```

- Open another terminal window, run db2top and type **A** to show the HADR monitoring view:

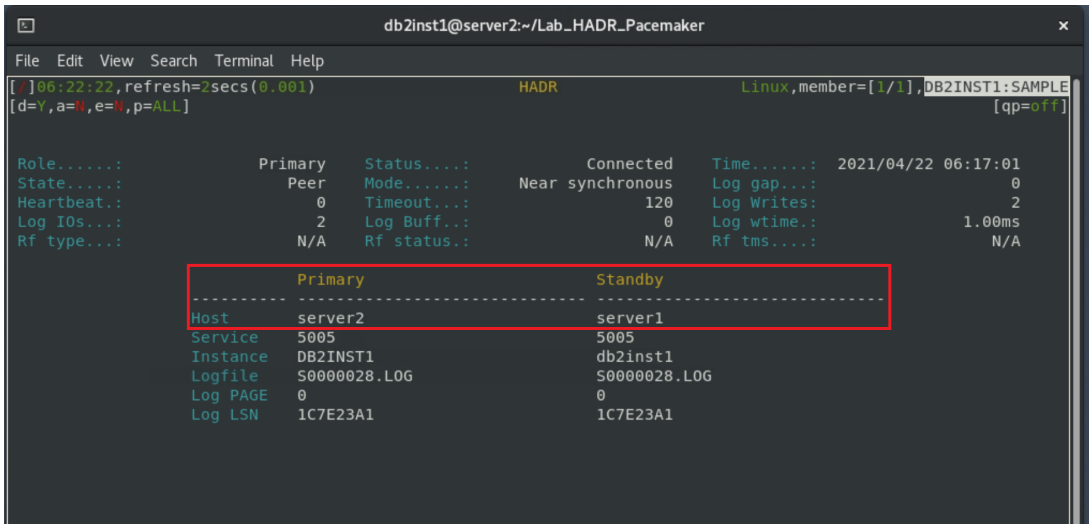
```
[db2inst1@server1 ]$ db2top -d sample
```



Perform takeover on the current **STANDBY_MEMBER_HOST** and check in db2top how the standby and primary roles are changing:

```
[db2inst1@server2 ]$ db2 takeover hadr on database sample
```

- Check in db2top that the HADR roles were switched successfully:



- Finally, perform another take over on **server1** to make sure server1 becomes the HADR primary again and verify the new roles in db2top:

```
[db2inst1@server1 ]$ db2 takeover hadr on database sample
```

3.3 Installing the Pacemaker cluster software

For the installation the Pacemaker software is downloaded already and extracted in the directory /tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/.
You have to install pacemaker on all cluster server (**server1** and **server2**), but not on the quorum server.

3.3.1 Pre-setup checklist

For the pacemaker installation there are some prerequitites, which are done already in the lab environment:

- Instance user ID and group ID are setup
- `/etc/hosts` are setup with both hosts using long and short host names.
- Both hosts have TCP/IP connectivity between their Ethernet network interfaces
- Both root and instance user ID (db2inst1) can use ssh between the two hosts, using both long and short host names.
- The Pacemaker cluster software has been downloaded to both hosts.

3.3.2 Installation

As root on server1, extract the tar file in the `/tmp` folder.

```
[root@server1 ]$ cd /tmp
[root@server1 ]$ tar -zxf Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64.tar.gz
```

As we are using RHEL 8.1, install the epel-release, followed by the RPMs in the untarred Pacemaker directory:

```
[root@server1 ]$ cd /tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64
[root@server1 ]$ cd RPMS
[root@server1 ]$ dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
[root@server1 ]$ dnf install */*.rpm
```

Verify that the following packages are installed

- *pacemaker*
  Pacemaker ist he cluster management
- *corosync*
  Corosync is the underlying messaging layer for the pacemaker clusters
- *crmsh*
  crm is the pacemaker command line interface for configuration and management

Use the command:

```
[root@server1 ]$ rpm -q <packagename>
```

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

```
[root@server1 ]$ cp /tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/Db2/db2cm /home/db2inst1/sqllib/adm/.
[root@server1 ]$ chmod 755 /home/db2inst1/sqllib/adm/db2cm
```

Now the Pacemaker cluster utility db2cm is available. You should be able to use it from any directory since the path of db2cm is added in the $PATH parameter already. Check it with:

```
[root@server1 ]$ db2cm -help
```

Note: Also repeat the above steps of this chapter 3.3.2 on the second host **server2.**

Finally, copy the resource agent scripts (**db2hadr**, **db2inst**, **db2ethmon**)
from /tmp/.../Db2agents into /usr/lib/ocf/resource.d/heartbeat/

```
[root@server1 ]$ db2cm -copy_resources
/tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/Db2agents -host server1
[root@server1 ]$ db2cm -copy_resources
/tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/Db2agents -host server2
```

3.4 Pacemaker Cluster Setup

The following steps are required to **run only once** on any one of the hosts **server1** or **server2** by root. There is no need to run them in both hosts. Choose one of the hosts to perform all actions on the same host.

For the network device name use the name of the network adapter which is configured on your corresponding server (ens33 in our environment)
The domain name (aka cluster) is hadom in our environment. You can only ever have one domain per server.

Use a terminal window and switch to root:

```
[db2inst1@server1 ~]$ su - root
Password:
```

Pacemaker reacts to events regarding the cluster with its resource management. A resource is a service made highly available by a cluster. Every resource has a resource agent. A resource agent is an external program (see scripts in /usr/lib/ocf/resource.d/heartbeat/) that abstracts the service it provides and present a consistent view to the cluster.

For our HADR environment we have to define this resources:

- Public network resources
- Instance resource
- Database resource
- Virtual IP address (VIP) resources

Create the Pacemaker cluster and the public network resources:

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -cluster -domain hadom -host
server1 -publicEthernet ens33 -host server2 -publicEthernet ens33
Created db2_server1_ens33 resource.
Created db2_server2_ens33 resource.
Cluster created successfully.
```

Create the instance resource model for both, **server1** and **server2**:

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -instance db2inst1 -host server1
Created db2_server1_db2inst1_0 resource.
Instance resource for db2inst1 on server1 created successfully.
```

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -instance db2inst1 -host server2
Created db2_server2_db2inst1_0 resource.
Instance resource for db2inst1 on server2 created successfully.
```

Verify the cluster by using the *crm* statement and the *status* option:

```
[root@server1 ~]$ crm status
```

Now create the HADR database resources. Be sure that the appropriate database is installed and HADR is up and running.

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -db SAMPLE -instance db2inst1
Database resource for SAMPLE created successfully.
```

Create the virtual IP address (VIP) resources for the newly created database

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -primaryVIP 10.0.0.101 -db
SAMPLE -instance db2inst1
Primary VIP resource created successfully.
```

The virtual IP address is only available on one server of the cluster at any time. Check on both, **server1** and **server2** the network configuration. Which server has the VIP?

```
[root@server1 ~]$ ip addr show
```

```
[root@server2 ~]$ ip addr show
```

Check again the cluster configuration to become familiar with the resource types and the configuration.

Verify the cluster again using crm status

```
[root@server1 ~]$ crm status
```

Verify that the associated constraints have been created by running the crm config show command

```
[root@server1 ~]$ crm config show
```

3.5 Install and configure a QDevice quorum

The QDevice quorum requires a third host accessible via a TCP/IP network by the other hosts in the cluster (server3 in our lab example). However, the third host itself does not need to be configured as a part of the cluster. There is no need to have the Db2 or Pacemaker software installed. The only requirement is to install a corosync-qnetd RPM on it.

- On **server1** and **server2**, ensure the **corosync-qdevice** package is installed with the following command:

```
[db2inst1@server1 ~]$ rpm -qa | grep corosync-qdevice
```

- If it is not installed, install it using the following command on both servers:

```
[db2inst1@server1 ~]$ su - root
[root@server1 ~]$ dnf install
/tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/RPMS/x86_64/corosync-qdevice*
```

- On **server3**, install the Corosync QNet software with the following commands:

```
[db2inst1@server1 ~]$ su - root
[root@server1 ~]$ dnf install
/tmp/Db2_v11.5.5.0_Pacemaker_20201118_RHEL8.1_x86_64/RPMS/x86_64/corosync-qnetd*
```

- As root user, run the following **db2cm** command to setup the QDevice from one of the cluster nodes (server1 or server2). In our example, we execute it from **server1**:

```
[root@server1 ~]$ /home/db2inst1/sqllib/adm/db2cm -create -qdevice server3
```

- Run the following corosync command on the **server1** and **server2** to verify that the quorum was setup correctly:

```
[root@server1 ~]# corosync-qdevice-tool -s
Qdevice information
-------------------
Model:              Net
Node ID:            1
Configured node list:
    0  Node ID = 1
    1  Node ID = 2
Membership node list:      1, 2

Qdevice-net information
----------------------
Cluster name:       hadom
QNetd host:         server3:5403
Algorithm:          LMS
Tie-breaker:        Node with lowest node ID
State:              Connected
```

- Run the following corosync command on the QDevice host (**server3**) to verify that the quorum device is running correctly:

```
[root@server3 ~]# corosync-qnetd-tool -l
```

- Verify that both hosts and the cluster resources are online. As a root user run the following command:

```
[root@server1 ~]# /home/db2inst1/sqllib/adm/db2cm -list
...
Qdevice information
-------------------
Model:               Net
Node ID:             2
Configured node list:
    0  Node ID = 1
    1  Node ID = 2
Membership node list:      1, 2

Qdevice-net information
----------------------
Cluster name:        hadom
QNetd host:          server3:5403
Algorithm:           LMS
Tie-breaker:         Node with lowest node ID
State:               Connected
```

3.6 Play with the Db2 cluster manager utility

There are a few commands that an administrator should know to maintain the cluster. You always use the utility *db2cm* which you will find by default in the directory /opt/ibm/db2/V11.5/adm/. The utility can be run from any cluster server.

First, let's examine the cluster configuration. Display the configuration and status information with

```
[root@server1 ~]$ db2cm -list
```

Look at the output. Which resource types are being used?

