



# EDB

## **Failover Manager**

リリース 4.7

**EnterpriseDB Corporation**

2023 年 07 月 19 日



# 目次

|              |                                                                          |           |
|--------------|--------------------------------------------------------------------------|-----------|
| <b>第 1 章</b> | <b>Release notes</b>                                                     | <b>3</b>  |
| 1.1          | Version 4.7 . . . . .                                                    | 3         |
| 1.2          | Version 4.6 . . . . .                                                    | 4         |
| 1.3          | Version 4.5 . . . . .                                                    | 4         |
| 1.4          | Version 4.4 . . . . .                                                    | 5         |
| 1.5          | Version 4.3 . . . . .                                                    | 6         |
| 1.6          | Version 4.2 . . . . .                                                    | 6         |
| 1.7          | Version 4.1 . . . . .                                                    | 7         |
| 1.8          | Version 4.0 . . . . .                                                    | 7         |
| 1.8.1        | EFM プロパティの変更 . . . . .                                                   | 8         |
| <b>第 2 章</b> | <b>Supported platforms</b>                                               | <b>9</b>  |
| <b>第 3 章</b> | <b>Architecture</b>                                                      | <b>11</b> |
| <b>第 4 章</b> | <b>Choosing a deployment architecture</b>                                | <b>13</b> |
| 4.1          | Failover Manager with virtual IP . . . . .                               | 14        |
| 4.1.1        | VIP でのフェールオーバーマネージャーの使用 . . . . .                                        | 14        |
| 4.2          | Failover Manager with client connect failover . . . . .                  | 16        |
| 4.2.1        | クライアント接続フェールオーバーでのフェールオーバーマネージャーの使用 . . . . .                            | 17        |
| 4.3          | Failover Manager with EDB PgBouncer . . . . .                            | 18        |
| 4.3.1        | オンプレミスの EDB PgBouncer を使用したフェールオーバーマネージャー . . . . .                      | 18        |
| 4.3.2        | クラウド内の EDB PgBouncer を備えた Failover Manager . . . . .                     | 19        |
| 4.3.3        | PgBouncer でフェールオーバーマネージャーを使用する . . . . .                                 | 20        |
| 4.4          | Failover Manager with EDB Pgpool-II . . . . .                            | 25        |
| 4.4.1        | オンプレミスの EDB Pgpool-II を使用したフェールオーバーマネージャー . . . . .                      | 25        |
| 4.4.2        | クラウド内の EDB Pgpool-II を使用したフェールオーバーマネージャー . . . . .                       | 26        |
| 4.4.3        | EDB Pgpool-II でのフェールオーバーマネージャーの使用 . . . . .                              | 26        |
| <b>第 5 章</b> | <b>Installing Failover Manager on Linux</b>                              | <b>33</b> |
| 5.1          | Linux x86-64 (amd64) . . . . .                                           | 33        |
| 5.1.1        | Red Hat Enterprise Linux (RHEL) およびその派生物 . . . . .                       | 33        |
| 5.1.2        | SUSE Linux Enterprise (SLES) . . . . .                                   | 33        |
| 5.1.3        | Debian とその派生物 . . . . .                                                  | 33        |
| 5.2          | Linux Installing Failover Manager on Linux IBM Power (ppc64le) . . . . . | 34        |
| 5.2.1        | Red Hat Enterprise Linux (RHEL) およびその派生物 . . . . .                       | 34        |
| 5.2.2        | SUSE Linux Enterprise (SLES) . . . . .                                   | 34        |
| <b>第 6 章</b> | <b>Upgrading Failover Manager</b>                                        | <b>57</b> |
| 6.1          | Failover Manager のアンインストール . . . . .                                     | 58        |
| 6.2          | メンテナンスタスクの実行 . . . . .                                                   | 59        |

|               |                                                                     |            |
|---------------|---------------------------------------------------------------------|------------|
| <b>第 7 章</b>  | <b>Configuring streaming replication</b>                            | <b>61</b>  |
| 7.1           | カスケードレプリケーションの制限付きサポート                                              | 61         |
| <b>第 8 章</b>  | <b>Configuring Failover Manager</b>                                 | <b>65</b>  |
| 8.1           | The cluster properties file                                         | 65         |
| 8.1.1         | クラスタープロパティの指定                                                       | 66         |
| 8.1.2         | クラスタープロパティ                                                          | 70         |
| 8.2           | Encrypting your database password                                   | 104        |
| 8.2.1         | EFMPASS 環境変数の使用                                                     | 106        |
| 8.3           | The cluster members file                                            | 106        |
| 8.4           | Extending Failover Manager permissions                              | 107        |
| 8.4.1         | 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                                      | 109        |
| 8.5           | Using Failover Manager with virtual IP addresses                    | 110        |
| 8.5.1         | VIP のテスト                                                            | 111        |
| 8.6           | Configuring for Eager Failover                                      | 113        |
| 8.6.1         | Eager フェールオーバーを有効にする                                                | 113        |
| 8.6.2         | Eager Failover の無効化                                                 | 114        |
| 8.6.3         | Eager Failover モードでの Failover Manager のアップグレード                      | 114        |
| 8.6.4         | 重要な注意事項                                                             | 115        |
| <b>第 9 章</b>  | <b>Configuring SSL authentication on a Failover Manager cluster</b> | <b>117</b> |
| <b>第 10 章</b> | <b>Using Failover Manager</b>                                       | <b>119</b> |
| 10.1          | フェールオーバーマネージャークラスターの管理                                              | 119        |
| 10.1.1        | フェールオーバーマネージャークラスターの開始                                              | 120        |
| 10.1.2        | クラスターへのノードの追加                                                       | 120        |
| 10.1.3        | スタンバイの優先順位の変更                                                       | 121        |
| 10.1.4        | フェールオーバーマネージャークラスターノードの昇格                                           | 122        |
| 10.1.5        | Failover Manager エージェントの停止                                          | 123        |
| 10.1.6        | フェールオーバーマネージャークラスターの停止                                              | 124        |
| 10.1.7        | クラスターからのノードの削除                                                      | 124        |
| 10.2          | 単一のノードで複数のエージェントを実行する                                               | 124        |
| 10.2.1        | RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x                  | 126        |
| <b>第 11 章</b> | <b>Using the efm utility</b>                                        | <b>129</b> |
| 11.1          | efm allow-node                                                      | 129        |
| 11.2          | efm disallow-node                                                   | 129        |
| 11.3          | efm cluster-status                                                  | 129        |
| 11.4          | efm cluster-status-json                                             | 130        |
| 11.5          | efm encrypt                                                         | 131        |
| 11.6          | efm プロモート                                                           | 131        |
| 11.7          | efm 履歴書                                                             | 132        |
| 11.8          | efm set-priority                                                    | 132        |
| 11.9          | efm stop-cluster                                                    | 132        |
| 11.10         | efm upgrade-conf                                                    | 132        |
| 11.11         | efm node-status-json                                                | 133        |
| 11.12         | efm -help                                                           | 133        |

|                                                                                            |            |
|--------------------------------------------------------------------------------------------|------------|
| <b>第 12 章 Monitoring a Failover Manager cluster</b>                                        | <b>135</b> |
| 12.1 クラスターステータスレポートの確認 . . . . .                                                           | 135        |
| 12.1.1 終了コード . . . . .                                                                     | 137        |
| 12.2 Postgres Enterprise Manager を使用したストリーミングストリーミングレプリケーションの監視 . . . . .                  | 137        |
| <b>第 13 章 Controlling the Failover Manager service</b>                                     | <b>141</b> |
| 13.1 RHEL/CentOS 7.x および RHEL/Rocky Linux/AlmaLinux 8.x で systemctl ユーティリティを使用する . . . . . | 141        |
| <b>第 14 章 Controlling logging</b>                                                          | <b>143</b> |
| 14.1 syslog ログファイルエントリの有効化 . . . . .                                                       | 144        |
| <b>第 15 章 Notifications</b>                                                                | <b>147</b> |
| <b>第 16 章 Supported failover and failure scenarios</b>                                     | <b>159</b> |
| 16.1 プライマリデータベースがダウンしています . . . . .                                                        | 159        |
| 16.2 ノードをクラスターに戻す . . . . .                                                                | 161        |
| 16.3 ノードをプライマリの役割に戻します . . . . .                                                           | 161        |
| 16.4 スタンバイデータベースがダウンしています . . . . .                                                        | 161        |
| 16.5 プライマリエージェントが終了するか、ノードに障害が発生しました . . . . .                                             | 161        |
| 16.6 スタンバイエージェントが終了するか、ノードに障害が発生します . . . . .                                              | 164        |
| 16.7 専用の証人エージェントが終了/ノードに障害が発生する . . . . .                                                  | 166        |
| 16.8 ノードはクラスターから分離されます . . . . .                                                           | 166        |
| <b>第 17 章 Troubleshooting</b>                                                              | <b>169</b> |
| 17.1 認証ファイルが見つかりません。ローカルエージェントは実行されていますか? . . . . .                                        | 169        |
| 17.2 このコマンドを実行する権限がありません。ユーザー「<os user>」`efm` グループのメンバーではありません。 . . . .                    | 169        |
| 17.3 通知。予期しないエラーメッセージ . . . . .                                                            | 169        |
| 17.4 OpenJDK バージョンの確認 . . . . .                                                            | 170        |
| <b>第 18 章 Creating a Failover Manager cluster</b>                                          | <b>171</b> |
| 18.1 スイッチオーバーを実行する . . . . .                                                               | 174        |
| 18.2 オンラインヘルプにアクセスする . . . . .                                                             | 174        |



Failover Manager (EFM) は、Postgres データベースクラスターを管理するためのツールであり、ストリーミングレプリケーションを使用してプライマリ-スタンバイ展開アーキテクチャの高可用性を実現します。フェールオーバーマネージャーは、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 プライマリデータベースノードからスタンバイデータベースノードへの自動フェールオーバーを提供します。PostgreSQL または EDB Postgres Advanced Server で Failover Manager を使用できます。



# 第1章 Release notes

Failover Manager のドキュメントには、マイナーリリースとパッチを含む Failover Manager 4 の最新バージョンが記載されています。これらのリリースノートには、各リリースの新機能が記載されています。マイナーリリースまたはパッチリリースで導入された新機能については、その機能を導入したリリースに関するコンテンツにインジケータもあります。

| Version            | Release Date |
|--------------------|--------------|
| <i>Version 4.7</i> | 2023 Jun 20  |
| <i>Version 4.6</i> | 2023 Feb 14  |
| <i>Version 4.5</i> | 2022 Aug 30  |
| <i>Version 4.4</i> | 2022 Jan 5   |
| <i>Version 4.3</i> | 2021 Dec 18  |
| <i>Version 4.2</i> | 2021 Apr 19  |
| <i>Version 4.1</i> | 2021 Dec 11  |
| <i>Version 4.0</i> | 2021 Sep 2   |

## 1.1 Version 4.7

EFM 4.7 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type        | Description                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------|
| Enhancement | フェールオーバーマネージャーは、新しいプライマリに従うように再構成するときに、スタンバイで pg_wal のバックアップを 1 つだけ保持します。[サポートチケット：# 1277862、# 91506] |
| Bug Fix     | フェールオーバーマネージャーの起動スクリプトが、小数にコンマを使用するロケールで機能しない問題を修正しました。たとえば、「1,8」。【サポートチケット #89616】                   |
| Bug Fix     | フェールオーバーマネージャーがフェールオーバーマネージャー 4.6 の「efm」コマンドのリプレイ値にスタンバイの WAL 受信値を誤って使用した問題を修正しました。                   |

## 1.2 Version 4.6

EFM 4.6 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type         | Description                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Security Fix | Failover Manager は、セキュリティの脆弱性 CVE-2018-10936 に対処するために、PostgreSQL JDBC Driver バージョン 42.4.3 を使用するようにアップグレードされました。注：フェールオーバーマネージャーはこの脆弱性の影響を受けませんが、影響を受けるバージョンのライブラリは含まれなくなりました。   |
| Enhancement  | フェールオーバーマネージャーは、カスタムの場所でのエージェント pid ファイルの作成をサポートするようになりました。[サポートチケット #88686 #89016]                                                                                             |
| Bug Fix      | まれに、プライマリが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は「プライマリノードが見つかりません」通知を送信し続けません。【サポートチケット #87106】                                                                                            |
| Bug Fix      | フェールオーバーまたは過負荷のノードが停止せずに、フェールオーバーマネージャーエージェントの実行が妨げられるなど、ノードが過負荷になっている場合、クラスターの内部状態が矛盾し、将来のフェールオーバーが妨げられる可能性があります。この場合、このようなイベントの後に内部状態が修正されるように処理されるようになりました。【サポートチケット #87975】 |
| Bug Fix      | フェールオーバーマネージャーのプライマリエージェントは、データベース障害とノード分離が同時に発生するまれなケースでも、VIP の解放に失敗しなくなりました。【サポートチケット #79280】                                                                                 |
| Bug Fix      | クラスター内のスタンバイノードに障害が発生した場合の誤解を招く通知の件名が修正されました。                                                                                                                                   |
| Bug Fix      | フェールオーバーマネージャーは、クラスター内の 9 つ以上の同期スタンバイをサポートするようになりました。                                                                                                                           |

## 1.3 Version 4.5

EFM 4.5 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type         | Description                                                                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Security Fix | Failover Manager は、セキュリティの脆弱性 CVE-2022-31197 に対処するために、PostgreSQL JDBC Driver バージョン 42.4.1 を使用するようにアップグレードされました。注：フェールオーバーマネージャーはこの脆弱性の影響を受けませんが、影響を受けるバージョンのライブラリは含まれなくなりました。 |
| Enhancement  | プライマリノードがクラスターから分離されて再参加すると、他のノードが昇格していない場合、フェールオーバーマネージャーは分離されたデータベースを再起動し、クラスター内のプライマリノードとして監視を再開します。【サポートチケット #937960、#968619、#81770、#81894】                               |
| Bug Fix      | フェールオーバーマネージャーは、LSN 値が誤って比較されるというまれな場合に、昇格するスタンバイを適切に選択するようになりました。【サポートチケット #83186】                                                                                           |
| Bug Fix      | フェールオーバーマネージャーは、クラスターステータスの `efm` コマンドを実行するときに、プロパティファイル内の余分な空白を処理するようになりました。                                                                                                 |
| Bug Fix      | フェールオーバーマネージャーは、プライマリエージェントの実行が遅い場合に、昇格に失敗しなくなりました。【サポートチケット #79847】                                                                                                          |
| Bug Fix      | フェールオーバーマネージャーは、`efm` コマンドを使用して手動でプロモート可能にした後に再起動されたスタンバイの <code>promotable=false</code> 値に従います。【サポートチケット #83192】                                                             |
| Bug Fix      | 物理レプリケーションスロットを使用する場合、フェールオーバーマネージャーは、`xmin` 値が設定されている場合、LSN 値を進めるときにスタンバイのスロットをドロップして再作成します。【サポートチケット #83910】                                                                |
| Bug Fix      | Systemd ステータスは、Failover Manager エージェントを停止するために使用された後、「failed」ではなく「inactive」ステータスを表示します。                                                                                       |

## 1.4 Version 4.4

EFM 4.4 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type         | Description                                                                               |
|--------------|-------------------------------------------------------------------------------------------|
| Security Fix | Failover Manager は log4j バージョン 2.17.1 を使用するようにアップグレードされ、セキュリティの脆弱性 CVE-2021-44832 に対処します。 |

## 1.5 Version 4.3

EFM 4.3 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type         | Description                                                                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Security Fix | Failover Manager は、log4j バージョン 2.17.0 を使用してセキュリティの脆弱性 CVE-2021-44228、CVE-2021-45046、および CVE-2021-45105 に対処するようにアップグレードされました。CVE-2021-44228 は CVSS スコア 10.0 の重大な脆弱性と見なされているため、この更新プログラムをできるだけ早く適用する必要があります。 |

## 1.6 Version 4.2

EFM 4.2 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type        | Description                                                                                                                                                                                       |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enhancement | プライマリリソースを、監視されていない正常なノードから完全に動作可能な正常なノードに移動するための、新しい Eager Failover セットアップが追加されました。この設定では、プライマリ EFM エージェントが停止したときにフェールオーバーが実行されます。Eager Failover がなければ、プライマリ Postgres インスタンスは使用可能なままですが、管理されません。 |
| Enhancement | EDB Postgres 高可用性および水平方向の読み取りスケーリングアーキテクチャは、PgPool-II 4.2 で認定されており、非 sudo モードでサポートされています。                                                                                                         |
| Enhancement | スタンバイのプロモーション優先度をプライマリごとに定義できるようになりました。[サポートチケット # 1092697]                                                                                                                                       |
| Enhancement | EFM 構成で stop-cluster コマンドを無効にできるようになりました。                                                                                                                                                         |
| Enhancement | プライマリエージェントがクラスターを離れても、プライマリデータベースがまだ実行されている場合、フェールオーバーマネージャーはフェールオーバーではなく警告通知を送信します。これはフェールオーバー保護がないことを意味するため、バージョン 4.2 の拡張機能では、プライマリエージェントがクラスターに再参加するまで警告通知が繰り返されます。                           |
| Bug Fix     | アーカイブで構成されたスタンバイは、新しいプライマリに従うように適切に再構成されます。[サポートチケット # 1127627、# 1182654]                                                                                                                         |
| Bug Fix     | プライマリ EFM エージェントに障害が発生したが Postgres がまだ実行されている場合、新しいオプションは LoadBalancer デタッチスクリプトを無効にすることができます。[サポートチケット # 1096007]                                                                                |
| Bug Fix     | エージェントの起動時に pg_wal/pg_xlog の親ディレクトリが書き込み可能かどうかを確認する検証を追加しました。EFM がそのバックアップを作成し、後で復元できるように、親ディレクトリは書き込み可能である必要があります。新しい検証が失敗すると、エラーがログに記録され、エージェントは起動を拒否します。                                       |

新しいプロパティの詳細については、[The cluster properties file](#) を参照してください。

## 1.7 Version 4.1

EFM 4.1 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type        | Description                                                                       |
|-------------|-----------------------------------------------------------------------------------|
| Enhancement | EFM を使用した HA アーキテクチャと、Azure Network Load Balancer を使用した Pgpool デプロイのサポートが追加されました。 |
| Enhancement | スタンバイエージェントは、ローカルデータベースの接続が回復すると、ヘルスモニタリングの再開を試みます。                               |
| Enhancement | 新しいオプションは、同期スタンバイがクラスターに追加されるか、クラスターに再参加すると、 <code>num_sync</code> を自動的に増やします。    |
| Bug Fix     | 同期スタンバイが追加されると、 <code>num_sync</code> を増やすことができます。[サポートチケット # 1092053]            |
| Bug Fix     | 特定のプロパティが存在しないとエージェントを起動できません。[サポートチケット # 1087362]                                |
| Bug Fix     | セキュリティの脆弱性のため、JDBC ドライバーをアップグレードしました。[サポートチケット # 1052383]                         |
| Bug Fix     | 「arping」は、非 sudo モードで VIP を取得した後に呼び出されるようになりました。[サポートチケット # 1039625]              |

新しいプロパティの詳細については、[The cluster properties file](#) を参照してください。

## 1.8 Version 4.0

EFM 4.0 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

| Type        | Description                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Enhancement | プロパティの名前が変更されました。詳細については、次の表を参照してください。アップグレードユーティリティを使用してフェールオーバーマネージャインストールをアップグレードすると、それらは自動的に更新されます。                           |
| Enhancement | 暗号化されたデータベースパスワードのプロパティ値は、新しいハッシュアルゴリズムを満たすように更新する必要があります。アップグレードツールは、必要な変更を加えます。プロパティファイルを手動で作成する場合、代わりにパスワードを暗号化する標準の手順を使用できます。 |
| Enhancement | データベースのパスワードの暗号化が改善されました。エージェント間の通信では暗号化も有効になりました。                                                                                |
| Enhancement | 新しいプライマリを選択しているときにスタンバイサーバーが停止なくなりました。この機能強化により、プロモーションプロセスが大幅にスピードアップされます。                                                       |

次のページに続く

表 9 – 前のページからの続き

| Type        | Description                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enhancement | コミュニティの命名ガイドラインと一貫性を保つために、Failover Manager の製品とドキュメントでは、「マスター」という用語が「プライマリ」に置き換えられました。upgrade-conf ツールは、アップグレード後に影響を受けるプロパティの名前を変更するタスクを処理します。script.load.balancer.attach、script.load.balancer.detach などのロードバランサースクリプトは、文字 'm' ではなく 'p' を引数として受け入れるようになりました。 |
| Enhancement | プロモーション後にスタンバイのリスタートを遅延するサポートが追加されました。スタンバイの再起動をずらすことで、可用性を高めることができます。                                                                                                                                                                                       |
| Bug Fix     | /var/run および /var/lock の tmpfs マウントポイントを使用するユーザーを支援するためにドキュメントが更新されました [サポートチケット # 977437]                                                                                                                                                                   |
| Bug Fix     | IDLE エージェントは、他のエージェントが自分のデータベースに到達できると、モニタリングを再開しようとします。[サポートチケット # 1001533]                                                                                                                                                                                 |

### 1.8.1 EFM プロパティの変更

| Former name                     | New name                    |
|---------------------------------|-----------------------------|
| pingServerCommand               | ping.server.command         |
| stop.isolated.master            | top.isolated.primary        |
| stop.failed.master              | stop.failed.primary         |
| mas-<br>ter.shutdown.as.failure | primary.shutdown.as.failure |
| reconfigure.sync.master         | reconfigure.sync.primary    |
| script.master.isolated          | script.primary.isolated     |

ロードバランサースクリプトを使用している場合、' %t ' プレースホルダーは、プライマリの m ではなく p に置き換えられます。

## 第2章 Supported platforms

フェールオーバーマネージャーでサポートされているプラットフォームとバージョンの詳細については、[Platform Compatibility](#) を参照してください。

---

**注釈:** CPU アーキテクチャの混合モードがサポートされており、プライマリノードとスタンバイノードが Linux on IBM Power 上にあり、監視ノードが x86-64 Linux 上にあります。

---



## 第3章 Architecture

フェールオーバーマネージャーは、Postgres ストリーミングレプリケーションクラスターの状態を監視し、障害をすばやく検証する高可用性ツールです。データベース障害が発生すると、フェールオーバーマネージャーは、ストリーミングレプリケーションスタンバイノードを書き込み可能なプライマリノードに自動的に昇格できます。この機能により、継続的なパフォーマンスが保証され、サービスの中断を最小限に抑えてデータ損失から保護されます。

フェールオーバーマネージャークラスターは、ネットワーク上の次のホストにあるフェールオーバーマネージャプロセスで構成されています。

- プライマリノードは、データベースクライアントにサービスを提供しているプライマリデータベースサーバーです。
- 1つ以上のスタンバイノードが、プライマリノードに関連付けられたストリーミングレプリケーションサーバーです。
- 監視ノードは、フェールオーバーシナリオでプライマリまたはスタンバイのいずれかのアサーションを確認します。障害が発生したときに、プライマリが半分以上のノードを持つパーティションにある場合、プライマリはそのままです。このように、フェールオーバーマネージャーは、偶数のエージェントを含むクラスターでの実行をサポートしています。

証人ではない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、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判明した場合、エージェントはスタンバイの役割を引き継ぎます。
- データベースが復旧していない場合、エージェントはプライマリの役割を引き受けます。

フェ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の最新のスタンバイであることを保証しようとします。スタンバイノードがプライマリノードと同期していない場合、データが失われる可能性があります。

JGroups は、フェールオーバーマネージャーが、メンバーノードが互いに通信し、ノードの障害を検出できるクラスターを作成できるようにするテクノロジーを提供します。

この図は、仮想 IP アドレスを使用するフェールオーバーマネージャークラスターを示しています。データベースが追加または削除されるたびに独自の *script* を提供してロードバランサーを再構成する場合、[Using Failover Manager with virtual IP addresses](#) の代わりにロードバランサーを使用できます。高可用性のためにネイティブ EFM-Pgpool 統合を有効にすることもできます。詳細については、[Choosing a deployment architecture](#) を参照してください。

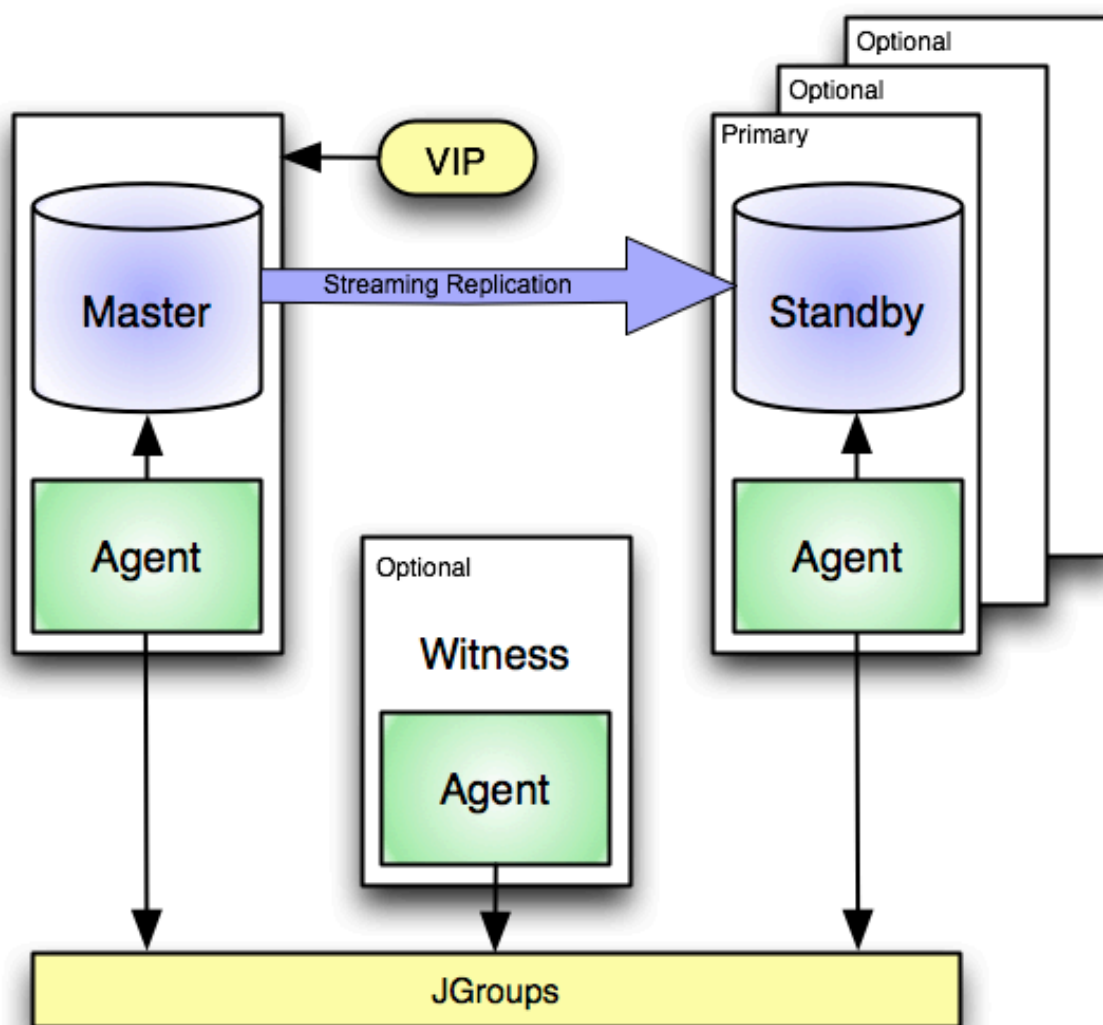


図 1: A Failover Manager scenario employing a virtual IP address

## 第4章 Choosing a deployment architecture

フェールオーバーマネージャーは、Postgres 接続プラーと接続ライブラリを使用して、EDB Postgres Advanced Server にさまざまな高可用性オプションを提供します。これらのオプションは、高可用性アーキテクチャに影響します。

