



EDB Failover Manager

リリース **3.8**

User's Guide

2020 年 01 月 29 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	サポートされているプラットフォーム	5
2.2	前提条件	6
2.3	チュートリアル-シンプルなフェイルオーバーマネージャークラスターの設定	9
第 3 章	Failover Manager のインストール	13
3.1	RedHat、CentOS、または OEL ホストへのフェイルオーバーマネージャーのインストール	13
3.1.1	設置場所	14
3.2	Debian または Ubuntu ホストへのフェイルオーバーマネージャーのインストール	16
3.3	SLES ホストへのフェイルオーバーマネージャーのインストール	17
第 4 章	フェイルオーバーマネージャーの設定	18
4.1	クラスタープロパティファイル	18
4.1.1	クラスタープロパティの指定	19
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 なしでフェイルオーバーマネージャーを実行する	43
4.4	仮想 IP アドレスでのフェイ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	フェイルオーバーマネージャークラスターの管理	48
5.1.1	フェイ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	49
5.1.3	スタンバイの優先度の変更	50
5.1.4	フェイルオーバーマネージャーノードの昇格	51
5.1.5	フェイ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	53
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	RHEL 6.x または CentOS 6.x	57
5.2.2	RHEL 7.x または CentOS 7.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	Postgres Enterprise Manager を使用したストリーミングレプリケーションの監視	62
第 7 章	efm ユーティリティの使用	65
第 8 章	Failover Manager サービスの制御	69
8.1	RHEL 6.x および CentOS 6.x でのサービスユーティリティの使用	70
8.2	RHEL 7.x および CentOS 7.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	76
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	87
11.1	マスターデータベースがダウンしています	88
11.2	スタンバイデータベースがダウンしている	90
11.3	マスターエージェントの終了またはノードの失敗	91
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	93
11.5	専用の監視エージェントの終了/ノードの失敗	94
11.6	ノードがクラスターから分離される	95
第 12 章	既存のクラスターのアップグレード	96
12.1	Failover Manager のアンインストール	98
12.2	データベース更新の実行 (マイナーバージョン)	99
第 13 章	トラブルシューティング	100
第 14 章	ストリーミングレプリケーションの設定	101
14.1	カスケード複製の限定サポート	102
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	104
第 16 章	結論	106
索引		109

EDB Failover Manager™

EDB Postgres Failover Manager (EFM) は、MasterDB の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に Postgres マスター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する EnterpriseDB の高可用性モジュールです。

このガイドでは、Failover Manager 3.8 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。

このドキュメントでは、PostgreSQL または EDB Postgres Advanced Server データベースを意味する Postgres を使用しています。EDB Postgres 製品の詳細については、“EnterpriseDB Web サイト”をご覧ください。 <<https://www.enterprisedb.com/>>’_。

新機能

バージョン 3.8 を作成するために、EDB Postgres Failover Manager に次の変更が加えられました。

- *restore.command* を使用できます 新しいマスターが昇格された場合に *restore_command* パラメータを更新するプロパティ。
- *reconfigure.sync.master* を使用できます 同期スタンバイの数が必要なレベルを下回った場合のフェールオーバーマネージャーの動作を指定するプロパティ。
- 新しい 通知 同期複製マスターの状態が変化した場合、マスターノードにリカバリファイルが存在する場合、またはトリガーファイルパスが書き込み可能でない場合に通知します。

第 2

フェイルオーバーマネージャーの概要

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

- マスターノード-マスターノードは、データベースクライアントにサービスを提供す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、マスター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、フェールオーバーシナリオでマスター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使用できます。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フェールオーバーマネージャークラスターを指します。フェールオーバーマネージャークラスターは、クラウドまたは従来のネットワーク上のサーバーに存在し、JGroups ツールキットを使用して通信するマスターエージェント、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。

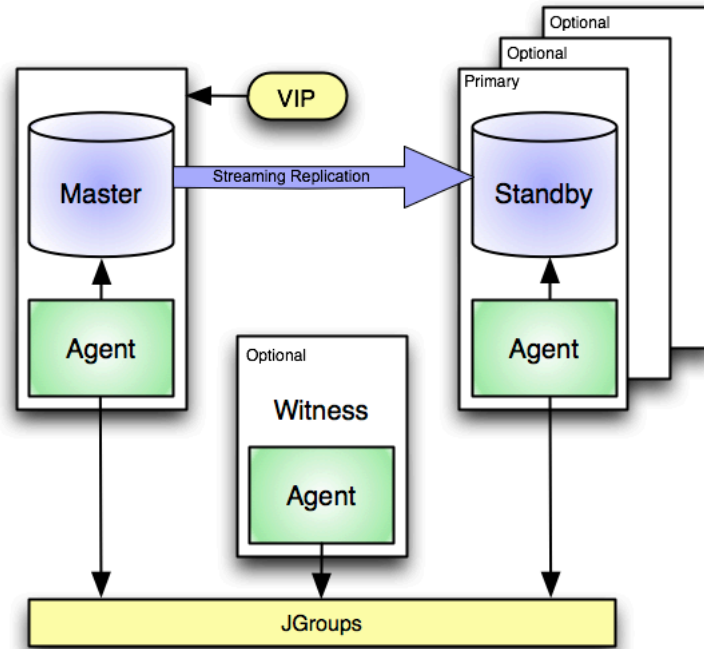


図 1 仮想 IP アドレスを使用する EFM シナリオ

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受けます。
- データベースが回復中でない場合、エージェントはマスターの役割を引き受けます。

フェールオーバーが発生した場合、フェールオーバーマネージャーは、昇格したスタンバイがクラスター内で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マスター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 は、フェールオーバーマネージャーが、メンバーノードが相互に通信してノード障害を検出できるクラスターを作成できるようにする技術を提供します。

上記の図は、仮想 IP アドレスを使用する Failover Manager クラスターを示しています。仮想 IP アドレスの代わりにロードバランサーを使用できます。独自のフェンシングスクリプトを提供する場合、障害が発生した場合にロードバランサーを再構成します。

2.1 サポートされているプラットフォーム

フェールオーバーマネージャー 3.8 は、次で実行されている EDB Postgres Advanced Server または PostgreSQL (バージョン 9.4 からバージョン 12) のインストールでサポートされています。

- CentOS 6.x および 7.x
- Red Hat Enterprise Linux 6.x および 7.x
- Oracle Enterprise Linux 6.x および 7.x
- Red Hat Enterprise Linux (IBM Power8 Little Endian または ppc64le) 7.x
- Debian 9
- SLES 12
- Ubuntu 18.04

2.2 前提条件

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

Java 1.8（以降）をインストールします

Failover Manager を使用する前に、最初に Java（バージョン 1.8 以降）をインストールする必要があります。Failover Manager は OpenJDK でテストされています。そのバージョンの Java をインストールすることを強くお勧めします。 **‘Java のインストール手順’**[<https://openjdk.java.net/install/>](https://openjdk.java.net/install/) はプラットフォーム固有です。

SMTP サーバーを提供する

ユーザー定義の通知スクリプト、電子メール、またはその両方で指定されているように、フェールオーバーマネージャークラスターから通知を受信できます。

- 電子メール通知を使用している場合、SMTP サーバーが Failover Manager シナリオの各ノードで実行されている必要があります。
- `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。

イベントが発生すると、フェールオーバーマネージャークラスターはスクリプト（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、次を参照してください。

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

ストリーミングレプリケーションの構成

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

データベースバージョン 11（またはそれ以前）では、`-sourcenode` オプションで指定しない限り、スイッチオーバー中に `recovery.conf` ファイルがランダムスタンバイノードから停止したマスターにコピーされます。スイッチオーバーを実行する前に、スタンバイノード上の `recovery.conf` ファイル内のパスが一貫していることを確認する必要があります。`-sourcenode` オプションの詳細については、[フェールオーバーマネージャークラスターの昇格を参照してください](#)。

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`、`restore_command`、および `promote_trigger_file` のプロパティが停止したマスターにコピーされます（`-sourcenode` オプションで指定されていない限り）。

レプリケーションスロットを使用して WAL セグメントを管理する場合、フェールオーバーマネージャークラスターは、フェールオーバー後のスタンバイデータベースの自動再構成をサポートしないことに注意してください。レプリケーションスロットを使用する場合、`auto.reconfigure` パラメーターを `false` に設定し、フェールオーバーが発生した場合にスタンバイサーバーを手動で再構成する必要があります。

pg_hba.conf ファイルを変更します

マスターノードとスタンバイノードの pg_hba.conf ファイルを変更し、クラスター内のすべてのノード間の通信を許可するエントリを追加する必要があります。次の例は、マスターノード上の 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。

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

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

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

マスターノードが再起動すると、フェールオーバーマネージャーはデータベースがマスターノードでダウンしていることを検出し、スタンバイノードをマスターの役割に昇格させる場合があります。これが発生した場合、(リブートされた) マスターノード上の Failover Manager エージェントは recovery.conf ファイルを書き込む機会を得られません。recovery.conf ファイルはデータベースサーバーの起動を妨げます。これが発生すると、リブートされたマスターノードは 2 番目のマスターノードとしてクラスターに戻ります。

これを防ぐには、データベースサーバーを起動する前に Failover Manager エージェントを起動します。エージェントはアイドルモードで起動し、クラスター内に既にマスターが存在するかどうかを確認します。マスターノードがある場合、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 番目のマスター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 (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、クラスター内のフェールオーバーマネージャープロセス間の tcp 通信を許可するルールをファイアウォール構成に追加する必要があります。例:

```
# iptables -I INPUT -p tcp --dport 7800:7810 -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。フェールオーバーマネージャーは、クラスタープロパティファイルで指定されたポートに対応するポートを介して接続します。

データベースユーザ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、フェールオーバーマネージャーに代わって次の機能呼び出すための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_reload_conf()
```

これらの各関数の詳細については、“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/index.html>>‘_

ユーザーには、構成変数の値を読み取るためのアクセス許可も必要です。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、pg_read_all_settings を参照してください<<https://www.postgresql.org/docs/12/default-roles.html>>‘_

2.3 チュートリアル-シンプルなフェイルオーバーマネージャークラスターの設定

このチュートリアルでは、テスト環境でのフェイルオーバーマネージャークラスターの迅速な構成について説明します。このガイドの他のセクションでは、運用展開用にフェイルオーバーマネージャークラスターを構成する前に読んで理解する必要がある重要な情報を提供します。

このチュートリアルでは、次のことを前提としています。

- データベースサーバーが実行されており、マスターと 1 つまたは 2 つのスタンバイノード間でストリーミングレプリケーションがセットアップされています。
- 各ノードに Failover Manager をインストールしました。

次の例では、efm という名前のクラスターを作成します。

マスターまたはスタンバイノードで構成プロセスを開始する必要があります。次に、構成ファイルを他のノードにコピーして、時間を節約します。

ステップ 1: 作業用構成ファイルの作成

提供されたサンプルファイルをコピーして EFM 構成ファイルを作成し、所有権を修正します。

```
cd /etc/edb/efm-3.8
cp efm.properties.in efm.properties
cp efm.nodes.in efm.nodes
chown efm:efm efm.properties
chown efm:efm efm.nodes
```

ステップ 2: 暗号化されたパスワードを作成する

暗号化されたパスワードを作成します（プロパティファイルに必要）：

```
/usr/edb/efm-3.8/bin/efm encrypt efm
```

画面の指示に従って、データベースパスワードの暗号化バージョンを作成します。

ステップ 3: efm.properties ファイルを更新する

cluster_name.properties ファイルには、フェイルオーバーマネージャークラスターの接続プロパティと動作を指定するパラメーターが含まれています。プロパティ設定の変更は、フェイルオーバーマネージャークラスターの起動時に適用されます。

次のプロパティは、Failover Manager クラスターを構成するために必要な最小限のプロパティです。本番システムを設定している場合は、[クラスタプロパティファイルを参照してください](#)。プロパティの完全なリスト。

データベース接続プロパティ（ミラーリング監視でも必要なため、必要に応じて他のデータベースに接続できます）：

```
db.user  
  
db.password.encrypted  
  
db.port  
  
db.database
```

データディレクトリの所有者（通常は `postgres` または `enterprisedb`）：

```
db.service.owner
```

EFM は、サーバーを再起動するときに `db.service.name` および `db.bin` プロパティを使用します。 `db.service.name` プロパティで提供されるサービス名は、`service` または `systemctl` でサーバーを再起動するときに使用されます。 `db.bin` プロパティ（Postgres の `bin` ディレクトリへのパス）で指定した値は、`pg_ctl` の呼び出しに使用されます。 `db.bin` は必須フィールドです。データベースをサービスとして実行している場合、`db.service.name` が必要です。

```
db.service.name  
  
db.bin
```

EFM が `recovery.conf` ファイルまたは `standby.signal` ファイルを検索または作成するデータディレクトリ：

```
db.recovery.dir
```

電子メール通知を受信するように設定します（通知テキストはエージェントログにも含まれます）。

```
user.email
```

これは、ノードのローカルアドレスと EFM に使用するポートです。他のノードはこのアドレスを使用してエージェントに到達し、エージェントはこのアドレスを使用してローカルデータベースに接続します（ローカルホストに接続するものではありません）。形式の例を以下に示します。

```
bind.address=1.2.3.4:7800
```

