PANDORAFMS

冗長化構成(HA)

冗長化構成(HA)

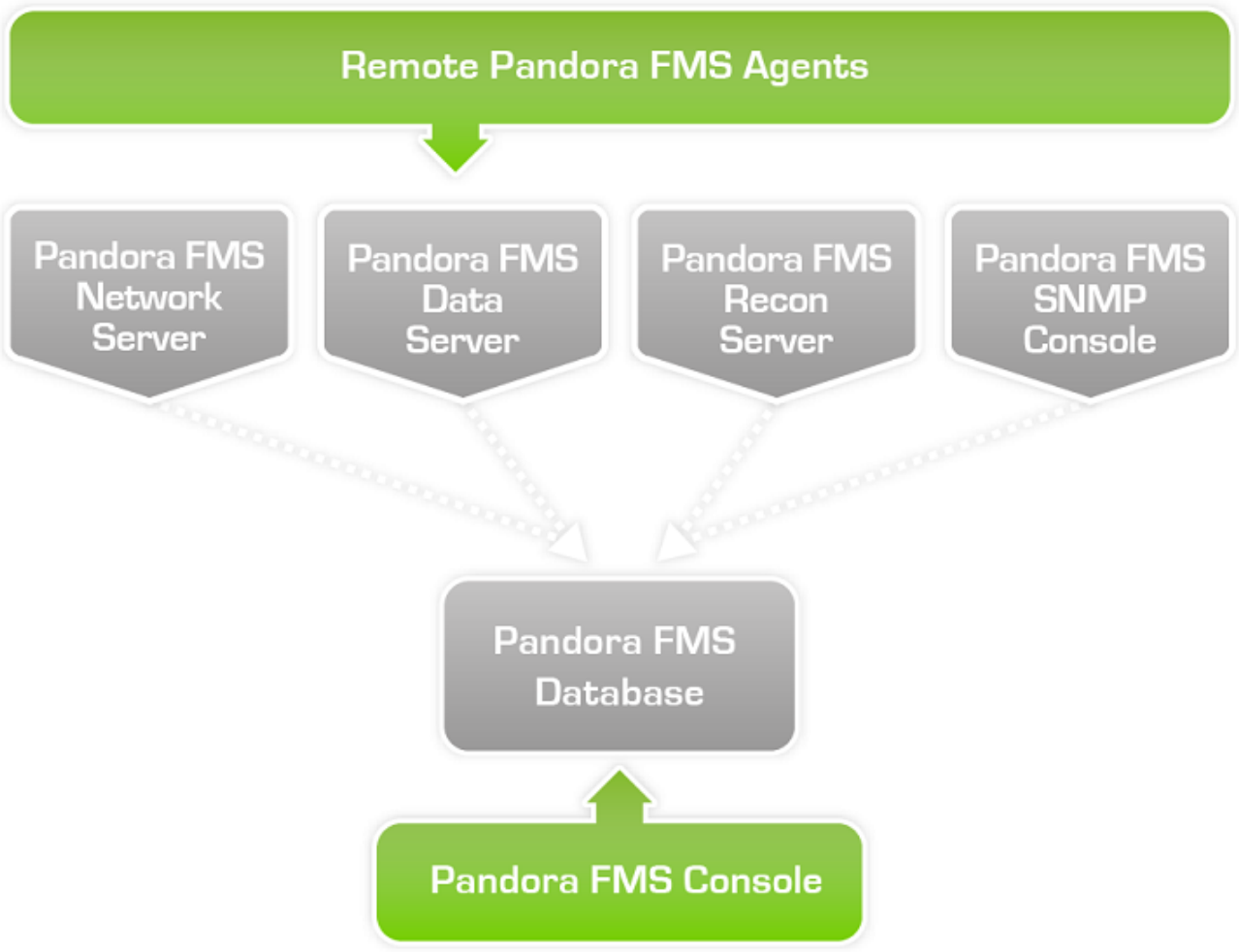
Pandora FMS ドキュメント一覧に戻る

概要

Pandora FMS はとても安定したアプリケーションになっています。(みなさんのテストのおかげで、それぞれのバージョンでは拡張が行われ、数百の不具合が報告され、修正されてきました。)しかし、クリティカルや高負荷な環境では、複数のサーバに処理を分散させることが可能です Pandora FMS では、いずれかのコンポーネントで障害が発生しても、システム自体はダウンしません。

Pandora FMS は、細かいモジュールで設計されています。それぞれのモジュールは、独立して動作することができます。しかし、それぞれはまた、連携して動作することができ、負荷を分配することができます。

Pandora FMS の基本設計を以下に示します。



もちろん、エージェントは冗長化構成ではありません。エージェントがダウンすると、モジュールの実行ができず、また、複数のエージェントを平行して実行することはできないため、またはシステム自体がダウンしているため、そのエージェントの全てのデータ収集はできなくなります。 重要なシステムに対する最良の解決策は、Pandora FMS エージェントのある無しに関わらず、冗長化しモニタリングすることです。