データベースの高可用性を確保するために、フェールオーバーマネージャーのコア機能を Postgres 接続ライブラリ（クライアント接続フェールオーバー）および接続プラーと組み合わせることができます。

Failover Manager の機能を使用して、EDB は高可用性環境を実行するための 4 つの基本アーキテクチャを設計しました。

### 1. *Failover Manager using VIP (virtual IP)*

：フェールオーバーマネージャーには、すぐに VIP アドレスを管理する重要な機能があります。VIP アドレスを使用すると、アプリケーションは、プライマリデータベースサーバーにルーティングされている単一の IP アドレスに接続できます。このアーキテクチャは、環境で VIP アドレスを使用できる場合に実行する最も基本的なソリューションです。

### 2. *Failover Manager using client connect failover* :

libpq や jdbc などの PostgreSQL クライアントライブラリにより、クライアント接続のフェールオーバーが可能になります。クライアント接続フェールオーバーでは、接続文字列に複数のサーバー（host=srv1, srv2）が含まれ、クライアントライブラリは利用可能なホストをループして、利用可能で読み書き操作が可能な接続を見つけます。この機能により、クライアントはスイッチオーバー中にマスターをフォローできます。このソリューションは、仮想 IP アドレスに依存しません。このようなクライアント構成を設定できるすべての環境で使用できます。

### 3. *PgBouncer でフェールオーバーマネージャーを使用する*

：PgBouncer は、接続プーリングやトラフィックを停止するオプションなどの機能を追加します。クライアントと Postgres データベースサーバー間のプロキシとして使用することもできます。フェールオーバーマネージャーの統合オプションを活用して、フェールオーバー中に PgBouncer の再構成を実行することにより、PgBouncer を使用してトラフィックを正しいプライマリデータベースサーバーにルーティングできます。

### 4. *Failover Manager with EDB Pgpool-II* : EDB

Pgpool-II は、クライアントと Postgres データベースサーバー間のプロキシとして使用される別のツールです。EDB Pgpool-II は、ウォッチドッグを使用したクラスターモードでの実行、VIP の管理、読み取り専用のスケーラビリティなどの機能を追加します。フェールオーバーマネージャーには、EDB Pgpool-II と統合して、データベースフェールオーバー操作中にトラフィックを別のプライマリにリダイレクトするネイティブ機能があります。

これらの機能は、各アーキテクチャでサポートされています。

## クライアント接続フェイルオーバー

| Features                 | Failover Manager with VIP | Failover Manager with client connect failover | Failover Manager with EDB PgBouncer | Failover Manager with EDB Pgpool-II |
|--------------------------|---------------------------|-----------------------------------------------|-------------------------------------|-------------------------------------|
| Connection pooling       |                           |                                               | Yes                                 | Yes                                 |
| Runs on cloud (no VIP)   |                           | Yes                                           | Yes                                 | Yes                                 |
| Halt traffic option      |                           |                                               | Yes                                 |                                     |
| Read-only scalability    |                           | Yes (using multiple connection factories)     |                                     | Yes                                 |
| Clustered proxy          |                           |                                               |                                     | Yes                                 |
| Proxy integration        |                           |                                               | ssh                                 | PCP                                 |
| Minimum servers required | 3                         | 3                                             | 5                                   | 6                                   |
| Complexity               | Low                       | Low                                           | Medium                              | High                                |
| Network hops             | 1                         | 1                                             |                                     |                                     |
| Failover duration        | Low                       | Medium                                        | Low                                 | Low                                 |

## 4.1 Failover Manager with virtual IP

フェールオーバーマネージャーは、仮想 IP (VIP) を使用するクラスターのサポートを提供します。

### 4.1.1 VIP でのフェールオーバーマネージャーの使用

#### インストール

次のように、Advanced Server データベースと Failover Manager を 3 つのサーバーにインストールして構成します。

システム | コンポーネント | ----- |-----|  
 ----- - PG プライマリ、PG スタンバイ 1、および PG スタンバイ 2 | Advanced Server 13 および Failover Manager 4.2 を実行しているプライマリ/スタンバイノード

#### VIP の指定

クラスタープロパティファイルで、`virtual.ip` プロパティにホスト名または IP アドレスを指定します。`virtual.ip.prefix` プロパティに対応するプレフィックスを指定します。`virtual.ip.interface` プロパティを使用して、VIP が使用するネットワークインターフェイスを提供します。デフォルトでは、`virtual.ip` および `virtual.ip.prefix` 値はすべてのエージェントで同じです。

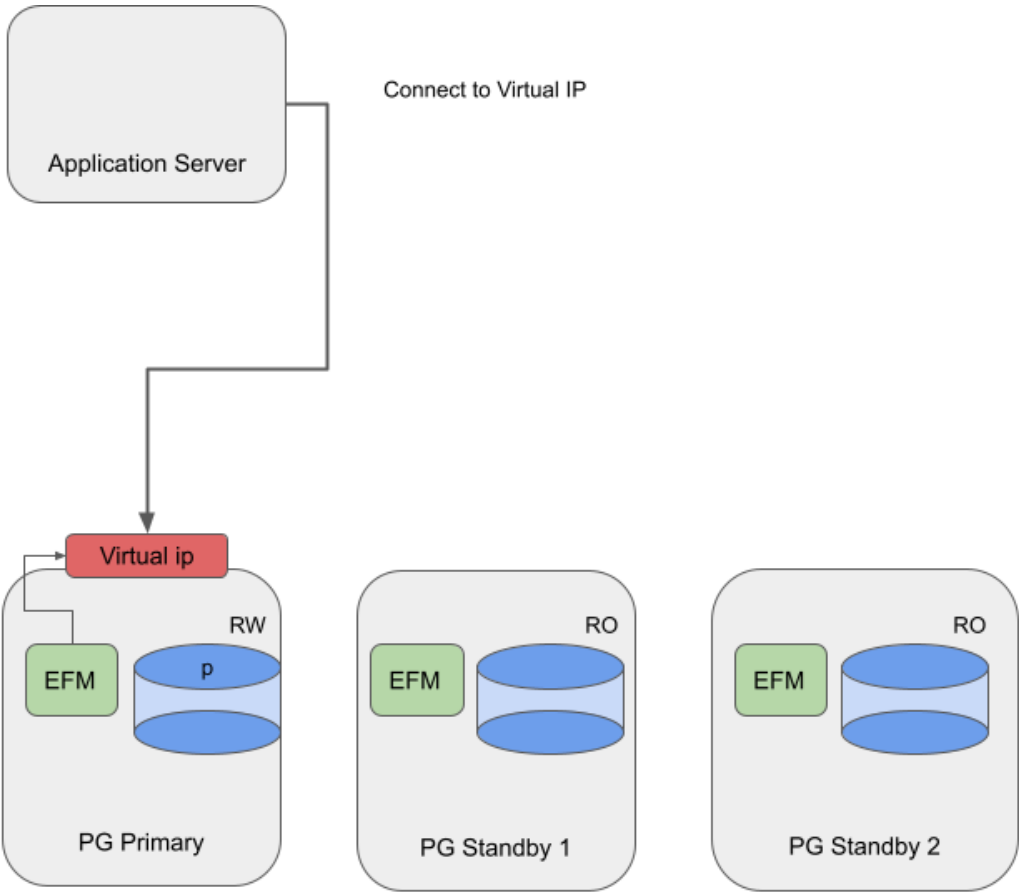


図 1: Failover Manager with VIP

指定された仮想 IP アドレスは、クラスターのプライマリノードにのみ割り当てられます。`virtual.ip.single=true` を指定した場合、フェールオーバー中に新しいプライマリで同じ 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 アドレスの使用の詳細については、[Using Failover Manager with virtual IP addresses](#) を参照してください。

## 4.2 Failover Manager with client connect failover

PostgreSQL 接続ライブラリのほとんどは、クライアント接続のフェイルオーバーをサポートしています。これらのライブラリは、複数のデータベースサーバーとの接続文字列をサポートします。最初の接続試行時または接続が失われた場合（フェイルオーバー中にも発生します）、クライアントドライバーは、読み取り/書き込み接続が見つかるまで、指定されたホストに 1 つずつ接続します。新しいマスターに再接続する時間は、ドライバーと TCP レイヤーで構成された接続タイムアウトによって異なります。

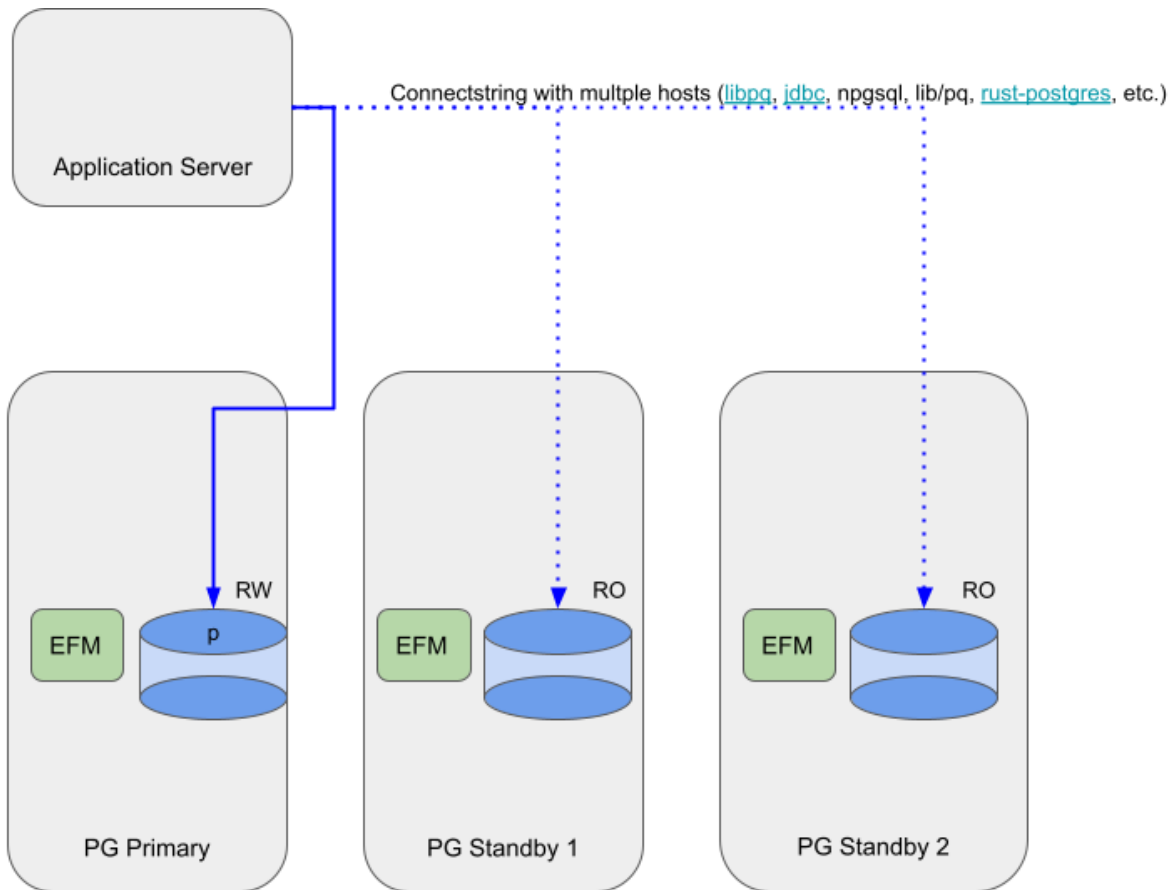


図 2: Failover Manager traffic routing diagram for client connect failover

図 2: クライアント接続フェールオーバーを使用したフェールオーバーマネージャーのトラフィックルーティング

## 4.2.1 クライアント接続フェールオーバーでのフェールオーバーマネージャーの使用

### インストール

次のように、Advanced Server と Failover Manager を 3 つのサーバーにインストールして構成します。

| Systems                            | Components                                                                      |
|------------------------------------|---------------------------------------------------------------------------------|
| PG Primary, Standby1, and Standby2 | PG Primary or standby nodes running Advanced Server 13 and Failover Manager 4.2 |

`efm.properties` (`virtual.ip`、`virtual.ip.interface`、`virtual.ip.prefix`、および `virtual.ip.single`) で仮想 IP 構成を構成する必要はありません。

### クライアント接続フェールオーバーの構成

| Driver | Client connection failover support | Con-<br>nection | Version<br>ported      | sup-<br>ported | Configuration                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------|------------------------------------|-----------------|------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JDBC   | Yes                                |                 | All supported versions |                | Supply multiple hosts in the connection string and set the <code>targetServerType</code> attribute as primary. Example: <code>jdbc:postgresql://host1:5444,host2:5444/accounting?targetServerType=primary</code> . More information: <a href="https://jdbc.postgresql.org/documentation/head/connect.html#connection-failover">https://jdbc.postgresql.org/documentation/head/connect.html#connection-failover</a> |
| libpq  | Yes                                |                 | 10 and above           |                | Supply multiple hosts in the connection string and set the <code>target_session_attrs</code> attribute as read-write. Example: <code>postgresql://host1:5444,host2:5444/edb?target_session_attrs=read-write</code> . More information: <a href="https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING">https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING</a>       |
| .NET   | Yes                                |                 | 6 and above            |                | Supply multiple hosts in the connection string and set the Target Session Attributes attribute as primary. Example: <code>Host=host1,host2;Username=test;Password=test;TargetSessionAttributes=primary</code> . More information: <a href="https://www.npgsql.org/doc/failover-and-load-balancing.html">https://www.npgsql.org/doc/failover-and-load-balancing.html</a>                                            |
| OCL    | Yes                                |                 | 10 and above           |                | OCL is based on <code>libpq</code> hence check <code>libpq</code> for details.                                                                                                                                                                                                                                                                                                                                     |

次のページに続く

表 3 – 前のページからの続き

| Driver | Client<br>connection<br>failover<br>support | Con-<br>nection | Version<br>ported | sup- | Configuration |
|--------|---------------------------------------------|-----------------|-------------------|------|---------------|
| ODBC   | No                                          |                 |                   |      |               |

## 4.3 Failover Manager with EDB PgBouncer

Failover Manager と EDB PgBouncer を使用して、オンプレミスのセットアップとクラウドのセットアップで高可用性を提供できます。EDB PgBouncer は一般的な接続プーラーですが、マルチホスト構成、フェイルオーバー、または検出がないため、PostgreSQL の高可用性を単独で達成するには十分ではありません。

### 4.3.1 オンプレミスの EDB PgBouncer を使用したフェールオーバーマネージャー

オンプレミスセットアップの場合、接続ライブラリを使用して、複数のホストとの接続文字列を使用して高可用性を提供します。

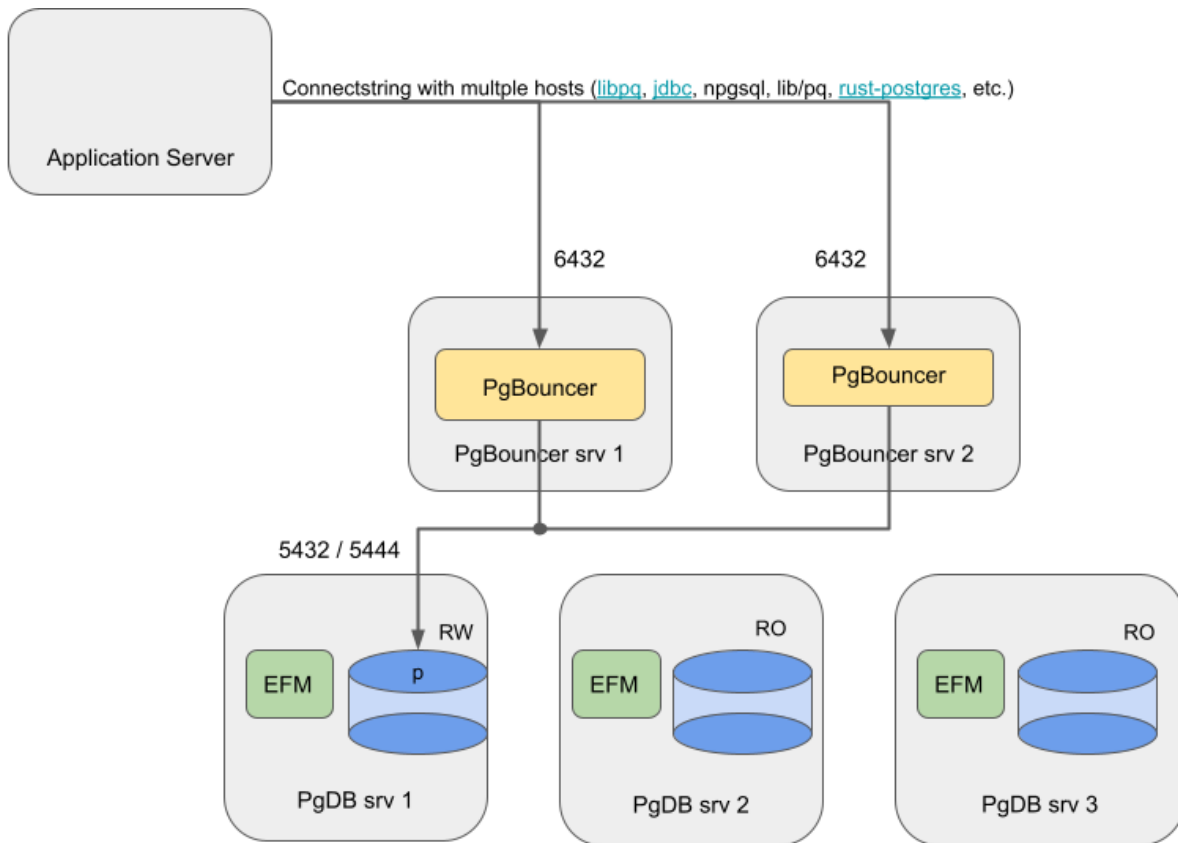


図 3: Failover Manager using pgBouncer on-premises architecture diagram

図 3：オンプレミスの EDB PgBouncer を使用したフェールオーバーマネージャーのトラフィックルーティング

### 4.3.2 クラウド内の EDB PgBouncer を備えた Failover Manager

クラウドセットアップの場合、ネットワークロードバランサー（NLB）を使用して、EDB PgBouncer の両方のインスタンスのトラフィックを分散します。

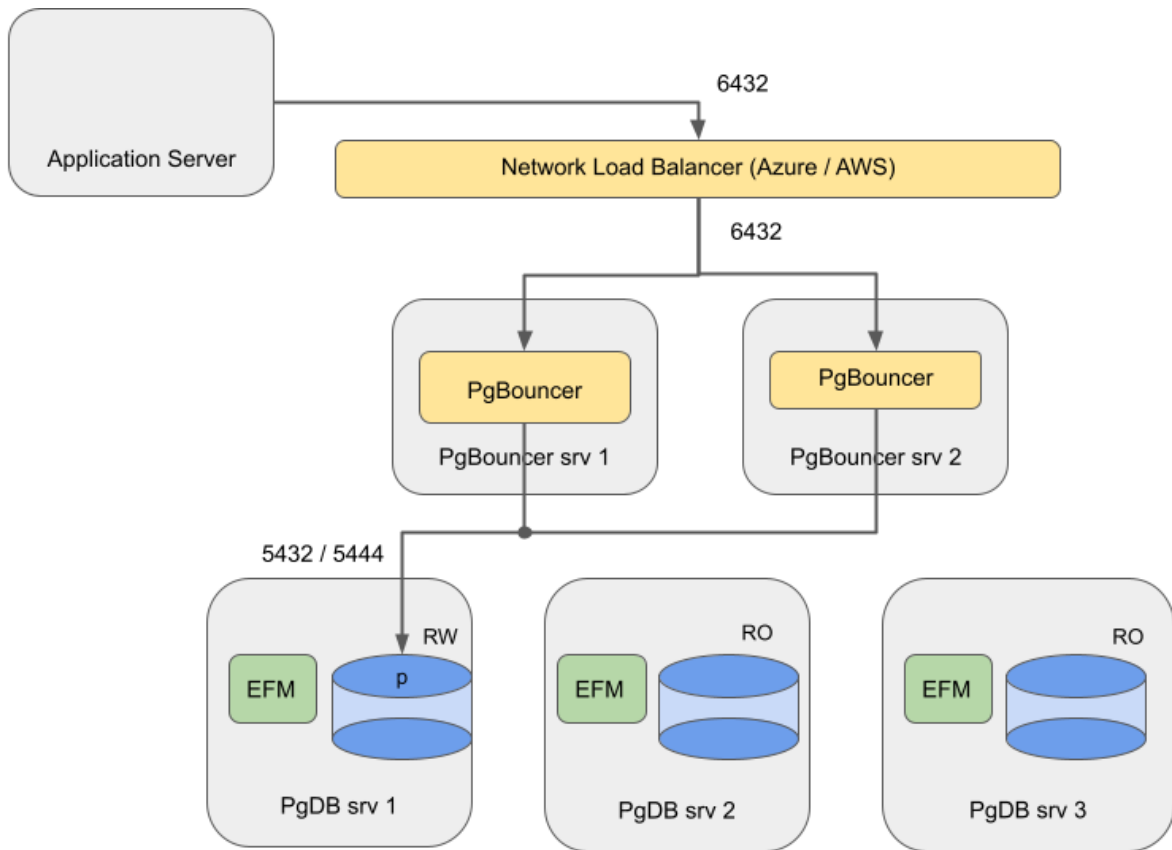


図 4: Failover Manager with PgBouncer cloud architecture diagram

図 4：クラウドで EDB PgBouncer を使用した Failover Manager のトラフィックルーティング

EDB は、同じマシンで実行されている EDB PgBouncer と Failover Manager / PostgreSQL では、このアーキテクチャをサポートしていません。

- クラウドネットワークロードバランサー [Azure](#) で [NLB を構成する](#) による制限

ソースと宛先が同じマシン上にある場合、トラフィックを適切にルーティングしません。

- 混合アーキテクチャでは、EDB PgBouncer と Postgres 間のトラフィックがアンバランスになる場合があります（ローカルの場合も、ネットワークの場合もあります）。
- EDB PgBouncer と PostgreSQL はリソースを競合します。
- マスター障害は、ルーティング（EDB PgBouncer）とデータベースの両方に影響します。これらの 2 つのコンポーネントが同じマシンで結合されます。

### 4.3.3 PgBouncer でフェールオーバーマネージャーを使用する

#### インストール

次のように、AWS 仮想マシンに Advanced Server データベース、Failover Manager、および EDB PgBouncer をインストールして構成します。

| Systems            | Components                                                                                                                                               |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| PgDB srv 1, 2, 3   | Primary / standby node running Advanced Server 13 and Failover Manager 4.2                                                                               |
| PgBouncer srv 1, 2 | PgBouncer node running EDB PgBouncer 1.15. Register these two nodes as targets in the target group. Two is the minimum and is sufficient for most cases. |

#### フェールオーバーマネージャーの構成

Failover Manager documentation で提供されている手順を使用してください

フェールオーバーマネージャーを構成します。これらの手順に加えて、次の手順を実行します。

1. すべての（リモート）EDB PgBouncer ホストに接続し、リダイレクトスクリプトを実行する統合スクリプトを作成します。/usr/edb/efm-4.2/bin/efm\_pg\_bouncer\_functions でスクリプトを見つけます。ユーザー efm が次の内容のスクリプトを実行できることを確認します。

```
#!/bin/bash -x
set -e
IFS=, read -r -a PGB_HOSTS <<< "$4"
FAILED_PGB_HOST=
for PGB_HOST in "${PGB_HOSTS[@]"; do
    echo "redirecting to $2 on enterprisedb@${PGB_HOST}"
    if [ "$3" == "p" ]; then
        ssh "enterprisedb@${PGB_HOST}" /usr/edb/pgbouncer1.15/bin/redirect.sh "$2" ||
        FAILED_PGB_HOST="$FAILED_PGB_HOST $PGB_HOST" < /dev/null
    fi
done

# return exit code to inform EFM agent about failure. The agent would send a failure
# notification accordingly for manual intervention
if [ ! -z "$FAILED_PGB_HOST" ]; then
    echo "Failed to redirect to $2 on $FAILED_PGB_HOST"
    exit 1
fi
```