監視ノードでこのプロパティを `true` に設定し、マスターまたはスタンバイの場合は `false` に設定します。

```
is.witness
```

インターネットにアクセスせずにネットワークで実行している場合は、これをネットワークで使用可能なアドレスに変更します。

```
pingServerIp=8.8.8.8
```

実稼働クラスターを構成するとき、以下のプロパティは、システム構成および使用法に応じて `true` または `false` になります。EFM テストクラスターを構成している場合は、両方を `true` に設定して起動を簡素化します。

```
auto.allow.hosts=true

stable.nodes.file=true
```

ステップ 4:efm.nodes ファイルを更新する

cluster_name.nodes ファイルは起動時に読み込まれ、クラスターの残りの部分を見つける方法をエージェントに指示します。最初に起動したノードの場合は、後続のノードの認証を簡素化するために使用できます。

クラスター内の各ノードのアドレスとポートをこのファイルに追加します。1つのノードがメンバーシップコーディネーターとして機能します。リストには、少なくともメンバーシップコーディネーターの住所を含める必要があります。例えば:

```
1.2.3.4:7800

1.2.3.5:7800

1.2.3.6:7800
```

Failover Manager エージェントは efm.nodes ファイルの内容を検証しないことに注意してください。エージェントは、ファイル内の一部のアドレスに到達できないことを予期しています（たとえば、別のエージェントがまだ開始されていないなど）。efm.nodes ファイルの詳細については、[クラスターメンバーファイルを参照してください](#)。

ステップ 5:他のノードの構成

efm.properties および efm.nodes ファイルをサンプルクラスターの他のノードの /etc/edb/efm-3.8 ディレクトリにコピーします。ファイルをコピーした後、ファイルの所有権を変更して、ファイルが efm:efm によって所有されるようにします。efm.properties ファイルは、以下のプロパティを除いて、すべてのノードで同じにすることができます:

- ノードのローカルアドレスを使用するように bind.address プロパティを変更します。
- ノードが監視ノードの場合は、is.witness を true に設定します。ノードが監視ノードである場合、ローカルデータベースのインストールに関連するプロパティは無視されます。

ステップ 6:EFM クラスターを開始します

任意のノードで、Failover Manager エージェントを起動します。エージェントの名前は efm-3.8 です。プラットフォーム固有のサービスコマンドを使用して、サービスを制御できます。たとえば、CentOS または RHEL 7.x ホストでは、次のコマンドを使用します。

```
systemctl start efm-3.8
```

CentOS または RHEL 6.x ホストでは、次のコマンドを使用します。

```
service efm-3.8 start
```

エージェントが起動したら、次のコマンドを実行して、シングルノードクラスターのステータスを確認します。Allowed node host リストに他のノードのアドレスが表示されます。

```
/usr/edb/efm-3.8/bin/efm cluster-status efm
```

他のノードでエージェントを起動します。任意のノードで `efm cluster-status efm` コマンドを実行して、クラスターの状態を確認します。

エージェントの起動に失敗した場合は、起動ログで問題の詳細を確認してください。

```
cat /var/log/efm-3.8/startup-efm.log
```

スイッチオーバーの実行

マスターとスタンバイが同期していることがクラスターステータス出力に示されている場合は、次のコマンドでスイッチオーバーを実行できます。

```
/usr/edb/efm-3.8/bin/efm promote efm -switchover
```

このコマンドは、スタンバイを昇格させ、マスターデータベースをクラスター内の新しいスタンバイとして再構成します。元に戻すには、コマンドを再度実行します。

オンラインヘルプにすばやくアクセスするには、次のコマンドを呼び出すことができます。

```
/usr/edb/efm-3.8/bin/efm --help
```

`efm` コマンドラインツールの使用方法の詳細については、[EFM ユーティリティの使用を参照してください](#)。。

第 3

Failover Manager 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 EDB Web 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 ホストへのフェールオーバーマネージャーのインストール

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。

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。

```
yum -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。

`edb.repo` ファイルを作成した後、エディターの選択を使用して、有効なパラメーターの値が 1 であることを確認し、`edb.repo` の `username` および `password` プレースホルダーを置き換えます 4 '' 登録済みの EnterpriseDB ユーザーの名前とパスワードを含む仕様。

```
[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
```

設定ファイルへの変更を保存した後、`yum install` 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。

```
yum install edb-efm38
```

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、手動で解決する必要がある依存関係のリストが提供されます。

フェールオーバーマネージャーは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 Failover Manager サービスを制御するスクリプトを呼び出すための十分な特権を持つ efm という名前のユーザーも作成します。

フェールオーバーマネージャーを使用して、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場合は、enterprisedb を参照してください。<extending_efm_permissions>’。

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

1. Modify the *cluster properties file* on each node.

2. *cluster members* ファイルを変更します 各ノード。

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特定されているスクリプトを構成およびテストします。

4. クラスターの各ノードで Failover Manager エージェントを起動します。Failover Manager サービスの制御の詳細については、[“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

フェールオーバーマネージャーコンポーネントは、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-3.8/bin
図書館	/usr/edb/efm-3.8/lib
クラスター構成ファイル	/etc/edb/efm-3.8
ログ	/var/log/efm-3.8
ファイルをロックする	/var/lock/efm-3.8
ログローテーションファイル	/etc/logrotate.d/efm-3.8
sudo 設定ファイル	/etc/sudoers.d/efm-38
sudo なしで VIP にアクセスするためのバイナリ	/usr/edb/efm-3.8/bin/secure

3.2 Debian または Ubuntu ホストへのフェールオーバーマネージャーのインストール

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も必要です。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>

EDB Failover Manager テーブルのリンクに従って、資格情報を要求します。

次の手順では、EnterpriseDB apt 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. EnterpriseDB apt 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. フェールオーバーマネージャーをインストールします。

```
apt-get install edb-efm38
```

3.3 SLES ホストへのフェールオーバーマネージャーのインストール

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

フェールオーバーマネージャーをインストールする前に、スーパーユーザー権限を引き受け、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 http://zypp.enterprisedb.com/suse/epas96-sles.repo
zypper addrepo http://zypp.enterprisedb.com/suse/epas-sles-tools.repo
zypper addrepo http://zypp.enterprisedb.com/suse/epas-sles-dependencies.repo
```

このコマンドは、`/etc/zypp/repos.d` ディレクトリにリポジトリ構成ファイルを作成します。次に、次のコマンドを使用して SLES ホストのメタデータを更新し、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入力し、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。

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

```
zypper install SUSEConnect
SUSEConnect -r *registration_number* -e *user_id*
SUSEConnect -p PackageHub/12/x86_64
SUSEConnect -p sle-sdk/12/x86_64
```

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

```
zypper install edb-efm38
```

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>

フェイルオーバーマネージャーの設定

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

- *efm.properties*
- *efm.nodes*

4.1 クラスタープロパティファイル

フェールオーバーマネージャークラスターの各ノードには、それが存在する個々のノードのプロパティを含むプロパティファイル（デフォルトでは *efm.properties* という名前）があります。Failover Manager インストーラーは、`/etc/edb/efm-3.8` 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。

フェールオーバーマネージャーのインストールが完了したら、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-3.8/efm.properties.in /etc/edb/efm-3.8/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
```

注:既定では、フェールオーバーマネージャーはクラスタープロパティファイルの名前が *efm.properties* であると想定しています。プロパティファイルに *efm.properties* 以外の名前を付ける場合は、サービススクリプト

トまたはユニットファイルを変更して、フェールオーバーマネージャーに別の名前を使用するように指示する必要があります。

クラスタープロパティファイルを作成したら、必要に応じて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、“[クラスタープロパティの指定](#)”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-3.8 directory 内のファイルを見つけることを想定しています。プロパティファイルを別の場所に移動する場合は、新しい場所を指定するシンボリックリンクを作成する必要があります。

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 ユーザーとして呼び出されることに注意してください。

4.1.1 クラスタープロパティの指定

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

プロパティ値では大文字と小文字が区別されます。Postgres はパラメーター値に引用符付き文字列を使用しますが、Failover Manager ではプロパティ値に引用符付き文字列を使用できないことに注意してください。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'
```

フェールオーバーマネージャーでは、値を引用符で囲まないでください。

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

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

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

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

指定した `db.user` には、フェールオーバーマネージャーに代わって選択した PostgreSQL コマンドを呼び出すための十分な権限が必要です。詳細については、[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、[データベースパスワードの暗号化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してい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.recovery.dir' specified below.
db.service.owner=
```

サービスを開始または停止するときに `service` または `systemctl` コマンドを使用する場合、`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=
```

`db.recovery.dir` プロパティを使用して、クラスターのマスターノードで回復ファイルが書き込まれる場所を指定し、トリガーファイルがスタンバイで書き込まれます。このプロパティは、マスターノードとスタンバイノードで必要です。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. On a standby node,
# the trigger file location is read from the file in this directory.
# After a failover, the recovery.conf files on remaining standbys are changed
# to point to the new master db (a copy of the original is made first). On a
# master node, a recovery.conf file will be written during failover and
# promotion to ensure that the master node can not be restarted as the
# master database.
db.recovery.dir=
```

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、Advanced Server 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.recovery.dir location specified by the db.recovery.dir property will be used.
db.config.dir=
```

データベース設定ファイルの詳細については、**‘PostgreSQL ウェブサイトをご覧ください’**
<https://www.postgresql.org/docs/12/runtime-config-file-locations.html> ‘_

jdbc.sslmode 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 SSL 接続を使用するよう指示します。デフォルトでは、SSL は無効になっています。

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

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/12/static/ssl-tcp.html>

と

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

`user.email` プロパティを使用して、**Failover Manager** から送信された通知を受信する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、[通知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、フェールオーバーマネージャーがユーザー通知を送信する最小の重大度レベル、または通知スクリプトが呼び出されるタイミングを指定します。通知の完全なリストについては、[通知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
```

`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。このスクリプトは、フェールオーバーマネージャーがユーザー通知を生成するたびに呼び出されます。

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

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

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

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

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

`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=
```

Postgres の `pg_is_in_recovery()` 関数は、データベースの回復状態を報告するブール関数です。この関数は、データベースが回復中の場合は `true` を返し、データベースが回復中でない場合は `false` を返します。エージェントが起動すると、ローカルデータベースに接続し、`pg_is_in_recovery()` 関数を呼び出します。サーバーが `true` と応答すると、エージェントはスタンバイの役割を引き受けます。サーバーが `false` と応答した場合、エージェントはマスターの役割を引き継ぎます。ローカルデータベースがない場合、エージェントはアイドル状態になります。

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

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` property は、ローカルデータベースサーバーからの肯定的な応答をエージェントが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の試行が 60 秒以内に正に戻らない場合、フェールオーバーマネージャーはデータベースへの接続の最終試行を行います。応答が 10 秒以内に受信されない場合、Failover Manager はデータベース障害を宣言し、`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
# 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マスター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 master 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。クラスター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
```

`stop.isolated.master` プロパティを使用して、マスターエージェントが分離されていることを検出した場合、データベースをシャットダウンするようフェールオーバーマネージャーに指示します。 `true` (デフォルト) の場合、**Failover Manager** は `script.master.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 master 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.master.isolated' script, if a script is specified.
stop.isolated.master=true
```

`stop.failed.master` プロパティを使用して、マスターデータベースがデータベースに到達できない場合に、マスターデータベースのシャットダウンを試みるようフェールオーバーマネージャーに指示します。 `true` の場合、フェールオーバーマネージャーは、データベース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 master 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
# master node.
# If specified, a 'script.db.failure' script is run after this attempt.
stop.failed.master=true
```

`master.shutdown.as.failure` パラメーターを使用して、マスターノード上の **Failover Manager** エージェントのシャットダウンを障害として扱う必要があることを示します。このパラメーターが `true` に設定され、マスターエージェントが (何らかの理由で) 停止した場合、クラスターはマスター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 master agent shutdown as a failure. This can be set to
# true to treat a master 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
```

(次のページに続く)

(前のページからの続き)

```
# cause an unwanted promotion in the case of performing master
# database maintenance.
# Please see the user's guide for more information.
master.shutdown.as.failure=false
```

`master.shutdown.as.failure` プロパティは、マスター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。ユーザーがマスターフェールオーバーマネージャーエージェントを停止したように、ノードの適切なシャットダウンがクラスターの残りに表示されることがあります（たとえば、マスターデータベースのメンテナンスを実行するため）。`master.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が必要です。

`master.shutdown.as.failure` が `true` のときに **master** データベースでメンテナンスを実行するには、マスターエージェントを停止し、マスター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、**master** 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。

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

```
# This is the address of a well-known server that EFM can ping
# in an effort to determine network reachability issues. It
# might be the IP address of a nameserver within your corporate
# firewall or another server that \*should\* always be reachable
# via a 'ping' command from each of the EFM nodes.
#
# There are many reasons why this node might not be considered
# reachable: firewalls might be blocking the request, ICMP might
# be filtered out, etc.
#
# Do not use the IP address of any node in the EFM cluster
# (master, 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.
pingServerIp=8.8.8.8
```

`pingServerCommand` 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 'virtualIp' and 'pingServer'.
#
```

(次のページに続く)

(前のページからの続き)

