High Availavility (HA)

Go back to Pandora FMS documentation index

We are working on the translation of the Pandora FMS documentation. Sorry for any inconvenience.

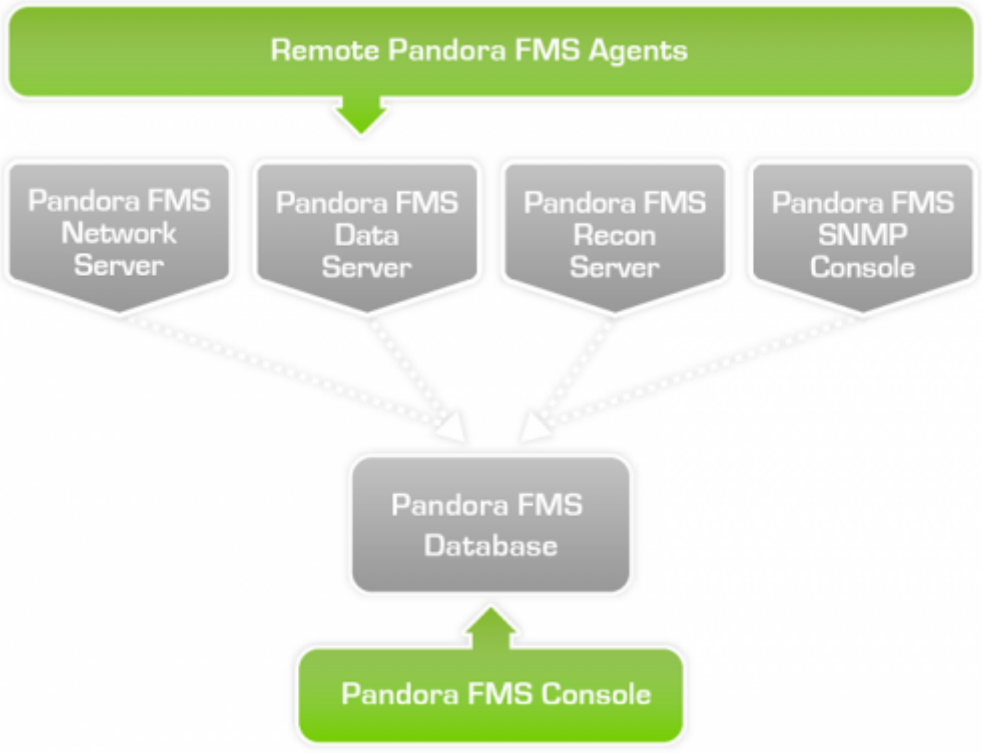
High Availability

Introduction

Pandora FMS is a very stable application (thanks to the test and improvements included in each version and to the fixing of some failures discovered by users. In spite of this, in critical environments and/or with high load, it is possible that it would be necessary to distribute the load among several machines, making sure that if any component of Pandora FMS fails, the system will not be down.

Pandora FMS has been designed to be very modular. Any of its modules could work in an independent way. But it has also been designed to work with other components and for being able to take the load from those components that have been down.

The Pandora FMS standard design could be this one:



Obviously, the agents are not redundant. If an agent is down, it makes no sense to execute another one, since the only cause for an agent downs is that data could not be obtained because the execution of any module is failing, and this could not be solved with another agent running in parallel, or because the system is isolated or fails. The best solution is to make the critical systems redundant - regardless of they having Pandora FMS agents or not- and so to make the monitoring of these systems redundant.

It is possible to use HA in several scenaries:

- Data Server Balancing and HA.
- Network Servers,WMI, Plugin, Web and Prediction Balancing and HA
- DDBB Load Balancing.
- Recon Servers Balancing and HA.
- Pandora FMS Console Balancing and HA.

Dimensioning and HA architecture designs

The most important components of Pandora FMS are:

1. Database
2. Server
3. Console

Each of these components can be replicated to protect the monitoring system from any catastrophe.

To designate the number of nodes needed to balance the load, the number of targets to be monitored and the quantity, type and frequency of capture of the metrics to be collected will be studied.

Depending on the monitoring needs, the different architectures will be defined.

Note: The tests carried out to define the architectures have been carried out using different equipment:

Intel (R) Core (TM) i5-8600K CPU @ 3.60GHz

Instance *t2.large* from Amazon [1)]

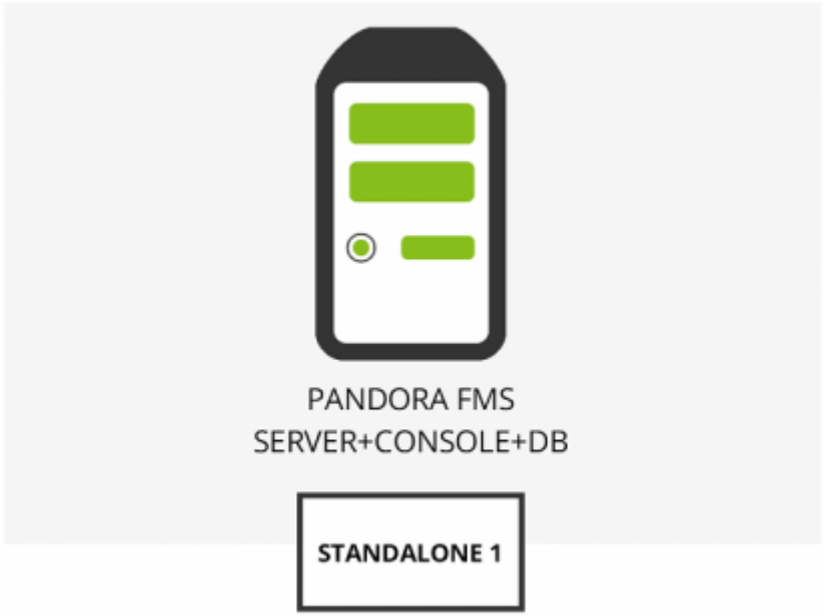
Sizing

Depending on the needs:

1. Standalone (without high availability) up to 2500 agents / 50000 modules every 5 minutes, even data, no historical data.

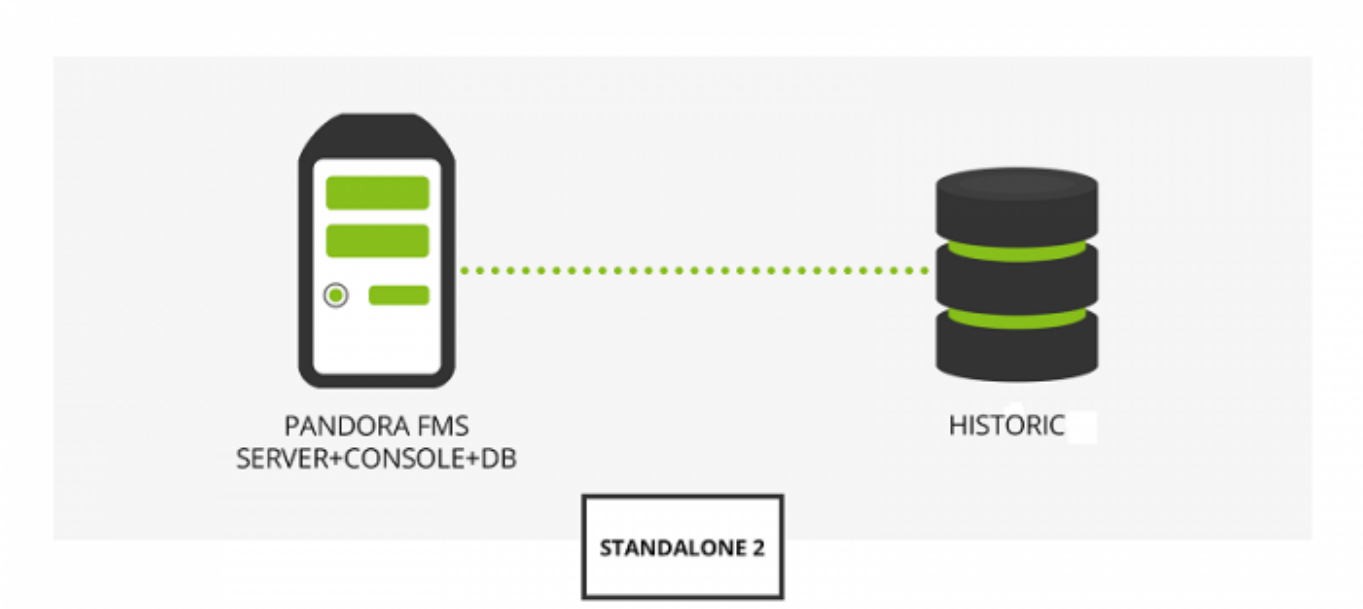
```
Servers: 1 (shared)

Main:
----------
CPU: 6 cores
```

```
RAM: 8 GB
Disk: 100GB
```



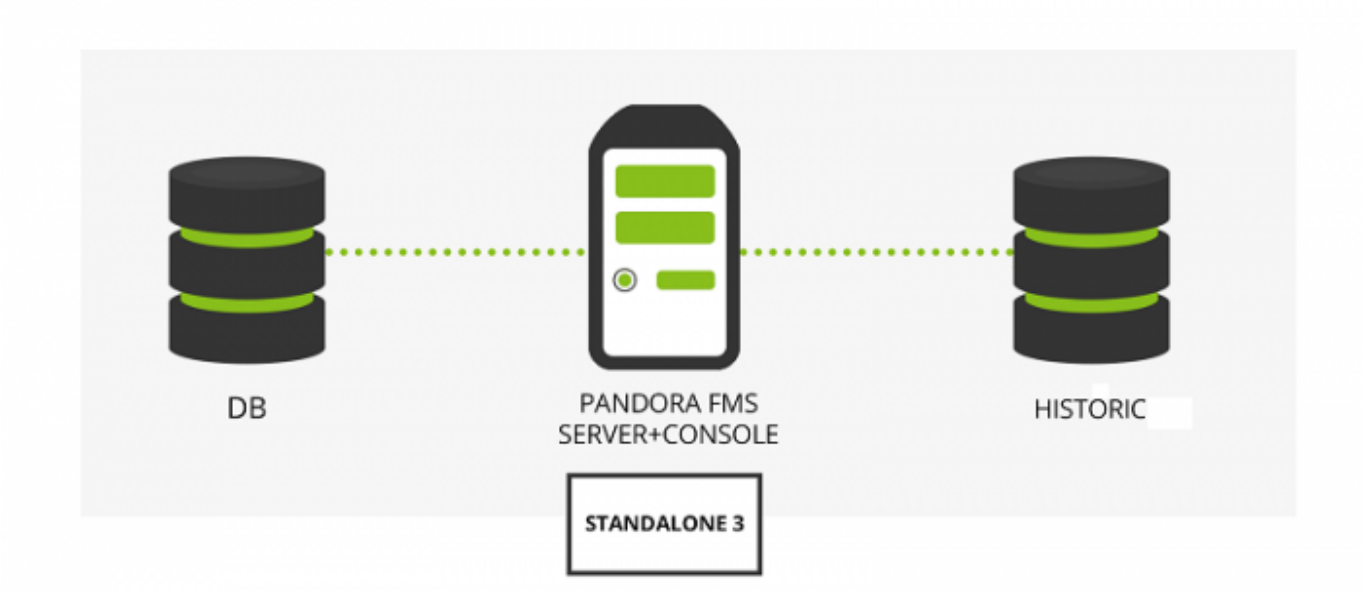
2. Standalone (without high availability) up to 2500 agents / 50000 modules every 5 minutes, even data, with historical data (1 year).

```
Servers: 2 (1 shared, 1 historical)

Main:
----------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Historical:
----------
CPU: 2 cores
RAM: 4 GB
Disk: 200GB
```



3. Standalone (without high availability) up to 5000 agents / 100000 modules every 5 minutes, even data, with historical data (1 year).