2. 各データベースノードで、script.load.balancer.attach を efm プロパティファイルのカスタムスクリプトに設定します。

edb\_notranlate\_1 <pgbs1> は PgBouncer サーバー 1 のホスト名または IP アドレスであり、<pgbs2> は PgBouncer サーバー 2 のホスト名または IP アドレスです。

## PostgreSQL の構成

通常のオペレーション中、トラフィックは両方の PgBouncer インスタンス間で分散され、両方が PostgreSQL への接続を開きます。したがって、PostgreSQL では、両方のインスタンスから十分な接続を受け入れるように `max_connections` パラメーターが補償されていることを確認してください。

## EDB PgBouncer の構成

EDB PgBouncer documentation で提供されている手順を使用できます

EDB PgBouncer を設定します。これらの手順に加えて、次の手順を実行します。

1. 次の行を `edb-pgbouncer-1.15.ini` ファイルに追加します。

```
%include /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
```

2. `edb-pgbouncer-1.15.ini` ファイルで、`listen_addr` の値を\*に設定します。

```
listen_addr = *
```

3. `edb-pgbouncer-1.15.ini` ファイルの `[databases]` セクションを空のままにし、別のファイル `/etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini` でこのセクションを構成します。この追加の構成ファイルが `enterprisedb` によって読み取りおよび書き込み可能であることを確認します。

以下は、ファイルを作成する `bash` コマンドの例です。

```
echo "[databases]" > /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
echo "edb= host=svr1" >> /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
chown enterprisedb: /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
```

4. データベースの章を再構成し、`pgbouncer` をリロードするために使用するスクリプト `/usr/edb/pgbouncer1.15/bin/redirect.sh` を作成します。スクリプトの所有者が `root` であること、および `user/group/other (0755)` に読み取りおよび実行アクセスがあることを確認してください。スクリプトの内容は次のとおりです。

```
#!/bin/bash
set -e

#Some defaults
PGBOUNCER_DATABASE_INI=/etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini

PGMSTR=${1:-localhost}

# enterprisedb user does not have permissions to write in folder directly, so `sed -
→i` will not work
TMPFILE=$(mktemp)
sed "s/host=[A-Za-z0-9.]* /host=${PGMSTR} /" "${PGBOUNCER_DATABASE_INI}" > "${TMPFILE}"
```

(次のページに続く)

(前のページからの続き)

```
if ! diff -q "${PGBOUNCER_DATABASE_INI}" "${TMPFILE}" >/dev/null; then
    cat "${TMPFILE}" > "${PGBOUNCER_DATABASE_INI}"
    pkill -SIGHUP pgbouncer
fi
```

## パスワードレス ssh の構成

EDB PgBouncer 統合では、パスワードなしの ssh アクセスが必要です。ssh を構成する方法は複数あります。組織の推奨プロセスに従って、パスワードなしの ssh を構成します。クイックスタートとして、この例に従って、パスワードなしの ssh を構成することもできます。ユーザー efm user は、PgBouncer を実行しているユーザーとして ssh できる必要があります。たとえば、enterprisedb などです。

## EDB PgBouncer ホストで構成する

1. すべての EDB PgBouncer ホストで、enterprisedb ユーザーのパスワードを一時的に設定します。root として `passwd enterprisedb` を実行し、一時パスワードを 2 回入力します。
  2. パスワードなしの ssh が有効になっていることを確認します。次のコマンドで確認できます。
- `edb_notranlate_6 yes` に設定されていることを確認します。必要に応じて変更し、ssh を再起動します。

## フェールオーバーマネージャー/PostgreSQL ホストで構成する

すべてのフェールオーバーマネージャー/postgres ホストで、efm ユーザーとして：

1. 次のコマンドを実行します。

```
ssh-keygen -P "" -f ~/.ssh/id_rsa
```

2. すべての EDB PgBouncer ホストについて、次のコマンドで ssh キーをコピーします。

`edb_notranlate_8 enterprisedb` ユーザーのデフォルトのホームディレクトリは `/var/lib/edb` です。このディレクトリがまだ存在しない場合は、手動で作成します。sudo ユーザーとして、各 EDB PgBouncer ホストで次のコマンドを実行します。

```
mkdir -p /var/lib/edb
chown -R enterprisedb:enterprisedb /var/lib/edb
```

## EDB PgBouncer ホストで一時的パスワードをリセットする

root として次のコマンドを実行することにより、すべての EDB PgBouncer ホストで enterprisedb ユーザーの一時的パスワードをリセットできます。

```
shell passwd -d enterprisedb ### ネットワークロードバランサーの構成
```

AWS または Azure のネットワークロードバランサーを使用した Failover Manager EDB PgBouncer 統合の場合、追加の手順を実行する必要があります。

EDB PgBouncer とデータベースインスタンスが使用するセキュリティグループに次のルールを追加します。

- EDB PgBouncer インスタンス (SG PgBouncer) が使用するセキュリティグループのルール。

| Type       | Protocol | Port range | Source        | Description |
|------------|----------|------------|---------------|-------------|
| Custom TCP | TCP      | 6432       | Entire Subnet | PgBouncer   |
| Custom TCP | TCP      | 22         | Entire Subnet | ssh         |

これらのルールに加えて、要件に応じて SSH と Ping のルールを追加します。

- データベースインスタンス (SG DB) が使用するセキュリティグループのルール：

| Type       | Protocol | Port range | Source        | Description    |
|------------|----------|------------|---------------|----------------|
| Custom TCP | TCP      | 7800       | Entire Subnet | フェールオーバーマネージャー |
| Custom TCP | TCP      | 5444       | Entire Subnet | Postgres       |
| Custom TCP | TCP      | 22         | Entire Subnet | ssh            |

これらのルールにより、データベース、フェールオーバーマネージャー、および EDB PgBouncer の実行に必要なポートが、トラフィックルーティングとヘルスマモニタリングのためにノードとロードバランサー間の通信用に開かれます。

これらのルールに加えて、要件に応じて SSH と Ping のルールを追加します。

## Azure で NLB を構成する

AWS を使用している場合は、[AWS で NLB を設定する](#) を参照してください。

### Creating rules for security groups

で説明されているルールを構成した後、Azure のドキュメントに従って、次のことを行います。

- EDB PgBouncer インスタンスを実行している 2 つの仮想マシンで構成されるバックエンドプールを作成します。仮想マシンのプライベート IP を使用して、バックエンド プールを作成します。
- ヘルスプローブを追加して、EDB PgBouncer インスタンスが仮想マシンで利用可能かどうかを確認します。プロトコルとして TCP を選択し、ポートとして 6432 を選択します。

- ポート 6432 の負荷分散ルールを追加します。このルールにより、そのポートに向かうネットワークトラフィックが、バックエンドプールに存在するすべての仮想マシンに均等に分散されます。タイプとして **Public** ロードバランサーまたは **Internal** ロードバランサーを選択します。

これらの構成を完了した後、ポート 6432 を使用して、ネットワークロードバランサーの IP アドレスでデータベースに接続できます。プライマリデータベースサーバーで障害が発生すると、フェールオーバーマネージャーは新しいプライマリを昇格し、EDB PgBouncer を再構成してトラフィックを再分散します。EDB PgBouncer プロセスのいずれかがトラフィックを受け入れることができない場合、ネットワークロードバランサーはすべてのトラフィックを残りの EDB PgBouncer プロセスに再分散します。max\_client\_conn パラメーターが、フェールオーバーの場合に多数の接続を補償するように調整されていることを確認してください。

## AWS で NLB を設定する

次のサンプル構成では、次のことを前提としています。

- すべての EC2 インスタンスとロードバランサーが同じサブネットにデプロイされます。必要に応じて、データベースノードを別のサブネットに追加できますが、これにはより複雑な構成が必要であり、パフォーマンスに影響を与える可能性があります。
- PgBouncer のセキュリティグループとデータベースインスタンスのセキュリティグループがあります。

Creating rules for security groups

で説明されているルールを構成した後、AWS のドキュメントに従って

- 次の詳細でターゲットグループを作成します。

| Name      | Type      | Protocol | Port | VPC                                                  |
|-----------|-----------|----------|------|------------------------------------------------------|
| pgbouncer | Instances | TCP      | 6432 | Select the VPC to which the instances are connected. |

残りの設定（ヘルスチェック TCP および 高度なヘルスチェック設定）はデフォルトのままにします。

作成したターゲットグループを EDB PgBouncer を実行しているインスタンスに登録します。

- 次の詳細でロードバランサーを作成します。

| Type                                                                               | VPC                                           | Listener                                                                                 |
|------------------------------------------------------------------------------------|-----------------------------------------------|------------------------------------------------------------------------------------------|
| <i>Public</i> or <i>Internal</i> . EDB recommends using an internal load balancer. | Choose a VPC and map it to the desired zones. | Create a listener with <i>TCP</i> as 6432, and forward it to the target group pgbouncer. |

構成が完了したら、ポート 6432 でネットワークロードバランサーの IP アドレスでデータベースに接続できます。プライマリデータベースサーバーで障害が発生すると、フェールオーバーマネージャーは新しいプライマリを昇格し、EDB PgBouncer を再構成してトラフィックを再分散します。EDB PgBouncer プロセスのいずれかがトラフィックを受け入れることができない場合、ネットワークロードバランサーはすべて

のトラフィックを残りの EDB PgBouncer プロセスに再分散します。max\_client\_conn パラメーターが、フェイルオーバーの場合に多数の接続を補償するように調整されていることを確認してください。

## 4.4 Failover Manager with EDB Pgpool-II

Pgpool-II は、読み取り専用トラフィックの負荷分散、接続プーリング、クラスター化プロキシとしての実行など、多くの機能を提供できる一般的な接続プーラーです。フェールオーバーマネージャーで可用性を管理し、正しいプライマリにプロキシするように EDB Pgpool-II を再構成すると、セットアップはオンプレミスセットアップとクラウドセットアップで高可用性を提供できます。

### 4.4.1 オンプレミスの EDB Pgpool-II を使用したフェールオーバーマネージャー

オンプレミスセットアップの場合、VIP を使用して、トラフィックを使用可能な EDB Pgpool-II インスタンスにルーティングできます。このセットアップでは、EDB Pgpool-II の自動フェールオーバーは無効になっており、フェールオーバーマネージャーは EDB Pgpool-II を管理するように構成されています。

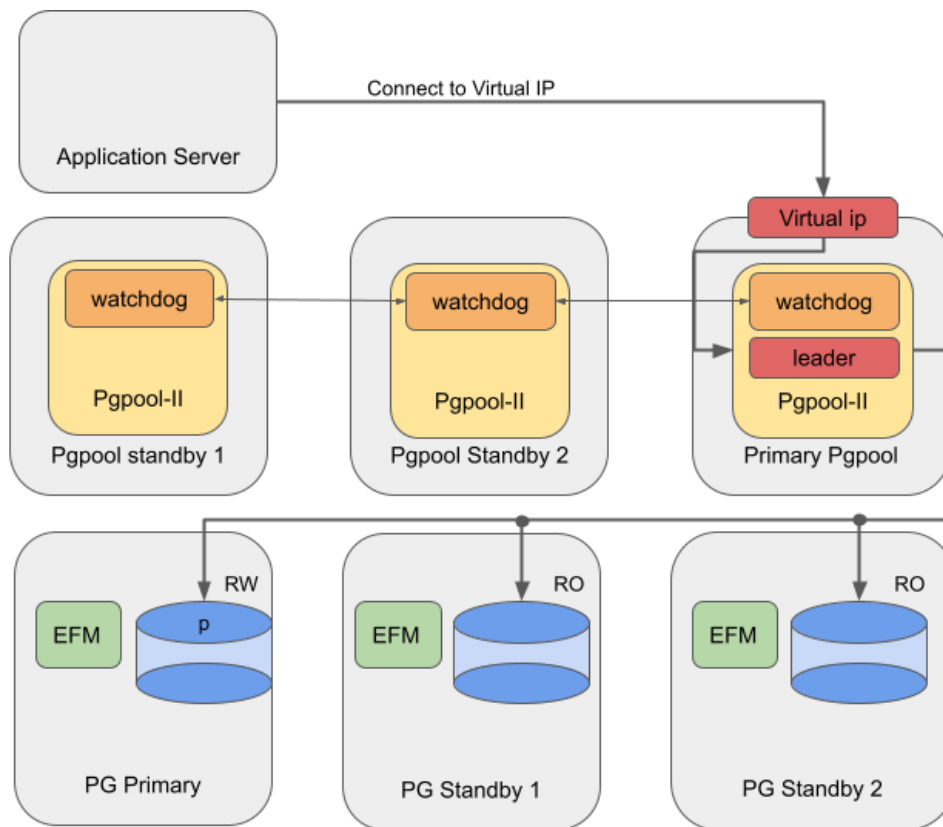


図 5: Failover Manager with Pgpool on-premises

図 5：オンプレミスの EDB Pgpool-II を使用したフェールオーバーマネージャーのトラフィックルーティング

4.4.2 クラウド内の EDB Pgpool-II を使用したフェールオーバーマネージャー

ネットワークロードバランサーを使用する環境（クラウド環境など）では、ネットワークロードバランサー（NLB）を使用して、VIP を必要とせずに、使用可能なすべての EDB Pgpool-II インスタンスにトラフィックを分散できます。

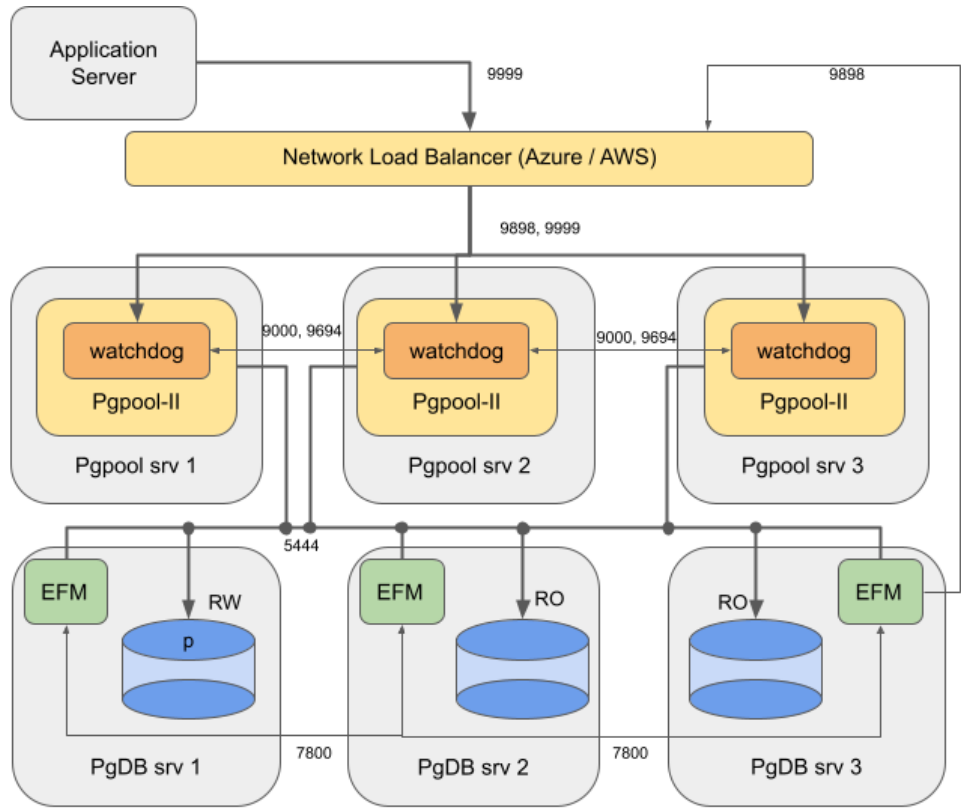


図 6: Failover Manager with Pgpool in cloud

図 6：クラウドで EDB Pgpool-II を使用したフェールオーバーマネージャーのトラフィックルーティング

4.4.3 EDB Pgpool-II でのフェールオーバーマネージャーの使用

インストール

次のように Advanced Server データベース、Failover Manager、および EDB Pgpool-II をインストールして構成します。

| Systems                              | Components                                                                 |
|--------------------------------------|----------------------------------------------------------------------------|
| PgDB server 1, server2, and server 3 | Primary / standby node running Advanced Server 13 and Failover Manager 4.2 |

次のページに続く

表 9 – 前のページからの続き

| Systems       | Components                                                                                                                                                                           |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EDB Pgpool-II | Pgpool node running EDB Pgpool-II 4.2 in a watchdog configuration. Register these three nodes as targets in the target group. Three is the minimum and is sufficient for most cases. |

EDB は、次の理由により、EDB Pgpool-II とフェールオーバーマネージャー/ PostgreSQL が同じマシンで実行されているこのアーキテクチャをサポートしていません。

- ソースと宛先が同じマシンにある場合、クラウドネットワークロードバランサーの制限により、トラフィックが適切にルーティングされません。
- 混合アーキテクチャでは、Pgpool と Postgres 間のトラフィックがアンバランスになる場合があります。
- Pgpool と PostgreSQL がリソースを競合する可能性があります。

### フェールオーバーマネージャーの構成

フェールオーバーマネージャーは、障害が発生したデータベースノードを EDB Pgpool-II 負荷分散から削除できます。また、フェールオーバーマネージャークラスターに戻ったときに、ノードを EDB Pgpool-II に再接続できます。EDB Pgpool-II を使用して高可用性を実現するようにフェールオーバーマネージャーを構成するには、クラスタープロパティファイルで次のプロパティを設定する必要があります。

```
pgpool.enable = true

pcp.user = User invoking PCP commands

pcp.host = Virtual IP of EDB Pgpool-II or IP of NLB

pcp.port = 9898

pcp.pass.file = Absolute path of PCPPASSFILE

pgpool.bin = Absolute path of pgpool bin directory
```

### EDB Pgpool-II の構成

pgpool.conf ファイルでいくつかの重要なパラメーターを構成して、EDB Pgpool-II をフェールオーバーマネージャーと統合できます。

## バックエンドノード設定

3つの PostgreSQL バックエンドノードがあります。1つのプライマリノードと2つのスタンバイノードです。pgpool.conf の backend\_\*構成パラメーターを使用して構成し、すべてのノードに等しいバックエンドの重みを使用します。これにより、読み取りクエリがすべてのノードに均等に分散されます。

```
backend_hostname0 = server1_IP

backend_port0 = 5444

backend_weight0 = 1

backend_flag0 = ALLOW_TO_FAILOVER

backend_hostname1 = server2_IP

backend_port1 = 5444

backend_weight1 = 1

backend_flag1 = ALLOW_TO_FAILOVER

backend_hostname2 = server3_IP

backend_port2 = 5444

backend_weight2 = 1

backend_flag2 = ALLOW_TO_FAILOVER
```

## ロードバランシングとストリーミングレプリケーションモードの有効化

pgpool.conf ファイルで次の構成パラメーターを設定して、ロードバランシングとストリーミングレプリケーションモードを有効にします。

- EDB Pgpool-II バージョン 4.2 の場合：

```
backend_clustering_mode = streaming_replication

load_balance_mode = on
```

- 4.2 より前の EDB Pgpool-II バージョンの場合：

```
master_slave_mode = on
```

(次のページに続く)

(前のページからの続き)

```
master_slave_sub_mode = stream

load_balance_mode = on
```

### ヘルスチェックとフェイルオーバーの無効化

ヘルスチェックとフェイルオーバーは Failover Manager によって処理されるため、EDB Pgpool-II 側で無効にします。EDB Pgpool-II 側でヘルスチェックとフェイルオーバーを無効にするには、次の値を割り当てます。

```
health_check_period = 0

failover_on_backend_error = off

failover_if_affected_tuples_mismatch = off

failover_command =

failback_command =
```

pgpool.conf ファイルの値を設定する際には、次のことを確認してください。

- pgpool.conf の wd\_priority の値は、各ノードで異なるようにします。値が最も高いノードが最高の優先順位を取得します。
- プロパティ backend\_hostname0、backend\_hostname1、backend\_hostname2 などは共有プロパティ（フェイルオーバーマネージャーの用語）であり、すべての EDB Pgpool-II ノードの pgpool.conf ファイルで同じ値を保持する必要があります。
- pgpool.conf ファイルの if\_ \* および arping cmd 小道具の正しいインターフェイス値を更新します。
- すべてのノードの pgpool.conf ファイル内のノード数に応じて、プロパティ heartbeat\_destination0、heartbeat\_destination1、heartbeat\_destination2などを追加します。ここで heartbeat\_destination0 をローカルノードの IP アドレスまたはホスト名に設定します。

## PCP の設定

スクリプトは PCP インターフェイスを使用するため、パスワードプロンプトなしで PCP 接続を許可するには、PCP および .PCPPASS ファイルを設定する必要があります。

PCP をセットアップするには、次を参照してください。 [Configuring pcplib.conf](#)。

PCPPASS を設定するには、次を参照してください。 [PCP commands](#)。

ロードバランシングがオンになって、スタンバイノード全体に読み取りトラフィックを分散することにより、読み取りのスケラビリティを確保します。

フェールオーバーマネージャーは、ヘルスチェックの実行とフェールオーバーのトリガーを担当するため、ヘルスチェックとエラートリガーのバックエンドフェールオーバーはオフになります。この場合、フェールオーバーマネージャーとの競合またはフェールオーバーの時期尚早の実行を回避するために、EDB Pgpool-II を使用してヘルスチェックを実行することはお勧めしません。

最後に、 `search_primary_node_timeout` を低い値に設定して、フェールオーバーマネージャーによってトリガーされるフェールオーバー時に EDB Pgpool-II サービスの迅速な回復を保証します。

## 仮想 IP アドレスの使用

EDB Pgpool-II とフェールオーバーマネージャーはどちらも、シームレスなフェールオーバーのために仮想 IP を採用する機能を提供します。どちらもこの機能を提供しますが、EDB Pgpool-II リーダーは、仮想 IP を介してアプリケーション接続を受信するプロセスです。このデザインのように、そのような仮想 IP 管理は EDB Pgpool-II ウォッチドッグシステムによって実行されます。Failover Manager VIP はこの設計では役に立たないため、無効にします。

EDB Pgpool-II のアクティブインスタンス（サンプルアーキテクチャのプライマリ EDB Pgpool-II サーバー）の障害状況では、次に利用可能なスタンバイ EDB Pgpool-II インスタンス（ウォッチドッグ優先度による）がアクティブ化され、リーダーとして担当します EDB Pgpool-II インスタンス。

## ネットワークロードバランサーの構成

AWS または Azure のネットワークロードバランサーを使用した Failover Manager / EDB Pgpool-II 統合の場合、いくつかの追加手順を実行する必要があります。

EDB Pgpool-II インスタンスが使用するセキュリティグループに次のルールを追加します。

- EDB Pgpool-II インスタンス (SG Pgpool) が利用するセキュリティグループのルール

| Type       | Protocol | Port range | Source        | Description |
|------------|----------|------------|---------------|-------------|
| Custom TCP | TCP      | 9000       | Entire Subnet | ウォッチドッグ     |
| Custom TCP | TCP      | 9694       | Entire Subnet | ハートビート      |
| Custom TCP | TCP      | 9898       | Entire Subnet | PCP         |
| Custom TCP | TCP      | 9999       | Entire Subnet | Pgpool      |

これらのルールに加えて、要件に応じて SSH と Ping のルールを追加します。

- データベースインスタンス (SG DB) が使用するセキュリティグループのルール：

| Type       | Protocol | Port range | Source        | Description    |
|------------|----------|------------|---------------|----------------|
| Custom TCP | TCP      | 7800       | Entire Subnet | フェールオーバーマネージャー |
| Custom TCP | TCP      | 5444       | Entire Subnet | Postgres       |
|            |          |            |               | 該当なし           |

これらのルールを設定すると、データベース、フェールオーバーマネージャー、および EDB Pgpool-II の実行に必要なポートが、トラフィックルーティングとヘルスマonitoringのためにノードとロードバランサー間の通信用に開かれます。

これらのルールに加えて、要件に応じて SSH と Ping のルールを追加します。

### Azure で NLB を構成する

AWS を使用している場合、[AWS で NLB を設定する](#) を参照してください。

*Configuring the network load balancer* で説明されているセキュリティグループルールを構成した後、Azure のドキュメントに従って、次のことを行います。

- EDB Pgpool-II インスタンスを実行しているすべての仮想マシンで構成されるバックエンドプールを作成します。仮想マシンのプライベート IP を使用して、バックエンド プールを作成します。
- ヘルスプローブを追加して、EDB Pgpool-II インスタンスが仮想マシンで利用できるかどうかを確認します。プロトコルを TCP に、ポートを 9999 に設定します。
- ポート 9898 とポート 9999 にそれぞれ 1 つずつ、2 つの負荷分散ルールを追加します。これらのルールにより、そのポートに向かうネットワークトラフィックが、バックエンドプールに存在するすべての仮想マシンに均等に分散されます。タイプを **Public** ロードバランサーまたは **Internal** ロードバランサーに設定します。

これらの構成を完了した後、ポート 9999 のネットワークロードバランサーの IP アドレスでデータベースに接続できます。プライマリデータベースサーバーで障害が発生すると、フェールオーバーマネージャーは新しいプライマリを昇格し、EDB Pgpool-II を再構成してトラフィックを再配布します。。 EDB Pgpool-II プロセスのいずれかがトラフィックを受け入れることができない場合、ネットワークロードバランサーはすべてのトラフィックを残りの 2 つの EDB Pgpool-II プロセスに再分散します。listen\_backlog\_multiplier パラメーターが、フェールオーバーの場合に多数の接続を補償するように調整されていることを確認してください。

## AWS で NLB を設定する

サンプル構成では、次の前提が取られています。

- すべての EC2 インスタンスとロードバランサーが同じサブネットにデプロイされます。必要に応じて、データベースノードを別のサブネットに追加できますが、これにはより複雑な構成が必要であり、パフォーマンスに影響を与える可能性があります。
- EDB Pgpool-II のセキュリティグループとデータベースインスタンスのセキュリティグループがあります。

*Configuring the network load balancer* で説明されているセキュリティグループルールを構成した後、AWS のドキュメントに従って、次のことを行います。