```
# Make sure this command returns reasonably quickly - test it
# from a shell command line first to make sure it works properly.
pingServerCommand=/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` プロパティを使用すると、管理者はフェールオーバーマネージャーが同じデータベース接続を再利用してデータベースの状態を確認する回数を指定できます。デフォルト値は 0 です。これは、Failover Manager が毎回新しい接続を作成することを示します。このプロパティは、専用の監視ノードでは必要ありません。

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

`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 master
# 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
```

プライマリスタンバイがマスターに昇格した後、残りのスタンバイサーバーの自動再構成を有効または無効にするようにフェールオーバーマネージャーに指示するには、`auto.reconfigure` プロパティを使用します。プロパティを `true` に設定して自動再構成を有効にし（デフォルト）、または `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。Advanced Server 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 master. 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.recovery.dir. To turn off
# automatic reconfiguration, set this property to false.
auto.reconfigure=true
```

注意: `primary_conninfo` は、スペースで区切られた `keyword = value` ペアのリストです。

注:レプリケーションスロットを使用して WAL セグメントを管理している場合、自動再構成はサポートされていません。`auto.reconfigure` を `false` に設定する必要があります。フェールオーバーが発生した場合は、スタンバイサーバーを手動で再構成する必要があります。

`promotable` プロパティを使用して、ノードを昇格させないことを示します。設定を上書きするには、実行時に `efm set-priority` コマンドを使用します。`efm set-priority` コマンドの詳細については、[efm ユーティリティの使用](#) を参照してください。

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

`application.name` プロパティを使用して、古いマスター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 master. 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 master database as a standby. If this property is
# not set, Failover Manager will remove the parameter value
```

(次のページに続く)

(前のページからの続き)

```
# from primary_conninfo.
application.name=
```

注意: マスターおよびプロモーション可能なスタンバイで `application.name` プロパティを設定する必要があります。フェールオーバー/スイッチオーバーのイベントでは、マスター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。

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

`restore_command` の使用に関する詳細については、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 master node, use this_
↪property
# to have Failover Manager change the command when a new master is promoted.
#
# Use the %h placeholder to represent the address of the new master. During promotion
# it will be replaced with the address of the new master.
#
# 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=
```

スタンバイノードの数が必要なレベルを下回った場合、マスターデータベースを同期レプリケーションモードから解除するには、`reconfigure.sync.master` 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.master` を `false` に設定します。

```
# Take the master database out of synchronous replication mode when needed.
# If set to true, Failover Manager will clear the synchronous_standby_names
# configuration parameter on the master if the number of synchronous
# standbys drops below the required level for the master 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.
reconfigure.sync.master=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に低下した場合、マスター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、データベースが復旧していないかどうかを確認する前に **Failover Manager** が待機する秒数を指定します。

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

`restart.connection.timeout` プロパティを使用して、フェールオーバーマネージャーが新しく再構成されたマスターノードまたはスタンバイノードへの接続を試行する秒数を指定します。

```
# Time in seconds to keep trying to connect to a database after a
# start or restart command returns successfully but the database
# is not ready to accept connections yet (a rare occurrence). This
# applies to standby databases that are restarted when being
# reconfigured for a new master, and to master 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
```

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

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、`virtualIp.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、クラスターのマスターノードにのみ割り当てられます。 `virtualIp.single=true` を指定すると、フェイルオーバーの発生時に新しいマスターで同じ 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、“[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](#)”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 virtualIp 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 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 virtualIp and virtualIp.prefix values must be
# the same across all agents. If you set virtualIp.single to
# false, you can specify unique values for virtualIp and
# virtualIp.prefix on each node.
#
# If you are using an IPv4 address, the virtualIp.interface value
# should not contain a secondary virtual ip id (do not include
# ":1", etc).
virtualIp=
virtualIp.interface=
virtualIp.prefix=
virtualIp.single=true
```

注:マスター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。マスター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 が、VIP が使用中かどうかを確認してから障害が発生した場合、新しいマスターに割り当てる前に確認しないことを示します。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。マスター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場合を除き、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 master failure. Turning this off may allow
# the new master 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 master node will be isolated or
```

(次のページに続く)

(前のページからの続き)

```
# 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マスター障害のシナリオで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（マスター、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、ノードをロードバランサーからデタッチする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。文字列に `m`（マスターノードの場合）または `s`（スタンバイノードの場合）を含めるようにフェールオーバーマネージャーに指示するには、`%t` プレースホルダーを含めます。

```
# Absolute path to load balancer scripts
# The attach script is called when a node should be attached to
# the load balancer, for example after a promotion. The detach
# script is called when a node should be removed, for example
# when a database has failed or is about to be stopped. Use %h to
# represent the IP/hostname of the node that is being
# attached/detached. Use %t to represent the type of node being
# attached or detached: the letter m will be passed in for master 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マスター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 master
# and new primary node addresses. Use %p for new primary and %f
# for failed master. On a node that has just been promoted, %p
# should be the same as the node's efm binding address.
```

(次のページに続く)

(前のページからの続き)

```
#
# 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マスター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 master
# and new primary node addresses. Use %p for new primary and %f
# for failed master. 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、監視するデータベースが失敗したことをエージェントが検出した場合にフェールオーバーマネージャーが呼び出すオプションのユーザー指定スクリプトへの完全なパスを指定します。

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

`script.master.isolated` プロパティを使用して、マスターデータベースを監視するエージェントがマス

ターがフェールオーバーマネージャークラスターの大部分から分離されていることを検出した場合にフェールオーバーマネージャーが呼び出すオプションのユーザー指定スクリプトへの完全なパスを指定します。このスクリプトは、VIP がリリースされた直後に呼び出されます (VIP が使用中の場合)。

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

`script.remote.pre.promotion` プロパティを使用して、ノードがそのデータベースをマスター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マスター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-master agent nodes
# after a promotion.
#
# This optional user-supplied script will be invoked on nodes
# (except the new master) after a promotion occurs. The exit code
# from this script has no effect on Failover Manager, but will be
# included in a notification sent after the script executes.
#
# Pass a parameter (%p) with the script to identify the new
# primary node address.
#
```

(次のページに続く)

(前のページからの続き)

```
# 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、フェールオーバーマネージャーは通知を送信します。

`custom.monitor.safe.mode` を **true** に設定すると、スクリプトからゼロ以外の終了コードを報告するようにフェールオーバーマネージャーに指示しますが、終了コードの結果としてスタンバイを昇格させません。

```
# Absolute path to a custom monitoring script.
#
# Use script.custom.monitor to specify the location and name of
# an optional user-supplied script that will be invoked
# periodically to perform custom monitoring tasks. A non-zero
# exit value means that a check has failed; this will be treated
# as a database failure. On a master 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、拡張アクセス許可が必要なタスクを実行するときにフェールオーバーマネージャーによって呼び出されるコマンドを指定します。このオプションを使用して、システム認証に固有のコマンドオプションを含めます。

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

```
# Command to use in place of 'sudo' if desired when efm runs
# the efm_db_functions or efm_root_functions, or efm_address
# scripts.
# Sudo is used in the following ways by efm:
#
# sudo /usr/edb/efm-<version>/bin/efm_address <arguments>
# sudo /usr/edb/efm-<version>/bin/efm_root_functions <arguments>
# sudo -u <db service owner> /usr/edb/efm-<version>/bin/efm_db_functions <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、フェールオーバーマネージャーロックファイルの代替場所を指定します。このファイルにより、フェールオーバーマネージャーがノード上の単一クラスターに対して複数の（孤立している可能性のある）エージェントを起動できなくなります。

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

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

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

Failover Manager ホストで UDP または TCP プロトコルを有効にした後、syslog へのロギングを有効にできます。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
```

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

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

syslog ロギングの設定の詳細については、[syslog ログファイルエントリの有効化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、[ロギングの制御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。デフォルト設定は、Failover Manager エージェントが使用できるメモリの量を指定します。

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

データベースパスワードの暗号化

フェールオーバーマネージャーでは、データベース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 ユーティリティを使用します（`0 /usr/edb/efm-3.8/bin/1`）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>フェールオーバーマネージャークラスターの名前を指定します。

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

```
export EFMPASS=password
```

--from-env オプションを含めない場合、フェールオーバーマネージャークラスターは、クラスタープロパティファイルに配置するための暗号化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:多くの Java ベンダーは、完全な強度の暗号化が含まれたバージョンの Java を出荷していますが、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 Java Cryptography Extension (JCE) をダウンロードして有効にする必要があります。

次の例は、暗号化ユーティリティを使用して acctg クラスターのパスワードを暗号化する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-3.8/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: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、パスワードをプロパティファイルに貼り付け、Failover Manager サービスを開始します。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 efm-3.8 start
Starting local efm-3.8 service: [FAILED]
```

Failover Manager サービスの開始時にこのメッセージを受け取った場合、詳細については起動ログ (/var/log/efm-3.8/startup-efm.log にあります) を参照してください。

RHEL 7.x または CentOS 7.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 efm-3.8
```

クラスターが誤って別のクラスターのデータベースに接続するのを防ぐため、クラスター名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（`1safepassword`）に設定します。

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

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

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

暗号化されたパスワード（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 クラスターの各ノードには、現在の Failover Manager クラスターメンバーのリストを含むクラスターメンバーファイル（デフォルトでは `efm.nodes` 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-3.8` 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。

フェイルオーバーマネージャーのインストールが完了したら、テンプレートの作業用コピーを作成する必要があります。

```
# cp /etc/edb/efm-3.8/efm.nodes.in /etc/edb/efm-3.8/efm.nodes
```

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

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

デフォルトでは、フェイルオーバーマネージャーはクラスターメンバーファイルの名前が `efm.nodes` であることを期待します。クラスターメンバーファイルに `efm.nodes` 以外の名前を付ける場合、Failover Manager サービススクリプトを変更して、新しい名前を使用するように Failover Manager に指示する必要があります。

開始された最初の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。

メンバーシップコーディネーターは、クラスターの現在のメンバーに一致するように `efm.nodes` ファイルの内容を更新します。エージェントがクラスターに参加またはクラスターから離脱すると、他のエージェントの `efm.nodes` ファイルが更新され、現在のクラスターメンバーシップが反映されます。 *efm stop-cluster* を呼び出す場合 コマンド、フェイルオーバーマネージャーはファイルを変更しません。

メンバーシップコーディネーターがクラスターを離れると、別のノードが役割を引き受けます。 *efm cluster-status* を使用できます メンバーシップコーディネーターのアドレスを見つけるコマンド。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した場合、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* ）は `'1'` に設定されます。

`stable.nodes.file` プロパティ（ *クラスタープロパティファイルにある場合* ）が `'1'` *stable.nodes.file* ファイルを更新しません。この動作は、クラスターメンバーの IP アドレスが頻繁に変更されない場合に最も役立ちます。

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

Failover Manager のインストール中に、インストーラーは `efm` という名前のユーザーを作成します。efm には、通常データベース所有者またはオペレーティングシステムのスーパーユーザーに制限されている管理機能を実行するための十分な特権がありません。

- データベース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_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 ユーザーの代わりに管理機能を実行します。

`sudoers` ファイルには、ユーザー `efm` が `postgres` または `enterprisedb` が所有するクラスターの Failover Manager サービスを制御できるエントリが含まれています。`sudoers` ファイルのコピーを変更して、他のユーザーが所有する Postgres クラスターを `efm` に管理する権限を付与できます。

`efm-38` 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-3.8 /bin/efm_db_functions
efm ALL=(enterprisedb) NOPASSWD: /usr/edb/efm-3.8
/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-3.8 /bin/efm_root_functions
# Allow user 'efm' to sudo efm_address as root for VIP tasks.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-3.8 /bin/efm_address
```

(次のページに続く)

(前のページからの続き)

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

Failover Manager を使用して、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している場合は、efm-38 ファイルのコピーを作成し、ユーザーが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 efm_functions スクリプト。

権限の問題のためにエージェントを開始できない場合、デフォルトの /etc/sudoers ファイルの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 Manager は `sudo` を使用して、システム機能へのアクセスを安全に管理します。 `sudo` アクセスなしで実行するようにフェールオーバーマネージャーを構成する場合、次の操作にはルートアクセスが必要であることに注意してください。

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

`sudo` を使用せずにフェールオーバーマネージャーを実行するには、フェールオーバーマネージャーに代わって管理機能を実行する権限を持つデータベースプロセスの所有者を選択する必要があります。ユーザーは、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-3.8` および `/var/lock/efm-3.8` 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-3.8/efm.properties.in <directory/>cluster_name>.properties
```

```
cp /etc/edb/efm-3.8/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。また、`db.service.name` プロパティが空白であることを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-3.8/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と名前を指定します。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提供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 をサービスとして管理できるようにするには、カスタムスクリプトまたはユニットファイルを提供する必要があります。

フェイルオーバーマネージャーは、`/usr/edb/efm-3.8/bin/secure/`にある `manage-vip` という名前のバイナリを使用して、`sudo` 権限なしで VIP 管理操作を実行します。このスクリプトは、`setuid` を使用して、仮想 IP アドレスの管理に必要な特権を取得します。

- このディレクトリには、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` グループによってのみ実行可能です。

セキュリティ上の理由から、`/usr/edb/efm-3.8/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。

`sudo` なしで Failover Manager を使用方法の詳細については、次の Web サイトをご覧ください。

<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 は `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。

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ないことに注意してください。これらの環境では、別のメカニズム（AWS の Elastic IP アドレスなど）を使用する必要があります。これは、フェンシングまたはプロモーション後のスクリプトで必要に応じて変更できます。

デフォルトでは、スクリプトは次の場所にあります。

