PANDORAFMS

High Availability (HA)

High Availability (HA)

Introduction

In critical and/or high-loaded environments, it is possible that the load needs to be distributed among several machines and to be sure that if any Pandora FMS component fails, the system remains online.

Pandora FMS has been designed to be modular but it is also designed to work in collaboration with other components and to be able to assume the load of those components that failed.

Obviously, agents are not redundant. The solution is to redundate critical systems -regardless of whether they have Pandora FMS agents running or not- and thus redundate the monitoring of these systems.

High Availability (HA) can be used in several scenarios:

- Data server.
- Network Servers, WMI, Plugin, Web, Prediction, Recon, and similar.
- Database (BBDD).
- Pandora FMS web console.

HA of Data Server

- For Pandora FMS data server you need to build two machines with a Pandora FMS data server configured (and different hostname and server name).
- A Tentacle server must be configured in each of them.
- Each machine will have a different IP address.
- If you are going to use an external balancer, it will provide a single IP address to which the agents will connect to send their data.
- In the case of using the HA mechanism of the agents, there will be a small delay in sending data, since at each agent execution, it will try to connect with the primary server, and if it does not answer, it will try to connect with the secondary server (if it was configured this way).

If two data servers are to be used and both handle remote policies, collections, and configurations, key directories must be shared so that all instances of the data server can read and write to these directories. Consoles must also have access to these shared directories.

- `/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 Software Agents

From Software Agents, it is possible to perform data server balancing, since it is possible to configure a master (main) and a backup (operational backup) data server. In the Software Agent configuration file `pandora_agent.conf` it should be configured and uncommented the following part of the agent configuration file:

- secondary_server_ip: IP address of the secondary server.
- secondary_server_path: Path where the XML is copied to the secondary server, typically `/var/spool/pandora/data_in` .
- secondary_server_port: Port through which the XML will be copied to the secondary server, in Tentacle 41121, in SSH port 22 and in FTP 21.
- secondary_transfer_mode: Transfer mode to be used to copy the XML to the secondary server, Tentacle, SSH, FTP.
- secondary_server_pwd: Password option for FTP transfer.
- secondary_server_ssl: You may set `yes` or `no` depending on whether you want to use SSL to transfer data through Tentacle.
- secondary_server_opts: Other options required for the transfer will be entered in this field.
- secondary_mode: Mode in which the secondary server must be. It may have two values:

1. on_error: It sends data to the secondary server only if it cannot send it to the primary server.
2. always: It always sends data to the secondary server, regardless of whether or not it can contact the primary server.

Only the remote configuration of the Software Agent is operative, if enabled, on the main server.

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

You need to install multiple servers: network server, WMI server, Plugin server, Web server or prediction server, on several machines on the network (all with the same visibility to the systems you want to monitor) and all on the same segment (so that network latency data is consistent).

Servers can be marked as primary. These servers will automatically collect the data of all the modules assigned to a server that is checked as “down”. Pandora FMS servers themselves implement a mechanism to detect that one of them is down through a verification of its last contact date (`server threshold x 2`). It is enough that there is only one Pandora FMS server active for it to be able to detect that the rest are down.

The obvious way to implement HA and load balancing in a two-node system is to assign 50% of the

modules to each server and check both servers as primary (Master). In the case of having more than two main servers and a third server down with modules pending execution, the first of the main servers that runs the module “self-assigns” the module of the downed server. In case of recovery of one of the downed servers, the modules that were assigned to the primary server are automatically reassigned.

Server configuration

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

- Master mode (main mode or MASTER mode).
- Non-master mode.

If a server goes “down” (is offline), its modules will be run by the master server so that no data is lost.

At any given time, there can only be one master server which is chosen among the servers that have the master configuration option in `/etc/pandora/pandora_server.conf` with a value higher than 0:

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

If the current master server goes down, a new master is chosen. If there is more than one candidate, the one with the highest master value is chosen.

Be careful when disabling servers. If a server with network modules goes down and the Network Server of the master server is disabled, those modules will not run.