- 次の詳細で 2 つのターゲットグループを作成します。

| Name   | Type      | Protocol | Port | VPC                                                  |
|--------|-----------|----------|------|------------------------------------------------------|
| pcp    | Instances | TCP      | 9898 | Select the VPC to which the instances are connected. |
| pgpool | Instances | TCP      | 9999 | Select the VPC to which the instances are connected. |

残りの設定（ヘルスチェック TCP および 高度なヘルスチェック設定）はデフォルトのままにします。

作成したターゲットグループを PgBouncer を実行しているインスタンスに登録します。

- 次の詳細でロードバランサーを作成します。

| Type                                                                               | VPC                                           | Listener                                                                                                                                                                               |
|------------------------------------------------------------------------------------|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Public</i> or <i>Internal</i> . EDB recommends using an internal load balancer. | Choose a VPC and map it to the desired zones. | Create a listener with <i>TCP</i> set to 9898, and forward it to the target group pcp. Create another listener with <i>TCP</i> set to 9999, and forward it to the target group pgpool. |

構成が完了したら、ポート 9999 でネットワークロードバランサーの IP アドレスのデータベースに接続できます。プライマリデータベースサーバーで障害が発生すると、フェールオーバーマネージャーは新しいプライマリを昇格し、EDB Pgpool-II を再構成してトラフィックを再配布します。EDB Pgpool-II プロセスのいずれかがトラフィックを受け入れることができない場合、ネットワークロードバランサーはすべてのトラフィックを残りの 2 つの EDB Pgpool-II プロセスに再分散します。listen\_backlog\_multiplier がフェイルオーバーの場合に多数の接続を補償するように調整されていることを確認してください。

## 第5章 Installing Failover Manager on Linux

リンクを選択して、該当するインストール手順にアクセスします。

### 5.1 Linux x86-64 (amd64)

#### 5.1.1 Red Hat Enterprise Linux (RHEL) およびその派生物

- *Installing Failover Manager on RHEL 9 or OL 9 x86\_64*、*Installing Failover Manager on RHEL 8 or OL 8 x86\_64*、*Installing Failover Manager on RHEL 7 or OL 7 x86\_64*
- *Oracle Linux (OL) 9*、*Oracle Linux (OL) 8*、*Oracle Linux (OL) 7*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86\_64*、*Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86\_64*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86\_64*、*RHEL/CentOS 7.x* または *RHEL/Rocky Linux/AlmaLinux 8.x*
- *RHEL/CentOS 7.x* または *RHEL/Rocky Linux/AlmaLinux 8.x*

#### 5.1.2 SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86\_64*、*Installing Failover Manager on SLES 12 x86\_64*

#### 5.1.3 Debian とその派生物

- *Installing Failover Manager on Ubuntu 22.04 x86\_64*、*Installing Failover Manager on Ubuntu 20.04 x86\_64*、*Installing Failover Manager on Ubuntu 18.04 x86\_64*
- *Installing Failover Manager on Debian 11 x86\_64*、*Installing Failover Manager on Debian 10 x86\_64*

## 5.2 Linux Installing Failover Manager on Linux IBM Power (ppc64le)

### 5.2.1 Red Hat Enterprise Linux (RHEL) およびその派生物

- *Installing Failover Manager on RHEL 9 or OL 9 x86\_64*、*Installing Failover Manager on RHEL 8 or OL 8 x86\_64*

### 5.2.2 SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86\_64*、*Installing Failover Manager on SLES 12 x86\_64*

#### Prerequisites

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

#### Java 1.8（以降）のインストール

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。

[Installation instructions for Java](#) はプラットフォーム固有です。

#### SMTP サーバーを提供する

ユーザー定義の通知スクリプト、電子メール、またはその両方で指定されたとおりに、フェールオーバーマネージャー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 サーバーは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することもできます。SMTP サーバーの使用の詳細については、[Red Hat deployment guide](#) を参照してください。

## ストリーミングレプリケーションを構成する

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

データベースバージョン 11 (またはそれ以前) では、`-sourcenode` オプションで指定しない限り、`recovery.conf` ファイルは、スイッチオーバー中にランダムスタンバイノードから停止したプライマリにコピーされます。スイッチオーバーを実行する前に、スタンバイノードの `recovery.conf` ファイルのパスが一貫していることを確認します。`-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ノードの昇格](#) を参照してください。

データベースバージョン 12 以降では、`-sourcenode` オプションで特に指定がない限り、`primary_conninfo` および `restore_command` プロパティは、スイッチオーバー中にランダムスタンバイノードから停止したプライマリにコピーされます。

## pg\_hba.conf を変更します

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

そこで：

`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、変更を有効にするには、各ノードで構成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 は Postgres バージョンを指定します。

### データベースサーバーの自動起動の使用

プライマリノードが再起動すると、フェールオーバーマネージャーは、プライマリノードでデータベースがダウンしていることを検出し、スタンバイノードをプライマリのロールに昇格させる場合があります。これが発生した場合、リブートされたプライマリノードの Failover Manager エージェントは `recovery.conf` ファイルを書き込む機会を得られず、`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。これが発生した場合、再起動されたプライマリノードは 2 番目のプライマリノードとしてクラスターに戻ります。

この状態を防ぐには、Failover Manager エージェントがデータベースサーバーの前に自動起動するようにします。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プライマリがあるかどうかを確認します。プライマリノードがある場合、エージェントは `recovery.conf` または `standby.signal` ファイルが存在することを確認します。どちらのファイルも存在しない場合、エージェントは `recovery.conf` ファイルを作成します。

### ファイアウォールを介した通信を確保する

フェールオーバーマネージャーノードのホストで Linux ファイアウォール（つまり、iptables）が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 -j ACCEPT

/sbin/service iptables save
```

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

### データベースユーザーに十分な権限があることを確認します

`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には、フェールオーバーマネージャーに代わって次の機能呼び出すための十分な特権が必要です。

`pg_current_wal_lsn()`

`pg_last_wal_replay_lsn()`

`pg_wal_replay_resume()`

`pg_wal_replay_pause()`

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

- データベースバージョン 10 以降の場合、db.user には `pg_read_all_stats` 特権と `pg_reload_conf()` を実行する権限が必要です。

これらの各関数の詳細については、[PostgreSQL core documentation](#) を参照してください。

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

```
GRANT pg_read_all_settings TO <user_name>;
```

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

## Installing Failover Manager on Linux x86 (amd64)

リポジトリへのアクセスを含む、オペレーティングシステム固有のインストール手順については、以下を参照してください。

### Red Hat Enterprise Linux (RHEL) およびその派生物

- *Installing Failover Manager on RHEL 9 or OL 9 x86\_64*
- *Installing Failover Manager on RHEL 8 or OL 8 x86\_64*
- *Installing Failover Manager on RHEL 7 or OL 7 x86\_64*
- *Oracle Linux (OL) 9*
- *Oracle Linux (OL) 8*
- *Oracle Linux (OL) 7*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86\_64*
- *Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86\_64*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86\_64*
- *RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x*
- *RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x*

### SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86\_64*
- *Installing Failover Manager on SLES 12 x86\_64*

### Debian とその派生物

- [Installing Failover Manager on Ubuntu 22.04 x86\\_64](#)
- [Installing Failover Manager on Ubuntu 20.04 x86\\_64](#)
- [Installing Failover Manager on Ubuntu 18.04 x86\\_64](#)
- [Installing Failover Manager on Debian 11 x86\\_64](#)
- [Installing Failover Manager on Debian 10 x86\\_64](#)

### Installing Failover Manager on RHEL 9 or OL 9 x86\_64

#### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- [Installing EDB Postgres Advanced Server](#) を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
  - EPEL リポジトリをインストールします

```
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.  
↪noarch.rpm
```

## パッケージをインストールする

```
sudo dnf -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

## 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on RHEL 8 or OL 8 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- [Installing EDB Postgres Advanced Server](#) を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは 1 回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
    - EPEL リポジトリをインストールします

```
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.  
↪noarch.rpm
```

### パッケージをインストールする

```
sudo dnf -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照

- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
- EPEL リポジトリをインストールします

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

- 追加のリポジトリを有効にして依存関係を解決します

```
sudo dnf config-manager --set-enabled powertools
```

## パッケージをインストールする

```
sudo dnf -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

## 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。

4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

### Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86\_64

#### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。

- EPEL リポジトリをインストールします

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

- 追加のリポジトリを有効にして依存関係を解決します

```
sudo dnf config-manager --set-enabled powertools
```

#### パッケージをインストールする

```
sudo dnf -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

## 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on RHEL 7 or OL 7 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
  - EPEL リポジトリをインストールします

```
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
↪noarch.rpm
```

- 追加のリポジトリを有効にして依存関係を解決します

```
subscription-manager repos --enable "rhel-*optional-rpms"--enable"rhel-* -extras-  
→rpms" --enable "rhel-ha-for-rhel-*server-rpms"
```

### パッケージをインストールする

```
sudo yum -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は edb-efm47 になります。

インストールプロセスにより、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特権を持つ efm という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on CentOS 7 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは 1 回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. EDB repositories に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
- EPEL リポジトリをインストールします

```
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.  
↪noarch.rpm
```

### パッケージをインストールする

```
sudo yum -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[\*Extending Failover Manager permissions\*](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[\*Controlling the Failover Manager service\*](#) を参照してください。

## Installing Failover Manager on SLES 15 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
zypper lr -E | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. EDB repositories に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
    - 必要な SUSE モジュールをアクティブにします

```
sudo SUSEConnect -p PackageHub/15.4/x86_64
```

- メタデータを更新する

```
sudo zypper refresh
```

### パッケージをインストールする

```
sudo zypper -n install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

## 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on SLES 12 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは 1 回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
zypper lr -E | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。
  - 必要な SUSE モジュールをアクティブにします

```
sudo SUSEConnect -p PackageHub/12.5/x86_64
sudo SUSEConnect -p sle-sdk/12.5/x86_64
```

- メタデータを更新する

```
sudo zypper refresh
```

### パッケージをインストールする

```
sudo zypper -n install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on Ubuntu 22.04 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは 1 回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。

### パッケージをインストールする

```
sudo apt-get -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on Ubuntu 20.04 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- [Installing EDB Postgres Advanced Server](#) を参照
- [PostgreSQL Downloads](#) を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。

### パッケージをインストールする

```
sudo apt-get -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on Ubuntu 18.04 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. EDB repositories に移動します。
  2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。

### パッケージをインストールする

```
sudo apt-get -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

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している場合は、

*Extending Failover Manager permissions* を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。

- 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
- クラスターの各ノードでエージェントを起動します。詳細については、*Controlling the Failover Manager service* を参照してください。

### Installing Failover Manager on Debian 11 x86\_64

#### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは 1 回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

- EDB repositories に移動します。
- EDB リポジトリへのアクセスを提供するボタンを選択します。
- ダウンロードするプラットフォームとソフトウェアを選択します。

#### パッケージをインストールする

```
sudo apt-get -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`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特権を持つ `efm` という名前のユーザーが作成されます。

## 初期構成

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している場合は、

[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、[Controlling the Failover Manager service](#) を参照してください。

## Installing Failover Manager on Debian 10 x86\_64

### 前提条件

インストールプロセスを開始する前に：

- Postgres を同じホストにインストールします（監視ノードには必要ありません）
- Installing EDB Postgres Advanced Server を参照
- PostgreSQL Downloads を参照
- EDB リポジトリをセットアップする

リポジトリのセットアップは1回限りのタスクです。既にリポジトリをセットアップしている場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のコマンドを入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされていません。

EDB リポジトリをセットアップするには：

1. [EDB repositories](#) に移動します。
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
1. ダウンロードするプラットフォームとソフトウェアを選択します。

### パッケージをインストールする

```
sudo apt-get -y install edb-efm<4x>
```

<4x> は、インストールする Failover Manager のバージョンです。たとえば、バージョン 4.7 をインストールする場合、パッケージ名は `edb-efm47` になります。

インストールプロセスにより、`enterisedb` 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特権を持つ `efm` という名前のユーザーが作成されます。

### 初期構成

フェールオーバーマネージャーを使用して、`enterisedb` または `postgres` 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、

*Extending Failover Manager permissions* を参照してください。

クラスターの各ノードにインストールした後：

1. 各ノードで *The cluster properties file* を変更します。
2. 各ノードで *The cluster members file* を変更します。
3. 必要に応じて、仮想 IP アドレス設定と、クラスタープロパティファイルで指定されているスクリプトを構成およびテストします。
4. クラスターの各ノードでエージェントを起動します。詳細については、*Controlling the Failover Manager service* を参照してください。

### Installing Failover Manager on Linux IBM Power (ppc64le)

リポジトリへのアクセスを含む、オペレーティングシステム固有のインストール手順については、以下を参照してください。

#### Red Hat Enterprise Linux (RHEL)

- *Installing Failover Manager on RHEL 9 or OL 9 x86\_64*
- *Installing Failover Manager on RHEL 8 or OL 8 x86\_64*

#### SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86\_64*
- *Installing Failover Manager on SLES 12 x86\_64*

## Installation details

コンポーネントは次の場所にインストールされます。4.x はマイナーリリースを示します。

| Component                         | Location                           |
|-----------------------------------|------------------------------------|
| Executables                       | <i>/usr/edb/efm-4.x/bin</i>        |
| Libraries                         | <i>/usr/edb/efm-4.x/lib</i>        |
| Cluster configuration files       | <i>/etc/edb/efm-4.x</i>            |
| Logs                              | <i>/var/log/efm-4.x</i>            |
| Lock files                        | <i>/var/lock/efm-4.x</i>           |
| Log rotation file                 | <i>/etc/logrotate.d/efm-4.x</i>    |
| sudo configuration file           | <i>/etc/sudoers.d/efm-4.x</i>      |
| Binary to access VIP without sudo | <i>/usr/edb/efm-4.x/bin/secure</i> |



## 第6章 Upgrading Failover Manager

Failover Manager は、Failover Manager によって管理されるクラスターをアップグレードする際に役立つユーティリティを提供します。既存のクラスターをアップグレードするには、以下を行う必要があります。

1. クラスターの各ノードに Failover Manager 4.7 をインストールします。Failover Manager のインストールの詳細については、[Installing Failover Manager on Linux](#) を参照してください。
2. フェールオーバーマネージャーをインストールした後、`efm upgrade-conf` ユーティリティを呼び出して、Failover Manager 4.7 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`）を `/usr/edb/efm-4.7/bin` directory

にインストールします。ユーティリティを呼び出すには、root 権限を引き受けて、次のコマンドを呼び出します。

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

`efm upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、パラメーター値をフェールオーバーマネージャーで使用する新しい構成ファイルにコピーします。ユーティリティは、構成ファイルの更新されたコピーを `/etc/edb/efm-4.7` ディレクトリに保存します。

3. フェールオーバーマネージャー 4.7 の `.properties` および `.nodes` ファイルを変更し、新しい設定を指定します。そのノードのサービスを開始する前に、エディターの選択を使用して、プロパティファイル（`/etc/edb/efm-4.7` ディレクトリにあります）の追加プロパティを変更します。プロパティ設定の詳細については、

[The cluster properties file](#) を参照してください。

---

**注釈:** `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は、`db.bin` プロパティが `Postgres bin` ディレクトリの場所を指定していることを確認します。

---

4. Eager Failover を使用している場合は、Failover Manager クラスターを停止する前に無効にする必要があります。詳細については、

[Eager Failover の無効化](#) を参照してください。

5. バージョン固有のコマンドを使用して、古いフェールオーバーマネージャークラスターを停止します。たとえば、次のコマンドを使用して、バージョン 4.4 クラスターを停止できます。

```
/usr/efm-4.4/bin/efm stop-cluster efm
```

6. クラスターの各ノードで新しい [Controlling the Failover Manager service](#) (`edb-efm-4.7`) を起動します。

次の例は、アップグレードユーティリティを呼び出して、Failover Manager インストール用の `.properties` および `.nodes` ファイルを作成する方法を示しています。

```
[root@hostname ~]# /usr/edb/efm-4.7/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-4.6
Processing efm.properties file
Checking directory /etc/edb/efm-4.5
Processing efm.nodes file

Upgrade of files is finished. The owner and group for properties and nodes files
have been set as efm.
[root@hostname ~]#

:ref:`using a Failover Manager configuration without sudo <Extending Failover Manager
permissions>` の場合、 ``-source`` フラグを含め、 ``upgrade-conf``
```

を呼び出すときに構成ファイルが存在するディレクトリの名前を指定します。ディレクトリが構成のデフォルトディレクトリでない場合、アップグレードされたファイルは `upgrade-conf` コマンドが呼び出されたディレクトリに作成されます。

---

**注釈:** 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ファイルを手動で更新して、新しい Failover Manager サービス名を反映します。

---

## 6.1 Failover Manager のアンインストール

---

**注釈:** カスタムスクリプトを使用している場合は、Failover Manager スクリプトを呼び出しているかどうかを確認します。たとえば、昇格後に実行してさまざまなタスクを実行し、フェールオーバーマネージャの `efm_address` スクリプトを呼び出して仮想 IP アドレスを取得するスクリプト。フェールオーバーマネージャスクリプトを呼び出すカスタムスクリプトがある場合は、カスタムスクリプトを更新して、新しくインストールされたバージョンのフェールオーバーマネージャスクリプトを使用してから、古いバージョンのフェールオーバーマネージャスクリプトをアンインストールします。

---

フェールオーバーマネージャ 4.7 にアップグレードした後、ネイティブパッケージマネージャを使用して、以前にインストールしたフェールオーバーマネージャを削除できます。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 4.4 および不要な依存関係を削除します。

- RHEL または CentOS 7.x の場合：

```
yum remove edb-efm45
```

- RHEL または Rocky Linux または AlmaLinux 8.x の場合：

```
dnf remove edb-efm45
```

- Debian または Ubuntu の場合：

```
apt-get remove edb-efm45
```

- SLES では：

```
zypper remove edb-efm45
```

## 6.2 メンテナンスタスクの実行

OS パッチやデータベースのマイナーバージョンアップグレードなどのメンテナンスアクティビティを実行できます。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。

まず、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します。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プライマリの役割に昇格させます。次に、古いプライマリノードでデータベースの更新を実行します。

クラスターの各ノードで、次の手順を実行してデータベースサーバーを更新します。

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。

EDB Postgres Advanced Server サービスの制御または EDB Postres Advanced Server のバージョンのアップグレードの詳細については、

EDB Postgres Advanced Server Guide を参照してください

更新が完了したら、

*efm set-priority* コマンドを使用して古いプライマリをスタンバイリストの先頭に追加し（必要に応じて）、スイッチオーバーしてクラスターを元の状態に戻すことができます。



## 第7章 Configuring streaming replication

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、

[PostgreSQL core documentation](#) を参照してください。

.pgpass ファイルを使用して、レプリケーションユーザーの md5 認証を有効にすることができます。これは、ご使用の環境にとって最も安全な認証方法では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、

[PostgreSQL core documentation](#) を参照してください。

---

**注釈:** バージョン 3.10 以降、フェールオーバーマネージャーはスタンバイプロモーションに `pg\_ctl` ユーティリティを使用します。スタンバイサーバーの昇格のために `trigger\_file` または `promote\_trigger\_file` パラメーターを設定する必要はありません。

---

### 7.1 カスケードレプリケーションの制限付きサポート

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サポートするわけではありませんが、カスケードレプリケーションシナリオでの単純なフェールオーバーに対して制限付きのサポートを提供します。カスケードレプリケーションにより、スタンバイノードは別のスタンバイノードにストリーミングできるため、プライマリノードへの接続数（および処理オーバーヘッド）が削減されます。

カスケードレプリケーションの構成の詳細については、[PostgreSQL documentation](#) を参照してください。

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、スタンバイノード # 2 で次のプロパティ値を設定します。

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

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

スタンバイノード # 1 に障害が発生した場合、フェールオーバー保護はありませんが、ノードの障害を通知するメールが届きます。

---

**注釈:** スイッチオーバーを実行して元のプライマリに戻すと、カスケードレプリケーションシナリオが保

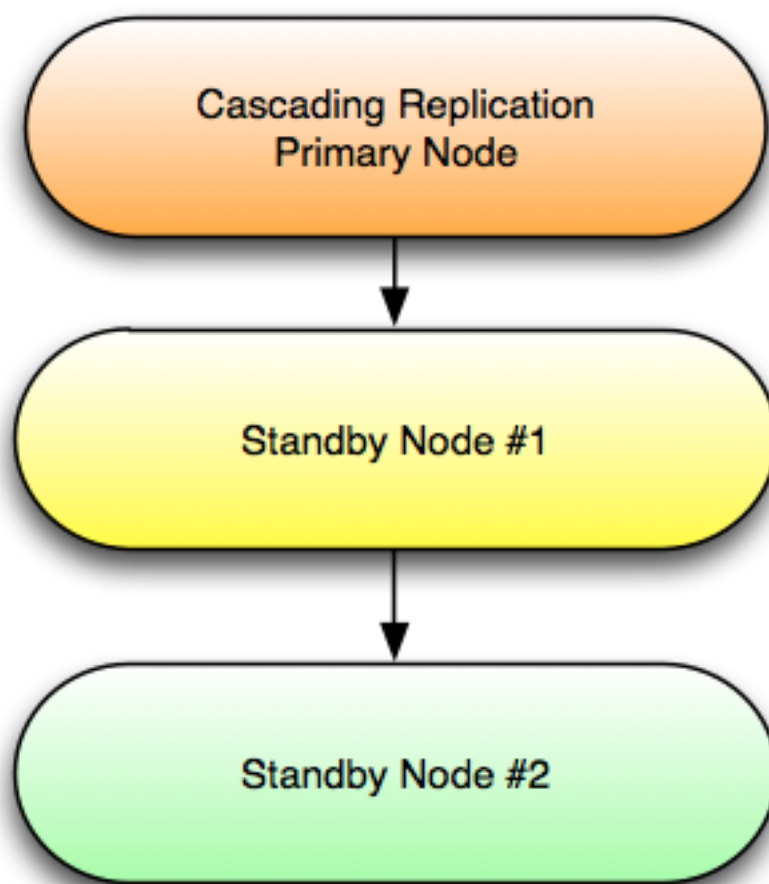


図 1: Cascading replication.

持されない場合があります。

---



## 第8章 Configuring Failover Manager

フェールオーバーマネージャーを構成する方法については、次のトピックを参照してください。

- *The cluster properties file*
- *Encrypting your database password*
- *The cluster members file*
- *Extending Failover Manager permissions*
- *Using Failover Manager with virtual IP addresses*
- *Configuring for Eager Failover*

### 8.1 The cluster properties file

フェールオーバーマネージャークラスター内の各ノードには、ノードが存在する個々のノードのプロパティを含むプロパティファイル（デフォルトでは `efm.properties` という名前）があります。Failover Manager インストーラーは、`/etc/edb/efm-4.<x>` 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します。

```
# cp /etc/edb/efm-4.7/efm.properties.in /etc/edb/efm-4.7/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
```

---

**注釈:** デフォルトでは、フェールオーバーマネージャーはクラスタープロパティファイルの名前が ``efm.properties`` であると想定します。プロパティファイルに ``efm.properties`` 以外の名前を付ける場合、サービススクリプ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します。

---

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の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の所有者は root です。Failover Manager サービススクリプトは、`/etc/edb/efm-4.<x>` ディレクトリでファイルを見つけようとします。プロパティファイルを別の場所に移動する場合、新しい場所を指定するシンボリックリンクを作成する必要があります。

**注釈:** プロパティファイルで参照されてい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ます。

### 8.1.1 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、フェー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、変更を適用するにはフェールオーバーマネージャーを再起動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 はパラメーター値で引用符で囲まれた文字列を使用しますが、フェールオーバーマネージャーはプロパティ値で引用符で囲まれた文字列を許可しません。たとえば、Postgres 構成パラメーターで IP アドレスを次のように指定できますが、