```
/usr/edb/efm-x.x/bin/efm_address
```

ここで、`x.x` は EFM バージョンを指定します。

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 IPv4 IP アドレスを割り当てるには:

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

仮想 IPv6 IP アドレスを割り当てるには:

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

仮想アドレスを解放するには:

```
# 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 `virtualIp.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 `virtualIp` プロパティで指定された名前と一致します。

`prefix` は、クラスタープロパティファイルの `virtualIp.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、[クラスタープロパティファイルを参照してください](#)。

。

`root` ユーザーとして `efm_address` スクリプトを呼び出す必要があります。efm ユーザーはインストール中に作成され、`sudoers` ファイルで `efm_address` スクリプトを実行する権限が付与されます。`sudoers` ファイルの詳細については、`efm_address` を参照してください。<extending_efm_permissions>‘。

注:VIP アドレス（または `bind.address` 以外のアドレス）がノードに割り当てられている場合、オペレーティングシステムはデータベース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

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接続を許可してください。

VIP のテスト

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

```
virtualIp=172.24.38.239
virtualIp.interface=eth0
virtualIp.prefix=24
pingServerCommand=/bin/ping -q -c3 -w5
```

注意: `virtualIp.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、`pingServerCommand` 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. マスターノードで `efm_address add4 eth0 172.24.38.239/24` 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 master 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. すべてのノードに **VIP** を割り当てるには、すべてのスタンバイノードで手順 2 を繰り返します。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、フェールオーバーマネージャーを起動する前に、非マスターノードから **VIP** を解放します。

注:**VIP** に使用されるネットワークインターフェイスは、**Failover Manager** エージェントの `bind.address` 値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。マスターエージェントは、フェールオーバー中に必要に応じて **VIP** をドロップし、フェールオーバーマネージャーは、スタンバイを昇格する前に **VIP** が利用できないことを確認します。バインドアドレスネットワークに障害が発生すると、マスターの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している場合、マスターエージェントがドロップする前に、残りのクラスターが到達可能な **VIP** をチェックするタイミング条件が発生する場合があります。この場合、EFM は `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、フェールオーバーが期待どおりに行われるようになります。

フェイルオーバーマネージャーの使用

フェイルオーバーマネージャーは、1 つ以上のスタンバイサーバーでクラスターの監視とフェイルオーバーをサポートします。リソースの需要が増減するにつれて、クラスターにノードを追加または削除できます。

マスターノードが再起動すると、フェイルオーバーマネージャーはデータベースがマスターノードでダウンしていることを検出し、スタンバイノードをマスターの役割に昇格させることがあります。これが発生した場合、(リブートされた) マスターノード上の **Failover Manager** エージェントは `recovery.conf` ファイル (サーバーバージョン 11 以前の場合) または `standby.signal` ファイル (サーバーバージョン 12 の場合) を書き込む機会を得ません。; 再起動されたマスターノードは、2 番目のマスターノードとしてクラスターに戻ります。これを防ぐには、データベースサーバーを起動する前に **Failover Manager** エージェントを起動します。エージェントはアイドルモードで起動し、クラスター内に既にマスターが存在するかどうかを確認します。マスターノードがある場合、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 番目のマスターとして起動しません。

5.1 フェイルオーバーマネージャークラスターの管理

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

デフォルトでは、いくつかの *efm* コマンドは '0' 'efm' '0' グループに追加することで、ユーザーがこれらのコマンドを呼び出すことを選択的に許可できます。コマンドは次のとおりです。

- *efm allow-node*

- *efm disallow-node*
- *efm* プロモート
- *efm* 履歴書
- *efm set-priority*
- *efm stop-cluster*
- *efm upgrade-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 クラスターのノードは任意の順序で起動できます。

RHEL 6.x または CentOS 6.x で Failover Manager クラスターを起動するには、スーパーユーザー権限を引き受けて、コマンドを呼び出します。

```
service efm-3.8 start
```

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

```
systemctl start efm-3.8
```

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

ノードが専用の監視ノードではない場合、フェールオーバーマネージャーはローカルデータベースに接続し、`pg_is_in_recovery()` 関数を呼び出します。サーバーが `false` に応答すると、エージェントはノードがマスター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、フェールオーバーマネージャー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、フェールオーバーマネージャーエージェントは、提供されたデータベース資格情報をチェックして、クラスター内のすべてのデータベースに接続できることを確認します。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。

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

5.1.2 クラスターへのノードの追加

いつでもフェールオーバーマネージャークラスターにノードを追加できます。クラスターにノードを追加するときは、クラスターを変更して新しいノードを許可し、クラスターを見つける方法を新しいノードに伝える必要があります。

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 Failover Manager 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フェールオーバーマネージャーサービスの制御の詳細については、[EFM ユティリティの使用を参照してください](#)。

フェールオーバーマネージャーエージェントをインストールし、新しいノードでクラスタープロパティファイルを作成します。プロパティファイルの変更の詳細については、[クラスタープロパティファイルを参照してください](#)。

2. 新しいノードでクラスターメンバーファイルを作成し、Membership Coordinator のエントリを追加します。クラスターメンバーファイルの変更の詳細については、[クラスターメンバーファイル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、Failover Manager エージェントを起動します。RHEL 6.x または CentOS 6.x で Failover Manager クラスターを起動するには、スーパーユーザー権限を引き受けて、コマンドを呼び出します。

```
service efm-3.8 start
```

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

```
systemctl start efm-3.8
```

新しいノードがクラスターに参加すると、フェールオーバーマネージャーは `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:現在のノードの便利なスタンバイになるには、PostgreSQL ストリーミングレプリケーションシナリオでノードをスタンバイにする必要があります。

5.1.3 スタンバイの優先度の変更

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優先度に影響を与えることができます。フェールオーバーマネージャークラスターの既存のメンバーでコマンドを呼び出し、メンバーの IP アドレスの後に優先度の値を指定します。

たとえば、次のコマンドは、10.0.1.9 を監視している `acctg` クラスターメンバーがプライマリスタンバイ (1) であることをフェールオーバーマネージャーに指示します。

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

スタンバイの優先順位を 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。

たとえば、ノード 10.0.1.10 のプロパティファイルに `promotable=false` の設定が含まれており、`efm set-priority` を使用して 10.0.1.10 のプロモーション優先度をイベントで使用されるスタンバイに設定する場合フェイルオーバーの場合、`efm set-priority` コマンドで指定された値はプロパティファイルの値を上書きします。

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

フェールオーバーが発生した場合、フェールオーバーマネージャーはまず **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最新のデータが等しく含まれている場合、ユーザーが指定した優先順位の値が高いノードがマスターに昇格します。スタンバイノードの優先度の値を確認するには、次のコマンドを使用します。

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

注:ノードがクラスターから分離され、後でクラスターに再び参加すると、プロモーションの優先順位が変更される場合があります。

5.1.4 フェールオーバーマネージャーノードの昇格

フェールオーバーマネージャークラスターの任意のノードで `efm promote` を呼び出して、スタンバイデータベースのマスターデータベースへの手動昇格を開始できます。

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、フェールオーバーマネージャークラスターの名前です。

元のマスターをスタンバイとして再構成するには、`-switchover` オプションを含めます。`-switchover` キーワードを含める場合、クラスターにはマスターノードと少なくとも 1 つのスタンバイを含める必要があります、ノードは同期している必要があります。

リカバリ設定をマスターにコピーするノードを指定するには、`-sourcenode` キーワードを含めます。

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。

フェンシングスクリプトと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フェールオーバーマネージャーに指示しないようにするには、`-noscripts` キーワードを含めます。

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マスターノードにコピーされます。サーバーバージョン 12 の場合、`primary_conninfo`、`restore_command`、および `promote_trigger_file` パラメーターがコピーされ、メモリーに保存されます。
- **master** データベースは停止しています。
- **VIP** を使用している場合、アドレスはマスターノードから解放されます。
- スタンバイがマスターノードを置き換えるために昇格され、**VIP** を取得します。
- 新しいマスター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 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。
- 古いマスターが開始されます。エージェントはスタンバイとしての監視を再開します。

手動プロモーション中、マスターエージェントは仮想 IP アドレスを解放してから、`db.recovery.dir` プロパティで指定されたディレクトリに `recovery.conf` ファイルを作成します。`recovery.conf` ファイルはすべてのサーバーバージョンで作成され、ファイルが削除されるまで古い **master** データベースが起動しないようにし、ノードがクラスター内の 2 番目のマスターとして起動するのを防ぎます。

マスターエージェントは実行されたままで、ステータスが `Idle` になります。

スタンバイエージェントは、既知のアドレスに **ping** を送信する前に仮想 IP アドレスが使用されていないことを確認して、エージェントがネットワークから隔離されないようにします。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マスターに昇格させます。次に、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。

ノードをマスターの役割に戻すには、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、[EFM ユーティリティの使用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ーエージェントの停止

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

RHEL 6.x または CentOS 6.x で Failover Manager エージェントを停止するには、スーパーユーザー権限を引き受けて、コマンドを呼び出します。

```
service efm-3.8 stop
```

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

```
systemctl stop efm-3.8
```

efm disallow-node コマンド（ノードのアドレスを許可ノードホストリストから削除する）を呼び出すまで、service efm-3.8 start コマンドを使用して、efm allow-node コマンドをもう一度。

master.shutdown.as.failure がなければ、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。プロパティは '0' に設定されます。

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

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

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

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

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

efm disallow-node コマンドは、ノードの IP アドレスを Failover Manager の許可ノードホストリストから削除します。efm の ID または既存のノード（現在実行中のクラスターの一部）の OS スーパーユーザーを想定し、クラスター名とノードの IP アドレスを指定して efm disallow-node コマンドを呼び出します。

```
efm disallow-node <cluster_name> <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。エージェントを停止するまで、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、*`efm allow-node`* コマンドを使用する必要があります ノードをクラスターに再度追加します。

5.2 単一ノードでの複数のエージェントの実行

フェールオーバーマネージャーノードで複数のマスター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。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。複数のデータベースクラスターを監視するようにフェールオーバーマネージャーを構成し、異なるクラスターのフェールオーバーマネージャーエージェントが相互に干渉しないようにするには、以下を行う必要があります。

1. 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各クラスターのメンバーごとに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 CentOS 6.x システム上）またはユニットファイル（RHEL または CentOS 7.x システム上）をカスタマイズして、クラスタープロパティとクラスターメンバーファイルの名前を指定します。
4. 各クラスターのサービスを開始します。

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

- acctg のデータは /opt/pgdata1 にあります。そのサーバーはポート 5444 を監視しています。
- 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。

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

virtualIp（使用する場合）

virtualIp.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=7c801b32a05c0c5cb2ad4ffbdba5e8f9a

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
```

同じノードで複数の Failover Manager クラスターエージェントをセットアップする場合、一部のパラメーターは特別な注意が必要です。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場合は、情報を明確に保つ方が簡単かもしれません。

各クラスターのクラスタープロパティファイルを作成する場合、db.recovery.dir パラメーターは、各データベースクラスター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。Failover Manager クラスターが仮想 IP アドレスを使用しない場合は、これらのパラメーターを空白のままにします。

```
virtualIp

virtualIp.interface

virtualIp.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目指す各クラスター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。このステップはプラットフォーム固有です。RHEL 6.x または CentOS 6.x を使用している場合は、[RHEL 6.x](#) または [CentOS 6.x](#) を参照してください。; RHEL 7.x または CentOS 7.x を使用している場合は、[RHEL 7.x](#) または [CentOS 7.x](#) を参照してください。。

注: 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 RHEL 6.x または CentOS 6.x

RHEL 6.x または CentOS 6.x を使用している場合、efm-3.8 サービススクリプトを各クラスターに一意の名前で新しいファイルにコピーする必要があります。例:

```
# cp /etc/init.d/efm-3.8 /etc/init.d/efm-acctg
# cp /etc/init.d/efm-3.8 /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 7.x または CentOS 7.x

RHEL 7.x または CentOS 7.x を使用している場合、efm-3.8 ユニットファイルを各クラスターに一意の名前で新しいファイルにコピーする必要があります。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。

```
/etc/systemd/system/efm-acctg.service
/etc/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、値は次を指定します。

```
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-3.8/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.2Fadd_a_custom_unit_file.3F

フェールオーバーマネージャークラスターの監視

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

6.1 クラスターステータスレポートの確認

`efm cluster-status` [クラスタープロパティファイル](#) コマンドは、Failover Manager クラスターのステータスに関する情報を含むレポートを返します。コマンドを呼び出すには、次を入力します。

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

次のステータスレポートは、4 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	
Standby		172.19.12.163	UP	UP	
Master		172.19.14.9	UP	UP	
Allowed node host list:					
172.19.14.9 172.19.12.163 172.19.10.2					
Membership coordinator: 172.19.14.9					

(次のページに続く)

(前のページからの続き)

```
Standby priority host list:
172.19.12.163 172.19.10.2
```

```
Promote Status:
```

DB Type	Address	WAL Received LSN	WAL Replayed LSN	Info
Master	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 master. 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	
Standby	172.19.12.163	UP	UP	
Master	172.19.14.9	UP	UP	