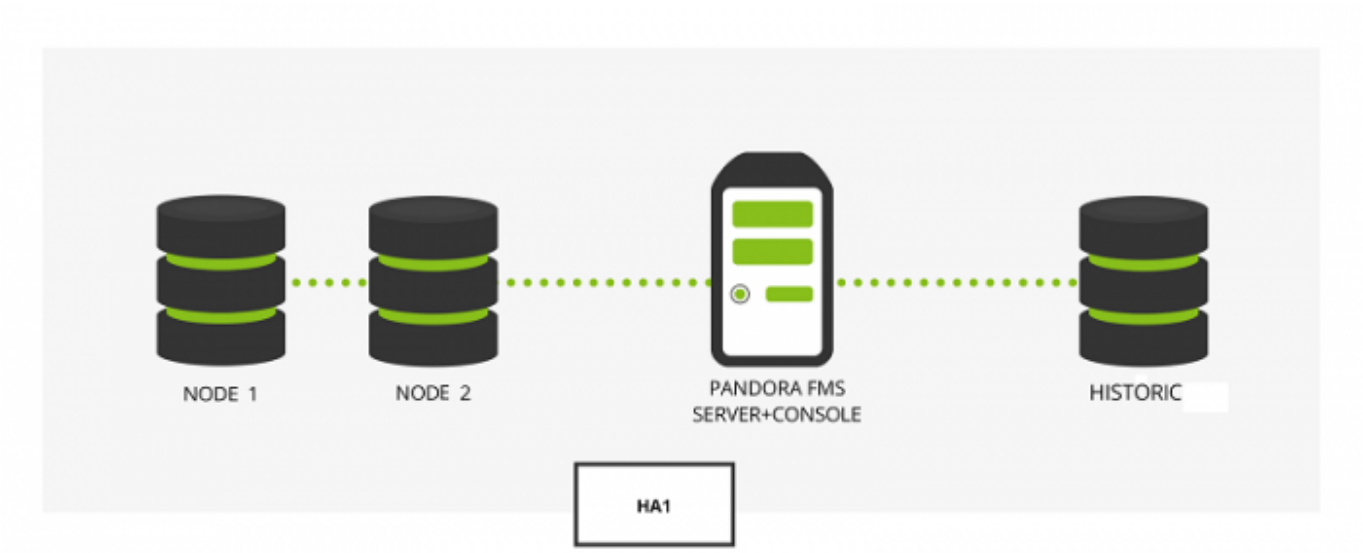
```
Servers: 3 (1 server + console, 1 main database, 1 historical)

Server + console:
-------------------
CPU: 6 cores
RAM: 8 GB
Disk: 40GB

Main database:
------------------------
CPU: 4 cores
RAM: 8 GB
Disk: 100GB

Historical:
----------
CPU: 2 cores
RAM: 4 GB
Disk: 200GB
```



HA Architecture designs

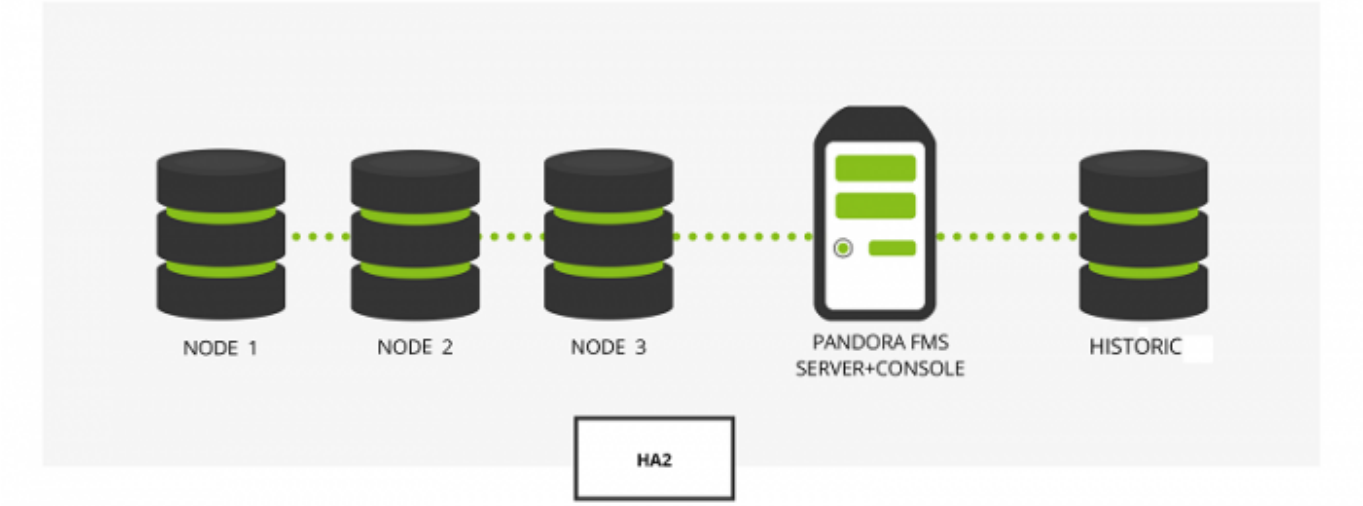
1. Database in simple HA, up to 7500 agents / 125000 modules every 5 minutes, even data, with historical data (1 year).

```
Servers: 4 (1 server + console, 2 database, 1 historical)

 Server + console:
 -------------------
 CPU: 6 cores
 RAM: 8 GB
 Disk: 40GB

 Database node 1:
 ----------------------
 CPU: 6 cores
 RAM: 8 GB
 Disk: 100GB

 Database node 2:
 ----------------------
 CPU: 6 cores
 RAM: 8 GB
 Disk: 100GB

 Historical:
 ----------
 CPU: 2 cores
 RAM: 4 GB
 Disk: 300GB
```



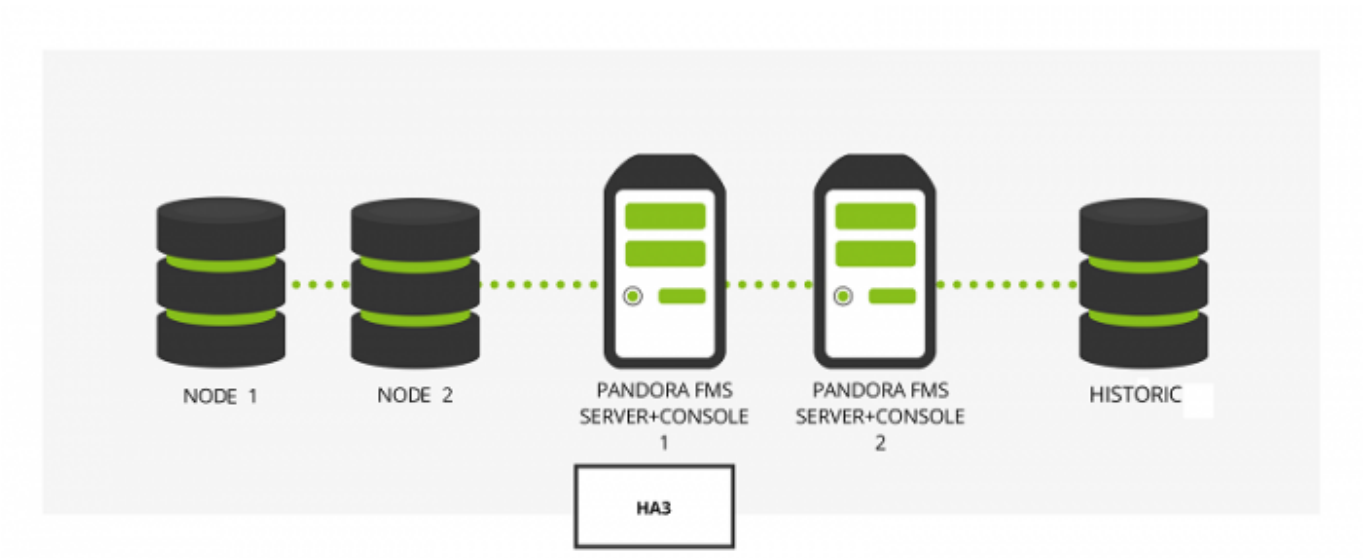
2. Database in complete HA (with quorum), up to 7500 agents / 125000 modules every 5 minutes, even data, with historical data (1 year).

```
Servers: 5 (1 server + console, 3 database, 1 historical)

Server + console:
------------------
CPU: 6 cores
RAM: 8 GB
Disk: 40GB

Database node 1:
----------------------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Database node 2:
----------------------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Database node 3:
----------------------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Historical:
----------
CPU: 2 cores
RAM: 4 GB
Disk: 200GB
```



3. Database in HA simple and Pandora FMS in HA balanced, up to 7500 agents / 125000 modules every 5 minutes, even data, with historical data (1 year).

```
Servers: 5 (2 server + console, 2 database, 1 historical)

Server + console:
--------------------
CPU: 6 cores
RAM: 8 GB
Disk: 40GB

Server + console:
--------------------
CPU: 6 cores
RAM: 8 GB
Disk: 40GB

Database node 1:
----------------------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Database node 2:
----------------------
CPU: 6 cores
RAM: 8 GB
Disk: 100GB

Historical:
----------
CPU: 2 cores
RAM: 4 GB
Disk: 200GB
```



4. Basic HA balanced on server, main and replica database, up to 4000 agents / 90000 modules every 5 minutes, even data, with historical data (1 year).

```
Servers: 3 (2 shared, 1 historical)

Main: (console + server + database node 1)
----------
CPU: 8 cores
RAM: 12 GB
Disk: 100GB

Secondary: (console + server + database node 2)
----------
CPU: 8 cores
RAM: 12 GB
Disk: 100GB

Historical:
----------
CPU: 2 cores
RAM: 4 GB
Disk: 200GB
```

In this overview, Pandora FMS database nodes are configured in each of the two available servers (main and secondary).

Note: For large environments, each of the configuration overviews previously described as computing nodes will be defined.

Example

If you need to monitor 30,000 agents with 500,000 modules, configure as many nodes as necessary to cover these requirements. Follow the example:

If you choose the HA # 1 design (1 server + console, 2 database nodes in HA, and a historical database), you must configure 30,000 / 7500 = 4 nodes.

To manage the entire environment, it will be necessary to have an installed Metaconsole, from which to configure the entire monitoring infrastructure.

The Metaconsole will require:

```
Servers: 1 (shared)

Main:
----------
CPU: 8 cores
RAM: 12 GB
Disk: 100GB
```

Total servers with independent historical databases: 17

Total servers with combined historical databases: 13

Note > To combine all the historical databases (4) in a single team, resize their characteristics to take on the extra load:

```
Historical combined:
--------------------
CPU: 8 cores
```

```
RAM: 12 GB
Disk: 1200GB
```

HA of Data Server

The easiest way is to use the HA implemented in the agents (which allow you to contact an alternative server if the main one does not reply). However, since the data server supports port 41121 and it is a standard TCP port, it is possible to use any commercial solution that allows balancing or clustering an ordinary TCP service.