いくつかの場面において、次のような HA 構成が可能です。

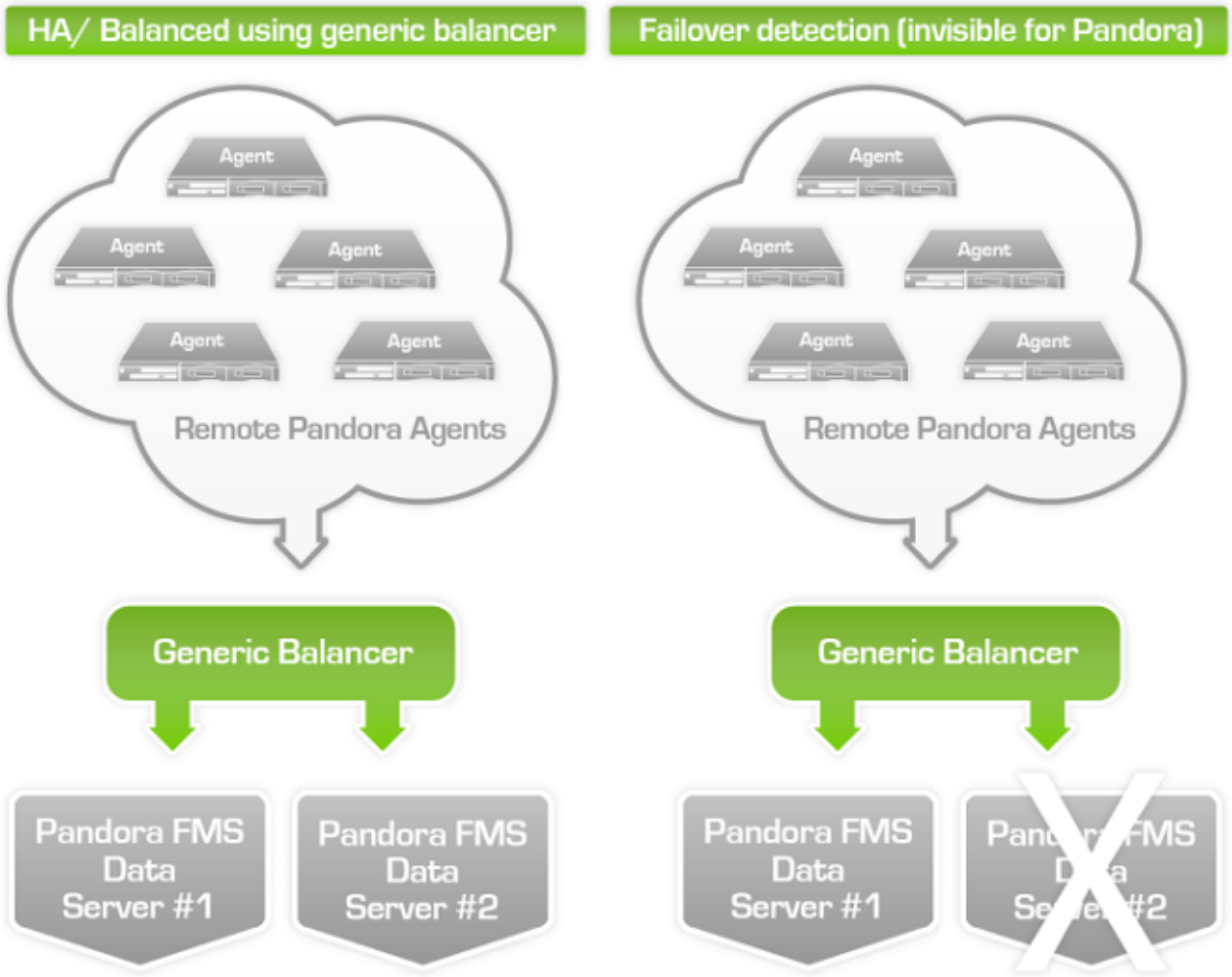
- データサーバのバランシングと HA
- ネットワーク、WMI、プラグイン、Web および予測サーバのバランシングと HA
- データベースのロードバランシング
- 自動検出サーバのバランシングと HA
- Pandora FMS コンソールのバランシングと HA

データサーバの HA

最も簡単な方法は、エージェントに実装されている HA を使用することです(プライマリが応答しない場合は代替サーバーに接続できます)。 ただし、データサーバはポート 41121 をサポートし、標準の TCP ポートであるため、通常の TCP サービスのバランシングまたはクラスタリングを可能にする商用ソリューションを使用することができます。

Pandora FMS データサーバでは、(異なるホスト名およびサーバ名で)設定された 2つの Pandora FMS データサーバを利用する必要があります。それぞれに tentacle サーバーを設定する必要があります。各マシンは異なる IP アドレスを持ちます。 外部バランサを使用する場合、バランサにエージェントからのデータ送信用の接続 IP アドレス(VIP)を割り当てます。

外部バランサを使用していて、いずれかのサーバが故障した場合、アクティブなサーバにのみ接続するようになります、Pandora FMS エージェントは変更に気付くことなく、これまでと同様のアドレスへ接続します。しかし、この場合は、ロードバランサーは障害が発生したサーバにはデータを送信せず、アクティブな方にのみ送信します。



Pandora FMS データサーバの設定を変更する必要はありません。それぞれのサーバに別々の名前を設定してさえいえれば、サーバの状態ビューでいずれかがダウンしていることがわかり便利です Pandora FMS データモジュールは、いずれかのサーバで処理することができます。事前のサーバ指定は不要です。簡単に HA 構成がとれるように設計されています。

エージェントのHAメカニズムを使用する場合は、データの送信にわずかな遅延があります。これは、エージェントの実行ごとにプライマリサーバとの接続を試みるためです。応答しない場合 セカンダリに対してこれを行います(設定されている場合)。これについては、次の“ ソフトウエアエージェントでのバランシング ”で説明します。

2つのデータサーバーを使用し、ポリシー、コレクション、およびリモート構成を管理する場合は、サーバデータディレクトリの複数 Pandora サーバでの共有 に従って次のディレクトリを共有する必要があります。データサーバのすべてのインスタンスでこれらのディレクトリを読み書きします。また、コンソールもこれらの共有ディレクトリにアクセスできる必要があります。

- /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

ソフトウエアエージェントでのバランシング

ソフトウエアエージェントで、データサーバのバランシングを行うことができます。データサーバの一方をマスター、もう一方をバックアップとして設定することができます。

エージェントの設定ファイル pandora_agent.conf において、次の部分のコメントを外します。

```
# 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
```

次に示すオプションがあります。(詳細は、エージェント設定の章を参照してください。)

- secondary_mode: セカンダリサーバのモードを設定します。次のいずれかが設定できます。
  - on_error: メインサーバにデータを送信できなかった場合のみセカンダリサーバにデータ送信します。
  - always: メインサーバに接続できるできないに関わらず、常にセカンダリサーバにもデータを送信します。
- secondary_server_ip: セカンダリサーバの IP アドレスを指定します。
- secondary_server_path: セカンダリサーバの XML ファイルを置く場所を指定します。通常は、/var/spool/pandora/data_in です。
- secondary_server_port: セカンダリサーバに XML ファイルを置くためのポート番号を指定します。tentacle では 41121、ssh は 22、ftp は 21 です。
- secondary_transfer_mode: セカンダリサーバに XML を送信するモード (tentacle, ssh, ftp など) を設定します。
- secondary_server_pwd: FTP で送信するためのパスワードを設定します。
- secondary_server_ssl: Tentacle その他でデータを送信するときに、ssl を使うかどうかを yes または no で指定します。
- secondary_server_opts: 転送に必要なその他オプションを設定します。