```
listen_addresses='192.168.2.47'
```

フェールオーバーマネージャーでは、値を引用符で囲まないでください。

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

`efm.properties` ファイルのプロパティを使用して、フェールオーバーマネージャーの接続、管理、および操作の詳細を指定します。

凡例：次の表内：

- A：プライマリまたはスタンバイノードで必須
- W：witness ノードで必要
- Y：はい

| Property name           | A | W | Default value | Comments                        |
|-------------------------|---|---|---------------|---------------------------------|
| <i>efm set-priority</i> | Y | Y |               | データベースのユーザー名。                   |
| <i>efm set-priority</i> | Y | Y |               | 「efm encrypt」を使用して暗号化されたパスワード。  |
| <i>efm set-priority</i> | Y | Y |               | この値は、すべてのエージェントで同じである必要があります。   |
| <i>efm set-priority</i> | Y | Y |               | データベース名。                        |
| <i>efm set-priority</i> | Y |   |               | db.database の\$PGDATA dir の所有者。 |
| <i>efm set-priority</i> |   |   |               | データベースをサービスとして実行する場合は必須。        |

次のページに続く

表 1 – 前のページからの続き

| Property name           | A | W | Default value           | Comments                                                                    |
|-------------------------|---|---|-------------------------|-----------------------------------------------------------------------------|
| <i>efm set-priority</i> | Y |   |                         | '/usr/edb/as*nn*/bin' などの pg_controldata/pg_ctl コマンドを含むディレクトリ。              |
| <i>efm set-priority</i> | Y |   |                         | クエリ 'show data_directory;' の出力と同じ                                           |
| <i>efm set-priority</i> |   |   |                         | クエリ 'show config_file;' の出力と同じ。<br><b>*db.data.dir*</b> と異なる場合に指定する必要があります。 |
| <i>efm set-priority</i> | Y | Y | disable                 | <b>重要な注意事項</b> を参照してください。                                                   |
| <i>efm set-priority</i> |   |   |                         | この値はすべてのエージェントで同じである必要があります。通知スクリプトを使用する場合は空白のままにできます。                      |
| <i>efm set-priority</i> |   |   | <i>efm set-priority</i> | デフォルトの <i>efm set-priority</i> を使用するには、空白のままにします。                           |
| <i>efm set-priority</i> | Y | Y | INFO                    | <i>efm set-priority</i> を参照してください。                                          |
| <i>efm set-priority</i> |   |   |                         | 該当なし                                                                        |
| <i>efm set-priority</i> |   |   |                         | user.email プロパティを使用しない場合は必須。両方のパラメーターを一緒に使用できます。                            |
| <i>efm set-priority</i> | Y | Y |                         | 例: <ip_address>:<port>                                                      |
| <i>efm set-priority</i> |   |   |                         | 例: <ip_address/hostname>                                                    |
| <i>efm set-priority</i> | Y | Y | 7809                    | デフォルトのポートがすでに使用されている場合は変更します。                                               |
| <i>efm set-priority</i> | Y | Y |                         | 説明を参照してください。                                                                |
| <i>efm set-priority</i> | Y |   | 10                      | 該当なし                                                                        |
| <i>efm set-priority</i> | Y |   | 60                      | 該当なし                                                                        |
| <i>efm set-priority</i> | Y |   | 10                      | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | 10                      | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | 50                      | この値は、すべてのエージェントで同じである必要があります。                                               |
| <i>efm set-priority</i> | Y | Y | false                   | この値はすべてのエージェントで同じである必要があります                                                 |
| <i>efm set-priority</i> |   |   | true                    | この値は、すべてのエージェントで同じである必要があります。 <b>フェールオーバーマネージャー 4.2 以降で利用できます。</b>          |
| <i>efm set-priority</i> | Y |   | true                    | この値は、すべてのエージェントで同じである必要があります。                                               |
| <i>efm set-priority</i> | Y |   | true                    | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | false                   | 該当なし                                                                        |
| <i>efm set-priority</i> | Y |   | 0                       | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | 8.8.8.8                 | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | /bin/ping -q -c3 -w5    | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | false                   | 該当なし                                                                        |
| <i>efm set-priority</i> | Y | Y | false                   | 該当なし                                                                        |

次のページに続く

表 1 – 前のページからの続き

| Property name           | A | W | Default value          | Comments                                                                                        |
|-------------------------|---|---|------------------------|-------------------------------------------------------------------------------------------------|
| <i>efm set-priority</i> | Y |   | 0                      | 該当なし                                                                                            |
| <i>efm set-priority</i> | Y | Y | true                   | 該当なし                                                                                            |
| <i>efm set-priority</i> | Y |   | true                   | この値は、すべてのエージェントで同じである必要があります。                                                                   |
| <i>efm set-priority</i> | Y |   | true                   | 該当なし                                                                                            |
| <i>efm set-priority</i> | Y | Y | true                   | この値は、すべてのエージェントで同じである必要があります。                                                                   |
| <i>efm set-priority</i> |   |   | 0                      | 該当なし                                                                                            |
| <i>efm set-priority</i> |   |   |                        | 元のプライマリデータベースをスタンバイとして起動する前に、primary_conninfo エントリの application_name 部分をこのプロパティ値に置き換えるように設定します。 |
| <i>efm set-priority</i> |   |   |                        | 例：<br>restore.command=scp<br>&lt;db_service_owner>@%h: &lt;archive_path>/%f<br>%p               |
| <i>efm set-priority</i> | Y |   | false                  | フェールオーバーマネージャー 4.1 を使用している場合は、reconfigure_num_sync_max を参照して num_sync を上げてください。                 |
| <i>efm set-priority</i> |   |   |                        | フェールオーバーマネージャー 4.1 以降で利用できます。                                                                   |
| <i>efm set-priority</i> | Y |   | false                  | 該当なし                                                                                            |
| <i>efm set-priority</i> | Y | Y | 0                      | この値は、すべてのノードで同じである必要があります。                                                                      |
| <i>efm set-priority</i> |   |   |                        | フェールオーバーマネージャー 4.2 以降で利用できます。                                                                   |
| <i>efm set-priority</i> | Y |   | 1                      | 該当なし                                                                                            |
| <i>efm set-priority</i> |   |   | 60                     | 該当なし                                                                                            |
| <i>efm set-priority</i> | Y |   | 0                      | 該当なし                                                                                            |
| <i>efm set-priority</i> |   |   | (see virtual.ip.single | VIP を指定しない場合は空白のままにします。                                                                         |
| <i>efm set-priority</i> |   |   |                        | VIP を指定する場合は必須です。                                                                               |
| <i>efm set-priority</i> |   |   |                        | VIP を指定する場合は必須です。                                                                               |
| <i>efm set-priority</i> | Y | Y | Yes                    | この値は、すべてのノードで同じである必要があります。                                                                      |
| <i>efm set-priority</i> | Y | Y | Yes                    | 該当なし                                                                                            |
| <i>efm set-priority</i> |   |   | false                  | フェールオーバーマネージャー 4.1 以降で利用できます。                                                                   |
| <i>efm set-priority</i> |   |   |                        | pgpool.enable が true に設定されている場合は必須。<br>フェールオーバーマネージャー 4.1 以降で利用できます。                            |

次のページに続く

表 1 – 前のページからの続き

| Property name           | A | W | Default value | Comments                                                                                    |
|-------------------------|---|---|---------------|---------------------------------------------------------------------------------------------|
| <i>efm set-priority</i> |   |   |               | pgpool.enable が true に設定されている場合必須、この値はすべてのエージェントで同じである必要があります。フェールオーバーマネージャー 4.1 以降で利用できます。 |
| <i>efm set-priority</i> |   |   |               | pgpool.enable が true に設定されている場合必須、この値はすべてのエージェントで同じである必要があります。フェールオーバーマネージャー 4.1 以降で利用できます。 |
| <i>efm set-priority</i> |   |   |               | pgpool.enable が true に設定されている場合は必須。フェールオーバーマネージャー 4.1 以降で利用できます。                            |
| <i>efm set-priority</i> |   |   |               | pgpool.enable が true に設定されている場合は必須。<br>**フェールオーバーマネージャー 4.1 以降で利用できます。 **                   |
| <i>efm set-priority</i> |   |   |               | 例： script.load.balancer.attach= /&lt;path>/&lt;attach_script> %h %t                         |
| <i>efm set-priority</i> |   |   |               | 例： script.load.balancer.detach= /&lt;path>/&lt;detach_script> %h %t                         |
| <i>efm set-priority</i> |   |   | true          | 実行中のプライマリデータベースを接続したままにする場合は、false に設定します。フェールオーバーマネージャー 4.2 以降で利用できます。                     |
| <i>efm set-priority</i> |   |   |               | 例： script.fence= /&lt;path>/&lt;script_name> %p %f                                          |
| <i>efm set-priority</i> |   |   |               | 例： script.post.promotion= /&lt;path>/&lt;script_name> %f %p                                 |
| <i>efm set-priority</i> |   |   |               | 例： script.resumed= /&lt;path>/&lt;script_name>                                              |
| <i>efm set-priority</i> |   |   |               | 例： script.db.failure= /&lt;path>/&lt;script_name>                                           |
| <i>efm set-priority</i> |   |   |               | 例： script.primary.isolated= /&lt;path>/&lt;script_name>                                     |
| <i>efm set-priority</i> |   |   |               | 例： script.remote.pre.promotion= /&lt;path>/&lt;script_name> %p                              |
| <i>efm set-priority</i> |   |   |               | 例： script.remote.post.promotion= /&lt;path>/&lt;script_name> %p                             |
| <i>efm set-priority</i> |   |   |               | 例： script.custom.monitor= /&lt;path>/&lt;script_name>                                       |
| <i>efm set-priority</i> |   |   |               | カスタム監視スクリプトが指定されている場合は必須。                                                                   |
| <i>efm set-priority</i> |   |   |               | カスタム監視スクリプトが指定されている場合は必須。                                                                   |
| <i>efm set-priority</i> |   |   |               | カスタム監視スクリプトが指定されている場合は必須。                                                                   |
| <i>efm set-priority</i> | Y | Y | sudo          | 該当なし                                                                                        |
| <i>efm set-priority</i> | Y | Y | sudo -u %u    | 該当なし                                                                                        |

次のページに続く

表 1 – 前のページからの続き

| Property name           | A | W | Default value | Comments                                     |
|-------------------------|---|---|---------------|----------------------------------------------|
| <i>efm set-priority</i> |   |   |               | 指定されない場合、デフォルトは '/var/lock/efm-&lt;version>' |
| <i>efm set-priority</i> |   |   |               | 指定されない場合、デフォルトは '/var/log/efm-&lt;version>'  |
| <i>efm set-priority</i> |   |   | localhost     | 該当なし                                         |
| <i>efm set-priority</i> |   |   | 514           | 該当なし                                         |
| <i>efm set-priority</i> |   |   |               | 該当なし                                         |
| <i>efm set-priority</i> |   |   | UDP           | 該当なし                                         |
| <i>efm set-priority</i> | Y | Y | true          | 該当なし                                         |
| <i>efm set-priority</i> | Y | Y | false         | 該当なし                                         |
| <i>efm set-priority</i> |   |   | info          | 該当なし                                         |
| <i>efm set-priority</i> |   |   | info          | 該当なし                                         |
| <i>efm set-priority</i> |   |   | -             | 該当なし                                         |
|                         |   |   | Xmx128m       |                                              |

### 8.1.2 クラスタプロパティ

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=
```

指定された `db.user` には、フェールオーバーマネージャに代わって選択した PostgreSQL コマンドを呼び出すための十分な特権が必要です。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、[Encrypting your database password](#) を参照してください。

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

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

db.service.owner=
```

サービスの開始または停止時に `service` または `systemctl` コマンドを使用する場合、`db.service.name` プロパティに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`）を使用します。`pg_ctl` プログラムを使用してサービスを制御する場合、`db.bin` プ

ロパティで `pg_ctl` プログラムの場所を指定します。

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

`db.data.dir` プロパティを使用して、昇格中にクラスターのプライマリノードでリカバリファイルを書き込む場所を指定します。このプロパティは、プライマリノードとスタンバイノードで必要です。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.  
  
# After a failover, the recovery.conf files on remaining standbys are  
  
# changed to point to the new primary db (a copy of the original is made  
  
# first). On a primary node, a recovery.conf file will be written during  
  
# failover and promotion to ensure that the primary node can not be  
  
# restarted as the primary database.  
  
# This corresponds to database environment variable PGDATA and should  
  
# be same as the output of query show data_directory; on respective  
  
# database.  
  
db.data.dir=
```

`db.data.dir` プロパティを使用して、昇格中にクラスターのプライマリノードでリカバリファイルを書き込む場所を指定します。このプロパティは、プライマリノードとスタンバイノードで必要です。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.
#
# After a failover, the recovery.conf files on remaining standbys are
# changed to point to the new primary db (a copy of the original is made
# first). On a primary node, a recovery.conf file will be written during
# failover and promotion to ensure that the primary node can not be
# restarted as the primary database.
#
# This corresponds to database environment variable PGDATA and should
# be same as the output of query show data_directory; on respective
# database.
db.data.dir=

```

データベース構成ファイルが `recovery.conf` または `standby.signal` ファイルと同じディレクトリに保存されていない場合、`db.config.dir` プロパティを使用してデータベース構成ファイルの場所を指定します。これは、EDB Postgres 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、EDB Postgres Advanced Server data ディレクトリの場所として使用されます。

```

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

```

(次のページに続く)

(前のページからの続き)

```
# be used. This corresponds to the output of query show config_file;  
  
# on respective database.  
  
db.config.dir=
```

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

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

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

---

**注釈:** `jdbc.sslmode` の値を `verify-ca` に設定し、証明書の検証に Java トラストストアを使用する場合は、次の値を設定する必要があります。

---

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

SSL の構成と使用の詳細については、PostgreSQL ドキュメントの [Secure TCP/IP Connections with SSL](#) および [Using SSL](#) を参照してください。

`user.email` プロパティを使用して、Failover Manager によって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の送信者のアドレスとして使用する値を指定します。次のことができます。

- `from.email` を空白のままにして、デフォルト値 (`efm@localhost`) を使用します。
- 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し、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

`hostname` ユーティリティによって返されるホストの名前に置き換えられます。

通知の詳細については、*Notifications* を参照してください。

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

# Leave blank to use the default, efm@localhost.

from.email=
```

`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、

*Notifications* を参照してください。

```
# Minimum severity level of notifications that will be sent by
# the agent. The minimum level also applies to the notification
```

(次のページに続く)

(前のページからの続き)

```
# script (below). Valid values are INFO, WARNING, and SEVERE.

# A list of notifications is grouped by severity in the users

# guide.

notification.level=INFO
```

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

```
# Text to add to the beginning of every notification. This could

# be used to help identify what the cluster is used for, the role

# of this node, etc. To use multiple lines, add a backslash \ to

# the end of a line of ini. To include a newline use \n.

# Example:

# notification.ini.prefix=Development cluster for Example dept.\n\

# Used by Dev and QA \

# See Example group for questions.

notification.ini.prefix=
```

`script.notification` プロパティを使用して、通知サービスとして機能するユーザー指定のスクリプトへのパスを指定します。スクリプトには、メッセージの件名とメッセージの本文が渡されます。スクリプトは、フェールオーバーマネージャーがユーザ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.
```

(次のページに続く)

(前のページからの続き)

`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にあるエージェントの 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=`

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

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

`external.address=`

`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
```

(次のページに続く)

```
# 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=
```

EDB Postgres Advanced Server `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、データベースが復旧中の場合は `true` を返し、データベースが復旧中でない場合は `false` を返します。エージェントは起動すると、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` を応答した場合、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` で応答した場合、エージェントはプライマリの役割を引き受けます。
- ローカルデータベースがない場合、エージェントはアイドル状態を想定します。

---

**注釈:** `is.witness` が `true` の場合、フェールオーバーマネージャーは回復状態をチェックしません。

---

次のプロパティがローカルサーバーに適用されます。

- `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の秒数を指定します。
- `local.timeout` プロパティは、エージェントがローカルデータベースサーバーからの肯定的な応答を待機する秒数を指定します。
- `local.timeout.final` プロパティは、以前のチェックが現在のノード上のデータベースサーバーへの接続に失敗した後、エージェントが待機する秒数を指定します。`local.timeout.final` プロパティで指定された秒数以内にデータベースから応答を受信しない場合、データベースに障害が発生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が指定されている場合、ローカルデータベースのチェックは 10 秒ごとに実行されます。ローカルデータベースへの接続の試行が 60 秒以内に返されない場合、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を最終試行します。10 秒以内に応答が受信されない場合、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します。

`remote.timeout` プロパティを使用して、エージェントがリモートエージェントまたはデータベースからの応答を待機する秒数を制限します。エージェントは、クラスターイベント中にのみ相互にメッセージを送信します。例は次のとおりです。

- 失敗した可能性のあるリモートデータベースに接続しようとし、他のエージェントに接続できるかどうかを確認します。
- スイッチオーバーの一部としてスタンバイエージェントからリカバリ設定を要求するプライマリエージェント。
- フェールオーバーマネージャークラスターを停止するときにシャットダウンの準備をするようにノードに指示します。

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

remote.timeout=10
```

`node.timeout` プロパティを使用して、ノードに障害が発生したかどうかを判断するときに、エージェントが別のノードからのハートビートを待機する秒数を指定します。

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

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

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

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

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

`stop.isolated.primary` プロパティを使用して、プライマリエージェントがデータベースが分離された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に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、フェールオーバーマネージャーは、`script.primary.isolated` プロパティで指定されたスクリプトを呼び出す前にデータベースを停止します。

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

stop.isolated.primary=true
```

`stop.failed.primary` プロパティを使用して、データベースに到達できない場合にプライマリデータベースのシャットダウンを試みるように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、データベースのシャットダウンを試行した後、`script.db.failure` プロパティで指定されたスクリプトを実行します。

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

stop.failed.primary=true
```

`primary.shutdown.as.failure` プロパティを使用して、プライマリノードでの Failover Manager エージェントのシャットダウンを障害として扱います。このプロパティが `true` に設定され、プライマリエージェントがシャットダウンされている場合、クラスターの残りの部分はシャットダウンを失敗として扱います。これには、ノード全体のシャットダウンなど、エージェントの適切なシャットダウンが含まれます。この場合、タイムアウトプロパティはいずれも適用されません。エージェントが終了すると、クラスターの残りの部分にすぐに通知されます。エージェントが終了した後、クラスターの残りの部分は、プライマリエージェントに障害が発生した場合に発生する `checks` を実行します。

`checks` には、プライマリデータベースへの接続の試行、VIP が使用されている場合に到達可能かどうかの確認などが含まれます。

- 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、フェールオーバーが発生します。

```
# Treat a primary agent shutdown as an agent failure. This can be set
# to true to treat a primary agent shutdown as a failure situation.
# Caution should be used when using this feature, as it could
```

(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 primary  
  
# database maintenance.  
  
# Please see the users guide for more information.  
  
primary.shutdown.as.failure=false
```

`primary.shutdown.as.failure` プロパティは、プライマリノードの偶発的なシャットダウンなどの障害ではなく、ユーザーエラーをキャッチすることを目的としています。ノードの適切なシャットダウンは、クラスターの残りの部分には、ユーザーがプライマリ Failover Manager エージェントを停止したかのように見えます。たとえば、プライマリデータベースのメンテナンスを実行します。`primary.shutdown.as.failure` プロパティを `true` に設定した場合、メンテナンス時に注意してください。

`primary.shutdown.as.failure` が `true` の場合にプライマリデータベースのメンテナンスを実行するには、プライマリエージェントを停止し、プライマリエージェントに障害が発生したが、データベースはまだ実行されているという通知を受信するのを待ちます。その後、プライマリデータベースを安全に停止できます。または、`stop-cluster` コマンドを使用して、障害チェックを実行せずにすべてのエージェントを停止できます。

`update.physical.slots.period` プロパティを使用して、データベースバージョン 12 以降のスロットアドバンス周波数を定義します。`update.physical.slots.period` が正の整数値に設定されている場合、プライマリエージェントは `update.physical.slots.period` 秒ごとに物理レプリケーションスロットの現在の `restart_lsn` を読み取り、この情報を `pg_current_wal_lsn` および `primary_slot_name` (postgresql.conf ファイルに設定されている場合) とともにスタンバイに送信します。エージェントが物理スロットを見つけるには、プライマリに物理スロットがすでに存在する必要があります。物理スロットがスタンバイにまだ存在しない場合、スタンバイエージェントはスロットを作成し、これらのスロットの `restart_lsn` パラメーターを更新します。プロモーション不可のスタンバイは新しいスロットを作成しませんが、存在する場合は更新します。

スロットの `restart_lsn` 値を更新する前に、エージェントは `xmin` 値が設定されているかどうかを確認します。これは、これが以前にプライマリノードであった場合に発生する可能性があります。`xmin` 値がスロットに設定されている場合、エージェントは `restart_lsn` 値を更新する前にスロットを削除して再作成します。

注：必要に応じて現在のプライマリ上の 1 つのセットを含むすべてのスロット名は、一意である必要があります。

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

(次のページに続く)

(前のページからの続き)

```
update.physical.slots.period=0
```

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

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

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

```
# This command will be used to test the reachability of certain
#
# nodes.
#
# Do not include an IP address or hostname on the end of
```

(次のページに続く)

(前のページからの続き)

```
# this command - it will be added dynamically at runtime with the
# values contained in virtual.ip and ping.server.ip.
#
# Make sure this command returns reasonably quickly - test it
# from a shell command line first to make sure it works properly.

ping.server.command=/bin/ping -q -c3 -w5
```

`auto.allow.hosts` プロパティを使用して、最初の `nostarts` の `.nodes` ファイルで指定されたアドレスを使用して許可ホストリストを更新するようにサーバーに指示します。`auto.allow.hosts` を `true` に設定してこのプロパティを有効にすると、クラスターの起動を簡素化できます。

```
# Have the first nostarts 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離脱するときに `nodes` 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、変更されない 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、フェールオーバーマネージャー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` に設定されます。

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

auto.failover=true
```

`auto.reconfigure` プロパティを使用して、プライマリスタンバイがプライマリに昇格した後、残りのスタンバイサーバーの自動再構成を有効または無効にするようにフェールオーバーマネージャーに指示します。プロパティを `true`（デフォルト）に設定して自動再構成を有効にするか、`false` を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。EDB Postgres 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 primary. For database
# versions before 12, Failover Manager will back up recovery.conf.
# Then it will change the host parameter of the primary_conninfo entry
# in recovery.conf or postgresql.auto.conf, and restart the database.
# The restart command is contained in either the efm_db_functions or
# efm_root_functions file; default when not running db as an os
# service is: "pg_ctl restart -m fast -w -t <timeout> -D <directory>"
```

(次のページに続く)

(前のページからの続き)

```
# where the timeout is the local.timeout property value and the  
  
# directory is specified by db.data.dir. To turn off  
  
# automatic reconfiguration, set this property to false.  
  
auto.reconfigure=true
```

**注釈:** `primary_conninfo` は、`keyword=value` ペアのスペース区切りのリストです。

`promotable` プロパティを使用して、ノードをプロモートしないことを示します。`promotable` プロパティは、プライマリエージェントの起動時に無視されます。これにより、スイッチオーバーまたはフェールオーバー後に元のプライマリに簡単に戻すことができます。設定をオーバーライドするには、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、

*Using the efm utility* を参照してください。

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

同じ量のデータが複数のスタンバイノードに書き込まれ、フェールオーバーが発生した場合、`use.replay.tiebreaker` 値は、フェールオーバーマネージャーが交換用のプライマリを選択する方法を決定します。`use.replay.tiebreaker` プロパティを `true` に設定して、ログシーケンス番号で決定されるように、より迅速に回復から復帰するノードにフェールオーバーするようにフェールオーバーマネージャーに指示します。ログシーケンス番号を無視し、ユーザー設定に基づいてノードを昇格するには、`use.replay.tiebreaker` を `false` に設定します。

```
# Use replay LSN value for tiebreaker when choosing a standby to  
  
# promote before using failover priority. Set this property to true to
```

(次のページに続く)

(前のページからの続き)

```
# consider replay location as more important than failover priority

# (as seen in cluster-status command) when choosing the "most ahead"

# standby to promote.

use.replay.tiebreaker=true
```

`standby.restart.delay` プロパティを使用して、スタンバイが昇格後に新しいプライマリをフォローするように再構成 (stopstarts) するまで待機する時間を秒単位で指定します。

```
# Time in seconds for this standby to delay restarting to follow the

# primary after a promotion. This can be used to have standbys restart

# at different times to increase availability. Caution should be used

# when using this feature, as a delayed standby will not be following

# the new primary and care must be taken that the new primary retains

# enough WAL for the standby to follow it.

# Please see the users guide for more information.

standby.restart.delay=0
```

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

```
# During a switchover, recovery settings are copied from a standby

# to the original primary. If the application.name property is set,

# Failover Manager will replace the application_name portion of the

# primary_conninfo entry with this property value before starting

# the original primary database as a standby. If this property is

# not set, Failover Manager will remove the parameter value

# from primary_conninfo.

application.name=
```

**注釈:** プライマリおよび昇格可能なスタンバイで `application.name` プロパティを設定します。フェールオーバー/スイッチオーバーが発生した場合、プライマリノードは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、新しいプライマリが昇格したときに `restore_command` 値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、新しいプライマリ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` を新しいプライマリのアドレスに置き換えます。 `%f` および `%p` は、サーバーが使用するプレースホルダーです。プロパティを空白のままにすると、フェールオーバーマネージャーはプロモーション後にスタンバイの `restore_command` 値を更新しません。

`restore_command`

の使用の詳細については、PostgreSQL のドキュメントを参照してください。

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

プライマリノードのデータベースパラメーター `synchronous_standby_names` は、データの受信を確認して、プライマリノードが書き込みトランザクションを受け入れられるようにする同期スタンバイサーバーの名前と数を指定します。 `reconfigure.num.sync` プロパティが `true` に設定されている場合、フェールオーバーマネージャーは同期スタンバイサーバーの数を減らし、プライマリノードの構成をリロードして現在の値を反映します。

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

reconfigure.num.sync=false
```

**注釈:** `reconfigure.num.sync` プロパティを使用している場合、プライマリデータベースの ``wal_sender_timeout`` 値が ``efm.node.timeout`` 値よりも少なくとも 10 秒短く設定されていることを確認してください。

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

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

reconfigure.num.sync.max=
```

`reconfigure.sync.primary` プロパティを `true` に設定して、スタンバイノードの数が必要なレベルを下回った場合、プライマリデータベースの同期レプリケーションモードを終了します。`reconfigure.sync.primary` を `false` に設定して、同期レプリケーションを中断せずにスタンバイカウントが低下した場合に

通知を送信します。

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

---

**注釈:** `reconfigure.sync.primary` プロパティを使用している場合、プライマリデータベースの ``wal_sender_timeout`` 値が ``efm.node.timeout`` 値よりも少なくとも 10 秒短く設定されていることを確認します。

---

`minimum.standbys` プロパティを使用して、クラスターに保持するスタンバイノードの最小数を指定します。スタンバイカウントが指定された最小値まで低下した場合、プライマリノードに障害が発生してもレプリカノードは昇格されません。

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

(次のページに続く)

(前のページからの続き)

```
# each node.

minimum.standbys=0
```

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

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

priority.standbys=
```

`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フェールオーバーマネージャーが待機する秒数を指定します。

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

recovery.check.period=1
```

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

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

# This retry mechanism is unrelated to the auto.resume.period
```

(次のページに続く)

(前のページからの続き)

```
# 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を指定します。このプロパティは、監視対象データベースに障害が発生し、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に適用されます。

```
# 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 を使用する場合は、`virtual.ip` プロパティにホスト名または IP アドレスを指定します。`virtual.ip.prefix` プロパティに対応するプレフィックスを指定します。`virtual.ip` のままにして、仮想 IP サポートを無効にします。

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

指定された仮想 IP アドレスは、クラスターのプライマリノードにのみ割り当てられます。`virtual.ip.single=true` を指定すると、フェイルオーバーが発生した場合、新しいプライマリで同じ 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 アドレスの使用の詳細については、[Using Failover Manager with virtual IP addresses](#) を参照してください。

```
# These properties specify the IP and prefix length that will be  
  
# remapped during failover. If you do not use a VIP as part of  
  
# your failover solution, leave the virtual.ip property blank to  
  
# disable Failover Manager support for VIP processing (assigning,  
  
# releasing, testing reachability, etc).  
  
#
```