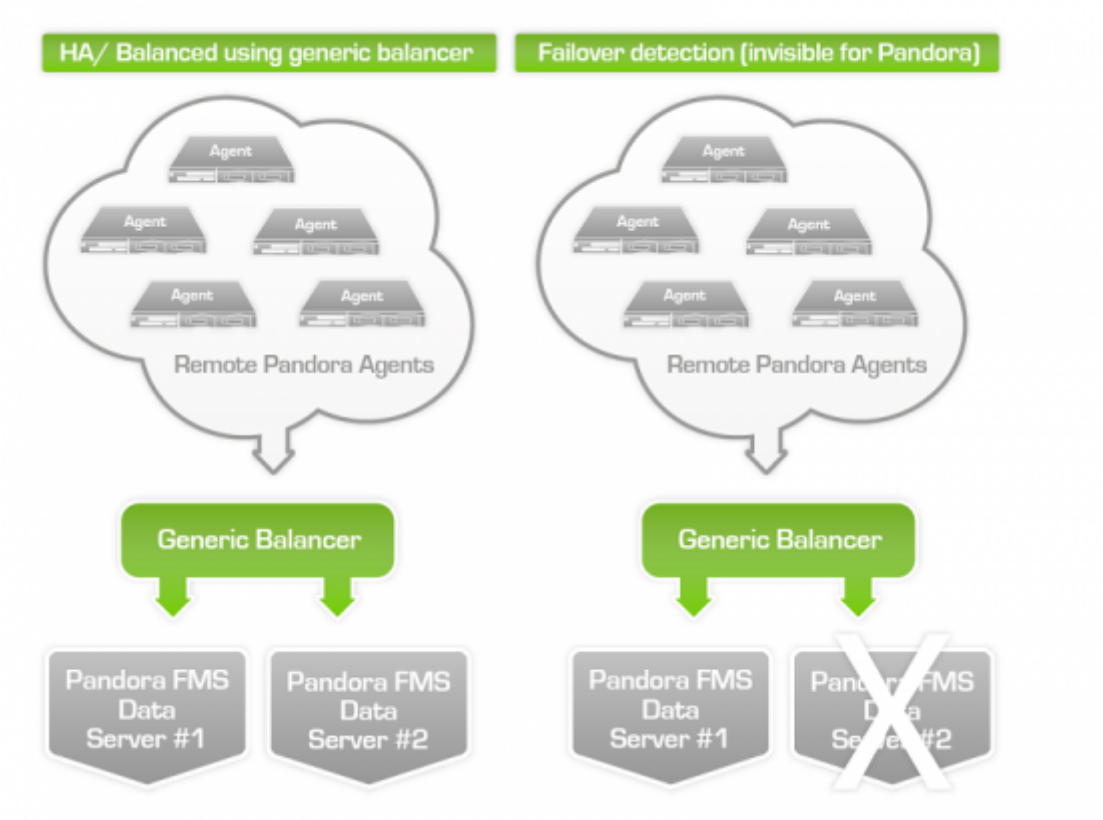
For Pandora FMS data server, you will need to mount two machines with a configured Pandora FMS data server (and different hostname and server name). You will have to configure a Tentacle server in each of them. Each machine will have a different IP address. If we are going to use an external balancer, this one will provide a single IP address to which the agents will connect to send their data.

If you are using an external balancer, and one of the servers fails, the HA mechanism enables one of the available active servers and Pandora FMS agents will keep on connecting with the same address as before, without noticing the change, but in this case, the load balancer will no longer send data to the server that failed, but to another active server. There is no need to change anything in every Pandora FMS data server, even each server can keep its own name. This is useful to find out if any of them has failed in the server status view. Pandora FMS data modules can be processed by any server without pre-assignment being necessary. It is designed precisely that way so that HA can be implemented more easily.

In the case of using the agent HA mechanism, there will be a small delay when sending data, since at each agent execution, it will try to connect with the primary server, and if it does not answer, it will do so against the secondary one (if it has been configured like that). This is described below as “Balancing in Software Agents”.

If you wish to use two data servers and for both to manage policies, collections, and remote configurations, you will need to share the following directories so that all data server instances can read and write over these directories. Consoles must have access to these shared directories as well.

- /var/spool/pandora/data_in/conf
- /var/spool/pandora/data_in/collections
- /var/spool/pandora/data_in/md5
- /var/spool/pandora/data_in/netflow
- /var/www/html/pandora_console/attachment



Balancing in the Software Agents

From the software agents, it is possible to balance data servers so it is possible to configure a master and backup data servers.

In the agent configuration file pandora_agent.conf, configure and uncomment the following part of the agent configuration file:

```
 # Secondary server configuration
 # ==============================
 # If secondary_mode is set to on_error, data files are copied to the
secondary
 # server only if the primary server fails. If set to always, data files are
 # always copied to the secondary server
 secondary_mode on_error
 secondary_server_ip localhost
 secondary_server_path /var/spool/pandora/data_in
 secondary_server_port 41121
 secondary_transfer_mode tentacle
 secondary_server_pwd mypassword
 secondary_server_ssl no
 secondary_server_opts
```

There are the following options (for more information, go to the Agent configuration chapter).

- **secondary_mode**: Secondary server mode. It may have two values:
  - **on_error**: Send data to the secondary server only if it cannot send them to the main server.

    - **always**: It always sends data to the secondary server, regardless of it being able to connect or not with the main server.
- **secondary_server_ip**: Secondary server IP.
- **secondary_server_path**: Path where the XML are copied in the secondary server, usually /var/spoo/pandora/data_in
- **secondary_server_port**: Port through which the XML will be copied to the secondary server, in tentacle 41121, in ssh 22 and in ftp 21.
- **secondary_transfer_mode**: Transfer mode that will be used to copy the XML to the secondary server, Tentacle, ssh, ftp, etc.
- **secondary_server_pwd**: Password option for FTP transfer.
- **secondary_server_ssl**: Yes or not should be typed in depending if you want to use ssl to transfer data through Tentacle or not.
- **secondary_server_opts**: This field is for other options that are needed for the transfer.

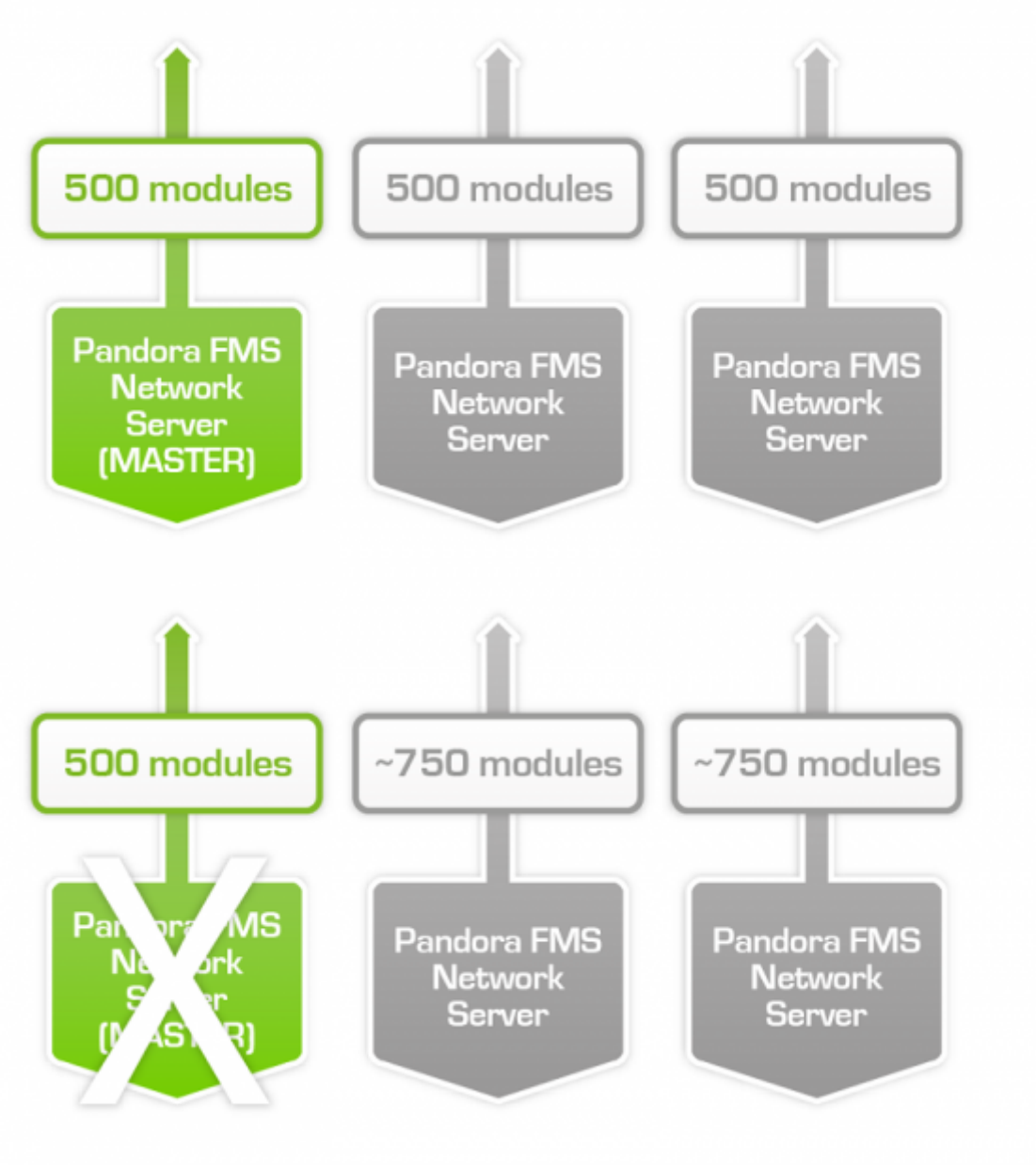
Only the remote configuration of the agent is operative in the main server, if enabled.

HA of network, WMI, plugin, web and prediction servers, among others

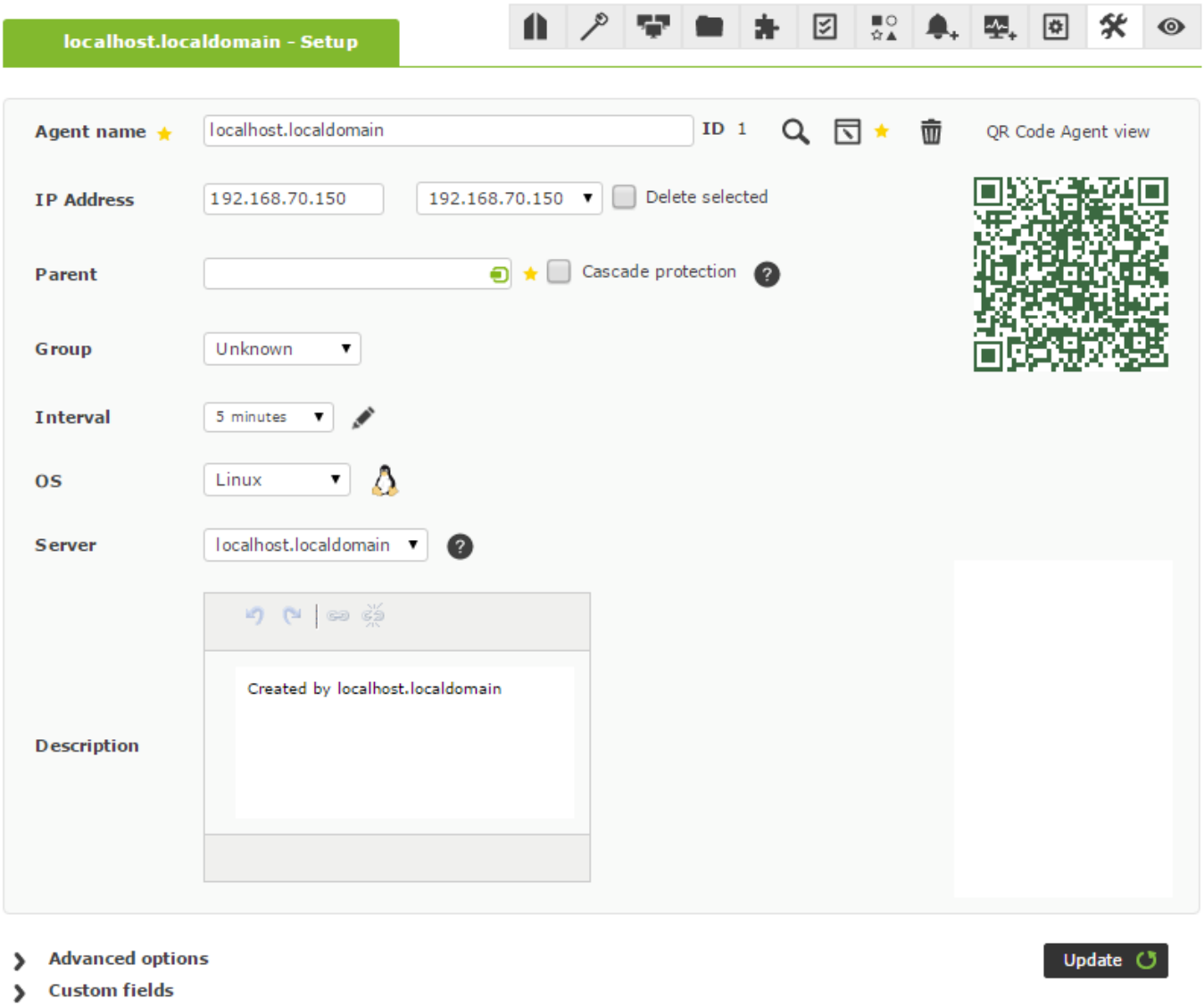
You must install several servers, network, WMI, plugin, web or prediction, in several machines of the network (all with the same visibility for the systems that you want monitor). All these machines should be in the same segment (so that network latency data are coherent).