エージェントのリモート設定が有効になっている場合、メインサーバでのみ操作できます。

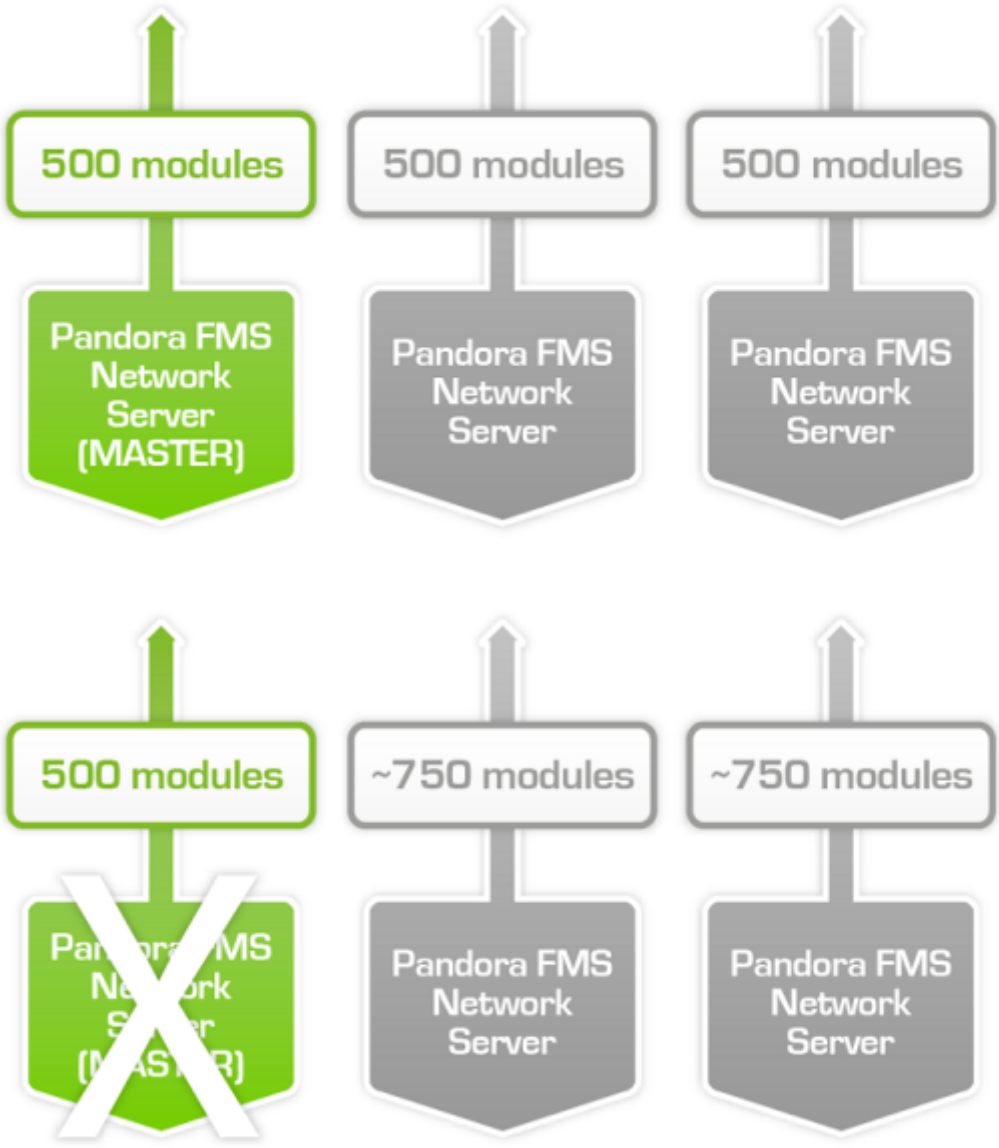
ネットワーク、WMI、プラグイン、ウェブ、予測サーバのバランシングと HA

これは簡単です。複数のサーバに、ネットワーク、WMI、プラグイン、ウェブ、予測サーバを (モニタ

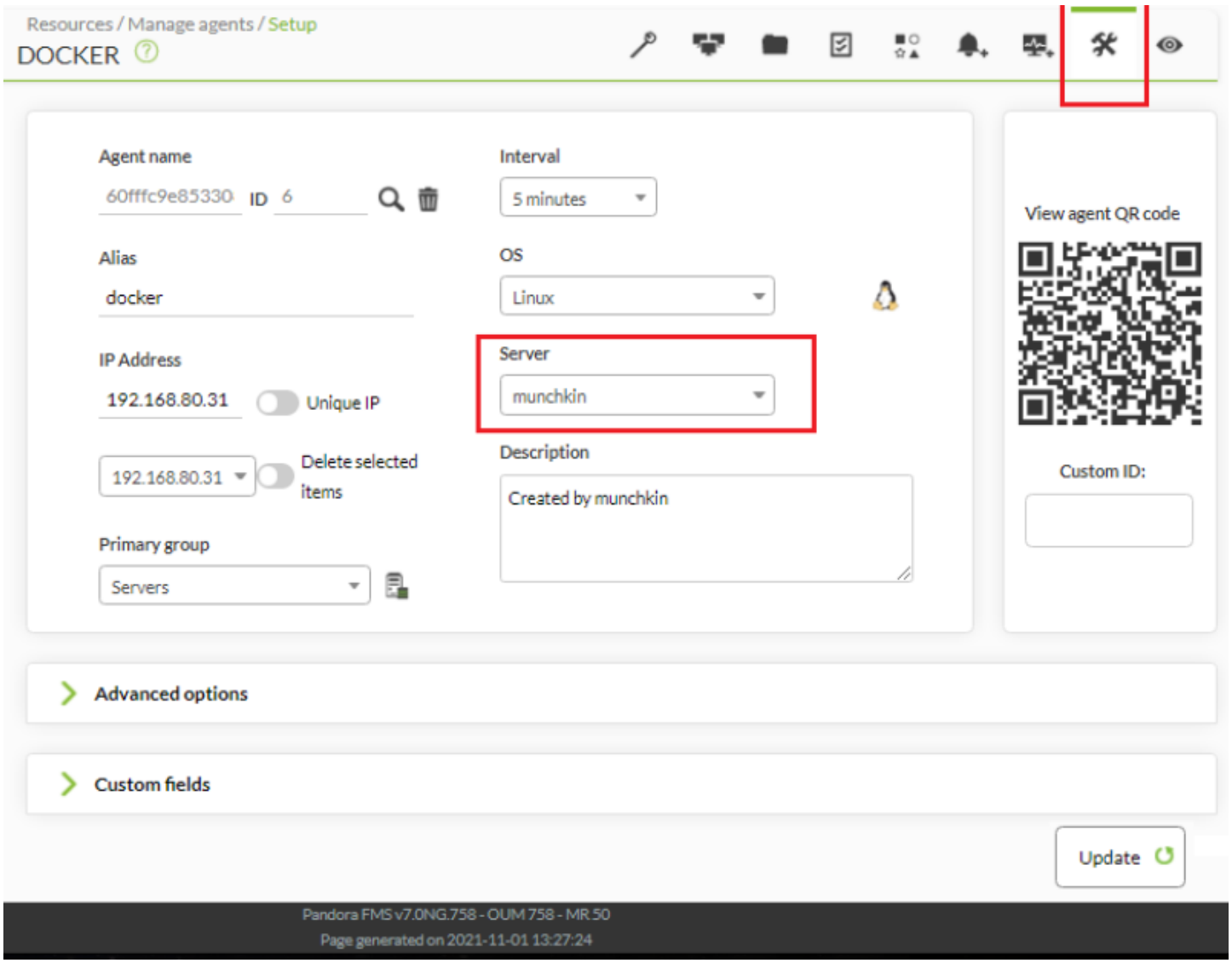
したいシステムを見れるよう同じように) それぞれインストールします。複数のサーバは (ネットワーク遅延が同じになるように) 同一セグメントに置く必要があります。

