

EDB Failover Manager User Guide 4.2

EnterpriseDB

EDB Failover Manager User Guide

**** EDB Failover Manager ****

EDB Postgresのフェイルオーバーマネージャー（EFM）は、プライマリ上のソフトウェアまたはハードウェア障害が発生しイベントにスタンバイ・ノードに自動的にフェイルオーバーするPostgresのプライマリノードを可能EnterpriseDBのより高可用性モジュールです。

このガイドでは、フェールオーバーマネージャーのインストール、構成、および使用に関する情報を提供します。 Failover Managerがサポートするプラットフォームとバージョンの詳細については、次のEnterpriseDB ウェブサイトしてください。

<https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms#efm>

このドキュメントでは、 PostgreSQLまたはEDB Postgres Advanced Serverデータベースを意味するPostgresを使用しています。

whats_new

failover_manager_overviewinstalling_efmconfiguring_efmusing_efmmonitoring_efm_clusterusing_efm_utilitycontrolled_efm_servicecontrolling_logging通知supported_scenarios
upgrade_existing_clusterトラブルシューティング
configuring_streaming_replicationconfiguring_ssl_authenticationの結論

</ div>

What's New

バージョン4.2を作成するために、 EDB Postgres Failover Managerに次の変更が加えられました。

- 非sudoモードでPgpool EFM統合を有効にするサポートが追加されました。
- フェイルオーバーマネージャーは現在、プライマリエージェントで障害が発生した場合に、ロードからノードを切り離す必要はありませんが、データベースがまだ到達可能であるかどうかを指定するためのオプションを提供します。
- 特定のプライマリノードに適用可能なスタンバイの優先度を定義するサポートが追加されました。
- スタンバイを再構成する方法が強化され、アーカイビングが使用されるときにスタンバイが新しいプライマリに適切に従うようになりました。
- 以前は古いプライマリがアクティブのままで監視されていなかった状況でのフェールオーバーを可能にするために、文書にEager Failoverという実行モードが追加されました。

た。

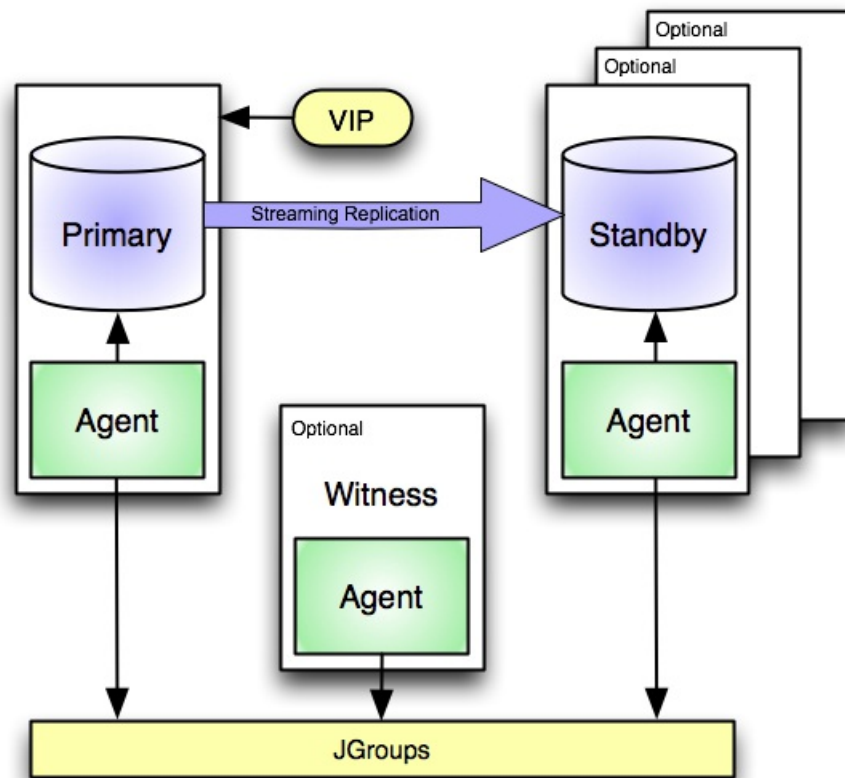
- stop-clusterコマンドを無効にするオプションが追加されました。

Failover Manager Overview

EDB Postgres Failover Manager (EFM) クラスターは、ネットワーク上の次のホストに存在するFailover Managerプロセスで構成されます。

- プライマリノード-プライマリノードは、データベースクライアントにサービスを提供しているプライマリデータベースサーバです。
- 1つ以上のスタンバイノード-スタンバイノードは、プライマリノードに関連付けられたストリーミングレプリケーションサーバです。
- 監視ノード-監視ノードは、フェイルオーバーシナリオでプライマリまたはスタンバイのアサーションを確認します。クラスターに3つ以上のノードが含まれる場合、クラスターには専用の監視ノードは必要ありません。データベースホストである3番目のクラスターメンバがない場合は、専用の監視ノードを追加できます。クラスターには、複数の監視ノードが含まれる場合があります。

従来、クラスターはマルチプルのデータベースを管理するPostgresの単一インスタンスです。このドキュメントでは、クラスターという用語はフェールオーバーマネージャークラスターを指します。フェールオーバーマネージャークラスターは、クラウドまたは従来のネットワーク上のサーバーに存在し、JGroupsツールキットを使用して通信するプライマリエージェント、1つ以上のスタンバイエージェント、およびオプションの監視エージェントで構成されます。



An EFM scenario employing a virtual IP address

- 仮想IPアドレスを使用するEFMシナリオ*

非監視エージェントが起動すると、ローカルデータベースに接続し、データベースの状態を確認しデータベース。

- エージェントがデータベースに到達できない場合、アイドルモードでスタートします。
- データベースがリカバリであることがわかると、エージェントはスタンバイのロールを引き受けます。
- データベースがリカバリでない場合、エージェントはプライマリのロールを引き受けま

す。

フェイルオーバーが発生しイベント、フェイルオーバーマネージャーの試みが進めスタンバイ内の最新スタンバイであることを保証します。スタンバイノードがプライマリノードと同期していない場合、データロスる可能性があることにノートしてください。

JGroupsは、フェールオーバーマネージャーが、メンバノードが相互に通信してノード障害を検出できるクラスターを作成できるようにする技術を提供します。

上記の図は、仮想IPアドレスを使用するFailover Managerクラスターを示しています。あなたが再構成するために、データベースが追加または削除されるたびにロードを独自に提供している場合は、代わりにロードを使用することができます。また、高可用性のためにネイティブEFM-Pgpool統合を有効にすることもできます。

前提条件

Prerequisites

フェールオーバーマネージャークラスターを構成する前に、以下で説明する前提条件を満たす必要があります。

**** Java 1.8（以降）をインストールします****

Failover Managerを使用する前に、最初にJava（バージョン1.8以降）をインストールする必要があります。フェールオーバーマネージャーはOpenJDKでテストされています。そのバージョンのJavaをインストールすることを強くお勧めします。 [Installation instructions for Java](#)はプラットフォーム固有です。

**** SMTPサーバーを提供する****

ユーザー定義の通知スクリプト、メール、またはその両方で指定されたとおりに、フェールオーバーマネージャーから通知を受信できます。

- メール通知を使用している場合、フェールオーバーマネージャーシナリオの各ノードでSMTPサーバーが実行されている必要があります。
- スクリプトに値を指定した場合。通知プロパティでは、ユーザを残すことができます。メールフィールドは空白です。SMTPサーバーは必要ありません。

イベントが発生すると、フェールオーバーマネージャーはスクリプト（提供されている場合）を呼び出し、ユーザで指定された任意のメールアドレスに通知メールを送信することもできます。クラスタプロパティファイルのメールパラメータ。SMTPサーバーの使用の詳細については、次のWebサイトをご覧ください。

<https://access.redhat.com/site/documentation>

ストリーミングレプリケーションの構成

Failover Managerでは、PostgreSQLストリーミングレプリケーションがプライマリノードとスタンバイノード間で構成されている必要があります。フェールオーバーマネージャーは、他の種類のレプリケーションをサポートしていません。

データベースバージョン11（またはそれ以前）では、`-sourcenode` オプションで指定しない限り、スイッチオーバー中に `recovery.conf` ファイルがランダムスタンバイノードから停止したプライマリにコピーされます。スイッチオーバーを実行する前に、スタンバイノードの `recovery.conf` ファイル内のパスが一貫していることを保証する必要があります。 `-sourcenode` オプションの詳細については、[Promoting a Failover Manager Node](#)を参照してください。

データベースバージョン12以降では、スイッチオーバー中に `primary_conninfo` および `restore_command` プロパティがランダムスタンバイノードから停止したプライマリにコピーされます（`-sourcenode` オプションで指定されていない場合）。

**** pg_hba.confファイルを変更します****

プライマリノードとスタンバイノードで `pg_hba.conf` ファイルを変更し、クラスター内のすべてのノード間の通信を許可するエントリを追加する必要があります。次の例は、プライマリノード上の `pg_hba.conf` ファイルに作成される可能性のあるエントリを示しています。

```
# access for itself
host fmdb efm 127.0.0.1/32 md5
# access for standby
host fmdb efm 192.168.27.1/32 md5
# access for witness
host fmdb efm 192.168.27.34/32 md5
```

どこで：

`efm` は、有効なデータベースユーザの名前を指定します。

`fmdb` は、`efm`ユーザが接続できるデータベースの名前を指定します。

デフォルトでは、`pg_hba.conf` ファイルはPostgresインストールの `data` ディレクトリにあります。`pg_hba.conf` ファイルを変更した後、変更を有効にするには、各ノードで構成ファイルをリロードロードする必要があります。次のコマンドを使用できます。

```
# systemctl reload edb-as-x
```

`x` はPostgresバージョンを指定します。

データベースサーバーの自動起動の使用

プライマリノードが再起動した場合は、フェイルオーバーマネージャーは、データベースがプライマリノード上でダウンして検出し、プライマリのロールにスタンバイ・ノードをプロモートことができます。これが発生した場合、（リブートされた）プライマリノードのFailover Managerエージェントは `recovery.conf` ファイルを書き込む機会を得られません。`recovery.conf` ファイルは、データベースサーバの起動を妨げます。これが発生した場合、再起動されたプライマリノードは、2番目のプライマリノードとしてクラスタに結果ま

ず。

これを防ぐために、フェイルオーバーマネージャー・エージェントの自動データベースサーバの前に開始されることを保証。エージェントはアイドルモードでスタートし、クラスター内に既にプライマリがあるかどうかを確認します。プライマリノードがある場合、エージェントは `recovery.conf` または `standby.signal` ファイルが存在することを確認し、データベースは2番目のプライマリとしてスタートしません。

ファイアウォールを介した通信を保証

Linuxファイアウォール（`iptables`）がFailover Managerノードのホストで有効になっている場合、クラスター内のFailover Managerプロセス間のTCP通信を許可するルールをファイアウォール構成に追加する必要があります。例：

```
# iptables -I INPUT -p tcp --dport 7800 -j ACCEPT
/sbin/service iptables save
```

上記のコマンドは、ポート7800を開きます。フェールオーバーマネージャーは、クラスタープロパティファイルで指定されたポートに対応するポートを介して接続します。

データベースユーザに十分な権限があることを確認してください

`efm.properties` ファイルの `db.user` プロパティで指定されたデータベースユーザには、フェールオーバーマネージャーに代わって次の機能を呼び出すための十分な権限が必要です。

```
pg_current_wal_lsn()
```

```
pg_last_wal_replay_lsn()
```

```
pg_wal_replay_resume()
```

```
pg_wal_replay_pause()
```

`reconfigure.num.sync` または `reconfigure.sync.primary` プロパティが `true` に設定されている場合：

- データベースバージョン9.6の場合、`db.user` はスーパーユーザニーズがあります。
- バージョン9.6以降のデータベースバージョンの場合、`db.user` には `pg_reload_conf()` を実行するための `pg_read_all_stats` 権限と権限が必要です。

これらの各機能の詳細については、[PostgreSQL core documentation](#)をご覧ください。

ユーザには、構成変数の値を読み取る権限も必要です。データベーススーパーユーザは、PostgreSQL `GRANT` コマンドを使用して、必要な権限を提供できます。

```
GRANT pg_read_all_settings TO user_name;
```

`pg_read_all_settings` の詳細については、[PostgreSQL core documentation](#)を参照してください。

Installing Failover Manager

</ div>

</ div>

EnterpriseDBリポジトリへのアクセスを許可する資格情報を要求するには、次のEDB ウェブサイトしてください。

<https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf>

RedHatまたはCentOSホスト

システムで認識されないソースによって署名されたRPMパッケージをインストールすると、yumは、ローカルサーバにキーをインポートするパーミッションを求める場合があります。プロンプトが表示され、パッケージソフトが信頼できるソースからのものであることに満足したら、yを入力し、Returnキーを押して続行します。

インストール中に、yumは解決できない依存関係に遭遇する場合があります。存在する場合は、手動で解決する必要がある依存関係のリストが提供されます。

フェイルオーバーマネージャーはルートによってインストールする必要があります。インストールプロセス中に、インストーラは、enterprisedbまたはpostgresが所有するクラスターのFailover Managerサービスを制御するスクリプトを呼び出すための十分な特権を持つefmという名前付けのユーザも作成します。

資格情報を受け取ったら、クラスターの各ノードでEnterpriseDBリポジトリ構成ファイルを作成し、ファイルを変更してアクセスを有効にする必要があります。次の手順は、EnterpriseDBリポジトリへのアクセスに関する詳細情報を提供します。これらの手順は、クラスターの各ノードで実行する必要があります。

RHELまたはCentOS 7 PPCLEホスト

1. 次のコマンドを使用して構成ファイルを作成し、Advance Toolchainをインストールします。

```
rpm --import https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain

cat > /etc/yum.repos.d/advance-toolchain.repo <<EOF
# Begin of configuration file
[advance-toolchain]
name=Advance Toolchain IBM FTP
baseurl=https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/r
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redh
# End of configuration file
EOF
```

2. EDBリポジトリをインストールします。

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.r
```

3. 以下の「USERNAME : PASSWORD」をEDBリポジトリのユーザー名とパスワードに置き換えます。

```
sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
```

4. EPELリポジトリをインストールします。

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.no
```

5. RHEL 7 PPCLEで、追加のリポジトリを有効にしてEPEL依存関係を解決します。

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-e
```

6. 選択したパッケージをインストールします。

```
yum -y install edb-efm42
```

RHELまたはCentOS 7ホスト

1. リポジトリ構成ファイルを作成するには、スーパーユーザ特権を引き受けて、次のコマンドを呼び出します。

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.r
```

2. 以下の「USERNAME : PASSWORD」をEDBリポジトリのユーザー名とパスワードに置き換えます。

```
sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
```

3. EPELリポジトリをインストールします。

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.no
```

4. RHEL 7では、追加のリポジトリを有効にして依存関係を解決します。

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-e
```

5. 選択したパッケージをインストールします。

```
yum -y install edb-efm42
```

RHELまたはCentOS 8ホスト

1.リポジトリ構成ファイルを作成するには、スーパーユーザ特権を引き受けて、次のコマンドを呼び出します。

```
dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.r
```

2.以下の「USERNAME : PASSWORD」をEDBリポジトリのユーザー名とパスワードに置き換えます。

```
sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
```

3. EPELリポジトリをインストールします。

- CentOS 8で

```
dnf -y install epel-release
```

- RHEL 8で

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.no
```

4.追加のリポジトリを有効にして、依存関係を解決します。

- CentOS 8で

```
dnf config-manager --set-enabled PowerTools
```

- RHEL 8で

```
ARCH=$( /bin/arch ) subscription-manager repos --enable "codeready-builder-fo
```

5.ビルトインのPostgreSQLモジュールを無効にします。

```
dnf -qy module disable postgresql
```

6.選択したパッケージをインストールします。

```
dnf -y install edb-efm42
```

DebianまたはUbuntuホスト

Failover Managerをインストールするには、EnterpriseDBリポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格証明をリクエストするには、次のEnterpriseDBウェブサイトしてください。

< <https://www.enterprisedb.com/> ユーザ/ ログイン >

次の手順では、EnterpriseDB aptリポジトリを使用してフェールオーバーマネージャーをインストールします。

Debianホスト

1.スーパーユーザ特権を想定します。

- Debian 9で

```
sh -c 'echo "deb https://username:password@apt.enterprisedb.com/$(lsb_release
```

- Debian 10で

```
sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)
sh -c 'echo "machine apt.enterprisedb.com login <username> password <password
```

3.安全なAPTリポジトリのサポートをシステムに追加します。

```
apt-get install apt-transport-https
```

4. EDB署名キーを追加しキー。

```
wget -q -O - https://<username>:<password>@apt.enterprisedb.com/edb-deb.gpg.k
```

5.リポジトリのメタデータを更新します。

```
apt-get update
```

6.フェールオーバーマネージャーをインストールします。

```
apt-get -y install edb-efm42
```

Ubuntuホスト

1.スーパーユーザ特権を想定します。

- Ubuntu 18.04で

```
sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)
sh -c 'echo "machine apt.enterprisedb.com login <username> password <password>
```

3.安全なAPTリポジトリのサポートをシステムに追加します。

```
apt-get -y install edb-efm42
```

SLESホスト

Failover Managerをインストールするには、EnterpriseDBリポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格証明をリクエストするには、次のEnterpriseDBウェブサイトしてください。

<https://www.enterprisedb.com/repository-access-request>

zypperパッケージマネージャを使用して、SLES 12ホストにFailover Managerエージェントをインストールできます。zypperは、パッケージをインストールするときにパッケージの依存関係を満たそうとしますが、EnterpriseDBでホストされていない特定のリポジトリへのアクセスが必要です。

1.フェールオーバーマネージャーをインストールする前に、スーパーユーザ権限を引き受け、ファイアウォールを停止する必要があります。次に、以下のコマンドを使用して、EnterpriseDBリポジトリをシステムに追加します。

```
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo
```

2.コマンドは、`/etc/zypp/repos.d`ディレクトリにリポジトリ構成ファイルを作成します。次に、次のコマンドを使用してSLESホストのメタデータをリフレッシュし、EnterpriseDBリポジトリを含めます。

```
zypper refresh
```

- プロンプトが表示されたら、リポジトリの資格情報を指定し、指定されたキーを常に信頼するように指定し、メタデータを更新してEnterpriseDBリポジトリを含めます。

3.また、SUSE ConnectとSUSE Package Hub拡張をSLESホストに追加し、ホストをSUSEに登録して、SUSEリポジトリへのアクセスを許可する必要があります。次のコマンドを使用します。

```
zypper install SUSEConnect
SUSEConnect -r <registration_number> -e <user_id>
SUSEConnect -p PackageHub/12.4/x86_64
SUSEConnect -p sle-sdk/12.4/x86_64
```

4. SUSEConnectをインストールしてホストをSUSEに登録し、SUSEリポジトリへのアクセスを許可します。

```
zypper addrepo https://download.opensuse.org/repositories/Apache:/Modules/SLE
```

5. Javaベースのコンポーネント用のOpenJDK（バージョン1.8）をインストールします。

```
zypper -n install java-1_8_0-openjdk
```

- 6.これで、zypperユーティリティを使用してFailover Managerエージェントをインストールできます。

```
zypper -n install edb-efm42
```

インストール後のタスクの実行

フェールオーバーマネージャーをモニタ、`enterprisedb` または `postgres` 以外のユーザが所有するクラスターを監視している場合は、[Extending Failover Manager Permissions](#)を参照してください。

クラスターの各ノードにインストールした後、以下を行う必要があります。

- 1.各ノードでを変更します。各ノードでを変更します。必要に応じて、仮想IPアドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。4. クラスターの各ノードでエージェントを起動します。サービスの制御の詳細については、を参照してください。

インストール場所

コンポーネントは次の場所にインストールされます。

Component	Location
Executables	/usr/edb/efm-4.2/bin
Libraries	/usr/edb/efm-4.2/lib
Cluster configuration files	/etc/edb/efm-4.2

Component	Location
Logs	/var/log/efm- 4.2
Lock files	/var/lock/efm-4.2
Log rotation file	/etc/logrotate.d/efm-4.2
sudo configuration file	/etc/sudoers.d/efm-42

Configuring Failover Manager

</ div>

構成可能なフェールオーバーマネージャーのプロパティは、ユーザーが変更可能な2つのファイルで指定されます。

- [efm.properties](#)
- [efm.nodes](#)

cluster_properties encrypt_database_password cluster_members extends_efm_permissions
using_vip_addresses

</ div>

The Cluster Properties File

</ div>

フェールオーバーマネージャークラスター内の各ノードには、それが存在する個々のノードのプロパティを含むプロパティファイル（デフォルトでは `efm.properties` 名前付け名前）があります。Failover Managerインストーラは、`/etc/edb/efm-4.2` ディレクトリに `efm.properties.in` 名前付けのプロパティファイルのファイルテンプレートを作成します。

Failover Managerのインストールが完了したら、ファイルの内容を変更前にテンプレートの作業用コピーをmakeする必要があります。

```
# cp /etc/edb/efm-4.2/efm.properties.in /etc/edb/efm-4.2/efm.properties
```

テンプレートファイルをコピーした後、ファイルの所有者を `efm` に変更します。

```
# chown efm:efm efm.properties
```

Note * デフォルトでは、フェールオーバーマネージャーはクラスタープロパティファイルの名前付けが `efm.properties` であると想定しています。あなたは `efm.properties` 以外の何かをプロパティファイルに名前を名前場合は、別の名前を使用するフェイル

オーバーマネージャー指示するサービスまたはユニットファイルを変更する必要があります。

クラスタープロパティファイルを作成したら、必要に応じて設定パラメータ値を追加（または変更）します。各プロパティの詳細については、[Specifying Cluster Properties](#)を参照してください。

プロパティファイルの所有者は `root` です。Failover Managerサービススクリプトは、`/etc/edb/efm-4.2` ディレクトリ内のファイルを見つけることを想定しています。プロパティファイルを別の場所に移動する場合は、新しい場所を指定するシンボリックリンクを作成する必要があります。

Note * プロパティファイルで被参照れるすべてのユーザスクリプトは、Failover Managerユーザとして呼び出されます。

</div>

クラスタープロパティの指定

クラスタープロパティファイルにリストされているプロパティを使用して、Failover Managerクラスターの接続プロパティと動作を指定できます。プロパティ設定の変更は、フェールオーバーマネージャーの起動時に適用されます。プロパティ値を変更した場合は、フェールオーバーマネージャーをリスタートで変更を適用する必要があります。

プロパティ値では大文字と小文字が区別されます。Postgresはパラメータ値に引用符付き文字列を使用しますが、Failover Managerではプロパティ値に引用符付き文字列を使用できないことに注意してください。例、あなたのようにPostgresの設定パラメータでIPアドレスを指定するかもしれないが。