Inside `pandora_server.conf`, the following parameters were entered:

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

Add PFMS servers to an HA DB cluster

If High Availability is enabled in the Database, some extra configurations are needed to connect more Pandora FMS servers to MySQL cluster. In the `pandora_server.conf` file (located by default at `/etc/pandora`), for each of the independent Pandora FMS servers to be added, the following parameters must be configured:

- `dbuser`: You must have the username to access MySQL cluster. For example:

```
dbuser pandora
```

- `dbpass`: It must contain the user password to access the MySQL cluster. For example:

```
dbpass pandora
```

- `ha_hosts`: The `ha_host` parameter must be configured followed by the IP addresses or FQDN of the MySQL servers that make up the HA environment. For instance:

```
ha_hosts 192.168.80.170,192.168.80.172
```

High availability of Pandora FMS console

Just install another console pointing to the database. Any console can be used simultaneously from different locations by different users. A web load balancer can be used before the consoles, in case horizontal growth is needed for console load sharing. The session system is managed through cookies and these are 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 input data directory (default: `/var/spool/pandora/data_in`) for remote configuration of agents, collections and other directories (see topic "Security architecture").

It is important to only share subdirectories within `data_in` and not `data_in`, itself as it will negatively affect server performance.

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 Management → Warp update → Update offline. The OUM package may be downloaded 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.

High Availability Database

The main goal of this section is to offer a complete solution for HA in Pandora FMS environments. This is the only HA model with official support for Pandora FMS, and it is provided from version 770 onwards. This system replaces cluster configuration with Corosync and Pacemaker from previous versions.

The new Pandora FMS HA solution is integrated into the product (inside the pandora_ha binary). It implements an HA that supports geographically isolated sites, with different IP ranges, which is not possible with Corosync/Pacemaker.

In the new HA model, the usual setup is in pairs of two, so the design does not implement a quorum system and simplifies configuration and the necessary resources. That way the monitoring system will work as long as there is a DB node available and in case there is a DB Split-Brain, the system will work in parallel until both nodes are merged again.

Starting with version 772, the HA changed to make it simpler and less buggy. For this new HA, it is recommended to use SSD disks for a higher writing/reading speed (IOPS), minimum 500 Mb/s (or more, depending on the environment). Latency between servers must also be taken into account since with very high latency it is difficult to synchronize both databases in time.

The new proposal seeks to solve the current three problems:

1. Complexity and maintainability of the current system (up to version NG 770).
2. Possibility of having an HA environment spread over different geographical locations with non-local network segmentation.
3. Data recovery procedure in case of Split-Brain and secured system operation in case of communication breakdown between the two geographically separated sites.

The new HA system for DB is implemented on Percona 8, although in future versions we will detail how to do so also on MySQL/MariaDB 8.

Pandora FMS is based on a MySQL database to configure and store data, so a failure in the database can temporarily paralyze the monitoring tool. Pandora FMS high availability database cluster allows to easily deploy a strong and fail-safe architecture.

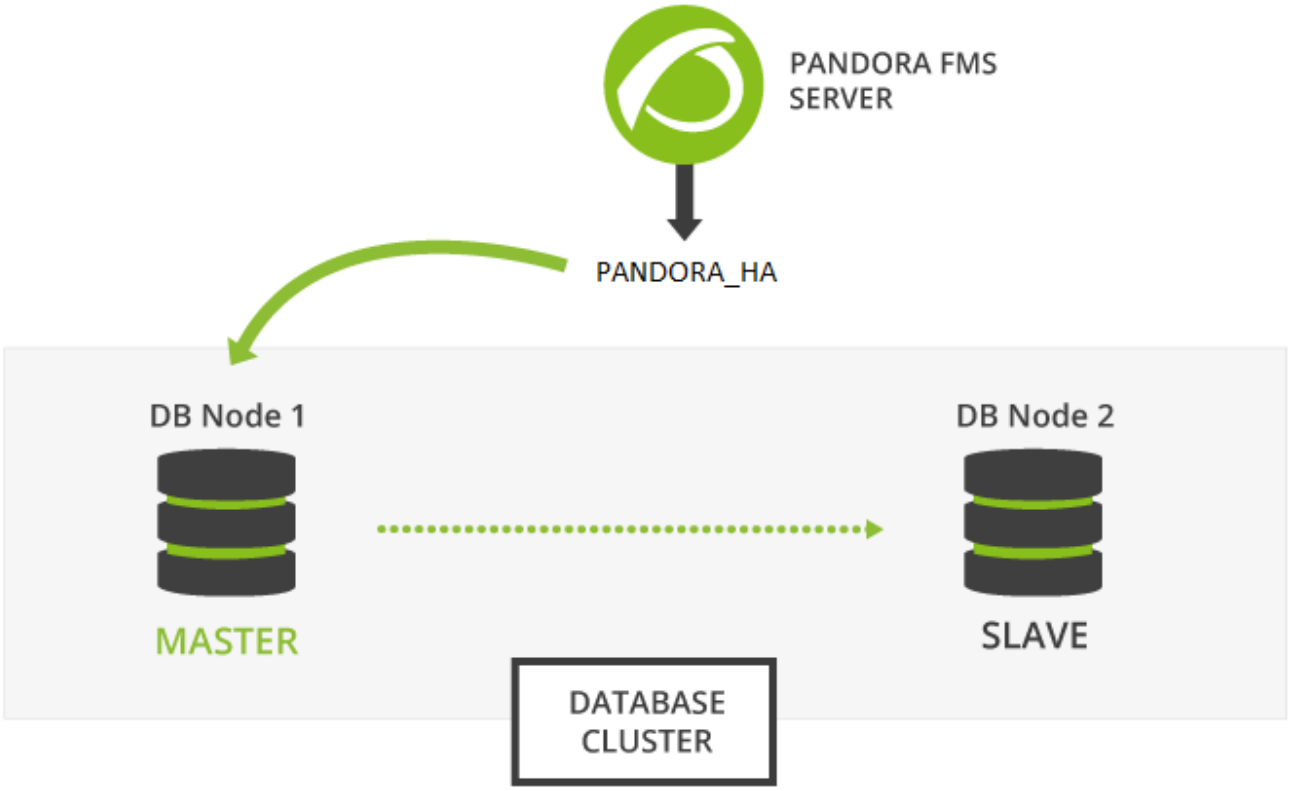
This is an advanced feature that requires knowledge in GNU/Linux systems. It is important that all servers have the time synchronized with an NTP server (chronyd service in Rocky Linux 8).

