



EDB Failover Manager

リリース **4.00**

User Guide

2020 年 09 月 13 日

目次

第 1 章	新機能	2
第 2 章	フェイルオーバーマネージャーの概要	3
2.1	前提条件	5
第 3 章	FailoverManager のインストール	8
3.1	RedHat、CentOS、または OEL ホスト	8
3.1.1	設置場所	10
3.2	Debian または Ubuntu ホスト	11
3.3	SLES ホスト	12
第 4 章	フェイルオーバーマネージャーの設定	13
4.1	クラスタープロパティファイル	13
4.1.1	クラスタープロパティの指定	14
4.2	クラスターメンバーファイル	40
4.3	フェイルオーバーマネージャーのパーミッションの拡張	41
4.3.1	sudo なしでフェールオーバーマネージャーを実行する	43
4.4	仮想 IP アドレスでのフェールオーバーマネージャーの使用	45
第 5 章	フェイルオーバーマネージャーの使用	48
5.1	フェールオーバーマネージャークラスターの管理	48
5.1.1	フェールオーバーマネージャークラスターの起動	49
5.1.2	クラスターへのノードの追加	50
5.1.3	スタンバイの優先度の変更	51
5.1.4	フェールオーバーマネージャーノードの昇格	51
5.1.5	フェールオーバーマネージャーエージェントの停止	53
5.1.6	フェールオーバーマネージャークラスターの停止	53
5.1.7	クラスターからノードを削除する	54
5.2	単一ノードでの複数のエージェントの実行	55
5.2.1	RHEL6.x または CentOS6.x	57
5.2.2	RHEL/CentOS7.x または RHEL/CentOS8.x	57

第 6 章	フェールオーバーマネージャークラスターの監視	59
6.1	クラスターステータスレポートの確認	59
6.2	PostgresEnterpriseManager を使用したストリーミングレプリケーションの監視	62
第 7 章	efm ユーティリティの使用	65
第 8 章	FailoverManager サービスの制御	69
8.1	RHEL6.x および CentOS6.x でのサービスユーティリティの使用	70
8.2	RHEL/CentOS7.x および RHEL/CentOS8.x での systemctl ユーティリティの使用	71
第 9 章	ロギングの制御	72
9.1	syslog ログファイルエントリの有効化	74
第 10 章	通知	75
第 11 章	サポートされているフェイルオーバーと障害のシナリオ	85
11.1	プライマリデータベースがダウンしています	86
11.2	スタンバイデータベースがダウンしている	88
11.3	プライマリエージェントの終了またはノードの失敗	89
11.4	スタンバイエージェントの終了またはノードの失敗	91
11.5	専用の監視エージェントの終了/ノードの失敗	92
11.6	ノードがクラスターから分離される	93
第 12 章	既存のクラスターのアップグレード	94
12.1	FailoverManager のアンインストール	96
12.2	データベース更新の実行（マイナーバージョン）	97
第 13 章	トラブルシューティング	98
第 14 章	ストリーミングレプリケーションの設定	100
14.1	カスケード複製の限定サポート	101
第 15 章	フェールオーバーマネージャークラスターでの SSL 認証の構成	103
第 16 章	結論	105
索引		107

EDBFailoverManager™

EDBPostgresFailover Manager (EFM) は、EnterpriseDB の高可用性モジュールであり、プライマリでソフトウェアまたはハードウェアに障害が発生した場合、Postgres プライマリノードがスタンバイノードに自動的にフェールオーバーできるようにします。

このガイドは、Failover Manager 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。バージョン。Failover Manager でサポートされているプラットフォームとバージョンについてはバージョン、次の EnterpriseDBWeb サイトにアクセスします。

<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 EDBPostgresAdvancedServer データベースを意味する Postgres を使用しています。

新機能

EDBPostgres に次の変更が加えられました Failover Manager バージョン 3.10 を作成するには:

- Postgres の物理複製スロット機能のサポート
- NAT アドレスのサポート
- 再起動のためのスタガ待機
- ローカルノードのステータスを確認する新しいコマンドの導入
- スタンバイ昇格のために `trigger_file` を `pg_ctl` ユーティリティに置き換えます

第2

フェイルオーバーマネージャーの概要

EDBPostgresFailover Manager (EFM) クラスターは、ネットワーク上の次のホストにあるフェイルオーバーマネージャー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ーです。
- 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ーです。
- 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ーがない場合は、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*は、複数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 Failover Manager を指します。AFailover Manager 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。

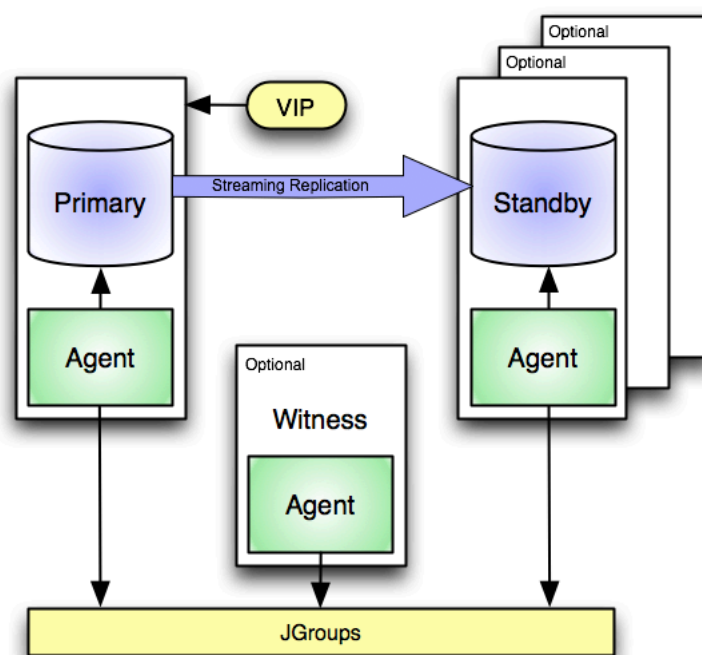


図 1 仮想 IP アドレスを使用する EFM シナリオ

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ます。

- エージェントがデータベースに到達できない場合、エージェントはアイドルモードで起動します。
- データベースが回復中であることがわかると、エージェントはスタンバイの役割を引き継ぎます。
- データベースが回復中でない場合、エージェントはプライマリの役割を引き受けます。

フェイルオーバーの場合、Failover Manager プロモートされたスタンバイがクラスター内の最新のスタンバイであることを確認しようとします。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、データが失われる可能性があることに注意してください。

JGroups<<http://www.jgroups.org/>>は Failover Manager を可能にするテクノロジーを提供します。メンバーノードが互いに通信し、ノード障害を検出できるクラスターを作成します。

上記の図は Failover Manager を示しています。仮想 IP アドレスを使用するクラスター。独自の `<script<cluster_properties>` を提供して、障害が発生した場合にロードバランサーを再構成する場合は、`<using_vip_addresses>` の代わりにロードバランサーを使用できます。

2.1 前提条件

Failover Manager を構成する前にクラスタの場合、以下で説明する前提条件を満たす必要があります。

Java1.8（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。Failover ManagerOpenJDK 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。`Java<<https://openjdk.java.net/install/>>`__のインストール手順はプラットフォーム固有です。

SMTP サーバーを提供する

Failover Manager から通知を受け取ることができますユーザー定義の通知スクリプト、電子メール、またはその両方で指定されたとおり。

- 電子メール通知を使用している場合、SMTP サーバーは Failover Manager の各ノードで実行されている必要がありますシナリオ。
- `script.notification` プロパティに値を指定する場合、`user.email` フィールドを空白のままにできます。SMTP サーバーは必要ありません。

イベントが発生した場合、Failover Manager スクリプト（提供されている場合）を呼び出し、クラスタープロパティファイルの `user.email` パラメーターで指定された任意の電子メールアドレスに通知電子メールを送信します。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/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s1-email-mta.html>`__

ストリーミングレプリケーションの構成

Failover ManagerPostgreSQL ストリーミングレプリケーションをプライマリノードとスタンバイノード間で構成する必要があります。Failover Manager 他のタイプのレプリケーションはサポートしていません。

On database versions 11 (or prior), unless specified with the `-sourcenode` option, a `recovery.conf` file is copied from a random standby node to the stopped primary during switchover. You should ensure that the paths within the `recovery.conf` file on your standby nodes are consistent before performing a switchover. For more information about the `-sourcenode` option, please see *Promoting a Failover Manager Node*.

データベースバージョン 12 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限り）。

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の pg_hba.conf ファイルを変更して、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ーの名前を指定します。

fmdb は efm ユーザーが接続できるデータベースの名前を指定します。

By default, the pg_hba.conf file resides in the data directory, under your Postgres installation. After modifying the pg_hba.conf file, you must reload the configuration file on each node for the changes to take effect. You can use the following command:

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

ここで、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

If a Primary node reboots, Failover Manager may detect the database is down on the Primary node and promote a Standby node to the role of Primary. If this happens, the Failover Manager agent on the (rebooted) Primary node will not get a chance to write the recovery.conf file; the recovery.conf file prevents the database server from starting. If this happens, the rebooted Primary node will return to the cluster as a second Primary node.

これを防ぐには、Failover Manager を開始しますデータベースサーバーを起動する前のエージェント。エージェントはアイドルモードで起動し、クラスター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起動しません。

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（つまり、iptables）がフェールオーバーマネージャーノードのホストで有効になっている場合、Failover Manager 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。クラスター内のプロセス。例:

```
# iptables -I INPUT -p tcp --dport 7800:7810 -j ACCEPT
/sbin/service iptables save
```

上記のコマンドは、狭い範囲のポート（7800～7810）を開きます。Failover Manager クラスタ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ー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ーは、Failover Manager に代わって次の関数を呼び出すのに十分な権限を持っている必要があります。

```
pg_current_wal_lsn()  
  
pg_last_wal_replay_lsn()  
  
pg_wal_replay_resume()  
  
pg_wal_replay_pause()  
  
pg_reload_conf()
```

pg_reload_conf() 特権は、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にのみ必要です。

これらの各関数の詳細については、`PostgreSQL コアドキュメント<<https://www.postgresql.org/docs/10/static/index.html>>`を参照してください。

ユーザーには、構成変数の値を読み取る権限も必要です。データベースのスーパーユーザーは、PostgreSQL の 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 コアドキュメント<<https://www.postgresql.org/docs/current/default-roles.html>>`を参照してください。

FailoverManager のインストール

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 EDBWeb サイトにアクセスしてください。

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

3.1 RedHat、CentOS、または OEL ホスト

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。クラスターの各ノードで手順を実行する必要があります。

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

- RHEL または CentOS7 の場合:

```
yum -y install https://yum.enterprisedb.com/edb-repo-rpms/edb-repo-latest.  
↪noarch.rpm
```

- RHEL または CentOS8 の場合:

```
dnf -y install https://yum.enterprisedb.com/edb-repo-rpms/edb-repo-latest.  
↪noarch.rpm
```

リポジトリ設定ファイルの名前は `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。

2. After creating the `edb.repo` file, use your choice of editor to ensure that the value of the enabled parameter is 1, and replace the username and password placeholders in the `baseurl` specification with the

name and password of a registered EnterpriseDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/redhat/rhel-
↪$releasever-$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

3. 構成ファイルへの変更を保存した後、EPEL リポジトリをインストールします。

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

4. RHEL または CentOS8.x を使用している場合、PowerTools リポジトリを有効にする必要があります。

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

5. これで、次のコマンド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できます。

- RHEL または CentOS7 の場合:

```
yum install edb-efm40
```

- RHEL または CentOS8 の場合:

```
dnf install edb-efm40
```

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、ローカルサーバーにキーをインポートする許可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 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。

Failover Manager ルートで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、インストーラーは、enterprisedb または postgres が所有するクラスターの FailoverManager 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 という名前のユーザーも作成します。

Failover Manager を使用している場合 enterprisedb または postgres 以外のユーザーが所有するクラスターを監視するには、:doc:`ExtendingFailoverManagerPermissions<extending\_efm\_permissions>`を参照してください。

Failover Manager をインストールした後クラスターの各ノードで、以下を行う必要があります。

1. 各ノードで :doc:`クラスタープロパティファイル<cluster\_properties>`を変更します。
2. 各ノードで :doc:`clustermembersfile<cluster\_members>`を変更します。
3. 該当する場合は、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してテストします。

4. Failover Manager を開始しますクラスターの各ノード上のエージェント。Failover Manager の制御の詳細についてはサービス、`''`セクション 5<# controlling-the-failover-manager-service>`__を参照してください。

3.1.1 設置場所

Failover Manager コンポーネントは次の場所にインストールされます。

成分	場所
実行可能ファイル	/usr/edb/efm-4.00/bin
図書館	/usr/edb/efm-4.00/lib
クラスター構成ファイル	/etc/edb/efm-4.00
ログ	/var/log/efm-4.00
ファイルをロックする	/var/lock/efm-4.00
ログローテーションファイル	/etc/logrotate.d/efm-4.00
sudo 設定ファイル	/etc/sudoers.d/efm-40
sudo なしで VIP にアクセスするためのバイナリ	/usr/edb/efm-4.00/bin/secure

.. index:: Debian Installation Ubuntu Installation

3.2 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 EnterpriseDBWeb サイトにアクセスしてください。

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

EDB 内のリンクをたどる Failover Manager 資格情報を要求するテーブル。

次の手順では、EnterpriseDBapt リポジトリを使用して Failover Manager をインストールします。コマンドを使用する場合、***username***および***password***を EnterpriseDB が提供する資格情報に置き換えます。

1. スーパーユーザー特権を想定します。

```
sudo su -
```

2. EnterpriseDBapt リポジトリを構成します。

```
sh -c 'echo "deb https://<username>:<password>@apt.enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
```

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 署名キーを追加します。

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

5. リポジトリのメタデータを更新します。

```
apt-get update
```

6. Failover Manager をインストールします。

```
apt-get -y install edb-efm40
```

3.3 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 Web サイトにアクセスしてください。

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

zypper パッケージマネージャーを使用して Failover Manager をインストールできます。SLES12 ホスト上のエージェント。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. Failover Manager をインストールする前に、スーパーユーザー権限を引き受け、ファイアウォールを停止する必要があります。次に、次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設定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更新し、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、SUSEConnect と SUSEPackageHub 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_12_
↳SP4/Apache:Modules.repo
```

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 をインストールできますエージェント:

```
zypper -n install edb-efm40
```

SUSE ホストの登録の詳細については、次をご覧ください。

<https://www.suse.com/support/kb/doc/?id=7016626>

第 4

フェイルオーバーマネージャーの設定

設定可能 Failover Manager 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*

4.1 クラスタープロパティファイル

Failover Manager の各ノードクラスターには、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* という名前）があります。Failover Manager インストーラーは、`/etc/edb/efm-4.0` ディレクトリに *efm.properties.in* という名前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 を完了した後インストールの場合、ファイルの内容を変更する前にテンプレートの作業用コピーを作成する必要があります。

```
# cp /etc/edb/efm-4.00/efm.properties.in /etc/edb/efm-4.00/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
```

Please note: : By default, Failover Manager expects the cluster properties file to be named *efm.properties*. If you name the properties file something other than *efm.properties*, you must modify the service script or unit file to instruct Failover Manager to use a different name.

クラスタープロパティファイルを作成したら、必要に応じて構成パラメーター値を追加（または変更）します。各プロパティの詳細については、`:ref:`クラスタープロパティの指定<specifying_cluster_properties>`` を参

照してください。

プロパティファイルは root によって所有されています。Failover Manager サービススクリプトは /etc/edb/efm-4.0 directory 内のファイルを見つけることを期待しています。プロパティファイルを別の場所に移動する場合、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。

注意: :プロパティファイルで参照されるすべてのユーザースクリプトは、Failover Manager として呼び出されます。ユーザー。

4.1.1 クラスタプロパティの指定

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 の接続プロパティと動作を指定できます。クラスター。プロパティ設定の変更は、Failover Manager の場合に適用されます開始します。プロパティ値を変更した場合、再起動する必要があります Failover Manager 変更を適用します。

プロパティ値では大文字と小文字が区別されます。Postgres はパラメーター値に引用符付き文字列を使用しますが、Failover Manager プロパティ値に引用符付き文字列を使用できません。たとえば、Postgres 構成パラメーターで IP アドレスを次のように指定できます。

```
listen_addresses='192.168.2.47'
```

Failover Manager 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:エージェントノードで必要
- W:監視ノードで必須
- Y:はい

表 1: クラスターのプロパティ

プロパティ名	A	W	デフォルト値	コメント
<i>db.user</i>	Y	Y		データベースのユーザー名
<i>db.password.encrypted</i>	Y	Y		``efmencrypt``を使用して暗号化されたパスワード
<i>db.port</i>	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
<i>db.database</i>	Y	Y		データベース名
<i>db.service.owner</i>	Y			db.database の\$PGDATA ディレクトリの所有者

次のページに続く

表 1 – 前のページからの続き

プロパティ名	A	W	デフォルト値	コメント
<i>db.service.name</i>				データベースをサービスとして実行する場合は必須
<i>db.bin</i>	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
<i>db.data.dir</i>	Y			クエリ 'showdata_directory;' の出力と同じ
<i>db.config.dir</i>				クエリ 'showconfig_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
<i>jdbc.sslmode</i>	Y	Y	無効にする	:ref:`note<jdbc.note>` を参照してください
<i>user.email</i>				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
<i>from.email</i>			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
<i>notification.level</i>	Y	Y	情報	:ref:`通知のリスト<notifications>` を参照してください
<i>notification.text.prefix</i>				
<i>script.notification</i>				<i>user.email</i> プロパティを使用しない場合は必須。両方のパラメーターを一緒に使用できます
<i>bind.address</i>	Y	Y		例:<ip_address>:<port>
<i>external.address</i>				例:<ip_address/hostname>
<i>admin.port</i>	Y	Y	7809	デフォルトのポートがすでに使用されている場合は変更します
<i>is.witness</i>	Y	Y		説明を見る
<i>local.period</i>	Y		10	
<i>local.timeout</i>	Y		60	
<i>local.timeout.final</i>	Y		10	
<i>remote.timeout</i>	Y	Y	10	
<i>node.timeout</i>	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
<i>encrypt.agent.messages</i>	Y	Y	偽	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
<i>stop.isolated.primary</i>	Y		本当	
<i>stop.failed.primary</i>	Y		本当	
<i>primary.shutdown.as.failure</i>	Y	Y	偽	
<i>update.physical.slots.period</i>	Y		0	
<i>ping.server.ip</i>	Y	Y	8.8.8.8	

次のページに続く

表 1 – 前のページからの続き

プロパティ名	A	W	デフォルト値	コメント
<i>ping.server.command</i>	Y	Y	/bin/ping-q-c3-w5	
<i>auto.allow.hosts</i>	Y	Y	偽	
<i>stable.nodes.file</i>	Y	Y	偽	
<i>db.reuse.connection.count</i>	Y		0	
<i>auto.failover</i>	Y	Y	本当	
<i>auto.reconfigure</i>	Y		本当	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
プロモーション可能	Y		本当	
<i>use.replay.tiebreaker</i>	Y	Y	本当	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
<i>standby.restart.delay</i>			0	
<i>application.name</i>				元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、 <code>primary_conninfo</code> エントリの <code>application_name</code> 部分をこのプロパティ値で置き換えるように設定します。
<i>restore.command</i>				例: <code>restore.command=scp</code> <code><db_service_owner>@% h:</code> <code><archive_path>/% f % p</code>
<i>reconfigure.num.sync</i>	Y		偽	
<i>reconfigure.sync.primary</i>	Y		偽	
<i>minimum.standbys</i>	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
<i>recovery.check.period</i>	Y		2	
<i>restart.connection.timeout</i>			60	
<i>auto.resume.period</i>	Y		0	
<i>virtual.ip</i>			(<code>virtual.ip.single</code> を参照)	VIP を指定しない場合は空白のままにします
<i>virtual.ip.interface</i>				VIP を指定する場合は必須
<i>virtual.ip.prefix</i>				VIP を指定する場合は必須
<i>virtual.ip.single</i>	Y	Y	はい	この値はすべてのノードで同じでなければなりません
<i>check.vip.before.promotion</i>	Y	Y	はい	

次のページに続く

表 1 – 前のページからの続き

プロパティ名	A	W	デフォルト値	コメント
<i>script.load.balancer.attach</i>				例:script.load.balancer.attach= /<パス>/<アタッチスクリプト>% h % t
<i>script.load.balancer.detach</i>				例:script.load.balancer.detach= /<パス>/<デタッチスクリプト>% h % t
<i>script.fence</i>				例:script.fence= /<パス>/<スクリプト名>% p % f
<i>script.post.promotion</i>				例:script.post.promotion= /<パス>/<スクリプト名>% f % p
<i>script.resumed</i>				例:script.resumed= /<パス>/<スクリプト名>
<i>script.db.failure</i>				例:script.db.failure= /<パス>/<スクリプト名>
<i>script.primary.isolated</i>				例:script.primary.isolated= /<パス>/<スクリプト名>
<i>script.remote.pre.promotion</i>				例:script.remote.pre.promotion= /<パス>/<スクリプト名>% p
<i>script.remote.post.promotion</i>				例:script.remote.post.promotion= /<パス>/<スクリプト名>% p

次のページに続く

表 1 – 前のページからの続き

プロパティ名	A	W	デフォルト値	コメント
<i>script.custom.monitor</i>				例:script.custom.monitor= /<パス>/<スクリプト名>
<i>custom.monitor.interval</i>				カスタム監視スクリプトが指定されている 場合は必須
<i>custom.monitor.timeout</i>				カスタム監視スクリプトが指定されている 場合は必須
<i>custom.monitor.safe.mode</i>				カスタム監視スクリプトが指定されている 場合は必須
<i>sudo.command</i>	Y	Y	須藤	
<i>sudo.user.command</i>	Y	Y	sudo-u % u	
<i>lock.dir</i>				指定しない場合、デフォルトは '/var/lock/efm-<version>' になります
<i>log.dir</i>				指定しない場合、デフォルトは ``/var/log/efm-<version>`` になります
<i>syslog.host</i>			ローカルホスト	
<i>syslog.port</i>			514	
<i>syslog.protocol</i>				
<i>syslog.facility</i>			UDP	
<i>file.log.enabled</i>	Y	Y	本当	
<i>syslog.enabled</i>	Y	Y	偽	
<i>jgroups.loglevel</i>			情報	
<i>efm.loglevel</i>			情報	
<i>jvm.options</i>			-Xmx128m	

次のプロパティを使用して、Failover Manager の接続の詳細を指定しますクラスター:

```
# 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=
```

指定された `db.user` には、Failover Manager の代わりに選択された PostgreSQL コマンドを呼び出すための十分な特権が必要です。詳細については、:doc:`前提条件<前提条件>` を参照してください。

データベースユーザーのパスワードの暗号化については、:doc:`データベースパスワードの暗号化<encrypt-`

ing_database_password>`を参照してください。

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ー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ー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=
```

サービスを開始または停止するときに service コマンドまたは 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ます。

```
# 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=
```

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、service、または systemctl) を使用する必要があります。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/as11/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=
```

昇格中にクラスターのプライマリノードで回復ファイルが書き込まれる場所を指定するには、db.data.dir プロパティを使用します。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
```

(次のページに続く)

(前のページからの続き)

```
# 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=
```

データベース設定ファイルが `recovery.conf` または `standby.signal` ファイルと同じディレクトリに保存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設定ファイルの場所を指定します。これは、**AdvancedServer** または **PostgreSQL** インストールの `config_file` パラメーター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** の `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=
```

データベース設定ファイルの詳細については、**PostgreSQL** ウェブサイト<<https://www.postgresql.org/docs/current/runtime-config-file-locations.html>>_をご覧ください。

`jdbc.sslmode` プロパティを使用して **Failover Manager** を指示します **SSL** 接続を使用します。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
```

注釈: `jdbc.sslmode` の値を `verify-ca` に設定し、証明書の検証に **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/static/ssl-tcp.html>

そして

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

`user.email` プロパティを使用して、**FailoverManager** から送信された通知を受信する電子メールアドレス（または複数の電子メールアドレス）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=
```

`from.email` プロパティは、**Failover Manager** からの電子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` プレースホルダー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:`example@%h`）。プレースホルダーは、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、`:doc:Notifications<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=
```

`notification.level` プロパティを使用して、**Failover Manager** の最小の重大度レベルを指定しますユーザー通知を送信するか、通知スクリプトが呼び出されます。通知の完全なリストについては、`:doc:通知<notifications>`を参照してください。

```
# 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
```

`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=
```

`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、メッセージの件名とメッセージ本文が渡されます。スクリプトは毎回呼び出されます **Failover Manager** ユーザー通知を生成します。

```
# 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=
```

`bind.address` プロパティは、**Failover Manager** 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クラスター。

```
# 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=
```

`external.address` プロパティを使用して、NAT 環境内の他のすべての **FailoverManager** 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=
```

`admin.port` プロパティを使用して、**Failover Manager** のポートを指定します管理コマンドをリッスンします。

```
# 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
```

`is.witness` プロパティを `true` 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ます。`is.witness` が `true` 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=
```

The Postgres `pg_is_in_recovery()` function is a boolean function that reports the recovery state of a database. The function returns `true` if the database is in recovery, or `false` if the database is not in recovery. When an agent starts, it connects to the local database and invokes the `pg_is_in_recovery()` function. If the server responds `true`, the agent assumes the role of standby; if the server responds `false`, the agent assumes the role of primary. If there is no local database, the agent will assume an idle state.

注釈: `is.witness` が `true` の場合、Failover Manager リカバリ状態をチェックしません。

次のプロパティは、ローカルサーバー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ー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、現在のノード上のデータベースサーバーへの最後の接続試行後に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い場合、データベースは失敗したと見なされます。

たとえば、これらのプロパティのデフォルト値を指定すると、ローカルデータベースのチェックは 10 秒ごとに 1 回行われます。ローカルデータベースへの接続試行が 60 秒以内に正に戻らない場合、Failover Manager データベースへの接続を最後に試みます。応答が 10 秒以内に受信されない場合、Failover Manager データベース障害を宣言し、`user.email` プロパティにリストされている管理者に通知します。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
```

(次のページに続く)

(前のページからの続き)

```
# 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。

`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、フェー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。

```
# 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
```

`node.timeout` プロパティを使用して、ノードが失敗したかどうかを判断するときにエージェントがノードからの応答を待つ秒数を指定します。`node.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。

```
# 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
```

`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
```

`stop.isolated.primary` プロパティを使用して **Failover Manager** を指示しますプライマリエージェントがデータベースの分離を検出した場合、データベースをシャットダウンします。`true`（デフォルト）の場合、**Failover Manager** `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
```

`stop.failed.primary` プロパティを使用して **Failover Manager** を指示しますプライマリデータベースが

データベースに到達できない場合、プライマリデータベースをシャットダウンしようとしています。true の場合、Failover Manager データベースをシャットダウンしようとした後、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
```

primary.shutdown.as.failure パラメーターを使用して、Failover Manager のシャットダウンを示します。プライマリノードのエージェントは障害として扱われる必要があります。このパラメーターが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）停止すると、クラスターはプライマリノード上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 a failure. This can be set to
# true to treat a primary agent shutdown as a failure situation,
# e.g. during the shutdown of a node, accidental or otherwise.
# 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、ユーザーエラーをキャッチするためのものです。ユーザーがプライマリを停止したように、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 Failover Manager エージェント（たとえば、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注意する必要があります。

primary.shutdown.as.failure がtrue のときにプライマリデータベースで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、efm 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。

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 秒ごとに物理複製スロットの現在の restart_lsn を読み取り、pg_current_wal_lsn でこの情報を送信します。および primary_slot_name (postgresql.conf ファイルで設定されている場合) スタンバイに。物理スロットがまだ存在しない場合、このパラメーター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
```

ping.server.ip プロパティを使用して、サーバーの IP アドレスを指定します Failover Manager 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
```

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
```

auto.allow.hosts プロパティを使用して、許可されたホストリストを更新するために開始された最初のノードの .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 (auto.allow.hosts を true に設定する) と、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
```

`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クラスターから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
```

`db.reuse.connection.count` プロパティにより、管理者は **Failover Manager** の回数を指定できます。同じデータベース接続を再利用して、データベースの状態を確認します。デフォルト値は 0 です。これは **Failover Manager** を示す毎回新しい接続を作成します。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
```

`auto.failover` プロパティは自動フェイルオーバーを有効にします。デフォルトでは、`auto.failover` は `true` に設定されています。

```
# 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
```

`auto.reconfigure` プロパティを使用して **Failover Manager** を指示します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します。プロパティを `true` に設定して自動再構成を有効にする（デフォルト）か、`false` に設定して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。AdvancedServer または PostgreSQL バージョン 11 以前を使用している場合、再構成プロセス中に `recovery.conf` ファイルが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
```

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

Use the `promotable` property to indicate that a node should not be promoted. The `promotable` property is ignored when a primary agent is started. This simplifies switching back to the original primary after a switchover or failover. To override the setting, use the `efm set-priority` command at runtime; for more information about the `efm set-priority` command, see *Using the efm Utility*.