```
listen_addresses='192.168.2.47'
```

フェールオーバーマネージャーでは、値を引用符で囲まないでください。

```
bind.address=192.168.2.54:7800
```

`efm.properties` ファイルのプロパティを使用して、Failover Managerの接続、管理、および操作の詳細を指定します。

凡例：次の表：

- `A`：プライマリノードまたはスタンバイノードで必要
- `W`：監視ノードで必須
- `Y`：はい

Property Name	A	W	Default Value	Comments
<code>db.user</code>	Y	Y		データベースのユーザー名

Property Name	A	W	Default Value	Comments
db.password.encrypted	Y	Y		「efm encrypt」を使用して暗号化されたパスワード
db.port	Y	Y		この値はすべてのエージェントで同じでなければなりません
db.database	Y	Y		データベース名前
db.service.owner	Y			データベースの\$PGDATAディレクトリの所有者。データベース
db.service.name				データベースをサービスとして実行する場合は必須
db.bin	Y			'/usr/edb/as12/bin'などのpg_controldata/pg_ctlコマンドを含むディレクトリ
db.data.dir	Y			クエリー'show data_directory;'の出力と同じ
db.config.dir				クエリー'show config_file;'の出力と同じです。 * db.data.dir *と同じでない場合に指定する必要があります
jdbc.sslmode	Y	Y	disable	note を参照してください
user.email				この値は、すべてのエージェントで同じでなければなりません。通知スクリプトを使用する場合は空白のままにすることができます
from.email			efm@localhost	デフォルトの efm@localhost を使用するには空白のままにします

Property Name	A	W	Default Value	Comments
notification.level	Y	Y	INFO	list of notifications を参照してください
notification.text.prefix				NA
				NA
script.notification				ユーザの場合は必須。 メールプロパティは使用されません。両方のパラメーターを一緒に使用できます
bind.address	Y	Y		例：< ip_address>：< port>
external.address				例：< ip_address / hostname>
admin.port	Y	Y	7809	デフォルトポートがすでに使用されている場合は変更します
is.witness	Y	Y		説明を見る
local.period	Y		10	NA
				NA
local.timeout	Y		60	NA
				NA
local.timeout.final	Y		10	NA
				NA
remote.timeout	Y	Y	10	NA
				NA
node.timeout	Y	Y	50	この値はすべてのエージェントで同じでなければなりません
encrypt.agent.messages	Y	Y	false	この値はすべてのエージェントで同じでなければなりません
enable.stop.cluster			true	この値はすべてのエージェントで同じでなければなりません

Property Name	A	W	Default Value	Comments
stop.isolated.primary	Y		true	この値はすべてのエージェントで同じでなければなりません
stop.failed.primary	Y		true	NA
				NA
primary.shutdown.as.failure	Y	Y	false	NA
				NA
update.physical.slots.period	Y		0	NA
				NA
ping.server.ip	Y	Y	8.8.8.8	NA
				NA
ping.server.command	Y	Y	/bin/ping -q -c3 -w5	NA
				NA
auto.allow.hosts	Y	Y	false	NA
				NA
stable.nodes.file	Y	Y	false	NA
				NA
db.reuse.connection.count	Y		0	NA
				NA
auto.failover	Y	Y	true	NA
				NA
auto.reconfigure	Y		true	この値はすべてのエージェントで同じでなければなりません
promotable	Y		true	NA
				NA
use.replay.tiebreaker	Y	Y	true	この値はすべてのエージェントで同じでなければなりません
standby.restart.delay			0	NA

Property Name	A	W	Default Value	Comments
				NA
application.name				オリジナルのプライマリデータベースをスタンバイとして起動する前に、primary_conninfoエントリーのapplication_name部分をこのプロパティ値に置き換えるように設定します。
restore.command				例：restore.command = scp db_service_owner>@%h： archive_path> /%f%p
reconfigure.num.sync	Y		false	NA
				NA
reconfigure.num.sync.max				NA
				NA
reconfigure.sync.primary	Y		false	NA
				NA
minimum.standbys	Y	Y	0	この値はすべてのノードで同じでなければなりません
priority.standbys				NA
				NA
recovery.check.period	Y		1	NA
				NA
restart.connection.timeout			60	NA
				NA
auto.resume.period	Y		0	NA
				NA
virtual.ip			(see virtual.ip.single)	VIPを指定しない場合は空白のままにします

Property Name	A	W	Default Value	Comments
virtual.ip.interface				VIPを指定する場合は必須
virtual.ip.prefix				VIPを指定する場合は必須
virtual.ip.single	Y	Y	Yes	この値はすべてのノードで同じでなければなりません
check.vip.before.promotion	Y	Y	Yes	NA
				NA
pgpool.enable			false	NA
				NA
pcp.user				pgpool.enableが真に設定されている場合は必須
pcp.host				pgpool.enableが真に設定されている場合、この値はすべてのエージェントで同じでなければなりません
pcp.port				pgpool.enableが真に設定されている場合、この値はすべてのエージェントで同じでなければなりません
pcp.pass.file				pgpool.enableが真に設定されている場合は必須
pgpool.bin				pgpool.enableが真に設定されている場合は必須
script.load.balancer.attach				例：スクリプト。ロード.balancer.attach = /<パス> /<attach_script>%h%t
script.load.balancer.detach				例：スクリプト。ロード.balancer.detach = /<パス> /<detach_script>%h%t

Property Name	A	W	Default Value	Comments
detach.on.agent.failure			true	実行中のプライマリデータベースをアタッチしたままにする場合は、falseに設定します。
script.fence				例：script.fence = /<パス> /<script_name>%p%f
script.post.promotion				例：script.post.promotion = /<パス> /<script_name>%f%p
script.resumed				例：script.resumed = /<パス> /<script_name>
script.db.failure				例：script.db.failure = /<パス> /<script_name>
script.primary.isolated				例：スクリプト。プライマリ.isolated = /<パス> /<script_name>
script.remote.pre.promotion				例：スクリプト。 リモート.pre.promotion = /<パス> /<script_name>%p
script.remote.post.promotion				例：スクリプト。 リモート.post.promotion = /<パス> /<script_name>%p
script.custom.monitor				例：script.custom。 モニタ= /<パス> /<script_name>
custom.monitor.interval				カスタムモニタリングスクリプトが指定されている場合は必須
custom.monitor.timeout				カスタムモニタリングスクリプトが指定されている場合は必須
custom.monitor.safe.mode				カスタムモニタリングスクリプトが指定されている場合は必須

Property Name	A	W	Default Value	Comments
sudo.command	Y	Y	sudo	NA
				NA
sudo.user.command	Y	Y	sudo -u %u	NA
				NA
lock.dir				指定しない場合、デフォルトは '/var/ロック/efm-<バージョン>'
log.dir				指定しない場合、デフォルトは '/var/ログ/efm-<バージョン>' になります。バージョン>'
syslog.host			localhost	NA
				NA
syslog.port			514	NA
				NA
syslog.protocol				NA
				NA
syslog.facility			UDP	NA
				NA
file.log.enabled	Y	Y	true	NA
				NA
syslog.enabled	Y	Y	false	NA
				NA
jgroups.loglevel			info	NA
				NA
efm.loglevel			info	NA
				NA
jvm.options			-Xmx128m	NA
				NA

クラスターのプロパティ

</ div>

</ div>

</ div>

</ div>

次のプロパティを使用して、フェールオーバーマネージャークラスターの接続の詳細を指定します。

```
# The value for the password property should be the output from
# 'efm encrypt' -- do not include a cleartext password here. To
# prevent accidental sharing of passwords among clusters, the
# cluster name is incorporated into the encrypted password. If
# you change the cluster name (the name of this file), you must
# encrypt the password again with the new name.
# The db.port property must be the same for all nodes.
db.user=
db.password.encrypted=
db.port=
db.database=
```

指定した `db.user` には、フェールオーバーマネージャに代わって選択したPostgreSQLコマンドを呼び出すための十分な権限が必要です。詳細については、[Prerequisites](#)を参照してください。

データベースユーザのパスワードの暗号化については、[Encrypting Your Database Password](#)を参照してください。

</ div>

`db.service.owner` プロパティを使用して、フェールオーバーマネージャによって管理されているクラスターを所有するオペレーティングシステムユーザの名前を指定します。このプロパティは、専用の監視ノードでは必要ありません。

```
# This property tells EFM which OS user owns the $PGDATA dir for
# the 'db.database'. By default, the owner is either 'postgres'
# for PostgreSQL or 'enterprisedb' for EDB Postgres Advanced
# Server. However, if you have configured your db to run as a
# different user, you will need to copy the /etc/sudoers.d/efm-XX
# conf file to grant the necessary permissions to your db owner.
#
# This username must have write permission to the
# 'db.data.dir' specified below.
db.service.owner=
```

</ div>

サービスの開始または停止時にサービスまたはsystemctlコマンドを使用する場合は、`db.service.name` プロパティでデータベースサービスの名前を指定しサービス。

```
# Specify the proper service name in order to use service commands
# rather than pg_ctl to start/stop/restart a database. For example, if
# this property is set, then 'service <name> restart' or 'systemctl
# restart <name>'
# (depending on OS version) will be used to restart the database rather
# than pg_ctl.
# This property is required if running the database as a service.
db.service.name=
```

</div>

データベースサービスを開始または停止するたびに、同じサービス制御メカニズム (pg_ctl、サービス、またはsystemctl) を使用スタート必要があります。 pg_ctl プログラムを使用してサービスを制御する場合は、 db.bin プロパティで pg_ctl プログラムの場所を指定します。

```
# Specify the directory containing the pg_controldata/pg_ctl commands,
# for example:
# /usr/edb/as12/bin. Unless the db.service.name property is used, the
# pg_ctl command is used to start/stop/restart databases as needed
# after a failover or switchover. This property is required.
db.bin=
```

</div>

db.data.dir プロパティを使用して、昇格中にクラスタのプライマリノードでリカバリファイルが書き込まれる場所を指定します。このプロパティは、プライマリノードとスタンバイノードが必要です。専用の監視ノードでは必要ありません。

```
# For database version 12 and up, this is the directory where a
# standby.signal file will exist for a standby node. For previous
# versions, this is the location of the db recovery.conf file on
# the node.
# After a failover, the recovery.conf files on remaining standbys are
# changed to point to the new primary db (a copy of the original is made
# first). On a primary node, a recovery.conf file will be written during
# failover and promotion to ensure that the primary node can not be
# restarted as the primary database.
# This corresponds to database environment variable PGDATA and should
# be same as the output of query 'show data_directory;' on respective
# database.
db.data.dir=
```

</div>

db.data.dir プロパティを使用して、昇格中にクラスタのプライマリノードでリカバリファイルが書き込まれる場所を指定します。このプロパティは、プライマリノードとスタンバイノードが必要です。専用の監視ノードでは必要ありません。

```
# For database version 12 and up, this is the directory where a
# standby.signal file will exist for a standby node. For previous
# versions, this is the location of the db recovery.conf file on
# the node.
# After a failover, the recovery.conf files on remaining standbys are
# changed to point to the new primary db (a copy of the original is made
# first). On a primary node, a recovery.conf file will be written during
# failover and promotion to ensure that the primary node can not be
# restarted as the primary database.
# This corresponds to database environment variable PGDATA and should
# be same as the output of query 'show data_directory;' on respective
# database.
db.data.dir=
```

</div>

`recovery.conf` または `standby.signal` ファイルと同じディレクトリに格納されていない場合は、`db.config.dir` プロパティを使用してデータベース構成ファイルの場所を指定します。これは、Advanced ServerまたはPostgreSQLインストールの `config_file` パラメータディレクトリで指定された値である必要があります。この値は、データベースを停止、起動、または再起動するときにPostgres `data` ディレクトリの場所として使用されます。

```
# Specify the location of database configuration files if they are
# not contained in the same location as the recovery.conf or
# standby.signal file. This is most likely the case for Debian
# installations. The location specified will be used as the -D value
# (the location of the data directory for the cluster) when calling
# pg_ctl to start or stop the database. If this property is blank,
# the db.data.dir location specified by the db.data.dir property will
# be used. This corresponds to the output of query 'show config_file;'
# on respective database.
db.config.dir=
```

データベース構成ファイルの詳細については、[PostgreSQL website](#)をご覧ください。

</div>

`jdbc.sslmode` プロパティを使用して、SSL接続を使用するようにフェールオーバーマネージャーに指示します。デフォルトでは、SSLは無効になっています。

```
# Use the jdbc.sslmode property to enable ssl for EFM
# connections. Setting this property to anything but 'disable'
# will force the agents to use 'ssl=true' for all JDBC database
# connections (to both local and remote databases).
# Valid values are:
#
# disable - Do not use ssl for connections.
# verify-ca - EFM will perform CA verification before allowing
# the certificate.
# require - Verification will not be performed on the server
# certificate.
jdbc.sslmode=disable
```

</div>

Note * `jdbc.sslmode` の値を `verify-ca` に設定し、証明書のバリデーションにJavaトラストストアを使用する場合は、次の値を設定する必要があります。

- `jdbc.properties=sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory`

SSLの構成と使用の詳細については、以下を参照してください。

< <https://www.postgresql.org/docs/current/ 静的/ssl-tcp.html> >

そして

< <https://jdbc.postgresql.org/ 文書/head/ssl.html> >

</div>

`user.email` プロパティを使用して、Failover Managerから送信された通知を受信するメールアドレス（またはマルチプルのメールアドレス）を指定します。

```
# Email address(es) for notifications. The value of this
# property must be the same across all agents. Multiple email
# addresses must be separated by space. If using a notification
# script instead, this property can be left blank.
user.email=
```

</div>

`from.email` プロパティは、Failover Managerからのメール通知で送信者のアドレスとして使用される値を指定します。次のことができます。

- デフォルト値（`efm@localhost`）を使用するには、`from.email` を空白のままにします。
- メールアドレスのカスタム値を指定します。
- `%h` プレースホルダを使用してノードホストの名前を表すカスタムメールアドレスを指定します（例：）。プレースホルダは、Linuxホスト名ユーティリティによって返されるホストの名前に置き換えられます。

通知の詳細については、[Notifications](#)を参照してください。

```
# Use the from.email property to specify the from email address that
# will be used for email notifications. Use the %h placeholder to
# represent the name of the node host (e.g. example@%h). The
# placeholder will be replaced with the name of the host as returned
# by the hostname command.
# Leave blank to use the default, efm@localhost.
from.email=
```

</div>

notification.level プロパティを使用して、フェールオーバーマネージャーがユーザ通知を送信する最小の重大度レベル、または通知スクリプトが呼び出されるタイミングを指定します。通知の完全なリストについては、[Notifications](#)をご覧ください。

```
# Minimum severity level of notifications that will be sent by
# the agent. The minimum level also applies to the notification
# script (below). Valid values are INFO, WARNING, and SEVERE.
# A list of notifications is grouped by severity in the user's
# guide.
notification.level=INFO
```

</div>

notification.text.prefix プロパティを使用して、すべての通知の先頭に追加するテキストを指定します。

```
# Text to add to the beginning of every notification. This could
# be used to help identify what the cluster is used for, the role
# of this node, etc. To use multiple lines, add a backslash \ to
# the end of a line of text. To include a newline use \n.
# Example:
# notification.text.prefix=Development cluster for Example dept.\n\
# Used by Dev and QA \
# See Example group for questions.
notification.text.prefix=
```

</div>

script.notification プロパティを使用して、通知サービスとして機能するユーザー指定のスクリプトへのパスを指定します。スクリプトには、メッセージのサブジェクトとメッセージ本文が渡されます。スクリプトは、フェールオーバーマネージャーがユーザ通知を生成するたびに呼び出されます。

```
# Absolute path to script run for user notifications.
#
# This is an optional user-supplied script that can be used for
# notifications instead of email. This is required if not using
# email notifications. Either/both can be used. The script will
# be passed two parameters: the message subject and the message
# body.
script.notification=
```

</div>

bind.address プロパティは、フェールオーバーマネージャークラスターの現在のノード上のエージェントのIPアドレスとポート番号を指定します。

```
# This property specifies the ip address and port that jgroups
# will bind to on this node. The value is of the form
# <ip>:<port>.
# Note that the port specified here is used for communicating
# with other nodes, and is not the same as the admin.port below,
# used only to communicate with the local agent to send control
# signals.
# For example, <provide_your_ip_address_here>:7800
bind.address=
```

</div>

external.address プロパティを使用して、NAT環境内の他のすべてのFailover Manager エージェントとの通信に使用するIPアドレスまたはホスト名を指定します。

```
# This is the ip address/hostname to be used for communication with all
# other Failover Manager agents. All traffic towards this address
# should be routed by the network to the bind.address of the node.
# The value is in the ip/hostname format only. This address will be
# used in scenarios where nodes are on different networks and broadcast
# an IP address other than the bind.address to the external world.
external.address=
```

</div>

admin.port プロパティを使用して、フェールオーバーマネージャークラスターが管理コマンドをリスンするポートを指定します。

```
# This property controls the port binding of the administration
# server which is used for some commands (ie cluster-status). The
# default is 7809; you can modify this value if the port is
# already in use.
admin.port=7809
```

</div>

`is.witness` プロパティを真に設定して、現在のノードが監視ノードであることを示しノード。 `is.witness`が真の場合、ローカルエージェントはローカルデータベースが実行されているかどうかを確認しません。

```
# Specifies whether or not this is a witness node. Witness nodes
# do not have local databases running.
is.witness=
```

Postgres `pg_is_in_recovery()` ファンクションは、データベースのリカバリ状態を報告するブールファンクションです。データベースがリカバリしていない場合は、データベースがリカバリ、または偽である場合、ファンクションが戻る `true`。エージェントが起動すると、ローカルデータベースに接続し、`pg_is_in_recovery()` ファンクションを呼び出します。サーバーが真と応答した場合、エージェントはスタンバイのロールを引き受けます。サーバーがfalseと応答した場合、エージェントはプライマリのロールを引き受けます。ローカルデータベースがない場合、エージェントはアイドル状態になります。

Note * `is.witness` が `true` の場合、フェールオーバーマネージャーはリカバリ状態をチェックしません。

</div>

</div>

</div>

次のプロパティは、ローカルサーバに適用されるプロパティを指定します。

- `local.period` プロパティは、データベースサーバへの接続を試行する間隔を秒単位で指定します。
- `local.timeout` プロパティは、エージェントがローカルデータベースサーバーからの肯定的な応答を待つ時間を指定します。
- `local.timeout.final` プロパティは、上記の以前のチェックが現在のノード上のデータベースサーバへの接続に失敗した後、エージェントが待機する時間を指定します。 `local.timeout.final` プロパティで指定された秒数内にデータベースから応答を受信しなかった場合、データベースは失敗したと見なされます。

例、これらのプロパティのデフォルト値が与えられ、ローカルデータベースのチェックは10秒ごとに一度起こります。ローカルデータベースへの接続の試行が60秒以内に正に戻らない場合、フェールオーバーマネージャーはデータベースへの接続の最終試行を行います。応答が10秒以内に受信されない場合、Failover Managerはデータベース障害を宣言し、ユーザにリストされている管理者に通知します。メールプロパティ。これらのプロパティは、専用の監視ノードでは必要ありません。

```
# These properties apply to the connection(s) EFM uses to monitor
# the local database. Every 'local.period' seconds, a database
# check is made in a background thread. If the main monitoring
# thread does not see that any checks were successful in
# 'local.timeout' seconds, then the main thread makes a final
# check with a timeout value specified by the
# 'local.timeout.final' value. All values are in seconds.
# Whether EFM uses single or multiple connections for database
# checks is controlled by the 'db.reuse.connection.count'
# property.
local.period=10
local.timeout=60
local.timeout.final=10
```

必要に応じて、ビジネスモデルに合わせてこれらの値を変更する必要があります。

</div>

`remote.timeout` プロパティを使用して、エージェントがリモートデータベースサーバーからの応答を待機する秒数（つまり、フェイルオーバーを実行する前にプライマリデータベースが実際にダウンしていることを確認するためにスタンバイエージェントが待機する時間）を指定します。`remote.timeout` プロパティ値は、エージェント間の通信のタイムアウト値を指定します。クラスタプロパティファイルの他のタイムアウトプロパティは、エージェントからデータベースへの通信の値を指定します。

```
# Timeout for a call to check if a remote database is responsive.
# For example, this is how long a standby would wait for a
# DB ping request from itself and the witness to the primary DB
# before performing failover.
remote.timeout=10
```

</div>

ノードに障害が発生したかどうかを決定する際に、エージェントがノードからの応答を待つ秒数を指定するには、`node.timeout` プロパティを使用します。

```
# The total amount of time in seconds to wait before determining
# that a node has failed or been disconnected from this node.
#
# The value of this property must be the same across all agents.
node.timeout=50
```

</div>

`encrypt.agent.messages` プロパティを使用して、エージェント間で送信されるメッセージを暗号化する必要があるかどうかを指定します。

```
# Set to true to encrypt messages that are sent between agents.
# This property must be the same on all agents or else the agents
# will not be able to connect.
encrypt.agent.messages=false
```

</div>

`enable.stop.cluster` プロパティを使用して、`stop-cluster` コマンドを有効または無効にします。このコマンドは、一部の環境では便利な場合がありますが、意図せずに呼び出されると問題が発生する可能性があります。Eager Failoverモードでは、コマンドはフェイルオーバーなしでPostgresを停止します。

```
# Whether or not the 'efm stop-cluster <cluster name>' command is enabled.
# Set to false to disable the command, in which case all Failover
# Manager agents must be stopped individually. Note that stopping each
# agent separately will change the .nodes files on remaining agents
# unless stable.nodes.file is also true. This property value must
# be the same on all agents if set. The default is true if not set.
enable.stop.cluster=true
```

</div>

プライマリエージェントが分離されていることを検出した場合、`stop.isolated.primary` プロパティを使用して、フェールオーバーマネージャーにデータベースをシャットダウンするよう指示します。真（デフォルト）の場合、Failover Managerは `script.primary.isolated` プロパティで指定されたスクリプトを呼び出す前にデータベースを停止します。

```
# Shut down the database after a primary agent detects that it has
# been isolated from the majority of the efm cluster. If set to
# true, efm will stop the database before running the
# 'script.primary.isolated' script, if a script is specified.
stop.isolated.primary=true
```

</div>

`stop.failed.primary` プロパティを使用して、フェールオーバーマネージャーに、プライマリデータベースに到達できない場合にシャットダウンを試行するよう指示しデータベース。true の場合、フェールオーバーマネージャーは、データベースのシャットダウンを試行した後、`script.db.failure` プロパティで指定されたスクリプトを実行します。

```
# Attempt to shut down a failed primary database after EFM can no
# longer connect to it. This can be used for added safety in the
# case a failover is caused by a failure of the network on the
# primary node.
# If specified, a 'script.db.failure' script is run after this attempt.
stop.failed.primary=true
```

</div>

`primary.shutdown.as.failure` パラメータを使用して、プライマリノード上のFailover Managerエージェントのシャットダウンを障害として扱う必要があることを示します。このパラメータが `true` に設定され、プライマリエージェントが（何らかの理由で）停止した場合、クラスターはプライマリノードのデータベースが実行されているかどうかを確認しようとしています。

- データベースに到達すると、エージェントのステータスを通知する通知が送信されます。
- データベースに到達しない場合、フェイルオーバーが発生します。

```
# Treat a primary agent shutdown as an agent failure. This can be set
# to true to treat a primary agent shutdown as a failure situation.
# Caution should be used when using this feature, as it could
# cause an unwanted promotion in the case of performing primary
# database maintenance.
# Please see the user's guide for more information.
primary.shutdown.as.failure=false
```

`primary.shutdown.as.failure` プロパティは、プライマリノードの偶発的なシャットダウンなどの障害ではなく、ユーザエラーをキャッチするためのものです。ユーザがプライマリフェールオーバーマネージャエージェントを停止したように、ノードの適切なシャットダウンがクラスターの残りの部分に表示されることがあります（例、プライマリデータベースのメンテナンスを実行するため）。`primary.shutdown.as.failure` プロパティを `true` に設定した場合、メンテナンスの実行時には注意が必要です。

`primary.shutdown.as.failure` が `true` であるときにプライマリデータベースのメンテナンスを実行するには、プライマリエージェントを停止し、プライマリエージェントが失敗したがデータベースがまだ実行されているという通知を受信するまで待機する必要があります。その後、プライマリデータベースを停止しても安全です。または、`stop-cluster` コマンドを使用して、障害チェックを実行せずにすべてのエージェントを停止できます。

</div>

`update.physical.slots.period` プロパティを使用して、データベースバージョン12以降のスロットアドバンス頻度を定義します。`update.physical.slots.period` がゼロ以外の値に設定されている場合、プライマリエージェントは `update.physical.slots.period` 秒ごとに物理的レプリケーションスロットの現在の `restart_lsn` を読み取り、`pg_current_wal_lsn` および `primary_slot_name` でこの情報を送信します（`postgresql.conf`ファイルで設定されている場合）スタンバイに。物理的スロットがまだ存在しない場合、このパラメータをゼロ以外の値に設定すると、スロットが作成され、これらのスロットの `restart_lsn` parameter が更新されます。昇格できないスタンバイは新しいスロットを作成しませんが、存在する場合は更新します。

```
# Period in seconds between having the primary agent update promotable
# standbys with physical replication slot information so that
# the cluster will continue to use replication slots after a failover.
# Set to zero to turn off.
update.physical.slots.period=0
```

</div>

`ping.server.ip` プロパティを使用して、フェールオーバーマネージャーがネットワーク接続に問題がないことを確認するために使用できるサーバーのIPアドレスを指定します。

```
# This is the address of a well-known server that EFM can ping
# in an effort to determine network reachability issues. It
# might be the IP address of a nameserver within your corporate
# firewall or another server that *should* always be reachable
# via a 'ping' command from each of the EFM nodes.
#
# There are many reasons why this node might not be considered
# reachable: firewalls might be blocking the request, ICMP might
# be filtered out, etc.
#
# Do not use the IP address of any node in the EFM cluster
# (primary, standby, or witness) because this ping server is meant
# to provide an additional layer of information should the EFM
# nodes lose sight of each other.
#
# The installation default is Google's DNS server.
ping.server.ip=8.8.8.8
```

</div>

`ping.server.command` プロパティを使用して、ネットワーク接続のテストに使用するコマンドを指定します。

```
# This command will be used to test the reachability of certain
# nodes.
#
# Do not include an IP address or hostname on the end of
# this command - it will be added dynamically at runtime with the
# values contained in 'virtual.ip' and 'ping.server.ip'.
#
# Make sure this command returns reasonably quickly - test it
# from a shell command line first to make sure it works properly.
ping.server.command=/bin/ping -q -c3 -w5
```

</div>

`auto.allow.hosts` プロパティを使用して、許可ホストリストの更新を開始した最初のノードの`nodes`ファイルで指定されたアドレスを使用するようにサーバーに指示します。このプロパティを有効にすると（`auto.allow.hosts` を真に設定）、クラスターの起動を簡素化で

きます。

```
# Have the first node started automatically add the addresses
# from its .nodes file to the allowed host list. This will make
# it faster to start the cluster when the initial set of hosts
# is already known.
auto.allow.hosts=false
```

</ div>

`stable.nodes.file` プロパティを使用して、ノードがクラスターに参加または離脱するときにノードファイルを書き換えないようにサーバーに指示します。このプロパティは、不変のIPアドレスを持つクラスターで最も役立ちます。

```
# When set to true, EFM will not rewrite the .nodes file whenever
# new nodes join or leave the cluster. This can help starting a
# cluster in the cases where it is expected for member addresses
# to be mostly static, and combined with 'auto.allow.hosts' makes
# startup easier when learning failover manager.
stable.nodes.file=false
```

</ div>

`db.reuse.connection.count` プロパティを使用すると、管理者は、フェールオーバーマネージャーが同じデータベースコネクションを再利用してデータベースの状態を確認する回数を指定できます。デフォルト値は0です。これは、Failover Managerが毎回新しい接続を作成することを示します。このプロパティは、専用の監視ノードでは必要ありません。