Binary replication MySQL cluster nodes are managed with the `pandora_ha` binary, starting with version 770. 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 secondaries (read-only). If the master node fails, one of the secondaries is upgraded to master and `pandora_ha` takes care of updating the IP address of the master node.

The environment will consist of the following elements:

- MySQL8 servers with binary replication enabled (Active - Passive).
- Server with `pandora_ha` with the configuration of all MySQL servers to carry out ongoing monitoring and perform the slave-master and master-slave promotions necessary for the correct operation of the cluster.



Installation of Percona 8

Version 770 or later.

Percona 8 Installation for RHEL 8 and Rocky Linux 8

First of all, it is necessary to have the Percona repository installed in all the nodes of the environment in order to be able to install the Percona server packages later on.

Open a terminal window with root rights or as root user. You are solely responsible for this key. The following instructions indicate whether you should run instructions on all devices, on some devices or on one device in particular, please pay attention to the statements.

Execute on all devices involved:

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

Activate version 8 of the Percona repository on all devices:

```
percona-release setup ps80
```

Install the Percona server next to the backup tool with which backups are to be performed for manual synchronization of both environments. Run on all devices involved:

```
yum install percona-server-server percona-xtrabackup-80
```

In case you install the Percona server together with the Web Console and the PFMS server, you will be able to use the deploy indicating the MySQL 8 version by means of the MYVER=80 parameter:

```
curl -Ls https://pfms.me/deploy-pandora-el8 | env MYVER=80 bash
```

Installing Percona 8 on Ubuntu Server

Install Percona repository version 8 on all devices:

```
curl -O https://repo.percona.com/apt/percona-release_latest.generic_all.deb
apt install -y gnupg2 lsb-release ./percona-release_latest.generic_all.deb
```

Activate Percona repository version 8 on all devices:

```
percona-release setup ps80
```

Install the Percona server next to the backup tool with which backups are to be performed for manual synchronization of both environments. On all devices run:

```
apt install -y percona-server-server percona-xtrabackup-80
```

Binary replication configuration

It is recommended that to store the *binlogs* generated by replication in an additional partition or external disk, its size must be the same as the one reserved for the database. See also the token `log-bin` and `log-bin-index`.

Once you have installed MySQL server in all the cluster nodes, proceed to configure both environments to have them replicated.

First of all, configure the configuration file `my.cnf` preparing it for the binary replication to work correctly.