それぞれのサーバはプライマリとして選択できます。それぞれのサーバは、他方がダウンした場合、そのサーバに割り当てられていた全てのモジュールデータの収集を自動的に引き継ぎます。Pandora FMS サーバには、最終接続時間 (サーバの threshold x 2) を確認して、他のサーバがダウンしていることを検知する仕組が実装されています。Pandora FMS サーバが 1台でも稼働していれば、他のサーバのダウンを検出することができます。すべての Pandora FMS サーバがダウンした場合は、検出する手段はありません。

2つのノードのシステムで HA およびロードバランシングを実装する簡単な方法は、それぞれのサーバに 50% ずつモジュールを割り当て、両方のサーバをマスターとして選択します。2つ以上のマスターサーバがあり、3台目がダウンした場合は、1台目のマスターサーバがダウンしたサーバに割り当てられていたモジュールの実行を自分自身に割り当てます。ダウンしたサーバが復旧した場合は、再度モジュールの実行が自動的にプライマリサーバに割り当てられます。



異なるサーバ間でのロードバランシングは、エージェント管理(Agent Administration)の設定メニューで実施します。



“サーバ(server)”フィールドにて、監視を実行するサーバを選択することができます。

サーバの設定

Pandora FMS サーバは、異なる 2つのモードで実行できます。

- マスターモード
- 非マスターモード

もしサーバがダウンすると、処理が止まらないように、そのサーバが持っていたモジュールはマスターサーバで実行されます。

/etc/pandora/pandora_server.conf で *master* の設定が 0 より大きい値になっているすべてのサーバから、一つのマスターサーバが選択されます。

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

もし現在のマスターサーバがダウンすると、新たなマスターサーバが選択されます。複数の候補がある場合は、*master* の値が最も高いものが選択されます。

サーバの無効化には注意してください。ネットワークモジュールを持つサーバがダウンした場合、他のマスターサーバでネットワークサーバが無効化されていると、モジュールは実行されません。

例えば、*master* を 1 に設定した 3台の Pandora FMS サーバがある場合、マスターサーバはランダムに選択され、他の 2台は非マスターモードで動作します。マスターサーバがダウンすると、新たなマスターがランダムに選択されます。

pandora_server.conf に以下のパラメータを設定できます。

- ha_file: HA のテンポラリバイナリファイルの場所
- ha_pid_file: HA の現在のプロセス ID ファイル
- pandora_service_cmd: Pandora サービスの状態制御

Pandora FMS サーバの HA DB クラスタへの追加

データベースで高可用性が有効になっている 場合、より多くの Pandora FMS サーバを MySQL クラスタに接続するには、いくつかの追加設定が必要です。pandora_server.conf ファイル (デフォルトでは /etc/pandora にあります) で、追加する独立した Pandora FMS サーバごとに、次のパラメータを設定する必要があります。

- dbuser: MySQL クラスタへアクセスするためのユーザ名を指定する必要があります。例:

```
dbuser pandora
```

- dbpass: MySQL クラスタへアクセスするためのユーザのパスワードを指定する必要があります。例:

```
dbpass pandora
```

- ha_hosts: ha_host パラメータの後に、HA 環境を構成する MySQL サーバの IP アドレスまたは FQDN を設定する必要があります。例:

```
ha_hosts 192.168.80.170,192.168.80.172
```

Pandora FMS コンソールの HA

他のサーバにコンソールをインストールするだけです。それらのいずれからも、異なるユーザによって異なる場所から同時に使用することができます。 コンソールの前段でロードバランサを使用する

と、セッションシステムが「クッキー」によって管理され、これがブラウザーに保管されるため、どちらがアクセスされているかを実際に知らなくてもアクセスできます。

リモート設定を使用する場合、双方のデータサーバとコンソールは、データ入力ディレクトリ(デフォルト: /var/spool/pandora/data_in) 配下の configuration collections その他を共有する必要があります。("セキュリティアーキテクチャ" も参照ください。)

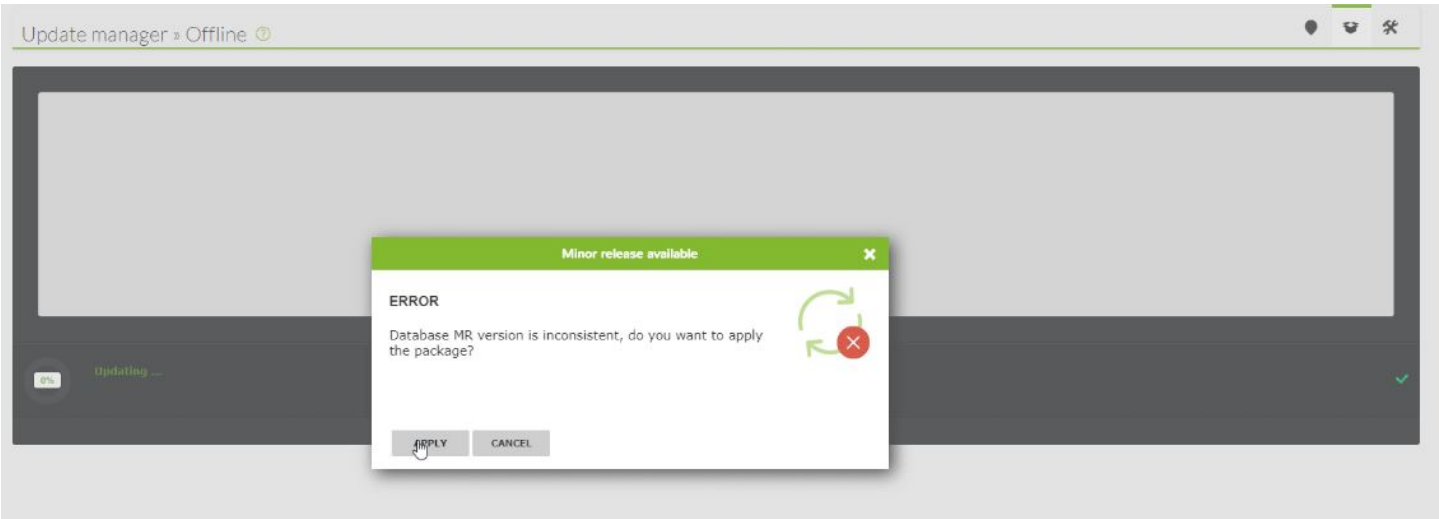
サーバのパフォーマンスに影響するため、data_in 以下のサブディレクトリのみ共有し、data_in 自体は共有しないことが重要です。