```
# This property controls how many times a database connection is
# reused before creating a new one. If set to zero, a new
# connection will be created every time an agent pings its local
# database.
db.reuse.connection.count=0
```

</ div>

`auto.failover` プロパティにより、自動フェイルオーバーが有効になります。デフォルトでは、auto。フェイルオーバーは真に設定されます。

```
# Whether or not failover will happen automatically when the primary
# fails. Set to false if you want to receive the failover notifications
# but not have EFM actually perform the failover steps.
# The value of this property must be the same across all agents.
auto.failover=true
```

</ div>

`auto.reconfigure` プロパティを使用して、プライマリスタンバイがプライマリに昇格した

後、残りのスタンバイサーバーの自動再構成を有効または無効にするようフェールオーバーマネージャーに指示します。プロパティを `true` に設定して自動再構成（デフォルト）を有効にするか、`false` に自動再構成を無効にします。このプロパティは、専用の監視ノードでは必要ありません。Advanced ServerまたはPostgreSQLバージョン11以前を使用している場合、`recovery.conf` ファイルは再構成プロセス中にバックアップされます。

```
# After a standby is promoted, Failover Manager will attempt to
# update the remaining standbys to use the new primary. For database
# versions before 12, Failover Manager will back up recovery.conf.
# Then it will change the host parameter of the primary_conninfo entry
# in recovery.conf or postgresql.auto.conf, and restart the database.
# The restart command is contained in either the efm_db_functions or
# efm_root_functions file; default when not running db as an os
# service is: "pg_ctl restart -m fast -w -t <timeout> -D <directory>"
# where the timeout is the local.timeout property value and the
# directory is specified by db.data.dir. To turn off
# automatic reconfiguration, set this property to false.
auto.reconfigure=true
```

Note * `primary_conninfo` は、keyword = valueペアのスペース区切りリストです。

</div>

`promotable` プロパティを使用して、ノードを昇格させないことを示します。プライマリエージェントが開始されると、`promotable` プロパティは無視されます。これにより、スイッチオーバーまたはフェイルオーバー後のオリジナルのプライマリへの切り替えが簡単になります。設定を上書きするには、ランタイムに`efm set-priority`コマンドを使用します。`efm set-priority`コマンドの詳細については、[Using the efm Utility](#)を参照してください。

```
# A standby with this set to false will not be added to the
# failover priority list, and so will not be available for
# promotion. The property will be used whenever an agent starts
# as a standby or resumes as a standby after being idle. After
# startup/resume, the node can still be added or removed from the
# priority list with the 'efm set-priority' command. This
# property is required for all non-witness nodes.
promotable=true
```

</div>

同じ量のデータが複数のスタンバイノードに書き込まれ、フェイルオーバーが発生した場合、`use.replay.tiebreaker` 値によって、フェイルオーバーマネージャーが代替プライマリを選択する方法が決まります。`use.replay.tiebreaker` プロパティを `true` に設定して、ログシーケンス番号で決定されるように、リカバリ早く出るノードにフェイルオーバーするようにフェールオーバーマネージャーに指示します。ログシーケンス番号を無視し、ユーザ設定に基づいてノードをプロモートには、`use.replay.tiebreaker` を `false` に設定します。

```
# Use replay LSN value for tiebreaker when choosing a standby to
# promote before using failover priority. Set this property to true to
# consider replay location as more important than failover priority
# (as seen in cluster-status command) when choosing the "most ahead"
# standby to promote.
use.replay.tiebreaker=true
```

</div>

`standby.restart.delay` プロパティを使用して、スタンバイが再構成（停止/開始）されてから昇格後に新しいプライマリを追跡するまで待機する時間を秒単位で指定します。

```
# Time in seconds for this standby to delay restarting to follow the
# primary after a promotion. This can be used to have standbys restart
# at different times to increase availability. Caution should be used
# when using this feature, as a delayed standby will not be following
# the new primary and care must be taken that the new primary retains
# enough WAL for the standby to follow it.
# Please see the user's guide for more information.
standby.restart.delay=0
```

</div>

`application.name` プロパティを使用して、古いプライマリノードをスタンバイとして再起動する前に、`primary_conninfo` パラメーターにコピーされるアプリケーションの名前をパラメータ。

```
# During a switchover, recovery settings are copied from a standby
# to the original primary. If the application.name property is set,
# Failover Manager will replace the application_name portion of the
# primary_conninfo entry with this property value before starting
# the original primary database as a standby. If this property is
# not set, Failover Manager will remove the parameter value
# from primary_conninfo.
application.name=
```

Note * プライマリおよびプロモーション可能なスタンバイで `application.name` プロパティを設定する必要があります。フェイルオーバー/スイッチオーバーが発生しイベントに、プライマリノードは、潜在的に再びスタンバイノードになる可能性があります。

</div>

`restore.command` プロパティを使用して、新しいプライマリが昇格したときに `restore_command` を更新するようフェールオーバーマネージャーに指示します。`%h` は、新しいプライマリのアドレスを表します。Failover Managerは、`%h` を新しいプライマリのアドレスに置き換えます。`%f` および `%p` は、サーバーが使用するプレースホルダーです。プロパティが空白のままの場合、フェールオーバーマネージャーは、昇格後にスタンバイの `restore_command` 値を更新しません。

[restore_command](#)の使用の詳細については、PostgreSQLの文書を参照してください。

```
# If the restore_command on a standby restores directly from the
# primary node, use this property to have Failover Manager change
# the command when a new primary is promoted.
#
# Use the %h placeholder to represent the address of the new primary.
# During promotion it will be replaced with the address of the new
# primary.
#
# If not specified, failover manager will not change the
# restore_command value, if any, on standby nodes.
#
# Example:
# restore_command=scp <db service owner>@%h:/var/lib/edb/as12/data/archive/%
restore_command=
```

</div>

プライマリノード上のデータベース・パラメータは、名前を指定し、プライマリノードが書き込みトランザクションを受け入れることができることを保証する保証に、データの受信を確認します同期的スタンバイサーバのカウンタ。 `reconfigure.num.sync` プロパティが真に設定されている場合、フェールオーバーマネージャーは同期的スタンバイサーバの数を減らし、プライマリノードの構成をリロードして現在の値を反映します。

```
# Reduce num_sync when the number of synchronous standbys drops below
# the value required by the primary database. If set to true, Failover
# Manager will reduce the number of standbys needed in the primary's
# synchronous_standby_names property and reload the primary
# configuration. Failover Manager will not reduce the number below 1,
# taking the primary out of synchronous replication, unless the
# reconfigure.sync.primary property is also set to true.
# To raise num_sync, see the reconfigure.num.sync.max property below.
reconfigure.num.sync=false
```

</div>

`reconfigure.num.sync.max` プロパティを使用して、スタンバイがクラスターに追加されたときにnum-syncを上げることができる最大数を指定します。

```
# If reconfigure.num.sync is set to true and this property is set,
# Failover Manager will check if num_sync can be raised when a standby
# is added to the cluster.
# Failover Manager will not raise the value above the maximum set here.
# If the primary database has been taken out of synchronous mode
# completely (see the reconfigure.sync.primary property), then Failover
# Manager will not reconfigure the primary database if standbys are
# added to the cluster.
reconfigure.num.sync.max=
```

</div>

`reconfigure.sync.primary` プロパティを `true` に設定して、スタンバイノードの数が必要なレベルを下回った場合にプライマリデータベースを同期レプリケーションモードから解除します。 `reconfigure.sync.primary` を `false` に設定して、スタンバイカウントが低下した場合に通知を送信しますが、同期レプリケーションは中断しません。

```
# Take the primary database out of synchronous replication mode when
# needed. If set to true, Failover Manager will clear the
# synchronous_standby_names configuration parameter on the primary
# if the number of synchronous standbys drops below the required
# level for the primary to accept writes.
# If set to false, Failover Manager will detect the situation but
# will only send a notification if the standby count drops below the
# required level.
#
# CAUTION: TAKING THE PRIMARY DATABASE OUT OF SYNCHRONOUS MODE MEANS
# THERE MAY ONLY BE ONE COPY OF DATA. DO NOT MAKE THIS CHANGE UNLESS
# YOU ARE SURE THIS IS OK.
reconfigure.sync.primary=false
```

</div>

`minimum.standbys` プロパティを使用して、クラスターに保持されるスタンバイノードの最小数を指定します。スタンバイが指定された最小値まで低下した場合、レプリカ・ノードは、プライマリノードの障害が発生しイベントに推進されることはありません。

```
# Instead of setting specific standbys as being unavailable for
# promotion, this property can be used to set a minimum number
# of standbys that will not be promoted. Set to one, for
# example, promotion will not happen if it will drop the number
# of standbys below this value. This property must be the same on
# each node.
minimum.standbys=0
```

</div>

`priority.standbys` プロパティを使用して、このノードの昇格後のスタンバイの優先度を指定します。

```
# Space-separated list of standby addresses that are high priority for
# promotion when this node is the primary. If set, when this node is
# promoted, addresses in this list will be added to the front of the
# standby priority list. If this list contains addresses that are not
# standbys at the time of promotion, they will not be added.
priority.standbys=
```

</div>

`recovery.check.period` プロパティを使用して、データベースがリカバリしていないかと

うかを確認する前にFailover Managerが待機する秒数を指定します。

```
# Time in seconds between checks to see if a promoting database
# is out of recovery.
recovery.check.period=1
```

</div>

`restart.connection.timeout` プロパティを使用して、そのノード上のデータベースが接続を受け入れる準備をしている間に、フェールオーバーマネージャーが新しく再構成されたプライマリノードまたはスタンバイノードへの接続を試行する秒数を指定します。

```
# Time in seconds to keep trying to connect to a database after a
# start or restart command returns successfully but the database
# is not ready to accept connections yet (a rare occurrence). This
# applies to standby databases that are restarted when being
# reconfigured for a new primary, and to primary databases that
# are stopped and started as standbys during a switchover.
# This retry mechanism is unrelated to the auto.resume.period
# parameter.
restart.connection.timeout=60
```

</div>

`auto.resume.period` プロパティを使用して、エージェントがそのデータベースのモニタリングを再開しようとする秒数を指定します（監視対象データベースが失敗し、エージェントがアイドル状態になった後、またはIDLEモードで起動したとき）。

```
# Period in seconds for IDLE agents to try to resume monitoring
# after a database failure or when starting in IDLE mode. Set to
# 0 for agents to not try to resume (in which case the
# 'efm resume <cluster>' command is used after bringing a
# database back up).
auto.resume.period=0
```

</div>

フェールオーバーマネージャーは、仮想IPを使用するクラスターのサポートを提供します。クラスターが仮想IPを使用する場合、`virtual.ip` プロパティにホスト名前またはIPアドレスを指定します。`virtual.ip.prefix` プロパティで対応するプレフィックスを指定します。`virtual.ip` を空白のままにすると、仮想IPサポートが無効になります。

`virtual.ip.interface` プロパティを使用して、VIPが使用するネットワークインターフェースを提供します。

指定された仮想IPアドレスは、クラスターのプライマリノードにのみ割り当てられます。あなたが `virtual.ip.single=true` を指定した場合、同じVIPアドレスは、フェールオーバーのイベントに新しいプライマリで使用されます。`false`の値を指定して、クラスターの各ノードに一意的IPアドレスを提供します。

仮想IPアドレスの使用については、[Using Failover Manager with Virtual IP Addresses](#)を参照してください。

```
# These properties specify the IP and prefix length that will be
# remapped during failover. If you do not use a VIP as part of
# your failover solution, leave the virtual.ip property blank to
# disable Failover Manager support for VIP processing (assigning,
# releasing, testing reachability, etc).
#
# If you specify a VIP, the interface and prefix are required.
#
# If you specify a host name, it will be resolved to an IP address
# when acquiring or releasing the VIP. If the host name resolves
# to more than one IP address, there is no way to predict which
# address Failover Manager will use.
#
# By default, the virtual.ip and virtual.ip.prefix values must be
# the same across all agents. If you set virtual.ip.single to
# false, you can specify unique values for virtual.ip and
# virtual.ip.prefix on each node.
#
# If you are using an IPv4 address, the virtual.ip.interface value
# should not contain a secondary virtual ip id (do not include
# ":1", etc).
virtual.ip=
virtual.ip.interface=
virtual.ip.prefix=
virtual.ip.single=true
```

Note * プライマリエージェントが起動され、ノードに現在VIPがない場合、EFMエージェントはそれを取得します。プライマリエージェントを停止しても、ノードからVIPは削除されません。

</div>

check.vip.before.promotion プロパティをfalseに設定して、Failover Managerが、VIPが使用中かどうかを確認してから障害が発生したイベントに新しいプライマリに割り当てることを示さないことを示します。これにより、マルチブルのノードが同じVIPアドレスでブロードキャストすることに注意してください。プライマリノードが分離されているか、別のプロセスを介してシャットダウンできない限り、このプロパティを真に設定する必要があります。

```
# Whether to check if the VIP (when used) is still in use before
# promoting after a primary failure. Turning this off may allow
# the new primary to have the VIP even though another node is also
# broadcasting it. This should only be used in environments where
# it is known that the failed primary node will be isolated or
# shut down through other means.
check.vip.before.promotion=true
```

</div>

`pgpool.enable` プロパティを使用して、高可用性のためにFailover ManagerとPgpoolの統合を有効にするかどうかを指定します。非sudoモード（DB所有者として実行）でPgpool統合を有効にする場合、PCPPASSファイルはDB所有者のオペレーティングシステムユーザが所有し、ファイルのアクセス許可を600に設定する必要があります。

```
# A boolean property to enable Failover Manager managed Pgpool HA.
# If enabled, Failover Manager would natively update the joining
# and leaving status of database nodes to active pgpool instance.
# Failover manager expects properly configured and running pgpool
# instances on required nodes. It does not manage setup and
# configuration of pgpool on any node.
#
# By default the property is disabled.
pgpool.enable=false
```

次のパラメーターを使用して、Pgpool統合に使用する値を指定します。

```
# Configurations required for pgpool integration.
# 'pcp.user' - User that would be invoking PCP commands
# 'pcp.host' - Virtual IP that would be used by pgpool. Same as
# pgpool parameter 'delegate_IP'
# 'pcp.port' - The port on which pgpool listens for pcp commands.
# 'pcp.pass.file' - Absolute path of PCPPASSFILE.
# 'pgpool.bin' - Absolute path of pgpool bin directory

# These properties are required if 'pgpool.enable' is set to true.
pcp.user=
pcp.host=
pcp.port=
pcp.pass.file=
pgpool.bin=
```

</div>

スイッチオーバーまたはプライマリ障害シナリオのイベントに、ロード・バランサを再構成するスクリプトへの経路を提供するために、次のプロパティを使用します。スクリプトは、スタンバイに失敗したイベントにも呼び出されます。あなたはこれらのプロパティを使用している場合は、データベース・ノードに障害が発生した場合、別のノードに障害が発生したノードのアドレスをデタッチスクリプトを呼び出すようにする保証に（プライマリ、スタンバイ、および証人）クラスタのすべてのノードに提供されるべきです。

Pgpoolをロードバランサーソリューションとして使用していて、Pgpool統合プロパティを設定している場合、以下のプロパティを設定する必要はありません。

ノードは、ロードに接続する必要がありますときに呼び出されるスクリプトを識別するために `script.load.balancer.attach` プロパティ後にスクリプト名前です。ノードは、ロードから切り離されるべきときに呼び出されるスクリプトの名前を指定する `script.load.balancer.detach` プロパティを使用します。 `%h` プレースホルダを含め

て、クラスターに接続またはクラスターから削除されるノードのIPアドレスを表します。文字列にp（プライマリノード）またはs（スタンバイノード）を含めるようにフェールオーバーマネージャーに指示するには、`%t` プレースホルダを含めます。

```
# Absolute path to load balancer scripts
# The attach script is called when a node should be attached to
# the load balancer, for example after a promotion. The detach
# script is called when a node should be removed, for example
# when a database has failed or is about to be stopped. Use %h to
# represent the IP/hostname of the node that is being
# attached/detached. Use %t to represent the type of node being
# attached or detached: the letter m will be passed in for primary nodes
# and the letter s for standby nodes.
#
# Example:
# script.load.balancer.attach=/somepath/attachscript %h %t
script.load.balancer.attach=
script.load.balancer.detach=
```

</div>

プライマリエージェントが失敗しても、データベースがまだ到達可能であるシナリオでは、ロードからノードを切り離したくないかどうかを示すために `detach.on.agent.failure` プロパティを使用します。デフォルト値は真です。

```
# If set to true, Failover Manager will detach the node from load
# balancer if the primary agent fails but the database is still
# reachable. In most scenarios this is NOT the desired situation. In
# scenarios where the detach script should run with a failed primary
# agent, even when the primary database is still healthy this parameter
# should be set to true. If no value specified it defaults to true (for
# backwards compatibility).
# This is not applicable for standbys.
detach.on.agent.failure=
```

</div>

`script.fence` は、スタンバイノードのプライマリノードへの昇格中に呼び出されるオプションのユーザー指定スクリプトへのパスを指定します。

```
# absolute path to fencing script run during promotion
#
# This is an optional user-supplied script that will be run
# during failover on the standby database node. If left blank,
# no action will be taken. If specified, EFM will execute this
# script before promoting the standby.
#
# Parameters can be passed into this script for the failed primary
# and new primary node addresses. Use %p for new primary and %f
# for failed primary. On a node that has just been promoted, %p
# should be the same as the node's efm binding address.
#
# Example:
# script.fence=/somepath/myscript %p %f
#
# NOTE: FAILOVER WILL NOT OCCUR IF THIS SCRIPT RETURNS A NON-ZERO EXIT
# CODE.
script.fence=
```

</div>

script.post.promotion プロパティを使用して、スタンバイノードがプライマリに昇格した後に呼び出されるオプションのユーザー指定スクリプトへのパスを指定します。

```
# Absolute path to fencing script run after promotion
#
# This is an optional user-supplied script that will be run after
# failover on the standby node after it has been promoted and
# is no longer in recovery. The exit code from this script has
# no effect on failover manager, but will be included in a
# notification sent after the script executes.
#
# Parameters can be passed into this script for the failed primary
# and new primary node addresses. Use %p for new primary and %f
# for failed primary. On a node that has just been promoted, %p
# should be the same as the node's efm binding address.
#
# Example:
# script.post.promotion=/somepath/myscript %f %p
script.post.promotion=
```

</div>

script.resumed property を使用して、エージェントがデータベースの監視を再開モニタリングときに呼び出されるユーザー指定のスクリプトへのオプションのパスを指定します。

```
# Absolute path to resume script
#
# This script is run before an IDLE agent resumes
# monitoring its local database.
script.resumed=
```

</div>

`script.db.failure` プロパティを使用して、監視するデータベースが失敗したことをエージェントが検出した場合にフェールオーバーマネージャーが呼び出すオプションのユーザー指定スクリプトへの完全なパスを指定します。

```
# Absolute path to script run after database failure
# This is an optional user-supplied script that will be run after
# an agent detects that its local database has failed.
script.db.failure=
```

</div>

`script.primary.isolated` プロパティを使用して、プライマリデータベースをモニタリングするエージェントがプライマリがフェールオーバーマネージャークラスターの大部分から分離されていることを検出した場合にフェールオーバーマネージャーが呼び出すオプションのユーザー提供スクリプトへの完全なパスを指定します。このスクリプトは、VIPがリリースされた直後に呼び出されます（VIPが使用中の場合）。

```
# Absolute path to script run on isolated primary
# This is an optional user-supplied script that will be run after
# a primary agent detects that it has been isolated from the
# majority of the efm cluster.
script.primary.isolated=
```

</div>

ノードがプライマリにそのデータベースをプロモートしようとするときにプロモーションに関与しない任意のエージェント・ノード上で呼び出されるスクリプトのパスと名前を指定するには、`script.remote.pre.promotion` プロパティを使用します。

%pプレースホルダを含めて、新しいプライマリノードのアドレスを識別します。

```
# Absolute path to script invoked on non-promoting agent nodes
# before a promotion.
#
# This optional user-supplied script will be invoked on other
# agents when a node is about to promote its database. The exit
# code from this script has no effect on Failover Manager, but
# will be included in a notification sent after the script
# executes.
#
# Pass a parameter (%p) with the script to identify the new
# primary node address.
#
# Example:
# script.remote.pre.promotion=/path_name/script_name %p
script.remote.pre.promotion=
```

</div>

`script.remote.post.promotion` プロパティを使用して、昇格後にプライマリ以外のノードで呼び出されるスクリプトのパスと名前を指定します。

%p プレースホルダを含めて、新しいプライマリノードのアドレスを識別します。

```
# Absolute path to script invoked on non-primary agent nodes
# after a promotion.
#
# This optional user-supplied script will be invoked on nodes
# (except the new primary) after a promotion occurs. The exit code
# from this script has no effect on Failover Manager, but will be
# included in a notification sent after the script executes.
#
# Pass a parameter (%p) with the script to identify the new
# primary node address.
#
# Example:
# script.remote.post.promotion=/path_name/script_name %p
script.remote.post.promotion=
```

</div>

`script.custom.monitor` プロパティを使用して、定期的な間隔（`custom.monitor.interval` プロパティで秒単位で指定）で呼び出されるオプションのスクリプトの名前と場所を指定します。

`custom.monitor.timeout` を使用して、スクリプトの実行を許可する最大時間を指定します。指定した時間内にスクリプトの実行が完了しない場合、フェールオーバーマネージャーは通知を送信します。

`custom.monitor.safe.mode` を `true` に設定して、フェールオーバーマネージャーにスクリプトからゼロ以外の終了コードをレポートするように指示しますが、終了コードの結果としてスタンバイをプロモートせません。

```
# Absolute path to a custom monitoring script.
#
# Use script.custom.monitor to specify the location and name of
# an optional user-supplied script that will be invoked
# periodically to perform custom monitoring tasks. A non-zero
# exit value means that a check has failed; this will be treated
# as a database failure. On a primary node, script failure will
# cause a promotion. On a standby node script failure will
# generate a notification and the agent will become IDLE.
#
# The custom.monitor.* properties are required if a custom
# monitoring script is specified:
#
# custom.monitor.interval is the time in seconds between executions
# of the script.
#
# custom.monitor.timeout is a timeout value in seconds for how
# long the script will be allowed to run. If script execution
# exceeds the specified time, the task will be stopped and a
# notification sent. Subsequent runs will continue.
#
# If custom.monitor.safe.mode is set to true, non-zero exit codes
# from the script will be reported but will not cause a promotion
# or be treated as a database failure. This allows testing of the
# script without affecting EFM.
#
script.custom.monitor=
custom.monitor.interval=
custom.monitor.timeout=
custom.monitor.safe.mode=
```

</div>

sudo.command プロパティを使用して、拡張アクセス許可が必要なタスクを実行するときにフェールオーバーマネージャーによって呼び出されるコマンドを指定します。このオプションを使用して、システム認証に固有のコマンドオプションを含めオプション。

sudo.user.command プロパティを使用して、データベース所有者が実行するコマンドを実行するときにフェールオーバーマネージャーによって呼び出されるコマンドを指定します。

```
# Command to use in place of 'sudo' if desired when efm runs
# the efm_db_functions or efm_root_functions, or efm_address
# scripts.
# Sudo is used in the following ways by efm:
#
# sudo /usr/edb/efm-<version>/bin/efm_address <arguments>
# sudo /usr/edb/efm-<version>/bin/efm_root_functions <arguments>
# sudo -u <db service owner> /usr/edb/efm-<version>/bin/efm_db_functions <ar
#
# 'sudo' in the first two examples will be replaced by the value
# of the sudo.command property. 'sudo -u <db service owner>' will
# be replaced by the value of the sudo.user.command property.
# The '%u' field will be replaced with the db owner.
sudo.command=sudo
sudo.user.command=sudo -u %u
```

</ div>

lock.dir プロパティを使用して、フェールオーバーマネージャーロックファイルの代替場所を指定します。このファイルにより、フェールオーバーマネージャーがノード上の単一クラスターに対してマルチプルの（孤立している可能性のある）エージェントを起動できなくなります。

```
# Specify the directory of lock file on the node. Failover
# Manager creates a file named <cluster>.lock at this location to
# avoid starting multiple agents for same cluster. If the path
# does not exist, Failover Manager will attempt to create it. If
# not specified defaults to '/var/lock/efm-<version>'
lock.dir=
```

</ div>

log.dir プロパティを使用して、エージェントログファイルが書き込まれる場所を指定します。ディレクトリが存在しない場合、Failover Managerはディレクトリを作成しようとします。

```
# Specify the directory of agent logs on the node. If the path
# does not exist, Failover Manager will attempt to create it. If
# not specified defaults to '/var/log/efm-<version>'. (To store
# Failover Manager startup logs in a custom location, modify the
# path in the service script to point to an existing, writable
# directory.)
# If using a custom log directory, you must configure
# logrotate separately. Use 'man logrotate' for more information.
log.dir=
```

</ div>

Failover ManagerホストでUDPまたはTCPプロトコルを有効にした後、syslogへのロギング

を有効にできます。 `syslog.protocol` パラメータを使用してプロトコルタイプ（UDPまたはTCP）を指定し、 `syslog.port` パラメータとしてsyslogホストのリスナポートを指定します。 `syslog.facility` 値は、エントリーを作成したプロセスの識別子として使用できます。値はLOCAL0とLOCAL7の間でなければなりません。