Next backup the cluster configuration to a file with the use of the `–export` option. The generated file is in text format.

```
[root@server1 ~]$ db2cm -export ./db2cm_backup.conf
```

Inspect the file (e.g. with *cat ./db2cm_backup.conf*). Find out when the physical hostnames are used and when the virtual IP address.

Just in case you want to restore the cluster configuration from a previously saved configuration, use the utility with the *–import* option

```
[root@server1 ~]$ db2cm -import ./db2cm_backup.conf
```

For some maintenance work it is necessary to disable the cluster automation. To do this, execute the following two statements and check the status of the cluster.
Disable automation for all Pacemaker resources of all Db2 instances in the Pacemaker domain

```
[root@server1 ~]$ db2cm -disable
```

Have a closer look at the cluster status. Which status parameters have now changed?

Why is the state of each cluster resource still online?

Enable automation for all Pacemaker resources of all Db2 instances in the Pacemaker domain

```
[root@server1 ~]$ db2cm -enable
```

Check the cluster status once more to be sure that all resources are managed again.

3.7 Planned Db2 takeover with Pacemaker

Before performing a takeover of the database we prepare server3 to be a application server for the Db2 client. Therefore catalog on **server3** the cluster using the virtual IP address first.

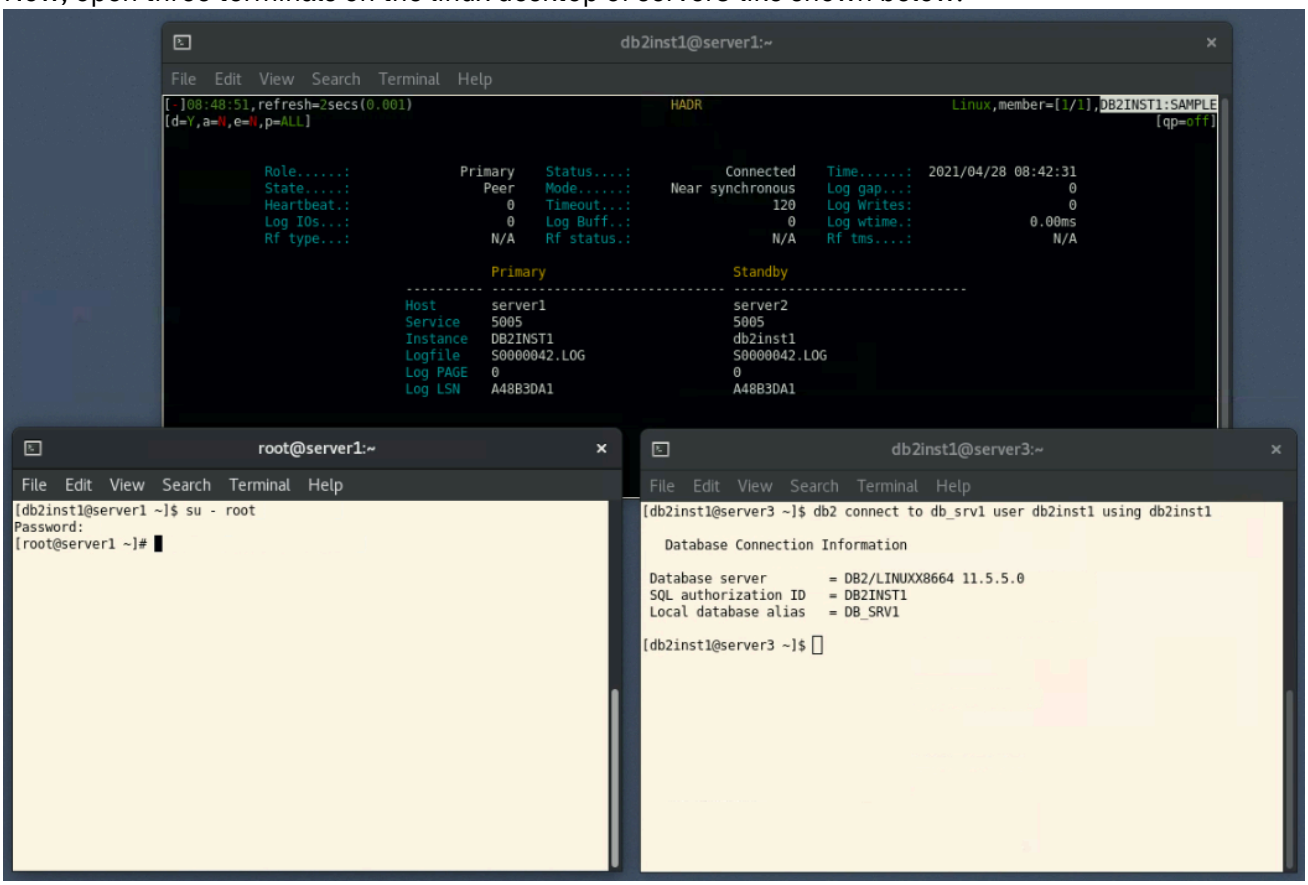
```
[db2inst1@server3 ~]$ db2 catalog TCPIP NODE srv1 REMOTE 10.0.0.101 SERVER 50000
[db2inst1@server3 ~]$ db2 CATALOG DATABASE sample AS db_srv1 AT NODE srv1
```