The servers could be selected as primaries. These servers will automatically collect the data from all assigned modules to a server that is selected as “down”. Pandora FMS own servers implement a system to detect that one of them is down through verifying its last contact date (server threshold x 2). It will be enough if only one Pandora FMS server is active for it to detect whether the other ones fall down. If all Pandora FMS are down, there is no way to detect or to implement HA.

The obvious way to implement HA and load balancing in a system of two nodes is to assign 50% of the modules to each server and select both servers as masters. In case that there would be more than two master servers ,and a third server down with modules yet to be executed, the first of the master servers that executes the module will “self-assign” the module of the down server. In case of recovering one of the down servers, the modules that have been assigned to the primary server are automatically assigned again.



The load balancing between the different servers is done in the Agent Administration section in the “setup” menu.



In the “server” field, there is a combo where you can choose the server that will do the checking.

Server configuration

A Pandora FMS server can be running in two different modes:

- Master mode.
- Non-master mode.

If a server fails, its modules will be executed by the master server so that no data is lost.

At any given time there can only be one master server, which is chosen from all the servers with the *master* configuration option in */etc/pandora/pandora_server.conf* set to a value higher than 0:

```
master [1..7]
```

If the current master server fails, a new master server is chosen. If there is more than one candidate,

the one with the highest *master* value is chosen.

Be careful about disabling servers. If a server with Network modules fails and the Network Server is disabled in the master server, those modules will not be executed.

For example, if you have three Pandora FMS Servers with *master* set to 1, a master server will be randomly chosen and the other two will run in non-master mode. If the master server fails, a new master will be randomly chosen.

The following parameters have been entered in pandora_server.conf:

- ha_file: HA temporary binary file address.
- ha_pid_file: HA current process.
- pandora_service_cmd: Pandora FMS service status control.

Pandora FMS Console HA

Install another console. Any of them can be used simultaneously from different locations by different users. You may use a web balancer encompassing all consoles in case you need horizontal growth to manage console load. The session system is managed by cookies and they stay stored in the browser.

In the case of using remote configuration and to manage it from all consoles, both data servers and consoles must share the data directory (/var/spool/pandora/data_in) for remote agent, collections and directory configuration.

You can learn how to share the key folders with NFS or GlusterFS using this guide.

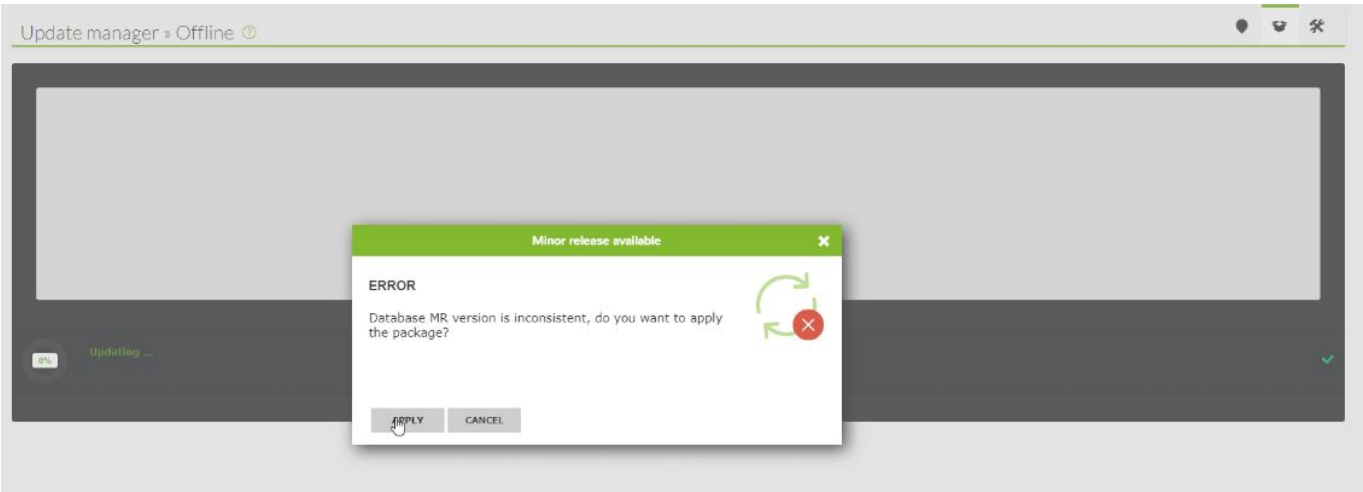
It is important to only share data_in's subdirectories and not data_in folder itself, since doing so would affect server performance negatively.

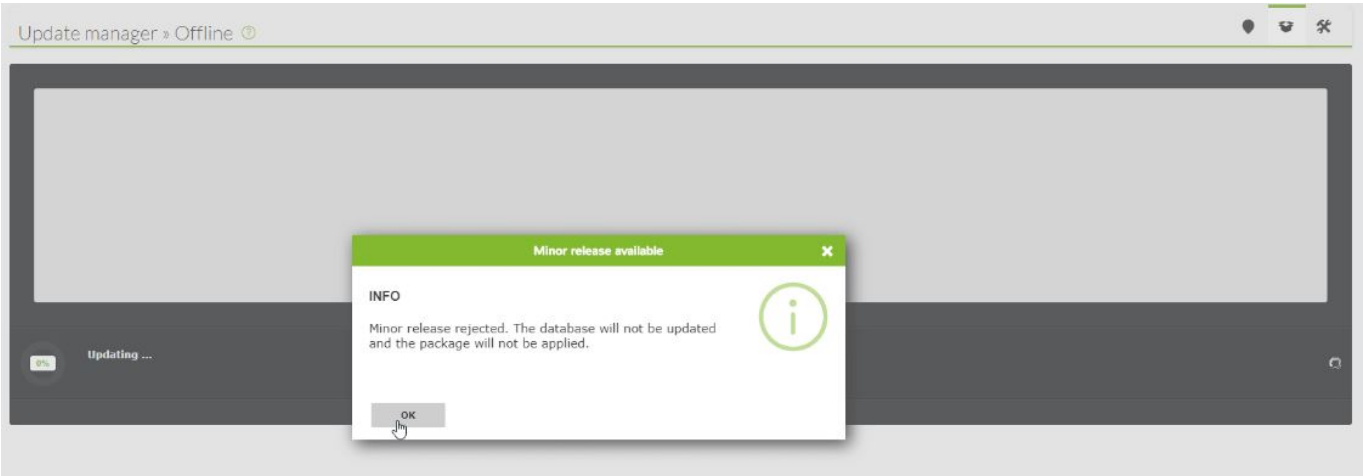
Update

When updating Pandora FMS console in an HA environment, it is important to bear in mind the following points when updating by means of OUM through Update Manager > Update Manager offline.

Enterprise version users can download the OUM package from Pandora FMS support website.

When in a balanced environment with a shared database, updating the first console applies the corresponding changes to the database. This means that when updating the secondary console, Pandora FMS shows an error massage when finding the already entered information in the database. However, the console is still updated.





Database HA

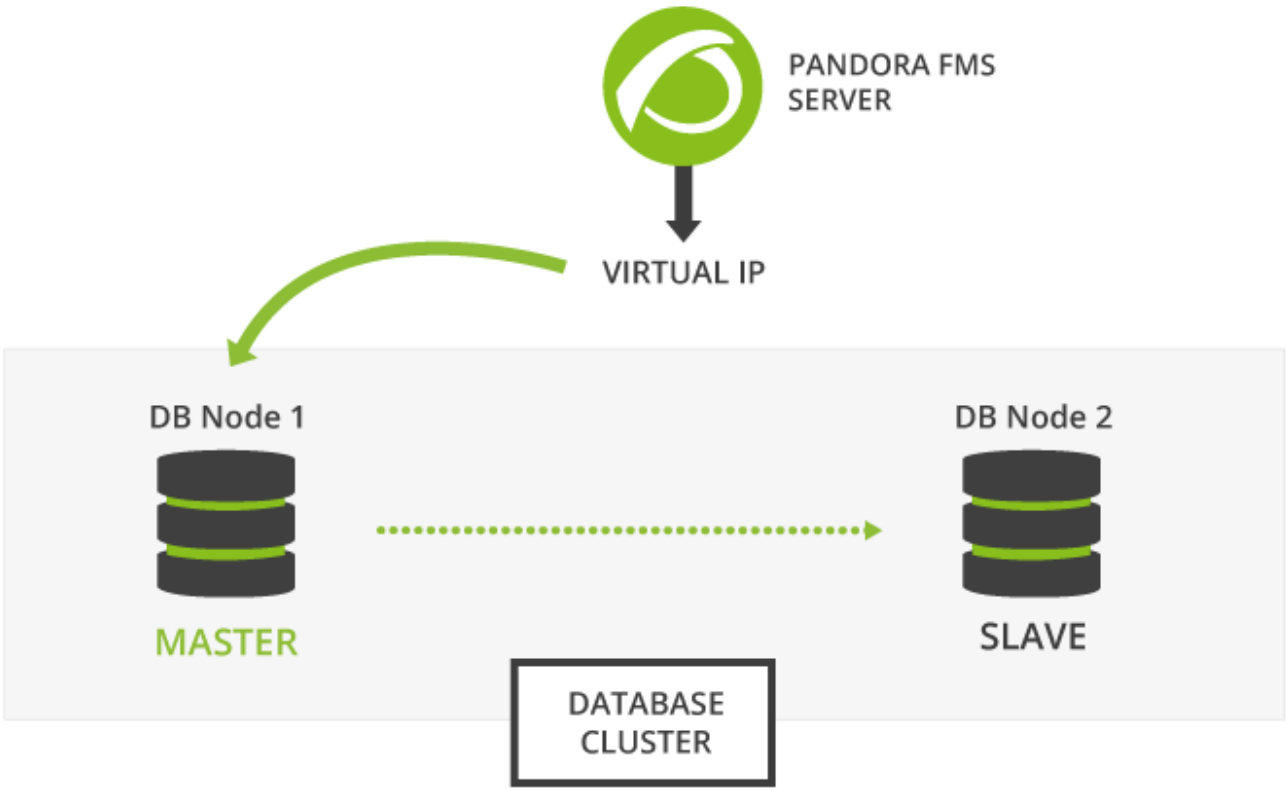
This solution is provided to offer a fully-featured solution for HA in Pandora FMS environments. This is the only officially-supported HA model for Pandora FMS. This solution is provided -preinstalled- since OUM 724. This system replaces DRBD and other HA systems recommended in the past.