Node 1

Node 1 `/etc/my.cnf` (`/etc/mysql/my.cnf` for Ubuntu server):

```
[mysqld]
server_id=1 # It is important that it is different in all nodes.
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
# OPTIMIZATION FOR PANDORA FMS
innodb_buffer_pool_size = 4096M
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
key_buffer_size=4M
read_buffer_size=128K
read_rnd_buffer_size=128K
sort_buffer_size=128K
join_buffer_size=4M
sql_mode=""
# SPECIFIC PARAMETERS FOR BINARY REPLICATION
binlog-do-db=pandora
replicate-do-db=pandora
max_binlog_size = 100M
```

```
binlog-format=ROW
binlog_expire_logs_seconds=172800 # 2 DAYS
#log-bin=/path                     # Uncomment for adding an additional storage
path for binlogs and setting a new path
#log-bin-index=/path/archive.index # Uncomment for adding an additional storage
path for binlogs and setting a new path
sync_source_info=1
sync_binlog=1
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 = 0
replica_compressed_protocol = 1
replica_parallel_workers = 1
innodb_flush_log_at_trx_commit = 2
innodb_flush_log_at_timeout = 1800

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

- The tokens after the OPTIMIZATION FOR PANDORA FMS comment perform the optimized configuration for Pandora FMS.
- After the comment SPECIFIC PARAMETERS FOR BINARY REPLICATION the specific parameters for the binary replication are configured.
- The token called `binlog_expire_logs_seconds` is configured for a period of two days.
- In the `[client]` subsection, enter the user and password used for the database, by default when installing PFMS, they are `root` and `pandora` respectively. *These values are necessary to perform the backups without indicating user (automated)*.

It is important for the `server_id` token to be different in all nodes, in this example for node 1 this number is used.

Node 2

Node 2 `/etc/my.cnf` (`/etc/mysql/my.cnf` for Ubuntu server):

```
[mysqld]
server_id=2 # It is important that it is different in all nodes.
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
# OPTIMIZATION FOR PANDORA FMS
```

```
innodb_buffer_pool_size = 4096M
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
key_buffer_size=4M
read_buffer_size=128K
read_rnd_buffer_size=128K
sort_buffer_size=128K
join_buffer_size=4M
sql_mode=""
# SPECIFIC PARAMETERS FOR BINARY REPLICATION
binlog-do-db=pandora
replicate-do-db=pandora
max_binlog_size = 100M
binlog-format=ROW
binlog_expire_logs_seconds=172800 # 2 DAYS
#log-bin=/path                     # Uncomment for adding an additional storage
path for binlogs and setting a new path
#log-bin-index=/path/archive.index # Uncomment for adding an additional storage
path for binlogs and setting a new path
sync_source_info=1
sync_binlog=1
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 = 0
replica_compressed_protocol = 1
replica_parallel_workers = 1
innodb_flush_log_at_trx_commit = 2
innodb_flush_log_at_timeout = 1800

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

- The tokens after the OPTIMIZATION FOR PANDORA FMS comment perform the optimized configuration for Pandora FMS.
- After the comment SPECIFIC PARAMETERS FOR BINARY REPLICATION, the specific parameters for the binary replication are configured.
- The token called `binlog_expire_logs_seconds` is configured for a period of two days.
- In the `[client]` subsection, enter the user and password used for the database, by default when installing PFMS they are `root` and `pandora` respectively. *These values are necessary to perform the backups without indicating a user (automated)*.

It is important for the `server_id` token to be different in all nodes, in this example for node 2 this number is used.

Master node configuration

Once you have the correct configuration on both nodes, start the configuration of the node *that will take the role of master server*.

1.- Start `mysqld` service:

```
systemctl start mysqld
```

2.- Access with the temporary root password that will have been generated in the log, the file `/var/log/mysqld.log`:

```
grep "temporary password" /var/log/mysqld.log
```

With the password that appears, access MySQL server:

```
mysql -u root -p
```

`Password:` → Enter the password seen with the grep command.

3.- Change the temporary password to `pandora` of the root user. Remember that the mysql> prompt corresponds to the MySQL command interpreter (MYSQL CLI):

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Pandor4!';

mysql> UNINSTALL COMPONENT 'file://component_validate_password';

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'pandora';
```

4.- Create the binary replication user and root user for remote connections and cluster management:

```
mysql> CREATE USER slaveuser@'%' IDENTIFIED WITH mysql_native_password BY 'pandora';

mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE on *.* to slaveuser@'%';

mysql> CREATE USER root@'%' IDENTIFIED WITH mysql_native_password BY
```

'pandora';

mysql> GRANT ALL PRIVILEGES ON *.* to root@'%';

5.- Create Pandora FMS database:

mysql> create database pandora;

mysql> use pandora;

mysql> source /var/www/html/pandora_console/pandoradb.sql

mysql> source /var/www/html/pandora_console/pandoradb_data.sql

For the source command: As long as Pandora FMS console is installed in the same server, *otherwise send this file to the master server*.

6.- Create the pandora user and give the access privileges to this user:

mysql> CREATE USER pandora@'%' IDENTIFIED WITH mysql_native_password BY 'pandora';

mysql> grant all privileges on pandora.* to pandora@'%';

At this point you have the master server ready to start replicating Pandora FMS database.

Database cloning

The next step is to make a clone of the master database (MASTER) in the slave node (SLAVE). To do so, follow the steps below:

1.- Make a complete download (dump) of the MASTER database:

MASTER #