In this lab we are using automatic client reroute (ACR) to transfer client application requests from a failed database server to a standby database server. ACR does not use the *hadr_remote_host* and *hadr_remote_svc* database configuration parameters. Use the UPDATE ALTERNATE SERVER FOR DATABASE command on both, **server1** and **server2** to enable automatic client reroute.

```
[db2inst1@server1 ~]$ db2 update alternate server for database sample using hostname
10.0.0.101  port 50000
```

```
[db2inst1@server2 ~]$ db2 update alternate server for database sample using hostname
10.0.0.101  port 50000
```

Now, open three terminals on the linux desktop of server3 like shown below:



You will use the terminals to monitor the status of HADR and Pacemaker:

1. Run *db2top* in the first terminal (on top of the linux desktop). We use *db2top* to monitor HADR.

```
[db2inst1@server3 ~]$ s1
[db2inst1@server1 ~]$ db2top -d sample
```

In *db2top*, change to the HADR view (A) .

2. In the second terminal, switch to the standby server (this should be **server1** if you followed the previous steps) and check the status of pacemaker:

```
[db2inst1@server3 ~]$ s1
[db2inst1@server1 ~]$ su - root
[root@server1 ~]$ db2cm -list
```

3. In the third terminal, run a sample database workload as follows. Connect to the database:

```
[db2inst1@server3 ~]$ db2 connect to db_srv1 user db2inst1 using db2inst1

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.5.5.0
 SQL authorization ID   = DB2INST1
 Local database alias   = DB_SRV1
```

Usually when performing a planned takeover, all applications should be disconnected from the database. Therefore, disconnect from the database.

```
[db2inst1@server3 ~]$ db2 connect reset
DB20000I  The SQL command completed successfully.
```

Now, takeover the database on the standby server. In terminal 1 or 2, check which of the servers currently is the HADR primary server. In the following take over command we assume that server2 is the standby server. Use terminal 2 to execute the takeover as instance user.

```
[[root@server1 ~]$ su – db2inst1
[db2inst1@server1 ~]$ s2
[db2inst1@server2 ~]$ db2 takeover hadr on database sample
```

Have a look at the HADR status in terminal 1 and at the cluster status in terminal 2 (as user *root* run `crm status` and `db2cm -list`).

Connect once more in terminal 3 to the database and run a SQL query. Does it work?

3.8 Outage Simulation and Unplanned Failover

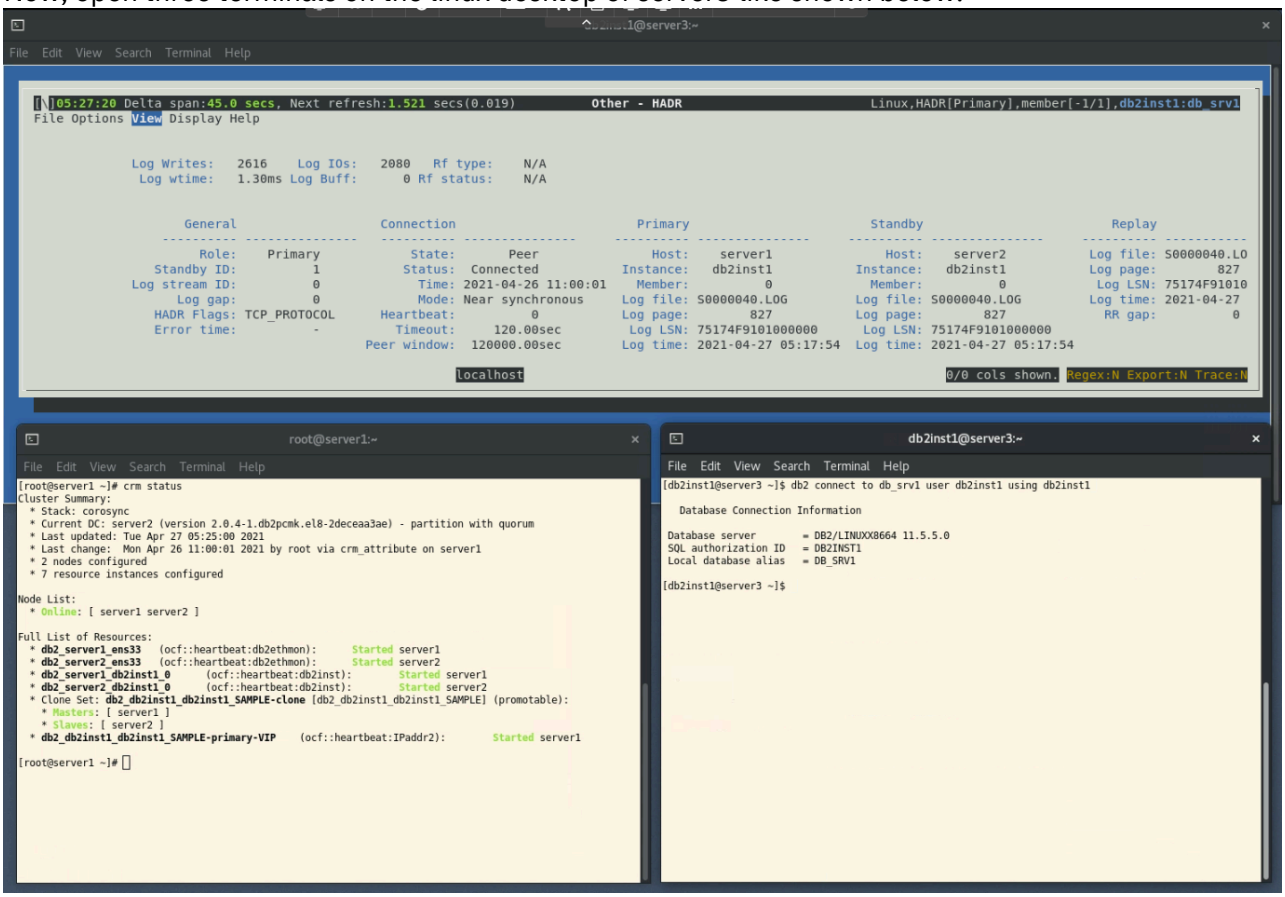
The primary purpose of Pacemaker is protection against unplanned failures of hardware or software. In this section, we simulate a server failure and demonstrate, how Pacemaker ensures that applications continue to run without interruption.