```
# 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
```

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` の値によって **Failover Manager** が決定されます。置換プライマリを選択します。Failover Manager に指示するには、`use.replay.tiebreaker` プロパティを `true` に設定します。ログシーケンス番号で決定されるように、復旧から早くなるノードにフェールオーバーする。ログのシーケンス番号を無視し、ユーザー設定に基づいてノードを昇格させるには、`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
```

`standby.restart.delay` プロパティを使用して、昇格後に新しいプライマリを追跡するためにスタンバイが再構成（停止/開始）され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
```

`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指定できます。

```
# 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
```

(次のページに続く)

(前のページからの続き)

```
# not set, Failover Manager will remove the parameter value
# from primary_conninfo.
application.name=
```

注意: プライマリおよびプロモーション可能なスタンバイに `application.name` プロパティを設定する必要があります。フェールオーバー/スイッチオーバーが発生した場合、プライマリノードが再びスタンバイノードになる可能性があります。

Use the `restore.command` property to instruct Failover Manager to update the `restore_command` when a new primary is promoted. `%h` represents the address of the new primary; Failover Manager will replace `%h` with the address of the new primary. `%f` and `%p` are placeholders used by the server. If the property is left blank, Failover Manager will not update the `restore_command` values on the standbys after a promotion.

``restore_command<https://www.postgresql.org/docs/current/runtime-config-wal.html#RUNTIME-CONFIG-WAL-ARCHIVE-RECOVERY>``の使用の詳細については、PostgreSQL のドキュメント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/%f %p
restore.command=
```

プライマリノードのデータベースパラメータ `synchronous_standby_names` は、データの受信を確認する同期スタンバイサーバーの名前と数を指定して、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ようにします。 `reconfigure.num.sync` プロパティが `true` に設定されている場合、Failover Manager 同期スタンバイサーバーの数を減らし、プライマリノードの構成を再読み込み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.
reconfigure.num.sync=false
```

スタンバイノードの数が必要なレベルを下回った場合、プライマリデータベースの同期レプリケーションモ-

ドを解除するには、`reconfigure.sync.primary` プロパティを `true` に設定します。スタンバイカウントが低下した場合に通知を送信し、同期レプリケーションを中断しないようにするには、`reconfigure.sync.primary` を `false` に設定します。

```
# 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
```

`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。スタンバイカウントが指定された最小値まで低下した場合、プライマリノードに障害が発生してもレプリカノードは昇格されません。

```
# 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
```

`recovery.check.period` プロパティを使用して、Failover Manager の秒数を指定しますデータベースが回復していないかどうかを確認する前に待機します。

```
# Time in seconds between checks to see if a promoting database
# is out of recovery.
recovery.check.period=2
```

`restart.connection.timeout` プロパティを使用して、Failover Manager の秒数を指定します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、新しく再構成されたプライマリまたはスタンバイノードへの接続を試みます。

```
# 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
```

`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監視を再開しようとする秒数（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）を指定します。

```
# 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
```

Failover Manager provides support for clusters that use a virtual IP. If your cluster uses a virtual IP, provide the host name or IP address in the `virtual.ip` property; specify the corresponding prefix in the `virtual.ip.prefix` property. If `virtual.ip` is left blank, virtual IP support is disabled.

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

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。 `virtual.ip.single=true` を指定すると、フェイルオーバーが発生した場合に同じプライマリで新しい VIP アドレスが使用されます。 `false` の値を指定して、クラスターの各ノードに一意の 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、`:doc:仮想 IP アドレスでのフェールオーバーマネージャーの使用<using_vip_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
```

注意: 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。

`check.vip.before.promotion` プロパティを `false` に設定して **Failover Manager** を示す障害が発生した場合に VIP を新しいプライマリに割り当てる前に、VIP が使用中かどうかを確認しません。これにより、複数のノードが同じ VIP アドレスでブロードキャストする可能性があ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、別のプロセスでシャットダウンできない限り、このプロパティを `true` 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
```

次のプロパティを使用して、スイッチオーバーまたはプライマリ障害シナリオの場合にロードバランサーを再構成するスクリプトへのパスを提供します。スクリプトは、スタンバイに失敗した場合にも呼び出されます。これらのプロパティを使用している場合は、クラスターのすべてのノード（プライマリ、スタンバイ、および監視）でそれらを提供して、データベースノードに障害が発生した場合に、別のノードが障害ノードのアドレスで `detach` スクリプトを呼び出すようにします。

`script.load.balancer.attach` プロパティの後にスクリプト名を指定して、ノードをロードバランサーにアタッチする必要があるときに呼び出されるスクリプトを識別します。 `script.load.balancer.detach` プロパティを使用して、ロードバランサーからノードを切り離す必要があるときに呼び出されるスクリプトの名前を指定します。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 アドレスを表す `%h` プレースホルダーを含めます。 **Failover Manager** を指示する `%t` プレースホルダーを含めます文字列に `p`（プライマリノードの場合）または `s`（スタンバイノードの場合）を含める。

```
# 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=
```

`script.fence` は、スタンバイノードから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=
```

`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=
```

`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する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=
```

`script.db.failure` プロパティを使用して、オプションのユーザー提供のスクリプトへの完全なパスを指定します **Failover Manager** 監視するデータベースに障害が発生したことをエージェントが検出した場合に呼び出されます。

```
# 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=
```

`script.primary.isolated` プロパティを使用して、オプションのユーザー提供のスクリプトへの完全なパスを指定します **Failover Manager** プライマリデータベースを監視するエージェントが、プライマリが **Failover Manager** の大部分から隔離されていることを検出した場合に起動しますクラスター。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=
```

`script.remote.pre.promotion` プロパティを使用して、ノードがデータベースをプライマリに昇格させようとしているときに昇格に関与しないエージェント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 プレースホルダーを含めて、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=
```

`script.remote.post.promotion` プロパティを使用して、昇格が発生した後に非プライマリ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 プレースホルダーを含めて、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.
```

(次のページに続く)

(前のページからの続き)

```
#
# 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=
```

`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、**Failover Manager** 通知を送信します。

Failover Manager に指示するには、`custom.monitor.safe.mode` を `true` に設定します。スクリプトからゼロ以外の終了コードを報告しますが、終了コードの結果としてスタンバイをプロモートしません。

```
# 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=
```

`sudo.command` プロパティを使用して **Failover Manager** によって呼び出されるコマンドを指定します。拡張アクセス許可を必要とするタスクを実行するとき。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ます。

`sudo.user.command` プロパティを使用して **Failover Manager** によって呼び出されるコマンドを指定します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。

```
# 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
#   ↳<arguments>
#
# '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
```

`lock.dir` プロパティを使用して **Failover Manager** の代替の場所を指定しますロックファイル。ファイルは **Failover Manager** を防ぎますノード上の単一クラスターに対して複数の（潜在的に孤立した）エージェントを開始することから。

```
# 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=
```

`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。 **Failover Manager** ディレクトリが存在しない場合、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=
```

Failover Manager で UDP または TCP プロトコルを有効にした後ホストでは、syslog へのロギングを有効にできます。プロトコルタイプ (UDP または TCP) を指定するには `syslog.protocol` パラメーターを、syslog ホストのリスナーポートを指定するには `syslog.port` パラメーターを使用します。 `syslog.facility` 値は、エントリ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
```

Use the `file.log.enabled` and `syslog.enabled` properties to specify the type of logging that you wish to implement. Set `file.log.enabled` to `true` to enable logging to a file; enable the UDP protocol or TCP protocol and set `syslog.enabled` to `true` to enable logging to syslog. You can enable logging to both a file and syslog.

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

syslog ロギングの設定の詳細については、[:ref: syslog ログファイルエントリの有効化<enabling_syslog>](#)を参照してください。

`jgroups.loglevel` および `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、[:doc: ControllingLogging<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
```

JVM 関連の設定情報を渡すには、`jvm.options` プロパティを使用します。デフォルト設定では、Failover Manager のメモリ量を指定しますエージェントは使用を許可されます。

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

データベースパスワードの暗号化

Failover Manager 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。efm ユーティリティ<efm_encrypt>（`/usr/edb/efm-4.0/bin` ディレクトリにあります）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、ユーティリティを呼び出すときにコマンドラインでパスワードを渡すか、環境変数 EFMPASS 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 ]
```

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

--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、Failover Manager クラスタープロパティファイルに配置する暗号化された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 回入力するように求められます。ユーティリティが暗号化されたパスワードを共有したら、暗号化されたパスワードをコピーして、クラスタープロパティファイルに貼り付けます。

注意: 多くの Java ベンダーは、完全な暗号化が含まれたバージョンの Java を出荷していますが、輸出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参照する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 JavaCryptographyExtension (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方法を示しています。

```
# efm encrypt acctg
This utility will generate an encrypted password for you to place in
your EFM cluster property file:
/etc/edb/efm-4.00/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
```

注意: 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。

暗号化されたパスワードを受け取ったら、プロパティファイルにパスワードを貼り付けて Failover Manager を開始しますサービス。暗号化されたパスワードに問題がある場合、Failover Manager サービスは開始されません:

```
[witness@localhost ~]# service edb-efm-4.00 start
Starting local edb-efm-4.00 service: [FAILED]
```

Failover Manager の開始時にこのメッセージを受け取った場合サービスの詳細については、スタートアップログ（`/var/log/efm-4.0/startup-efm.log` にあります）を参照してください。

RHEL/CeonOS7.x または RHEL/CentOS8.x を使用している場合、起動情報は次のコマンドでも利用できます。

```
systemctl status edb-efm-4.00
```

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ことを防ぐため、クラスター名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方法を示しています。 `efm encrypt` コマンドを呼び出す前に、EFMPASS の値をパスワード（`lsafepassword`）に設定します。

```
# export EFMPASS=lsafepassword
```

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（0）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。実行が成功すると 0 が返されます。

4.2 クラスターメンバーファイル

Failover Manager の各ノードクラスターには、現在の FailoverManager 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` という名前）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 インストーラーは、`/etc/edb/efm-4.0` ディレクトリに `efm.nodes.in` という名前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 を完了した後インストールの場合、テンプレートの作業コピーを作成する必要があります。

```
# cp /etc/edb/efm-4.00/efm.nodes.in /etc/edb/efm-4.00/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
```

By default, Failover Manager expects the cluster members file to be named `efm.nodes`. If you name the cluster members file something other than `efm.nodes`, you must modify the Failover Manager service script to instruct Failover Manager to use the new name.

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、メンバーシップコーディネーター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:ポート形式、空白で区切られた複数のエントリ。

The Membership Coordinator will update the contents of the `efm.nodes` file to match the current members of the cluster. As agents join or leave the cluster, the `efm.nodes` files on other agents are updated to reflect the current cluster membership. If you invoke the `efm stop-cluster` command, Failover Manager does not modify the file.

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、別のノードが役割を引き受けます。`efmcluster-status<efm_cluster_status>` コマンドを使用して、MembershipCoordinator 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、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が含まれていることを手動で確認する必要があります。

クラスターに参加するノードの IP 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。起動時に、`auto.allow.hosts` プロパティ（クラスタープロパティファイル<`auto.allow.hosts`>内）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（クラスタープロパティファイル<`auto.allow.hosts`>にあります）が `true` に設定されている場合、MembershipCoordinator は `.nodes` ファイルを更新しません。クラスターメンバーがクラスターに参加または離脱するとき。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。

4.3 フェイルオーバーマネージャーのパーミッションの拡張

During the Failover Manager installation, the installer creates a user named `efm`. `efm` does not have sufficient privileges to perform management functions that are normally limited to the database owner or operating system superuser.

- データベースのスーパーユーザー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ー特権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 script` を呼び出します。

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

The sudoers file contains entries that allow the user `efm` to control the Failover Manager service for clusters owned by `postgres` or `enterprisedb`. You can modify a copy of the sudoers file to grant permission to manage Postgres clusters owned by other users to `efm`.

`efm-40` 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 'enterprisedb'. If you run your db service under a
# non-default account, you will need to copy this file to grant
# the proper permissions and specify the account in your efm
# cluster properties file by changing the 'db.service.owner'
# property.

efm ALL=(postgres) NOPASSWD: /usr/edb/efm-4.00/bin/efm_db_functions
efm ALL=(enterprisedb) NOPASSWD: /usr/edb/efm-4.00/bin/efm_db_functions

# Allow user 'efm' to sudo efm_root_functions as 'root' to
# write/delete the PID file, validate the db.service.owner
# property, etc.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-4.00/bin/efm_root_functions
# Allow user 'efm' to sudo efm_address as root for VIP tasks.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-4.00/bin/efm_address
# relax tty requirement for user 'efm'
Defaults:efm !requiretty
```

Failover Manager を使用している場合 `postgres` または `enterprisedb` 以外のユーザーが所有するクラスターを監視するには、`efm-40` ファイルのコピーを作成し、ユーザーが `efm_functions` にアクセスでき

るようにコンテンツを変更しますクラスターを管理するスクリプト。

パーミッションの問題のためにエージェントを起動できない場合、デフォルトの `/etc/sudoers` ファイルの最後に次の行が含まれていることを確認してください。

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

```
#includedir /etc/sudoers.d
```

4.3.1 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 Managersudo 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。構成することを選択した場合 Failover Managersudo アクセスなしで実行するには、root アクセスがまだ必要であることに注意してください:

- Failover Manager をインストールします RPM。
- Failover Manager を実行します設定タスク。

Failover Manager を実行するには sudo を使用しない場合、Failover Manager の代わりに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。ユーザーは、デフォルトのデータベーススーパーユーザー（enterprisedb や postgres など）または別の特権ユーザーである可能性があります。ユーザーを選択した後:

1. 次のコマンドを使用して、ユーザーを efm グループに追加します。

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

これにより、ユーザーは /var/run/efm-4.0 および /var/lock/efm-4.0 に書き込むことができます。

2. クラスター名を再利用する場合は、以前に作成したログファイルをすべて削除します。新しいユーザーは、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-4.00/efm.properties.in <directory/cluster_name>.properties

cp /etc/edb/efm-4.00/efm.nodes.in <directory>/<cluster_name>.nodes
```

次に、クラスタ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ーの名前を指定します。db.service.name プロパティが空白であることも確認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、新しいユーザー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-4.00/bin/runefm.sh start|stop <directory/cluster_name>.properties
```

ここで、<directory/cluster_name.properties> はクラスタプロパティファイルのフルパスを指定します。ユーザーは、デフォルト以外のユーザーがエージェントを制御しているとき、または efm スクリプトを使用しているときは常に、プロパティファイルへのフルパスを指定する必要があることに注意してください。

新しいユーザーが Failover Manager を管理できるようにするにはサービスとして、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 /usr/edb/efm-4.0/bin/secure/ に存在する manage-vip という名前のバイナリを使用して、sudo 特権なしで 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。

- このディレクトリには、ルートと efm グループのユーザーのみがアクセスできます。
- バイナリはルートと efm グループでのみ実行可能です。

セキュリティ上の理由から、`/usr/edb/efm-4.0/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しないことをお勧めします。

Failover Manager の使用の詳細については `sudo` を使用しない場合は、次をご覧ください。

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

4.4 仮想 IP アドレスでのフェールオーバーマネージャーの使用

Failover Manager `efm_address` スクリプトを使用して、仮想 IP アドレスを割り当てたり解放したりします。

注意: 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 ElasticIP 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-4.0/bin/efm_address
```

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解放します。

仮想 IPv4IP アドレスを割り当てるには:

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

仮想 IPv6IP アドレスを割り当てるには:

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

仮想アドレスを解放するには:

```
# 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、`:ref: クラスタープロパティファイル<virtual.ip>` を参照してください。

You must invoke the `efm_address` script as the root user. The `efm` user is created during the installation, and is granted privileges in the `sudoers` file to run the `efm_address` script. For more information about the `sudoers` file, see [Extending Failover Manager Permissions](#).