(次のページに続く)

(前のページからの続き)

```
# If you specify a VIP, the interface and prefix are required.

#
# If you specify a host name, it will be resolved to an IP address

# when acquiring or releasing the VIP. If the host name resolves

# to more than one IP address, there is no way to predict which

# address Failover Manager will use.

#
# By default, the virtual.ip and virtual.ip.prefix values must be

# the same across all agents. If you set virtual.ip.single to

# false, you can specify unique values for virtual.ip and

# virtual.ip.prefix on each node.

#
# If you are using an IPv4 address, the virtual.ip.interface value

# should not contain a secondary virtual ip id (do not include

# ":1", etc).

virtual.ip=
virtual.ip.interface=
virtual.ip.prefix=
virtual.ip.single=true
```

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

**check.vip.before.promotion** プロパティを **false** に設定して、障害が発生した場合に VIP を新しいプライマリに割り当てる前に、フェールオーバーマネージャーが VIP が使用中であるかどうかを確認しないようにします。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場合があります。プライマリノードが分離されている場合、または別のプロセスを介してシャットダウンできる場合を除き、このプロパティを設定します **to true**。

```
# Whether to check if the VIP (when used) is still in use before
```

(次のページに続く)

(前のページからの続き)

```
# promoting after a primary failure. Turning this off may allow
# the new primary to have the VIP even though another node is also
# broadcasting it. This should only be used in environments where
# it is known that the failed primary node will be isolated or
# shut down through other means.

check.vip.before.promotion=true
```

`pgpool.enable` プロパティを使用して、高可用性のためにフェールオーバーマネージャーと Pgpool の統合を有効にするかどうかを指定します。非 `sudo` モード (DB 所有者として実行) で Pgpool 統合を有効にする場合、PCPPASS ファイルは DB 所有者のオペレーティングシステムユーザーが所有し、ファイル権限を 600 に設定する必要があります。

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

pgpool.enable=false
```

以下のパラメーターを使用して、Pgpool 統合に使用する値を指定します。

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

(次のページに続く)

(前のページからの続き)

```
pcp.user=
pcp.host=
pcp.port=
pcp.pass.file=
pgpool.bin=
```

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

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

**script.load.balancer.attach** プロパティの名前を付けたスクリプトを提供して、ノードをロードバランサーに接続するときに呼び出すスクリプトを識別します。**script.load.balancer.detach** プロパティを使用して、ロードバランサーからノードをデタッチするときに呼び出すスクリプトの名前を指定します。**%h** プレースホルダーを含めて、クラスターに接続または削除されるノードの IP アドレスを表します。**%t** プレースホルダーを含めて、文字列に p（プライマリノードの場合）または s（スタンバイノードの場合）を含めるようにフェールオーバーマネージャーに指示します。

```
# Absolute path to load balancer scripts

# The attach script is called when a node should be attached to

# the load balancer, for example after a promotion. The detach

# script is called when a node should be removed, for example

# when a database has failed or is about to be stopped. Use %h to

# represent the IP/hostname of the node that is being

# attached/detached. Use %t to represent the type of node being

# attached or detached: the letter m will be passed in for primary nodes

# and the letter s for standby nodes.

#

# Example:

# script.load.balancer.attach=/somepath/attachscript %h %t

script.load.balancer.attach=
```

(次のページに続く)

(前のページからの続き)

```
script.load.balancer.detach=
```

`detach.on.agent.failure` プロパティを使用して、プライマリエージェントに障害が発生したが、データベースにはまだ到達可能なシナリオで、ロードバランサーからノードをデタッチしないことを示します。デフォルト値は `true` です

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

detach.on.agent.failure=
```

`script.fence` プロパティは、スタンバイノードからプライマリノードへの昇格中に呼び出すオプションのユーザー指定スクリプトへのパスを指定します。

```
# absolute path to fencing script run during promotion
#
# This is an optional user-supplied script that will be run
# during failover on the standby database node. If left blank,
# no action will be taken. If specified, EFM will execute this
# script before promoting the standby.
#
# Parameters can be passed into this script for the failed primary
# and new primary node addresses. Use %p for new primary and %f
# for failed primary. On a node that has just been promoted, %p
```

(次のページに続く)

(前のページからの続き)

```
# should be the same as the nodes efm binding address.

#
# Example:

# script.fence=/somepath/myscript %p %f

#
# NOTE: FAILOVER WILL NOT OCCUR IF THIS SCRIPT RETURNS A NON-ZERO EXIT
# CODE.

script.fence=
```

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

```
# Absolute path to fencing script run after promotion

#
# This is an optional user-supplied script that will be run after
# failover on the standby node after it has been promoted and
# is no longer in recovery. The exit code from this script has
# no effect on failover manager, but will be included in a
# notification sent after the script executes.

#
# Parameters can be passed into this script for the failed primary
# and new primary node addresses. Use %p for new primary and %f
# for failed primary. On a node that has just been promoted, %p
# should be the same as the nodes efm binding address.

#
# Example:

# script.post.promotion=/somepath/myscript %f %p

script.post.promotion=
```

`script.resumed` プロパティを使用して、エージェントがデータベースの監視を再開したときに呼び出す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に障害が発生したことをエージェントが検出した場合に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.primary.isolated` プロパティを使用して、プライマリデータベースを監視しているエージェントが、プライマリが Failover Manager 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 Failover Manager 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 が解放された直後に呼び出されます（VIP が使用中の場合）。

```
# Absolute path to script run on isolated primary

# This is an optional user-supplied script that will be run after

# a primary agent detects that it has been isolated from the

# majority of the efm cluster.

script.primary.isolated=
```

`script.remote.pre.promotion` プロパティを使用して、ノードがデータベースをプライマリに昇格しようとしているときに、昇格に関与していないエージェントノードで呼び出す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
```

(次のページに続く)

(前のページからの続き)

```
# 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、プロモーションが発生した後に非プライマリノードで呼び出すスクリプトのパスと名前を指定します。

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

```
# Absolute path to script invoked on non-primary agent nodes

# after a promotion.

#
# This optional user-supplied script will be invoked on nodes
# (except the new primary) after a promotion occurs. The exit code
# from this script has no effect on Failover Manager, but will be
# included in a notification sent after the script executes.

#
# Pass a parameter (%p) with the script to identify the new
```

(次のページに続く)

(前のページからの続き)

```
# primary node address.  
  
#  
# Example:  
  
# script.remote.post.promotion=/path_name/script_name %p  
  
script.remote.post.promotion=
```

`script.custom.monitor` プロパティを使用して、`custom.monitor.interval` プロパティで秒単位で指定された定期的な間隔で呼び出すオプションのスクリプトの名前と場所を提供します。

`custom.monitor.timeout` を使用して、スクリプトを実行する最大時間を指定します。スクリプトの実行が指定された時間内に終了しない場合、フェールオーバーマネージャーは通知を送信します。

`custom.monitor.safe.mode` を `true` に設定して、スクリプトからゼロ以外の終了コードを報告するが、終了コードの結果としてスタンバイをプロモートしないようにフェールオーバーマネージャーに指示します。

```
# Absolute path to a custom monitoring script.  
  
#  
# Use script.custom.monitor to specify the location and name of  
  
# an optional user-supplied script that will be invoked  
  
# periodically to perform custom monitoring tasks. A non-zero  
  
# exit value means that a check has failed; this will be treated  
  
# as a database failure. On a primary node, script failure will  
  
# cause a promotion. On a standby node script failure will  
  
# generate a notification and the agent will become IDLE.  
  
#  
# The custom.monitor.* properties are required if a custom  
  
# monitoring script is specified:  
  
#  
# custom.monitor.interval is the time in seconds between executions  
  
# of the script.
```

(次のページに続く)

(前のページからの続き)

```
#
# 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呼び出すフェールオーバーマネージャーの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呼び出すフェールオーバーマネージャーの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>
```

(次のページに続く)

(前のページからの続き)

```
# 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の場所を指定します。このファイルは、フェールオーバーマネージャーが、ノード上の単一のクラスターに対して、孤立した可能性のある複数のエージェントを起動できないようにします。

```
# 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を書き込む場所を指定します。フェールオーバーマネージャーは、ディレクトリが存在しない場合、ディレクトリを作成しようとします。

```
# Specify the directory of agent logs on the node. If the path
# does not exist, Failover Manager will attempt to create it. If
# not specified defaults to /var/log/efm-<version>. (To store
# Failover Manager startup logs in a custom location, modify the
# path in the service script to point to an existing, writable
# directory.)
```

(次のページに続く)

(前のページからの続き)

```
# If using a custom log directory, you must configure

# logrotate separately. Use man logrotate for more information.

log.dir=
```

フェールオーバーマネージャーホストで UDP または TCP プロトコルを有効にした後、syslog へのロギングを有効にできます。syslog.protocol パラメーターを使用してプロトコルタイプ (UDP または TCP) を指定し、syslog.port パラメーターを使用して syslog ホストのリスナーポートを指定します。syslog.facility 値を、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。LOCAL0~LOCAL7 の値を使用してください。

```
# 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 ロギングの構成の詳細については、[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、フェールオーバーマネージャーによってログに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。ログの詳細については、[Controlling logging](#) を参照してください。

```
# Logging levels for JGroups and EFM.

# Valid values are: TRACE, DEBUG, INFO, WARN, ERROR

# Default value: INFO

# It is not necessary to increase these values unless debugging a
```

(次のページに続く)

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

```
# Extra information that will be passed to the JVM when starting
# the agent.

jvm.options=-Xmx128m
```

`encrypting_database_password`

## 8.2 Encrypting your database password

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。`/usr/edb/efm-4.<x>/bin` ディレクトリにある [Using the efm utility](#) を使用して、パスワードを暗号化します。パスワードを暗号化する場合、ユーティリティを呼び出すときにコマンドラインでパスワードを渡すか、`EFMPASS` 環境変数を使用できます。

パスワードを暗号化するには、次のコマンドを使用します。

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

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

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

```
export EFMPASS=password
```

`--from-env` オプションを含めない場合、フェールオーバーマネージャーは、クラスタープロパティファイルに配置する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する場合、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。

**注釈:** 多くの Java ベンダーは、完全な強度の暗号化を含む Java のバージョン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無制限のポリシーを提供する Java 暗号化拡張機能 (JCE) をダウンロードして有効にします。

この例では、`encrypt` 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ます。

```
# efm encrypt acctg

This utility will generate an encrypted password for you to place in
your Failover Manager cluster property file:
/etc/edb/efm-4.7/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
```

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

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て、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 ~]# systemctl start edb-efm-4.7
Job for edb-efm-4.7.service failed because the control process exited with error code.
See "systemctl status edb-efm-4.7.service" and "journalctl -xe" for details.
```

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-4.7/startup-efm.log` を参照してください。

RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x を使用している場合、次のコマンドで起動情報も利用できます。

```
systemctl status edb-efm-4.7
```

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に、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。

### 8.2.1 EFMPASS 環境変数の使用

この例では、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 `7ceecd8965fa7a5c330eaa9e43696f83` はテキスト値として返されます。スクリプトを使用する場合、コマンドの終了コードを確認して、コマンドが成功したことを確認できます。実行が成功すると、 `0` が返されます。

## 8.3 The cluster members file

フェールオーバーマネージャークラスター内の各ノードには、現在のフェールオーバーマネージャー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。エージェントは、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。フェールオーバーマネージャーストローラーは、 `/etc/edb/efm-4.<x>` ディレクトリに `efm.nodes.in` という名前のクラスターメンバーファイルのファイルテンプレートを作成します。

Failover Manager 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します。

```
cp /etc/edb/efm-4.7/efm.nodes.in /etc/edb/efm-4.7/efm.nodes
```

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

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

デフォルトでは、フェールオーバーマネージャークラスターメンバーファイルの名前が `efm.nodes` であることを想定しています。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、フェールオーバーマネージャースerviceスクリプトを変更して、新しい名前を使用するようにフェールオーバーマネージャークラスターに指示します。

起動した最初のノードのクラスターメンバーファイルは空でもかまいません。このノードは、メンバーシップコーディネーターになります。後続の各ノードでは、クラスターメンバーファイルには、少なくとも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:ポートの形式でリストする必要があります。複数のエントリはスペースで区切ります。

エージェントは、クラスターの現在のメンバーと一致するように `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離脱すると、他のエージェントの `efm.nodes` ファイルが更新され、現在のクラスターメンバーシップが反映されます。

*efm stop-cluster* コマンドを呼び出した場合、フェールオーバーマネージャーはファイルを変更しません。注：エージェントは、起動時に他のメンバーを検出するために独自の場所を必要としないため、独自の情報をファイルに書き込みません。

メンバーシップコーディネーターがクラスターを離れると、別のノードがロールを引き受けます。

*efm cluster-status* コマンドを使用して、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ことができます。エージェントのダウン中にノードがクラスターに参加または脱退する場合、そのエージェントを起動する前に、ファイルに少なくとも現在のメンバーシップコーディネーターのアドレスとポートが含まれていることを手動で確認します。

クラスターに参加しているノードのアドレスとポートがわかっている場合は、いつでもクラスターメンバーファイルにアドレスを含めることができます。

*The cluster properties file* の `auto.allow.hosts` プロパティが `true`

に設定されていない限り、起動時にクラスターメンバーを識別しないアドレスは無視されます。

*The cluster properties file* ) の `stable.nodes.file` プロパティが `true`

に設定されている場合、クラスターメンバーがクラスターに参加または離脱するときに、エージェントは `.nodes` ファイルを更新しません。この動作は、クラスターメンバーの IP アドレスが頻繁に変更されない場合に最も役立ちます。

## 8.4 Extending Failover Manager permissions

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` スクリプトを呼び出します。
- Pgpool 統合を有効にすると、`efm` は `efm_pgpool_functions` スクリプトを呼び出します。

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

`sudoers` ファイルには、ユーザー `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ようにするエントリが含まれています。`sudoers` ファイルのコピーを変更して、他のユーザーが所有する Postgres クラスターを管理する権限を `efm` に付与できます。

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

```
# Copyright EnterpriseDB Corporation, 2014-2021. All Rights Reserved.
#
# Do not edit this file. Changes to the file may be overwritten
# during an upgrade.
```

(次のページに続く)

```
#
# This file assumes you are running your efm cluster as user efm.  If not,
# then you will need to copy this file.

# Allow user efm to sudo efm_db_functions as either postgres or enterprisedb.

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

efm    ALL=(postgres)      NOPASSWD:  /usr/edb/efm-4.7/bin/efm_db_functions
efm    ALL=(enterprisedb)  NOPASSWD:  /usr/edb/efm-4.7/bin/efm_db_functions

# Allow user efm to sudo efm_root_functions as root to write/delete the PID file,
# validate the db.service.owner property, etc.

efm    ALL=(ALL)           NOPASSWD:  /usr/edb/efm-4.7/bin/efm_root_functions

# Allow user efm to sudo efm_address as root for VIP tasks.

efm    ALL=(ALL)           NOPASSWD:  /usr/edb/efm-4.7/bin/efm_address

# Allow user efm to sudo efm_pgpool_functions as root for pgpool tasks.

efm    ALL=(ALL)           NOPASSWD:  /usr/edb/efm-4.7/bin/efm_pgpool_functions

# relax tty requirement for user efm

Defaults:efm !requiretty
```

フェールオーバーマネージャーを使用して、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-42 ファイルのコピーを作成し、コンテンツを変更して、ユーザーが efm\_functions スクリプトにアクセスしてクラスターを管理できるようにします。

パーミッションの問題が原因でエージェントを開始できない場合は、デフォルトの/etc/sudoers ファイルの最後に次の行が含まれていることを確認してください。

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

# includedir /etc/sudoers.d
```

### 8.4.1 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

デフォルトでは、フェールオーバーマネージャーは `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、次の操作には引き続きルートアクセスが必要です。

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

`sudo` なしで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特権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（たとえば、`enterprisedb` または `postgres`）または別の特権ユーザーです。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
```

このコマンドにより、ユーザーは `/var/run/efm-4.<x>` および `/var/lock/efm-4.<x>` 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によって作成されたログファイルに書き込むことができません。

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

```
su - enterprisedb

cp /etc/edb/efm-4.7/efm.properties.in <directory/cluster_name>.properties

cp /etc/edb/efm-4.7/efm.nodes.in <directory>/<cluster_name>.nodes
```

次に、クラスタープロパティファイルを変更し、`db.service.owner` プロパティに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してください。sudo がないと、ルートアクセスなしでサービスを実行できません。

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

```
/usr/edb/efm-4.7/bin/runefm.sh start|stop <directory/cluster_name>.properties
```

`<directory/cluster_name.properties>` は、クラスタープロパティファイルのフルパスを指定します。ユーザーは、デフォルト以外のユーザーがエージェントを制御するか、`efm` スクリプトを使用するたびに、プロパティファイルへのフルパスを提供します。

新しいユーザーがフェールオーバーマネージャー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します。

フェールオーバーマネージャーは、`/usr/edb/efm-4.<x>/bin/secure/` にある `manage-vip` という名前のバイナリを使用して、`sudo` 特権なしで VIP 管理操作を実行します。このスクリプトは、`setuid` を使用して、仮想 IP アドレスを管理するための権限を取得します。

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

セキュリティ上の理由から、`/usr/edb/efm-4.<x>/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 サイトにアクセスしてください。

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

## 8.5 Using Failover Manager with virtual IP addresses

フェールオーバーマネージャーは `efm_address` スクリプトを使用して、仮想 IP アドレスを割り当てまたは解放します。

---

**注釈:** 仮想 IP アドレスは、多くのクラウドプロバイダーでサポートされていません。これらの環境では、AWS の Elastic IP アドレスなど、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る別のメカニズムを使用します。

---

デフォルトでは、スクリプトは次の場所にあります。

`/usr/edb/efm-4.<x>/bin/efm_address`

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

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>
```

そこで：

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

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

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

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

root ユーザーとして `efm_address` スクリプトを呼び出します。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特権が付与されます。`sudoers` ファイルの詳細については、[Extending Failover Manager permissions](#) を参照してください。

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

### 8.5.1 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。VIP のテスト中に、フェールオーバーマネージャーが実行されていないことを確認します。

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。

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

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

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

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

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

100 %のパケット損失が表示されます。!!!重要フェールオーバーマネージャーは、ping コマンドの終了コードを使用して、アドレスに到達可能かどうかを判断します。この場合、終了コードはゼロではありません。ping 以外のコマンドを使用している場合、アドレスに到達できない場合、ゼロ以外の終了コードを返す必要があります。

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

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

3. 他のノードから VIP に ping を実行して、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
```

パケットロスが発生しません。!!!重要 フェールオーバーマネージャーは、ping コマンドの終了コードを使用して、アドレスに到達可能かどうかを判断します。この場合、終了コードはゼロです。ping 以外のコマンドを使用している場合、アドレスに到達可能な場合、ゼロの終了コードを返す必要があります。

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

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

この手順の出力には、eth0 インターフェイスが表示されません。

5. 手順3を繰り返します。今回は、スタンバイと監視に使用中のVIPが表示され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%のパケットロスが発生。すべてのノードでこの手順を繰り返します。

6. すべてのスタンバイノードで手順2を繰り返し、VIPをすべてのノードに割り当てます。任意のノードからVIPにpingを実行して、それが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
```

(次のページに続く)

(前のページからの続き)

```
inet addr:172.24.38.239 Bcast:172.24.38.255
...
```

これらのテスト手順の後、フェールオーバーマネージャーを起動する前に、非プライマリノードから VIP を解放します。

**注釈:** 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。プライマリエージェントは、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の障害は、一次分離とフェールオーバーにつながります。

VIP が別のインターフェイスを使用している場合、プライマリエージェントがドロップする前に、残りのクラスターが到達可能な VIP をチェックするタイミング条件が発生する場合があります。この場合、フェールオーバーマネージャーは `node.timeout` プロパティで指定された秒数だけ VIP チェックを再試行して、フェールオーバーが期待どおりに発生することを確認します。

## 8.6 Configuring for Eager Failover

デフォルトの実行モードでは、プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動するまでフェールオーバー保護はありません。このケースを回避するには、`systemd` を介してプライマリノードを設定して、プライマリエージェントが終了したときにフェールオーバーを発生させることができます。これを Eager Failover と呼びます。

次の手順を実行して、Eager Failover をセットアップできます。この例では、EDB Postgres Advanced Server バージョン 12 と Failover Manager バージョン 4.5 を使用しています。

### 8.6.1 Eager フェールオーバーを有効にする

- Failover Manager エージェントが停止または失敗するとすぐにデータベースサーバーが停止するため、Failover Manager を起動する前に、すべてのエージェントに次のプロパティを設定する必要があります。

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

フェールオーバーマネージャーを起動する前にこのプロパティを設定しない場合、フェールオーバーマネージャーエージェントをシャットダウンすると、フェールオーバーなしでデータベースがシャットダウンされます。

- Eager Failover を有効にした状態で、`efm stop-cluster` コマンドを使用すると、すべての Failover Manager エージェントが停止し、プライマリデータベースがシャットダウンされます。エージェントが実行されていないため、フェールオーバーはありません。このシナリオを回避するには、`enable.stop.cluster` プロパティを使用してコマンドを無効にします。

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

- データベースサーバーとローカル Failover Manager エージェントが実行されていることを確認します。
- root として、`/etc/systemd/system/edb-as-12.service` ファイルを作成し、以下を含めます。

```
.include /lib/systemd/system/edb-as-12.service
[Unit]
BindsTo=edb-efm-4.5.service
```

- 次のコマンドを実行して、構成ファイルをリロードします。

```
systemctl daemon-reload
```

これらの変更により、Failover Manager エージェントが停止または終了すると、クラスターの残りの部分はこの状況を障害として扱い、フェールオーバーを試みます。

### 8.6.2 Eager Failover の無効化

- データベースを停止せずにフェールオーバーマネージャーを停止する場合は、`/etc/systemd/system/edb-as-12.service` の次の行をコメントアウトします。

```
BindsTo=edb-efm-4.5.service
```

- 次のコマンドを実行して、構成ファイルをリロードします。

```
systemctl daemon-reload
```

### 8.6.3 Eager Failover モードでの Failover Manager のアップグレード

EDB Postgres Advanced Server を停止せずにフェールオーバーマネージャーをアップグレードするには、Eager Failover モードを一時的に無効にします。

1. Disable Eager Failover
2. *Stop and upgrade Failover Manager*
3. Enable Eager Failover

### 8.6.4 重要な注意事項

- 非 `sudo` セットアップでフェールオーバーマネージャーを実行中に `systemd` コマンドを使用してデータベースを管理しないため、Eager Failover は `sudo` モードでのみサポートされます。非 `sudo` モードではサポートされていません。
- Eager Failover は、古いプライマリによって VIP が解放されない状況には適していません。
- Eager Failover は次の状況に適しています。
- EDB Postgres Advanced Server の高可用性セットアップを使用。 -  
`jdbc` または `libpq` (`target-session-attrs`) によるクライアント接続フェイルオーバーを使用したセットアップ。

- `script.fence`

によってトリガーされるカスタムスクリプトが古いプライマリサーバー（STONITH）をフェンスする場合。いくつかの例は、VMWare vCenter 統合、openstack 統合、または Lights-Out Management を使用して VM をシャットダウンすることです。- `script.fence` によってトリガーされるカスタムスクリプトが `ssh` を使用して VIP を非アクティブ化する場合。

---

**注釈:** `check.vip.before.promotion=false` 設定は、古いプライマリが VIP を解放する前に新しいプライマリが VIP をアタッチできるようにするために必要です。

---

- `primary.shutdown.as.failure=true` を使用するときには注意してください。必要に応じてデータベースを安全に停止する方法については、

`primary.shutdown.as.failure` プロパティの説明を参照してください。

- フェールオーバーのたびに、プライマリは最終的に障害が発生したプライマリになり、運用スタンバイとして自動的に回復しません。したがって、クラスターに複数の昇格可能なスタンバイが含まれ、スタンバイの合計数が `minimum.standbys` で指定された値よりも少なくとも 2 多いことを確認してくださいプロパティ。これは一般的な推奨事項ですが、Eager Failover を使用する場合、より緊急になります。
- データベースサーバーが停止している場合、データベースを再起動すると Failover Manager も起動します。

---

**注釈:** -不良プロパティ値など、フェールオーバーマネージャーの起動に問題がある場合、データベースサーバーは実行されていないという警告を表示せずに再起動およびシャットダウンします。-フェールオーバーマネージャープロセスが以前に終了した場合、ロックファイルはまだ存在し、エージェントは自動的に再起動できません。- データベースサーバーまたは Failover Manager エージェントの起動時に問題が発生した場合は、Failover Manager の起動ログで情報を確認します。

---

- `stop-cluster` コマンドを実行した結果、すべてのノードで Failover Manager が停止します。Eager Failover モードでは、`stop-cluster` コマンドはフェールオーバーなしで EDB Postgres Advanced Server も停止します。 `enable.stop.cluster=false` を設定して、`stop-cluster` コマンドが誤って呼び出されないようにします。



## 第9章 Configuring SSL authentication on a Failover Manager cluster

フェールオーバーマネージャーの SSL 認証を有効にできます。接続するすべてのクライアントは、クラスター内のデータベースサーバーに接続するときに SSL 認証を使用する必要があります。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：

1. `server.crt` および `server.key` ファイルを EDB Postgres Advanced Server インストールの下での `data` ディレクトリに配置します。当局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、

[PostgreSQL core documentation](#) を参照してください。

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

```
ssl=on
```

`postgresql.conf` ファイルを変更した後、サーバーを再起動する必要があります。

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

```
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にサーバーに指示しています。これにより、接続するクライアントに 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、[PostgreSQL core documentation](#) を参照してください。

1. `server.crt` および `server.key` ファイルをデータディレクトリに配置した後、証明書を Java が理解できる形式に変換します。次のコマンドを使用できます。