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。

フェールオーバーマネージャーエージェントは、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
Master	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ているが、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、マスターノードとスタンバイノードのデータベースが実行中で同期していることを示します。
- ゼロ以外の終了コードは、問題があることを示します。次の問題により、ゼロ以外の終了コードがトリガーされる場合があります。

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

フェールオーバーマネージャーは、指定されたデータベースパスワードを解読できません。

データベースに接続して xlog の場所を取得する際に問題が発生しました。

マスター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マスターと同期していません。

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

Postgres Enterprise Manager (PEM) を使用してサーバーを監視する場合、Streaming Replication Analysis ダッシュボード (PEM グラフィカルインターフェイスの一部) を構成して、Streaming Replication シナリオの一部であるマスターノードまたはスタンバイノードの状態を表示できます。

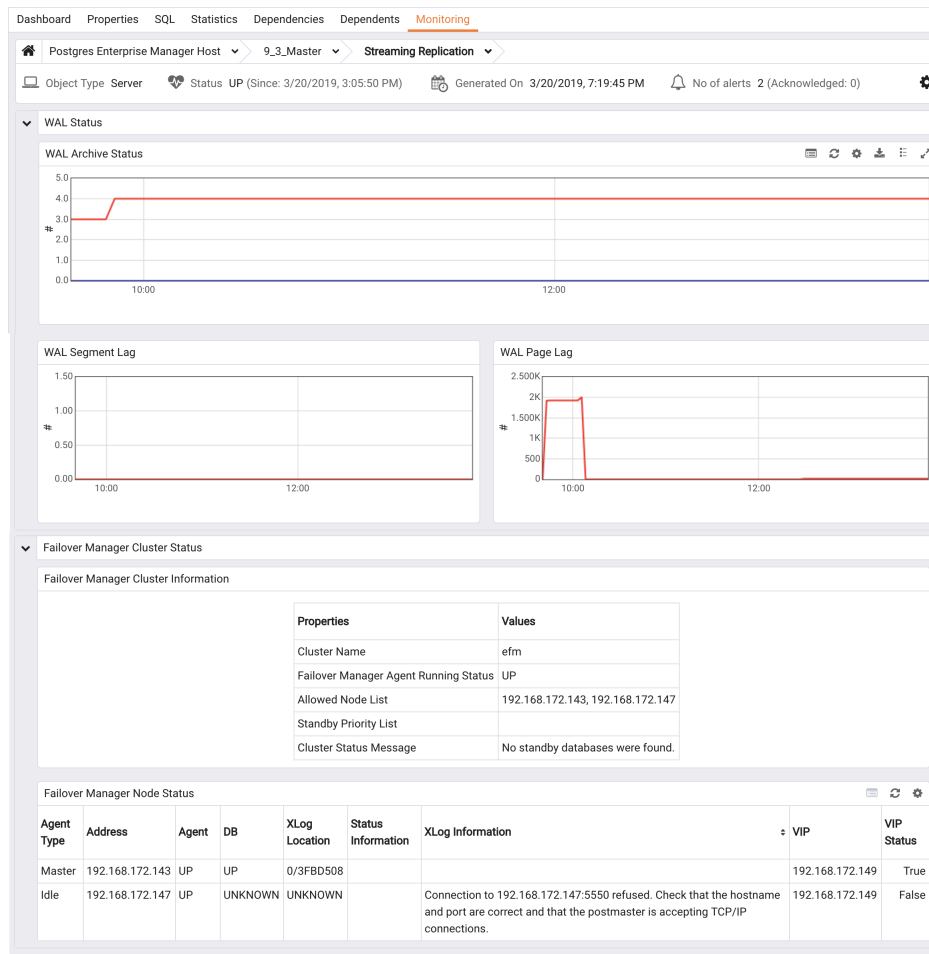


図 1 ストリーミングレプリケーションダッシュボード (マスターノード)

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 (レプリケーションマスターまたはレプリケーションスレーブ) のステータスを識別し、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。

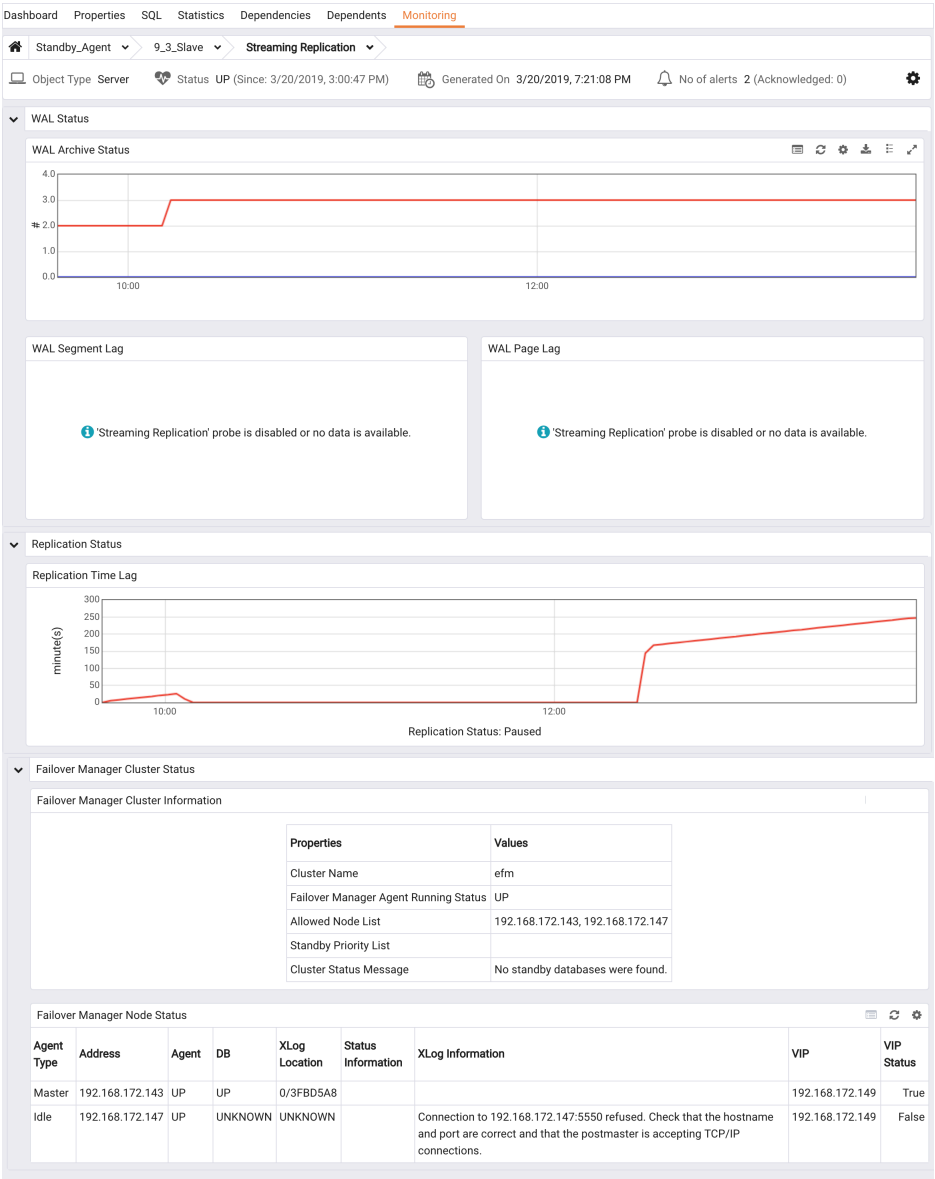


図 2 ストリーミングレプリケーションダッシュボード（スタンバイノード）

デフォルトでは、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情報を提供する PEM レプリケーションプローブは無効になっています。

レプリケーションシナリオのマスター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 EnterpriseDB Web 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

フェールオーバーマネージャーは、クラスター管理を支援する efm ユーティリティを提供します。RPM インストーラーは、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-3.8/bin ディレクトリに追加します。

```
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。

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 cluster-status <cluster_name>
```

efm cluster-status コマンドを呼び出して、フェールオーバーマネージャークラスターの状態を表示します。クラスターステータスレポートの詳細については、[フェールオーバーマネージャークラスターの監視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、フェールオーバーマネージャークラスターの状態を json 形式で表示します。表示される情報の形式は efm cluster-status コマンドによって生成される表示とは異なりますが、情報ソースは同じです。

次の例は、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": {
```

(次のページに続く)

(前のページからの続き)

```

    "172.16.144.176": {
      "type": "Witness",
      "agent": "UP",
      "db": "N\A",
      "vip": "",
      "vip_active": false
    },
    "172.16.144.177": {
      "type": "Master",
      "agent": "UP",
      "db": "UP",
      "vip": "",
      "vip_active": false,
      "xlog": "2\77000220",
      "xloginfo": ""
    },
    "172.16.144.180": {
      "type": "Standby",
      "agent": "UP",
      "db": "UP",
      "vip": "",
      "vip_active": false,
      "xlog": "2\77000220",
      "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 disallow-node <cluster_name> <ip_address>`

`efm disallow-node` コマンドを呼び出して、指定されたノードを許可ホストリストから削除し、ノードがクラスターに参加しないようにします。`efm disallow-node` コマンドを呼び出すときに、クラスターの名前とノードの IP アドレスを指定します。このコマンドは、efm、efm グループのメンバー、またはルートによって呼び出される必要があります。

```
efm 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。-from-env オプションを含めて、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するようにフェールオーバーマネージャーに指示します。詳細については、[データベースパスワードの暗号化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、フェールオーバーマネージャーにスタンバイからマスターへの手動フェールオーバーを実行するよう指示します。

手動での昇格は、ステータスコマンドで、クラスターにマスターと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行ってください。最新のスタンバイがない場合、フェールオーバーマネージャーは続行する前にプロンプトを表示します。

スタンバイノードを昇格させるために -switchover 句を含め、マスター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 ファイルを古いマスター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。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フェールオーバーマネージャーに指示するには、-noscripts キーワードを含めます。

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

このコマンドは、クラスタープロパティファイルの auto.failover パラメータ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。

```
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>
```

efm set-priority コマンドを呼び出して、フェールオーバー優先順位をスタンバイノードに割り当てます。値は、フェールオーバーのイベントで新しいノードが使用される順序を指定します。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。

priority is an integer value of 1 to *n*, where *n* is the number of standby nodes in the list. Specify a value of 1 to indicate that the new node is the primary standby, and will be the first node promoted in the event of a failover. A priority value of 0 instructs Failover Manager to not promote the standby.

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

efm stop-cluster コマンドを呼び出して、すべてのノードでフェールオーバーマネージャーを停止します。このコマンドは、フェールオーバーマネージャーにクラスター上の各ノードに接続し、既存のメン

バーにシャットダウンするよう指示します。このコマンドは実行中のデータベースには影響しませんが、コマンドが完了すると、フェールオーバー保護は機能しません。

注:efm stop-cluster コマンドを呼び出すと、許可されたノード情報がすべて許可ノードホストリストから削除されます。

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

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

efm upgrade-conf コマンドを呼び出して、既存の Failover Manager インストールから構成ファイルをコピーし、Failover Manager インストールに必要なパラメーターを追加します。ユーティリティを呼び出すときに、以前のクラスターの名前を指定します。このコマンドは、ルート権限で起動する必要があります。

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

```
efm --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

Failover Manager サービスの制御

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

- Failover Manager サービスで使用するプロパティを含む `efm.properties` という名前の設定ファイル。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。

単一の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 サービスを制御するコマンドはプラットフォーム固有です。

8.1 RHEL 6.x および CentOS 6.x でのサービスユーティリティの使用

RHEL 6.x および CentOS 6.x では、Failover Manager は、`/etc/init.d` にある（デフォルトで）`efm-3.8` 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。

次のサービスコマンドを使用して、RHEL 6.x または CentOS 6.x ホストにある Failover Manager エージェントを制御します。

```
service efm-3.8 start
```

`start` コマンドは、現在のノードで Failover Manager エージェントを起動します。ローカルフェールオーバーマネージャーエージェントはローカルデータベースを監視し、他のノードのフェールオーバーマネージャーと通信します。Failover Manager クラスター内のノードは、任意の順序で起動できます。

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 efm-3.8 stop
```

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

```
service efm-3.8 status
```

`status` コマンドは、呼び出された Failover Manager エージェントのステータスを返します。任意のノードで `status` コマンドを呼び出して、フェールオーバーマネージャーにステータス情報を返すように指示できます。例:

```
[witness@localhost ~]# service efm-3.8 status
efm-3.8 (pid 50836) is running...
```

```
service efm-3.8 help
```

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

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

RHEL 7.x および CentOS 7.x では、Failover Manager は、`/usr/lib/systemd/system` にある `efm-3.8.service` という名前の Linux サービス（デフォルト）として実行されます。フェールオーバーマネージャーによって監視される各データベースクラスターは、レプリケーションクラスターの各ノードでサービスのコピーを実行します。

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

```
systemctl start efm-3.8
```

`start` コマンドは、現在のノードで Failover Manager エージェントを起動します。ローカルフェールオーバーマネージャーエージェントはローカルデータベースを監視し、他のノードのフェールオーバーマネージャーと通信します。Failover Manager クラスター内のノードは、任意の順序で起動できます。

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 efm-3.8
```

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

```
systemctl status efm-3.8
```