注意: ノードに VIP アドレス（または `bind.address` 以外のアドレス）が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に接続するとき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

Failover Manager で仮想 IP（VIP）アドレスを使用する場合、フェールオーバーマネージャーを開始する前に VIP 機能を手動でテストすることが重要です。これにより、実際のフェー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。次の手順では、Failover Manager のアクションをテストしますかかります。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
```

注意: :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 ping 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。

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。

4. Use the efm_address del command to release the address on the primary node and confirm the node has been released with ip address:

```
# 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 インターフェイスが表示されないはずです。

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。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 ping 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、Failover Manager を開始する前に、非プライマリノードから VIP を解放します。

注意: :VIP に使用されるネットワークインターフェイスは、Failover Manager に使用されるインターフェイスと同じである必要はありません。エージェントの bind.address 値。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、Failover Manager 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ールオーバーが発生します。

VIP が別のインターフェイスを使用する場合、プライマリエージェントがドロップする前に、クラスタの残りが到達可能な 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、node.timeout プロパティで指定された秒数の間 VIP チェックを再試行して、フェイルオーバーが予想どおりに発生することを確認します。

フェイルオーバーマネージャーの使用

Failover Manager¹ つ以上のスタンバイサーバーを使用したクラスターの監視とフェールオーバーのサポートを提供します。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。

If a primary node reboots, Failover Manager may detect the database is down on the Primary node and promote a Standby node to the role of Primary. If this happens, the Failover Manager agent on the (rebooted) Primary node will not get a chance to write the `recovery.conf` file (for server version 11 or prior) or `standby.signal` file (for server version 12 or later); the rebooted Primary node will return to the cluster as a second Primary node. To prevent this, start the Failover Manager agent before starting the database server. The agent will start in idle mode, and check to see if there is already a primary in the cluster. If there is a primary node, the agent will verify that a `recovery.conf` or `standby.signal` file exists, and the database will not start as a second primary.

5.1 フェールオーバーマネージャークラスターの管理

構成すると、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。

By default, *some of the efm commands* must be invoked by `efm` or an OS superuser; an administrator can selectively permit users to invoke these commands by adding the user to the `efm` group. The commands are:

- `efmallow-node`
- `efmdisallow-node`
- `efm` プロモート
- `efmresume`

- *efmset-priority*
- *efmstop-cluster*
- *efmupgrade-conf*

5.1.1 フェールオーバーマネージャークラスターの起動

Failover Manager のノードを開始できます任意の順序でクラスター化します。

Failover Manager を開始するには RHEL6.x または CentOS6.x 上のクラスター、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
service edb-efm-4.00 start
```

Failover Manager を開始するには RHEL/CentOS7.x または RHEL/CentOS8.x 上のクラスター、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
systemctl start edb-efm-4.00
```

ノードのクラスタープロパティファイルで `is.witness` が `true` に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起動します。

ノードが専用の監視ノードではない場合、Failover Manager ローカルデータベースに接続し、`pg_is_in_recovery()` 関数を呼び出します。サーバーが `false` と応答すると、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、ノードに仮想 IP アドレスを割り当てます（該当する場合）。サーバーが `true` と応答した場合、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起動します。

クラスターに参加した後、Failover Manager 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確認します。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、既存のノードはすべて、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。

注釈: `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` に依存していることを確認してください。

5.1.2 クラスターへのノードの追加

ノードを Failover Manager に追加できますいつでもクラスター。クラスターにノードを追加する場合、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 IP 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と IP アドレスを指定します。

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

`efm allow-node` コマンドの使用または Failover Manager の制御の詳細についてはサービスは、[:ref:`EFM ユーティリティの使用<efm\\_allow\\_node>`](#)を参照してください。

Failover Manager をインストールしますエージェントを作成し、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、[:doc:`クラスタープロパティファイル<cluster\\_properties>`](#)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、`MembershipCoordinator` のエントリを追加します。クラスターメンバーファイルの変更の詳細については、[:doc:`クラスターメンバーファイル<cluster\\_members>`](#)を参照してください。
3. 新しいノードでスーパーユーザー権限を想定し、FailoverManager エージェントを起動します。Failover Manager を開始するには RHEL6.x または CentOS6.x 上のクラスター、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
service edb-efm-4.00 start
```

Failover Manager を開始するには RHEL/CentOS7.x または RHEL/CentOS8.x 上のクラスター、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
systemctl start edb-efm-4.00
```

新しいノードがクラスターに参加すると、Failover Manager `user.email` プロパティで提供される管理者の電子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。

注意: `:PostgreSQL` ストリーミングレプリケーションシナリオでは、現在のノードの便利なスタンバイになるには、ノードがスタンバイである必要があります。

5.1.3 スタンバイの優先度の変更

あなたの Failover Manager クラスターに複数のスタンバイサーバーが含まれる場合、`efm set-priority` コマンドを使用してスタンバイノードのプロモーションの優先度に影響を与えることができます。Failover Manager の既存のメンバーでコマンドを呼び出しますクラスター、およびメンバーの IP アドレスの後に優先順位値を指定します。

たとえば、次のコマンドは Failover Manager を指示します 10.0.1.9 を監視している `acctg` クラスタメンバーがプライマリスタンバイ (1) であること:

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

スタンバイの優先度を 0 に設定して、スタンバイをプロモーション不可に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、プロパティ値 `promotable=false` が上書きされます。

For example, if the properties file on node 10.0.1.10 includes a setting of `promotable=false` and you use `efm set-priority` to set the promotion priority of 10.0.1.10 to be the standby used in the event of a failover, the value designated by the `efm set-priority` command will override the value in the property file:

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

フェイルオーバーの場合、Failover Manager 最初に Postgres ストリーミングレプリケーションから情報を取得して、どのスタンバイノードに最新のデータがあるかを確認し、データ損失の可能性が最も低いノードを昇格させます。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る場合、`use.replay.tiebreaker<use.replay.tiebreaker>` が ``false``。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>
```

注意: ノードがクラスターから分離され、後でクラスターに再参加すると、プロモーションの優先度が変わる場合があります。

5.1.4 フェールオーバーマネージャーノードの昇格

Failover Manager の任意のノードで `efm promote` を呼び出すことができます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手動昇格を開始するクラスター。

手動プロモーションは、データベースクラスターのメンテナンス時間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。手動プロモーションを開始するには、`efm` または OS スーパーユーザーの 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] [-noscripts]
```

どこで:

`<cluster_name>` は Failover Manager の名前ですクラスター。

Include the `-switchover` option to reconfigure the original Primary as a Standby. If you include the

-switchover keyword, the cluster must include a primary node and at least one standby, and the nodes must be in sync.

-sourcename キーワードを含めて、リカバリ設定のプライマリからのコピー元の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。

指示を防ぐために -noscripts キーワードを含めます **Failover Manager** 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:

- サーバーバージョン 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。サーバーバージョン 12 以降では、`primary_conninfo` および `restore_command` パラメーターがコピーされ、メモリーに保存されます。
- 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。
- スタンバイは、プライマリノードを置き換えるために昇格され、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` の詳細がメモリーに保存されます。
- このノードに `application.name` プロパティが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` 情報がメモリーに保存されます。
- サーバーバージョン 12 以降を使用している場合、メモリー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。
- 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監視を再開します。

During a manual promotion, the Primary agent releases the virtual IP address before creating a `recovery.conf` file in the directory specified by the `db.data.dir` property. The `recovery.conf` file is created on all server versions, and is used to prevent the old primary database from starting until the file is removed, preventing the node from starting as a second primary in the cluster.

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` を実行して、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する前に、仮想 IP アドレスが使用中で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、ポストプロモーションスクリプトを実行します（該当する場合）。

このコマンドは、クラスタープロパティファイルの `auto.failover` パラメーター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役割に戻すには、プロモーションリストの最初にノードを配置します。

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

次に、手動プロモーションを実行します。

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

efm ユーティリティの詳細については、`:doc:UsingtheEFMUtility<using_efm_utility>`を参照してください。

5.1.5 フェールオーバーマネージャエージェントの停止

エージェントを停止すると、Failover Manager 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、Failover Manager からアドレスを削除しません。許可されたノードホストリスト。

Failover Manager を停止するには RHEL6.x または CentOS6.x のエージェント、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
service edb-efm-4.00 stop
```

Failover Manager を停止するには RHEL/CentOS7.x または RHEL/CentOS8.x 上のエージェント、スーパーユーザー権限を引き受け、コマンドを呼び出します。

```
systemctl stop edb-efm-4.00
```

Until you invoke the `efm disallow-node` command (removing the node's address of the node from the Allowed node host list), you can use the `service edb-efm-4.0 start` command to restart the node at a later date without first running the `efm allow-node` command again.

`primary.shutdown.as.failure<primary.shutdown.as.failure>` プロパティが ``true`` に設定されていない限り、エージェントを停止しても、エージェントが失敗したことをクラスターに通知しないことに注意してください。

5.1.6 フェールオーバーマネージャクラスターの停止

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

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

このコマンドは*all*Failover Manager を引き起こします終了するエージェント。Failover Manager の終了エージェントは、すべてのフェイルオーバー機能を完全に無効にします。

注意: `:efm stop-cluster` コマンドを呼び出すと、許可されたノードの情報はすべて許可ノードホストリストから失われます。

5.1.7 クラスターからノードを削除する

The `efm disallow-node` command removes the IP address of a node from the Failover Manager Allowed Node host list. Assume the identity of `efm` or the OS superuser on any existing node (that is currently part of the running cluster), and invoke the `efm disallow-node` command, specifying the cluster name and the IP address of the node:

```
efm disallow-node <cluster_name> <ip_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。:ref:`エージェントを停止する <stop\_efm\_agent>` まで、サービスは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、フェー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、:ref:`efmallow-node<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。

5.2 単一ノードでの複数のエージェントの実行

Failover Manager 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することにより、同じホストにある複数のデータベースクラスターを監視できます。ノード。単一のノードで複数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。Failover Manager を構成するには Failover Manager を確保しながら、複数のデータベースクラスターを監視する異なるクラスターのエージェントは互いに干渉しないため、以下を行う必要があります。

1. 各クラスターのメンバーごとに、クラスター内のノードの一意のプロパティセットとロール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。
2. クラスターのメンバーをリストする各クラスターのメンバーごとにクラスターメンバーファイルを作成します。
3. 各クラスターのサービススクリプト（RHEL または CentOS6.x システム）またはユニットファイル（RHEL/CentOS7.x または RHEL/CentOS8.x システム）をカスタマイズして、クラスタープロパティの名前とクラスターメンバーファイル。
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 を監視しています。
- sales のデータは /opt/pgdata2 にあります。そのサーバーはポート 5445 を監視しています。

Failover Manager を実行するにはこれら両方のデータベースクラスターのエージェントは、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 と sales のデータベースクラスターに一致するように acctg.properties と 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内で、db.port パラメーター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 パラメーターは同じ値または一意の値を持つことができます。たとえば、acctg.properties ファイルは以下を指定できます：

```
db.user=efm_user
```

```
db.password.encrypted=7c801b32a05c0c5cb2ad4ffbd5e8f9a
```

```
db.port=5444
```

```
db.database=acctg_db
```

sales.properties ファイル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
```

複数の Failover Manager をセットアップする場合、一部のパラメーターは特別な注意が必要です。同じノード上のクラスタエージェント。同じノードに複数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 パラメーターは、各データベースクラスタに固有の値も指定する必要があります。

以下のパラメーターは、仮想 IP アドレスをノードに割り当てるときに使用されます。あなたの Failover Manager クラスタは仮想 IP アドレスを使用しません。これらのパラメーターは空白のままにします。

```
virtual.ip
```

```
virtual.ip.interface
```

```
virtual.ip.prefix
```

このパラメーター値は、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。RHEL6.x または CentOS6.x を使用している場合は、:ref:`RHEL6.x` または CentOS6.x<rhel_or_centos_6>`を参照してください。RHEL/CentOS7.x または RHEL/CentOS8.x を使用している場合は、:ref:`RHEL/CentOS7.x` または RHEL/CentOS8.x<rhel\_or\_centos\_7>`を参照してください。

注意: :カスタムサービススクリプトまたはユニットファイルを使用している場合、Failover Manager をアップグレードするときに、新しいサービス名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。

5.2.1 RHEL6.x または CentOS6.x

RHEL6.x または CentOS6.x を使用している場合は、クラスターごとに一意の名前で edb-efm-4.0 サービススクリプトを新しいファイルにコピーする必要があります。例:

```
# cp /etc/init.d/edb-efm-4.00 /etc/init.d/efm-acctg  
  
# cp /etc/init.d/edb-efm-4.00 /etc/init.d/efm-sales
```

次に、CLUSTER 変数を編集し、クラスター名を efm から acctg または sales に変更します。

サービススクリプトを作成したら、次を実行します。

```
# chkconfig efm-acctg on  
  
# chkconfig efm-sales on
```

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起動します。たとえば、次のコマンドで acctg エージェントを起動できます。

```
# service efm-acctg start
```

5.2.2 RHEL/CentOS7.x または RHEL/CentOS8.x

RHEL/CentOS7.x または RHEL/CentOS8.x を使用している場合、edb-efm-4.0 ユニットファイルを各クラスターに固有の名前で新しいファイルにコピーする必要があります。たとえば、2つのクラスター（acctg と sales という名前）がある場合、ユニットファイル名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を efm から新しいクラスター名に変更します。たとえば、acctg という名前のクラスターの場合、値は以下を指定します。