This is the first Pandora DB HA implementation, and the installing process is almost fully manual, by using the Linux console as Root. In future versions setup from the GUI will be provided.

Pandora FMS relies on a MySQL database for configuration and data storage. A database failure can temporarily bring your monitoring solution to a halt. The Pandora FMS high-availability database cluster allows to easily deploy a fault-tolerant, robust architecture.

Cluster resources are managed by Pacemaker, an advanced, scalable High-Availability cluster resource manager. Corosync provides a closed process group communication model for creating replicated state machines. Percona was chosen as the default RDBMS for its scalability, availability, security and backup features.

Active/passive replication takes place from a single master node (with writing permissions) to any number of slaves (read only). A virtual IP address always points to the current master. If the master node fails, one of the slaves is promoted to master and the virtual IP address is updated accordingly.



The Pandora FMS Database HA Tool, *pandora_ha*, monitors the cluster and makes sure the Pandora FMS Server is always running, restarting it when needed. *pandora_ha* itself is monitored by systemd.

This is an advanced feature that requires knowledge of Linux systems.

Installation

Configure a two-node cluster, with hosts *node1* and *node2*. Change hostnames, passwords, etc. as needed to match your environment.

Commands that should be run on one node will be preceded by that node's hostname. For example:

```
node1# <command>
```

Commands that should be run on all nodes will be preceded by the word **all**. For example:

```
all# <command>
```

There is an additional host, which will be referred to as **pandorafms**, where Pandora FMS is or will be installed.

When referencing **all** it only refers to the Database nodes, the additional Pandora FMS node will always be referenced as **pandorafms** and is not part of **all**

Prerequisites

CentOS version 7 must be installed on all hosts, and they must be able to resolve each other's hostnames.

```
node1# ping node2
PING node2 (192.168.0.2) 56(84) bytes of data.

node2# ping node1
PING node1 (192.168.0.1) 56(84) bytes of data.

pandorafms# ping node1
PING node1 (192.168.0.1) 56(84) bytes of data.

pandorafms# ping node2
PING node2 (192.168.0.2) 56(84) bytes of data.
```

An Open SSH server must be installed and running on every host. Remove the banner displayed by Open SSH:

```
all# [ -f /etc/cron.hourly/motd_rebuild ] && rm -f
/etc/cron.hourly/motd_rebuild
all# sed -i -e 's/^Banner.*//g' /etc/ssh/sshd_config
all# systemctl restart sshd
```

The Pandora FMS Database HA Tool will not work properly if a banner is configured for Open SSH.

Generate new SSH authentication keys for each host and copy the public key to each of the other hosts:

Passwords can be generated for a non-root user for a later clusted installation with a non-root user.

```
node1# echo -e "\n\n\n" | ssh-keygen -t rsa
node1# ssh-copy-id -p22 root@node2
node1# ssh node2

node2# echo -e "\n\n\n" | ssh-keygen -t rsa
node2# ssh-copy-id -p22 root@node1
node2# ssh node1

pandorafms# echo -e "\n\n\n" | ssh-keygen -t rsa
pandorafms# ssh-copy-id -p22 root@node1
pandorafms# ssh-copy-id -p22 root@node2
pandorafms# ssh node1
pandorafms# ssh node2
```

On the Pandora FMS node, copy the key pair to */usr/share/httpd/.ssh/*. The Pandora FMS Console needs to retrieve the cluster status:

```
pandorafms# cp -r /root/.ssh/ /usr/share/httpd/
pandorafms# chown -R apache:apache /usr/share/httpd/.ssh/
```

The following steps are only necessary if the nodes are running SSH on a non-standard port. Replace *22* with the right port number:

```
all# echo -e "Host node1\n    Port 22" >> /root/.ssh/config
all# echo -e "Host node2\n    Port 22" >> /root/.ssh/config
```

Installing Percona

Install the required packages:

```
 all# yum install
https://repo.percona.com/yum/percona-release-latest.noarch.rpm

 all# yum install -y Percona-Server-server-57 percona-xtrabackup-24
```

Make sure the Percona service is disabled, since it will be managed by the cluster:

```
all# systemctl disable mysqld
```

If the system service is not disabled, the cluster's resource manager will not work properly.

Next, start the Percona server:

```
all# systemctl start mysqld
```

A new temporary password will be generated and linked to /var/log/mysqld.log. Log in the Percona server and change the root password:

```
all# mysql -uroot -p$(grep "temporary password" /var/log/mysqld.log | \
rev | cut -d' ' -f1 | rev)
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Pandor4!');
mysql> UNINSTALL PLUGIN validate_password;
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('pandora');
mysql> quit
```

At the time of carrying out the MySQL configuration, it will be possible to do it through the following autogenerator that is already included in the installation package of Pandora FMS Enterprise server if it is installed with the *–ha* modifier and the default Pandora FMS ISO.

Remember that if you have installed the server package manually, instead of using the ISO, you must pass the *–ha* parameter to access the HA server tools.

In case you have not done it, you should only reinstall the server package with the –ha parameter : In the example environment we have 2 database nodes (node1 and node2) and one application node (Pandorafms), so the pandora server package should only be installed in the application node, if the application node is installed with a Pandora iso (recommended option) this step is not necessary, otherwise the server package should be reinstalled with the –ha flag.

```
pandorafms# ./pandora_server_installer --install --ha
```

Once the server is installed with HA tools enabled, you will find the configuration generator for database replication in the path: */usr/share/pandora_server/util/myconfig_ha_gen.sh*

```
Example: ./myconfig_ha_gen.sh -i serverid [-l file_location] [-d database]
[-b binlog] [-u dbuser] [-p dbpass] [-s poolsize] [-h help]
Mandatory parameters:
      -i --serverid   Set the server id for the database (Mandatory)
Optional parameters:
      -l --location   Set my.cnf custom location including filename. [
default value: /etc/my.cnf ] (optional)
      -d --database   Set the database to be replicated. [ default value:
pandora ] (optional)
      -b --binlog     Set binlog file. [ default value: mysql-bin ]
(optional)
      -u --dbuser     Set dbuser for mysql connection and backups. [ default
value: root ] (optional)
      -p --dbpass     Set dbpassword for mysql connection and backups. [
default value: pandora ] (optional)
      -s --poolsize   Set innodb_buffer_pool_size static size in M
(Megabytes) or G (Gigabytes). [ default value: autocalculated ] (optional)
```

```
      -h --help       Print help.
```

In the current case where the databases are not on the same server as the application, it will be necessary to copy the script to the nodes to be executed locally.

```
 pandorafms# scp /usr/share/pandora_server/util/myconfig_ha_gen.sh
root@node1:/root/
 pandorafms# scp /usr/share/pandora_server/util/myconfig_ha_gen.sh
root@node2:/root/
```

As you see in the example, it will only be necessary to enter the parameter **serverid** (mandatory) in standard environments or deployed with the ISO and some optional parameters for custom environments.

If the default user or the defined one does not connect to the database, the script will end up giving a connection error.

You also have the possibility to enter as arguments the database, the user and the password. Otherwise, the default settings will be used.

In this case we will execute in both nodes the script only passing the server id if we have the default credentials, otherwise define the necessary parameters.

```
 node1# /root/myconfig_ha_gen.sh -i 1
 node2# /root/myconfig_ha_gen.sh -i 2
```

IMPORTANT: each node must have a unique ID.

The Percona configuration file will be written in /etc/my.cnf where the server id and the recommended configuration for database replication will be defined.

Restart the mysqld service to check that the configuration has been correctly applied.

```
all# systemctl restart mysqld
```

Installing Pandora FMS

New Pandora FMS installation

Install Pandora FMS on the newly created database. For more information see:

```
https://wiki.pandorafms.com/index.php?title=Pandora:Documentation_en:Install
ing
```

Stop the Pandora FMS server:

```
pandorafms# /etc/init.d/pandora_server stop
```

From version NG 754 onwards, additional options are available for manual startup and shutdown of High Availability (HA) environments.

Existing Pandora FMS installation

Stop your Pandora FMS Server:

```
pandorafms# /etc/init.d/pandora_server stop
```

Back up the Pandora FMS database:

```
 pandorafms# mysqldump -uroot -ppandora --databases pandora>
/tmp/pandoradb.sql
 pandorafms# scp /tmp/pandoradb.sql node1:/tmp/
```

Load it into the new database:

```
node1# mysql -uroot -ppandora </tmp/pandoradb.sql
```

Replication setup

Grant the required privileges for replication to work on all databases:

```
 all# mysql -uroot -ppandora
 mysql> GRANT ALL ON pandora.* TO 'root'@'%' IDENTIFIED BY 'pandora';
 mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.*
 TO 'root'@'%' IDENTIFIED BY 'pandora';
 mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE, SUPER, PROCESS, RELOAD
ON *.*
 TO 'root'@'localhost' IDENTIFIED BY 'pandora';
 mysql> GRANT select ON mysql.user TO 'root'@'%' IDENTIFIED BY 'pandora';
 mysql> FLUSH PRIVILEGES;
 mysql> quit
```

Back up the database of the first node and write down the master log file name and position (in the example, *mysql-bin.000001* and *785*):

```
node1# [ -e /root/pandoradb.bak ] && rm -rf /root/pandoradb.bak
node1# innobackupex --no-timestamp /root/pandoradb.bak/
node1# innobackupex --apply-log /root/pandoradb.bak/
node1# cat /root/pandoradb.bak/xtrabackup_binlog_info
mysql-bin.000001        785
```

Load the database on the second node and configure it to replicate from the first node (set

MASTER_LOG_FILE and MASTER_LOG_POS to the values found in the previous step):

If Pandora FMS installation was carried out through an **ISO** delete the original database from node 2. Only the database from node 1 will be needed, which will be the one to store information from both machines. That way, balancing will be correctly done.

```
node2# systemctl stop mysqld

node1# rsync -avpP -e ssh /root/pandoradb.bak/ node2:/var/lib/mysql/

node2# chown -R mysql:mysql /var/lib/mysql
node2# chcon -R system_u:object_r:mysqld_db_t:s0 /var/lib/mysql
node2# systemctl start mysqld
node2# mysql -uroot -ppandora
mysql> CHANGE MASTER TO MASTER_HOST='node1',
MASTER_USER='root', MASTER_PASSWORD='pandora',
MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=785;
mysql> START SLAVE;
mysql> SHOW SLAVE STATUS \G
   *************************** 1. row ***************************
                Slave_IO_State: Waiting for master to send event
                   Master_Host: node1
                   Master_User: root
                   Master_Port: 3306
                 Connect_Retry: 60
               Master_Log_File: mysql-bin.000002
           Read_Master_Log_Pos: 785
                Relay_Log_File: node2-relay-bin.000003
                 Relay_Log_Pos: 998
         Relay_Master_Log_File: mysql-bin.000002
              Slave_IO_Running: Yes
             Slave_SQL_Running: Yes
               Replicate_Do_DB: pandora
           Replicate_Ignore_DB:
            Replicate_Do_Table:
        Replicate_Ignore_Table:
       Replicate_Wild_Do_Table:
   Replicate_Wild_Ignore_Table:
                    Last_Errno: 0
                    Last_Error:
                  Skip_Counter: 0
           Exec_Master_Log_Pos: 785
               Relay_Log_Space: 1252
               Until_Condition: None
                Until_Log_File:
                 Until_Log_Pos: 0
```

```
              Master_SSL_Allowed: No
              Master_SSL_CA_File:
              Master_SSL_CA_Path:
                 Master_SSL_Cert:
               Master_SSL_Cipher:
                  Master_SSL_Key:
           Seconds_Behind_Master: 0
   Master_SSL_Verify_Server_Cert: No
                   Last_IO_Errno: 0
                   Last_IO_Error:
                  Last_SQL_Errno: 0
                  Last_SQL_Error:
     Replicate_Ignore_Server_Ids:
                Master_Server_Id: 1
                     Master_UUID: 580d8bb0-6991-11e8-9a22-16efadb2f150
                Master_Info_File: mysql.slave_master_info
                       SQL_Delay: 0
             SQL_Remaining_Delay: NULL
         Slave_SQL_Running_State: Slave has read all relay log; waiting for
more updates
              Master_Retry_Count: 86400
                     Master_Bind:
         Last_IO_Error_Timestamp:
        Last_SQL_Error_Timestamp:
                  Master_SSL_Crl:
              Master_SSL_Crlpath:
              Retrieved_Gtid_Set:
               Executed_Gtid_Set:
                   Auto_Position: 0
            Replicate_Rewrite_DB:
                    Channel_Name:
              Master_TLS_Version:
   1 row in set (0.00 sec)
mysql> QUIT

all# systemctl stop mysqld
```

It is recommended to **Slave_IO_Running** and **Slave_SQL_Running** show **Yes**. Other values may differ from the example.