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

```
[root@ONE ~]]> systemctl status efm-3.8
efm-3.8.service - EnterpriseDB Failover Manager 3.8
   Loaded: loaded (/usr/lib/systemd/system/efm-3.8.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/efm-3.8/bin/runefm.sh start $
   ↳ {CLUSTER} (code=exited, status=0/SUCCESS)
   Main PID: 58180 (java)
   CGroup: /system.slice/efm-3.8.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/efm-3.8/lib/EFM-3.8.0.jar -Xmx128m -
   ↳ agentlib:jdpw=transport...
```

ロギングの制御

フェイルオーバーマネージャーは、エージェントごとに 1 つのログファイルを書き込み、エージェントごとに 1 つの起動ログを /var/log/<cluster_name>-3.8 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の 0 ‘jgroups.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。<loglevel>：

```
# 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（ログ出力の多い順）は次のとおりです。

- 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。

既定では、フェールオーバーマネージャーのログファイルは毎日ローテーションされ、圧縮され、1 週間保存されます。ログローテーションファイル（`/etc/logrotate.d/efm-3.8`）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

フェールオーバーマネージャーは、syslog ロギングをサポートしています。syslog ロギングを実装するには、syslog を構成して UDP または TCP 接続を許可する必要があります。

syslog への接続を許可するには、`/etc/rsyslog.conf` ファイルを編集し、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
```

フェールオーバーマネージャーホスト上の `rsyslog.conf` ファイルを変更した後、フェールオーバーマネージャーのプロパティを変更してログを有効にする必要があります。選択したエディターを使用して、[プロパティファイルを変更します](#)。（*I*）実装したいロギングのタイプを指定します:

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

また、[syslog の詳細を指定する必要があります](#) ご使用のシステム用。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

フェールオーバーマネージャーは、クラスターに影響する重要なイベントが発生すると、電子メール通知を送信したり、通知スクリプトを呼び出したりします。電子メール通知を送信するようにフェールオーバーマネージャーを構成した場合は、クラスターの各ノードでポート 25 で実行されている SMTP サーバーが必要です。次のパラメーターを使用して、フェールオーバーマネージャーの通知動作を構成します。

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

構成プロパティの編集の詳細については、“[クラスタープロパティの指定](#)”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 は、各通知に重大度レベルを割り当てます。次のレベルは、必要な注意のレベルが高まることを示しています。

INFO は、エージェントに関する情報メッセージを示しており、手動による介入は必要ありません（たとえば、フェールオーバーマネージャーが起動または停止した）。

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、フェールオーバーが発生しました）。

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ましたが、完了できませんでした）。

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 の通知はすぐにユーザーの注意を必要としますが、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。

notification.level を使用できます 通知をトリガーする最小の重大度レベルを指定するプロパティ。

注意:管理用メールアドレスへの通知の送信に加えて、すべての通知はクラスターログファイル（`/var/log/efm-3.8/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
実行されたフェンシングスクリプト	実行されたフェンシングスクリプト* script_name 結果: script_results *
実行されたプロモーション後スクリプト	実行されたプロモーション後スクリプト* script_name 結果: script_results *
実行されたりモート事前プロモーションスクリプト	実行されたりモート事前プロモーションスクリプト* script_name 結果: script_results *
実行されたりモートポストプロモーションスクリプト	実行されたりモートポストプロモーションスクリプト* script_name 結果: script_results *
実行されたポストデータベース障害スクリプト	実行されたポストデータベースエラースクリプト* script_name 結果: script_results *
実行されたマスター分離スクリプト	実行されたマスター分離スクリプト* script_name 結果: script_results *
クラスター* cluster_name の node_address *で実行されている監視エージェント	監視エージェントが実行されています。
クラスター* cluster_name の node_address *で実行されているマスターエージェント	マスター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。
クラスター* cluster_name の node_address *で実行されているスタンバイエージェント	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。
クラスター* cluster_name の ノード node_address *で実行中のアイドルエージェント	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。
ノード* node_address *へのVIPの割り当て	VIP * VIP_address をノード node_address に割り当てる結果: script_results *
ノード* node_address *からVIPを解放する	ノード* node_address から VIP * VIP_address *を解放する結果: script_results *
クラスター* cluster_name *の自動再開チェックを開始しています	このノード上のエージェントは、* 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。
実行されたエージェント再開スクリプト	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 *
プロモーション中にバックアップされた WAL ログ	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、ディスク領域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
クラスター* <i>cluster_name</i> の <i>node_address</i> *で監視エージェントが終了しました	監視エージェントが終了しました。
マスターエージェントがクラスター* <i>cluster_name</i> の <i>node_address</i> *で終了しました	データベースの状態は監視されていません。
クラスター* <i>cluster_name</i> *は、マスターノードが脱退したことを通知しました	マスター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。
クラスター* <i>cluster_name</i> の <i>node_address</i> *でスタンバイエージェントが終了しました	データベースの状態は監視されていません。
クラスター* <i>cluster_name</i> の <i>node_address</i> *での昇格中にエージェントが終了しました	データベースの状態は監視されていません。
エージェントはクラスター* <i>cluster_name</i> の <i>node_address</i> *で終了しました	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。
クラスター* <i>cluster_name</i> *の <i>node_address</i> *でエージェントが終了しました	エージェントは終了しました。この通知は、通常、起動が完了する前にエージェントが終了する起動時に生成されます。
非マスターノードに割り当てられた仮想 IP アドレス	仮想 IP アドレスは、非マスターノードに割り当てられているようです。競合を避けるため、フェールオーバーマネージャーは VIP を解放します。VIP がマスター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。
マスターノードに割り当てられていない仮想 IP アドレス。	仮想 IP アドレスがマスターノードに割り当てられていないようです。EDB Postgres Failover Manager は VIP の再取得を試みます。
クラスター* <i>cluster_name</i> *のクラスターにスタンバイエージェントがありません	The standbys on <i>cluster_name</i> have left the cluster.
クラスター* <i>cluster_name</i> *のスタンバイエージェントが失敗しました	A standby agent on <i>cluster_name</i> has left the cluster, but the coordinator has detected that the standby database is still running.
クラスター* <i>cluster_name</i> *のスタンバイデータベースが失敗しました	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。

次のページに続く

表 1 – 前のページからの続き

件名	説明
スタンバイエージェントはクラスタ* cluster_name *のデータベースに到達できません	スタンバイエージェントがデータベース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。
クラスター* cluster_name *は3つのノードより下にドロップしました	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。
クラスター* cluster_name *のサブセットがマスターから切断されました	このノードは、クラスターの大部分* cluster_name 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。 node_address *
クラスター* cluster_name *でプロモーションが開始されました。	クラスター* cluster_name *でスタンバイのプロモーションが開始されました。
クラスター* cluster_name *の監視エラー	Witness running at node_address has left the cluster.
クラスター* cluster_name *のアイドルエージェント障害。	Idle agent running at node_address has left the cluster.
クラスタ* cluster_name *のネットワークから分離された1つ以上のノード	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:* node_name *
ノードはクラスター* cluster_name *のネットワークから分離されなくなりました。	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。
スタンバイエージェントが昇格しようとしたが、マスターDBはまだ実行中です	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、マスターDBが* node_address *でまだ実行されていることを検出しました。これは通常、マスター EFM エージェントが終了したことを示します。フェイルオーバーは発生していません。
スタンバイエージェントが昇格を開始しましたが、マスターが再参加しました。	スタンバイ EFM エージェントは自身の昇格を開始しましたが、マスターエージェントがクラスターに再参加している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。
スタンバイエージェントは昇格しようとしたが、マスターDBを確認できませんでした	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、マスターDBが* node_address *でまだ実行されているかどうかを検出できませんでした。フェイルオーバーは発生していません。
スタンバイエージェントは昇格しようとしたが、VIPはまだ割り当てられているようです	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(* VIP_address *)がまだ別のノードに割り当てられているように見えるため、できませんでした。このような状況で昇格すると、データが破損する可能性があります。フェイルオーバーは発生していません。

次のページに続く

表 1 – 前のページからの続き

件名	説明
スタンバイエージェントが昇格しようとしたが、孤立しているようです	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が、既知のサーバー (* server_address *) に到達できなかったためできませんでした。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェイルオーバーは発生していません。
フェールオーバーは発生していません	エージェントは、マスターデータベースがクラスター* cluster_name *で使用できなくなったことを検出しましたが、フェールオーバーに使用できるスタンバイノードがありません。
クラスター* cluster_name *で潜在的な手動フェールオーバーが必要です。	クラスター* cluster_name *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ールオーバーが無効になっているため、手動による介入が必要です。
クラスター* cluster_name *でフェールオーバーが完了しました	クラスター* cluster_name *でフェールオーバーが完了しました。
クラスター* cluster_name *のロックファイルが削除されました	クラスター* cluster_name のロックファイルは、ノード node_address の path_name *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。
クラスター* cluster_name *の recovery.conf ファイルがマスターノードで見つかりました	クラスター* cluster_name の recovery.conf ファイルは、マスターノード node_address の path_name *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。
リカバリ設定で recovery_target_timeline が最新に設定されていません	recovery_target_timeline パラメーターは、リカバリ設定で最新に設定されていません。スタンバイサーバーは、新しいマスター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。
recovery.conf で指定された trigger_file パスは書き込み可能ではありません	recovery.conf ファイルの trigger_file パラメータに指定されたパスは、* db_service_owner *ユーザーによって書き込み可能ではありません。フェールオーバーマネージャーは、必要に応じてデータベースを昇格できません。
クラスター* cluster_name *のプロモーションは発生していません	プロモーションが試行されましたが、すでにプロモーション中のノードがあります:* ip_address *。
クラスター* cluster_name *のフェールオーバー後にスタンバイが再構成されない	このノードの auto.reconfigure プロパティは false に設定されています。ノードは、フェールオーバー後に新しいマスターノードに従うように再構成されていません。
クラスター* cluster_name *の再生を再開できませんでした	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:* error_decription *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。

次のページに続く

表 1 – 前のページからの続き

件名	説明
スタンバイ* <code>standby_id</code> *の再生を再開できませんでした。	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:* <code>error_message</code> *。
データベースのタイムアウト値で起こりうる問題	<code>remote.timeout</code> 値 (* <code>value</code>) は <code>local.timeout</code> 値 (<code>value</code> *) よりも大きいです。ローカルデータベースの応答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する可能性があり、フェイルオーバー保護がなくなります。
クラスター* <code>cluster_name</code> *でプロモーションに使用できるスタンバイはありません	クラスター内のスタンバイノードの現在の数は、最小数* <code>number</code> *に低下しています。別のスタンバイノードを追加するか、プロモーション可能にしない限り、フェイルオーバーはありません。
クラスター* <code>cluster_name</code> *で使用可能な同期スタンバイが不足しています。	クラスター内の同期スタンバイノードの数が* <code>count</code> *に低下しました。十分な同期スタンバイノードが追加されるまで、マスターに対するすべての書き込みクエリはブロックされます。
クラスター* <code>cluster_name</code> *の同期レプリケーションは無効になっています。	クラスター内の同期スタンバイノードの数が* <code>count</code> *を下回りました。マスターは同期複製モードから解除されました。
データベース構成を再ロードできませんでした。	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:* <code>error_message</code> *。
クラスター* <code>cluster_name</code> *のカスタムモニタータイムアウト	次のカスタム監視スクリプトがタイムアウトしました:* <code>script_name</code> *
クラスター* <code>cluster_name</code> *のカスタムモニター “セーフモード” エラー	次のカスタムモニタースクリプトは失敗しましたが、“セーフモード” で実行されています:* <code>script_name</code> 。出力: <code>script_results</code> *

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
フェールオーバーマネージャーは、必要に応じてデータベースを昇格できません。	
スタンバイデータベースは再起動しましたが、EFM は接続できません	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。
マスターはクラスター* cluster_name * のローカルデータベースに ping できません	マスターエージェントは、* 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す。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、マスターは VIP を解放したり、recovery.conf ファイルを作成したりしません。データベースの監視を再開するために再開コマンドが実行されるまで、マスターエージェントはアイドル状態になります。
フェンシングスクリプトエラー	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。
プロモーション後スクリプトが失敗しました	プロモーション後スクリプト* script_name の実行に失敗しました。終了値: exit_code 結果: script_results *
リモートポストプロモーションスクリプトが失敗しました	リモートポストプロモーションスクリプト* script_name * を正常に実行できませんでした 終了値:* exit_code * 結果:* script_results * ノード:* node_address *
リモート事前プロモーションスクリプトが失敗しました	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 *
データベース障害後スクリプトエラー	ポストデータベースエラー* script_name を正常に実行できませんでした。終了値: exit_code 結果: script_results *