```
Environment=CLUSTER=acctg
```

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

```
PIDFile=/var/run/efm-4.00/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起動します。たとえば、次のコマンドで acctg エージェントを起動できます。

```
# 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。

http://fedoraproject.org/wiki/Systemd#How_do_I_customize_a_unit_file.2F_add_a_custom_unit_file.3F

第 6

フェールオーバーマネージャークラスターの監視

You can use either the Failover Manager `efm cluster-status` command or the PEM Client graphical interface to check the current status of a monitored node of a Failover Manager cluster.

6.1 クラスターステータスレポートの確認

`efmcluster-status :ref:` クラスタープロパティファイル`<efm_cluster_status>` コマンドは、Failover Manager のステータスに関する情報を含むレポートを返しますクラスター。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 つのノードが実行されている `edb` という名前のクラスターに関するものです。

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					

(次のページに続く)

(前のページからの続き)

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（プロパティファイル内）に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。

Failover Manager 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 は、どのノードがクラスターに参加することを許可されているか、およびノードの昇格順序を簡単に伝える方法を提供します。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、cluster-status 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の結果です。クエリは、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。

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、アイドル状態の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。

- 終了コード 0 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）。

Failover Manager 指定されたデータベースパスワードを解釈できません。

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。

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

PostgresEnterpriseManager（PEM）を使用してサーバーを監視する場合、StreamingReplicationAnalysis ダッシュボード（PEM グラフィカルインターフェイスの一部）を構成して、StreamingReplication シナリオの一部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。

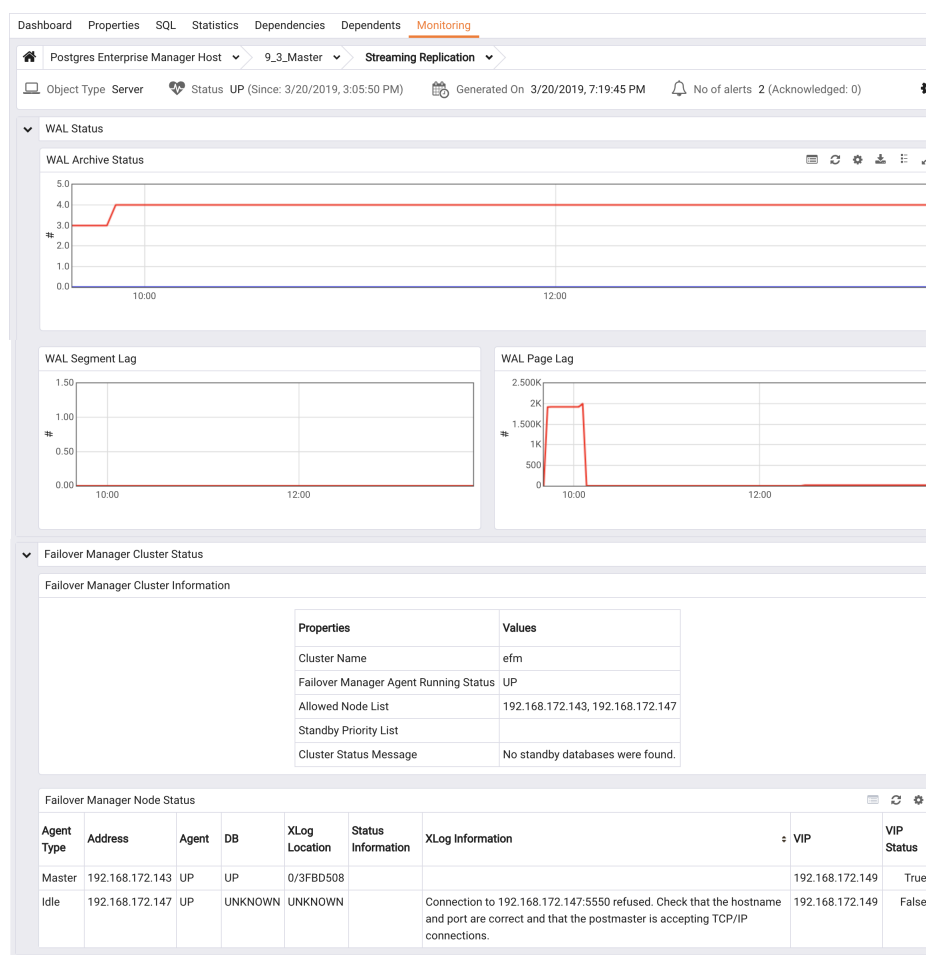


図 1 ストリーミングレプリケーションダッシュボード（プライマリノード）

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ラベルでサーバーのステータスが確認されます。

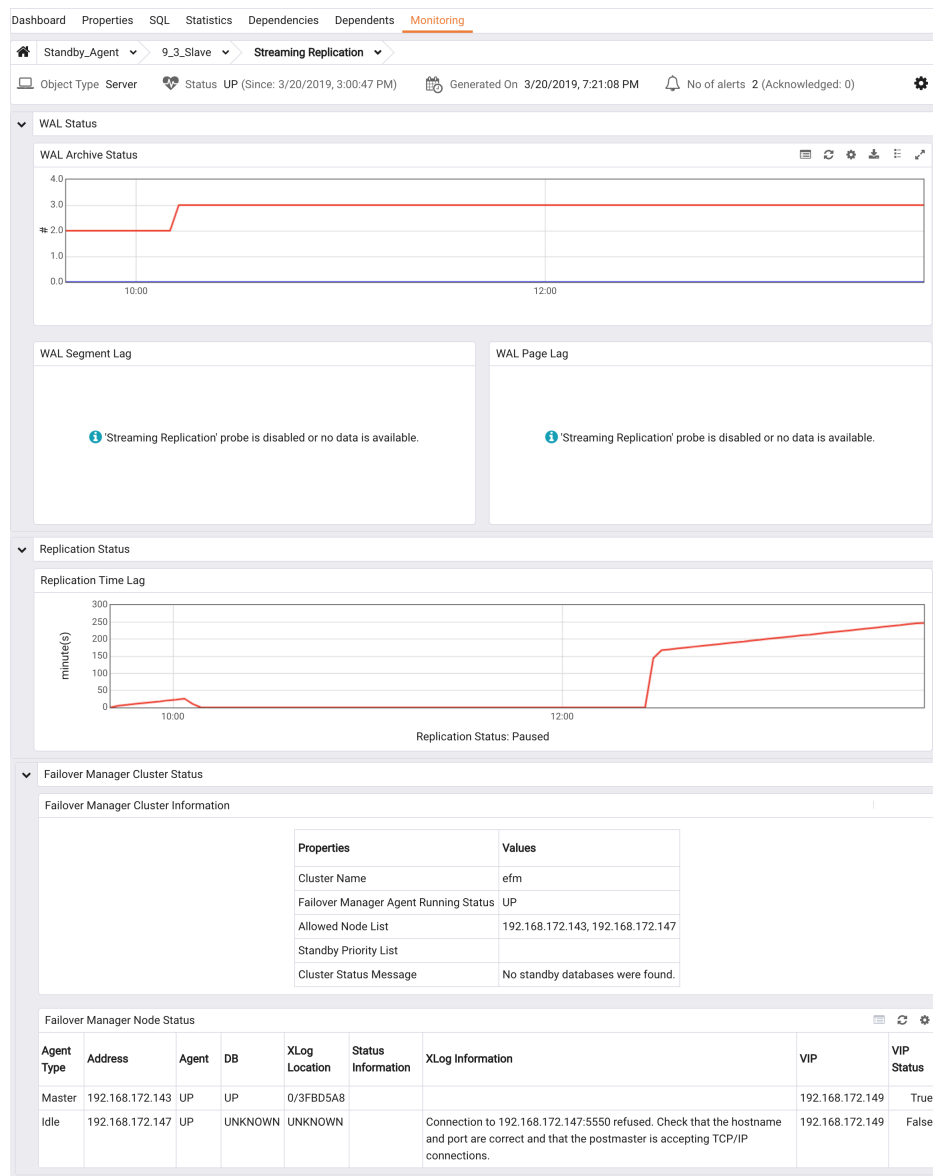


図 2 ストリーミングレプリケーションダッシュボード（スタンバイノード）

デフォルトでは、StreamingReplicationAnalysis 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表示するには、以下のプロ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表示するには、次のプロ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の遅延時間

PEM の詳細については、次の EnterpriseDBWeb 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>

第 7

efm ユーティリティの使用

Failover Manager クラスタ管理を支援する **efm** ユーティリティを提供します。RPM インストーラーは、Failover Manager をインストールするときに、ユーティリティを `/usr/edb/efm-4.0/bin` ディレクトリーに追加します。

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

`efm allow-node` コマンドを呼び出して、指定されたノードがクラスターに参加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。

This command must be invoked by **efm**, a member of the **efm** group, or root.

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

Invoke the `efm disallow-node` command to remove the specified node from the allowed hosts list, and prevent the node from joining a cluster. Provide the name of the cluster and the IP address of the node when calling the `efm disallow-node` command. This command must be invoked by **efm**, a member of the **efm** group, or root.

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

`efm cluster-status` コマンドを呼び出して、Failover Manager のステータスを表示しますクラスター。ステータスレポートの詳細については、`:doc:フェールオーバーマネージャークラスターの監視 <monitoring_efm_cluster>` を参照してください。

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

`efm cluster-status-json` コマンドを呼び出して、Failover Manager のステータスを表示します JSON 形式のクラスター。表示される情報の形式は `efmcluster-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、2 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 <cluster_name> [--from-env]
```

クラスター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。Failover Manager に指示する `--from-env` オプションを含めます EFMPASS 環境変数で指定された値を使用し、ユーザー入力なしで実行します。詳細については、`:doc:` データベースパスワードの暗号化`<encrypting_database_password>`を参照してください。

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

efm promote コマンドは Failover Manager を指示しますスタンバイからプライマリへの手動フェールオーバーを実行します。

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、Failover Manager 続行する前にプロンプトが表示されます。

スタンバイノードを昇格させるために -switchover 句を含め、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、recovery.conf ファイルを古いプライマリノードにコピーする（スタンバイにする）ノードを指定します。スイッチオーバープロセス中の通知を抑制するために、-quiet キーワードを含めます。Failover Manager に指示する -noscripts キーワードを含めますフェンシングまたは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。

This command must be invoked by efm, a member of the efm group, or root.

注意: このコマンドは、クラスタープロパティファイルの auto.failover パラメーター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 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監視を再開します。このコマンドは、efm、efm グループのメンバー、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 <cluster_name> <ip_address> <priority>
```

Invoke the efm set-priority command to assign a failover priority to a standby node. The value specifies the order in which the new node will be used in the event of a failover. This command must be invoked by efm, a member of the efm group, or root.

priorityは 1〜***n***の整数値です。***n***はリスト内のスタンバイノードの数です。値 1 を指定して、新しいノードがプライマリスタンバイであり、フェールオーバーのイベントで昇格される最初のノードであることを示します。優先順位値 0 は、Failover Manager を指示しますスタンバイを昇格させません。

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

停止するために efm stop-cluster コマンドを呼び出します Failover Manager すべてのノードで。このコマンドは Failover Manager を指示します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、フェールオーバー保護は機能しません。

注意:efm stop-cluster コマンドを呼び出すと、すべての許可された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。

This command must be invoked by efm, a member of the efm group, or root.

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

efm upgrade-conf コマンドを呼び出して、既存の Failover Manager から設定ファイルをコピーしますインストール、および Failover Manager に必要なパラメーターの追加インストール。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。

Failover Manager からアップグレードする場合 `sudo` を使用しない設定、`-source` フラグを含め、`upgrade-conf` を呼び出すときに設定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 <cluster_name>`

1 コマンドを呼び出して、`json` 形式でローカルノードのステータスを表示します。このコマンドが正常に実行されると、終了コードとして 1 が返されます。データベース障害またはエージェントのステータスが `IDLE` になった場合、コマンドは終了コードとして 1 を返します。

以下は `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 --help`

`efm --help` コマンドを呼び出して、フェールオーバーマネージャ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。

第 8

FailoverManager サービスの制御

Failover Manager の各ノードクラスターは Failover Manager をホストしますサービススクリプトによって制御されるエージェント。デフォルトでは、サービススクリプトは以下を検出することを想定しています。

- Failover Manager によって使用されるプロパティを含む `efm.properties` という名前の設定ファイルサービス。レプリケーションシナリオの各ノードには、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。
- クラスターメンバーのリストを含む `efm.nodes` という名前のクラスターメンバーファイル。レプリケーションシナリオの各ノードには、クラスターメンバーリストが含まれている必要があります。

単一ノードで複数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 を制御するコマンドサービスはプラットフォーム固有です。

8.1 RHEL6.x および CentOS6.x でのサービスユーティリティの使用

RHEL6.x および CentOS6.x では、Failover Manager（デフォルトで）edb-efm-4.00 という名前の Linux サービスとして実行されます/etc/init.d にあります。Failover Manager によって監視される各データベースクラスター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。

次のサービスコマンドを使用して Failover Manager を制御します RHEL6.x または CentOS6.x ホストに常駐するエージェント:

```
service edb-efm-4.00 start
```

start コマンドは Failover Manager を開始します現在のノードのエージェント。ローカル Failover Manager エージェントはローカルデータベースを監視し、Failover Manager と通信します他のノードで。Failover Manager でノードを開始できます任意の順序でクラスター化します。このコマンドは、ルートによって起動される必要があります。

```
service edb-efm-4.00 stop
```

Failover Manager を停止します現在のノードで。このコマンドは、ルートによって起動される必要があります。

```
service edb-efm-4.00 status
```

status コマンドは Failover Manager のステータスを返します呼び出されるエージェント。任意のノードで status コマンドを呼び出して Failover Manager を指示できます。ステータス情報を返します。例:

```
[witness@localhost ~]# service edb-efm-4.00 status
edb-efm-4.00 (pid 50836) is running...
```

```
service edb-efm-4.00 help
```

Failover Manager のオンラインヘルプを表示しますサービススクリプト。

8.2 RHEL/CentOS7.x および RHEL/CentOS8.x での systemctl ユーティリティの使用

RHEL/CentOS7.x および RHEL/CentOS8.x では、Failover Manager `/usr/lib/systemd/system` にある（デフォルトでは）`edb-efm-4.0.service` という名前の 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 Failover Manager を制御します RHEL/CentOS7.x および RHEL/CentOS8.x ホストに常駐するエージェント:

```
systemctl start edb-efm-4.00
```

`start` コマンドは Failover Manager を開始します現在のノードのエージェント。ローカル Failover Manager エージェントはローカルデータベースを監視し、Failover Manager と通信します他のノードで。Failover Manager でノードを開始できます任意の順序でクラスター化します。このコマンドは、ルートによって起動される必要があります。