```
# Syslog information. The syslog service must be listening on
# the port for the given protocol, which can be UDP or TCP.
# The facilities supported are LOCAL0 through LOCAL7.
syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

</div>

`file.log.enabled` および `syslog.enabled` プロパティを使用して、実装するロギングのタイプを指定します。 `file.log.enabled` を `true` に設定して、ファイルへのロギングを有効にします。UDPプロトコルまたはTCPプロトコルを有効にし、 `syslog.enabled` を `true` に設定して、syslogへのロギングを有効にします。ファイルとsyslogの両方へのロギングを有効にできます。

```
# Which logging is enabled.
file.log.enabled=true
syslog.enabled=false
```

syslogロギングの構成の詳細については、[Enabling syslog Log File Entries](#)を参照してください。

</div>

`jgroups.loglevel` パラメーターと `efm.loglevel` パラメーターを使用して、Failover Managerによって記録される詳細レベルを指定します。デフォルト値はINFOです。ロギングの詳細については、[Controlling Logging](#)を参照してください。

```
# Logging levels for JGroups and EFM.
# Valid values are: TRACE, DEBUG, INFO, WARN, ERROR
# Default value: INFO
# It is not necessary to increase these values unless debugging a
# specific issue. If nodes are not discovering each other at
# startup, increasing the jgroups level to DEBUG will show
# information about the TCP connection attempts that may help
# diagnose the connection failures.
jgroups.loglevel=INFO
efm.loglevel=INFO
```

</div>

`jvm.options` プロパティを使用して、JVM関連の構成情報を渡します。デフォルト設定は、Failover Managerエージェントが使用できるメモリの量を指定します。

```
# Extra information that will be passed to the JVM when starting
# the agent.
jvm.options=-Xmx128m
```

encrypting_database_password

</div>

Encrypting Your Database Password

</div>

フェールオーバーマネージャーでは、データベースのパスワードをクラスタープロパティファイルに含める前に暗号化する必要があります。（ディレクトリ）を使用してパスワードを暗号化します。パスワードを暗号化する場合、ユーティリティを起動するときにコマンドラインでパスワードを渡すか、 `EFMPASS` 環境変数を使用できます。

パスワードを暗号化するには、次のコマンドを使用します。

```
# efm encrypt <cluster_name> [ --from-env ]
```

`<cluster_name>` は、Failover Manager クラスターの名前を指定します。

`--from-env` オプションを含める場合、暗号化ユーティリティを呼び出す前に、暗号化する値をエクスポートする必要があります。例：

```
export EFMPASS=password
```

`--from-env` オプションを含めない場合、フェールオーバーマネージャーは、クラスタープロパティファイルに配置するための暗号化パスワードを生成する前に、データベースパスワードを2回入力するプロンプトます。ユーティリティ暗号化されたパスワードは、コピーして、クラスタのプロパティに暗号化されたパスワードを貼り付けた場合。

Note * 多くのJavaベンダーは、完全な強度の暗号化が含まれたバージョンのJavaを出荷していますが、エクスポート制限のため有効になっていません。データベースパスワードを暗号化しようとしたときに不正なキーサイズを指すエラーが発生した場合は、プラットフォームに制限のないポリシーを提供するJava Cryptography Extension (JCE) をダウンロードして有効にする必要があります。

次の例は、暗号化ユーティリティを使用して `acctg` クラスターのパスワードを暗号化する例を示しています。

```
# efm encrypt acctg
This utility will generate an encrypted password for you to place in
your EFM cluster property file:
/etc/edb/efm-4.2/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

Note * プロパティファイルが存在しない場合、ユーティリティは通知します。

あなたの暗号化されたパスワードを受け取った後、プロパティファイルにパスワードを貼り付けるとフェイルオーバーマネージャサービスをスタート。暗号化されたパスワードに問題がある場合、Failover Managerサービスはスタートされません。

```
[witness@localhost ~]# systemctl start edb-efm-4.2
Job for edb-efm-4.2.service failed because the control process exited with er
```

Failover Managerサービスの開始時にこのメッセージを受け取った場合、詳細については（`/var/log/efm-4.2/startup-efm.log` にある）スタートアップログを参照してください。

RHEL / CentOS 7.xまたはRHEL / CentOS 8.xを使用している場合、スタートアップ情報は次のコマンドでも利用できます。

```
systemctl status edb-efm-4.2
```

クラスターが誤って別のクラスターのデータベースに接続するのを防ぐため、クラスター名前は暗号化されたパスワードに組み込まれます。クラスター名前を変更する場合は、データベースパスワードを再暗号化し、クラスタープロパティファイルを更新する必要があります。

**** EFMPASS環境変数の使用****

次の例は、パスワードを暗号化するときに`--from-env`環境変数を使用する例を示しています。

`efm encrypt` コマンドを呼び出す前に、`EFMPASS` の値をパスワード（`1safepassword`）に設定します。

```
# export EFMPASS=1safepassword
```

次に、`--from-env` オプションを指定して、`efm encrypt` を呼び出します。

```
# efm encrypt acctg --from-env
# 7ceecd8965fa7a5c330eaa9e43696f83
```

暗号化されたパスワード（ `7ceecd8965fa7a5c330eaa9e43696f83` ）はテキスト値として返されます。スクリプトを使用する場合、コマンドの終了コードを確認して、コマンドが成功したことを確認できます。正常に実行されると、 `0` が返されます。

The Cluster Members File

</div>

Failover Manager クラスターの各ノードには、現在の Failover Manager クラスターメンバーのリストを含むクラスターメンバーファイル（デフォルトでは `efm.nodes` 名前付け名前）があります。エージェントが起動すると、ファイルを使用して他のクラスターメンバーを見つけます。Failover Manager インストーラは、 `/etc/edb/efm-4.2` ディレクトリに `efm.nodes.in` 名前付けのクラスターメンバーファイルのファイルテンプレートを作成します。

Failover Manager のインストールが完了したら、テンプレートの作業用コピーを `make` する必要があります。

```
cp /etc/edb/efm-4.2/efm.nodes.in /etc/edb/efm-4.2/efm.nodes
```

テンプレートファイルをコピーした後、ファイルの所有者を `efm` に変更します。

```
chown efm:efm efm.nodes
```

デフォルトでは、フェールオーバーマネージャーはクラスターメンバーファイルの名前付けが `efm.nodes` であると想定しています。クラスターメンバーのファイルに `efm.nodes` 以外の何かに名前場合は、新しい名前を使用するように指示フェールオーバーマネージャーにフェールオーバーマネージャーサービスを変更する必要があります。

最初に起動したノードのクラスターメンバーファイルは空にすることができます。このノードがメンバーシップコーディネーターになります。後続の各ノードで、クラスターメンバーファイルには、Membership Coordinator のアドレスとポート番号が含まれている必要があります。クラスターメンバーファイルの各エントリは、アドレスで区切られている必要があります：ポートフォーマット、空白文字で区切られたマルチプルのエントリ。

エージェントは、クラスターの現在のメンバーにマッチするように `efm.nodes` ファイルの内容を更新します。エージェントがクラスターに結合または離脱すると、他のエージェントの `efm.nodes` ファイルが更新され、現在のクラスターメンバーシップが反映されます。
[efm stop-cluster](#) コマンドを呼び出した場合、Failover Manager はファイルを変更しません。

メンバーシップコーディネーターがクラスターを離れると、別のノードがロールします。[efm cluster-status](#) コマンドを使用して、Membership Coordinator のアドレスを見つけることができます。エージェントがダウンしているときにノードがクラスターに参加または離脱する場合、そのエージェントを開始する前に、ファイルに少なくとも現在のメンバーシップコーディネーターのアドレスとポートが含まれていることを手動で保証必要があります。

クラスターに参加するノードのアドレスとポートがわかっている場合は、いつでもクラス

ターメンバーファイルにアドレスを含めることができます。スタートアップに、（内の） `auto.allow.hosts` プロパティが `true` に設定されていない限り、クラスターメンバーを識別しないアドレスは無視されます。

`stable.nodes.file` プロパティ（にあります）が `true` に設定されている場合、エージェントはクラスターメンバーがクラスターに参加または離脱結合ときに `.nodes` ファイルを更新しません。この動作は、クラスターメンバーのIPアドレスが頻繁に変更されない場合に最も役立ちます。

Extending Failover Manager Permissions

</ div>

Failover Managerのインストールに、インストーラは `efm` 名前付けのユーザを作成します。`efm` には、通常データベース所有者またはオペレーティングシステムのスーパーユーザに制限されている管理機能を実行するための十分な権限がありません。

- データベーススーパーユーザ権限を必要とする管理機能を実行する場合、`efm` は `efm_db_functions` スクリプトを呼び出します。
- オペレーティングシステムのスーパーユーザ権限を必要とする管理機能を実行する場合、`efm` は `efm_root_functions` スクリプトを呼び出します。
- 仮想IPアドレスの割り当てまたは解放時に、`efm` は `efm_address` スクリプトを呼び出します。
- Pgpool統合を有効にすると、`efm` は `efm_pgpool_functions` スクリプトを呼び出します。

`efm_db_functions` または `efm_root_functions` スクリプトは、`efm` ユーザに代わって管理機能を実行します。

`sudoers`ファイルには、ユーザ `efm` が `postgres` または `enterprisedb` が所有するクラスターのFailover Managerサービスを制御できるようにするエントリが含まれています。`sudoers`ファイルのコピーを変更して、他のユーザーが所有するPostgresクラスターを管理するパーミッションを `efm` に付与できます。

`efm-42` ファイルは `/etc/sudoers.d` にあり、次のエントリが含まれています。

```
# Copyright EnterpriseDB Corporation, 2014-2020. All Rights Reserved.
#
# Do not edit this file. Changes to the file may be overwritten
# during an upgrade.
#
# This file assumes you are running your efm cluster as user 'efm'. If not,
# then you will need to copy this file.

# Allow user 'efm' to sudo efm_db_functions as either 'postgres' or 'enterpr
# If you run your db service under a non-default account, you will need to c
# this file to grant the proper permissions and specify the account in your
# cluster properties file by changing the 'db.service.owner' property.
efm ALL=(postgres) NOPASSWD: /usr/edb/efm-4.2/bin/efm_db_functions
efm ALL=(enterprisedb) NOPASSWD: /usr/edb/efm-4.2/bin/efm_db_functions

# Allow user 'efm' to sudo efm_root_functions as 'root' to write/delete the
# validate the db.service.owner property, etc.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-4.2/bin/efm_root_function

# Allow user 'efm' to sudo efm_address as root for VIP tasks.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-4.2/bin/efm_address

# Allow user 'efm' to sudo efm_pgpool_functions as root for pgpool tasks.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-4.2/bin/efm_pgpool_functi

# relax tty requirement for user 'efm'
Defaults:efm !requiretty
```

あなたが `postgres` または `enterprisedb` 以外のユーザーによって所有されているクラスタをモニタするためにフェイルオーバーマネージャーを使用している場合は、`efm-42` ファイルのコピーをmake、ユーザが自分のクラスタを管理するために `efm_functions` スクリプトへのアクセスを許可するコンテンツを変更します。

エージェントが原因パーミッションの問題でスタートできない場合は、必ずデフォルトの `/etc/sudoers` ファイルは、ファイルの末尾に次の行が含まれていmake

```
## Read drop-in files from /etc/sudoers.d (the # here does not # mean a comm
# includedir /etc/sudoers.d
```

</div>

sudoなしでフェールオーバーマネージャーを実行する

デフォルトでは、Failover Managerはsudoを使用して、システム機能へのアクセスを安全に管理します。sudoアクセスなしで実行するようにフェールオーバーマネージャーを構成することを選択した場合、ルートアクセスが引き続き必要であることを注意してください。

- Failover Manager RPMをインストールします。
- フェールオーバーマネージャーのセットアップタスクを実行します。

sudoなしでFailover Managerを実行するには、Failover Managerに代わって管理機能を実行する権限を持つデータベースプロセスの所有者を選択する必要があります。ユーザは、デフォルトデータベースのスーパーユーザ（例、EnterpriseDBのかはpostgres）または別の特権ユーザである可能性があります。ユーザを選択した後：

1.次のコマンドを使用して、ユーザを `efm` グループに追加します。

```
usermod -a -G efm enterprisedb
```

- これにより、ユーザは `/var/run/efm-4.2` および `/var/lock/efm-4.2` に書き込むことがユーザ。

2.クラスター名前を再利用する場合は、以前に作成したログファイルを削除します。新しいユーザは、デフォルト（または他の）所有者が作成したログファイルに書き込むことができません。

3.クラスタープロパティテンプレートファイルとノードテンプレートファイルをコピーします。

```
su - enterprisedb

cp /etc/edb/efm-4.2/efm.properties.in <directory/cluster_name>.properties

cp /etc/edb/efm-4.2/efm.nodes.in <directory>/<cluster_name>.nodes
```

次に、クラスタープロパティファイルを変更し、`db.service.owner` プロパティでユーザの名前を指定します。また、`db.service.name` プロパティが空白である保証する必要があります。sudoがないと、ルートアクセスなしでサービスを実行できません。

構成を変更した後、新しいユーザは次のコマンドでフェールオーバーマネージャーを制御できます。

```
/usr/edb/efm-4.2/bin/runefm.sh start|stop <directory/cluster_name>.properties
```

ここで、`<directory/cluster_name.properties>` はクラスタープロパティファイルのフルパスを指定します。ユーザがデフォルト以外のユーザはエージェントを制御したり、EFMスクリプトを使用するたびにプロパティファイルへのフルパスが提供されなければならないことを保証しなければならないことに注意してください。

新しいユーザがFailover Managerをサービスとして管理できるようにするには、カスタムスクリプトまたはユニットファイルを提供する必要があります。

Failover Managerは、`/usr/edb/efm-4.2/bin/secure/` にある `manage-vip` という名前付けのバイナリを使用して、sudo権限なしでVIP管理操作を実行します。このスクリプトは、

setuidを使用して、仮想IPアドレスの管理に必要な特権を取得します。

- このディレクトリは、 ルートおよび `efm` グループのユーザーのみがアクセスできます。
- バイナリは、 ルートおよび `efm` グループによってのみ実行可能です。

セキュリティ上の理由から、 `/usr/edb/efm-4.2/bin/secure/` ディレクトリまたは `manage-vip` スクリプトのアクセス権限を変更ことをお勧めします。

sudoなしでFailover Managerを使用する方法の詳細については、次を参照してください。

<https://www.enterprisedb.com/blog/running-edb-postgres-failover-manager-without-sudo>

Using Failover Manager with Virtual IP Addresses

</div>

フェールオーバーマネージャーは、 `efm_address` スクリプトを使用して、仮想IPアドレスの割り当てまたはリリースます。

Note * 仮想IPアドレスは、多くのクラウドプロバイダーでサポートされていません。これらの環境では、別のメカニズム（AWSのElastic IPアドレスなど）を使用する必要があります。これは、フェンシングまたはプロモーション後のスクリプトで必要に応じて変更できます。

デフォルトでは、スクリプトは次の場所にあります。

```
/usr/edb/efm-4.2/bin/efm_address
```

フェールオーバーマネージャーは、次のコマンドバリエーションを使用して、IPv4またはIPv6 IPアドレスを割り当てまたはリリースします。

仮想IPv4 IPアドレスを割り当てるには：

```
# efm_address add4 <interface_name> <IPv4_addr>/<prefix>
```

仮想IPv6 IPアドレスを割り当てるには：

```
# efm_address add6 <interface_name> <IPv6_addr>/<prefix>
```

仮想アドレスをリリースするには：

```
# efm_address del <interface_name> <IP_address/prefix>
```

どこで：

`<interface_name>` は、クラスタープロパティファイルの `virtual.ip.interface` プロパティで指定された名前と一致します。

<IPv4_addr> または <IPv6_addr> は、クラスタープロパティファイルの `virtual.ip` プロパティで指定された値と一致します。

`prefix` は、クラスタープロパティファイルの `virtual.ip.prefix` プロパティで指定された値と一致します。

仮想IPアドレスを記述するプロパティの詳細については、[The Cluster Properties File](#)を参照してください。

ルートユーザとして `efm_address` スクリプトを呼び出す必要があります。 `efm` ユーザはインストール中に作成され、`efm_address` スクリプトを実行するための`sudoers`ファイルの特権が付与されます。`sudoers`ファイルの詳細については、[Extending Failover Manager Permissions](#)を参照してください。

Note * VIPアドレス（または `bind.address` 以外のアドレス）がノードに割り当てられている場合、オペレーティングシステムはデータベースへの接続時に使用されるソースアドレスを選択できます。すべての監視対象データベースの `pg_hba.conf` ファイルを変更して、レプリケーションシナリオ内のすべてのアドレスからの連絡を許可してください。

** VIPのテスト**

フェイルオーバーマネージャーと仮想IP（VIP）アドレスを使用する場合は、フェイルオーバーマネージャーを開始する前に、手動でVIP機能をテストすることが重要です。これにより、実際のフェイルオーバー中に問題が発生する前に、ネットワーク関連の問題をキャッチします。VIPのテスト中に、Failover Managerが実行されていない保証してください。

次の手順では、フェールオーバーマネージャーが実行するアクションをテストします。この例では、次のプロパティ値を使用します。

```
virtual.ip=172.24.38.239
virtual.ip.interface=eth0
virtual.ip.prefix=24
ping.server.command=/bin/ping -q -c3 -w5
```

Note * `virtual.ip.prefix` は、仮想IPアドレスの有効ビット数を指定します。

ノードからVIPをピングするように指示されたら、`ping.server.command` プロパティで定義されたコマンドを使用します。

1.すべてのノードからVIPにpingを実行して、アドレスがまだ使用されていないことを確認します。

```
# /bin/ping -q -c3 -w5 172.24.38.239
PING 172.24.38.239 (172.24.38.239) 56(84) bytes of data.
--- 172.24.38.239 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss,
time 3000ms
```

100%のパケット損失が表示されるはずですが。

2.プライマリノードで `efm_address add4` コマンドを実行してVIPを割り当て、IPアドレスで確認します。

```
# efm_address add4 eth0 172.24.38.239/24
# ip address
<output truncated>
eth0 Link encap:Ethernet HWaddr 36:AA:A4:F4:1C:40
inet addr:172.24.38.239 Bcast:172.24.38.255
...
```

3.他のノードからVIPをpingして、VIPに到達できることを確認します。

```
# /bin/ping -q -c3 -w5 172.24.38.239
PING 172.24.38.239 (172.24.38.239) 56(84) bytes of data.
--- 172.24.38.239 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.023/0.025/0.029/0.006 ms
```

パケット損失は見られないはずですが。

4. `efm_address del` コマンドを使用してプライマリノードのアドレスをリリースし、ノードがIPアドレスで解放されたことを確認します。

```
# efm_address del eth0 172.24.38.239/24
# ip address
eth0 Link encap:Ethernet HWaddr 22:00:0A:89:02:8E
inet addr:10.137.2.142 Bcast:10.137.2.191
...
```

この手順の出力には、eth0インタフェースが表示されないはずですが。

5.手順3を繰り返します。今回は、スタンバイと監視が使用中のVIPを認識しないことを確認します。

```
# /bin/ping -q -c3 -w5 172.24.38.239
PING 172.24.38.239 (172.24.38.239) 56(84) bytes of data.
--- 172.24.38.239 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss,
time 3000ms
```

100%のパケット損失が表示されるはずですが。すべてのノードでこの手順を繰り返します。

6.すべてのスタンバイノードで手順2を繰り返し、VIPをすべてのノードに割り当てます。任意のノードからVIPをピングして、使用中であることを確認できます。

```
# efm_address add4 eth0 172.24.38.239/24
# ip address
<output truncated>
eth0 Link encap:Ethernet HWaddr 36:AA:A4:F4:1C:40
inet addr:172.24.38.239 Bcast:172.24.38.255
...
```

上記のテスト手順の後、フェールオーバーマネージャーをスタート前に、非プライマリノードからVIPをリリースします。

Note * VIPに使用されるネットワークインタフェースは、Failover Managerエージェントの `bind.address` 値に使用されるインタフェースと同じである必要はありません。プライマリエージェントはフェイルオーバー中に必要に応じてVIPをドロップし、フェールオーバーマネージャーはスタンバイを昇格する前にVIPが利用できないことを確認します。バインドアドレスネットワークに障害が発生すると、プライマリ分離とフェイルオーバー。

VIPが別のインタフェースを使用している場合、プライマリエージェントがドロップする前に、残りのクラスターが到達可能なVIPをチェックするタイミング条件が発生する場合があります。この場合、EFMは、予想通り、フェイルオーバーが発生していること<保証する `node.timeout` プロパティで指定した秒数のVIPチェックをリトライします。

Configuring for Eager Failover

デフォルトの実行モードでは、プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動されるまでフェイルオーバー保護はありません。そのような場合を回避するには、イーガーFailover.Youと呼ばれるプライマリエージェントの終了は、次の手順を実行して、熱心なフェイルオーバーを設定することができたときにフェイルオーバーを引き起こすことが `systemd` を通じてプライマリノードを設定することができます（例> Advanced Server>12を使用していますおよびEFMバージョン4.2）：

</ div>

Eager Failoverを有効にする

- フェールオーバーマネージャーエージェントが停止または失敗するとすぐにデータベースサーバが停止するため、フェールオーバーマネージャーを起動する前に、すべてのエージェントに次のプロパティを設定する必要があります。

```
* primary.shutdown.as.failure=true
```

- フェールオーバーマネージャーを起動する前にこのプロパティを設定しない場合、EFMエージェントをシャットダウンすると、フェイルオーバーなしでデータベースがシャットダウンされます。
- Eager Failoverを有効にして、`efm stop-cluster` コマンドを使用すると、すべてのefmエージェントが停止し、プライマリデータベースがシャットダウンします。エー

ジェントが実行されていないため、フェイルオーバーはありません。このようなシナリオを回避するには、`enable.stop.cluster` プロパティを使用してコマンドを無効にします。

```
enable.stop.cluster=false
```

- データベースサーバとローカルフェイルオーバーマネージャエージェントが実行されていることを確認します。
- ルートとして、`/etc/systemd/system/edb-as-12.service` ファイルを作成し、次のとおりです。

```
.include /lib/systemd/system/edb-as-12.service
[Unit]
BindsTo=edb-efm-4.2.service
```

- 次のコマンドを実行して、構成ファイルをリロードします。

```
systemctl daemon-reload
```

上記の変更により、Failover Managerエージェントが停止または強制終了されると、クラスタの残りの部分はこのシチュエーションを障害として扱い、フェイルオーバーを試行します。

</div>

Eager Failoverの無効化

- あなたは、データベースを停止せずにフェイルオーバーマネージャを停止する場合は、`/etc/systemd/system/edb-as-12.service` に次の行コメントアウト：