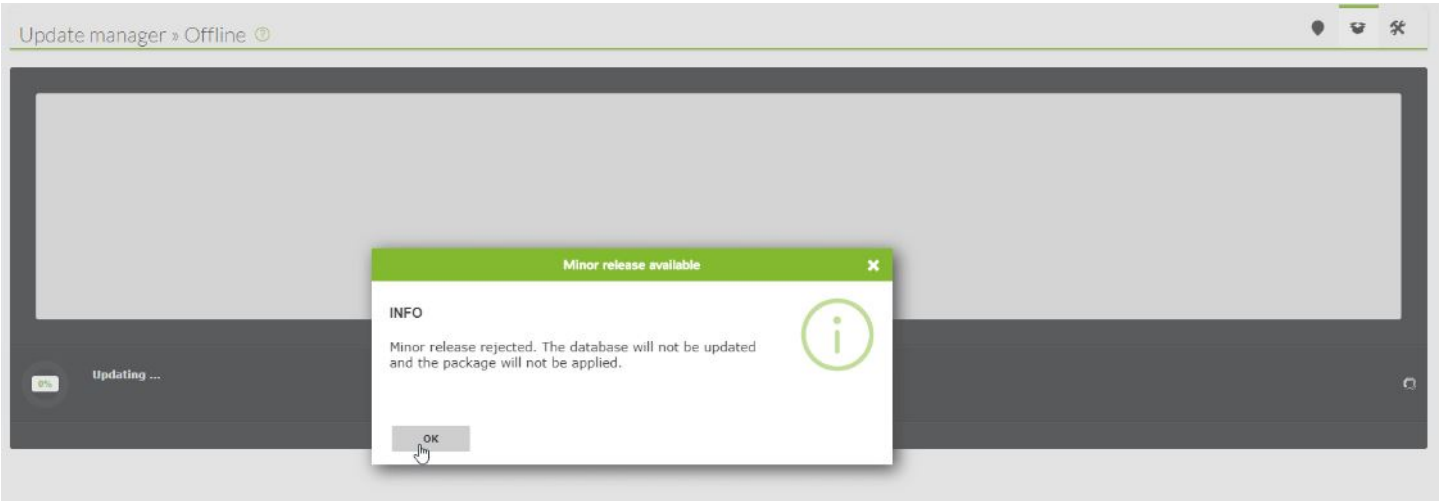
更新

HA 環境で Pandora FMS コンソールを更新する際に、アップデートマネージャ https://pandorafms.com/docs/index.php?title = Pandora:Documentation_ja:Anexo_Upgrade を介して OUM を使用して更新する場合は、次の点に考慮することが必要です。

Enterprise 版のユーザは、Pandora FMS サポートサイトから OUM パッケージをダウンロードできます。

共有データベースを使用するバランシング環境では、最初のコンソールを更新すると、対応する変更がデータベースに適用されます。 これは、セカンダリのコンソールを更新しようとすると、データベースがすでに更新されているため Pandora FMS がエラーメッセージを表示することになります。ただし、コンソールの更新は引き続き実行されます。





高可用性データベース

Enterprise この章の主な目的は、Pandora FMS 環境での HA の完全なソリューションを提供することです。 これは Pandora FMS で公式にサポートする唯一の HA モデルであり、バージョン 770 以降で提供されています。 このシステムは、クラスタ構成を以前のバージョンで利用していた Corosync と Pacemaker から置き換えます。

新しい Pandora FMS HA ソリューションは、製品 (pandora_ha バイナリ内) に統合されています Corosync/Pacemaker では実現不可能な、異なる IP 範囲を持つ地理的に分離されたサイトをサポートする HA を実装しています。

新しい HA モデルでは、通常のセットアップは 2 つのペアであり、クォーラムシステムを実装しない設計となっており、設定と必要なリソースが簡素化されています。 これにより、監視システムは利用可能な DB ノードがある限り機能し、DB スプリットブレインが発生している場合は、システムは両方のノードが再びマージされるまで並行して機能します。

新しい仕組では、現在の 3 つの問題を解決しようとしています。

1. 現在のシステムの複雑さと保守性 (バージョン NG 770 まで)。
2. ローカル以外のネットワークセグメントの利用で、HA 環境が地理的に異なる場所に分散している場合。
3. 地理的に離れた 2 つのサイト間で通信障害が発生した場合のスプリットブレインおよび安全なシステム運用のデータ復旧。

DB 用の新しい HA システムは Percona8 を用いて実装されていますが、将来のバージョンでは MySQL/MariaDB 8 でも実装する方法を詳しく説明します。

Pandora FMS は、MySQL データベースを用いて設定およびデータを保存するため、データベースに障害が発生すると監視ツールが一時的に麻痺する可能性があります Pandora FMS の高可用性データベースクラスタにより、強力でフェイルセーフなアーキテクチャを簡単に展開できます。

これは GNU/Linux システムの知識を必要とする高度な機能です。 すべて

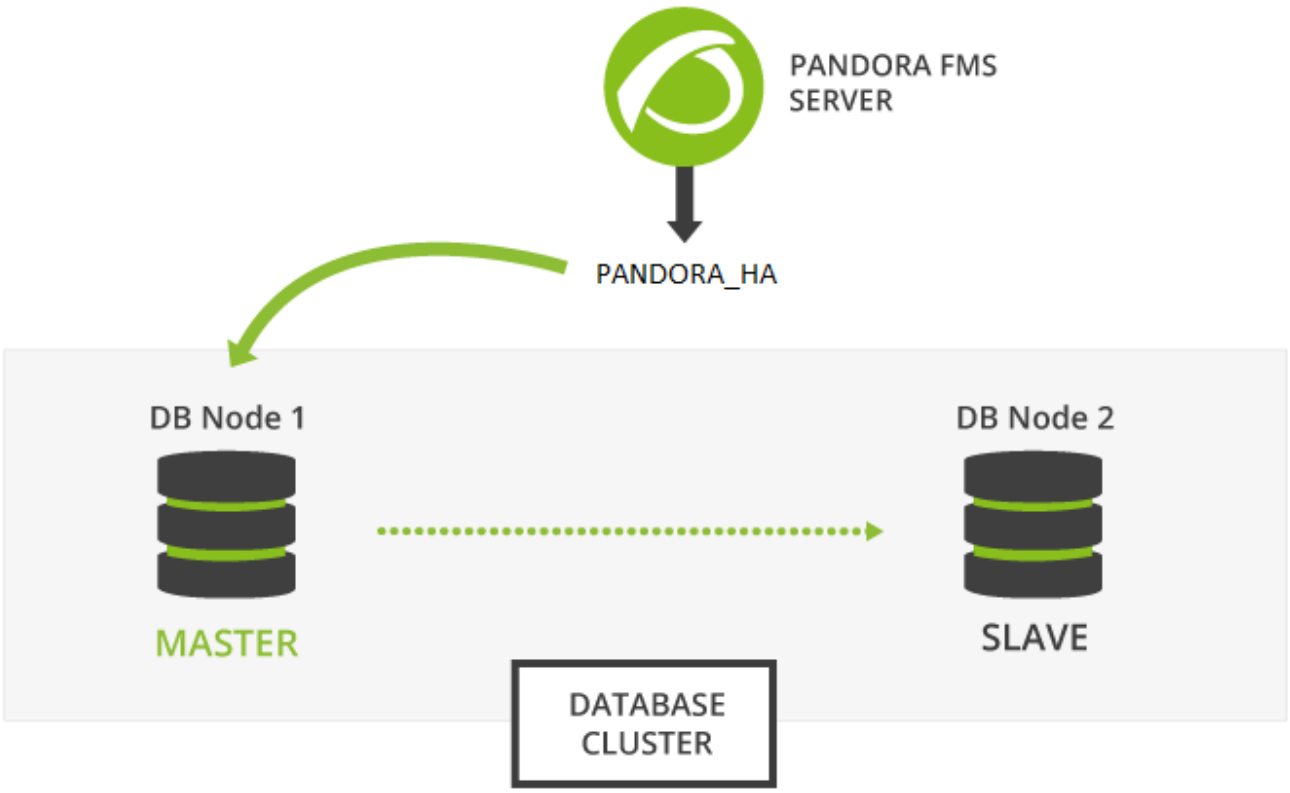
のサーバの時刻が NTP サーバ (Rocky Linux 8 の chronyd サービス) で同期されていることが重要です。