```
systemctl stop edb-efm-4.00
```

Failover Manager を停止します現在のノードで。このコマンドは、ルートによって起動される必要があります。

```
systemctl status edb-efm-4.00
```

`status` コマンドは Failover Manager のステータスを返します呼び出されるエージェント。任意のノードで `status` コマンドを呼び出して Failover Manager を指示できます。ステータスとサーバーの起動情報を返します。

```
[root@ONE ~]]> systemctl status edb-efm-4.00
edb-efm-4.00.service - EnterpriseDB Failover Manager 4.00
  Loaded: loaded (/usr/lib/systemd/system/edb-efm-4.00.service; disabled; vendor_
  ↳ preset: 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-4.00/bin/runefm.sh start $
  ↳ {CLUSTER} (code=exited, status=0/SUCCESS)
  Main PID: 58180 (java)
  CGroup: /system.slice/edb-efm-4.00.service
          └─ 58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/
  ↳ jre/bin/java -cp /usr/edb/edb-efm-4.00/lib/EFM-4.00.0.jar -Xmx128m -
  ↳ agentlib:jdpw=transport...
```

ロギングの制御

Failover Manager エージェントごとに 1 つのログファイルを書き込み、エージェントごとに 1 つの起動ログを /var/log/<cluster_name>-4.0 に書き込みます（<cluster_name> はクラスターの名前を指定します）。

クラスタープロパティファイル<loglevel>の ``jgroups.loglevel` および efm.loglevel パラメーターを変更することにより、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
```

ロギング機能は、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（ほとんどのログ出力から順に出力されます）：

- TRACE
- DEBUG
- INFO
- WARN
- ERROR

たとえば、`efm.loglevel` パラメーターを `WARN` に設定すると、フェイルオーバーマネージャーは `WARN` レベル以上のメッセージ（`WARN` および `ERROR`）のみをログに記録します。

デフォルトでは、**Failover Manager** 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-4.0`）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を参照してください。

```
$ man logrotate
```

9.1 syslog ログファイルエントリの有効化

Failover Managersyslog ロギングをサポートします。syslog ロギングを実装するには、syslog を設定して 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、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ている `UDPServerRun` または `TCPServerRun` エントリに、ログエントリの送信先のポート番号が含まれていることを確認する必要があります。たとえば、次の構成ファイルエントリは、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 サービスを再起動して接続を有効にします。

```
systemctl restart rsyslog.service
```

Failover Manager の `rsyslog.conf` ファイルを変更した後ホスト、Failover Manager を変更する必要がありますロギングを有効にするプロパティ。エディターの選択を使用して、プロパティファイル`<log-type.enabled>` (``/etc/edb/efm-4.0/efm.properties.in``) を変更し、実装するログの種類を指定します。

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

また、システムの syslog 詳細`<syslog.logging>`を指定する必要があります。プロトコルタイプ (UDP または TCP) を指定するには`syslog.protocol`` パラメーターを使用し、syslog ホストのリスナーポートを指定するには `syslog.port` パラメーターを使用します。syslog.facility 値は、エントリ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
```

第 10

通知

Failover Manager 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ます。**Failover Manager** を構成した場合電子メール通知を送信するには、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、**Failover Manager** の通知動作を構成します。

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

設定プロパティの編集の詳細については、[:doc: クラスタープロパティ<cluster_properties>](#)の指定を参照してください。

通知の本文には、通知をトリガーしたイベント、およびクラスターの現在の状態に関する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.
```

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

Failover Manager 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、手動の介入を必要としません（たとえば、**Failover Manager** が開始または停止しました）。[:ref: INFO レベル通知のリスト<notifications_info>](#)を参照してください

- 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（たとえば、フェイルオーバーが発生しました）。:ref:`警告レベル通知のリスト<notifications\_warning>`を参照してください
- 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（たとえば、フェールオーバーが試行されたが、完了できなかった）。:ref:`重大レベルの通知のリスト<notifications\_severe>`を参照してください

重大度レベルは、通知の緊急度を指定します。重大度レベルが SEVERE の通知はユーザーの注意をすぐに必要としますが、重大度レベルが INFO の通知はユーザーのアクションを必要としないクラスターに関する操作情報に注意を喚起します。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。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、すべての通知が送信されます。

:ref:`notification.level<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。

注意: :通知を管理用メールアドレスに送信することに加えて、すべての通知はクラスターログファイル（`/var/log/efm-4.0/<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。

件名	説明
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	実行されたリ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	実行されたリ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 *node_address* for cluster *cluster_name*	監視エージェントが実行されています。
Primary agent running on *node_address* for cluster *cluster_name*	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 *node_address* for cluster *cluster_name*	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 *node_address* for cluster *cluster_name*	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 *node_address*	VIP*VIP_address*をノード*node_address*に割り当てる結果:script_results
Releasing VIP from node *node_address*	ノード*node_address*からVIP*VIP_address*を解放する結果:script_results
Starting auto resume check for cluster *cluster_name*	このノードのエージェントは、*auto.resume.period*秒ごとにチェックして、障害のあるデータベースの監視を再開できるかどうかを確認します。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を再開しました*script_name*結果:script_results
WAL logs backed up during promotion	このスタンバイを新しいプライマリに合わせて再設定すると、pg_xlog または pg_wal の内容が*pgdata*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解放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。

件名	説明
Witness agent exited on *node_address* for cluster *cluster_name*	監視エージェントが終了しました。
Primary agent exited on *node_address* for cluster *cluster_name*	データベースの状態は監視されていません。
Cluster *cluster_name* notified that primary node has left	プライマリエージェントが再起動されるまで、クラスターのフェールオーバーは無効になります。
Standby agent exited on *node_address* for cluster *cluster_name*	データベースの状態は監視されていません。
Agent exited during promotion on *node_address* for cluster *cluster_name*	データベースの状態は監視されていません。
Agent exited on *node_address* for cluster *cluster_name*	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 *cluster_name*	エージェントは終了しました。通常、この通知は、起動中にエージェントが終了するときに起動時に生成されます。
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。競合を避けるために、Failover ManagerVIP 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。EDBPostgresFailover ManagerVIP の再取得を試みます。
No standby agent in cluster for cluster *cluster_name*	*cluster_name* のスタンバイがクラスターを離れました。
Standby agent failed for cluster *cluster_name*	*cluster_name* 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 *cluster_name*	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 *cluster_name*	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。

次のページに続く

表 1 – 前のページからの続き

件名	説明
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。
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>*cluster_name*</i> 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一部であるため、フェイルオーバーは試行されません。表示され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分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、ネットワークから分離されなくなりました。
Standby agent tried to promote, but primary DB is still running	スタンバイ EFM エージェントはそれ自体をプロモートしようとしたが、プライマリ DB が <i>*node_address*</i> でまだ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。
Standby agent started to promote, but primary has rejoined.	スタンバイ EFM エージェントはそれ自体の昇格を開始しましたが、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 エージェントはそれ自体をプロモートしようとしたが、プライマリデータベースがまだ <i>*node_address*</i> で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。
Standby agent tried to promote, but VIP appears to still be assigned	スタンバイ EFM エージェントは自身を昇格しようとしたが、仮想 IP アドレス (<i>VIP_address</i>) 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。

次のページに続く

表 1 – 前のページからの続き

件名	説明
Standby agent tried to promote, but appears to be orphaned	スタンバイ EFM エージェントは自身を昇格させようとしたましたが、既知のサーバー (<i>server_address</i>) 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> の潜在的なフェールオーバー状況が検出されました。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、手動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、複数のエージェントが同じノード上の同じクラスターを監視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> の回復ファイルが、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 を再起動しようとする、これは問題になる可能性があります。
<i>recovery_target_timeline</i> is not set to latest in recovery settings	<i>recovery_target_timeline</i> パラメーターは、リカバリー設定で最新に設定されていません。スタンバイサーバーは、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、すでに昇格されているノードがあります: <i>ip_address</i> 。
Standby not reconfigured after failover in cluster <i>*cluster_name*</i>	このノードの <i>auto.reconfigure</i> プロパティは <i>false</i> に設定されています。ノードは、フェイルオーバー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ていません。
Could not resume replay for cluster <i>*cluster_name*</i>	昇格中のスタンバイの再生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: <i>*error_decription*</i> このエラーは、スタンバイの昇格中に再生を呼び出すときにサーバーでエラーが発生した場合に返されます。
Could not resume replay for standby <i>*standby_id*</i> .	スタンバイの再生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: <i>error_message</i> 。

次のページに続く

表 1 – 前のページからの続き

件名	説明
Possible problem with database timeout values	<code>remote.timeout</code> の値 (<i>value</i>) は、 <code>local.timeout</code> の値 (<i>value</i>) よりも大きくなっています。ローカルデータベースが応答するのに時間がかかりすぎる場合、ローカルエージェントは、他のエージェントは接続できてもデータベースに障害が発生したと想定できます。これによりフェイルオーバーは発生しませんが、ローカルエージェントが監視を停止し、フェイルオーバーが保護されない可能性があります。
No standbys available for promotion in cluster <i>*cluster_name*</i>	クラスター内の現在のスタンバイノードの数は、最小数 <i>*number*</i> に低下しています。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 <i>*cluster_name*</i>	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 <i>*cluster_name*</i> 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 <i>*cluster_name*</i>	クラスター内の同期スタンバイノードの数が <i>*number*</i> を下回りました。プライマリの同期スタンバイ名は、 <i>*newsynchronous_standby_namesvalue*</i> に再構成されました。
Synchronous replication has been disabled for cluster <i>*cluster_name*</i> .	クラスター内の同期スタンバイノードの数が <i>*count*</i> 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除外されました。
Could not reload database configuration.	データベース構成を再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: <i>error_message</i> 。
Custom monitor timeout for cluster <i>*cluster_name*</i>	次のカスタム監視スクリプトがタイムアウトしました: <i>script_name</i>
Custom monitor 'safe mode' failure for cluster <i>*cluster_name*</i>	次のカスタムモニタースクリプトは失敗しましたが、 <code>“セーフモード”</code> で実行されています: <i>script_name</i> 。出力: <i>script_results</i>
<i>*primary.shutdown.as.failure*</i> set to true for primary node	このクラスターの <i>*primary.shutdown.as.failure*</i> プロパティーが <code>true</code> に設定されています。クラスター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、クラスターの残りの部分では、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で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 cannot ping 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 <code>IDLE</code> になり、データベースの監視を再開しようとします。

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。

件名	説明
Standby database restarted but EFM cannot connect	データベースの開始または再起動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、 *restart.connection.timeout* 秒まで接続を試行し続けます。
Unable to connect to DB on *node_address*	最大接続制限に達しました。
Unable to connect to DB on *node_address*	db.user= *user_name* のパスワードが無効です。
Unable to connect to DB on *node_address*	許可の指定が無効です。
Primary cannot resume monitoring local database for cluster *cluster_name*	プライマリエージェントは、 *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を解放したり、recovery.conf ファイルを作成したりしません。プライマリエージェントは、再開コマンドを実行してデータベースの監視を再開するまでIDLEのままになります。
Fencing script error	フェンシングスクリプト *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* 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 *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
Remote post-promotion script failed	リモートポストプロモーションスクリプト *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 ノード: node_address
Remote pre-promotion script failed	リモート事前プロモーションスクリプト *script_name* の実行に失敗しました 終了値: exit_code 結果: *script_results ノード: node_address
Post-database failure script error	ポストデータベースエラースクリプト *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
Agent resumed script error	エージェント再開スクリプト *script_name* は正常に実行できませんでした。結果: script_results
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。スタンバイを昇格できませんでした。エラーの詳細: error_details

次のページに続く

表 2 – 前のページからの続き

件名	説明
Error creating recovery.conf file on *node_address* for cluster *cluster_name*	プロモーション中にプライマリノード*node_address*で recovery.conf ファイルを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するには手動での介入が必要です。エラーの詳細: <i>message_details</i>
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: <i>error_details</i>
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、フェールオーバーマネージャークラスターの残りの部分がスタンバイを昇格するときに 2 つのプライマリを防ぐために、プライマリデータベースをフェンスするように*ip_address*のプライマリエージェントに指示しています。
Isolated primary database shutdown.	分離されたプライマリデータベースは、フェー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回避するために自身を分離しています。
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に障害が発生しました。
Agent is timing out for cluster *cluster_name*	このエージェントは、ローカルデータベースにアクセス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 ping でき、監視を再開しました。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ためにノードをチェックする必要があります。
Resume timed out for cluster *cluster_name*	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、監視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。

次のページに続く

表 2 – 前のページからの続き

件名	説明
Internal state mismatch for cluster <i>*cluster_name*</i>	フェールオーバーマネージャー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一致していませんでした。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で確認できます。
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が、フェールオーバーに使用できるスタンバイノードがないことを検出しました。
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回復中ではないことを検出しました。エージェントはアイドル状態になります。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回復中であることを検出しました。エージェントはアイドル状態になります。手動による介入が必要です。
Database connection failure for cluster <i>*cluster_name*</i>	このノードは、 <i>*node_address*</i> で実行されているデータベースに接続できません これが修正されるまで、このノードはデータベースが実行されているかどうかをチェックできないため、フェイルオーバーが適切に機能しない場合があります。
Standby custom monitor failure for cluster <i>*cluster_name*</i>	次のカスタム監視スクリプトは、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監視を停止します。スクリプトの場所: <i>script_name*</i> スクリプト出力: <i>*script_results</i>
Primary custom monitor failure for cluster <i>*cluster_name*</i>	次のカスタム監視スクリプトは、プライマリノードで失敗しました。EFM はスタンバイの昇格を試みます。スクリプトの場所: <i>script_name*</i> スクリプト出力: <i>*script_results</i>
Loopback address set for <i>*ping.server.ip*</i>	ループバックアドレスは、 <i>*ping.server.ip*</i> プロパティに設定されます。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 <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>
Load balancer detach script error	ロードバランサデタッチスクリプト <i>*script_name*</i> が正常に実行されませんでした。終了値: <i>exit_code*</i> 結果: <i>*script_results</i>
Not enough synchronous standbys available in cluster <i>*cluster_name*</i> .	クラスター内の同期スタンバイノードの数が <i>*count*</i> に減少しました。プライマリ上のすべての書き込みクエリは、十分な同期スタンバイノードが追加されるまでブロックされます。

第 11

サポートされているフェイルオーバーと障害のシナリオ

Failover Manager フェールオーバーを引き起こす可能性のある障害のクラスターを監視します。

Failover Manager フェールオーバーシナリオの非常に具体的で限定的なセットをサポートします。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した場合。
- プライマリデータベースをホスト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 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、**Failover Manager** クラスタでフェールオーバーアクションを実行しません。