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

詳細については、[Postgres JDBC documentation](#) を参照してください。

1. 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
```

ここで: `$JAVA_HOME` は、Java インストールのホームディレクトリです。`alias_name` には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、使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、次のように入力してください。

`edb_notranlate_4` 各データベースサーバーからの証明書は、各エージェントの信頼できる証明書ファイルにインポートする必要があります。`cacerts` ファイルの場所は、システムごとに異なる場合があります。詳細については、

[Postgres JDBC documentation](#) を参照してください。

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

## 第10章 Using Failover Manager

フェールオーバーマネージャーは、1つ以上のスタンバイサーバーを含むクラスターの監視とフェールオーバーのサポートを提供します。リソースの需要が増減すると、クラスターにノードを追加または削除できます。

プライマリノードが再起動すると、フェールオーバーマネージャーは、プライマリノードでデータベースがダウンしていることを検出し、スタンバイノードをプライマリのロールに昇格させる場合があります。これが発生した場合、リブートされたプライマリノードの Failover Manager エージェントは、`recovery.conf` ファイルを書き込んで、Postgres がセカンダリプライマリとして起動しないことを確認しようとします。したがって、データベースサーバーを起動する前に Failover Manager エージェントを起動する必要があります。エージェントはアイドルモードで起動し、クラスター内に既にプライマリがあるかどうかを確認します。プライマリノードがある場合、エージェントは `recovery.conf` または `standby.signal` ファイルが存在することを確認するか、必要に応じて `recovery.conf` を作成して、データベースが2番目のプライマリとして起動しないようにします。

### 10.1 フェールオーバーマネージャークラスターの管理

構成が完了すると、フェールオーバーマネージャークラスターに定期的なメンテナンスは必要ありません。ただし、フェールオーバーマネージャークラスターが時々必要とする管理タスクを実行できます。

デフォルトでは、

*some of the efm commands* は `efm` または OS スーパーユーザーが呼び出す必要があります。管理者は、ユーザーを `efm` グループに追加することにより、ユーザーがこれらのコマンドを呼び出すことを選択的に許可できます。コマンドは次のとおりです。

- *efm allow-node*
- *efm disallow-node*
- *efm プロモート*
- *efm 履歴書*
- *efm set-priority*
- *efm stop-cluster*
- *efm upgrade-conf*

### 10.1.1 フェールオーバーマネージャークラスターの開始

フェールオーバーマネージャークラスターのノードは、任意の順序で起動できます。

RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.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 edb-efm-4.<x>
```

ノードのクラスタープロパティファイルで `is.witness` が `true` であることが指定されている場合、ノードは witness ノードとして起動します。

ノードが専用の監視ノードでない場合、フェールオーバーマネージャークラスタ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` に応答した場合、エージェントはノードがプライマリノードであると想定し、該当する場合はノードに仮想 IP アドレスを割り当てます。サーバーが `true` に応答すると、Failover Manager エージェントはノードがスタンバイサーバーであると想定します。サーバーが応答しない場合、エージェントはアイドル状態で起動します。

クラスターに参加した後、Failover Manager エージェントは、指定されたデータベースの資格情報をチェックして、クラスター内のすべてのデータベースに接続できることを確認します。エージェントが接続できない場合、エージェントはシャットダウンします。

新しいプライマリまたはスタンバイノードがクラスターに参加すると、既存のすべてのノードも新しいノードのデータベースに接続できることを確認します。

---

**注釈:** `tmpfs`（一時ファイルシステム）で `/var/lock` または `/var/run` を実行している場合、フェールオーバーマネージャークラスターの `systemd` サービスファイルが `systemd-tmpfiles-setup.service` に依存していることを確認してください。

---

### 10.1.2 クラスターへのノードの追加

いつでもノードをフェールオーバーマネージャークラスターに追加できます。クラスターにノードを追加するときは、新しいノードを許可するようにクラスターを変更し、新しいノードにクラスターを見つける方法を指示する必要があります。

1. `auto.allow.hosts` が `true` に設定されていない限り、`efm allow-node` コマンドを使用して、新しいノードのアドレスを 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アドレスを指定します。

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

`efm allow-node` コマンドの使用またはフェールオーバーマネージャークラスターの制御の詳細については、

*Using the efm utility* を参照してください。

Failover Manager エージェントをインストールし、新しいノードにクラスタープロパティファイルを構成します。プロパティファイルの変更の詳細については、

*The cluster properties file* を参照してください。

2. 新しいノードでクラスターメンバーファイルを構成し、メンバーシップコーディネーターのエントリーを追加します。クラスターメンバーファイルの変更の詳細については、

*The cluster members file* を参照してください。

3. 新しいノードでスーパーユーザー権限を想定し、Failover Manager エージェントを起動します。RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x で Failover Manager クラスターを起動するには、次のコマンドを呼び出します。

```
systemctl start edb-efm-4.<x>
```

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` プロパティで指定された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。

---

**注釈:** 現在のノードの有用なスタンバイになるには、ノードが PostgreSQL ストリーミングレプリケーションシナリオでスタンバイである必要があります。

---

### 10.1.3 スタンバイの優先順位の変更

フェールオーバーマネージャー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順位の値を指定します。

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
```

スタンバイの優先度を `0` に設定して、スタンバイを昇格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、`promotable=false` のプロパティ値がオーバーライドされます。

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` の昇格優先度をフェールオーバーのイベントに使用されるスタンバイに設定する場合、`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オーバーライドします。

```
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最初に Postgres ストリーミングレプリケーションから情報を取得して、どのスタンバイノードに最新のデータがあるかを確認し、データ損失の可能性が最も少ないノードを昇格させます。2つのスタンバイノードに同等の最新のデータが含まれる場合、

*use.replay.tiebreaker* が `true`

に設定されていない限り、ユーザー指定の優先度値が高いノードがプライマリに昇格します。スタンバイノードのプライオリティ値を確認するには、次のコマンドを使用します。

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

**注釈:** 新しいプライマリが昇格すると、ノードの昇格優先度を変更されます。

---

`efm set-priority` コマンドを使用してスタンバイが昇格可能かどうかを変更した場合、昇格またはクラスター分割と再参加を介してスタンバイのプロパティファイルの値にリセットできます。エージェントが再起動されると、昇格可能なステータスはプロパティファイルの値に戻ります。

### 10.1.4 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プライマリデータベースへの手動プロモーションを開始できます。

データベースクラスターのメンテナンスウィンドウ中にのみ手動プロモーションを実行します。最新のスタンバイデータベースを利用できない場合、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、次のコマンドを呼び出します。

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

そこで：

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

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

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

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

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

スイッチオーバー中：

- サーバーバージョン 11 以前の場合、`recovery.conf` ファイルは既存のスタンバイからプライマリノードにコピーされます。サーバーバージョン 12 以降では、`primary_conninfo` および `restore_command` パラメーターがコピーされ、メモリに保存されます。
- プライマリデータベースは停止しています。
- VIP を利用している場合、プライマリノードからアドレスが解放されます。
- プライマリノードに代わってスタンバイが昇格し、VIP を取得します。
  - 新しいプライマリノードのアドレスが `recovery.conf` ファイルに追加されるか、`primary_conninfo` の詳細がメモリに格納されます。
  - 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` 情報がメモリに格納されます。
  - サーバーバージョン 12 以降を使用している場合、メモリに保存されたりリカバリ設定は `postgresql.auto.conf` ファイルに書き込まれます。`standby.signal` ファイルが作成されます。

- 古いプライマリが起動されます。エージェントはスタンバイとして監視を再開します。

プロモーション中に、プライマリエージェントは仮想 IP アドレスを解放します。スイッチオーバーでない場合、`db.data.dir` プロパティで指定されたディレクトリに `recovery.conf` ファイルが作成されます。`recovery.conf` ファイルは、ファイルが削除されるまで古いプライマリデータベースが起動しないようにし、ノードがクラスター内の 2 番目のプライマリとして起動しないようにするために使用されます。プロモーションがスイッチオーバーの一部である場合、リカバリ設定は上記のように処理されます。

プライマリエージェントは実行されたままで、ステータスが `Idle` になります。

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

ノードをプライマリのロールに戻すには、ノードをプロモーションリストの最初に配置します。

```
efm set-priority <cluster_name> <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、[Using the efm utility](#) を参照してください。

### 10.1.5 Failover Manager 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、クラスター内のすべての実行ノードのクラスターメンバーリストからノードのアドレスを削除しますが、フェールオーバーマネージャーの許可ノードホストリストからはアドレスを削除しません。

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

```
systemctl stop edb-efm-4.<x>
```

`efm disallow-node` コマンド（許可ノードホストリストからノードのアドレスを削除）を呼び出すまでは、最初に `efm allow-node` コマンドを再度実行せずに、`service edb-efm-4.<x> start` コマンドを使用してノードを再起動できます。

エージェントを停止しても、`primary.shutdown.as.failure` プロパティが `true` に設定されていない限り、エージェントに障害が発生したことをクラスターに通知しません。

### 10.1.6 フェールオーバーマネージャークラスターの停止

フェールオーバーマネージャークラスターを停止するには、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に接続し、efm または OS スーパーユーザーの ID を引き継ぎ、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、すべての Failover Manager エージェントが終了します。フェールオーバーマネージャークラスターを終了すると、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。

---

**注釈:** *efm stop-cluster* コマンドを呼び出すと、許可されたノードのすべての情報が許可ノードホストリストから失われます。

---

### 10.1.7 クラスターからのノードの削除

*efm disallow-node* コマンドは、フェールオーバーマネージャークラスターの許可ノードホストリストからノードの IP アドレスを削除します。現在実行中のクラスターの一部である既存のノードで、efm または OS スーパーユーザーの ID を想定します。次に、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> <address>
```

*efm disallow-node* コマンドは、実行中のエージェントを停止しません。サービスは、

*stop the agent*

までノードで実行され続けます。エージェントまたはクラスターが後で停止されると、ノードはクラスターに再参加できず、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* コマンドを使用して、ノードをクラスターに再度追加する必要があります。

## 10.2 単一のノードで複数のエージェントを実行する

その Failover Manager ノードで複数のプライマリまたはスタンバイエージェントを実行することにより、同じホスト上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監視エージェントを実行することもできます。異なるクラスターのフェールオーバーマネージャークラスターが互いに干渉しないようにしながら、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クラスターを構成するには:

1. 各クラスターのメンバーごとに、クラスター内のノードの一意のプロパティセットとロールを定義するクラスタープロパティファイルを作成します。
2. 各クラスターのメンバーごとに、クラスターのメンバーをリストするクラスターメンバーファイルを作成します。
3. 各クラスターのユニットファイル (RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.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。

これらの例では、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で 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、一意の名前を付ける必要があります。この例では、acctg.properties および sales.properties を作成して、acctg および sales データベースクラスターに一致します。

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

admin.port

bind.address

db.port

db.data.dir

virtual.ip（使用する場合）

db.service.name（使用する場合）

各クラスタープロパティファイルで、db.port パラメーターは、クラスターごとに一意の値を指定します。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 ファイルでは以下を指定できます。

db.user=efm\_user

db.password.encrypted=7c801b32a05c0c5cb2ad4ffbda5e8f9a

db.port=5444

db.database=acctg\_db

sales.properties ファイルでは以下を指定できますが、

db.user=efm\_user

db.password.encrypted=e003fea651a8b4a80fb248a22b36f334

db.port=5445

db.database=sales\_db

同じノードに複数のフェールオーバーマネージャークラスターエージェントを設定する場合、一部のパラメーターには特別な注意が必要です。複数のエージェントが同じノードに存在する場合、各ポートは一意である必要があります。どの 2 つのポートでも機能しますが、互いに近すぎないポートを使用すると、情報を明確にしておく方が簡単です。

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とき、db.data.dir パラメーターは、それぞれのデータベースクラスターに対して一意の値を指定する必要もあります。

ノードに仮想 IP アドレスを割り当てる場合、以下のパラメーターを使用します。フェールオーバーマネージャークラスターが仮想 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。

`virtual.ip`

`virtual.ip.interface`

`virtual.ip.prefix`

このパラメーター値は、使用されている仮想 IP アドレスによって決定され、`acctg.properties` と `sales.properties` の両方で同じにすることができます。

`acctg.properties` および `sales.properties` ファイルを作成した後、それぞれのプロパティファイルを指すクラスターごとにサービススクリプトまたはユニットファイルを作成します。この手順はプラットフォーム固有です。RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x を使用している場合、

*RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x を参照してください。*

---

**注釈:** ユニットファイルを使用している場合は、フェールオーバーマネージャークラスターをアップグレードするときに、ファイルを手動で更新して新しいサービス名を反映します。

---

### 10.2.1 RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x

RHEL/CentOS 7.x または RHEL/Rocky Linux/AlmaLinux 8.x を使用している場合、`edb-efm-4.<x>` ユニットファイルを、クラスターごとに一意の名前で新しいファイルにコピーします。たとえば、`acctg` と `sales` という 2 つのクラスターがある場合、ユニットファイル名は次のようになります。

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

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

次に、各ユニットファイルの `CLUSTER` 変数を編集し、指定したクラスター名を `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します。例：

```
PIDFile=/var/run/efm-4.7/acctg.pid
```

サービススクリプトをコピーした後、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、`acctg` エージェントを起動するには：

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

ユニットファイルのカスタマイズの詳細については、*Understanding and administering systemd* を参照してください。



## 第11章 Using the efm utility

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 のインストール時に、ユーティリティを `/usr/edb/efm-4.<x>/bin` ディレクトリに追加します。

### 11.1 efm allow-node

```
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。

### 11.2 efm disallow-node

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

### 11.3 efm cluster-status

```
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ステータスを表示します。ステータスレポートの詳細については、

*Monitoring a Failover Manager cluster* を参照してください。

## 11.4 efm cluster-status-json

`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ステータスを JSON 形式で表示します。`efm cluster-status` コマンドで生成される表示とは、表示される情報の形式は異なりますが、情報源は同じです。

次の例は、3 つのノードを持つ正常なクラスターのステータスをクエリすることによって生成されます。

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

(次のページに続く)

(前のページからの続き)

```
"minimumstandbys": 0,  
"missingnodes": [],  
"messages": []  
}
```

## 11.5 efm encrypt

`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、

*Encrypting your database password* を参照してください。

## 11.6 efm プロモート

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

`efm promote` コマンドは、スタンバイからプライマリへの手動フェールオーバーを実行するようにフェールオーバーマネージャーに指示します。

データベースクラスターのメンテナンスウィンドウ中に、クラスター内のすべてのスタンバイがプライマリと最新の状態であることが `status` コマンドによって報告された場合にのみ、手動昇格を試みます。

`-switchover` 句を含めて、スタンバイノードを昇格させ、プライマリノードをスタンバイノードとして再構成します。`-sourcenode` キーワードを含め、ノードアドレスを指定して、リカバリ設定を古いプライマリノードにコピーして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には、`-quiet` キーワードを含めます。`-noscripts` キーワードを含めて、フェ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。

---

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

---

## 11.7 efm 履歴書

```
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。

## 11.8 efm set-priority

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

`efm set-priority` コマンドを呼び出して、フェイルオーバー優先度をスタンバイノードに割り当てます。この値は、フェイルオーバーが発生した場合にノードを使用す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` オプションを使用して、優先順位リスト内のノードの場所を指定します。たとえば、1 の値を指定して、ノードがプライマリスタンバイであり、フェイルオーバーが発生した場合に昇格する最初のノードになることを示します。0 の優先順位値は、フェイルオーバーマネージャーにスタンバイを昇格させないように指示します。

## 11.9 efm stop-cluster

```
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イルオーバーマネージャーを停止します。このコマンドは、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に指示するようにフェイルオーバーマネージャーに指示します。このコマンドは実行中のデータベースには影響しませんが、コマンドが完了すると、フェイルオーバー保護は行われません。

---

**注釈:** `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。

## 11.10 efm upgrade-conf

```
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、root 権限で呼び出す必要があります。

`sudo` を使用しないフェイルオーバーマネージャー構成からアップグレードする場合は、`-source` フラグを含め、`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。

## 11.11 efm node-status-json

`efm node-status-json <cluster_name>`

`efm node-status-json` コマンドを呼び出して、ローカルノードのステータスを JSON 形式で表示します。このコマンドの実行が成功すると、終了コードとして `0` が返されます。データベースに障害が発生した場合、またはエージェントのステータスが `IDLE` になった場合、コマンドは終了コードとして `1` を返します。

以下は、`efm node-status-json` コマンドの出力例です。

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

## 11.12 efm --help

`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12章 Monitoring a Failover Manager cluster

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

### 12.1 クラスターステータスレポートの確認

`efm cluster-status` *The cluster properties file* コマンドは、Failover Manager クラスターのステータスに関する情報を含むレポートを返します。

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

次のステータスレポートは、3つのノードが実行されている `edb` という名前のクラスターに関するものです。

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

Allowed node host list:  
172.19.14.9 172.19.12.163 172.19.10.2

Membership coordinator: 172.19.14.9

Standby priority host list:  
172.19.12.163 172.19.10.2

Promote Status:

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

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

クラスターステータスセクションは、クラスターの各ノードにあるエージェントのステータスの概要を提供します。

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

VIP アドレスの後のアスタリスク (\*) は、アドレスが接続に使用できることを示します。VIP アドレスの後にアスタリスクが付いていない場合、アドレスはプロパティファイル内のノードに関連付けられていましたが、アドレスは現在使用されていません。

フェールオーバーマネージャエージェントは、[クラスターの状態] セクションに表示される情報を提供します。

Allowed node host list および Standby priority host list を使用すると、クラスターに参加できるノードと、ノードの昇格順序を簡単に確認できます。メンバーシップコーディネーターの IP アドレスもレポートに表示されます。

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

レポートの Promote Status セクションは、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。このクエリは、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。各データベースへのクエリは異なるタイミングで返されるため、クラスターでストリーミングレプリケーションが正常に動作している場合でも、LSN が一致しない場合があります。レプリケーションの最新のビューを取得するには、プライマリデータベースに接続し、SQL コマンド `SELECT * FROM pg_stat_replication;` を実行します。

```
Promote Status:

DB Type  Address          WAL Received LSN  WAL Replayed LSN  Info
-----
Primary   172.19.14.9      0/40000638        0/40000638
Standby   172.19.12.163    0/40000638        0/40000638
Standby   172.19.10.2      0/40000638        0/40000638
```

データベースがダウンしているか、データベースが再起動されたが resume 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 idle です。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
```

### 12.1.1 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て終了コードを返します。

- 0 の終了コードは、すべてのエージェントが実行されており、プライマリノードとスタンバイノードのデータベースが実行されて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。次の問題は、ゼロ以外の終了コードをトリガーする可能性があります。

データベースがダウンしているか、不明であるか、アイドル状態の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復号化できません。

WAL の場所を取得するためにデータベースに接続する際に問題が発生しました。

一次エージェントはいません。

スタンバイエージェントはありません。

1 つ以上のスタンバイノードがプライマリと同期していません。

## 12.2 Postgres Enterprise Manager を使用したストリーミング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、ストリーミングレプリケーション分析ダッシュボード (PEM インターフェイスの一部) を構成して、ストリーミングレプリケーションシナリオの一部であるプライマリまたはスタンバイノードの状態を表示できます。

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 (レプリケーションプライマリ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。

Replication Slave (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。

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

- ストリーミングレプリケーション
- WAL アーカイブステータス

レプリケーションシナリオのスタンバイノードのストリーミングレプリケーション分析ダッシュボードを表示するには、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ラグタイム

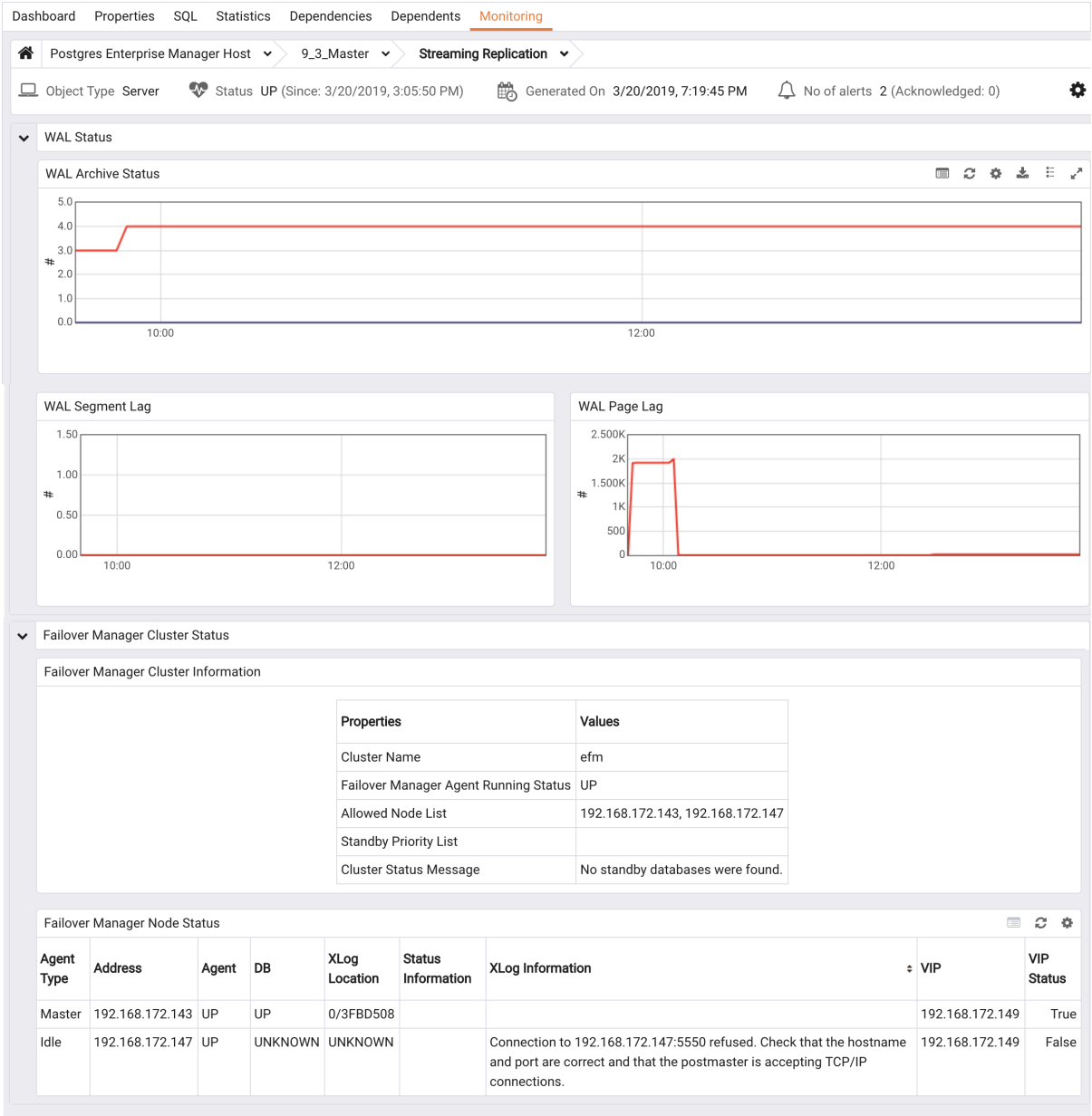
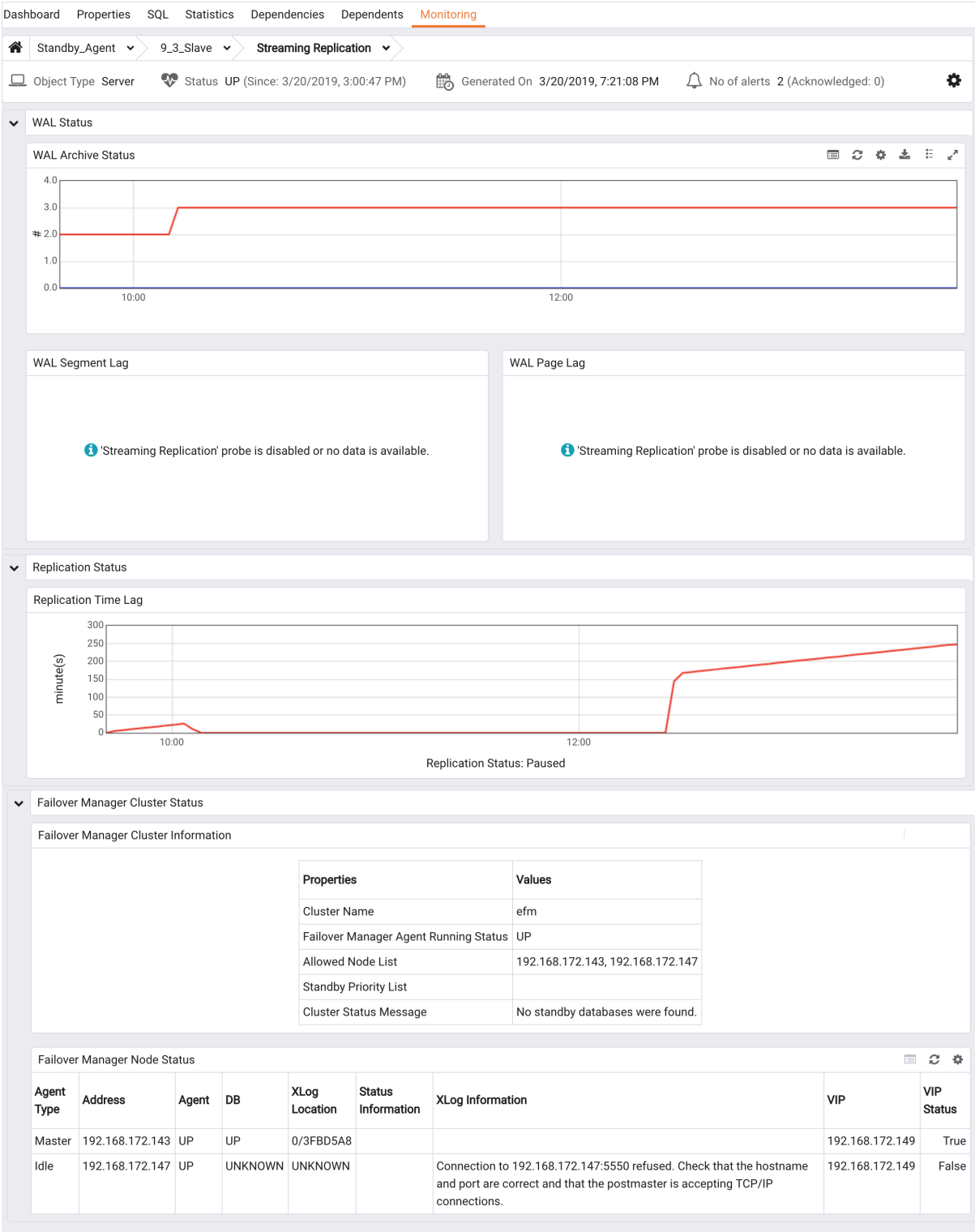