First, we make sure that Db2 does not start automatically upon a server restart. Verify, that an automatic start of the Db2 instance is disabled by running the following command on both, **server1** and **server2**:

```
[db2inst1@server1 ~]$ db2set DB2AUTOSTART -i db2inst1

DBI1303W  Variable not set.
```

Now, open three terminals on the linux desktop of server3 like shown below:



You will use the terminals to monitor the status of HADR and Pacemaker:

1. Run *dsmtop* in the first terminal (on top of the linux desktop). We use dsmtop to simulate an application that connects to the database from server3 and generates some SQL workload. We also use dsmtop to monitor HADR.

```
[db2inst1@server3 ~]$ dsmtop -n 10.0.0.101 -d sample -j 4 -u db2inst1 -p db2inst1  -x
```

   In dsmtop, change to the HADR view: View (V) → other (o) → HADR (A) .

2. In the second terminal, switch to the standby server (this should be **server2** if you followed the previous steps) and check the status of pacemaker:

```
[db2inst1@server3 ~]$ s2
[db2inst1@server2 ~]$ su - root
```

```
[root@server2 ~]$ db2cm -list
```

3. In the third terminal, run a sample database workload as follows. Connect to the database:

```
[db2inst1@server3 ~]$ db2 connect to db_srv1 user db2inst1 using db2inst1

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.5.5.0
 SQL authorization ID   = DB2INST1
 Local database alias   = DB_SRV1
```

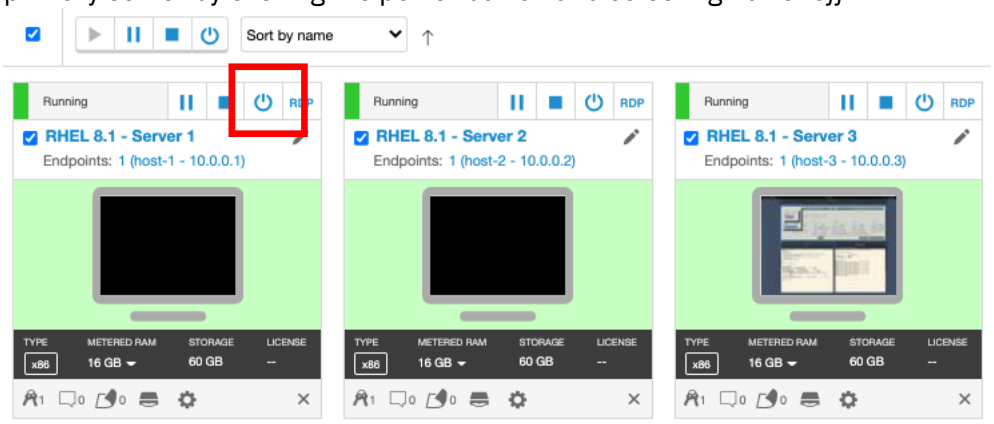
Insert a row into table app1:

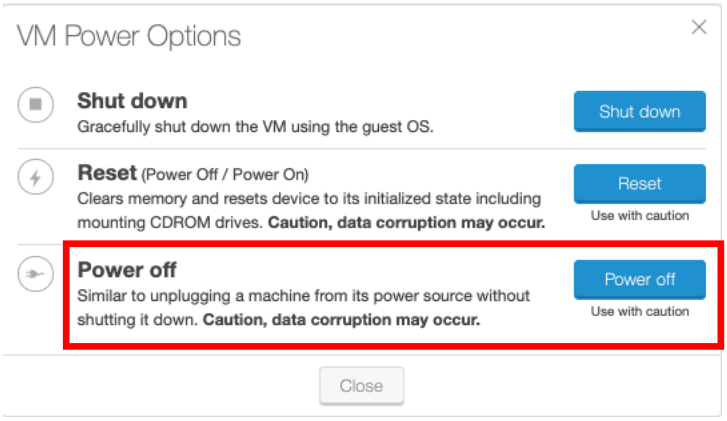
```
[db2inst1@server3 ~]$ db2 "insert into app1 values current timestamp"
DB20000I  The SQL command completed successfully.
```