Failover Manager **Failover Manager** が必要な状況では、`no**auto-*failover*`モードもサポートします。フェイルオーバー条件を監視および検出しますが、スタンバイへの自動フェイルオーバーは実行しません。このモードでは、フェー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。自動フェー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、`.ref: auto.failover<auto.failover>` パラメーターを `false` に設定します。

Failover Manager 管理者の介入を必要とするが、スタンバイデータベースをプライマリに昇格することに値しない状況について管理者に警告します。

11.1 プライマリデータベースがダウンしています

プライマリデータベースノードで実行されているエージェントがプライマリデータベースの障害を検出した場合、Failover Manager 失敗を確認するプロセスを開始します。

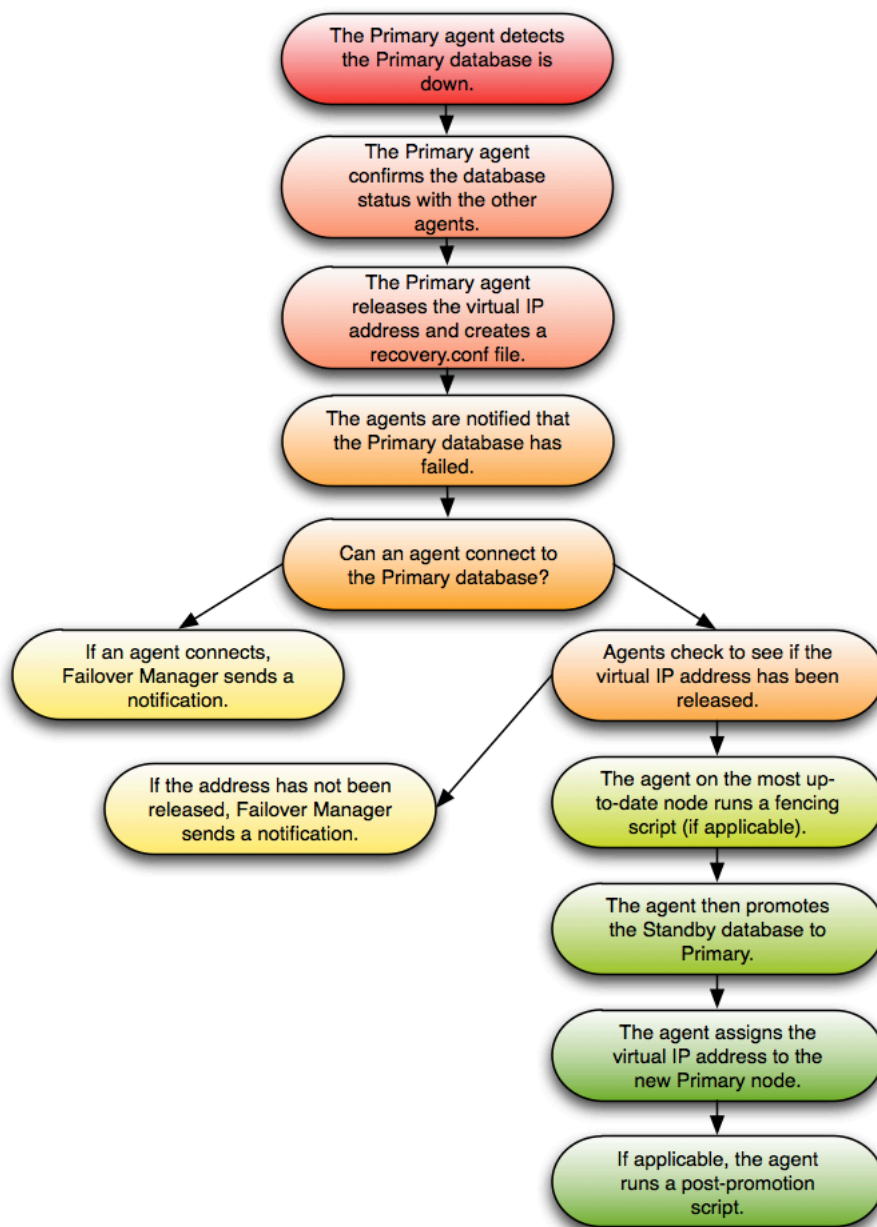


図 1 プライマリデータベースの障害の確認。

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の障害を検出すると、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、Failover Manager 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）。

仮想 IP アドレスまたはデータベースサーバーに到達できるエージェントがない場合、Failover Manager フェイルオーバープロセスを開始します。最新の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。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. 元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ます。
2. 元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

ノードをスタンバイとしてクラスターに戻した後、ノードをプライマリの役割に簡単に戻すことができます。

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 に設定します。
2. `.ref:efmpromote-switchover<efm_promote>` コマンドを呼び出して、ノードをプライマリノードの元の役割に昇格させます。

11.2 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。

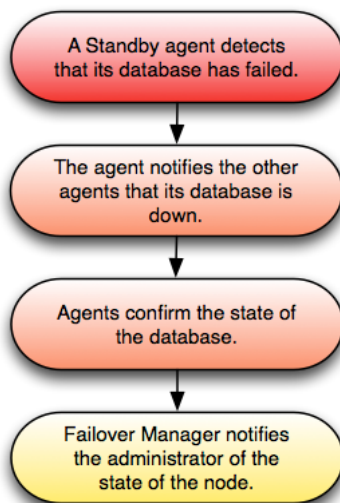


図 2 スタンバイデータベースの障害の確認。

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タンバイをクラスターに戻します。

11.3 プライマリエージェントの終了またはノードの失敗

Failover Manager 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。

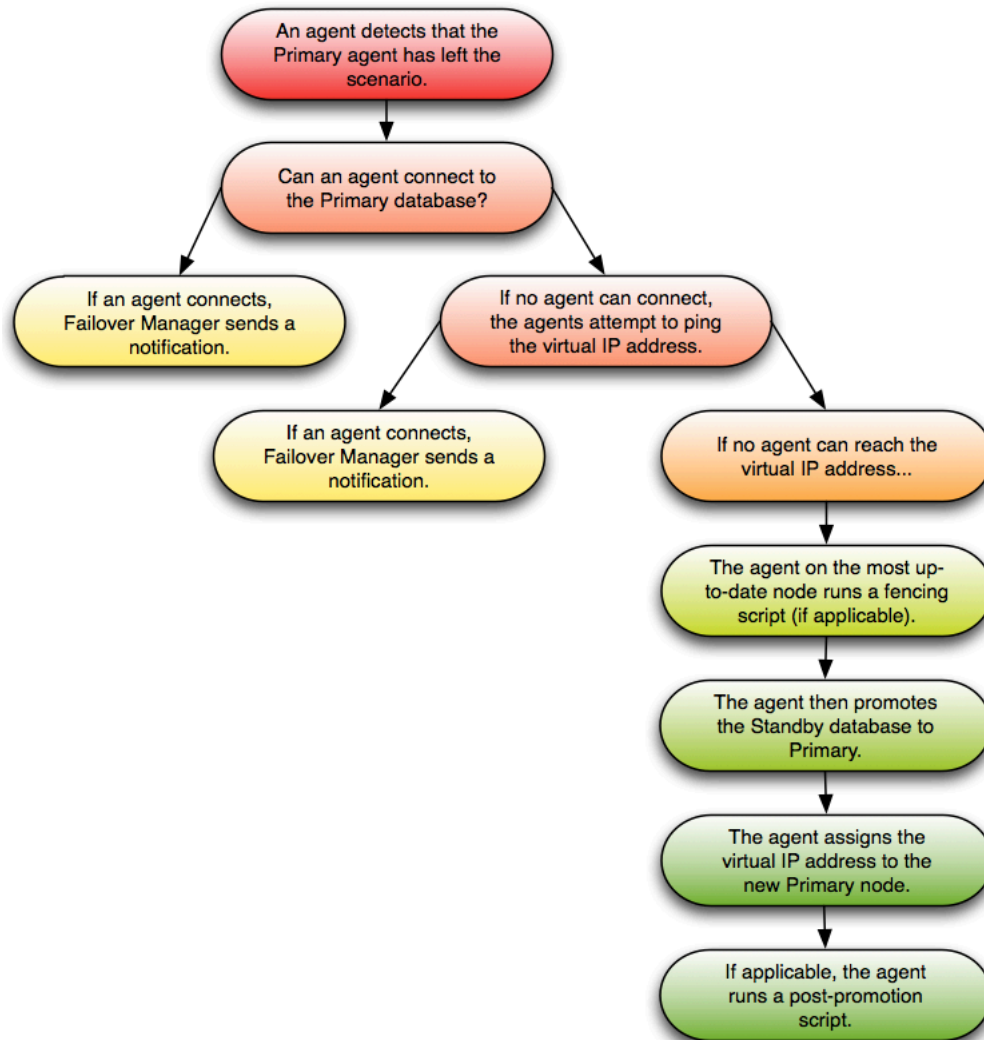


図3 プライマリエージェントの障害の確認。

プライマリエージェントが去ったことをエージェントが検出した場合、すべてのエージェントは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 アドレスを ping して、解放されたかどうかを判断しようとします。

仮想 IP アドレスまたはデータベースサーバーに到達できるエージェントがない場合、Failover Manager 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。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 アドレスを解放し、`recovery.conf` ファイルを作成します。Failover Manager 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. 元のプライマリノードを再起動します。
2. 元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。
3. 元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通知しないことに注意してください。

11.4 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。

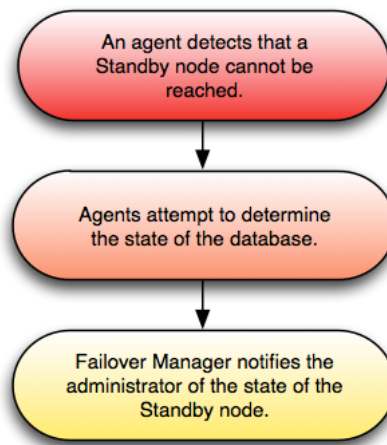


図4 スタンバイエージェントの障害。

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した場合、Failover Manager 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ー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、プライマリエージェントとスタンバイエージェントは、データベースの障害に同意し、フェールオーバーを続行できます。

11.5 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トしていないノード）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。

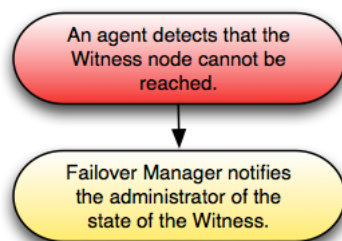


図 5 献身的な証人の失敗の確認

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、FailoverManager は Witness の状態を管理者に通知します。

注:監視ノードに障害が発生し、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、フェー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、フェールオーバーが引き続き発生します。

11.6 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。

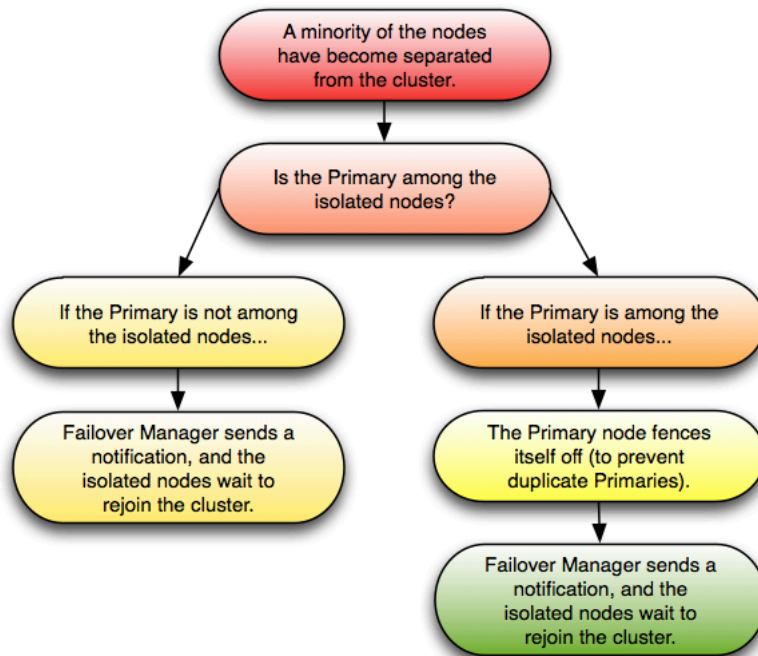


図 6 クラスターのメンバーが分離された場合

1 つ以上のノード（ただし、クラスターの半分未満）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体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。

Failover Manager その後、管理者に通知し、分離されたノードは、可能になると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、フェールオーバーの優先順位が変更される場合があります。

既存のクラスターのアップグレード

Failover Manager Failover Manager をアップグレードするときに役立つユーティリティを提供しますクラスター。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. Failover Manager をインストールします 4.00 クラスターの各ノード。Failover Manager のインストールに関する詳細は、[:doc:InstallingFailoverManager<installing_efm>](#)を参照してください。
2. Failover Manager をインストールした後、efm upgrade-conf ユーティリティを呼び出して、フェイルオーバーマネージャーの .properties および .nodes ファイルを作成します 4.00。Failover Manager インストーラーは、アップグレードユーティリティ (*efmupgrade-conf*) を /usr/edb/efm-4.0/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、root 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、パラメーター値を Failover Manager で使用するための新しい構成ファイルにコピーします。ユーティリティは、設定ファイルの更新されたコピーを /etc/edb/efm-4.0 ディレクトリに保存します。

3. EFM4.00 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。

選択したエディターを使用して、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-4.0 ディレクトリにある）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、[:doc:クラスタープロパティファイル<cluster_properties>](#)を参照してください。

注釈: バージョン 3.6 以降、db.bin は必須プロパティです。プロパティファイルを変更するとき、db.bin プロパティが Postgres の bin ディレクトリの場所を指定していることを確認してください。

- バージョン固有のコマンドを使用して、古い Failover Manager を停止します。たとえば、次のコマンドを使用してバージョン 3.10 クラスターを停止できます。

```
/usr/efm-3.10/bin/efm stop-cluster efm
```

- クラスターの各ノードで新しい FailoverManager サービス<controlling_efm_service> (``edb-efm-4.0``) を開始します。

次の例は、アップグレード変数を呼び出して Failover Manager の .properties および .nodes ファイルを作成する方法を示しています。インストール:

```
[root@localhost efm-4.0]# /usr/edb/efm-4.0/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-3.10
Processing efm.properties file

The following properties were added in addition to those in previous installed_
↳version:
notification.text.prefix
encrypt.agent.messages
standby.restart.delay

The following properties were renamed from those in previous installed version:
stop.failed.master => stop.failed.primary
master.shutdown.as.failure => primary.shutdown.as.failure
script.master.isolated => script.primary.isolated
stop.isolated.master => stop.isolated.primary
reconfigure.sync.master => reconfigure.sync.primary

Checking directory /etc/edb/efm-3.10
Processing efm.nodes file

db.password.encrypted re-encoded with stronger encryption.

Upgrade of files is finished. The owner and group for properties and nodes files_
↳have been set as 'efm'.
[root@localhost efm-4.0]#
```

If you are *using a Failover Manager configuration without sudo*, include the `-source` flag and specify the name of the directory in which the configuration files reside when invoking `upgrade-conf`. If the directory is not the configuration default directory, the upgraded files will be created in the directory from which the `upgrade-conf` command was invoked.