バイナリレプリケーション MySQL クラスタノードは、バージョン 770 (Enterprise 機能) 以降、pandora_ha バイナリで管理されます Percona は、そのスケーラビリティ、可用性、セキュリティ、およびバックアップ機能により、デフォルトの RDBMS として選択されました。

アクティブ/スタンバイレプリケーション は、単一のマスターノード (書き込み権限あり) から任意の数のセカンダリ (読み取りのみ) に対して行われます。マスターノードに障害が発生した場合、セカンダリの 1 つがマスターにアップグレードされ、pandora_ha がマスターノードの IP アドレスを更新します。

環境は次の要素で構成されます。

- バイナリレプリケーションが有効になっている MySQL8 サーバ (アクティブ - スタンバイ)。
- クラスタの正しい動作に必要なスレーブ - マスターおよびマスター - スレーブの継続的な監視の実施および昇格を実行するための、すべての MySQL サーバの設定を持つ pandora_ha を含むサーバ。



Percona 8 のインストール

バージョン 770 以降

RHEL 8 および Rocky Linux 8 への Percona 8 のインストール

最初に、後で Percona サーバパッケージをインストールできるように、環境のすべてのノードに Percona リポジトリをインストールする必要があります。

root 権限で、または root ユーザとしてターミナルウィンドウを開く必要があります。各ユーザの責任において実行してください。次の手順には、すべてのデバイスで実行する必要があるもの、一部のデバイスで実行する必要があるもの、特定の 1 つのデバイスで実行する必要があものがあります。注釈に注意してください。

関連するすべてのデバイスで実行します:

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

すべてのデバイスで Percona リポジトリのバージョン 8 を有効にします:

```
percona-release setup ps80
```

両方の環境を手動で同期するためのバックアップツールと Percona サーバをインストールします。関連する すべて のデバイスで実行します:

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

Percona サーバーを Web コンソールおよび Pandora FMS サーバと一緒にインストールする場合、MYVER=80 パラメーターを使用することで MySQL バージョン 8 デプロイするように指定できます。。

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

Percona 8 の Ubuntu Server へのインストール

すべて のデバイスに Percona リポジトリ バージョン 8 をインストールします。

```
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
```

すべて のデバイスで Percona リポジトリ バージョン 8 を有効化します:

```
percona-release setup ps80
```

両方の環境を手動で同期するためのバックアップツールと Percona サーバをインストールします。すべて のデバイスで以下を実行します:

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

バイナリレプリケーション設定

レプリケーションによって生成された *binlogs* を追加のパーティションまたは外部ディスクに保存する場合は、そのサイズがデータベース用に予約されているサイズと同じである必要があります。 `log-bin` および `log-bin-index` トークンも参照してください。

すべてのクラスタノードに MySQL サーバをインストールしたら、両方の環境をレプリケートするように設定します。

まず、設定ファイル `my.cnf` を設定して、バイナリレプリケーションが正しく機能するように準備します。

ノード 1

ノード 1 `/etc/my.cnf` (Ubuntu server の場合は `/etc/mysql/my.cnf`):

```
[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=MIXED
binlog_expire_logs_seconds=172800 # 2 DAYS
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
```

- OPTIMIZATION FOR PANDORA FMS というコメントの後の設定は、Pandora FMS 用に最適化された設定です。
- コメント SPECIFIC PARAMETERS FOR BINARY REPLICATION の後に、バイナリレプリケーションのための特定のパラメーターを設定しています。
- binlog_expire_logs_seconds というトークンは、2 日で設定しています。
- [client] サブセクションで、データベースに使用するユーザとパスワードを入力します。Pandora FMS をインストールするときのデフォルトでは、それぞれ root と pandora です。 *これらの値は、ユーザの入力無しに(自動化された)バックアップを実行するために 必要です。*

server_id トークンがすべてのノードで異なることが重要です。この例では、ノード 1 でこの番号が使用されています。

ノード 2

ノード 2 /etc/my.cnf (Ubuntu server の場合は /etc/mysql/my.cnf)

```
[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=MIXED
binlog_expire_logs_seconds=172800 # 2 DAYS
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
```

- OPTIMIZATION FOR PANDORA FMS というコメントの後の設定は、Pandora FMS 用に最適化された設定です。
- コメント SPECIFIC PARAMETERS FOR BINARY REPLICATION の後に、バイナリレプリケーションのための特定のパラメーターを設定しています。
- binlog_expire_logs_seconds というトークンは、2 日で設定しています。
- [client] サブセクションで、データベースに使用するユーザとパスワードを入力します Pandora FMS をインストールするときのデフォルトでは、それぞれ root と pandora です。 *これらの値は、ユーザの入力無しに(自動化された)バックアップを実行するために必要です。*

server_id トークンがすべてのノードで異なることが重要です。この例では、ノード 2 でこの番号が使用されています。

マスターノード設定

両方のノードで正しい設定ができたら、*マスターサーバの役割を果たすノードの設定を開始します*

1.- mysqld サービスを開始します:

```
systemctl start mysqld
```

2.- ログファイル /var/log/mysqld.log に書かれているテンポラリの root パスワードでアクセスします:

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

見つけたパスワードで、MySQL サーバへアクセスします:

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

Password: → grep コマンドで見つけたパスワードを入力します。

3.- root ユーザのテンポラリパスワードを pandora に変更します。mysql > プロンプトは MySQL コマンドインタプリタ(MYSQL CLI)です。

```
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.- バイナリレプリケーションユーザおよび、リモート接続とクラスタ管理のための root ユーザを作成します:

```
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.- Pandora FMS データベースを作成します:

```
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
```

source コマンド: 同じサーバへ Pandora FMS コンソールを
インストールしている場合。*そうでなければ、このファイ
ルをマスターサーバへコピーします。*