```
xtrabackup --backup --target-dir=/root/pandoradb.bak/
```

MASTER #

```
xtrabackup --prepare --target-dir=/root/pandoradb.bak/
```

2.- Get the position of the backup binary log:

MASTER # `cat /root/pandoradb.bak/xtrabackup_binlog_info`

It will return something like the following:

```
binlog.000003 157
```

Take note of these two values as they are needed for point number 6.

3.- Make a copy using rsync with the SLAVE server to send the backup:

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

4.- On the SLAVE server, configure the permissions so that MySQL server may access the files sent without any issues:

SLAVE # `chown -R mysql:mysql /var/lib/mysql`

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

5.- Start `mysqld` service on the SLAVE server:

```
systemctl start mysqld
```

6.- Start the SLAVE mode on this server (use the data noted in point 2):

SLAVE # `mysql -u root -ppandora`

SLAVE # mysql> `reset slave all;`

SLAVE # mysql>

```
CHANGE MASTER TO MASTER_HOST='nodo1',
 MASTER_USER='slaveuser',
 MASTER_PASSWORD='pandora',
 MASTER_LOG_FILE='binlog.000003',
 MASTER_LOG_POS=157;
```

SLAVE # mysql> `start slave;`

SLAVE # mysql> `SET GLOBAL read_only=1;`

Once you are done with all these steps, if you run the `show slave status` command inside MySQL shell you will notice that the node is set as slave. If it was configured correctly, you should see an output like the following:

```
*************************** 1. row ***************************
Slave_IO_State: Waiting for source to send event
Master_Host: node1
Master_User: root
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: binlog.000018
Read_Master_Log_Pos: 1135140
Relay_Log_File: relay-bin.000002
Relay_Log_Pos: 1135306
Relay_Master_Log_File: binlog.000018
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: 1135140
Relay_Log_Space: 1135519
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: fa99f1d6-b76a-11ed-9bc1-000c29cbc108
Master_Info_File: mysql.slave_master_info
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Replica 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:
Master_public_key_path:
Get_master_public_key: 0
Network_Namespace:
1 row in set, 1 warning (0,00 sec)
```

At this point you can be sure that you have binary replication enabled and working correctly.

pandora_server configuration

Version 770 or later.

It is necessary to configure inside the pandora_server.conf file a series of necessary parameters for the correct operation of the `pandora_ha`.

The parameters to be added are the following:

- ha_hosts <IP_ADDRESS1>,<IP_ADDRESS2> :

Configure the `ha_host` parameter followed by the IP addresses or FQDN of the MySQL servers that make up the HA environment. The IP address you set first will have preference to be the MASTER server or at least have the master role when you first start the HA environment. For example:

```
ha_hosts 192.168.80.170,192.168.80.172
```

- ha_dbuser and ha_dbpass:

These are the parameters where you must indicate the user and password of root user or otherwise a MySQL user with the maximum privileges that will be in charge of performing all the *master - slave* promotion operations on the nodes. For example:

```
ha_dbuser root
ha_dbpass pandora
```

- repl_dbuser and repl_dbpass:

The parameters to define the replication user that will use the SLAVE to connect to the MASTER. For example:

```
repl_dbuser slaveuser
repl_dbpass pandora
```

- ha_sshuser and ha_sshport:

The parameters to define the user/port with which it is connected by ssh to the Percona/MySQL servers to perform the recovery operations. For the correct operation of this option, it is necessary to have the ssh keys shared between the user with which the `pandora_ha` service is executed and the user indicated in the ha_sshuser parameter. Example:

```
ha_sshuser root
ha_sshport 22
```

- ha_resync `PATH_SCRIPT_RESYNC`:

By default the script to perform the resynchronization of the nodes, this is located at:

```
/usr/share/pandora_server/util/pandora_ha_resync_slave.sh
```

In the case of having a customized installation of the script, indicate in this parameter its location to perform the automatic or manual synchronization of the SLAVE node when needed.