```
BindsTo=edb-efm-4.2.service
```

- 次のコマンドを実行して、構成ファイルをリロードします。

```
systemctl daemon-reload
```

Eager FailoverモードでのEFMのアップグレード

Postgresを停止せずにFailover Managerをアップグレードするには、Eager Failoverモードを一時的に無効にする必要があります。次の手順を実行します。

1. [Disable Eager Failover](#)
2. [Stop and upgrade EFM](#)
3. [Enable Eager Failover](#)

重要な注意事項

- systemdにコマンドが非須藤セットアップでEFMの実行中にデータベースを管理するために使用されていないので、熱心なフェイルオーバーのみ`sudo>`でサポートおよび非須藤モードでサポートされていません。
- 熱心なフェイルオーバーは、VIPが古いプライマリによって解放されない状況には適していません。
- Eager Failoverは、次の状況に適しています。
 - - EDB Postgres High Availability セットアップの場合-またはlibpqを使用したクライアント接続フェイルオーバーを使用したセットアップの場合- `script.fence` によってトリガーされたカスタムスクリプトが古いプライマリサーバ (STONITH) をフェンスする場合。いくつかの例では、VMwareのvCenter統合、OpenStackの統合、またはLights-Out管理でVMをシャットダウンすることです。 - `script.fence` によってトリガーされるカスタムスクリプトがsshを使用してVIPを非アクティブ化する場合。
 - 注：古いプライマリがリリースする前に新しいプライマリがVIPを接続できるようにするには、 `check.vip.before.promotion=false` の設定が必要です。
- `primary.shutdown.as.failure=true` を使用する場合は注意が必要です。必要に応じてデータベースを安全に停止する方法については、[primary.shutdown.as.failure](#) プロパティの説明を参照してください。
- すべてのフェイルオーバーでは、というアッププライマリ端が自動的に運用スタンバイとして回復しない、プライマリに失敗しました。したがって、クラスターにはマルチプルの昇格可能なスタンバイが含まれている必要があり、スタンバイの総数は `minimum.standbys` プロパティに指定された値より少なくとも2多い必要があります。これは一般的な推奨事項ですが、Eager Failoverを使用する場合はより緊急になります。
- データベースサーバが停止している場合は、データベースを再起動することも、フェイルオーバーマネージャーをスタートします。
 - 注：-プロパティ値が正しくないなど、EFMの起動に問題がある場合、データベースサーバは実行されていないというワーニングを表示することなく、再スタートおよびシャットダウンします。 -フェールオーバーマネージャープロセスが以前に強制終了された場合、ロックファイルは引き続き存在し、エージェントは自動的に再起動できません。 -データベースサーバまたはFailover Managerエージェントのスタートアップに問題がある場合は、EFM起動ログで情報を確認してください。
- `stop-cluster` コマンドを実行した結果、すべてのノードでEFMが停止します。Eager Failoverモードでは、`stop-cluster` コマンドはフェイルオーバーなしでPostgresも停止します。必ず `stop-cluster` コマンドが意図せずに起動することができないmakeに設定 `enable.stop.cluster=false` 。

Using Failover Manager

</ div>

フェールオーバーマネージャーは、1つ以上のスタンバイサーバーでクラスターのモニタリングとフェールオーバーをサポートします。リソースの需要が増加または減少するにつれて、クラスターにノードを追加または削除できます。

プライマリノードが再起動した場合は、フェールオーバーマネージャーは、データベースがプライマリノード上でダウンして検出し、プライマリのロールにスタンバイ・ノードをプロモートことができます。これが発生すると、（リブートされた）プライマリノードのFailover Managerエージェントは `recovery.conf` ファイルの書き込みを試み、Postgresが二次的なプライマリとしてスタートしないようにします。したがって、データベースサーバーを起動する前に、Failover Managerエージェントを起動する必要があります。エージェントはアイドルモードでスタートし、クラスター内に既にプライマリがあるかどうかを確認します。プライマリノードがある場合、エージェントはリカバリ.confまたはスタンバイを確認します。シグナルファイルが存在するか、必要に応じてリカバリ.confを作成して、データベースが2プライマリとして起動しないようにします。

フェールオーバーマネージャークラスターの管理

構成が完了すると、Failover Managerクラスターは定期的なメンテナンスを必要としません。以下のセクションでは、フェールオーバーマネージャークラスターで必要になることがある管理タスクの実行に関する情報を提供します。

デフォルトでは、[some of the efm commands](#)は `efm` またはOSスーパーユーザーによって呼び出される必要があります。管理者は、ユーザーを `efm` グループに追加することにより、ユーザーがこれらのコマンドを呼び出すことを選択的に許可できます。コマンドは次のとおりです。

- [efm allow-node](#)
- [efm disallow-node](#)
- [efm promote](#)
- [efm resume](#)
- [efm set-priority](#)
- [efm stop-cluster](#)
- [efm upgrade-conf](#)

</ div>

フェールオーバーマネージャークラスターの起動

Failover Managerクラスターのノードは、任意のオーダーでスタートできます。

RHEL / CentOS 7.xまたはRHEL / CentOS 8.xでFailover Managerクラスターをスタートには、スーパーユーザー権限を引き受けて、コマンドを呼び出します。

```
systemctl start edb-efm-4.2
```

ノードのクラスタープロパティファイルで `is.witness` が `true` であると指定されている場合、ノードはウィットネスノードとしてスタートします。

ノードが専用の監視ノードではないノード、フェールオーバーマネージャーはローカルデータベースに接続し、`pg_is_in_recovery()` 関数を呼び出しファンクション。サーバーが `false` に応答すると、エージェントはノードがプライマリノードであると想定し、仮想IPアドレスをノードに割り当てます（該当する場合）。サーバーが `true` に応答する場合、Failover Managerエージェントはノードがスタンバイサーバーであると想定します。サーバーが応答しない場合、エージェントはアイドル状態でスタートします。

クラスターに参加した後、フェールオーバーマネージャーエージェントは、提供されたデータベース資格情報をチェックして、クラスター内のすべてのデータベースに接続できることを保証します。エージェントが接続できない場合、エージェントはシャットダウンします。

新しいプライマリノードまたはスタンバイノードがクラスターに参加する場合、既存のすべてのノードは、新しいノード上のデータベースに接続できることも確認します。

</div>

Note * `tmpfs`（一時ファイルシステム）で `/var/lock` または `/var/run` を実行している場合、Failover Managerのsystemdサービスファイルが `systemd-tmpfiles-setup.service` に依存しているmakeを確認してください。

クラスターへのノードの追加

いつでもフェールオーバーマネージャークラスターにノードを追加できます。クラスターにノードを追加するときは、クラスターを変更して新しいノードを許可し、クラスターを見つける方法を新しいノードに伝える必要があります。次の手順では、クラスターへのノードの追加について詳しく説明します。

1. `auto.allow.hosts` が `true` に設定されていない限り、`efm allow-node` コマンドを使用して、新しいノードのアドレスをフェールオーバーマネージャーの許可ノードホストリストに追加します。コマンドを呼び出すときに、新しいノードのクラスター名前とアドレスを指定します。

```
efm allow-node <cluster_name> <address>
```

- `efm allow-node` コマンドの使用またはFailover Managerサービスの制御の詳細については、[Using the EFM Utility](#)を参照してください。
- フェールオーバーマネージャーエージェントをインストールし、新しいノードでクラスタープロパティファイルを作成します。プロパティファイルの変更の詳細については、[The Cluster Properties File](#)を参照してください。

- 2.新しいノードでクラスターメンバーファイルを作成し、Membership Coordinatorのエントリーを追加します。クラスターメンバーファイルの変更の詳細については、[The Cluster](#)

[Members File](#)を参照してください。

3.新しいノードでスーパーユーザ権限を想定し、Failover Managerエージェントをスタートします。コマンド呼び出しRHEL / CentOSの7.xまたはRHEL / CentOSの8.xの上のフェイルオーバーマネージャークラスタを、スタートするには：

```
systemctl start edb-efm-4.2
```

新しいノードがクラスタに参加すると、フェールオーバーマネージャは、`user.email` プロパティで提供される管理者のメールに通知を送信し、指定された通知スクリプトを呼び出します。

</div>

Note * 現在のノードの便利なスタンバイになるには、PostgreSQLストリーミングレプリケーションシナリオでノードをスタンバイにする必要があります。

スタンバイの優先度を変更する

Failover Managerクラスタに複数のスタンバイサーバーが含まれている場合、`efm set-priority` コマンドを使用して、スタンバイノードの昇格の優先度に影響を与えることができます。フェイルオーバーマネージャークラスタのいずれかの既存のメンバ上でコマンドを起動し、メンバのIPアドレスの後に優先順位の値を指定します。

例 `10.0.1.9` をモニタリングしている `acctg` クラスタメンバがプライマリスタンバイ (1) であることをフェイルオーバーマネージャ指示し、次のコマンドを実行します。

```
efm set-priority acctg 10.0.1.9 1
```

あなたはスタンバイ非プロモートをmakeために `0` にスタンバイの優先度を設定することができます。スタンバイの優先度を `0` より大きい値に設定すると、`promotable=false` のプロパティ値がオーバーライドされます。

ノード `10.0.1.10` 上のプロパティファイルが `promotable=false` の設定が含まれており、フェイルオーバーのイベントに使用されるスタンバイする `10.0.1.10` のプロモーション優先順位を設定するために `efm set-priority` を使用した場合、例コマンドで指定された値は、の値を上書きしますプロパティファイル：

```
efm set-priority acctg 10.0.1.10 1
```

フェイルオーバーが発生しイベント、フェイルオーバーマネージャは、最初のスタンバイ・ノードは最新のデータを持って確認するためにストリーミングレプリケーションのPostgresから情報を取得し、データロスの最小確率を持つノードをプロモートしていきます。2つのスタンバイノードに等しく最新のデータが含まれている場合、`use.replay.tiebreaker`が `false` に設定されていない限り、ユーザー指定の優先順位の高いノードがプライマリに昇格します。スタンバイノードの優先度の値を確認するには、次の

コマンドを使用します。

```
efm cluster-status <cluster_name>
```

</div>

Note * 新しいプライマリが昇格されると、ノードの昇格優先度が変更されます。

- `efm set-priority` コマンドを使用してスタンバイがプロモーション可能かどうかを変更した場合、プロモーションまたはクラスターの分割と再結合により、スタンバイのプロパティファイルの値にリセットされる場合があります。

フェールオーバーマネージャーノードの昇格

Failover Manager クラスターの任意のノードで `efm promote` を呼び出して、スタンバイデータベースのプライマリデータベースへのマニュアル昇格をスタートできます。

手動昇格は、データベースクラスターのメンテナンス帯にのみ実行してください。利用可能な最新のスタンバイデータベースがない場合は、続行する前にプロンプトが表示されます。マニュアルプロモーションをスタートするには、`efm` または OS スーパーユーザの ID を想定して、コマンドを呼び出します。

```
efm promote <cluster_name> [-switchover] [-sourcenode <address>] [-quiet] [-n
```

どこで：

`<cluster_name>` は、フェールオーバーマネージャークラスターの名前です。

`-switchover` オプションを含めて、オリジナルのプライマリをスタンバイとして再構成します。`-switchover` キーワードを含める場合、クラスターにはプライマリノードと少なくとも1つのスタンバイを含める必要があり、ノードは同期している必要があります。

`-sourcenode` キーワードを含めて、リカバリ設定をプライマリにコピーするノードを指定します。

スイッチオーバー中の通知を抑制するには、`-quiet` キーワードを含めます。

`-noscripts` キーワードを含めて、フェールオーバーマネージャーがフェンシングおよびポストプロモーションスクリプトを呼び出さないようにします。

切り替え中：

- サーババージョン11以前の場合、`recovery.conf` ファイルは既存のスタンバイからプライマリノードにコピーされます。サーババージョン12以降では、`primary_conninfo` および `restore_command` パラメーターがコピーされ、メモリに保存されます。
- プライマリデータベースは停止しています。
- VIPを使用している場合、アドレスはプライマリノードから解放されます。
- プライマリノードを置き換えるためにスタンバイが昇格され、VIPを取得します。

- 新しいプライマリノードのアドレスが `recovery.conf` ファイルに追加されるか、`primary_conninfo` の詳細がメモリに保存されます。
- `application.name` プロパティがこのノードに設定されている場合、`application_name` プロパティが `recovery.conf` ファイルに追加されるか、`primary_conninfo` 情報がメモリに保存されます。
- サーババージョン12以降を使用している場合、メモリに保存されているリカバリ設定は `postgresql.auto.conf` ファイルに書き込まれます。`standby.signal` ファイルが作成されます。
- 古いプライマリが開始されます。エージェントは、スタンバイとしてのモニタリングを再開します。

プロモーション中、プライマリエージェントは仮想IPアドレスを解放します。スイッチオーバーではない場合、`db.data.dir` プロパティで指定されたディレクトリに `recovery.conf` ファイルが作成されます。`recovery.conf` ファイルは、ファイルが削除されるまで古いプライマリデータベースが起動しないようにするために使用され、ノードがクラスター内の2番目のプライマリとして起動しないようにします。昇格がスイッチオーバーのパートである場合、リカバリ設定は上記のように処理されます。

プライマリエージェントは実行されたままで、ステータスがIdleになります。

スタンバイエージェントは、既知のアドレスにpingを送信する前に仮想IPアドレスが使用されていないことを保証で、エージェントがネットワークから分離されていないことを確認します。スタンバイエージェントはフェンシングスクリプトを実行し、スタンバイデータベースをプライマリに昇格させます。スタンバイエージェントは、仮想IPアドレスをスタンバイノードに割り当て、プロモーション後スクリプト（該当する場合）を実行します。

このコマンドは、クラスタープロパティファイルの `auto.failover` パラメータで指定された値を無視するようにサービスに指示することに注意してください。

ノードをプライマリのロールに結果には、ノードをプロモーションリストの最初に配置しノード。

```
efm set-priority <cluster_name> <address> <priority>
```

次に、手動プロモーションを実行しマニュアル。

```
efm promote <cluster_name> -switchover
```

efmユーティリティの詳細については、[Using the EFM Utility](#)を参照してください。

</div>

フェールオーバーマネージャーエージェントの停止

エージェントを停止すると、フェールオーバーマネージャーは、クラスターの実行中のすべてのノードのクラスターメンバーリストからノードのアドレスを削除しますが、フェール

オーバーマネージャーの許可ノードホストリストからはアドレスを削除しません。

RHEL / CentOS 7.xまたはRHEL / CentOS 8.xでFailover Managerエージェントを停止するには、スーパーユーザ権限を引き受けて、コマンドを呼び出します。

```
systemctl stop edb-efm-4.2
```

あなたは `efm disallow-node` コマンド（可ノード・ホスト・リストからノードのアドレスを削除）を呼び出すまでは、最初は再び `efm allow-node` コマンドを実行せず、後日、ノードをリスタートする `service edb-efm-4.2 start` コマンドを使用することができません。

エージェントを停止しても、`primary.shutdown.as.failure`プロパティが `true` に設定されていない限り、エージェントが失敗したことをクラスターにシグナルしません。

フェールオーバーマネージャークラスターの停止

Failover Managerクラスターを停止するには、Failover Managerクラスターの任意のノードに接続し、`efm` またはOSスーパーユーザのIDを想定して、コマンドを呼び出します。

```
efm stop-cluster <cluster_name>
```

このコマンドにより、`* all *` Failover Managerエージェントが終了します。Failover Managerエージェントを終了すると、すべてのフェイルオーバー機能が完全に無効になります。

Note * `efm stop-cluster` コマンドを呼び出すと、許可ノードホストリストからすべての許可ノード情報が失われます。

クラスターからノードを削除する

`efm disallow-node` コマンドは、ノードのIPアドレスをFailover Managerの許可ノードホストリストから削除します。既存のノード（現在実行中のクラスターのパート）で `efm` またはOSスーパーユーザのIDを想定し、クラスター名前とノードのIPアドレスを指定して `efm disallow-node` コマンドを呼び出します。

```
efm disallow-node <cluster_name> <address>
```

`efm disallow-node` コマンドは、実行中のエージェントを停止しません。サービスが実行するまでノードで実行され続けます。その後、エージェントまたはクラスターが停止すると、ノードはクラスターに再参加できなくなり、フェイルオーバー優先順位リストから削除されます（昇格の対象外となります）。

`efm disallow-node` コマンドを呼び出した後、`efm allow-node` コマンドを使用してノードをクラスターに再度追加する必要があります。

</div>

単一ノードで複数のエージェントを実行する

フェールオーバーマネージャーノードで複数のプライマリエージェントまたはスタンバイエージェントを実行マルチプルにより、同じホストにあるマルチプルのデータベースクラスターをモニタできます。単一のノードでマルチプルのウィットネスエージェントを実行することもできます。複数のデータベースクラスターをモニタするようにフェールオーバーマネージャーを構成し、異なるクラスターのフェールオーバーマネージャーエージェントが相互に干渉しないようにするには、以下を行う必要があります。

1.各クラスターのメンバごとに、クラスター内のノードの一意のプロパティとロールのセットを定義するクラスタープロパティファイルを作成します。cluster.3のメンバーを一覧表示し、各クラスターの各メンバのためのクラスターメンバーのファイルを作成します。各クラスターのユニットファイル（RHEL / CentOS 7.xまたはRHEL / CentOS 8.xシステム）をカスタマイズして、クラスタープロパティとクラスターメンバーファイルの名前を指定します4。各クラスターのサービスを開始します。

以下の例では、同じノードで実行されている2つのデータベースクラスター（acctgとsales）を使用しています。

- acctg のデータは /opt/pgdata1 にあります。そのサーバーはポート 5444 をモニタリングします。
- sales のデータは /opt/pgdata2 にあります。そのサーバーはポート 5445 をモニタリングします。

これらの両方のデータベースクラスターに対してFailover Managerエージェントを実行するには、efm.properties.in テンプレートを使用して2つのプロパティファイルを作成します。各クラスタープロパティファイルには一意の名前が必要です。この例では、acctg および sales データベースクラスターにマッチするように acctg.properties および sales.properties を作成します。

次のパラメーターは、各クラスタープロパティファイルで一意である必要があります。

admin.port

bind.address

db.port

db.data.dir

virtual.ip （使用する場合）

virtual.ip.interface （使用する場合）

各クラスタープロパティファイルで、db.port パラメータは各クラスターに一意の値を指定する必要がありますが、db.user および db.database パラメータは同じ値または一意の値を持つことができます。例、acctg.properties ファイルが指定できます。

db.user=efm_user

```
db.password.encrypted=7c801b32a05c0c5cb2ad4ffbda5e8f9a
```

```
db.port=5444
```

```
db.database=acctg_db
```

`sales.properties` ファイルでは以下を指定できます：

```
db.user=efm_user
```

```
db.password.encrypted=e003fea651a8b4a80fb248a22b36f334
```

```
db.port=5445
```

```
db.database=sales_db
```

同じノードで複数のFailover Managerクラスタエージェントをセットアップする場合、一部のパラメータには特別な注意が必要です。同じノードにマルチプルのエージェントが存在する場合、各ポートは一意である必要があります。任意の2つのポートが機能しますが、互いに近すぎないポートを使用すると、情報を明確に保つ方が簡単な場合があります。

各クラスタのクラスタプロパティファイルを作成する場合、`db.data.dir` パラメータは、各データベースクラスタごとに一意の値も指定する必要があります。

仮想IPアドレスをノードに割り当てるときに、次のパラメーターが使用されます。Failover Managerクラスタが仮想IPアドレスを使用しない場合は、これらのパラメーターを空白のままにします。

```
virtual.ip
```

```
virtual.ip.interface
```

```
virtual.ip.prefix
```

このパラメータ値は、使用されている仮想IPアドレスによって決定され、`acctg.properties` と `sales.properties` の両方で同じ場合と異なる場合があります。

`acctg.properties` および `sales.properties` ファイルを作成した後、それぞれのプロパティファイルを指す各クラスタのサービススクリプトまたはユニットファイルを作成します。この手順はプラットフォーム固有です。RHEL / CentOS 7.xまたはRHEL / CentOS 8.xを使用している場合は、[RHEL/CentOS 7.x or RHEL/CentOS 8.x](#)を参照してください。

Note * ユニットファイルを使用している場合、Failover Managerをアップグレードときに、新しいサービス名前を反映するようにファイルを手動で更新する必要があります。

RHEL / CentOS 7.xまたはRHEL / CentOS 8.x

RHEL / CentOS 7.xまたはRHEL / CentOS 8.xを使用している場合、`edb-efm-4.2` ユニットファイルを、クラスタごとに一意の名前で新しいファイルにコピーする必要があります。あなたは（ACCTG・販売名前付けの）二つのクラスタを持っている場合例、ユニットファイルは次のようになります。

```
/usr/lib/systemd/system/efm-acctg.service
```

```
/usr/lib/systemd/system/efm-sales.service
```

次に、各ユニットファイル内の `CLUSTER` 変数を編集し、指定されたクラスター名前を `efm` から新しいクラスター名前ます。例、`acctg` 名前付けのクラスタのために、値のように指定します。

```
Environment=CLUSTER=acctg
```

また、`PIDfile` パラメータの値を更新して、新しいクラスター名前を指定する必要があります。例：

```
PIDFile=/var/run/efm-4.2/acctg.pid
```

サービススクリプトをコピーした後、次のコマンドを使用してサービスを有効にします。

```
# systemctl enable efm-acctg.service
```

```
# systemctl enable efm-sales.service
```

次に、新しいサービススクリプトを使用してエージェントをスタートします。例、次のコマンドで `acctg` エージェントをスタートすることができます。

```
# systemctl start efm-acctg`
```

ユニットファイルのカスタマイズについては、次をご覧ください。

< <https://docs.fedoraproject.org/en-US/quick-docs/understanding-and-administering-systemd/インデックス.html> >

Monitoring a Failover Manager Cluster

</ div>

Failover Manager `efm cluster-status` コマンドまたは PEM Client グラフィカルインタフェースを使用して、Failover Manager クラスターの監視対象ノードの現在のステータスを確認できます。

クラスターステータスレポートの確認

`efm cluster-status cluster properties file` コマンドは、Failover Manager クラスターのステータスに関する情報を含むレポートを返します。コマンドを呼び出すには、次を入力します。

```
# efm cluster-status <cluster_name>
```

次のステータスレポートは、3つのノードが実行名前付けedbという名前のクラスターレポートです。

Agent Type	Address	Agent	DB	VIP
Standby	172.19.10.2	UP	UP	192.168.225.190
Standby	172.19.12.163	UP	UP	192.168.225.190
Primary	172.19.14.9	UP	UP	192.168.225.190*

* Allowed node host list:
172.19.14.9 172.19.12.163 172.19.10.2
* Membership coordinator: 172.19.14.9
* Standby priority host list:
172.19.12.163 172.19.10.2

Promote Status:

DB Type	Address	WAL Received LSN	WAL Replayed LSN	Info
Primary	172.19.14.9		0/4000638	
Standby	172.19.12.163	0/4000638	0/4000638	
Standby	172.19.10.2	0/4000638	0/4000638	

* Standby database(s) in sync with primary. It is safe to promote.

クラスターステータスセクションには、クラスターの各ノードに存在するエージェントのステータスの概要が表示されます。

Agent Type	Address	Agent	DB	VIP
Standby	172.19.10.2	UP	UP	192.168.225.190
Standby	172.19.12.163	UP	UP	192.168.225.190
Primary	172.19.14.9	UP	UP	192.168.225.190*

VIPアドレスの後のアスタリスク(*)は、アドレスが接続に使用可能であることを示します。VIPアドレスの後にアスタリスクが付いていない場合、アドレスはノードに（プロパティファイルで）関連付けられていますが、アドレスは現在使用されていません。

フェールオーバーマネージャエージェントは、クラスターステータスセクションに表示される情報を提供します。

Allowed node host list と Standby priority host list を使用結合、クラスターへの参加が許可されているノードとノードの昇格オーダーを簡単に確認できます。メンバーシップコーディネーターのIPアドレスもレポートに表示されます。