6.- pandora ユーザを作成し、このユーザにアクセス権限を与えます:

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

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

これで Pandora FMS データベースのレプリケーションを開始するための
マスターサーバの準備が完了です。

データベースの複製

次のステップは、スレーブノード (SLAVE) でマスターデータベース (MASTER) の複製を作成することです。これを行うには次の手順に従います。

1.- MASTER データベースの完全な複製(dump)を作成します:

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

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

2.- *バックアップバイナリログのポジション*を取得します:

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

次のような出力があります:

```
binlog.000003 157
```

6 番目の手順で必要となるため、この 2つの値をメモしておきます。

3.- rsync を使って SLAVE サーバにバックアップのコピーを送ります:

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

4.- SLAVE サーバで、問題なく MySQL サーバがファイルにアクセスできるようにパーミッションを

設定します:

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

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

5.- SLAVE サーバで、mysqld サービスを開始します:

```
systemctl start mysqld
```

6.- このサーバで SLAVE モードを開始します(手順 2 で確認したデータを利用します):

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;

これらのすべての手順が完了したら、MySQL シェル内で show slave status コマンドを実行すると、ノードがスレーブとして設定されていることがわかります。 正しく設定されている場合は、次のような出力が表示されます。

```
*************************** 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)
```

この時点で、バイナリレプリケーションが有効になり、正しく動作していることを確認できます。

pandora_server 設定

バージョン 770 以降

pandora_server.conf ファイル内で、pandora_ha が正しく動作するために必要な一連のパラメータを設定する必要があります。

追加するパラメータは次の通りです。

- ha_mode [pandora|pacemaker] :

現在の pandora_ha 設定では pandora を指定します。以前のモード (バージョン 769 以前) を使用している場合、pacemaker を指定します。 例：

```
ha_mode pandora
```

- ha_hosts <IPアドレス1>,<IPアドレス2> :

HA 環境を構成する MySQL サーバの IP アドレスまたは FQDN を指定する ha_host パラメーターを設定します。 最初に入力した IP アドレスは、MASTER サーバになるか□HA 環境を最初に開始するときに少なくともマスターの役割を持つサーバです。 例：

```
ha_hosts 192.168.80.170,192.168.80.172
```

- ha_dbuser および ha_dbpass :

これらは、root ユーザのユーザとパスワードを指定する必要があるパラメータです□root 以外の場合は、各ノードで マスター - スレーブ昇格操作を実行できる特権を持つ MySQL ユーザを指定する必要があります。 例：

```
ha_dbuser root
ha_dbpass pandora
```

- repl_dbuser および repl_dbpass :

SLAVE から MASTER に接続するレプリケーションユーザを定義するパラメータです。 例：

```
repl_dbuser slaveuser
repl_dbpass pandora
```

- ha_sshuser および ha_sshport :

リカバリ操作を実行するために ssh によって Percona/MySQL サーバに接続されるユーザ/ポートを定義するパラメータです。このオプションを正しく動作させるには、pandora_ha サービスを実行するユーザと ha_sshuser パラメータで指定されたユーザの間で ssh キーを共有する必要があります。 例：

```
ha_sshuser root
ha_sshport 22
```

- ha_resyncPATH_SCRIPT_RESYNC :

デフォルトでは、ノードの再同期**を**実行**するスクリプトは**次の場所**にあります**P

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

スクリプトをカスタマイズしてインストールする場合**は、**必要**に応じて** SLAVE **ノードの**自動**または**手動同期**を**実行**するスクリプトの**場所**をこのパラメータで**指定**します。**

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

- ha_resync_log :

前のトークンで設定**された**同期**スクリプトの**実行**に関するすべての**情報**が格納されるログのパスです。** 例：

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

- ha_connect_retries :

環境に変更**を加える前に、**HA **環境**内の**各サーバで各チェック**を**試行する**回数です。** 例：

```
ha_connect_retries 2
```

これらのパラメータをすべて設定**したら、**pandora_ha **サービスで** Pandora FMS **サーバを**起動できます。** サーバは**環境の**状態**を**取得**し、その**時点でどれが** MASTER **サーバであるかを**認識**します。**

それを認識**すると、**/var/spool/pandora/data_in/conf/ **フォルダに** pandora_ha_hosts.conf **ファイルが**作成**されます。これには、**MASTER **の**役割**を**持つ** Percona/MySQL **サーバが**常に示**されています。**

pandora_server.conf **ファイルの** incomingdir **パラメータに**別のパス (PATH) **が**指定**されている**場合、**このファイルはそのパスに**配置**されます。**

このファイルは、MASTER **の**役割**を**持つ** Percona/MySQL **サーバの** IP **アドレスをいつでも**知れるように Pandora FMS **コンソールでも**使用**されます。**

サーバ間の SSH 鍵共有

OpenSSH **サーバがインストールされ、**各**ホストで**動作**している**必要があります OpenSSH **を**表示**するウェルカムメッセージまたはバナーを**抑制**します。すべて のデバイスで**以下を**実行**します:**

```
[ -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
```

pandora_ha と環境内に存在するすべての Percona/MySQL サーバ間で SSH 鍵を共有します。Pandora FMS サーバで以下を実行します:

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

- Ubuntu Server にインストールしている場合は、root ユーザが SSH 経由で接続できるようにします。 これは、sudo passwd root コマンドを実行して root ユーザにパスワードを設定することによって行えます。
- 次に、sshd サービスの設定ファイルにある “ PermitRootLogin without-password ” を使用して root ユーザの SSH 接続を有効にします。

同期スクリプトの利用

Pandora FMS サーバでは、同期が取れていない場合に SLAVE データベースを同期できるようにするスクリプトが実装されています。

スクリプトの手動実行は次の通りです:

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

例えば、ノード1 からノード2 へ手動で同期するには、次のように実行します:

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

MASTER と SLAVE の間で同期の問題が発生した際に HA 環境の自動回復を設定するには、サーバ設定ファイル(/etc/pandora/pandora_server.conf)内の設定トークン splitbrain_autofix を 1 にする必要があります。

そうすると、スプリットブレイン が発生した場合 (両方のサーバがマスターの役割を持っている場合)、または MASTER と SLAVE の間で同期の問題が発生した場合は、常に pandora_ha が pandora_ha_resync_slave.sh スクリプトを起動して、その時点のマスターサーバの状態をスレーブサーバへ同期しようとします。