In dsmtop, you can see a gap in the log sequence number (Log LSN) for a short time:

```
   Primary                        Standby
---------- ---------------     ---------- ---------------
     Host:    server1               Host:    server2
 Instance:   db2inst1           Instance:   db2inst1
   Member:          0             Member:          0
 Log file: S0000040.LOG         Log file: S0000040.LOG
 Log page:        827           Log page:        827
  Log LSN: 71194F9101000000      Log LSN: F2184F9101000000
 Log time: 2021-04-27 05:44:35  Log time: 2021-04-27 05:44:28
```

Now, let us simulate an outage of the primary server. In dsmtop, check which of the servers currently is the HADR primary server. Then switch to the skytap dashboard in your browser and power off the primary server by clicking the power button and selecting *Power off*





Switch back to the browser tab with your three terminal windows. Have a look at the HADR status in terminal 1 and at the cluster status in terminal 2 (`crm status` and `db2cm -list`).

```
[root@server2 ~]# crm status
Cluster Summary:
  * Stack: corosync
  * Current DC: server2 (version 2.0.4-1.db2pcmk.el8-2deceaa3ae) - partition with quorum
  * Last updated: Tue Apr 27 06:04:38 2021
  * Last change:  Tue Apr 27 06:02:31 2021 by root via crm_attribute on server2
  * 2 nodes configured
  * 7 resource instances configured

Node List:
  * Online: [ server2 ]
  * OFFLINE: [ server1 ]

Full List of Resources:
  * db2_server1_ens33      (ocf::heartbeat:db2ethmon):       Stopped
  * db2_server2_ens33      (ocf::heartbeat:db2ethmon):       Started server2
  * db2_server1_db2inst1_0 (ocf::heartbeat:db2inst):   Stopped
  * db2_server2_db2inst1_0 (ocf::heartbeat:db2inst):   Started server2
  * Clone Set: db2_db2inst1_db2inst1_SAMPLE-clone [db2_db2inst1_db2inst1_SAMPLE]
(promotable):
    * Masters: [ server2 ]
  * db2_db2inst1_db2inst1_SAMPLE-primary-VIP   (ocf::heartbeat:IPaddr2):   Started server2
```

```
[root@server2 ~]# db2cm -list
      Cluster Status

Domain information:
Domain name                 = hadom
Pacemaker version           = 2.0.4-1.db2pcmk.el8
Corosync version            = 3.0.4
Current domain leader       = server2
Number of nodes             = 2
Number of resources         = 7

Node information:
Name name           State
---------------     --------
server2             Online
server1             Offline
```

```
Resource Information:

Resource Name              = db2_db2inst1_db2inst1_SAMPLE
  Resource Type                 = HADR
    DB Name                     = SAMPLE
    Managed                     = true
    HADR Primary Instance       = db2inst1
    HADR Primary Node           = server2
    HADR Primary State          = Online
    HADR Standby Instance       =
    HADR Standby Node           =
    HADR Standby State          = Offline

Resource Name              = db2_db2inst1_db2inst1_SAMPLE-primary-VIP
  State                         = Online
  Managed                       = true
  Resource Type                 = IP
    Node                        = server2
    Ip Address                  = 10.0.0.101

Resource Name              = db2_server1_db2inst1_0
  State                         = Offline
  Managed                       = true
  Resource Type                 = Instance
    Node                        = server1
    Instance Name               = db2inst1

Resource Name              = db2_server1_ens33
  State                         = Offline
  Managed                       = true
  Resource Type                 = Network Interface
    Node                        = server1
    Interface Name              = ens33

Resource Name              = db2_server2_db2inst1_0
  State                         = Online
  Managed                       = true
  Resource Type                 = Instance
    Node                        = server2
    Instance Name               = db2inst1

Resource Name              = db2_server2_ens33
  State                         = Online
  Managed                       = true
  Resource Type                 = Network Interface
    Node                        = server2
    Interface Name              = ens33

Fencing Information:
  Not configured
Quorum Information:
  Qdevice

Qdevice information
-------------------
Model:              Net
Node ID:            2
```

```
Configured node list:
    0  Node ID = 1
    1  Node ID = 2
Membership node list:      2

Qdevice-net information
----------------------
Cluster name:       hadom
QNetd host:         server3:5403
Algorithm:          LMS
Tie-breaker:        Node with lowest node ID
State:              Connected
```

In terminal 1, you see that server1 and server2 switched the roles and the standby server is not online any more (no Log LSN is shown). The HADR status is *Disconnected*. In terminal 2, you see that one database server is offline and the cluster resources for this server are offline as well.