```
Allowed node host list:
172.19.14.9 172.19.12.163 172.19.10.2
Membership coordinator: 172.19.14.9
Standby priority host list:
172.19.12.163 172.19.10.2
```

レポートの **Promote Status** セクションは、クラスターステータスコマンドを呼び出しているノードからクラスター内の各データベースへの直接クエリーの結果です。クエリーは、各データベースのトランザクションログの場所も返します。各データベースへのクエリは異なる時点で結果れるため、クラスターでストリーミングレプリケーションが正常に動作していても、LSNはマッチしない場合があります。

Promote Status:

DB Type	Address	WAL Received LSN	WAL Replayed LSN	Info
Primary	172.19.14.9		0/4000638	
Standby	172.19.12.163	0/4000638	0/4000638	
Standby	172.19.10.2	0/4000638	0/4000638	

データベースがダウンしている場合（またはデータベースが再起動されたが、再開コマンドがまだ呼び出されていない場合）、そのホストに存在するエージェントの状態はアイドルになります。エージェントがアイドル状態の場合、クラスターステータスレポートにはアイドルノードの状態の概要が含まれます。例：

Agent Type	Address	Agent	DB	VIP
Idle	172.19.18.105	UP	UP	172.19.13.105

終了コード

クラスターステータスプロセスは、クラスターの状態に基づいた終了コードを返します。

- **0** の終了コードは、すべてのエージェントが実行中であり、プライマリノードとスタンバイノードのデータベースが実行中で同期していることを示します。
- ゼロ以外の終了コードは、問題があることを示します。次の問題により、ゼロ以外の終了コードがトリガーされる場合があります。

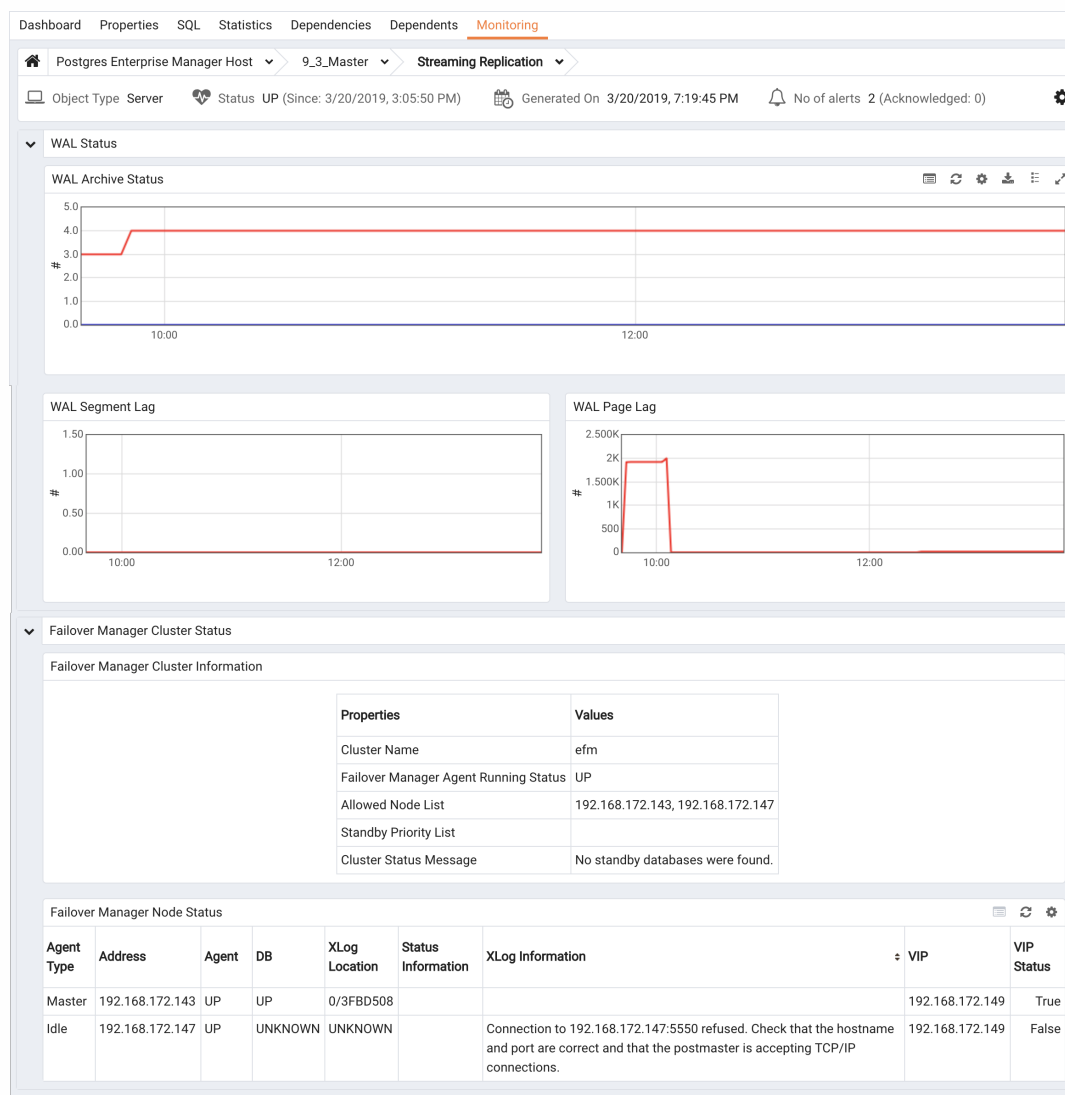
データベースがダウンしているか、不明です（またはアイドルエージェントがあります）。

- フェールオーバーマネージャーは、指定されたデータベースパスワードを解読できません。
- データベースに接続してWALの場所を取得する際に問題が発生しました。
- プライマリエージェントはありません。
- スタンバイエージェントはありません。

- 1つ以上のスタンバイノードがプライマリと同期していません。

Postgres Enterprise Managerを使用したストリーミングレプリケーションの監視

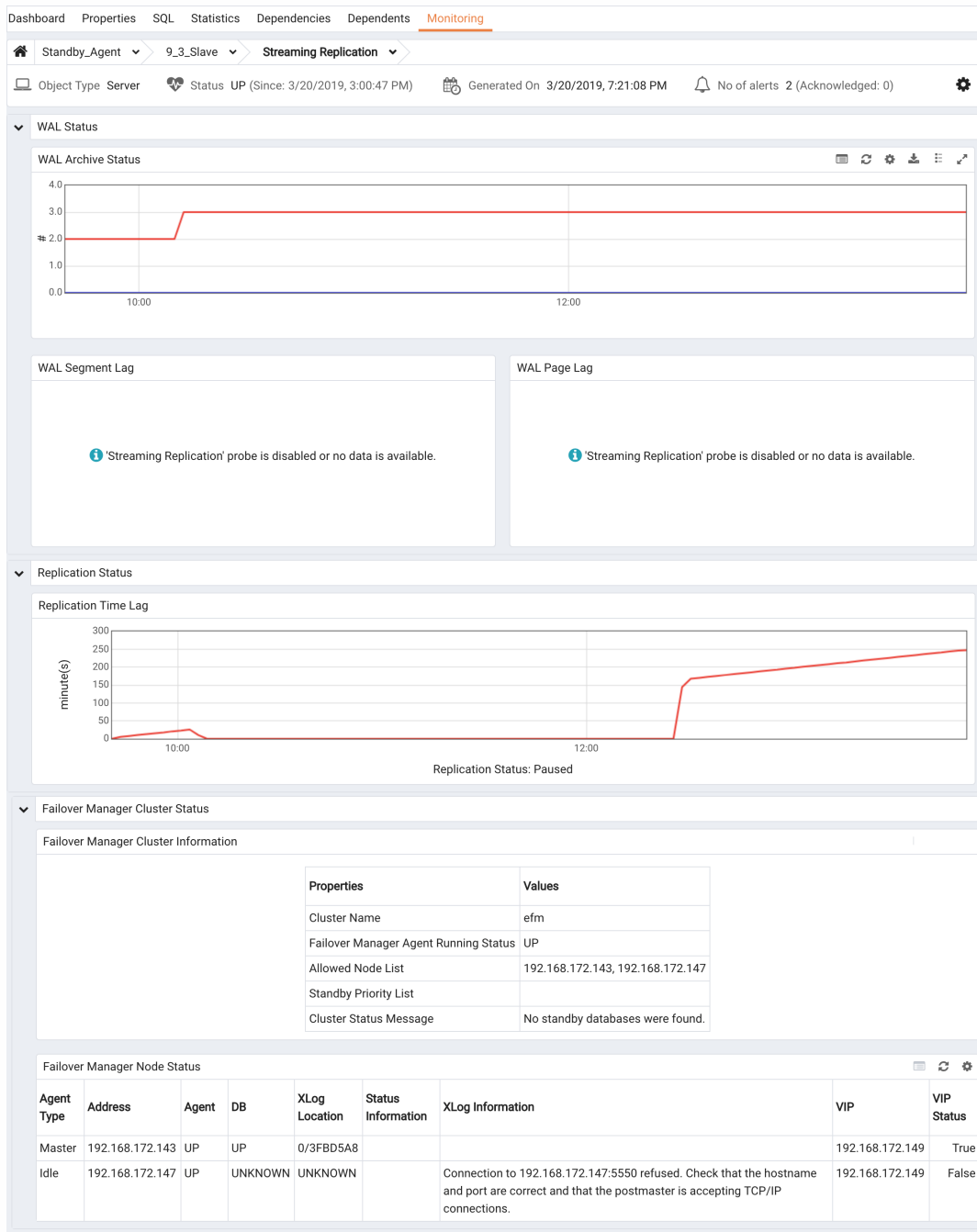
Postgres Enterprise Manager（PEM）をモニタてサーバーを監視する場合、Streaming Replication Analysisダッシュボード（PEMグラフィカルインタフェースのパート）を構成して、Streaming Replicationシナリオのパートであるプライマリノードまたはスタンバイノードの状態を表示できます。



The Streaming Replication dashboard (Primary node)

ストリーミングレプリケーション分析ダッシュボードには、ストリーミングレプリケーションが有効になっている監視対象サーバーのアクティビティに関する統計情報が表示されます。ダッシュボードヘッダーは、監視対象サーバーの状態（レプリケーションプライマリまたはレプリケーションスレーブのいずれか）を識別し、サーバーが最後に起動された日時、ページが最後に更新された日時、およびトリガーされたアラートの現在の数を表示しますサーバー用。

レプリケーションスレーブ（スタンバイノード）のダッシュボードを確認すると、ダッシュボードの下部にあるlabelでサーバーのステータスが確認されます。



The Streaming Replication dashboard (Standby node)

デフォルトでは、Streaming Replication Analysisダッシュボードに情報を提供するPEMレプリケーションプローブは無効になっています。

レプリケーションシナリオのプライマリノードのストリーミングレプリケーション分析ダッシュボードをビューするには、次のプローブを有効にする必要があります。

- ストリーミングレプリケーション

- WALアーカイブステータス

レプリケーションシナリオのスタンバイノードのストリーミングレプリケーション分析ダッシュボードをビューするには、次のプロンプトを有効にする必要があります。

- ストリーミングレプリケーションの遅延時間

PEMの詳細については、次のEnterpriseDB ウェブサイトにアクセスしてください。

<http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager>

Using the efm Utility

</ div>

フェールオーバーマネージャーは、クラスター管理を支援するefmユーティリティを提供します。RPMインストーラは、Failover Managerのインストール時に、ユーティリティを `/usr/edb/efm-42/bin` ディレクトリに追加します。

**** efm allow-node ****

</ div>

```
efm allow-node <cluster_name>
```

`efm allow-node` コマンドを呼び出して、指定したノードがクラスターに結合できるようにします。コマンドを呼び出すときに、クラスターの名前と参加ノードのIPアドレスを指定します。

このコマンドは、`efm`、`efm` グループのメンバ、またはルートによって呼び出される必要があります。

**** efm disallow-node ****

</ div>

```
efm disallow-node <cluster_name> <address>
```

`efm disallow-node` コマンド起動許可されたホストのリストから指定されたノードを除去するため、クラスターに参加するノードを防ぎます。`efm disallow-node` コマンドを呼び出すときに、クラスターの名前とノードのアドレスを指定します。このコマンドは、`efm` グループ、`efm` グループのメンバ、またはルートによって呼び出される必要があります。

**** efm cluster-status ****

</ div>

```
efm cluster-status <cluster_name>
```

`efm cluster-status` コマンドを呼び出して、フェールオーバーマネージャークラスターの状態を表示します。ステータスレポートの詳細については、[Monitoring a Failover Manager Cluster](#)を参照してください。

```
** efm cluster-status-json **
```

</ div>

```
efm cluster-status-json <cluster_name>
```

`efm cluster-status-json` コマンドを呼び出して、json形式でFailover Managerクラスターのステータスを表示しフォーマット。表示される情報のフォーマットはefm cluster-statusコマンドによって生成される表示とは異なりますが、情報ソースは同じです。

次の例は、3つのノードを持つ正常なクラスターのステータスを照会することにより生成されます。

```

{
*   "nodes": {
*       "172.16.144.176": {
*           "type": "Witness",
*           "agent": "UP",
*           "db": "N/A",
*           "vip": "",
*           "vip_active": false
*       },
*       "172.16.144.177": {
*           "type": "Primary",
*           "agent": "UP",
*           "db": "UP",
*           "vip": "",
*           "vip_active  :   false"
*           "xlogReceive :   0/14001478"
*           "xlog       :   0/14001478"
*           "xloginfo    :"
*       },
*       "172.16.144.180": {
*           "type": "Standby",
*           "agent": "UP",
*           "db": "UP",
*           "vip": "",
*           "vip_active  :   false"
*           "xlogReceive :   0/14001478"
*           "xlog       :   0/14001478"
*           "xloginfo    :"
*       }
*   },
*   "allowednodes": [
*       "172.16.144.177",
*       "172.16.144.160",
*       "172.16.144.180",
*       "172.16.144.176"
*   ],
*   "membershipcoordinator": "172.16.144.177",
*   "failoverpriority": [
*       "172.16.144.180"
*   ],
*   "minimumstandbys": 0,
*   "missingnodes": [],
*   "messages": []
}

```

**** efm encrypt ****

</ div>

```
efm encrypt <cluster_name> [--from-env]
```

クラスタプロパティファイルにパスワードを含める前に、 **efm encrypt** コマンドを呼び出

してデータベースパスワードを暗号化します。 `--from-env` オプションを含めて、フェールオーバーマネージャーに `EFMPASS` 環境変数で指定された値を使用し、ユーザ入力なしで実行ように指示します。詳細については、[Encrypting Your Database Password](#)を参照してください。

**** EFMプロモート ****

</ div>

```
efm promote cluster_name [-switchover [-sourcenode <address>]][-quiet][-noscri
```

`efm promote` コマンドは、フェールオーバーマネージャーにスタンバイからプライマリへのマニュアルフェイルオーバーを実行するよう指示します。

手動昇格は、ステータスコマンドで、クラスターにプライマリの最新のスタンバイノードが含まれていることが報告された場合にのみ試行してください。最新のスタンバイがない場合、フェールオーバーマネージャーは続行する前にプロンプトます。

`-switchover` 句を含めてスタンバイノードをプロモートさせ、プライマリノードをスタンバイノードとして再構成します。 `-sourcenode` キーワードを含め、ノードアドレスを指定して、リカバリ設定が古いプライマリノードにコピーされる（スタンバイ）ノードを示します。 `-quiet` キーワードを含めて、スイッチオーバープロセス中の通知を抑制します。フェンダーまたはプロモーション後のスクリプトを呼び出さないようにフェールオーバーマネージャーに指示するには、 `-noscripts` キーワードを含めます。

このコマンドは、 `efm` 、 `efm` グループのメンバ、またはルートによって呼び出される必要があります。

Note * このコマンドは、クラスタープロパティファイルの `auto.failover` パラメータで指定された値を無視するようにサービスに指示します。

**** EFM履歴書 ****

</ div>

```
efm resume <cluster_name>
```

`efm resume` コマンドを呼び出して、以前に停止したデータベースのモニタリングを再開します。このコマンドは、`efm`、`efm`グループのメンバ、またはルートによって呼び出される必要があります。

**** efm set-priority ****

</ div>

```
efm set-priority <cluster_name> <address> <priority>
```

efm set-priority コマンドを呼び出して、フェイルオーバー優先順位をスタンバイノードに割り当てます。この値は、フェイルオーバーしたイベントにノードが使用されるオーダーを指定します。このコマンドは、**efm**、**efm** グループのメンバ、またはルートによって呼び出される必要があります。

優先度オプションを使用して、優先度リスト内のノードの場所を指定します。例、ノードがプライマリスタンバイであることを示すために1の値を指定し、フェイルオーバーのイベントに促進最初のノードであろう。優先順位値0は、フェールオーバーマネージャーにスタンバイをプロモートさせないように指示します。

```
** efm stop-cluster **
```

```
</ div>
```

```
efm stop-cluster <cluster_name>
```

efm stop-cluster コマンドを呼び出して、すべてのノードでフェールオーバーマネージャーを停止します。このコマンドは、フェールオーバーマネージャーにクラスター上の各ノードに接続し、既存のメンバーにシャットダウンするよう指示します。このコマンドは実行中のデータベースには影響しませんが、コマンドが完了すると、フェイルオーバー保護は機能しません。

Note * **efm stop-cluster** コマンドを呼び出すと、許可されたすべてのノード情報が許可ノードホストリストから削除されます。

このコマンドは、**efm**、**efm** グループのメンバ、またはルートによって呼び出される必要があります。

```
** efm upgrade-conf **
```

```
</ div>
```

```
efm upgrade-conf <cluster_name> [-source <directory>]
```

efm upgrade-conf コマンド呼び出す既存のフェールオーバーマネージャーのインストールから設定ファイルをコピーして、フェイルオーバーManagerのインストールに必要なパラメータを追加します。ユーティリティを呼び出すときに、以前のクラスターの名前を指定します。このコマンドは、ルート権限で呼び出す必要があります。

sudoを使用しないFailover Manager構成からアップグレード処理場合は、**-source** フラグを含めて、upgrade-confを呼び出すときに構成ファイルが存在するディレクトリの名前を指定します。

```
** efm node-status-json **
```

```
</ div>
```

```
efm node-status-json <cluster_name>
```

`efm node-status-json` コマンドを呼び出して、ローカルノードのステータスをjson形式で表示しフォーマット。このコマンドが正常に実行されると、終了コードとして `0` が返されます。データベース障害またはエージェントステータスがIDLEになった場合、コマンドは `1` を終了コードとして返します。

次に、`efm node-status-json` コマンドの出力例です。

```
{
  * "type":"Standby",
  * "address":"172.16.144.130",
  * "agent":"UP",
  * "db":"UP",
  * "vip":"",
  * "vip_active":"false"
}
```

**** EFM-ヘルプ****

</div>

```
efm --help
```

`efm --help` コマンドを呼び出して、Failover Managerユーティリティコマンドのオンラインヘルプを表示します。

Controlling the Failover Manager Service

</div>

フェールオーバーマネージャークラスター内の各ノードは、サービススクリプトによって制御されるフェールオーバーマネージャージェントをホストします。デフォルトでは、サービススクリプトは以下を見つけることを期待しています。

- Failover Managerサービスで使用するプロパティを含む `efm.properties` という名前付けの構成ファイル。レプリケーションシナリオの各ノードに関する情報を提供するプロパティファイルが含まれている必要があります。
- クラスタメンバーのファイルは、クラスタメンバーのリストが含まれ `efm.nodes` の名前付け。レプリケーションシナリオの各ノードには、クラスタメンバーリストが含まれている必要があります。

単一ノードでマルチプルのクラスターを実行している場合、クラスター固有の名前で構成ファイルを手動で作成し、対応するクラスターのサービススクリプトを変更する必要があることに注意してください。

Failover Managerサービスを制御するコマンドはプラットフォーム固有です。

</div>

RHEL / CentOS 7.xおよびRHEL / CentOS 8.xでのsystemctlユーティリティの使用

RHEL / CentOS 7.xおよびRHEL / CentOS 8.xでは、Failover Managerは、`/usr/lib/systemd/system`にある（デフォルトで）`edb-efm-4.2.service` 名前付けのLinuxサービスとして実行されます。フェールオーバーマネージャーによって監視される各データベースクラスターは、レプリケーションクラスターの各ノードでサービスのコピーを実行します。

次のsystemctlコマンドを使用して、RHEL / CentOS 7.xおよびRHEL / CentOS 8.xホストにあるFailover Managerエージェントを制御します。

```
systemctl start edb-efm-4.2
```

スタートコマンドは、現在のノードでFailover Managerエージェントを起動します。ローカルフェールオーバーマネージャーエージェントはローカルデータベースを監視し、他のノード上のフェールオーバーマネージャーと通信します。フェールオーバーマネージャークラスター内のノードは、任意のオーダーでスタートできます。このコマンドはルートによって呼び出される必要があります。

```
systemctl stop edb-efm-4.2
```

現在のノードでフェールオーバーマネージャーを停止します。このコマンドはルートによって呼び出される必要があります。

```
systemctl status edb-efm-4.2
```

statusコマンドは、呼び出されたFailover Managerエージェントのステータスを返します。任意のノードでstatusコマンドを呼び出して、フェールオーバーマネージャーにステータスとサーバーのスタートアップ情報を結果ように指示できます。

```
[root@ONE ~]]> systemctl status edb-efm-4.2
* edb-efm-4.2.service - EnterpriseDB Failover Manager 4.2
*   Loaded: loaded (/usr/lib/systemd/system/edb-efm-4.2.service; disabled;
*   Active: active (running) since Wed 2013-02-14 14:02:16 EST; 4s ago
*   Process: 58125 ExecStart=/bin/bash -c /usr/edb/edb-efm-4.2/bin/runefm.sh
Main PID: 58180 (java)
*   CGroup: /system.slice/edb-efm-4.2.service
*           └─58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x
```

Controlling Logging

</div>

フェールオーバーマネージャーは、エージェントごとに1つのログファイルと、エージェントごとに1つのスタートアップログを `/var/log/<cluster_name>-4.2`（`<cluster_name>` がクラスタの名前を指定する）に書き込み、保存します。

[cluster properties file](#)の `jgroups.loglevel` および `efm.loglevel` パラメーターを変更ことにより、エージェントログに書き込まれる詳細レベルを制御できます。

```
# Logging levels for JGroups and EFM.
# Valid values are: TRACE, DEBUG, INFO, WARN, ERROR
# Default value: INFO
# It is not necessary to increase these values unless debugging a
# specific issue. If nodes are not discovering each other at
# startup, increasing the jgroups level to DEBUG will show
# information about the TCP connection attempts that may help
# diagnose the connection failures.
jgroups.loglevel=INFO
efm.loglevel=INFO
```

ロギングファシリティは、Javaロギングライブラリとロギングレベルを使用します。ログレベルは（ほとんどのロギング出力からオーダー出力されます）：

```
- TRACE > DEBUG > INFO > WARN > ERROR
```

あなたは `WARN` に `efm.loglevel` パラメータを設定した場合、フェールオーバーマネージャーは、のみ `WARN` レベルと（`WARN` と `ERROR`）上記のログメッセージされます。

デフォルトでは、フェールオーバーマネージャーのログファイルは毎日ローテーションされ、圧縮され、1週間保存されます。ログローテーションファイル（`/etc/logrotate.d/efm-4.2`）の設定を変更することにより、ファイルローテーションスケジュールを変更できます。ログローテーションスケジュールの変更の詳細については、[logrotateのマニュアルページ](#)。

```
$ man logrotate
```

</div>

syslogログファイルエントリの有効化

Failover Managerは、syslogロギングをサポートしています。syslogロギングを実装ロギングには、syslogを構成してUDPまたはTCP接続を許可する必要があります。

syslogへの接続を許可するには、`/etc/rsyslog.conf` ファイルを編集して、使用するプロトコルのコメントを外します。また、プロトコルに関連付けられた `UDPServerRun` または `TCPServerRun` エントリーに、ログエントリの送信先のポート番号が含まれている保証する必要があります。例、次の構成ファイルのエントリは、ポート514へのUDP接続を有効にします。

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

次の構成ファイルエントリは、ポート514へのTCP接続を有効にします。

```
# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

syslog構成ファイルを変更した後、`rsyslog` サービスをリスタートして接続を有効にします。

```
systemctl restart rsyslog.service
```

フェールオーバーマネージャーホストの `rsyslog.conf` ファイルを変更した後、フェールオーバーマネージャーのプロパティを変更してロギングを有効にする必要があります。選択したエディタを使用して、実装するロギングのタイプを `()` に指定します。

```
# Which logging is enabled.
file.log.enabled=true
syslog.enabled=false
```

また、システムにも必要です。パラメータを使用してプロトコルタイプ（UDPまたはTCP）を指定し、`syslog.port` パラメータしてsyslogホストのリスナポートを指定します。`syslog.facility` 値は、エントリーを作成したプロセスの識別子として使用できます。値は `LOCAL0` と `LOCAL7` の間でなければなりません。

```
# Syslog information. The syslog service must be listening # on the
port for the given protocol, which can be UDP or
# TCP. The facilities supported are LOCAL0 through LOCAL7.
# syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

syslogの詳細については、syslogのマニュアルページを参照してください。

```
syslog man
```

Notifications

</div>

フェールオーバーマネージャーは、クラスターに影響する重要なイベントが発生すると、電子メール通知を送信したり、通知スクリプトを呼び出したりします。メール通知を送信するようにフェールオーバーマネージャーを構成した場合は、クラスターの各ノードのポート25

で実行されているSMTPサーバーが必要です。次のパラメーターを使用して、フェールオーバーマネージャーの通知動作を構成します。

```
user.email  
script.notification  
from.email
```

構成プロパティの編集の詳細については、[Specifying Cluster Properties](#)を参照してください。

通知の本文は、通知をトリガしイベントについて、およびクラスタの現在の状態についての詳細が含まれています。例：