この処理は、開始、終了、およびそこでエラーが発生したかどうかを示すイベントをシステムで生成します。

Pandora FMS コンソールの設定

バージョン 770 以降

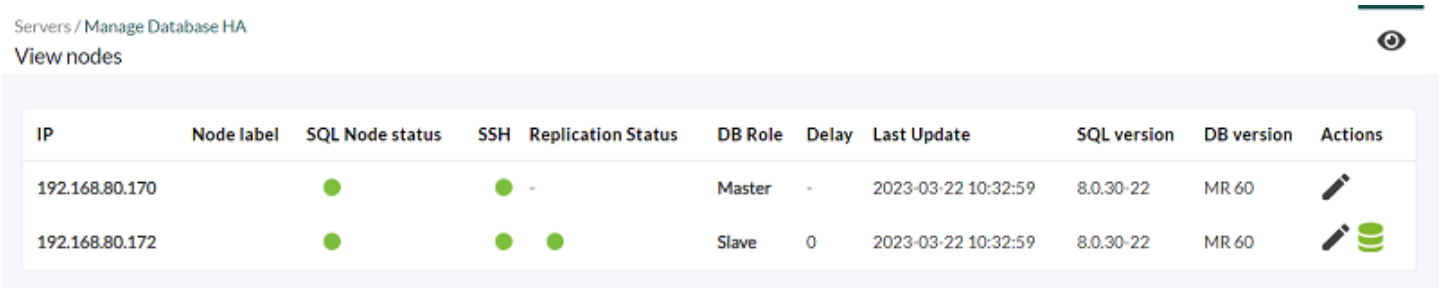
Pandora FMS が使用するデータ受け取りディレクトリのパス、デフォルトでは /var/spool/pandora/data_in を示す新しいパラメータが config.php に追加されました。

設定されている場合、/var/spool/pandora/data_in/conf/pandora_ha_hosts.conf ファイルを探して、接続先の IP アドレスを取得します。

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

Pandora FMS コンソールでは HA 管理画面にアクセスしてクラスタの状態を確認できます。

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



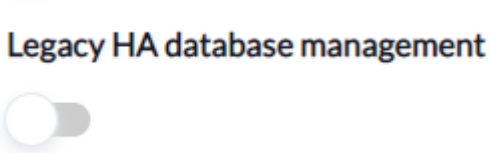
この画面のデータは、pandora_ha により常に更新されます。pandora_server.conf で前述のパラメータが正しく設定されている限り、このデータを表示するための特別な設定はありません。

実行可能なアクションとしては、各ノードのラベルを設定することができ、  アイコンにて SLAVE ノードを同期するオプションを実行できます。

このアイコンには次の状態があります。

- 緑: 正常、何の処理も実行されていない。
- 青: 同期保留中。
- 黄: 同期実行中。
- 赤: エラー、同期失敗。

セットアップ(Setup) → セットアップ(Setup) → Enterprise で、レガシーデータベース HA 管理(Legacy HA database management) オプションを無効化し、この新しいモード設定で HA 表示できるようにする必要があります。



Corosync-Pacemaker HA 環境マイグレーション

MySQL/Percona サーババージョン 5 で使用されている HA 環境と現在の HA モードの主な違いは、pandora_ha がクラスタノードの管理に使用されるようになったため、Corosync-Pacemaker は使用されなくなった点です。

環境の移行は次のようにできます。

1.- Percona のバージョン 5.7 からバージョン 8.0 へのアップグレード: “ MySQL 8 のインストールとアップグレード”

2.- すべてのデバイスで xtrabackup-80 のインストール:

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

Ubuntu server を利用している場合は、“ Ubuntu server への Percona 8 のインストール ” を参照してください。

3.- MASTER ノードで mysql_native_password トークンを指定しての全ユーザの再作成:

```
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.- MASTER ノードでのデータベースのダンプと SLAVE ノードへの適用:

4.1.- MASTER データベースのフルダンプの取得:

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

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

4.2.- バイナリログポジションの取得:

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

binlog.000003 157
```

4.6 の手順で必要になるため、この値はメモしておきます。

4.3.- バックアップを SLAVE サーバへ送るための rsync を使った同期。

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

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

4.4- SLAVE サーバにて MySQL サーバが送信したファイルに問題なくアクセスできるようにするためのパーミッションの設定。

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

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

4.5.- SLAVE サーバでの mysqld サービスの開始。

```
systemctl start mysqld
```

4.6.- このサーバで SLAVE モードを開始(4.2 の手順のデータを使います)。

```
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;
```

新しいサーバにゼロから環境を構築して移行したい場合は、新しい環境を構築したあとに Pandora FMS データベースを作成するステップで古い環境のデータベースのバックアップをインポートする必要があります。

同時に、前述の Pandora FMS コンソールとサーバの設定を新しい環境に行う必要があります。

スプリットブレイン

高いレイテンシー、ネットワークの停止など、いくつかの要因により、両方の MySQL サーバがマスターになり、かつ、サーバ自体がマスターとして機能するサーバを選択してマスターノードからスレーブへ同期を行う pandora_ha での autoresync オプションが有効になっていない場合があります。その際、サーバで収集されるデータをすべて失う可能性があります。

この問題を解決するには、次の手順でデータをマージすることができます。

この手動手順は、2つの日付の間のデータおよびイベントの
取得のみを対象としています。また、データマージが行わ
れるノードに既に存在するエージェント/モジュールからデー
タを復旧することのみを想定しています。

スプリットブレイン発生中に新しいエージェントが作成された場合、または新しい設定情報（アラート、ポリシーなど）がある場合、これらは考慮されません。取得されるのはデータとイベントのみです。つまり、tagente_datos、tagente_datos_string、tevento テーブル に関するデータです。

以下のコマンドは、切断されたノード（SLAVE に変更するノード）で実行します。yyyy-mm-dd hh:mm:ss はスプリットブレイン開始日時、 yyyy2-mm2-dd2 hh2:mm2:ss2 は終了日時です。

適切なユーザー権限で mysqldump コマンドを実行し、データダンプ（データダンプまたは単にダンプ）を取得します：

```
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
```

これらのテーブルのダンプを取得したら、MASTER ノードにデータをロードします。

```
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
```

SLAVE へ変更するノードから取得したデータをロードした後、以下の手順で同期を進めていきます：

1.- マスタ DB の全体のダンプを取得します:

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

MASTER # xtrabackup –prepare –target-dir=/root/pandoradb.bak/

2.- バックアップデータのバイナリログポジションを取得します:

MASTER # cat /root/pandoradb.bak/xtrabackup_binlog_info

次のような出力があります(値はメモしておきます):

```
binlog.000003   157
```

3.- rsync コマンドで、バックアップをスレーブサーバへ送ります:

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

4.- スレーブサーバでは、MySQL サーバが正しくファイルにアクセスできるようにパーミッションを調整します。

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

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

5.- スレーブサーバで mysqld サービスを開始します。

```
systemctl start mysqld
```

6.- このサーバでスレーブモードを開始します。

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;

これらの手順がすべて完了したら、MySQL シェル内で show slave status コマンドを実行すると、ノードがスレーブモードであることが確認できます。正しく設定されていれば、次の例のような出力が表示されるはずです：

```
*************************** 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)
```

この時点で、バイナリレプリケーションが有効になり、再び正しく動作していることを確認できます。