次のページに続く

表 2 – 前のページからの続き

件名	説明
エージェントがスクリプトエラーを再開しました	エージェント再開スクリプト* <code>script_name</code> は正常に実行できませんでした。結果: <code>script_results</code> *
マスター分離スクリプトが失敗しました	マスター分離スクリプト* <code>script_name</code> は正常に実行できませんでした。終了値: <code>exit_code</code> 結果: <code>script_results</code> *
スタンバイを昇格できませんでした	トリガーファイル* <code>file_name</code> をノードで作成できませんでした。スタンバイを昇格できませんでした。エラーの詳細: <code>message_details</code> *
クラスター* <code>cluster_name</code> の <code>node_address</code> *で <code>recovery.conf</code> ファイルを作成中にエラーが発生しました	昇格中にマスターノード* <code>node_address</code> で <code>recovery.conf</code> ファイルを作成中にエラーが発生しました。昇格は継続しましたが、古いマスター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: <code>message_details</code> *
クラスター* <code>cluster_name</code> *で予期しないエラーが発生しました	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:* <code>error_details</code> *
クラスター* <code>cluster_name</code> *のために隔離されているマスターデータベース	マスターデータベースは、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、* <code>ip_address</code> *のマスターエージェントに、残りのフェールオーバーマネージャークラスターがスタンバイを昇格するときに2つのマスターを防ぐために <code>master</code> データベースをフェンスするように指示しています。
分離された <code>master</code> データベースのシャットダウン。	分離された <code>master</code> 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。
クラスター* <code>cluster_name</code> *のために隔離されているマスターデータベース	マスターデータベースは、クラスターの大部分から分離されています。マスター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、複数のマスターデータベースを回避するために自身を分離しています。
ノード* <code>node_address</code> *に <code>VIP</code> を割り当てることができませんでした	フェールオーバーマネージャは、何らかの理由で <code>VIP</code> アドレスを割り当てることができませんでした。
クラスター* <code>cluster_name</code> の <code>master_or_standby</code> *データベース障害	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。
クラスター* <code>cluster_name</code> *のエージェントがタイムアウトしています	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 <code>ping</code> 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に、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。
クラスター* <code>cluster_name</code> *のタイムアウトが再開しました	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
クラスター* <code>cluster_name</code> *の内部状態の不一致	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、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発生す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。
フェールオーバーは発生していません	エージェントがマスターデータベースを検出しました クラスター* <code>cluster_name</code> *では使用できなくなりましたが、フェールオーバーに使用できるスタンバイノードが十分ではありません。
Database in wrong state on <code>node_address</code>	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 <code>node_address</code>	マスターエージェントは、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。
クラスター* <code>cluster_name</code> *のデータベース接続エラー	このノードは、* <code>node_address</code> *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、フェールオーバーが正しく機能しない場合があります。
クラスター* <code>cluster_name</code> *のスタンバイカスタムモニターエラー	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:* <code>script_name</code> * スクリプト出力:* <code>script_results</code> *
クラスター* <code>cluster_name</code> *のマスターカスタムモニターエラー	次のカスタム監視スクリプトは、マスター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:* <code>script_name</code> * スクリプト出力:* <code>script_results</code> *
リカバリ設定で指定された* <code>trigger_file</code> *パスは書き込み不可です。	リカバリ設定でトリガーファイルパラメータに指定されたパスは、* <code>cluster_name</code> *ユーザーによって書き込み可能ではありません。

次のページに続く

表 2 – 前のページからの続き

件名	説明
<i>property_name</i> set to true for master node	The <i>property_name</i> property has been set to true for this cluster. Stopping the master agent without stopping the entire cluster will be treated by the rest of the cluster as an immediate master agent failure. If maintenance is required on the master database, shut down the master agent and wait for a notification from the remaining nodes that failover will not happen.
ロードバランサー接続スクリプトエラー	ロードバランサー接続スクリプト* <i>script_name</i> の実行に失敗しました。終了値: <i>exit_code</i> 結果: <i>script_results</i> *
ロードバランサーデタッチスクリプトエラー	ロードバランサデタッチスクリプト* <i>script_name</i> の実行に失敗しました。終了値: <i>exit_code</i> 結果: <i>script_results</i> *

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

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

フェールオーバーマネージャーは、非常に限定された限定的なフェールオーバーシナリオをサポートします。フェイルオーバーが発生する可能性があります。

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

フェールオーバーマネージャーは、フェールオーバーマネージャーでフェールオーバー条件を監視および検出するが、スタンバイへの自動フェールオーバーを実行しない場合に、`* no ** auto - failover *`モードもサポートします。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、`auto.failover`を設定します パラメータを `false` に。

フェールオーバーマネージャーは、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マスター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 マスターデータベースがダウンしています

Master データベースノードで実行されているエージェントが Master データベースの障害を検出すると、Failover Manager は障害を確認するプロセスを開始します。

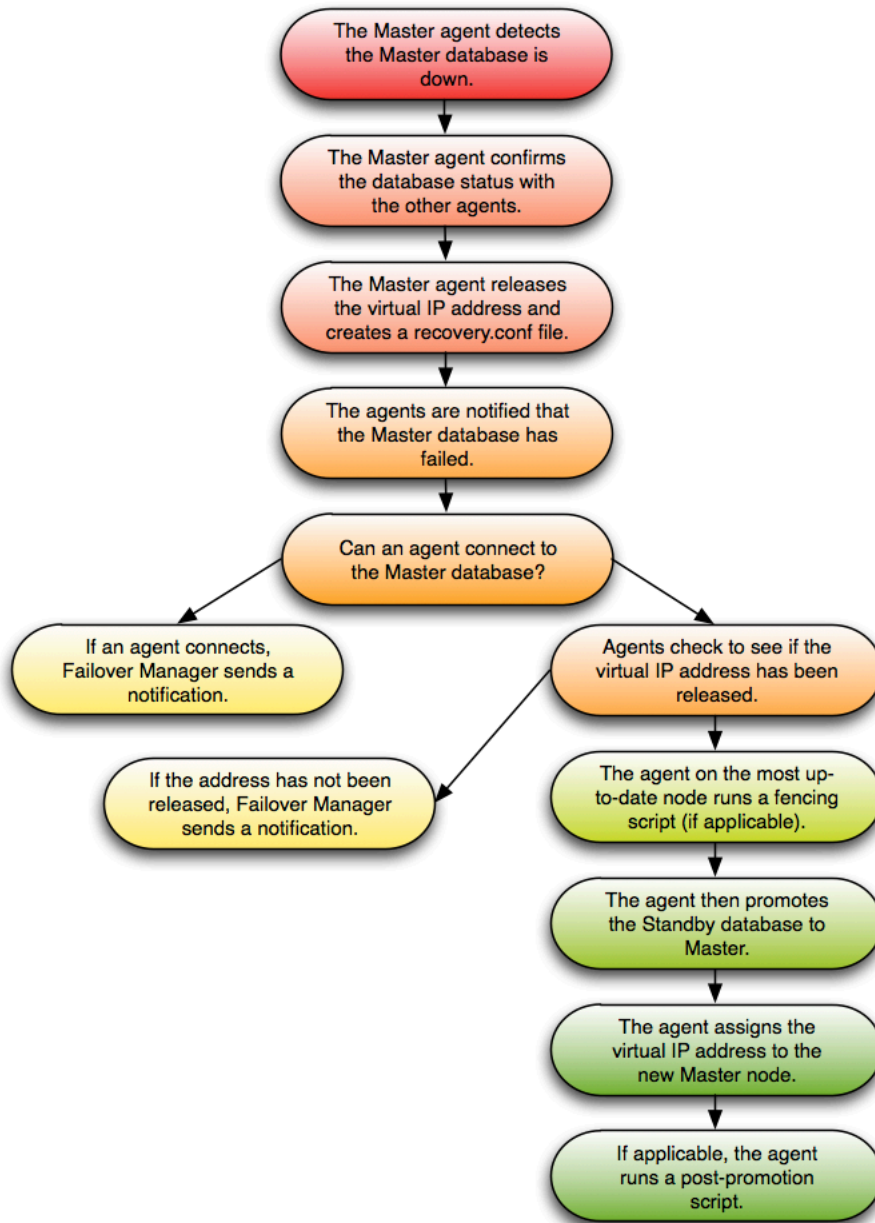


図 1 マスターデータベースの障害の確認。

Master ノードのエージェントが Master 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は Master データベースへの直接接続を試みます。エージェントがデータベースに接続できる場合、フェールオーバーマネージャーはマスター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、マスターエージェント

はデータベース障害を宣言し、VIP を解放します（該当する場合）。

エージェントが仮想 IP アドレスまたはデータベースサーバーに到達できない場合、フェールオーバーマネージャーはフェー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マスター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マスター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. 元のマスターノード上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. 元のマスターノードで `efm resume` コマンドを呼び出します。

ノードをマスターの役割に戻す

ノードをスタンバイとしてクラスターに戻した後、ノードをマスター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は、`efm allow-node` コマンドを使用して、ノードのフェールオーバー優先度を 1 に設定します。
2. *efm Promote -switchover* コマンドを呼び出します ノードをマスターノードの元の役割に昇格させる。

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。

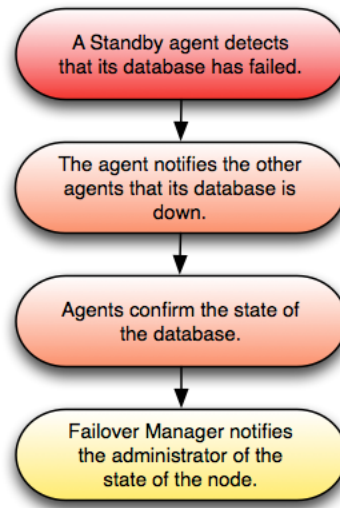


図 2 スタンバイデータベースの障害の確認。

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 マスターエージェントの終了またはノードの失敗

フェールオーバーマネージャーマスター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。

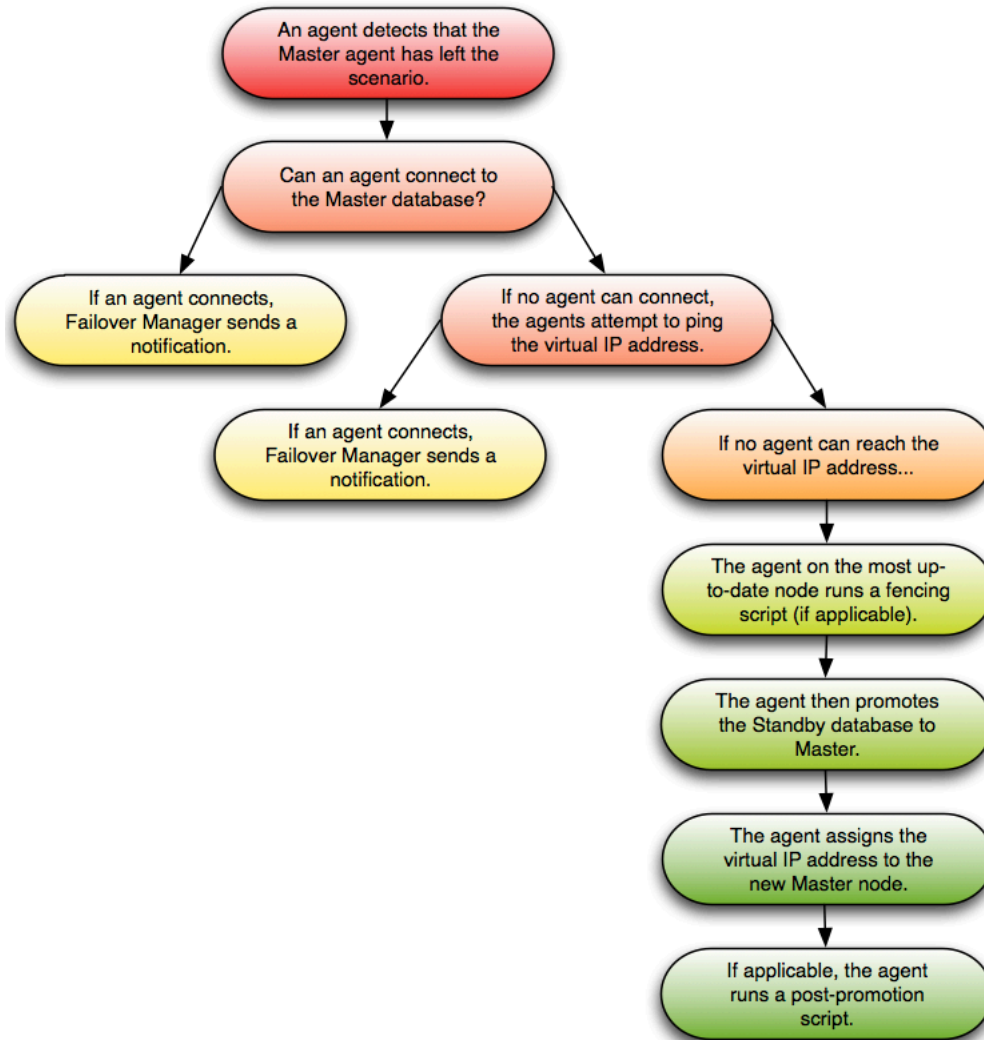


図3 マスターエージェントの障害の確認。

マスター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マスター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、エージェントはマスター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 アドレスに ping を試行して、解放されたかどうかを判断します。

エージェントが仮想 IP アドレスまたはデータベースサーバーに到達できない場合、フェールオーバーマネージャーマスターエージェントはフェールオーバープロセスを開始します。最新ノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをマスター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、エージェントはポストプロモーション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マスターから複製するように構成されます。

マスター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、マスター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。フェールオーバーマネージャーは、クラスターの残りのノードで前述の手順を実行します。

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

1. 元のマスターノードを再起動します。
2. 元のマスターデータベースをスタンバイノードとして起動します。
3. 元のマスター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。