```
EFM node: 10.0.1.11  
Cluster name: acctg  
Database name: postgres  
VIP: ip_address (Active|Inactive)  
Database health is not being monitored.
```

ノードに実装されている場合、VIPフィールドには仮想IPのIPアドレスと状態が表示されません。

フェールオーバーマネージャーは、各通知に重大度レベルを割り当てます。次のレベルは、必要な注意のレベルが高まることを示しています。

- **INFO** は、エージェントに関する情報メッセージを示しており、マニュアル介入は必要ありません（例、フェールオーバーマネージャーの起動または停止）。[List of INFO level notifications](#)を参照
- **WARNING** は、管理者がシステムをチェックする必要があるイベントが発生したことを示します（例、フェールオーバーが発生した）。[List of WARNING level notifications](#)を参照
- **SEVERE** は、重大なイベントが発生し、管理者の即時の注意が必要であることを示します（例、フェールオーバーが試行されましたが完了できませんでした）。[List of SEVERE level notifications](#)を参照

重大度レベルは、通知の緊急度を指定します。**INFO** の重大度レベルを持つ通知は、ユーザーのアクションを必要としないクラスタについての操作情報にご注意を促しますが、**SEVERE** の重大度レベルとの通知が、すぐにユーザーの注意が必要です。通知の重大度レベルはロギングレベルとは関係ありません。すべての通知は、構成ファイルで指定されたログレベルの詳細に関係なく送信されます。

[notification.level](#)プロパティを使用して、通知をトリガーする最小の重大度レベルを指定できます。

Note * 管理用メールアドレスへの通知の送信に加えて、すべての通知はエージェントログファイル（`/var/log/efm-4.2/<cluster name>.log`）に記録されます。

以下の表にリストされている条件は、**INFO** レベルの通知をトリガーします。

</ div>

Subject	Description
Executed fencing script	実行されたフェンシングスクリプト * script_name 結果 : script_results *
Executed post-promotion script	実行されたプロモーション後スクリプト * script_name 結果 : script_results *
Executed remote pre-promotion script	実行されたりモートプレプロモーションスクリプト * script_name 結果 : script_results *
Executed remote post-promotion script	実行されたりモートポストプロモーションスクリプト * script_name 結果 : script_results *
Executed post-database failure script	実行されたデータベース障害後スクリプト * script_name 結果 : script_results *
Executed primary isolation script	実行されたプライマリ分離スクリプト * script_name 結果 : script_results *
Witness agent running on <i>node_address</i> for cluster <i>cluster_name</i>	監視エージェントが実行されています。
Primary agent running on <i>node_address</i> for cluster <i>cluster_name</i>	プライマリエージェントが実行されており、データベースの状態が監視されています。
Standby agent running on <i>node_address</i> for cluster <i>cluster_name</i>	スタンバイエージェントが実行されており、データベースの状態が監視されています。
Idle agent running on node <i>node_address</i> for cluster <i>cluster_name</i>	アイドルエージェントが実行されています。ローカルデータベースを起動した後、エージェントを再開できます。
Assigning VIP to node <i>node_address</i>	VIP * VIP_address をノード <i>node_address</i> に割り当てる結果 : script_results *

Subject	Description
Releasing VIP from node <i>node_address</i>	ノード* <i>node_address</i> からVIP <i>VIP_address</i> を解放する結果： <i>script_results</i> *
Starting auto resume check for cluster <i>cluster_name</i>	このノードのエージェントは、障害が発生したデータベースのモニタリングを再開できるかどうかを確認するために、* <i>auto.resume.period</i> *秒ごとにチェックします。この時間中にクラスタを確認し、データベースが再び起動しない場合はエージェントを停止する必要があります。詳細については、エージェントログを参照してください。
Executed agent resumed script	実行されたエージェントはスクリプト <i>script_name</i> を再開しました結果： <i>script_results</i>
WAL logs backed up during promotion	このスタンバイを新しいプライマリーに追従するように再構成すると、 <i>pg_xlog</i> または <i>pg_wal</i> の内容が <i>pgdata</i> ディレクトリにバックアップされました。このバックアップは、ディスクスペースをフリーするのに都合が良いときに削除する必要があります。

以下の表にリストされている条件は、警告レベルの通知をトリガーします。

</ div>

Subject	Description
Witness agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	監視エージェントが終了しました。
Primary agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。
Cluster <i>cluster_name</i> notified that primary agent has left	プライマリーエージェントが再起動されるまで、クラスタのフェールオーバーは無効になります。

Subject	Description
Standby agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。
Agent exited during promotion on <i>node_address</i> for cluster <i>cluster name</i>	データベースの状態は監視されていません。
Agent exited on <i>node_address</i> for cluster <i>cluster name</i>	エージェントは終了しました。これは、アイドル状態のエージェントによって生成されます。
Agent exited for cluster <i>cluster name</i>	エージェントは終了しました。スタートアップが完了する前に、エージェントが終了したときに、この通知は、通常、スタートアップ時に生成されます。
Virtual IP address assigned to non-primary node	仮想IPアドレスは、非プライマリノードに割り当てられているようです。競合を避けるため、フェールオーバーマネージャーはVIPをリリースします。VIPがプライマリノードに割り当てられていることを確認し、割り当てられていない場合は手動でアドレスを再割り当てする必要があります。
Virtual IP address not assigned to primary node.	仮想IPアドレスは、プライマリノードに割り当てられていないようです。 EDB Postgres Failover ManagerはVIPの再取得を試みます。
No standby agent in cluster for cluster <i>cluster name</i>	* <i>cluster_name</i> *のスタンバイがクラスターを離れました。
Standby agent failed for cluster <i>cluster name</i>	* <i>cluster_name</i> *のスタンバイエージェントがクラスターを離れましたが、コーディネーターはスタンバイデータベースがまだ実行中であることを検出しました。
Standby database failed for cluster <i>cluster name</i>	スタンバイエージェントは、データベースに障害が発生したことを通知しました。他のノードもスタンバイデータベースに到達できません。
Standby agent cannot reach database for cluster <i>cluster name</i>	スタンバイエージェントはデータベースの障害を通知しましたが、他のノードはスタンバイデータベースがまだ実行中であることを検出しました。
Cluster <i>cluster name</i> has dropped below three nodes	完全なフェイルオーバー保護には、少なくとも3つのノードが必要です。監視ノードまたはエージェントノードをクラスターに追加してください。

Subject	Description
Subset of cluster <i>cluster name</i> disconnected from primary	このノードは、クラスターの大部分 <i>クラスター名</i> 前に接続されなくなりました。このノードはクラスターのサブセットのパートであるため、フェイルオーバーは試行されません。可視されている現在のノードは、次のとおりです。* <i>NODE_ADDRESS</i> *。
Promotion has started on cluster <i>cluster name</i> .	クラスター* <i>cluster_name</i> *でスタンバイのプロモーションが開始されました。
Witness failure for cluster <i>cluster_name</i>	* <i>node_address</i> *で実行されている目撃者がクラスターを離れました。
Idle agent failure for cluster <i>cluster_name</i> .	* <i>node_address</i> *で実行されているアイドルエージェントがクラスターを離れました。
One or more nodes isolated from network for cluster <i>cluster_name</i>	このノードはネットワークから隔離されているようです。クラスターで見られる他のメンバーは次のとおりです： <i>node_name</i>
Node no longer isolated from network for cluster <i>cluster_name</i> .	このノードは、もはやネットワークから隔離されていません。
Failover Manager tried to promote, but primary DB is still running	フェールオーバーマネージャーは昇格手順を開始しましたが、プライマリデータベースが* <i>address</i> *で実行されていることを検出しました。これは通常、プライマリEFMエージェントが終了したことを示しています。フェイルオーバーは発生していません。プライマリエージェントが再起動されるまで、フェイルオーバー保護はありません。
Primary agent missing for cluster <i>cluster_name</i>	プライマリエージェントは以前にクラスターを離れました。プライマリエージェントがクラスターに参加するまで、フェイルオーバー保護はありません。
Standby agent started to promote, but primary has rejoined.	スタンバイEFMエージェントは自分自身のプロモートを開始しましたが、プライマリエージェントがクラスターに再参加したことがわかりました。フェイルオーバーは発生していません。
Standby agent tried to promote, but could not verify primary DB	スタンバイEFMエージェントは自分自身をプロモートせようとしたが、* <i>node_address</i> *でプライマリDBがまだ実行されているかどうかを検出できませんでした。フェイルオーバーは発生していません。

Subject	Description
Standby agent tried to promote, but VIP appears to still be assigned	スタンバイEFMエージェントは自分自身をプロモートしようとしたが、仮想IPアドレス（VIP_address）がまだ別のノードに割り当てられているように見えるため、できませんでした。このような状況で昇格すると、データが破損する可能性があります。フェイルオーバーは発生していません。
Standby agent tried to promote, but appears to be orphaned	スタンバイEFMエージェントは自分自身をプロモートしようとしたが、既知のサーバー（server_address）に到達できなかったためできませんでした。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェイルオーバーは発生していません。
Potential manual failover required on cluster <i>cluster_name</i> .	クラスター* <i>cluster_name</i> *の潜在的なフェイルオーバーシチュエーションが検出されました。このクラスターでは自動フェイルオーバーが無効になっているため、マニュアルでの介入が必要です。
Failover has completed on cluster <i>cluster_name</i>	クラスター* <i>cluster_name</i> *でフェイルオーバーが完了しました。
Lock file for cluster <i>cluster_name</i> has been removed	クラスター* <i>cluster_name</i> のロックファイルは、ノード <i>node_address</i> *の <i>path_name</i> から削除されました。このロックは、マルチプルのエージェントが同じノード上の同じクラスターをモニタリングすることを防ぎます。このファイルを復元して、クラスターの別のエージェントを誤って起動しないようにしてください。
A recovery file for cluster <i>cluster_name</i> has been found on primary node	クラスター* <i>cluster_name</i> のリカバリファイルが、プライマリノード <i>node_address</i> *の <i>path_name</i> で見つかりました。このノードでDBをリスタートしようとする、これは問題になる可能性があります。
recovery_target_timeline is not set to latest in recovery settings	recovery_target_timelineパラメータは、リカバリ設定で最新に設定されていません。スタンバイサーバは、新しいプライマリが昇格したときに発生するタイムラインの変更を追跡できません。
Promotion has not occurred for cluster <i>cluster_name</i>	昇格が試みられましたが、昇格されるノードがすでに存在します：ip_address。
Standby will not be reconfigured after failover in cluster <i>cluster_name</i>	このノードの「auto.reconfigure」プロパティはfalseに設定されているため、昇格後に新しいプライマリノードを追跡するように再構成されません。

Subject	Description
Could not resume replay for standby standby_id.	スタンバイのリプレイを再開できませんでした。手動による介入が必要になる場合があります。エラー：error_message。
Possible problem with database timeout values	あなたのリモート。タイムアウト値（値）がローカルよりも高い。タイムアウト値（値）。ローカルデータベースが応答するために時間がかかりすぎる場合は、ローカルエージェントは他のエージェントが接続できますが、データベースが失敗したことを前提としていことができませんでした。これによりフェイルオーバーが発生することはありますが、ローカルエージェントが監視を停止モニタリングように強制され、フェイルオーバー保護が失われる可能性があります。
No standbys available for promotion in cluster cluster_name	クラスター内のスタンバイノードの現在の数は、最小数 number に低下しました。別のスタンバイノードを追加または昇格可能にしない限り、フェイルオーバーはありません。
No promotable standby for cluster cluster_name	クラスター内の現在のフェイルオーバー優先順位リストは空です。クラスター* cluster_name *の唯一の昇格可能なスタンバイを削除しました。別の昇格可能なスタンバイノードを追加するか、フェイルオーバー優先順位リストに追加して昇格可能にしない限り、フェイルオーバーはありません。
Synchronous replication has been reconfigured for cluster cluster_name	クラスター内の同期的スタンバイノードの数が数を下回っています。プライマリの同期的スタンバイ名は、新しいsynchronous_standby_names値に再構成されました。
Synchronous replication has been reconfigured for cluster cluster_name	プライマリのsynchronous_standby_namesはnew_synchronous_standby_namesに再構成されました
Synchronous replication has been disabled for cluster cluster_name.	クラスター内の同期的スタンバイノードの数がカウントを下回りました。プライマリは同期レプリケーションモードから削除されました。
Could not reload database configuration.	データベース構成をリロードできませんでした。手動による介入が必要です。エラー：error_message。
Custom monitor timeout for cluster cluster_name	次のカスタムモニタリングスクリプトがタイムアウトしました：script_name
Custom monitor 'safe mode' failure for cluster cluster_name	次のカスタム・モニタ・スクリプトが失敗したが、モードで実行されている：SCRIPT_NAME。出力：script_results

Subject	Description
primary.shutdown.as.failure set to true for primary node	プライマリ。このクラスターのシャットダウン.as.failureプロパティが真に設定されています。クラスタ全体を停止せずにプライマリエージェントを停止すると、クラスタの残りの部分では、プライマリエージェントの即時の障害として扱われます。プライマリデータベースのメンテナンスが必要な場合は、プライマリエージェントをシャットダウンし、フェイルオーバーが発生しないという残りのノードからの通知を待ちます。
Primary_or_Standby cannot ping local database for cluster <i>cluster_name</i>	Primary_or_Standbyエージェントは、* node_address *で実行されているローカルデータベースに到達できなくなりました。他のノードはデータベースにリモートでアクセスできるため、エージェントはIDLEになり、データベースのモニタリングを再開しようとします。
Standby cannot resume monitoring local database for cluster <i>cluster_name</i>	スタンバイエージェントは、* node_address *で実行されているローカルデータベースに到達できなくなりました。他のノードは、データベースにリモートでアクセスできます。データベースのモニタリングを再開するために再開コマンドが実行されるまで、スタンバイエージェントはIDLEのままになります。
Primary agent left the cluster and node detached from load balancer	スタンバイエージェントは、* node_address *で実行されているローカルデータベースに到達できなくなりました。他のノードは、データベースにリモートでアクセスできます。データベースのモニタリングを再開するために再開コマンドが実行されるまで、スタンバイエージェントはIDLEのままになります。
Standby database in cluster <i>cluster_name</i> not stopped before primary is promoted	スタンバイ。リスタート.delayプロパティがこのエージェントに設定されているため、昇格が完了してからnum_seconds秒まで新しいプライマリに従うように再構成されません。場合によっては、マニュアルで介入しないと新しいプライマリを追跡できない場合があります。

</ div>

以下の表にリストされている条件は、* SEVERE *通知をトリガーします。

Subject	Description
Standby database restarted but EFM cannot connect	データベースのスタートまたはリスタートコマンドは正常に実行されましたが、データベースは接続を受け入れていません。EFMは、接続をリスタートするまで接続を試行し続けます。タイムアウト秒。

Subject	Description
Unable to connect to DB on <i>node_address</i>	最大接続制限に達しました。
Unable to connect to DB on <i>node_address</i>	データベースのパスワードが無効です。ユーザ=ユーザー名。
Unable to connect to DB on <i>node_address</i>	許可の指定が無効です。
Primary cannot resume monitoring local database for cluster <i>cluster_name</i>	プライマリエージェントは、* <i>node_address</i> * で実行されているローカルデータベースにア クセスできなくなりました。他のノードは データベースにリモートでアクセスできるた め、プライマリはVIPをリリースしたり、リカ バリ.confファイルを作成したりしません。 データベースのモニタリングを再開するた めに再開コマンドが実行されるまで、プライマ リエージェントはIDLEのままになります。
Fencing script error	フェンシングスクリプト <i>script_name</i> を正常に 実行できませんでした。終了値： <i>exit_code</i> 結 果： <i>script_results</i> フェイルオーバーは発生して いません。
Post-promotion script failed	プロモーション後スクリプト <i>script_name</i> は正 常に実行できませんでした。終了値： <i>exit_code</i> 結果： <i>script_results</i>
Remote post-promotion script failed	リモートポストプロモーションスクリプト <i>script_name</i> を正常に実行できませんでした終 了値： <i>exit_code</i> 結果： <i>script_results</i> Node：* <i>node_address</i> *
Remote pre-promotion script failed	リモート事前プロモーションスクリプト <i>script_name</i> が正常に実行できませんでした終 了値： <i>exit_code</i> 結果： <i>script_results</i> Node：* <i>node_address</i> *
Post-database failure script error	データベース障害後スクリプト <i>script_name</i> を 正常に実行できませんでした。終了値： <i>exit_code</i> 結果： <i>script_results</i>
Agent resumed script error	エージェントはスクリプト <i>script_name</i> を再開 しましたが、正常に実行できませんでした。 結果： <i>script_results</i>

Subject	Description
Primary isolation script failed	プライマリ分離スクリプトscript_nameは正常に実行できませんでした。終了値：exit_code 結果：script_results
Could not promote standby	ノードでプロモートコマンドが失敗しました。スタンバイをプロモートできませんでした。エラーの詳細：error_details
Error creating recovery.conf file on node_address for cluster cluster_name	プロモーション中のプライマリノード* NODE_ADDRESS *上のリカバリ.confファイルの作成中にエラーが発生しました。プロモーションは継続しましたが、古いプライマリノードを再起動できないように保証にはマニュアルでの介入が必要です。エラーの詳細：message_details
An unexpected error has occurred for cluster cluster_name	このノードで予期しないエラーが発生しました。詳細については、エージェントログを確認してください。エラー：error_details
Primary database being fenced off for cluster cluster_name	プライマリデータベースは、クラスターの大部分から分離されています。クラスターは、フェイルオーバーマネージャクラスターの残りの部分がスタンバイをプロモートするときに、2つのプライマリを防ぐために、ip_addressのプライマリエージェントにプライマリデータベースをフェンスするように指示しています。
Isolated primary database shutdown.	分離されたプライマリデータベースは、フェイルオーバーマネージャによってシャットダウンされました。
Primary database being fenced off for cluster cluster_name	プライマリデータベースは、クラスターの大部分から分離されています。プライマリが分離の検出を完了する前に、スタンバイが昇格され、クラスター内のこのノードに再参加しました。このノードは、複数のプライマリデータベースを避けるために自分自身を隔離しています。
Could not assign VIP to node node_address	フェールオーバーマネージャは、何らかの理由でVIPアドレスを割り当てることができませんでした。
primary_or_standby database failure for cluster cluster_name	指定されたノードでデータベースが発生しました。

Subject	Description
Agent is timing out for cluster <i>cluster_name</i>	このエージェントは、ローカルデータベースに到達しようとしてタイムアウトしました。タイムアウト後、エージェントはデータベースに正常にピングでき、モニタリングを再開しました。ただし、データベースまたはエージェントの障害の可能性を防ぐため、ノードが正常に実行されていることを確認するmakeにノードをチェックする必要があります。
Resume timed out for cluster <i>cluster_name</i>	このエージェントは、ローカルデータベースを再構成して再起動した後、モニタリングを再開できませんでした。詳細については、エージェントログを参照してください。
Internal state mismatch for cluster <i>cluster_name</i>	フェイルオーバーマネージャクラスターの内部状態は、クラスターメンバーの実際の状態とマッチしませんでした。これはまれであり、ノードがクラスターに参加したり、状態を変更したりするタイミングの問題が原因である可能性があります。問題を解決する必要がありますが、クラスターの状態も確認して確認する必要があります。不一致の詳細は、エージェントログファイル。
Failover has not occurred	エージェントは、クラスター* <i>cluster_name</i> *でプライマリデータベースが使用できなくなったことを検出しましたが、フェイルオーバーに使用できるスタンバイノードがありません。
Failover has not occurred	エージェントは、プライマリデータベースがクラスター* <i>cluster_name</i> *で使用できなくなったことを検出しましたが、フェイルオーバーに使用できる十分なスタンバイノードがありません。
Database in wrong state on <i>node_address</i>	スタンバイエージェントは、ローカルデータベースがリカバリでないことを検出しました。これで、エージェントはIDLEになります。手動による介入が必要です。
Database in wrong state on <i>node_address</i>	プライマリエージェントは、ローカルデータベースがリカバリ中であることを検出しました。これで、エージェントはIDLEになります。手動による介入が必要です。

Subject	Description
Database connection failure for cluster <i>cluster_name</i>	このノードは、実行中のデータベースに接続できません：* node_address *これが修正されるまで、データベースが実行中であるかどうかをこのノードが確認できないため、フェイルオーバーが正しく機能しない場合があります。
Standby custom monitor failure for cluster <i>cluster_name</i>	次のカスタムモニタスクリプトは、スタンバイノードで失敗しました。エージェントは、ローカルデータベースのモニタリングを停止します。スクリプトの場所：script_nameスクリプト出力：script_results
Primary custom monitor failure for cluster <i>cluster_name</i>	次のカスタムモニタスクリプトは、プライマリノードで失敗しました。EFMはスタンバイのプロモートを試みます。スクリプトの場所：script_nameスクリプト出力：script_results
Loopback address set for ping.server.ip	ピング.server.ip プロパティにループバックアドレスが設定されています。この設定はネットワーク分離の検出に干渉する可能性があるため、変更する必要があります。
Load balancer attach script error	ロードバランサー接続スクリプトscript_nameを正常に実行できませんでした。終了値：exit_code結果：script_results
Load balancer detach script error	ロードバランサーデタッチスクリプトscript_nameを正常に実行できませんでした。終了値：exit_code結果：script_results
Pgpool attach node error	フェールオーバーマネージャーはpgpoolノードの接続に失敗しました。終了値：exit_code。結果：script_results
Pgpool detach node error	フェールオーバーマネージャーはpgpoolノードの切断に失敗しました。終了値：exit_code。結果：script_results

Supported Failover and Failure Scenarios

</ div>

フェールオーバーマネージャーは、フェールオーバーが発生する場合と発生しない場合がある障害についてクラスターを監視しフェイルオーバー。

Failover Managerは、非常に限定された限定的なフェイルオーバーシナリオをサポートします。フェイルオーバーが発生する可能性があります。

- プライマリデータベースがクラッシュまたはシャットダウン。
- プライマリデータベースをホスティングしているノードがクラッシュするか、到達不能になった場合。

Failover Managerは、これらの条件の正確性を検証するためにあらゆる試みを行います。プライマリデータベースまたはノードに障害が発生したことをエージェントが確認できない場合、フェールオーバーマネージャーはクラスターでフェイルオーバーアクションを実行しません。

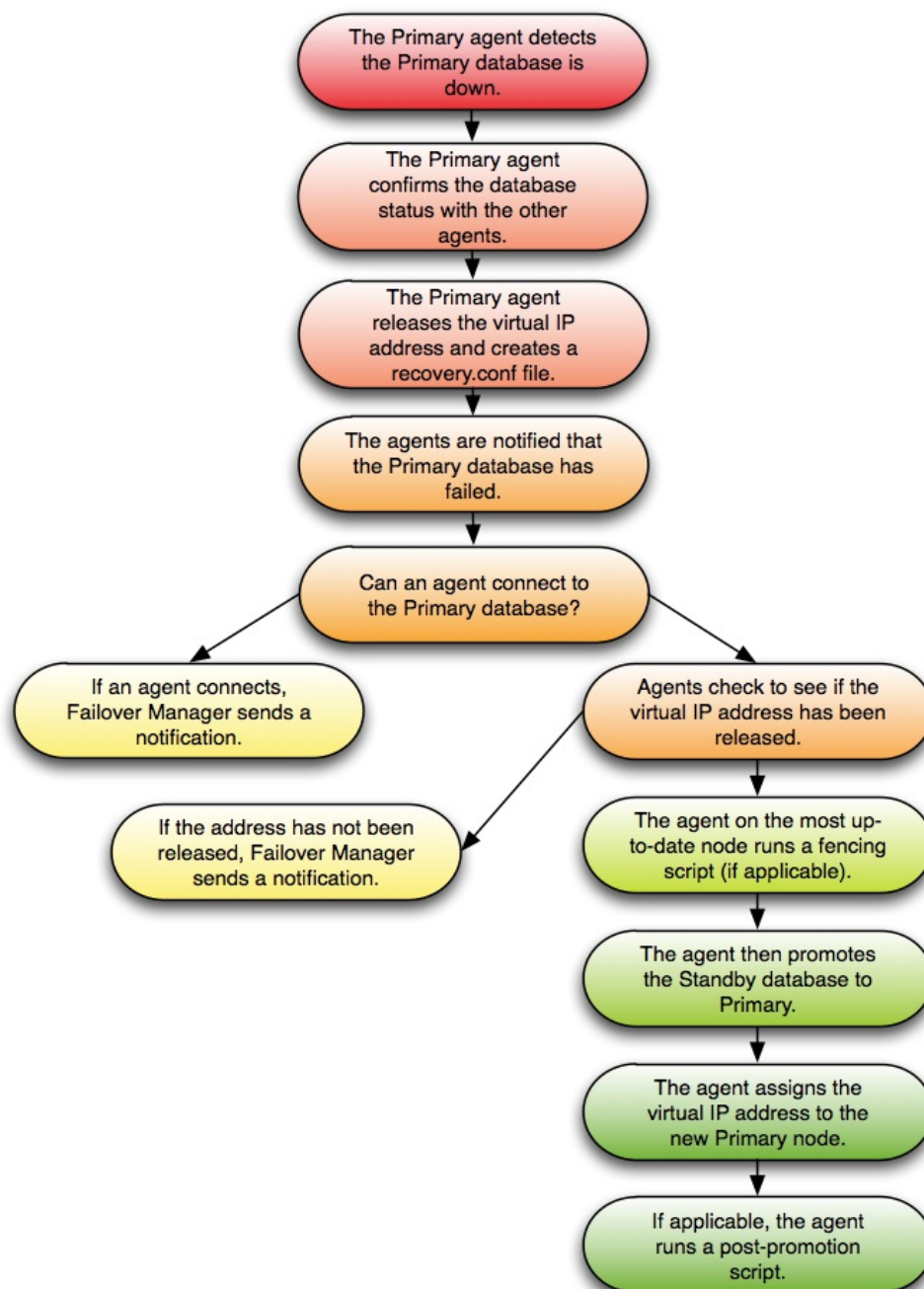
フェールオーバーマネージャーは、フェールオーバーマネージャーでフェイルオーバー条件をモニタおよび検出するが、スタンバイへの自動フェイルオーバーを実行しない場合の* no * * auto -フェイルオーバー*モードもサポートします。このモードでは、フェイルオーバー条件が満たされると管理者に通知が送信されます。自動フェイルオーバーを無効にするには、クラスタープロパティファイルを変更し、[auto.failover](#)パラメータをfalseに設定します。

フェイルオーバーマネージャーは、管理者の介入を必要とする状況に管理者に警告しますが、プライマリにスタンバイ・データベースを推進メリットないこと。

</ div>

プライマリデータベースがダウンしている

プライマリ・データベースのノードで実行されているエージェントは、プライマリ・データベースの障害を検出した場合は、フェールオーバーマネージャーは、障害を確認するプロセスを開始します。



Confirming the Failure of the Primary Database.

プライマリノードのエージェントがプライマリデータベースが発生したことを検出すると、すべてのエージェントがプライマリデータベースに直接接続しようとします。エージェントがデータベースに接続できる場合、フェールオーバーマネージャーはプライマリノードの状態に関する通知を送信します。接続できるエージェントがない場合、プライマリエージェントはデータベース障害を宣言し、VIPを解放します（該当する場合）。

エージェントが仮想IPアドレスまたはデータベースサーバに到達できない場合、フェールオーバーマネージャーはフェイルオーバープロセスを開始します。最新のノード上のスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをプライマリデータベースに昇格させ、仮想IPアドレスをスタンバイノードに割り

当てます。 `auto.reconfigure`が`false`に設定されていない限り、追加のスタンバイノードは新しいプライマリから複製するように構成されます。該当する場合、エージェントはポストプロモーションスクリプトを実行します。

ノードをクラスターに戻す

クラスター全体を再起動せずにこのシナリオから回復するには、次のことを行う必要があります。

1.オリジナルのプライマリノードのデータベースをスタンバイデータベースとして再起動しデータベース。オリジナルのプライマリノードで `efm resume` コマンドを呼び出します。

ノードをプライマリの役割に戻す

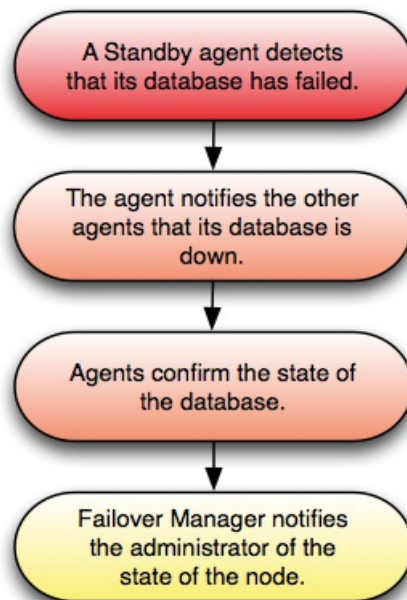
スタンバイとしてクラスターにノードを返した後、あなたは簡単にプライマリのロールにノードを結果することができます。

1.クラスターに複数のスタンバイノードがある場合、 `efm set-priority` コマンドを使用して、ノードのフェイルオーバー優先度を1.2に設定します。 `efm promote -switchover` コマンドを呼び出して、ノードをプライマリノードのオリジナルのロールにプロモートします。

</div>

スタンバイデータベースがダウンしている

スタンバイエージェントがデータベースの障害を検出した場合、エージェントは他のエージェントに通知します。他のエージェントはデータベースの状態を確認します。



Confirming the failure of a Standby Database.

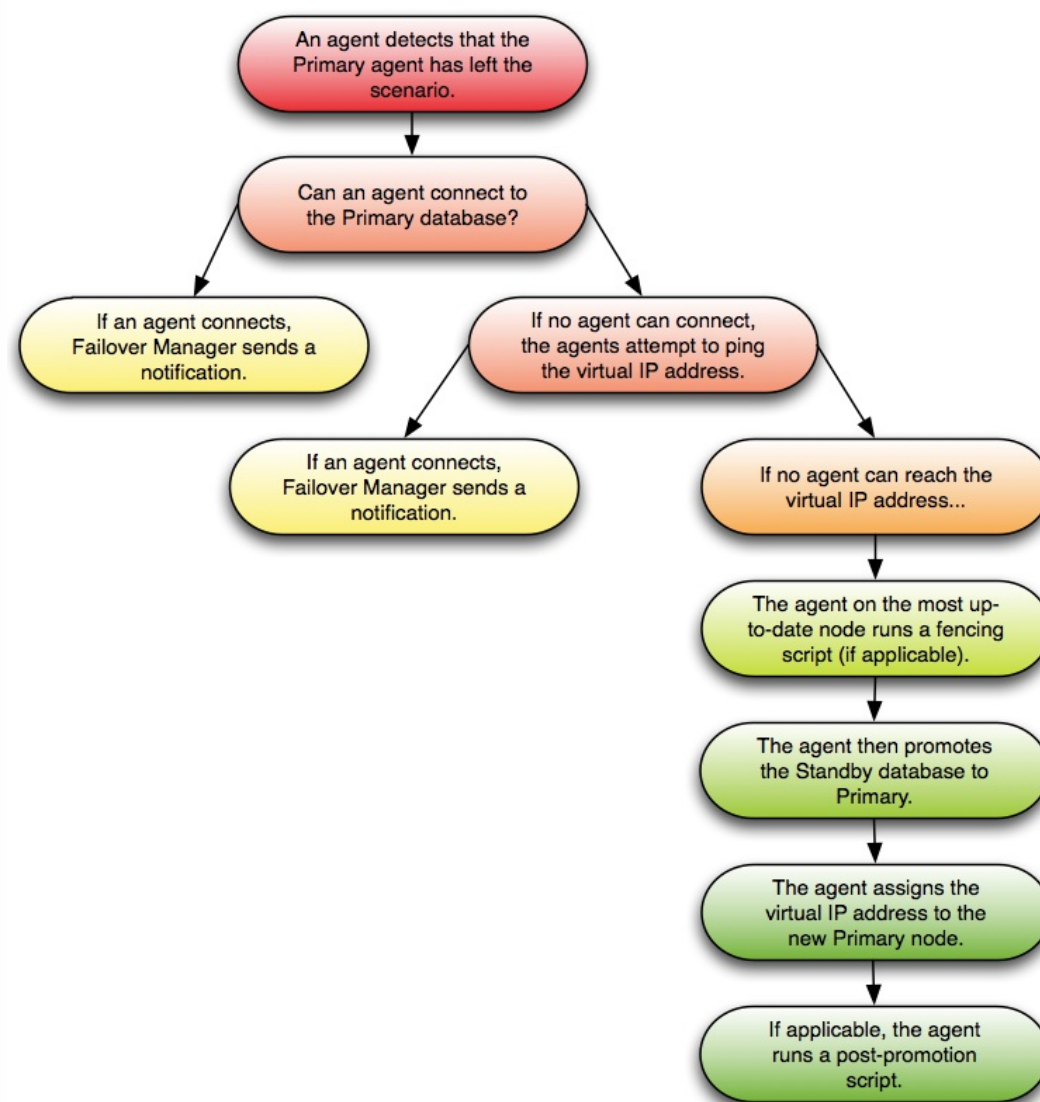
スタンバイデータベースを正常な状態に戻した後、 `efm resume` コマンドを呼び出して、ス

スタンバイをクラスターに結果ます。

</ div>

プライマリエージェントの終了またはノードの失敗

フェールオーバーマネージャーのプライマリエージェントがクラッシュするか、ノードに障害が発生すると、スタンバイエージェントが障害を検出し、（必要に応じて）フェールオーバーを開始します。



Confirming the failure of the Primary Agent.

プライマリエージェントの離脱をエージェントが検出すると、すべてのエージェントがプライマリデータベースに直接接続しようとします。データベースに接続できるエージェントがある場合、エージェントはプライマリエージェントの障害に関する通知を送信します。接続できるエージェントがない場合、エージェントは仮想IPアドレス（該当する場合）に対してピングを試行し、解放されたかどうかを判断します。

エージェントが仮想IPアドレスまたはデータベースサーバに到達できない場合、フェールオーバーマネージャーはフェイルオーバープロセスを開始します。最新ノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをプライマリデータベースに昇格させ、仮想IPアドレスをスタンバイノードに割り当てます。該当する場合、エージェントはプロモーション後スクリプトを実行します。`auto.reconfigure`が`false`に設定されていない限り、追加のスタンバイノードは新しいプライマリから複製するように構成されます。

プライマリがネットワークから分離されたためにこのシナリオが発生した場合、プライマリエージェントは分離を検出して仮想IPアドレスをリリースし、`リカバリ.conf`ファイルを作成します。フェールオーバーマネージャーは、クラスターの残りのノードで前述の手順を実行します。

クラスタ全体を再起動せずにこのシナリオから回復するには、次のことを行う必要があります。

1. オリジナルのプライマリノードを再起動します。オリジナルのプライマリデータベースをスタンバイノードとして起動します。オリジナルのプライマリノードでサービスを開始します。

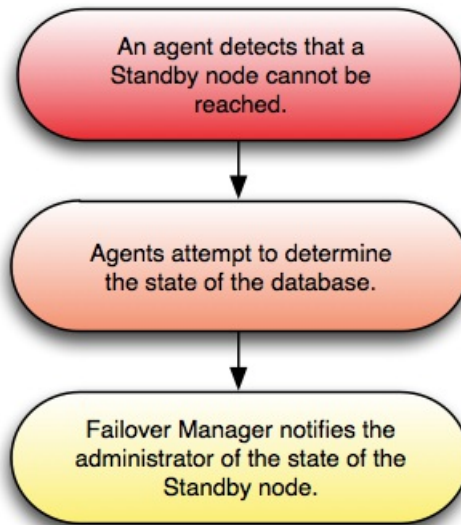
エージェントを停止しても、エージェントに障害が発生したことをクラスターにシグナルしないことにノートしてください。

プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動されるまでフェイルオーバー保護はありません。そのような場合を回避するために、`systemd`を介してプライマリノードをセットアップし、プライマリエージェントが終了したときにフェイルオーバーが発生させることができます。詳細については、セクション [Configuring for Eager Failover](#) を参照してください。

</div>

スタンバイエージェントの終了またはノードの失敗

スタンバイエージェントが終了するか、スタンバイノードに障害が発生すると、他のエージェントは、クラスタに接続されなくなったことを検出します。



Failure of Standby Agent.

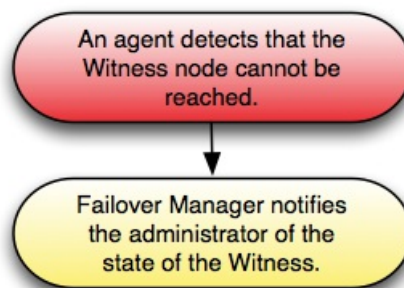
障害が検出されると、エージェントはノードにあるデータベースへの接続を試みます。エージェントが問題があることを確認すると、フェールオーバーマネージャーは適切な通知を管理者に送信します。

プライマリとスタンバイが1つだけ残っている場合、プライマリノードに障害が発生した場合のフェイルオーバー保護はありません。プライマリ・データベースに障害が発生した場合には、プライマリおよびスタンバイエージェントは、データベースが失敗したことに同意することができますし、フェイルオーバーを進めます。

</ div>

専用の監視エージェントの終了/ノードの失敗

次のシナリオでは、専用の証人（データベースをホスティングしていないノード）に障害が発生した場合に実行されるアクションについて詳しく説明します。



Confirming the Failure of a dedicated Witness.

Witnessノードに到達できないことをエージェントが検出すると、Failover ManagerはWitnessの状態を管理者に通知します。

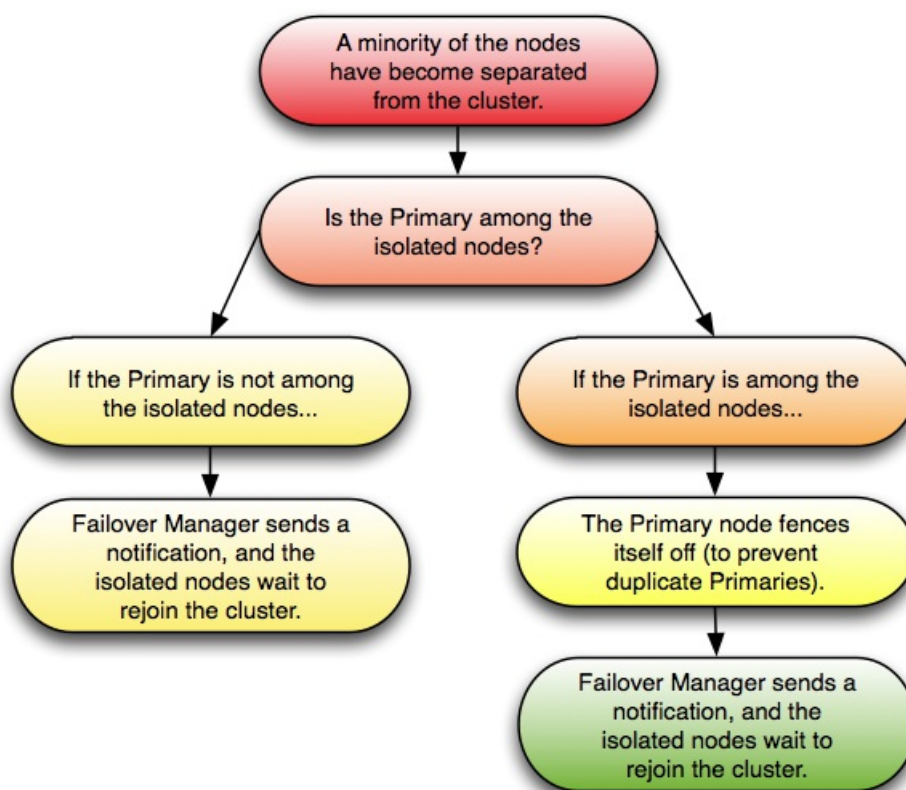
Note * ミラーリング監視に障害が発生し、クラスターに2つのノードしかない場合、ス

スタンバイノードにはプライマリが失敗したか切断されたかを知る方法がないため、フェイルオーバー保護はありません。2ノードクラスターでは、プライマリデータベースに障害が発生してもノードがまだ接続されている場合、スタンバイがプライマリデータベースの状態を確認できるため、フェイルオーバーが発生します。

</div>

ノードがクラスターから分離される

次のシナリオでは、1つ以上のノード（クラスターの少数）がクラスターの大部分から分離された場合に実行されるアクションについて詳しく説明します。



If members of the cluster become isolated.

1つ以上のノード（ただし、クラスターの半分より小さい）がクラスターの残りの部分から分離されると、残りのクラスターは、ノードに障害が発生したかのように動作します。エージェントは、プライマリノードが隔離されたノードに含まれているかどうかを識別しようとします。つまり、プライマリフェンス自分自身がクラスターから隔離され、スタンバイノード（クラスターの過半数内から）がそれを置き換えるために昇格されます。他のスタンバイノードは、`auto.reconfigure` が `false` に設定されていない限り、新しいプライマリから複製するように構成されます。

フェイルオーバーマネージャーは管理者に通知し、分離されたノードは可能な場合はクラスターに再参加します。ノードがクラスターに再参加すると、フェイルオーバーの優先順位が変更される場合があります。

Upgrading an Existing Cluster

</ div>

Failover Managerは、Failover Managerクラスターをアップグレード処理するときに役立つユーティリティを提供します。既存のクラスターをアップグレードするには、以下を行う必要があります。

1. クラスターの各ノードにFailover Manager 4.2をインストールします。Failover Managerのインストールの詳細については、[Installing Failover Manager](#)を参照してください。
2. Failover Managerのインストール後、`efm upgrade-conf` ユーティリティを呼び出して、Failover Manager 4.2の `.properties` および `.nodes` ファイルを作成します。Failover Managerインストーラは、アップグレードユーティリティ () を `/usr/edb/efm-4.2/bin directory` にインストールします。ユーティリティを呼び出すには、ルート権限を引き受けて、コマンドを呼び出します。