Make sure **not** use **root** user to perform this process. It is advised to grant permissions to other use in charge of managing the database to avoid possible conflicts.

Configuring the two-node cluster

Install the required packages:

```
all# yum install -y epel-release corosync ntp pacemaker pcs

all# systemctl enable ntpd
all# systemctl enable corosync
all# systemctl enable pcsd
all# cp /etc/corosync/corosync.conf.example /etc/corosync/corosync.conf
```

```
all# systemctl start ntpd
all# systemctl start corosync
all# systemctl start pcsd
```

Stop the Percona server:

```
node1# systemctl stop mysqld
```

```
node2# systemctl stop mysqld
```

Authenticate all nodes in the cluster:

Create and start the cluster:

```
all# echo hapass | passwd hacluster --stdin
```

```
node1# pcs cluster auth -u hacluster -p hapass --force node1 node2
node1# pcs cluster setup --force --name pandoraha node1 node2
node1# pcs cluster start --all
node1# pcs cluster enable --all
node1# pcs property set stonith-enabled=false
node1# pcs property set no-quorum-policy=ignore
```

Check cluster status:

```
node#1 pcs status
   Cluster name: pandoraha
   Stack: corosync
   Current DC: node1 (version 1.1.18-11.el7_5.2-2b07d5c5a9) - partition with
quorum
   Last updated: Fri Jun  8 12:53:49 2018
   Last change: Fri Jun  8 12:53:47 2018 by root via cibadmin on node1
   2 nodes configured
   0 resources configured
   Online: [ node1 node2 ]
   No resources
   Daemon Status:
     corosync: active/disabled
     pacemaker: active/disabled
```

```
     pcsd: active/enabled
```

Both nodes should be online (**Online: [node1 node2]**). Other values may differ from the example.

Install the Percona Pacemaker replication agent (it can be downloaded manually from our library):

```
all# cd /usr/lib/ocf/resource.d/
all# mkdir percona
all# cd percona
all# curl -L -o pacemaker_mysql_replication.zip
https://pandorafms.com/library/wp-content/uploads/2019/12/pacemaker_mysql_re
plication.zip
all# unzip pacemaker_mysql_replication.zip
all# rm -f pacemaker_mysql_replication.zip
all# chmod u+x mysql
```

Configure cluster resources. Replace **<VIRT_IP>** by the virtual IP address of your choice:

If you have changed the default password used in this guide for the database's root user, update *replication_passwd* and *test_passwd* accordingly.

Cluster ressource names must be exactly those indicated in this guide (“pandoraip” and “pandoradb”)

```
node1# export VIP=<VIRT_IP>
node1# pcs resource create pandoradb ocf:percona:mysql config="/etc/my.cnf"
\
pid="/var/run/mysqld/mysqld.pid" socket="/var/lib/mysql/mysql.sock" \
replication_user="root" replication_passwd="pandora" max_slave_lag="60" \
evict_outdated_slaves="false" binary="/usr/sbin/mysqld"
datadir="/var/lib/mysql" \
test_user="root" test_passwd="pandora" op start interval="0" timeout="60s" \
op stop interval="0" timeout="60s" op promote timeout="120" op demote
timeout="120" \
op monitor role="Master" timeout="30" interval="5" on-fail="restart" op
monitor role="Slave" \
timeout="30" interval="10"
node1# pcs resource create pandoraip ocf:heartbeat:IPaddr2 ip=$VIP
cidr_netmask=24 \
```

```
op monitor interval=20s
node1# pcs resource master master_pandoradb pandoradb meta master-max="1" \
master-node-max="1" clone-max="2" clone-node-max="1" notify="true" \
globally-unique="false" target-role="Master" is-managed="true"
node1# pcs constraint colocation add master master_pandoradb with pandoraip
node1# pcs constraint order promote master_pandoradb then start pandoraip
```

Check cluster status:

```
node1# pcs status
   Cluster name: pandoraha
   Stack: corosync
   Current DC: node1 (version 1.1.18-11.el7_5.2-2b07d5c5a9) - partition with
quorum
   Last updated: Fri Jun  8 13:02:21 2018
   Last change: Fri Jun  8 13:02:11 2018 by root via cibadmin on node1
   2 nodes configured
   3 resources configured
   Online: [ node1 node2 ]
   Full list of resources:
    Master/Slave Set: master_pandoradb [pandoradb]
        Masters: [ node1 ]
        Slaves: [ node2 ]
    pandoraip      (ocf::heartbeat:IPaddr2):       Started node1
   Daemon Status:
     corosync: active/disabled
     pacemaker: active/disabled
     pcsd: active/enabled
```

Both nodes should be online (**Online: [node1 node2]**).
Other values may differ from the example.

Configuring the two-node cluster with a non-root user

It will be done similarly to the previous one. The login information must have been copied, which has already been explained, and the following steps must be carried out:

```
# All nodes:
```

```
useradd <usuario>
passwd <usuario>
usermod -a -G haclient <usuario>
```

```
# Enable PCS ACL system
pcs property set enable-acl = true --force
```

```
# Create role
pcs acl role create <rol> description="RW role"  write xpath /cib
```

```
# Create PCS user - Local user
pcs acl user create <usuario> <rol>
```

```
# Login into PCS from ALL nodes
su - <usuario>
pcs status
Username: <usuario>
Password: *****
```

```
# Wait for 'Authorized' message, ignore output. Wait a second and retry 'pcs
status' command
```

Pandora FMS setup

Make sure *php-pecl-ssh2* is installed:

```
pandorafms# yum install php-pecl-ssh2
pandorafms# systemctl restart httpd
```

There are two parameters in */etc/pandora/pandora_server.conf* that control the performance of the Pandora FMS Database HA Tool. Adjust them to suit your needs:

```
# Pandora FMS Database HA Tool execution interval in seconds (PANDORA FMS
ENTERPRISE ONLY).
ha_interval 30
# Pandora FMS Database HA Tool monitoring interval in seconds. Must be a
multiple of ha_interval (PANDORA FMS ENTERPRISE ONLY).
ha_monitoring_interval 60
```

Point your Pandora FMS to the master's virtual IP address (replacing **<IP>** with the virtual IP address):

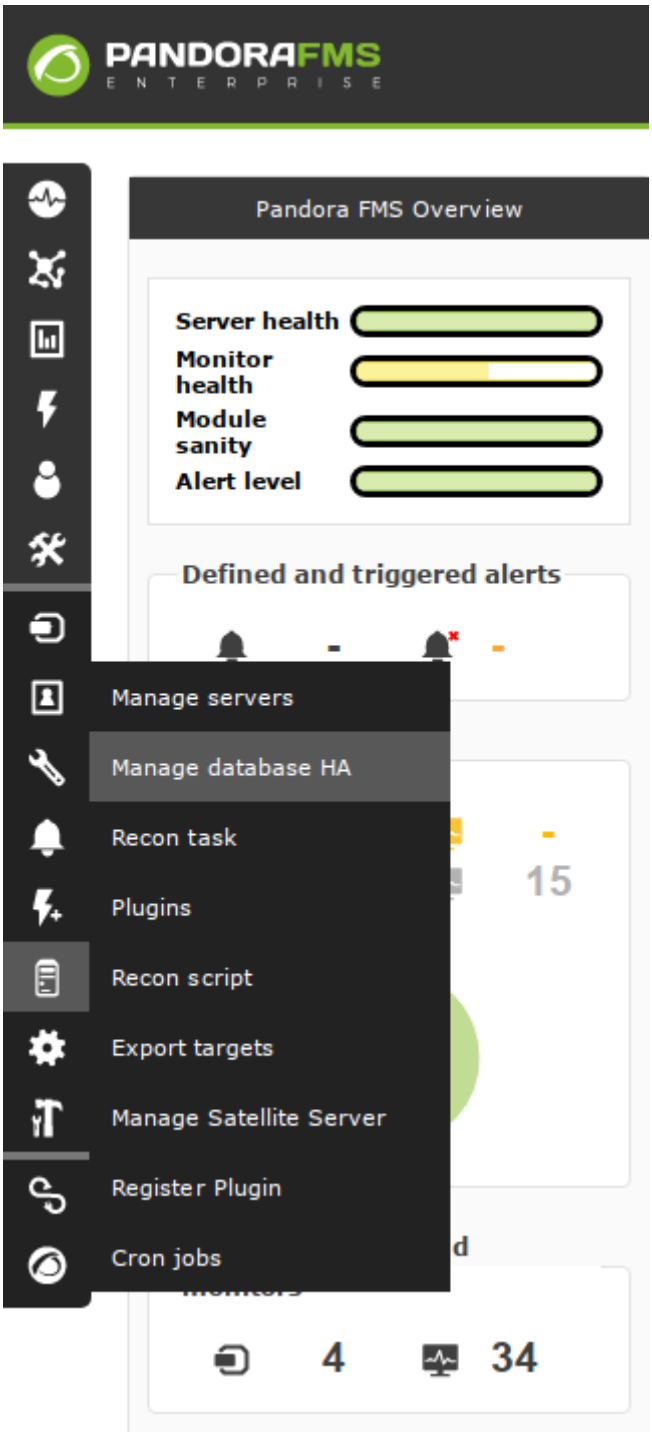
```
pandorafms# export VIRT_IP=<IP>
pandorafms# sed -i -e "s/^dbhost .*/dbhost $VIRT_IP/" \
/etc/pandora/pandora_server.conf
pandorafms# sed -i -e
"s/\$config\[\"dbhost\"\]=\".*\";/\$config[\"dbhost\"]=\"$VIRT_IP\";/" \
/var/www/html/pandora_console/include/config.php
```

Install and start the pandora_ha service:

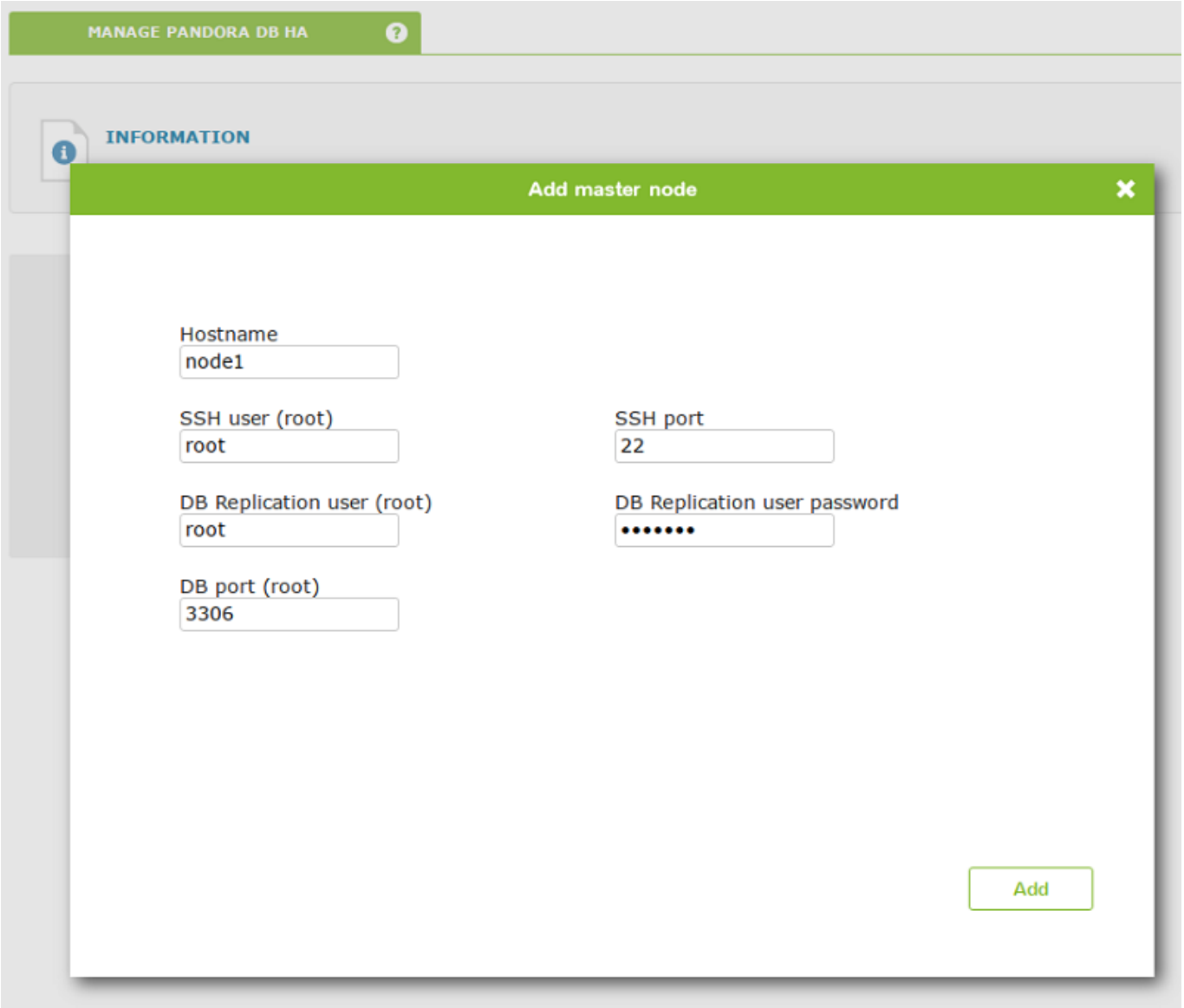
```
pandorafms# cat > /etc/systemd/system/pandora_ha.service <<-EOF
[Unit]
Description=Pandora FMS Database HA Tool
```

```
[Service]
Type=forking

PIDFile=/var/run/pandora_ha.pid
Restart=always
ExecStart=/usr/bin/pandora_ha -d -p /var/run/pandora_ha.pid
/etc/pandora/pandora_server.conf

[Install]
WantedBy=multi-user.target
EOF

pandorafms# systemctl enable pandora_ha
pandorafms# systemctl start pandora_ha
```

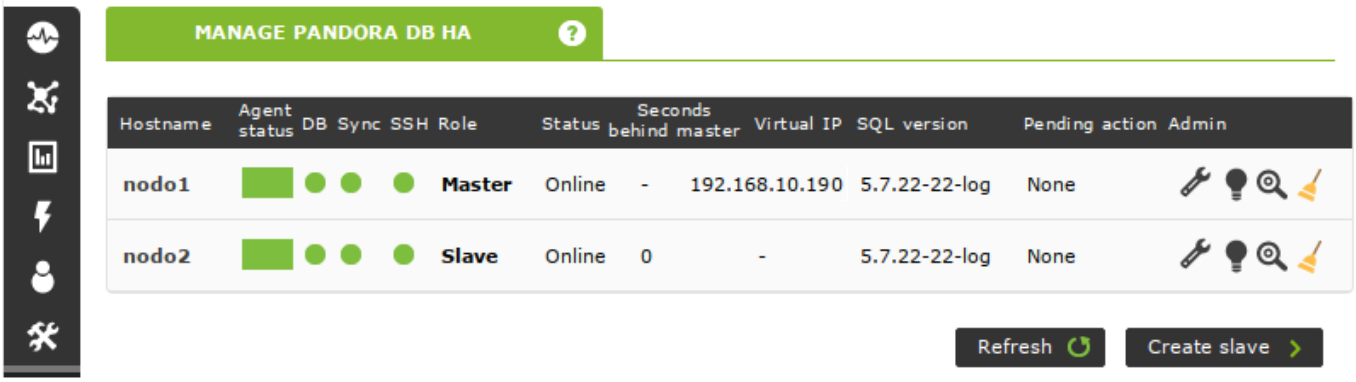
Log in your Pandora FMS Console and go to *Servers* > *Manage database HA*:



Click on *Add new node* and create an entry for the first node:



Next, click on *Create slave* and add an entry for the second node. You should see something similar to this:



It is necessary to check whether the **functions_HA_cluster.php** file contains the $ssh_user correct paths so that it is properly shown and it is possible to correctly interact with the icons.

Seconds behind master should be close to 0. If it keeps increasing, replication is nor working.

Adding a new node to the cluster

Remove the banner displayed by Open SSH:

```
node3# [ -f /etc/cron.hourly/motd_rebuild ] && rm -f
/etc/cron.hourly/motd_rebuild
node3# sed -i -e 's/^Banner.*//g' /etc/ssh/sshd_config
node3# systemctl restart sshd
```

Generate new SSH authentication keys for the new host and copy the public key to each of the other hosts:

Keys can be generated for a non-root user for a cluster installation with a non-root user.

```
node1# ssh-copy-id -p22 root@node3
node1# ssh node3
```

```
node2# ssh-copy-id -p22 root@node3
node2# ssh node3
```

```
pandorafms# ssh-copy-id -p22 root@node3
pandorafms# ssh node3
```

```
node3# echo -e "\n\n\n" | ssh-keygen -t rsa
node3# ssh-copy-id -p22 root@node1
node3# ssh-copy-id -p22 root@node2
node3# ssh node1
node3# ssh node2
```

On the Pandora FMS node, copy the “known_hosts” file for user apache:

```
pandorafms# yes | cp /root/.ssh/known_hosts /usr/share/httpd/.ssh/
```

The following steps are only necessary if the nodes are running SSH on a non-standard port. Replace 22 with the right port number:

```
```

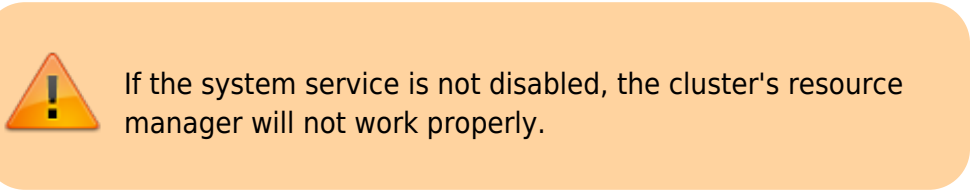
```
 all# echo -e "Host node1\n    Port 22">> /root/.ssh/config
 all# echo -e "Host node2\n    Port 22">> /root/.ssh/config
 all# echo -e "Host node3\n    Port 22">> /root/.ssh/config
```

Install the required packages:

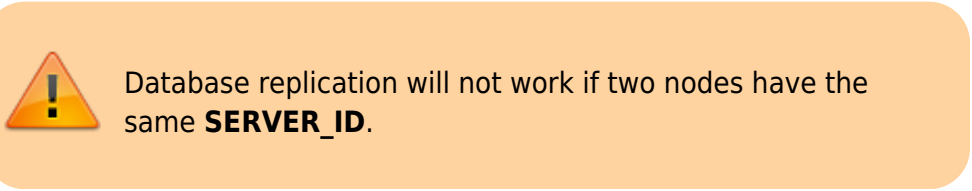
```
node3# yum install -y
http://www.percona.com/downloads/percona-release/redhat/0.1-4/percona-releas
e-0.1-4.noarch.rpm
node3# yum install -y Percona-Server-server-57 percona-xtrabackup-24
```

Make sure the Percona service is disabled, since it will be managed by the cluster:

```
 node3# systemctl stop mysqld
 node3# systemctl disable mysqld
```



Configure Percona, replacing **<ID>** with a number that must be exclusive for each cluster node:




```
node3# export SERVER_ID=<ID>
node3# export POOL_SIZE=$(grep -i total /proc/meminfo | head -1 | \
awk '{print $(NF-1)*0.4/1024}' | sed s/\\..*$/M/g)
node3# cat <<EOF> /etc/my.cnf
[mysqld]
server_id=$SERVER_ID

datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
symbolic-links=0
log-error=/var/log/mysqld.log
show_compatibility_56=on

max_allowed_packet = 64M
# Recommendation: not to assign a value greater than 35% of available RAM
memory
innodb_buffer_pool_size = $POOL_SIZE
innodb_lock_wait_timeout = 90
innodb_file_per_table
innodb_flush_method = O_DIRECT
innodb_log_file_size = 64M
```

```
innodb_log_buffer_size = 16M
thread_cache_size = 8
max_connections = 200
wait_timeout = 900
key_buffer_size=4M
read_buffer_size=128K
read_rnd_buffer_size=128K
sort_buffer_size=128K
join_buffer_size=4M
log-bin=mysql-bin
query_cache_type = 1
query_cache_size = 32M
query_cache_limit = 8M
sql_mode=""
expire_logs_days=3

binlog-format=ROW
log-slave-updates=true
sync-master-info=1
sync_binlog=1
max_binlog_size = 100M
replicate-do-db=pandora
port=3306
report-port=3306
report-host=master
gtid-mode=off
enforce-gtid-consistency=off
master-info-repository=TABLE
relay-log-info-repository=TABLE

sync_relay_log = 10
sync_relay_log_info = 10
slave_compressed_protocol = 1
slave_parallel_type = LOGICAL_CLOCK
slave_parallel_workers = 10
innodb_flush_log_at_trx_commit = 2
innodb_flush_log_at_timeout = 1800

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

[client]
user=root
password=pandora
EOF
```

Start the Percona server:

node3# systemctl start mysqld

A new temporary password will be generated and linked to /var/log/mysqld.log. Log in the Percona server and change the root password:

```
node3# mysql -uroot -p$(grep "temporary password" /var/log/mysqld.log | \
rev | cut -d' ' -f1 | rev)
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Pandor4!');
mysql> UNINSTALL PLUGIN validate_password;
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('pandora');
mysql> quit
```