```
ha_resync /usr/share/pandora_server/util/pandora_ha_resync_slave.sh
```

- ha_resync_log:

Log path where all the information related to the executions performed by the synchronization script configured in the previous token will be stored. For example:

```
ha_resync_log /var/log/pandoraha_resync.log
```

- ha_connect_retries:

Number of attempts it will perform on each check with each of the servers in the HA environment before making any changes to the environment. For example:

```
ha_connect_retries 2
```

Once all these parameters are configured, you could start Pandora FMS server with the `pandora_ha` service. The server will get an image of the environment and it will know at that moment who is the MASTER server.

When it knows it, it will create the `pandora_ha_hosts.conf` file in the `/var/spool/pandora/data_in/conf/` folder, where the Percona/MySQL server that has the MASTER role will be indicated at all times.

In case the `incomingdir` parameter of the `pandora_server.conf` file contains a different path (PATH), this file will be located at that PATH.

This file will be used as an interchange with Pandora FMS Console to find out at any time the IP address of the Percona/MySQL server with MASTER role.

- restart:

It will be indicated with a value of `0`, since the pandora_ha daemon is the one in charge of restarting the service in case of failure, thus avoiding possible conflicts. For example:

```
# Pandora FMS will restart after restart_delay seconds on critical errors.
restart 0
```

SSH key sharing between servers

An OpenSSH server must be installed and running on each host. Delete the welcome message or banner that displays OpenSSH, run on all devices:

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

Share the SSH keys between `pandora_ha` and all the existing Percona/MySQL servers in the environment, run on Pandora FMS server:

```
printf "\n\n\n" | ssh-keygen -t rsa -P ''
ssh-copy-id -p22 root@node1
ssh-copy-id -p22 root@node2
```

- In case you have the installation on Ubuntu Server, enable the root user to connect via SSH. This is done by generating a password to the root user by executing the `sudo passwd root` command.
- Then enable the SSH connection of the root user at least through shared keys "PermitRootLogin without-password" in the configuration file of the `sshd` service.

Using the synchronization script

With Pandora FMS server, a script is implemented that allows you to synchronize the SLAVE database in case it is out of sync.

The manual execution of this script is the following:

```
./pandora_ha_resync_slave.sh "pandora_server.conf file" MASTER SLAVE
```

For example, to make a manual synchronization from node 1 to node 2 the execution would be the following:

```
/usr/share/pandora_server/util/pandora_ha_resync_slave.sh
/etc/pandora/pandora_server.conf node1 node2
```

To configure the automatic recovery of the HA environment when there is any synchronization problem between MASTER and SLAVE, it is necessary to have the configuration token `splitbrain_autofix` configured to 1, inside the server configuration file (`/etc/pandora/pandora_server.conf`).

So, whenever a Split-Brain takes place (both servers have the master role) or there is any synchronization problem between MASTER and SLAVE node, `pandora_ha` will try to launch the `pandora_ha_resync_slave.sh` script to synchronize from that point the MASTER server status in the SLAVE server.

This process will generate events in the system indicating the start, the end and if any error took place in there.

Pandora FMS Console Configuration

Version 770 or later.

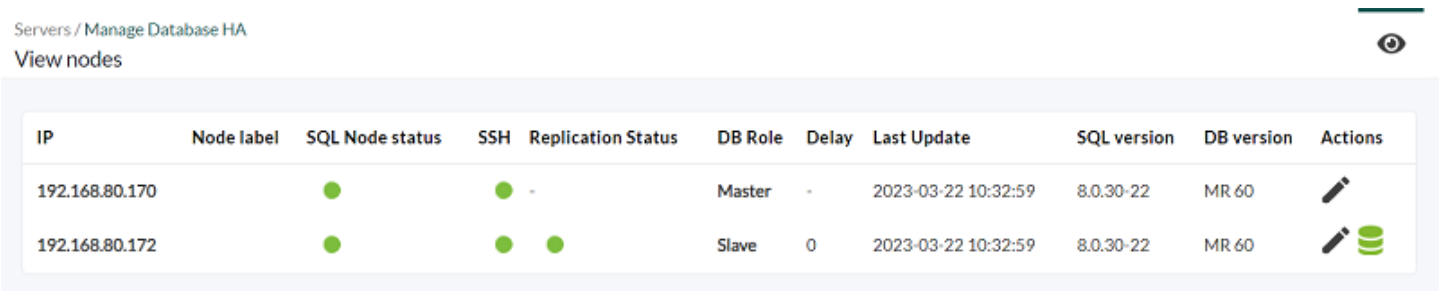
A new parameter was added to the `config.php` configuration indicating the path of the exchange directory that Pandora FMS uses by default `/var/spool/pandora/data_in`.

If it is configured, it will look for the file `/var/spool/pandora/data_in/conf/pandora_ha_hosts.conf` where it will get the IP address to make the connection.