Now, we execute another SQL query in terminal 3. This results in Db2 error *SQL30108N*:

```
[db2inst1@server3 ~]$ db2 "select count(*) from app1"
SQL30108N  A connection failed in an automatic client reroute environment. The
transaction was rolled back. Host name or IP address: "10.0.0.101". Service
name or port number: "50000". Reason code: "1". Connection failure code: "1".
Underlying error: "110".  SQLSTATE=08506
```

Repeat the same SQL query once again:

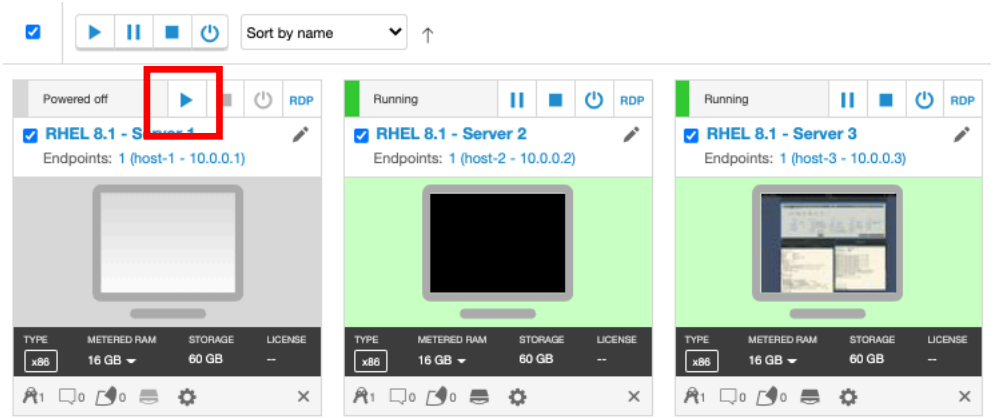
```
[db2inst1@server3 ~]$ db2 "select count(*) from app1"

1
-----------
          9

  1 record(s) selected.
```

The DB2 driver *automatically* reconnected to the database before the SQL query was executed. If you want, you can now take a break and have a coffee, because the application continues to run. You don't need to worry about the server failures anymore ☺.

Now, it is time to restart the failed server and monitor the status in terminal 1 and terminal 2 again.



The server is online again after about 60 seconds. Check the status in terminal 1 and 2. You notice, that the application in terminal 1 (dsmtop) continues to run without interruption.

Which steps were automatically performed by Pacemaker to restart the failed database and the corresponding Db2 resources?

As part of starting the Db2 instance, the db2inst resource agent will kill any existing db2sysc processes via `kill -9 <pid>` and then run `ipclean -a`. Then it will attempt to start the instance using a db2gcf command (equivalent to db2start). If the instance starts successfully, it will also attempt to activate all the automated databases asynchronously.

The way a database is activated after an outage depends on its previous HADR role:

- STANDBY: If the database had the STANDBY role assigned, it will simply be reactivated by Pacemaker (db2 activate db <database name>)
- PRIMARY: If the database had the PRIMARY role assigned before the outage, Pacemaker has issued an HADR takeover command when the database outage was detected. In this case the database needs to be reintegrated as STANDBY. To accomplish the reintegration, Pacemaker first forces off all connected applications from the database. Once all applications are forced off, Pacemaker reactivates the database with the standby role (db2 start hadr on db <database name> as standby).

3.9 Advanced Cluster Configuration with Pacemaker

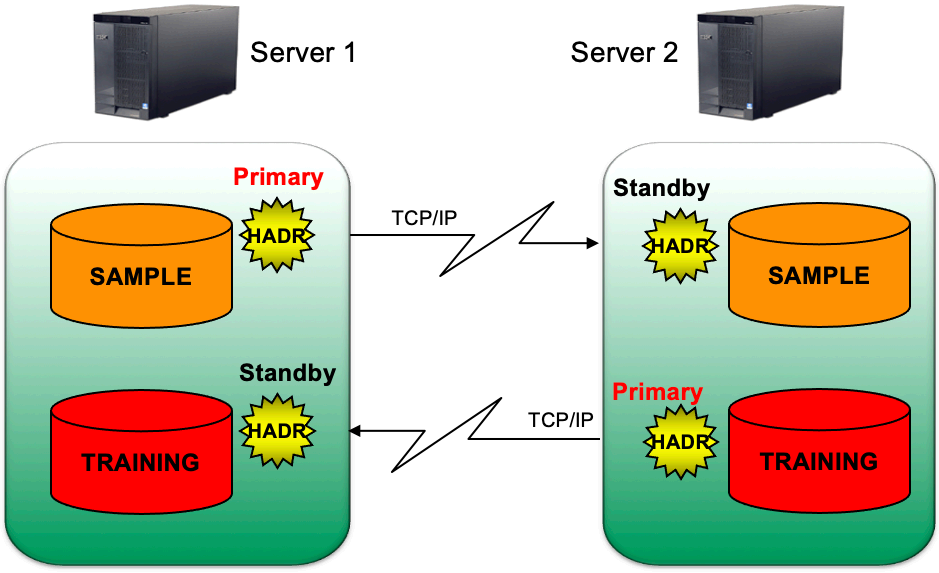
With the statement

```
db2cm  -create -cluster -domain hadom -host server1 -publicEthernet ens33
       -host server2 -publicEthernet ens33
```

you created a cluster with the domain name *hadom*.