Back up the database of the first node and write down the master log file name and position (in the example, mysql-bin.000001 and 785):

```
node1# [ -e /root/pandoradb.bak ] && rm -rf /root/pandoradb.bak
node1# innobackupex --no-timestamp /root/pandoradb.bak/
node1# innobackupex --apply-log /root/pandoradb.bak/
node1# cat /root/pandoradb.bak/xtrabackup_binlog_info
mysql-bin.000001        785
```

Load the database on the second node and configure it to replicate from the first node (set MASTER_LOG_FILE and MASTER_LOG_POS to the values found in the previous step):

```
node3# systemctl stop mysqld

node1# rsync -avpP -e ssh /root/pandoradb.bak/ node3:/var/lib/mysql/

node3# chown -R mysql:mysql /var/lib/mysql
node3# chcon -R system_u:object_r:mysqld_db_t:s0 /var/lib/mysql
node3# systemctl start mysqld
node3# mysql -uroot -ppandora
mysql> CHANGE MASTER TO MASTER_HOST='node1',
MASTER_USER='root', MASTER_PASSWORD='pandora',
MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=785;
mysql> START SLAVE;
mysql> SHOW SLAVE STATUS \G
   *************************** 1. row ***************************
                Slave_IO_State: Waiting for master to send event
                   Master_Host: node1
                   Master_User: root
                   Master_Port: 3306
                 Connect_Retry: 60
               Master_Log_File: mysql-bin.000002
           Read_Master_Log_Pos: 785
                Relay_Log_File: node3-relay-bin.000003
                 Relay_Log_Pos: 998
         Relay_Master_Log_File: mysql-bin.000002
              Slave_IO_Running: Yes
             Slave_SQL_Running: Yes
               Replicate_Do_DB: pandora
```

```
              Replicate_Ignore_DB:
               Replicate_Do_Table:
           Replicate_Ignore_Table:
          Replicate_Wild_Do_Table:
      Replicate_Wild_Ignore_Table:
                       Last_Errno: 0
                       Last_Error:
                     Skip_Counter: 0
              Exec_Master_Log_Pos: 785
                  Relay_Log_Space: 1252
                  Until_Condition: None
                   Until_Log_File:
                    Until_Log_Pos: 0
               Master_SSL_Allowed: No
               Master_SSL_CA_File:
               Master_SSL_CA_Path:
                  Master_SSL_Cert:
                Master_SSL_Cipher:
                   Master_SSL_Key:
            Seconds_Behind_Master: 0
    Master_SSL_Verify_Server_Cert: No
                    Last_IO_Errno: 0
                    Last_IO_Error:
                   Last_SQL_Errno: 0
                   Last_SQL_Error:
      Replicate_Ignore_Server_Ids:
                 Master_Server_Id: 1
                      Master_UUID: 580d8bb0-6991-11e8-9a22-16efadb2f150
                 Master_Info_File: mysql.slave_master_info
                        SQL_Delay: 0
              SQL_Remaining_Delay: NULL
          Slave_SQL_Running_State: Slave has read all relay log; waiting for
more updates
                 Master_Retry_Count: 86400
                        Master_Bind:
          Last_IO_Error_Timestamp:
         Last_SQL_Error_Timestamp:
                   Master_SSL_Crl:
               Master_SSL_Crlpath:
               Retrieved_Gtid_Set:
                Executed_Gtid_Set:
                    Auto_Position: 0
             Replicate_Rewrite_DB:
                     Channel_Name:
               Master_TLS_Version:
    1 row in set (0.00 sec)
mysql> QUIT

node3# systemctl stop mysqld
```

It is recommended to **Slave_IO_Running** and **Slave_SQL_Running** show **Yes**. Other values may differ from the example.

Install the required packages:

```
 node3# yum install -y epel-release corosync ntp pacemaker pcs

 node3# systemctl enable ntpd
 node3# systemctl enable corosync
 node3# systemctl enable pcsd

 node3# systemctl start ntpd
```

Add the new node to the cluster:

```
 node3# echo -n hapass | passwd hacluster --stdin
 node3# cd /usr/lib/ocf/resource.d/
 node3# mkdir percona
 node3# cd percona
 node3# curl -L -o pacemaker_mysql_replication.zip
https://pandorafms.com/library/wp-content/uploads/2019/12/pacemaker_mysql_re
plication.zip
 node3# unzip pacemaker_mysql_replication.zip
 node3# rm -f pacemaker_mysql_replication.zip
 node3# chmod u+x mysql
```

```
 node1# pcs cluster auth -u hacluster -p hapass --force node3
 node1# pcs cluster node add --enable --start node3
```

Set clone-max to the number of nodes in your cluster (3 in this example):

```
node3# pcs resource update master_pandoradb meta master-max="1" \
master-node-max="1" clone-max="3" clone-node-max="1" notify="true" \
globally-unique="false" target-role="Master" is-managed="true"
```

Check node status:

```
node3# pcs status
Cluster name: pandoraha
Stack: corosync
Current DC: node1 (version 1.1.18-11.el7_5.2-2b07d5c5a9) - partition with
quorum
Last updated: Fri Jun  1 10:55:47 2018
Last change: Fri Jun  1 10:55:09 2018 by root via crm_attribute on node3

3 nodes configured
```

```
3 resources configured

Online: [ node1 node2 node3 ]

Full list of resources:

pandoraip      (ocf::heartbeat:IPaddr2):       Started node1
Master/Slave Set: master_pandoradb [pandoradb]
    Masters: [ node1 ]
    Slaves: [ node2 node3 ]

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
```

All nodes should be online (**Online: [node1 node2 node3]**). Other values may differ from the example.

Register the cluster node in the Pandora console from the “Servers → Manage database HA” menu.

Fixing a broken node

node2 shall be used as example. Set node2 into standby mode:

```
node2# pcs node standby node2
node2# pcs status
    Cluster name: pandoraha
    Stack: corosync
    Current DC: node1 (version 1.1.18-11.el7_5.2-2b07d5c5a9) - partition
with quorum
    Last updated: Tue Jun 12 08:20:49 2018
    Last change: Tue Jun 12 08:20:34 2018 by root via cibadmin on node2
    2 nodes configured
    3 resources configured
    Node node2: standby
    Online: [ node1 ]
    Full list of resources:
     Master/Slave Set: master_pandoradb [pandoradb]
         Masters: [ node1 ]
         Stopped: [ node2 ]
     pandoraip      (ocf::heartbeat:IPaddr2):       Started node1
    Daemon Status:
      corosync: active/enabled
      pacemaker: active/enabled
```

```
      pcsd: active/enabled
```

node2 should be on standby (**Node node2: standby**). Other values may differ from the example.

Back up Percona's data directory:

```
node2# systemctl stop mysqld
node2# [ -e /var/lib/mysql.bak ] && rm -rf /var/lib/mysql.bak
node2# mv /var/lib/mysql /var/lib/mysql.bak
```

Back up the database of the master node (node1 in this example) and update the master node name and master log file name and position in the cluster (in the example, mysql-bin.000001 and 785):

```
node1# [ -e /root/pandoradb.bak ] && rm -rf /root/pandoradb.bak
node1# innobackupex --no-timestamp /root/pandoradb.bak/
node1# innobackupex --apply-log /root/pandoradb.bak/
node1# binlog_info=$(cat /root/pandoradb.bak/xtrabackup_binlog_info)
node1# crm_attribute --type crm_config --name pandoradb_REPL_INFO -s
mysql_replication \
-v "node1|$(echo $binlog_info | awk '{print $1}')|$(echo $binlog_info | awk
'{print $2}')"
```

Load the database of the broken node:

```
 node1# rsync -avpP -e ssh /root/pandoradb.bak/ node2:/var/lib/mysql/

 node2# chown -R mysql:mysql /var/lib/mysql
 node2# chcon -R system_u:object_r:mysqld_db_t:s0 /var/lib/mysql
```

Disable node2 standby mode:

```
 node2# pcs node unstandby node2
 node2# pcs resource cleanup --node node2
```

Check cluster status:

```
node2# pcs status
Cluster name: pandoraha
Stack: corosync
Current DC: node1 (version 1.1.18-11.el7_5.2-2b07d5c5a9) - partition with
quorum
Last updated: Fri Jun  1 10:55:47 2018
Last change: Fri Jun  1 10:55:09 2018 by root via crm_attribute on node3

2 nodes configured
3 resources configured
```

```
Online: [ node1 node2 ]

Full list of resources:

pandoraip      (ocf::heartbeat:IPaddr2):       Started node1
Master/Slave Set: master_pandoradb [pandoradb]
    Masters: [ node1 ]
    Slaves: [ node2 ]

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
```

Both nodes should be online (**Online: [|node1 node2]**).
Other values may differ from those of the example.

Check database replication status:

```
node2# mysql -uroot -ppandora
 mysql> SHOW SLAVE STATUS \G
   *************************** 1. row ***************************
                Slave_IO_State: Waiting for master to send event
                   Master_Host: node1
                   Master_User: root
                   Master_Port: 3306
                 Connect_Retry: 60
               Master_Log_File: mysql-bin.000002
           Read_Master_Log_Pos: 785
                Relay_Log_File: node2-relay-bin.000003
                 Relay_Log_Pos: 998
         Relay_Master_Log_File: mysql-bin.000002
              Slave_IO_Running: Yes
             Slave_SQL_Running: Yes
               Replicate_Do_DB: pandora
           Replicate_Ignore_DB:
            Replicate_Do_Table:
        Replicate_Ignore_Table:
       Replicate_Wild_Do_Table:
   Replicate_Wild_Ignore_Table:
                    Last_Errno: 0
                    Last_Error:
                  Skip_Counter: 0
           Exec_Master_Log_Pos: 785
               Relay_Log_Space: 1252
               Until_Condition: None
                Until_Log_File:
```

```
                  Until_Log_Pos: 0
              Master_SSL_Allowed: No
              Master_SSL_CA_File:
              Master_SSL_CA_Path:
                 Master_SSL_Cert:
               Master_SSL_Cipher:
                  Master_SSL_Key:
           Seconds_Behind_Master: 0
   Master_SSL_Verify_Server_Cert: No
                   Last_IO_Errno: 0
                   Last_IO_Error:
                  Last_SQL_Errno: 0
                  Last_SQL_Error:
     Replicate_Ignore_Server_Ids:
                Master_Server_Id: 1
                     Master_UUID: 580d8bb0-6991-11e8-9a22-16efadb2f150
                Master_Info_File: mysql.slave_master_info
                       SQL_Delay: 0
             SQL_Remaining_Delay: NULL
         Slave_SQL_Running_State: Slave has read all relay log; waiting for
more updates
              Master_Retry_Count: 86400
                     Master_Bind:
         Last_IO_Error_Timestamp:
        Last_SQL_Error_Timestamp:
                  Master_SSL_Crl:
              Master_SSL_Crlpath:
              Retrieved_Gtid_Set:
               Executed_Gtid_Set:
                   Auto_Position: 0
            Replicate_Rewrite_DB:
                    Channel_Name:
              Master_TLS_Version:
   1 row in set (0.00 sec)
```

Make sure **Slave_IO_Running** and **Slave_SQL_Running** show **Yes**. Other values may differ from the example.

Troubleshooting

What do I do if one of the cluster nodes does not work?

Hostname	Agent status	DB	Sync	SSH	Role	Status	Seconds behind master	Virtual IP	SQL version	Pending action	Admin
nodo1					Master	Online	-	192.168.10.190	5.7.22-22-log	None	
nodo2					-	Online	-	-	-	None	

The service will not compromised as long as the master node is running. If the master node fails, a slave node will be automatically promoted to master. See Fixing a broken node.

Go back to Pandora FMS documentation index

1)

https://aws.amazon.com/ec2/instance-types/t2/?nc1=h_ls