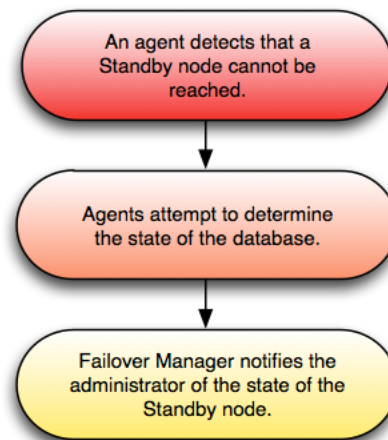


図 4 スタンバイエージェントの障害。

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すると、フェールオーバーマネージャーは適切な通知を管理者に送信します。

マスターが 1 つとスタンバイが 1 つだけ残っている場合、マスター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。マスターデータベースに障害が発生した場合、マスター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。

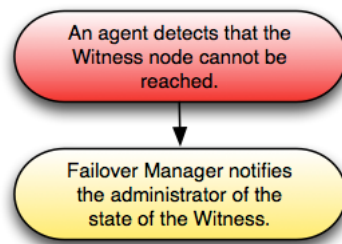


図 5 献身的な証人の失敗の確認

エージェントが監視ノードに到達できないことを検出すると、フェールオーバーマネージャーは管理者に監視の状態を通知します。

注: マスターとスタンバイが 1 つしか残っていない場合、マスター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。マスターデータベースに障害が発生した場合、マスター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。

11.6 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。

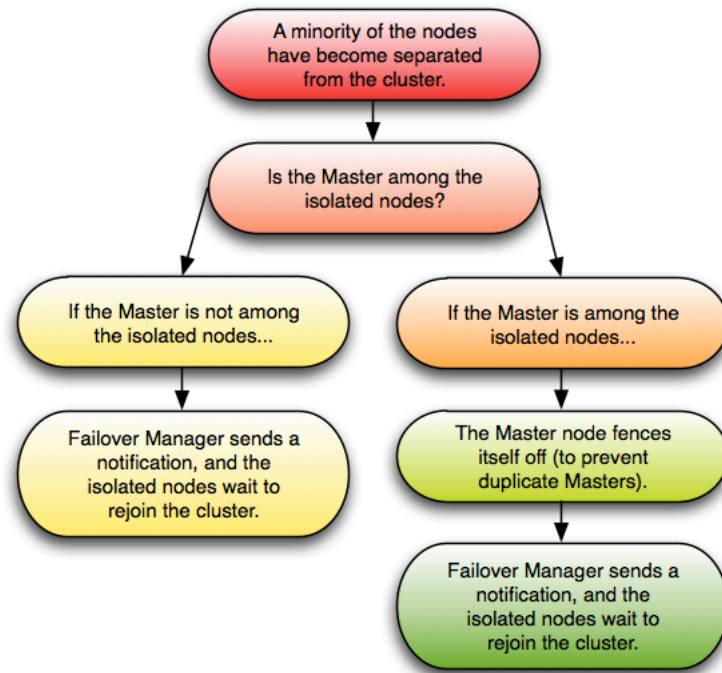


図 6 クラスターのメンバーが分離された場合

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、マスターノードが分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、マスターはクラスターから自身をフェンスしますが、スタンバイノード（クラスターの過半数内から）はそれを置き換えるために昇格されます。 `auto.reconfigure` が `false` に設定されていない限り、他のスタンバイノードは新しいマスターから複製するように構成されます。

フェールオーバーマネージャーは管理者に通知し、分離されたノードは可能な場合はクラスターに再参加します。ノードがクラスターに再参加すると、フェ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 は、Failover Manager クラスター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。既存のクラスターをアップグレードするには、以下を行う必要があります。

1. クラスターの各ノードに Failover Manager 3.8 をインストールします。フェールオーバーマネージャーのインストールの詳細については、[フェールオーバーマネージャーのインストールを参照してください](#)。
2. Failover Manager をインストールしたら、`efm upgrade-conf` ユーティリティを呼び出して、Failover Manager 3.8 用の `.properties` および `.nodes` ファイルを作成します。Failover Manager インストーラーは、アップグレードユーティリティ（*`efm upgrade-conf`*）から ‘3’ へ。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` の `upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、Failover Manager で使用するためにパラメーター値を新しい構成ファイルにコピーします。ユーティリティは、設定ファイルの更新されたコピーを `/etc/edb/efm-3.8` ディレクトリに保存します。

3. 新しい設定を指定して、EFM 3.8 の `.properties` および `.nodes` ファイルを変更します。

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-3.8` 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、[クラスタープロパティファイル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、古いフェールオーバーマネージャークラスターを停止します。たとえば、次のコマンドを使用してバージョン 3.7 クラスターを停止できます。

```
/usr/efm-3.7/bin/efm stop-cluster efm
```

- 新しい *Failover Manager* サービスを開始します（‘0’）クラスターの各ノード。

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

```
# efm upgrade-conf example
Checking directory /etc/edb/efm-3.7
Processing example.properties file
The following properties were added in addition to those in previous installed version:
    db.config.dir
Checking directory /etc/edb/efm-3.7
Processing example.nodes file
Upgrade of files is finished. The owner and group for properties and nodes files have
↳ been set as 'efm'.
```

sudo なしでフェールオーバーマネージャークラスター構成を使用している場合、‘0’-source‘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。ディレクトリが設定のデフォルトディレクトリではない場合、アップグレードされたファイルは upgrade-conf コマンドが呼び出されたディレクトリに作成されます。

注: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は、アップグレードを実行するときに、新しいフェールオーバーマネージャークラスター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。

12.1 Failover Manager のアンインストール

Failover Manager 3.8 にアップグレードした後、Yum 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、フェールオーバーマネージャー 3.7 と不要な依存関係を削除します。

```
yum remove edb-efm37
```

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。次の手順を使用して、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、フェールオーバーマネージャークラスターの各スタンバイノードでデータベースサーバーを更新する必要があります。次に、スイッチオーバーを実行し、スタンバイノードをフェールオーバーマネージャークラスター内のマスターの役割に昇格させます。次に、古いマスター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 エージェントを停止します。
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 エージェントを起動します。

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

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

更新が完了したら、`efm set-priority` を使用できます コマンドを使用して、古いマスターをスタンバイリストの先頭に追加し、スイッチオーバーしてクラスターを元の状態に戻します。

トラブルシューティング

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 ログファイルを確認してください<controlling_logging> “OutOfMemory” メッセージ。フェールオーバーマネージャーは、このプロパティで設定されたデフォルトのメモリ値で実行されます。

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

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

フェールオーバーマネージャーは OpenJDK でテストされています。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)
```

第 14

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

<https://www.postgresql.org/docs/12/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/12/static/client-authentication.html>

14.1 カスケード複製の限定サポート

フェールオーバーマネージャーはカスケードレプリケーションを完全にはサポートしていませんが、カスケードレプリケーションシナリオでは単純なフェールオーバーを限定的にサポートしています。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、マスターノードへの接続数（および処理オーバーヘッド）が削減されます。

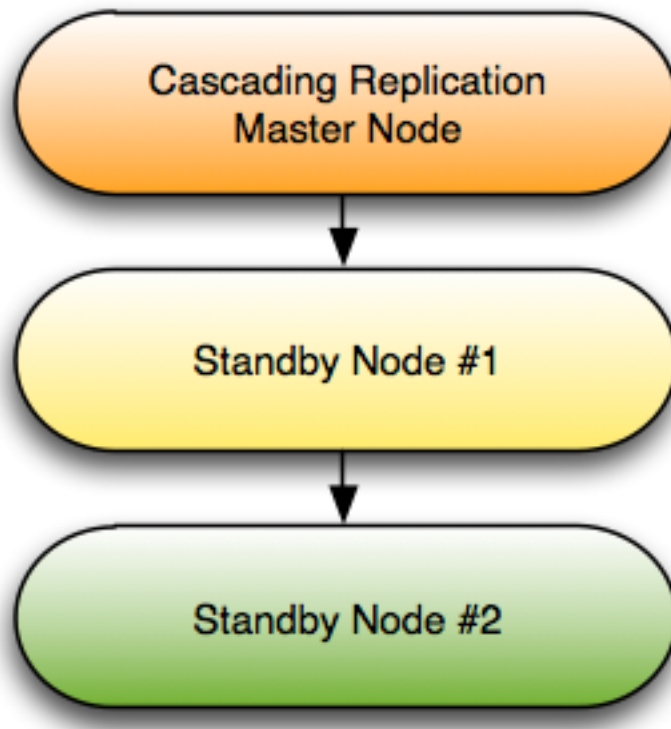


図 1 カスケード複製。

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/12/static/warm-standby.html#cascading-replication>

カスケードレプリケーションシナリオでフェールオーバーマネージャーを使用するには、クラスタープロパティファイルを変更して、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 がマスター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、新しいマスター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マスターに戻すと、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。

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、フェールオーバーマネージャークラスターの SSL 認証を有効にします。すべての接続クライアントは、クラスター内のデータベースサーバ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。

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

1. data ディレクトリに `server.crt` および `server.key` ファイルを配置します（下に **Advanced Server** のインストール）。機関によって署名された証明書を購入するか、独自の自己署名証明書を作成できます。自己署名証明書の作成については、次の PostgreSQL コアドキュメントを参照してください。

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

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

```
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. フェールオーバーマネージャークラスターの各ノードで `pg_hba.conf` ファイルを変更します 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. データに `server.crt` および `server.key` ファイルを配置した後 ディレクトリ、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 の信頼できる証明書ファイルに追加します。

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

`<alias_name>` 任意の文字列を指定できますが、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. *efm.properties* ファイルを変更します クラスタ内の各ノードで、 `jdbc.sslmode` プロパティを設定します。

結論

EDB Postgres Failover Manager ガイド

Copyright©2013-2020 EnterpriseDB Corporation。

無断複写・転載を禁じます。

EnterpriseDB®Corporation

34 Crosby Drive, Suite 201, Bedford, MA 01730, アメリカ

T +1 781 357 3390 F +1 978 467 1307 E

info@enterprisedb.com

www.enterprisedb.com

- EDB は、存在する場合、EDB Postgres 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その他の危険な入力を防ぐことはできません。
- 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, 23
application.name, 28
auto.allow.hosts, 27
auto.failover, 27
auto.reconfigure, 28
auto.resume.period, 30

bind.address, 23

Changing the priority of a Standby, 50

check.vip.before.promotion, 31
custom.monitor.interval, 35
custom.monitor.safe.mode, 35
custom.monitor.timeout, 35

db.bin, 20
db.config.dir, 21
db.database, 19
db.password.encrypted, 19
db.port, 19
db.recovery.dir, 20
db.reuse.connection.count, 27
db.service.name, 20
db.service.owner, 20
db.user, 19
Debian Installation, 15

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

Failover Manager サービスの制御, 69
Failover Manager のインストール, 13
file.log.enabled, 37
from.email, 22

is.witness, 23

jdbc.sslmode, 21

jgroups.loglevel, 37
jvm.options, 37

Linux Installation, 13
local.period, 24
local.timeout, 24
local.timeout.final, 24
lock.dir, 36
log.dir, 36

Master Agent Exits, 90
Master Database is Down, 87
Master Node Fails, 90
master.shutdown.as.failure, 25
minimum.standbys, 29
Multiple agents on a single node, 54

Node Becomes Isolated, 94
node.timeout, 24
notification.level, 22

pingServer, 26
pingServerCommand, 26
Postgres Enterprise Manager を使用したストリーミング
レプリケーションの監視, 62
promotable, 28
Promoting a node, 51

reconfigure.sync.master, 29
recovery.check.period, 30
remote.timeout, 24
restart.connection.timeout, 30
restore.command, 29

script.custom.monitor, 35
script.db.failure, 33
script.fence, 32
script.load.balancer.attach, 32
script.load.balancer.detach, 32
script.master.isolated, 33
script.notification.property, 22
script.post.promotion, 33
script.remote.post.promotion, 34
script.remote.pre.promotion, 34
script.resumed, 33
SLES Installation, 16
stable.nodes.file, 27
Standby Agent Exits, 92
Standby Database is Down, 89

Standby Node Fails, 92
Starting the EFM Cluster, 49
stop.failed.master, 25
stop.isolated.master, 25
Stopping a Cluster, 53
Stopping an agent, 52
sudo.command, 35
sudo.user.command, 35
syslog.enabled, 37
syslog.facility, 36
syslog.host, 36
syslog.port, 36
syslog.protocol, 36

Ubuntu Installation, 15
user.email, 22
Using the service utility, 69
Using the systemctl utility, 70

virtualIp, 30
virtualIp.interface, 30
virtualIp.prefix, 30
virtualIp.single, 30

Witness Agent Exits, 93
Witness Node Fails, 93

クラスタープロパティファイル, 18
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, 87
サポートされているプラットフォーム, 5

ストリーミングレプリケーションの設定, 101

チュートリアル-シンプルなフェイルオーバーマネージャークラスターの設定, 9

データベースパスワードの暗号化, 37

トラブルシューティング, 100

フェイルオーバーマネージャ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の概要, 3
フェイルオーバーマネージャの使用, 48
フェイルオーバーマネージャの設定, 18
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, 104
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の使用, 45

既存のクラスターのアップグレード, 96
結論, 106
新機能, 2
前提条件, 6
通知, 76