```
efm upgrade-conf <cluster_name>
```

`efm upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、Failover Managerで使用するためにパラメータ値を新しい構成ファイルにコピーします。ユーティリティは、構成ファイルの更新されたコピーを `/etc/edb/efm-4.2` ディレクトリに保存します。

3. EFM 4.2の `.properties` および `.nodes` ファイルを変更し、新しい設定を指定します。選択したエディタを使用して、そのノードのサービスを開始する前に、プロパティファイル (`/etc/edb/efm-4.2` ディレクトリ) の追加プロパティを変更します。プロパティ設定の詳細については、[The Cluster Properties File](#)を参照してください。

Note * `db.bin` は必須プロパティです。プロパティファイルを変更ときは、`db.bin` プロパティでPostgres `bin` ディレクトリの場所を指定している保証してください。

4. Eager Failoverを使用している場合、EFMクラスターを停止する前にそれを無効にする必要があります。詳細については、[Disabling Eager Failover](#)を参照してください。
5. バージョン固有のコマンドを使用して、古いフェールオーバーマネージャークラスターを停止します。例、次のコマンドを使用してバージョン4.1クラスターを停止できます。

```
/usr/efm-4.1/bin/efm stop-cluster efm
```

1. クラスターの各ノードで新しい () を開始します。

次の例は、アップグレードユーティリティを呼び出して、フェールオーバーマネージャークラスターのインストール用に `.properties` および `.nodes` ファイルを作成する方法を示しています。

```
[root@k8s-worker ~]# /usr/edb/efm-4.2/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-4.1
Processing efm.properties file

The following properties were added in addition to those in previous installed
priority.standbys
detach.on.agent.failure

Checking directory /etc/edb/efm-4.1
Processing efm.nodes file

Upgrade of files is finished. The owner and group for properties and nodes files
[root@k8s-worker ~]#
```

using a Failover Manager configuration without sudoの場合は、`-source` フラグを含めて、`upgrade-conf` を呼び出すときに構成ファイルが存在するディレクトリの名前を指定します。ディレクトリが構成デフォルトのディレクトリでない場合、アップグレードされたファイルは、`upgrade-conf` コマンドが呼び出されたディレクトリに作成されます。

注意ノート：ユニットファイルを使用している場合は、アップグレードを実行するときに、新しいFailover Managerサービス名前を反映するようにファイルを手動で更新する必要があります。

フェイルオーバーマネージャーのアンインストール

Failover Manager 4.2にアップグレード処理した後、ネイティブパッケージマネージャを使用して、Failover Managerの以前のインストールを削除できます。例、フェイルオーバーマネージャー4.1と不要な依存関係を削除するには、次のコマンドを使用します。

- RHELまたはCentOS 7.xの場合：

```
yum remove edb-efm41
```

- RHELまたはCentOS 8.xの場合：

```
dnf remove edb-efm41
```

- DebianまたはUbuntuの場合：

```
apt-get remove edb-efm41
```

- SLESの場合：

```
zypper remove edb-efm41
```

データベースの更新の実行（マイナーバージョン）

このセクションでは、簡単なマイナーデータベースバージョンアップグレードの実行方法について説明します。あなたは（バージョン10.2.7に10.1.5から、例）別のマイナーバージョンからアップグレードする手順を使用することができ、またはバージョンのためのパッチリリースを適用します。

最初に、フェールオーバーマネージャクラスターの各スタンバイノードでデータベースサーバを更新する必要があります。次に、スイッチオーバーを実行して、フェールオーバーマネージャクラスター内でスタンバイノードをプライマリのロールに昇格させます。次に、古いプライマリノードでデータベースの更新を実行します。

クラスターの各ノードで、次の手順を実行してデータベースサーバを更新する必要があります。

1. Failover Managerエージェントを停止します。データベースサーバを停止します。データベースサーバを更新します。4. データベースサービス。5を開始します。 Failover Managerエージェントを起動します。

Advanced Serverサービスの制御、またはAdvanced Serverのバージョンのアップグレード処理の詳細については、次のWebサイトで入手可能なEDB Postgres Advanced Serverガイドを参照してください。

<https://www.enterprisedb.com/docs>

更新が完了したら、コマンドを使用して古いプライマリをスタンバイリストの先頭に追加し（必要な場合）、スイッチオーバーしてクラスターをオリジナルの状態に結果ができます。

Troubleshooting

</div>

認証ファイルが見つかりません。ローカルエージェントは実行中ですか？

EFMクラスター管理コマンドを起動し、ノードでEFMが実行されていない場合、`efm` コマンドはエラーを表示します。

```
Authorization file not found. Is the local agent running?
```

このコマンドの実行は許可されていません。ユーザー '<os ユーザ>'は `efm\` グループのメンバではありません。

Using the `efm` Utilityに記載されている `efm` コマンドの一部を呼び出すには、特別な特権が必要です。これらのコマンドを実行する権限のないユーザがこれらのコマンドを呼び出した場合、`efm` コマンドはエラーを表示します。

```
Not authorized to run this command. User '<os user>' is not a member of the `
```

通知;予期しないエラーメッセージ

予期しないエラーメッセージメッセージに関する通知メッセージを受け取った場合は、[Failover Manager log file](#)メッセージで `OutOfMemory` メッセージを確認してください。Failover Managerは、このプロパティで設定されたデフォルトのメモリ値で実行されます。

```
# Extra information that will be passed to the JVM when starting the agent.
jvm.options=-Xmx128m
```

割り当てられている128 MBより小さいで実行している場合は、値を増やしてFailover Managerエージェントをリスタートする必要があります。

**** OpenJDKバージョンの確認****

フェイルオーバーマネージャーはOpenJDKでテストされています。OpenJDKを使用することを強くお勧めします。次のコマンドを使用して、Javaインストールのタイプを確認できます。

```
# java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

Configuring Streaming Replication

</div>

レプリケーションシナリオの構成は複雑になる場合があります。構成オプションの詳細については、次のURLで入手可能なPostgreSQLコア文書を参照してください。

< <https://www.postgresql.org/docs/current/ 静的/warm-standby.html#streaming-replication> >

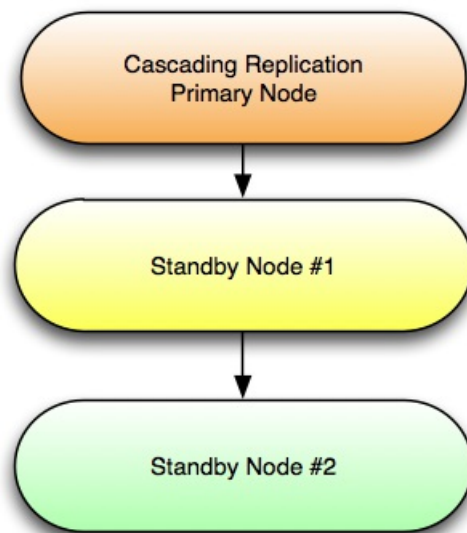
`.pgpass` ファイルを使用して、レプリケーションユーザのmd5認証を有効にすることができます。これは、環境にとって最も安全な認証メソッドである場合とそうでない場合があります。サポートされている認証オプションの詳細については、次のPostgreSQLコア文書を参照してください。

< <https://www.postgresql.org/docs/current/ 静的/client-authentication.html> >

Note * バージョン3.10以降、EFMはスタンバイプロモーションに `pg_ctl` ユーティリティを使用します。スタンバイサーバの昇格のために `trigger_file` または `promote_trigger_file` パラメータを設定する必要はありません。

カスケード複製の限定サポート

フェールオーバーマネージャーはカスケーディングレプリケーションの完全なサポートを提供しませんが、カスケーディングレプリケーションシナリオでのシンプルフェイルオーバーの限定的なサポートを提供します。レプリケーションはスタンバイ・ノードは、プライマリノードへの接続の数を減少させる（及びオーバーヘッド処理）、別のスタンバイ・ノードにストリームすることを可能にします。



Cascading replication.

カスケーディングレプリケーションの構成の詳細については、次のPostgreSQL文書を参照してください。

< <https://www.postgresql.org/docs/current/ 静的/warm-standby.html#cascading-replication> >

カスケーディングレプリケーションシナリオでフェールオーバーマネージャーを使用するには、クラスタープロパティファイルを変更し、スタンバイノード #2で次のプロパティ値を設定する必要があります。

```
promotable=false
auto.reconfigure=false
```

フェイルオーバーが発生しイベント、スタンバイノード #1は、プライマリノードのロールに昇格されます。フェイルオーバーが発生した場合、スタンバイノード #2は、3つのノードを含むようにレプリケーションシナリオを手動で再構成するアクションを実行するまで、新しいプライマリノードの読み取り専用レプリカとして機能し続けます。

スタンバイノード #1に障害が発生したイベント、フェイルオーバー保護は行われませんが、ノードの障害を通知するメールが届きます。

Note * スイッチオーバーを実行してオリジナルのプライマリにスイッチと、カスケーディングレプリケーションシナリオが保持されない場合があります。

Configuring SSL Authentication on a Failover Manager Cluster

</div>

次の手順では、フェールオーバーマネージャーのSSL認証を有効にします。すべての接続クライアントは、クラスター内のデータベースサーバに接続するときにSSL認証を使用する必要がありますことに注意してください。既存のクライアントが現在使用している接続方法を変更する必要があります。

フェールオーバーマネージャークラスターでSSLを有効にするには、以下を行う必要があります。

1. `server.crt` および `server.key` ファイルを `data` ディレクトリ（Advanced Server インストールの下）に配置します。認証権限によって署名された証明書を購入するか、独自の自己署名証明書を作成できます。自己署名証明書の作成については、次の PostgreSQL コア文書を参照してください。

◦ < <https://www.postgresql.org/docs/10/ 静的/ssl-tcp.html#ssl-certificate-creation> >

2. フェールオーバーマネージャークラスター内の各データベースで `postgresql.conf` ファイルを変更し、SSLを有効にします。

```
ssl=on
```

`postgresql.conf` ファイルを変更した後、サーバーをリスタートする必要があります。

1. フェールオーバーマネージャークラスターの各ノードで `pg_hba.conf` ファイルを変更し、ファイルの先頭に次の行を追加しファイル。

```
hostnossl all all all reject
```

この行は、SSL認証を使用していない接続を拒否するようサーバーに指示します。これにより、接続しているクライアントにSSL認証が強制されます。`pg_hba.conf` ファイルの変更については、次の PostgreSQL コア文書を参照してください。>>> <

<https://www.postgresql.org/docs/10/ 静的/auth-pg-hba-conf.html> >

1. `server.crt` とサーバーを配置した後。データディレクトリ内のキーファイル、証明書を Java が理解できるフォームに変換します。次のコマンドを使用できます。

```
openssl x509 -in server.crt -out server.crt.der -outform der
```

詳細については、次をご覧ください：>>> < <https://jdbc.postgresql.org/ 文書/94/ssl-client.html> >

1.次に、証明書をJavaの信頼できる証明書ファイルに追加します。

```
keytool -keystore $JAVA_HOME/lib/security/cacerts -alias <alias_name> -import
```

どこ>>> \$JAVA_HOME は、Javaインストールのホームディレクトリがある>>><;。ALIAS_NAME>任意の文字列ですが、証明書ごとに一意である必要があります>>>>あなたはリストを確認するために **keytool** コマンドを使用することができます。使用可能な証明書のリスト、または特定の証明書に関する情報を取得します。keytool コマンドの使用の詳細については、次のように入力してください。>>>>>>>man keytool>>>>>各データベースサーバからの証明書を、各エージェントの信頼できる証明書ファイルにインポートする必要があります。cacertsファイルの場所は、システムごとに異なる場合があることに注意してください。詳細については、次をご覧ください：>>>><[https://jdbc.postgresql.org/ 文書/94/ssl-client.html](https://jdbc.postgresql.org/文書/94/ssl-client.html)>

1. **jdbc.sslmode** プロパティを設定して、クラスター内の各ノードで**efm.properties file**を変更します。