Question:
How many domains can you define in a logical environment?
Why?

Now let us configure a more advanced database environment. Assuming we have 2 data centers and two databases which we want to keep high available with HADR.



Create a second database on our servers. In the lab directory */home/db2inst1/Lab_HADR_Pacemaker* there is already a script *11_create_db_TRAINING .sh* which creates a primary database on server2 and a standby database on server1 including parameter settings for HADR. Run the scripts as user db2inst1 on **server2**:

```
[db2inst1@server2 ]$ cd /home/db2inst1/Lab_HADR_Pacemaker
[db2inst1@server2 ]$ ./11_create_db_TRAINING.sh
```

Now think about which statements are necessary to expand the cluster configuration for the database TRAINING.

Just in case you want to cheat, the statements are in chapter 4.

If you get the message regarding "... target host is pingable", then run this statements as root on server1 and server2 and repeat your command after that.

```
[root@server1 ~]# iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
```

```
[root@server2 ~]# iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
```

Check your new definition with crm status. The cluster status should looks like this:

```
[root@server2 ~]# crm status
Cluster Summary:
  * Stack: corosync
  * Current DC: server2 (version 2.0.4-1.db2pcmk.el8-2deceaa3ae) - partition with quorum
  * Last updated: Thu Apr 29 16:54:40 2021
  * Last change:  Thu Apr 29 16:38:53 2021 by db2inst1 via crm_resource on server1
  * 2 nodes configured
  * 10 resource instances configured

Node List:
  * Online: [ server1 server2 ]

Full List of Resources:
  * db2_server1_ens33      (ocf::heartbeat:db2ethmon):       Started server1
  * db2_server2_ens33      (ocf::heartbeat:db2ethmon):       Started server2
  * db2_server1_db2inst1_0 (ocf::heartbeat:db2inst):   Started server1
  * db2_server2_db2inst1_0 (ocf::heartbeat:db2inst):   Started server2
  * Clone Set: db2_db2inst1_db2inst1_SAMPLE-clone [db2_db2inst1_db2inst1_SAMPLE]
(promotable):
    * Masters: [ server1 ]
    * Slaves: [ server2 ]
  * db2_db2inst1_db2inst1_SAMPLE-primary-VIP   (ocf::heartbeat:IPaddr2):   Started server1
  * Clone Set: db2_db2inst1_db2inst1_TRAINING-clone [db2_db2inst1_db2inst1_TRAINING]
(promotable):
    * Masters: [ server1 ]
    * Slaves: [ server2 ]
  * db2_db2inst1_db2inst1_TRAINING-primary-VIP (ocf::heartbeat:IPaddr2):   Started server1
```

4 Solutions

Questions chapter 3.6

Question:
Which resource types are being used?
Answer:

- HADR
  (db2_db2inst1_db2inst1_SAMPLE)
- IP
  (db2_db2inst1_db2inst1_SAMPLE-primary-VIP)
- Instance
  (db2_server1_db2inst1_0, db2_server2_db2inst1_0)
- Network Interface
  (db2_server1_ens33, db2_server2_ens33)

Question:
Find out when the physical hostnames are used and when the virtual IP address.
Answer:
Xxx

Question:
Have a closer look to the cluster status. Which status parameters have now changed?
Answer:
Managed

Question:
Why is the state of each cluster resource still online?
Answer:
The resource is still configured and the resource agent still available. But the cluster is not allowed to start the resource agent since the *managed* flag is false.

Questions chapter 3.93.6

Question:
How many domains can you define in a logical environment? Why?
Answer:
Xxx

Question:
Which statements are necessary to expand our cluster configuration for the database xy.
Answer:

The resources for the instance is there already. You only have to define a new resource for the database and a new resource for a virtual IP address (VIP).

Run this commands as user root on **server1** or **server2**:

```
[root@server2 ~]$ db2cm -create -db TRAINING -instance db2inst1
[root@server2 ~]$ db2cm -create -primaryVIP 10.0.0.102 -db TRAINING -instance db2inst1
```

Last, you have to define the alternate server address for your database as on both, server1 and server2:

```
[db2inst1@server1 ~]$ db2 update alternate server for database TRAINING using hostname
10.0.0.102  port 50000
[db2inst1@server2 ~]$ db2 update alternate server for database TRAINING using hostname
10.0.0.102  port 50000
```

5 Appendix

5.1 Preparational Steps not covered in the Lab

Here are the relevant documentation for setting up Pacemaker:

- Prerequisites for an integrated solution using Pacemaker
  https://www.ibm.com/docs/en/db2/11.5?topic=pacemaker-prerequisites-integrated-solution-using

- Installing the Pacemaker cluster software stack
  https://www.ibm.com/docs/en/db2/11.5?topic=utility-installing-pacemaker-cluster-software-stack

- Configuring a clustered environment using the Db2 cluster manager (db2cm) utility
  https://www.ibm.com/docs/en/db2/11.5?topic=pacemaker-configuring-clustered-environment-using-db2cm-utility
  This document outlines the db2cm commands to set up the cluster resources.