```
$config["remote_config"] = "/var/spool/pandora/data_in";
```

In Pandora FMS console you could see the cluster status accessing to the Manage HA view.

```
https://PANDORA_IP/pandora_console/index.php?sec=gservers&sec2=enterprise/godmode/servers/HA_cluster
```



The data of this view is constantly updated thanks to `pandora_ha`, there is no need to do any previous configuration procedure to be able to see this section as long as the `pandora_server.conf` is correctly configured with the parameters mentioned in the previous section.

Among the available actions, you may configure a label for each one of the nodes and you may perform the option to synchronize the SLAVE node through the icon  .

This icon may have the following states:

- Green: Normal, no operation to be performed.
- Blue: Pending resynchronization.
- Yellow: Resynchronization is in progress.
- Red: *Error*, resynchronization failed.

Corosync-Pacemaker HA environment migration

The main difference between an HA environment used in MySQL/Percona Server version 5 and the current HA mode is that `pandora_ha` is now used to manage the cluster nodes to the detriment of Corosync-Pacemaker, which will no longer be used from now on.

Environment migration will consist of:

1.- Upgrading Percona from version 5.7 to version 8.0: "Installation and upgrade to MySQL 8".

2.- Install xtrabackup-80 on all devices:

```
yum install percona-xtrabackup-80
```

If you use Ubuntu server, see section "Percona 8 Installation for Ubuntu Server".

3.- Create all users again with the token `mysql_native_password` in the MASTER node:

```
mysql> CREATE USER slaveuser@% IDENTIFIED WITH mysql_native_password BY 'pandora';

mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE on *.* to slaveuser@%;
```

mysql> CREATE USER pandora@% IDENTIFIED WITH mysql_native_password BY 'pandora';

mysql> grant all privileges on pandora.* to pandora@%;

4.- Dump the database from the MASTER node to the SLAVE node:

4.1.- Make the full dump of the MASTER database:

MASTER #

```
xtrabackup --backup --target-dir=/root/pandoradb.bak/
```

MASTER #

```
xtrabackup --prepare --target-dir=/root/pandoradb.bak/
```

4.2.- Get the position of the backup binary log:

MASTER # cat /root/pandoradb.bak/xtrabackup_binlog_info

```
binlog.000003 157
```

Take note of these two values as they are needed in point 4.6.

4.3.- Synchronize with rsync with the SLAVE server to send the backup.

SLAVE # rm -rf /var/lib/mysql/*

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

4.4- On the SLAVE server, configure permissions so that MySQL server may access the sent files without any issues.

SLAVE # chown -R mysql:mysql /var/lib/mysql

SLAVE # chcon -R system_u:object_r:mysqld_db_t:s0 /var/lib/mysql

4.5.- Start `mysqld` service on the SLAVE server.

```
systemctl start mysqld
```

4.6.- Start the SLAVE mode on this server (use the data from point 4.2):

SLAVE # mysql -u root -ppandora

SLAVE # mysql> `reset slave all;`

SLAVE # mysql>

```
CHANGE MASTER TO MASTER_HOST='nodo1',
MASTER_USER='slaveuser', MASTER_PASSWORD='pandora',
MASTER_LOG_FILE='binlog.000003', MASTER_LOG_POS=157;
```

SLAVE # mysql> `start slave;`

SLAVE # mysql> `SET GLOBAL read_only=1;`

In case you want to install from scratch the environment in a new server, in the migration procedure you should only install from scratch as the current procedure indicates in the new environment, and in the step of creating Pandora FMS database you should import the data with a backup of the database of the old environment.

At the same time it will be necessary to save in the new environment the Pandora FMS Console and Server configuration indicated in previous sections.

Split-Brain

Due to several factors, high latency, network outages, etc., you might find that both MySQL servers acquired the master role and there is no `autoresync` option enabled in `pandora_ha`, so that the server itself chooses the server that will work as master and synchronizes the master node with the slave one, thus losing all the information that could be collected from that server.

To solve this problem data can be merged following this procedure.

This manual procedure only covers the data and event retrieval covering the interval between two dates. It assumes that it only recovers data from agents/modules that already exist in the node where the data merge will be performed.

If new agents are created during Split-Brain time, or new configuration information (alerts, policies, etc.), these will not be taken into account. Only data and events will be retrieved. That is, data related to the `tagente_datos`, `tagente_datos_string` and `tevento` tables.

The following commands will be executed in the node that was disconnected (the one to be promoted to SLAVE), where `yyyy-mm-dd hh:mm:ss` is the Split-Brain start date and time and `yyyy2-mm2-dd2 hh2:mm2:ss2` its end date and time.

Run mysqldump command with appropriate user rights to get a data dump:

```
mysqldump -u root -p -n -t --skip-create-options --databases pandora --tables
tagente_datos --where='FROM_UNIXTIME(utimestamp)> "yyyy-mm-dd hh:mm:ss" AND
FROM_UNIXTIME(utimestamp) <"yyyy2-mm2-dd2 hh2:mm2:ss2"'> tagente_datos.dump.sql
```

```
mysqldump -u root -p -n -t --skip-create-options --databases pandora --tables
tagente_datos_string --where='FROM_UNIXTIME(utimestamp)> "yyyy-mm-dd hh:mm:ss"
AND FROM_UNIXTIME(utimestamp) <"yyyy2-mm2-dd2 hh2:mm2:ss2"'>
tagente_datos_string.dump.sql
```

```
mysqldump -u root -p -n -t --skip-create-options --databases pandora --tables
tevento --where='FROM_UNIXTIME(utimestamp)> "yyyy-mm-dd hh:mm:ss" AND
FROM_UNIXTIME(utimestamp) <"yyyy2-mm2-dd2 hh2:mm2:ss2"' | sed -e
"s/([0-9]*,/(NULL,/gi"> tevento.dump.sql
```

Once the dumps of these tables have been obtained, data will be loaded to the MASTER node:

MASTER # `cat tagente_datos.dump.sql | mysql -u root -p pandora`

MASTER # `cat tagente_datos_string.dump.sql | mysql -u root -p pandora`

MASTER # `cat tagente_evento.dump.sql | mysql -u root -p pandora`

After loading the data you retrieved from the node to be promoted to SLAVE, proceed to synchronize it using the following procedure:

0.- Delete in MASTER the directory `/root/pandoradb.bak/`.

1.- Dump the Master DB:

MASTER #

```
xtrabackup --backup --target-dir=/root/pandoradb.bak/
```

MASTER #

```
xtrabackup --prepare --target-dir=/root/pandoradb.bak/
```

2.- Get the position of the binary log of the backed up data:

MASTER # `cat /root/pandoradb.bak/xtrabackup_binlog_info`

You will get something like the following (take due note of these values):

```
binlog.000003   157
```

At this point in SLAVE clear the contents of `/var/lib/mysql/` as indicated in point 4.3 above:

SLAVE # rm -rf /var/lib/mysql/*

3.- Do a task with the rsync command with the slave server to send the backup done.

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

4.- On the slave server, configure permissions so that MySQL server may access the files sent without any issues.

SLAVE # chown -R mysql:mysql /var/lib/mysql

SLAVE # chcon -R system_u:object_r:mysqld_db_t:s0 /var/lib/mysql

5.- Start `mysqld` service in the slave server.

```
systemctl start mysqld
```

6.- Start the slave mode on this server.

SLAVE # mysql -u root -ppandora

SLAVE # mysql> reset slave all;

SLAVE # mysql> CHANGE MASTER TO MASTER_HOST='nodo1',
MASTER_USER='slaveuser', MASTER_PASSWORD='pandora',
MASTER_LOG_FILE='binlog.000003', MASTER_LOG_POS=157;

SLAVE # mysql> start slave;

SLAVE # mysql> SET GLOBAL read_only=1;

Once all these steps have been completed, if you run the `show slave status` command inside MySQL shell, you will see that the node is in slave mode. If it was configured properly, you should see an output like the following example:

```
*************************** 1. row ***************************
               Slave_IO_State: Waiting for source to send event
                  Master_Host: node1
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: binlog.000018
          Read_Master_Log_Pos: 1135140
               Relay_Log_File: relay-bin.000002
                Relay_Log_Pos: 1135306
        Relay_Master_Log_File: binlog.000018
             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: 1135140
              Relay_Log_Space: 1135519
              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: fa99f1d6-b76a-11ed-9bc1-000c29cbc108
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Replica 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:
       Master_public_key_path:
        Get_master_public_key: 0
            Network_Namespace:
1 row in set, 1 warning (0,00 sec)
```

At this point make sure that you have binary replication enabled and working properly again.

Go back to Pandora FMS documentation index