注意: :カスタムサービススクリプトまたはユニットファイルを使用している場合、新しい Failover Manager 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。アップグレードを実行するときのサービス名。

12.1 FailoverManager のアンインストール

Failover Manager にアップグレードした後 4.00、ネイティブパッケージマネージャーを使用して、以前にインストールした Failover Manager を削除できます。たとえば、次のコマンドを使用して Failover Manager を削除します 3.10 および不要な依存関係:

- RHEL または CentOS6.x または 7.x の場合:

```
yum remove edb-efm310
```

- RHEL または CentOS8.x の場合:

```
dnf remove edb-efm310
```

- Debian または Ubuntu の場合:

```
apt-get remove edb-efm310
```

- SLES の場合:

```
zypper remove edb-efm310
```

12.2 データベース更新の実行（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を実行する方法について説明します。以下の手順を使用して、1 つのマイナーバージョンから別のマイナーバージョン（10.1.5 からバージョン 10.2.7 など）にアップグレードしたり、バージョンにパッチリリースを適用したりできます。

最初に Failover Manager の各スタンバイノードでデータベースサーバーを更新する必要があります。クラスター。次に、切り替えを実行し、スタンバイノードを Failover Manager 内でプライマリの役割に昇格させます。クラスター。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ーを更新する必要があります。

1. Failover Manager を停止しますエージェント。
2. データベースサーバーを停止します。
3. データベースサーバーを更新します。
4. データベースサービスを開始します。
5. Failover Manager を開始しますエージェント。

AdvancedServer サービスの制御、または AdvancedServer のバージョンのアップグレードの詳細については、次の場所にある EDBPostgresAdvancedServer ガイドを参照してください。

<https://www.enterprisedb.com/resources/product-documentation>

更新が完了したら、`:ref:efmset-priority<efm_set_priority>` コマンドを使用して、スタンバイリストの先頭に古いプライマリを追加し、スイッチオーバーしてクラスターを元の状態に戻すことができます。

トラブルシューティング

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 '<osuser>' は `efm` グループのメンバーではありません。

You must have special privileges to invoke some of the efm commands documented in *Using the efm Utility*. If these commands are invoked by a user who isn't authorized to run them, the efm command will display an error:

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

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に関する通知メッセージを受け取った場合は、Failover Manager を確認してください。`OutOfMemory` メッセージのログファイル<controlling_logging>。Failover Manager 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
```

128MB 未満の割り当てで実行している場合は、値を増やして Failover Manager を再起動する必要があります。エージェント。

OpenJDK バージョンの確認

Failover ManagerOpenJDK 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)
```

ストリーミングレプリケーションの設定

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。設定オプションの詳細については、次の URL で入手可能な PostgreSQL コアドキュメントを参照してください。

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

レプリケーションユーザーの md5 認証を有効にするために .pgpass ファイルを使用したい場合があります。これは、環境にとって最も安全な認証方法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 コアドキュメントを参照してください。

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

注釈: 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。スタンバイサーバーの昇格のために trigger_file または promote_trigger_file パラメーターを設定する必要はありません。

14.1 カスケード複製の限定サポート

whileFailover Manager カスケードレプリケーションの完全なサポートは提供しませんが、カスケードレプリケーションシナリオでの単純なフェールオーバーの限定的なサポートは提供します。カスケード複製により、スタンバイノードが別のスタンバイノードにストリーミングできるため、プライマリノードへの接続数（および処理オーバーヘッド）が削減されます。

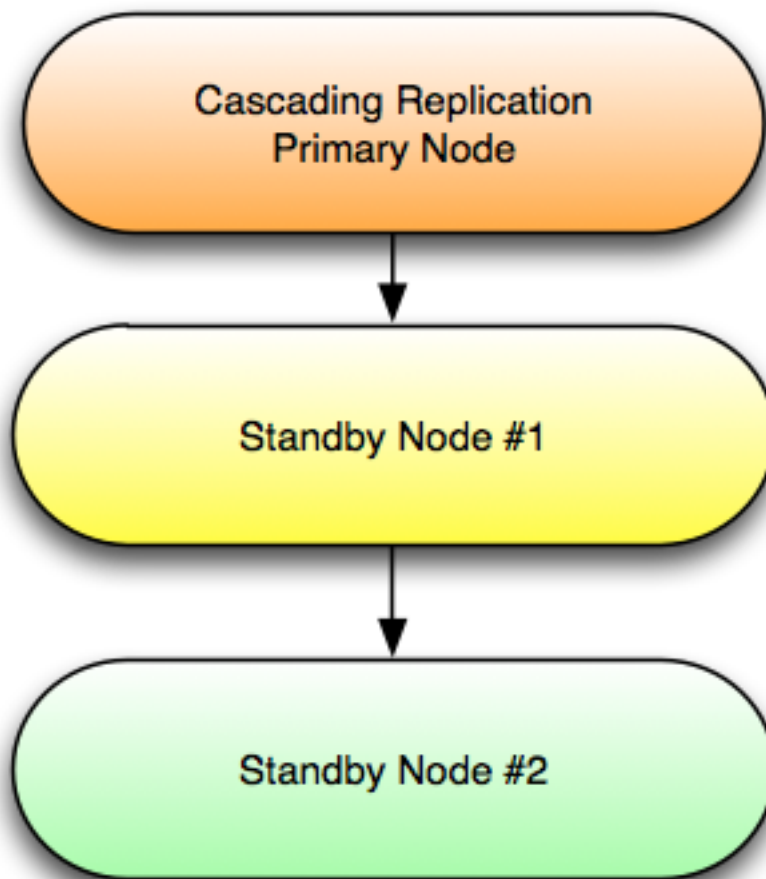


図 1 カスケード複製。

カスケードレプリケーションの構成の詳細については、次の PostgreSQL ドキュメントを参照してください。

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

Failover Manager を使用するにはカスケードレプリケーションシナリオでは、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
```

フェールオーバーが発生すると、スタンバイノード # 1 がプライマリノードの役割に昇格します。フェー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 に障害が発生した場合、フェールオーバー保護はありませんが、ノードの障害を通知する電子メールが届きます。

スイッチオーバーを実行して元のプライマリに切り替えると、カスケード複製シナリオが保持されない場合があることに注意してください。

第 15

フェールオーバーマネージャークラスターでの SSL 認証の構成

次の手順では、Failover Manager の SSL 認証を有効にします。接続するすべてのクライアントは、クラスター内のデータベースサーバーに接続するときに SSL 認証を使用する必要がある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。

Failover Manager で SSL を有効にするにはクラスターでは、以下を行う必要があります。

1. Place a `server.crt` and `server.key` file in the data directory (under your Advanced Server installation). You can purchase a certificate signed by an authority, or create your own self-signed certificate. For information about creating a self-signed certificate, see the PostgreSQL core documentation at:

<https://www.postgresql.org/docs/10/static/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の `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、サーバーを再起動する必要があります。

3. Failover Manager の各ノードで `pg_hba.conf` ファイルを変更します。クラスター、ファイルの先頭に次の行を追加します。

```
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 コアドキュメントを参照してください。

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

4. server.crt および server.key ファイルをデータディレクトリに配置した後、証明書を Java が理解できる形式に変換します。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/documentation/94/ssl-client.html>

5. 次に、証明書を Java 信頼証明書ファイルに追加します。

```
keytool -keystore $JAVA_HOME/lib/security/cacerts -alias <alias_name>  
-import -file server.crt.der
```

どこで

\$JAVA_HOME は Java インストールのホームディレクトリです。

<エイリアス名>には任意の文字列を使用できますが、証明書ごとに一意である必要があります。

keytool コマンドを使用して、利用可能な証明書のリストを確認したり、特定の証明書に関する情報を取得したりできます。keytool コマンドの使用の詳細については、次を入力してください。

```
man keytool
```

各データベースサーバーからの証明書は、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/documentation/94/ssl-client.html>

6. クラスタ内の各ノードで efm.properties ファイル<jdbc.sslmode>`を変更し、`jdbc.sslmode` プロパティを設定します。

第 16

結論

EDBPostgresFailoverManager ガイド

Copyright©2013-2020EnterpriseDBCorporation。

無断複写・転載を禁じます。

EnterpriseDB®Corporation

34CrosbyDrive、Suite201、Bedford、MA01730、アメリカ

T+17813573390F+19784671307E

info@enterprisedb.com

www.enterprisedb.com

- EnterpriseDB および PostgresEnterpriseManager は、EnterpriseDBCorporation の登録商標です。EDB および EDBPostgres は、EnterpriseDBCorporation の商標です。Oracle は、Oracle、Inc の登録商標です。その他の商標は、それぞれの所有者の商標です。
- EDB は、存在する場合、EDBPostgres 製品のログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および入力検証の検証を行います。EDB は追加の入力コンポーネントに対して同じアプローチに従いますが、製品の性質上、限定された検証が可能な信頼できるユーザーによって入力および送信されるフリーフォーム SQL、WMI またはその他の文字列を受け入れる必要がある場合があります。このような場合、ユーザーが誤った入力や危険な入力を禁止することはできません。
- EDB は、将来、認証された信頼できるユーザーからのフリーフォーム SQL、WMI またはその他の潜在的に危険な入力を受け入れる製品に機能を追加する権利を留保しますが、そのような機能はすべて、可能な限り最小のリスクを提供するように設計およびテストされます可能であれば、スーパーユーザーまたは同等の特権が必要です。

- EDB は、すべての潜在的な脅威を予測できることを保証しません。したがって、すべての潜在的な脆弱性が対処または検討されたことをプロセスが完全に保証することはできません。

索引

Adding a Node to a Cluster, 49
 admin.port, 22
 application.name, 28
 auto.allow.hosts, 26
 auto.failover, 27
 auto.reconfigure, 27
 auto.resume.period, 31

 bind.address, 22

 CentOS Installation, 8
 Changing the priority of a Standby, 50
 check.vip.before.promotion, 32
 custom.monitor.interval, 35
 custom.monitor.safe.mode, 35
 custom.monitor.timeout, 35

 db.bin, 19
 db.config.dir, 20
 db.data.dir, 19
 db.database, 18
 db.password.encrypted, 18
 db.port, 18
 db.reuse.connection.count, 27
 db.service.name, 19
 db.service.owner, 19
 db.user, 18

 efm --help, 68
 efm allow-node, 65
 efm cluster-status, 65
 efm disallow-node, 65
 efm encrypt, 66
 efm node-status-json, 68
 efm promote, 66
 efm resume, 67
 efm set-priority, 67
 efm stop-cluster, 67
 efm upgrade-conf, 67
 efm.loglevel, 37
 efm ユーティリティの使用, 65
 enabling_syslog, 73
 encrypt.agent.messages, 24
 external.address, 22

 FailoverManager サービスの制御, 69
 FailoverManager のインストール, 8
 file.log.enabled, 37
 from.email, 21

 is.witness, 23

 jdbc.sslmode, 20
 jgroups.loglevel, 37

 jvm.options, 37

 Linux Installation, 8
 local.period, 23
 local.timeout, 23
 local.timeout.final, 23
 lock.dir, 36
 log.dir, 36

 minimum.standbys, 30
 Multiple agents on a single node, 54

 Node Becomes Isolated, 92
 node.timeout, 24
 notification.level, 21
 notification.text.prefix, 21

 OEL Installation, 8

 ping.server.command, 26
 ping.server.ip, 26
 PostgresEnterpriseManager を使用したストリーミング
 レプリケーションの監視, 62
 Primary Agent Exits, 88
 Primary Database is Down, 85
 Primary Node Fails, 88
 primary.shutdown.as.failure, 25
 promotable, 28
 Promoting a node, 51

 reconfigure.num.sync, 29
 reconfigure.sync.primary, 29
 recovery.check.period, 30
 remote.timeout, 24
 restart.connection.timeout, 30
 restore.command, 29
 RHEL Installation, 8

 script.custom.monitor, 35
 script.db.failure, 33
 script.fence, 32
 script.load.balancer.attach, 32
 script.load.balancer.detach, 32
 script.notification, 22
 script.post.promotion, 33
 script.primary.isolated, 34
 script.remote.post.promotion, 34
 script.remote.pre.promotion, 34
 script.resumed, 33
 SLES Installation, 12
 stable.nodes.file, 27
 Standby Agent Exits, 90
 Standby Database is Down, 87
 Standby Node Fails, 90

`standby.restart.delay`, 28
Starting the EFM Cluster, 49
`stop.failed.primary`, 24
`stop.isolated.primary`, 24
Stopping a Cluster, 53
Stopping an agent, 53
`sudo.command`, 35
`sudo.user.command`, 35
`syslog.enabled`, 37
`syslog.facility`, 36
`syslog.host`, 36
`syslog.port`, 36
`syslog.protocol`, 36

`update.physical.slots.period`:, 25
`use.replay.tiebreaker`, 28
`user.email`, 21
Using the service utility, 69
Using the systemctl utility, 70

`virtual.ip`, 31
`virtual.ip.interface`, 31
`virtual.ip.prefix`, 31
`virtual.ip.single`, 31

Witness Agent Exits, 91
Witness Node Fails, 91

クラスタープロパティファイル, 13
クラスターメンバーファイル, 40

サポートされているフェイルオーバーと障害のシナリオ, 85

ストリーミングレプリケーションの設定, 100

データベースパスワードの暗号化, 38

トラブルシューティング, 98

フェイルオーバーマネージャーのパーミッションの拡張, 41
フェイルオーバーマネージャーの概要, 3
フェイルオーバーマネージャーの使用, 48
フェイルオーバーマネージャーの設定, 13
フェイルオーバーマネージャークラスターでの SSL 認証の構成, 103
フェイルオーバーマネージャークラスターの監視, 59

ロギングの制御, 72
仮想 IP アドレスでのフェイルオーバーマネージャーの使用, 45

既存のクラスターのアップグレード, 94
結論, 105
新機能, 2
前提条件, 5
通知, 75