図 1: The Streaming Replication dashboard (Primary node)



PEM の詳細については、 [Postgres Enterprise Manager documentation](#) を参照してください。

## 第13章 Controlling the Failover Manager service

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

- Failover Manager サービスが使用するプロパティを含む **efm.properties** という名前の構成ファイル。レプリケーションシナリオの各ノードには、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は、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ります。

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。

### 13.1 RHEL/CentOS 7.x および RHEL/Rocky Linux/AlmaLinux 8.x で **systemctl** ユーティリティを使用する

RHEL / CentOS 7.x および RHEL / Rocky Linux / AlmaLinux 8.x では、Failover Manager は、`/usr/lib/systemd/system` にある（デフォルトで）**edb-efm-4.<x>.service** という名前の Linux サービスとして実行されます。フェールオーバーマネージャによって監視される各データベースクラスターは、レプリケーションクラスターの各ノードでサービスのコピーを実行します。

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

```
systemctl start edb-efm-4.<x>
```

**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ジェントはローカルデータベースを監視し、他のノード上のフェールオーバーマネージャと通信します。フェールオーバーマネージャークラスター内のノードは、任意の順序で起動できます。このコマンドは、ルートによって呼び出される必要があります。

```
systemctl stop edb-efm-4.<x>
```

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

```
systemctl status edb-efm-4.<x>
```

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、ステータスとサーバーの起動情報を返すようにフェールオーバーマネージャーに指示できます。

```
[root@ONE ~]]> systemctl status edb-efm-4.7
edb-efm-4.7.service - EnterpriseDB Failover Manager 4.7
   Loaded: loaded (/usr/lib/systemd/system/edb-efm-4.7.service; disabled; vendor
   → preset: disabled)
   Active: active (running) since Wed 2013-02-14 14:02:16 EST; 4s ago
   Process: 58125 ExecStart=/bin/bash -c /usr/edb/edb-efm-4.7/bin/runefm.sh start $
   → {CLUSTER} (code=exited, status=0/SUCCESS)
  Main PID: 58180 (java)
    CGroup: /system.slice/edb-efm-4.7.service
            └─ 58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/jre/
   → bin/java -cp /usr/edb/edb-efm-4.7/lib/EFM-4.7.jar -Xmx128m...
```

## 第14章 Controlling logging

フェールオーバーマネージャーは、エージェントごとに1つのログファイルと、エージェントごとに1つの起動ログを/var/log/<cluster\_name>-4 に書き込んで保存します。

*The cluster properties file* の `jgroups.loglevel` および `efm.loglevel`

パラメーターを変更することにより、エージェントログに書き込まれる詳細レベルを制御できます。

```
# Logging levels for JGroups and EFM.

# Valid values are: TRACE, DEBUG, INFO, WARN, ERROR

# Default value: INFO

# It is not necessary to increase these values unless debugging a
# specific issue. If nodes are not discovering each other at
# startup, increasing the jgroups level to DEBUG will show
# information about the TCP connection attempts that may help
# diagnose the connection failures.

jgroups.loglevel=INFO
efm.loglevel=INFO
```

ロギング機能は、Java ロギングライブラリとロギングレベルを使用します。ログレベルは、ログ出力の多いものから順に、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、フェールオーバーマネージャーは **WARN** レベル以上、つまり **WARN** および **ERROR** のメッセージ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-4.<x>`）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
```

## 14.1 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してください。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により、ポート 514 への TCP 接続が有効になります。

```
# 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記録を有効にします。エディターの選択を使用して、実装するロギングのタイプを指定して *modify the properties file* `/etc/edb/efm-4.<x>/efm.properties.in` します。

```
# Which logging is enabled.

file.log.enabled=true
syslog.enabled=false
```

システムに *specify syslog details* も必要です。syslog.protocol パラメーターを使用してプロトコルタイプ（UDP または TCP）を指定し、syslog.port パラメーターを使用して syslog ホストのリッスナーポートを指定します。syslog.facility 値を、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。LOCAL0 と LOCAL7 の間の値を使用してください。

```
# 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15章 Notifications

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、通知スクリプトを呼び出します。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、クラスターの各ノードのポート 25 で SMTP サーバーを実行する必要があります。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と、クラスターの現在の状態に関する詳細が含まれています。例：

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

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

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

- **INFO** は、エージェントに関する情報メッセージを示し、手動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開始または停止しました）。

List of INFO level notifications を参照

- **WARNING** は、管理者のシステム確認が必要な事象（フェイルオーバーなど）が発生したことを示します。

List of WARNING level notifications を参照

- **SEVERE** は、重大なイベントが発生したことを示し、管理者の即時の注意が必要です（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）。

List of SEVERE level notifications を参照

重大度レベルは、通知の緊急度を指定します。重大度レベル **SEVERE** の通知はすぐにユーザーの注意を引く必要がありますが、重大度レベル **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運用情報に注意を促します。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* プロパティを使用して、通知をトリガーする最小の重大度レベルを指定できます。

**注釈:** 管理用メールアドレスに通知を送信することに加えて、すべての通知はエージェントログファイル (`/var/log/efm-4.<x>/<*cluster name*>.log`) に記録されます。

こ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。

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

[次のページに続く](#)

表 1 – 前のページからの続き

| Subject                                                    | Description                                                                                                                                                  | Note |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| Starting auto resume check for cluster <i>cluster_name</i> | このノードのエージェントは、 <i>auto.resume.period</i> 秒ごとにチェックして、障害が発生した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。この間にクラスターを確認し、データベースが再度起動されない場合はエージェントを停止します。詳細については、エージェントログを参照してください。 |      |
| Executed agent resumed script                              | 実行されたエージェント再開スクリプト <i>script_name</i> 結果: <i>script_results</i>                                                                                              |      |
| WAL logs backed up during promotion                        | 新しいプライマリに従うようにこのスタンバイを再構成すると、 <i>pg_wal</i> の内容が <i>pgdata</i> ディレクトリにバックアップされました。ディスク領域を解放する都合の良いときにこのバックアップを削除します。                                         |      |

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

| Subject                                                                              | Description                                                         | Notes |
|--------------------------------------------------------------------------------------|---------------------------------------------------------------------|-------|
| Witness agent exited on <i>node_address</i> for cluster <i>cluster_name</i>          | 証人エージェントは終了しました。                                                    |       |
| Primary agent exited on <i>node_address</i> for cluster <i>cluster_name</i>          | データベースの状態は監視されていません。                                                |       |
| Cluster <i>cluster_name</i> notified that primary agent has left                     | プライマリ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っています。                      |       |
| Standby agent exited on <i>node_address</i> for cluster <i>cluster_name</i>          | データベースの状態は監視されていません。                                                |       |
| Agent exited during promotion on <i>node_address</i> for cluster <i>cluster_name</i> | データベースの状態は監視されていません。                                                |       |
| Agent exited on <i>node_address</i> for cluster <i>cluster_name</i>                  | エージェントは終了しました。これは、Idle 状態のエージェントによって生成されます。                         |       |
| Agent exited for cluster <i>cluster_name</i>                                         | エージェントは終了しました。この通知は、通常、スタートアップが完了する前にエージェントが終了した場合、スタートアップ中に生成されます。 |       |

次のページに続く

表 2 – 前のページからの続き

| Subject                                                                 | Description                                                                                                                             | Notes |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|-------|
| Virtual IP address assigned to non-primary node                         | 仮想 IP アドレスは、非プライマリノードに割り当てられているようです。競合を回避するために、フェールオーバーマネージャーは VIP を解放します。VIP がプライマリノードに割り当てられていることを確認し、割り当てられていない場合はアドレスを手動で再割り当てします。  |       |
| Virtual IP address not assigned to primary node.                        | 仮想 IP アドレスがプライマリノードに割り当てられていないようです。フェールオーバーマネージャーは、VIP の再取得を試みます。                                                                       |       |
| Standby failed in cluster <i>cluster name</i>                           | <i>cluster_name</i> のスタンバイがクラスターを離れました。                                                                                                 |       |
| Standby agent failed for cluster <i>cluster name</i>                    | <i>cluster_name</i> のスタンバイエージェントはクラスターを離れましたが、コーディネーターは、スタンバイデータベースがまだ実行されていることを検出しました。                                                 |       |
| Standby database failed for cluster <i>cluster name</i>                 | スタンバイエージェントは、データベースに障害が発生したことを通知しました。他のノードもスタンバイデータベースに到達できません。                                                                         |       |
| Standby agent cannot reach database for cluster <i>cluster name</i>     | スタンバイエージェントはデータベース障害の信号を送信しましたが、他のノードはスタンバイデータベースがまだ実行されていることを検出しました。                                                                   |       |
| Cluster <i>cluster name</i> has dropped below three nodes               | 完全なフェー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。監視またはエージェントノードをクラスターに追加します。                                                                          |       |
| Subset of cluster <i>cluster name</i> disconnected from primary         | このノードは、クラスター <i>clustername</i> の大部分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、フェールオーバーは試行されません。現在表示されているノードは次のとおりです。 <i>node_address</i> 。 |       |
| Promotion has started on cluster <i>cluster name</i> .                  | スタンバイのプロモーションがクラスター <i>cluster_name</i> で開始されました。                                                                                       |       |
| Witness failure for cluster <i>cluster_name</i>                         | <i>node_address</i> で実行されている Witness がクラスターを離れました。                                                                                      |       |
| Idle agent failure for cluster <i>cluster_name</i> .                    | <i>node_address</i> で実行されているアイドル状態のエージェントがクラスターを離れました。                                                                                  |       |
| One or more nodes isolated from network for cluster <i>cluster_name</i> | このノードはネットワークから隔離されているようです。クラスターで見られる他のメンバーは次のとおりです。 <i>node_name</i>                                                                    |       |
| Node no longer isolated from network for cluster <i>cluster_name</i> .  | このノードはネットワークから隔離されなくなりました。                                                                                                              |       |

次のページに続く

表 2 – 前のページからの続き

| Subject                                                              | Description                                                                                                                                                                     | Notes                                                        |
|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| Failover Manager tried to promote, but primary DB is still running   | フェールオーバーマネージャーは昇格手順を開始しましたが、プライマリ DB が <b>アドレス</b> でまだ実行されていることを検出しました。これは通常、プライマリ Failover Manager エージェントが終了したことを示します。フェールオーバーは発生していません。プライマリエージェントが再起動されるまで、フェールオーバー保護はありません。 |                                                              |
| Primary agent missing for cluster <i>cluster_name</i>                | プライマリエージェントは以前にクラスターを離れました。プライマリエージェントがクラスターに参加するまで、フェールオーバー保護はありません。                                                                                                           | Available in Failover Manager 4.2 and later.                 |
| Standby agent started to promote, but primary has rejoined.          | スタンバイ Failover Manager エージェントは、自分自身の昇格を開始しましたが、プライマリエージェントがクラスターに再参加したことがわかりました。フェールオーバーは発生していません。                                                                              |                                                              |
| Standby agent tried to promote, but could not verify primary DB      | スタンバイ Failover Manager エージェントは自分自身を昇格しようとしたましたが、プライマリ DB が <i>node_address</i> 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。フェールオーバーは発生していません。                                                     | Description applicable for Failover Manager 4.2 and later.   |
| Standby agent tried to promote, but could not verify primary DB      | スタンバイ Failover Manager エージェントは自分自身を昇格させようとしたましたが、プライマリ DB が <i>node_address</i> でまだ実行されていることを検出しました。これは通常、プライマリ Failover Manager エージェントが終了したことを示します。フェールオーバーは発生していません。          | Description applicable for Failover Manager 4.1 and earlier. |
| Standby agent tried to promote, but VIP appears to still be assigned | スタンバイ Failover Manager エージェントは、自分自身を昇格させようとしたましたが、仮想 IP アドレス ( <i>VIP_address</i> ) がまだ別のノードに割り当てられているように見えるため、できませんでした。このような状況でプロモーションを行うと、データが破損する可能性があります。フェールオーバーは発生していません。 |                                                              |
| Standby agent tried to promote, but appears to be orphaned           | スタンバイフェールオーバーマネージャーエージェントは、自分自身を昇格しようとしたましたが、既知のサーバー ( <i>server_address</i> ) に到達できなかったため、できませんでした。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェールオーバーは発生していません。         |                                                              |

次のページに続く

表 2 – 前のページからの続き

| Subject                                                                        | Description                                                                                                                                                                       | Notes                                                                   |
|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Potential manual failover required on cluster <i>cluster_name</i> .            | クラスター <i>cluster_name</i> で潜在的なフェイ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、手動介入が必要です。                                                                                       |                                                                         |
| Failover has completed on cluster <i>cluster_name</i>                          | クラスター <i>cluster_name</i> でフェールオーバーが完了しました。                                                                                                                                       |                                                                         |
| Lock file for cluster <i>cluster_name</i> has been removed                     | クラスター <i>cluster_name</i> のロックファイルは、ノード <i>node_address</i> 上の <i>path_name</i> から削除されました。このロックにより、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できなくなります。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ます。 |                                                                         |
| A recovery file for cluster <i>cluster_name</i> has been found on primary node | クラスター <i>cluster_name</i> のリカバリファイルは次の場所にありました。プライマリノード <i>node_address</i> の <i>path_name</i> 。このノードでDBを再起動しようとする、問題が発生する可能性があります。                                               |                                                                         |
| recovery_target_timeline is not set to latest in recovery settings             | リカバリ設定で <i>recovery_target_timeline</i> パラメーターが <i>latest</i> に設定されていません。スタンバイサーバーは、新しいプライマリが昇格したときに発生するタイムラインの変更に従うことができません。                                                     |                                                                         |
| Promotion has not occurred for cluster <i>cluster_name</i>                     | 昇格が試行されましたが、昇格中のノードが既にあります： <i>ip_address</i> 。                                                                                                                                   |                                                                         |
| Standby will not be reconfigured after failover in cluster <i>cluster_name</i> | このノードの 'auto.reconfigure' プロパティは <i>false</i> に設定されているため、プロモーション後に新しいプライマリノードをフォローするように再構成されません。                                                                                  | Subject and Description applicable to Failover Manager 4.2 and later.   |
| Standby not reconfigured after failover in cluster <i>cluster_name</i>         | このノードの 'auto.reconfigure' プロパティは <i>false</i> に設定されていました。ノードは、フェールオーバー後に新しいプライマリノードに従うように再構成されませんでした。                                                                             | Subject and Description applicable to Failover Manager 4.1 and earlier. |
| Could not resume replay for standby <i>standby_id</i> .                        | スタンバイのリプレイを再開できませんでした。手動介入が必要になる場合があります。エラー： <i>error_message</i> 。                                                                                                               |                                                                         |

次のページに続く

表 2 – 前のページからの続き

| Subject                                                                                         | Description                                                                                                                                                                                                                               | Notes |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| Possible problem with database timeout values                                                   | <code>`remote.timeout`</code> の値 (値) は、 <code>`local.timeout`</code> の値 (値) よりも高くなっています。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ますが、データベースに障害が発生したと想定する場合があります。これによりフェールオーバーは発生しませんが、ローカルエージェントの監視が強制的に停止され、フェールオーバー保護がなくなります。 |       |
| No standbys available for promotion in cluster <i>cluster_name</i>                              | クラスター内のスタンバイノードの現在の数が最小数 <i>number</i> に低下しました。別のスタンバイノードが追加されるか、昇格可能にならない限り、フェールオーバーはありません。                                                                                                                                              |       |
| No promotable standby for cluster <i>cluster_name</i>                                           | クラスター内の現在のフェールオーバー優先順位リストは空です。クラスター <i>cluster_name</i> の唯一の昇格可能なスタンバイを削除しました。別の昇格可能なスタンバイノードが追加されるか、フェールオーバー優先順位リストに追加して昇格可能にならない限り、フェールオーバーはありません。                                                                                      |       |
| Synchronous replication has been re-configured for cluster <i>cluster_name</i>                  | クラスター内の同期スタンバイノードの数が <i>number</i> を下回りました。プライマリの同期スタンバイ名が次のように再構成されました。新しい <i>synchronous_standby_names</i> 値。                                                                                                                           |       |
| Synchronous replication has been disabled for cluster <i>cluster_name</i> .                     | クラスター内の同期スタンバイノードの数が <i>count</i> を下回りました。プライマリの同期レプリケーションモードが解除されました。                                                                                                                                                                    |       |
| Could not reload database configuration. Custom monitor timeout for cluster <i>cluster_name</i> | データベース構成をリロードできませんでした。手動介入が必要です。エラー: <i>error_message</i> 。<br>次のカスタム監視スクリプトがタイムアウトしました: <i>script_name</i>                                                                                                                               |       |
| Custom monitor 'safe mode' failure for cluster <i>cluster_name</i>                              | 次のカスタム監視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: <i>script_name</i> 。<br>出力: <i>script_results</i>                                                                                                                                                |       |

次のページに続く

表 2 – 前のページからの続き

| Subject                                                                                  | Description                                                                                                                                                                                                         | Notes                                                                 |
|------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| pri-<br>mary.shutdown.as.failure<br>set to true for primary<br>node                      | このクラスターの primary.shutdown.as.failure プロパティは true に設定されています。クラスター全体を停止せずにプライマリエージェントを停止すると、クラスターの残りの部分によって即時のプライマリエージェント障害として扱われます。プライマリデータベースでメンテナンスが必要な場合は、プライマリエージェントをシャットダウンし、フェールオーバーが発生しないという残りのノードからの通知を待ちます。 |                                                                       |
| Primary_or_Standby<br>cannot ping local<br>database for cluster<br>cluster_name          | Primary_or_Standby エージェントは、node_address で実行されているローカルデータベースに到達できなくなりました。他のノードはリモートでデータベースにアクセスできるため、エージェントは IDLE になり、データベースの監視を再開しようとします。                                                                            | Subject and Description applicable to Failover Manager 4.1 and later. |
| PPrimary cannot ping<br>local database for cluster<br>cluster_name                       | プライマリエージェントは、node_address で実行されているローカルデータベースに到達できなくなりました。他のノードはリモートでデータベースにアクセスできるため、プライマリは IDLE になり、データベースの監視を再開しようとします。                                                                                           | Subject and Description applicable to Failover Manager 4.0.           |
| Standby cannot re-<br>sume monitoring local<br>database for cluster<br>cluster_name      | スタンバイエージェントは、node_address で実行されているローカルデータベースに到達できなくなりました。他のノードはデータベースにリモートでアクセスできます。スタンバイエージェントは、`resume` コマンドが実行されてデータベースの監視を再開するまで IDLE のままです。                                                                    | Available in Failover Manager 4.1 and later.                          |
| Primary agent left the<br>cluster and node de-<br>tached from load bal-<br>ancer         | スタンバイエージェントは、node_address で実行されているローカルデータベースに到達できなくなりました。他のノードはデータベースにリモートでアクセスできます。スタンバイエージェントは、`resume` コマンドが実行されてデータベースの監視を再開するまで IDLE のままです。                                                                    | Available in Failover Manager 4.2 and later.                          |
| Standby database in<br>cluster cluster_name<br>not stopped before<br>primary is promoted | `standby.restart.delay` プロパティはこのエージェントに設定されているため、プロモーションが完了してから num_seconds 秒まで新しいプライマリをフォローするように再構成されません。場合によっては、手動介入なしでは新しいプライマリをたどれない場合があります。                                                                    | Available in Failover Manager 4.2 and later.                          |

この表にリストされている条件は、SEVERE 通知をトリガーします。

| Subject                                                                         | Description                                                                                                                                                                                                                   | Notes |
|---------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| Standby database restarted but Failover Manager cannot connect                  | データベースの起動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。フェールオーバーマネージャーは、restart.connection.timeout 秒まで接続を試行し続けます。                                                                                                                 |       |
| Unable to connect to DB on <i>node_address</i>                                  | 最大接続数の制限に達しました。                                                                                                                                                                                                               |       |
| Unable to connect to DB on <i>node_address</i>                                  | db.user=user_name のパスワードが無効です。                                                                                                                                                                                                |       |
| Unable to connect to DB on <i>node_address</i>                                  | 無効な認可仕様です。                                                                                                                                                                                                                    |       |
| Primary cannot resume monitoring local database for cluster <i>cluster_name</i> | プライマリエージェントは、 <i>node_address</i> で実行されているローカルデータベースに到達できなくなりました。他のノードはリモートでデータベースにアクセスできるため、プライマリはVIPを解放したり、 <code>recovery.conf</code> ファイルを作成したりしません。 <code>resume</code> コマンドを実行してデータベースの監視を再開するまで、プライマリエージェントはIDLEのままです。 |       |
| Fencing script error                                                            | フェンシングスクリプト <i>script_name</i> は、正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i> フェールオーバーは発生していません。                                                                                                             |       |
| Post-promotion script failed                                                    | プロモーション後スクリプト <i>script_name</i> は正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i>                                                                                                                               |       |
| Remote post-promotion script failed                                             | リモートプロモーション後スクリプト <i>script_name</i> は正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i> Node: <i>node_address</i>                                                                                                 |       |
| Remote pre-promotion script failed                                              | リモートプレプロモーションスクリプト <i>script_name</i> は正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i> Node: <i>node_address</i>                                                                                                |       |
| Post-database failure script error                                              | データベース障害後スクリプト <i>script_name</i> は、正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i>                                                                                                                             |       |
| Agent resumed script error                                                      | エージェント再開スクリプト <i>script_name</i> は正常に実行できませんでした。結果: <i>script_results</i>                                                                                                                                                     |       |
| Primary isolation script failed                                                 | プライマリ分離スクリプト <i>script_name</i> は、正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i>                                                                                                                               |       |

次のページに続く

表 3 – 前のページからの続き

| Subject                                                                    | Description                                                                                                                                              | Notes |
|----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| Could not promote standby                                                  | ノードでプロモートコマンドが失敗しました。スタンバイを昇格できませんでした。エラーの詳細：error_details                                                                                               |       |
| Error creating recovery.conf file on node_address for cluster cluster_name | プロモーション中に、プライマリノード <i>node_address</i> で recovery.conf ファイルを作成中にエラーが発生しました。プロモーションは継続されていますが、古いプライマリノードを再起動できないように手動での介入が必要です。エラーの詳細：message_details     |       |
| An unexpected error has occurred for cluster cluster_name                  | このノードで予期しないエラーが発生しました。詳細については、エージェントログを確認してください。エラー：error_details                                                                                        |       |
| Primary database being fenced off for cluster cluster_name                 | プライマリデータベースは、クラスターの大部分から分離されています。クラスターは、ip_address のプライマリエージェントに、プライマリデータベースをフェンスオフして、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 つのプライマリがないように指示しています。       |       |
| Isolated primary database shutdown.                                        | 分離されたプライマリデータベースは、フェールオーバーマネージャによってシャットダウンされました。                                                                                                         |       |
| Primary database being fenced off for cluster cluster_name                 | プライマリデータベースは、クラスターの大部分から分離されました。プライマリが分離の検出を完了する前に、スタンバイが昇格し、クラスター内のこのノードに再参加しました。このノードは、複数のプライマリデータベースを回避するために自分自身を分離しています。                             |       |
| Could not assign VIP to node node_address                                  | 何らかの理由で、フェールオーバーマネージャは VIP アドレスを割り当てるできませんでした。                                                                                                           |       |
| primary_or_standby database failure for cluster cluster_name               | 指定されたノードでデータベースに障害が発生しました。                                                                                                                               |       |
| Agent is timing out for cluster cluster_name                               | このエージェントは、ローカルデータベースに到達しようとしてタイムアウトになりました。タイムアウト後、エージェントはデータベースへの ping に成功し、監視を再開しました。ただし、ノードをチェックして、ノードが正常に実行されていることを確認して、データベースまたはエージェントの障害の可能性を防止します。 |       |

次のページに続く

表 3 – 前のページからの続き

| Subject                                                        | Description                                                                                                                                                                       | Notes |
|----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| Resume timed out for cluster <i>cluster_name</i>               | このエージェントは、ローカルデータベースを再構成して再起動した後、モニタリングを再開できませんでした。詳細については、エージェントログを参照してください。                                                                                                     |       |
| Internal state mismatch for cluster <i>cluster_name</i>        | フェールオーバーマネージャークラスターの内部状態が、クラスターメンバーの実際の状態と一致していませんでした。これはまれであり、ノードがクラスターに参加するか、それらの状態を変更するタイミングの問題が原因である可能性があります。問題は解決するはずですが、クラスターのステータスも確認して確認します。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せん。 |       |
| Failover has not occurred                                      | エージェントは、プライマリデータベースがクラスター <i>cluster_name</i> で使用できなくなったことを検出しましたが、フェ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。                                                                                  |       |
| Failover has not occurred                                      | エージェントは、プライマリデータベースがクラスター <i>cluster_name</i> で使用できなくなったことを検出しましたが、フェールオーバーに使用できる十分なスタンバイノードがありません。                                                                               |       |
| Database in wrong state on <i>node_address</i>                 | スタンバイエージェントは、ローカルデータベースが復旧していないことを検出しました。エージェントは IDLE になります。手動介入が必要です。                                                                                                            |       |
| Database in wrong state on <i>node_address</i>                 | プライマリエージェントは、ローカルデータベースが復旧中であることを検出しました。エージェントは IDLE になります。手動介入が必要です。                                                                                                             |       |
| Database connection failure for cluster <i>cluster_name</i>    | このノードは、次で実行されているデータベースに接続できません: <i>node_address</i> 。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確認できないため、フェールオーバーが正しく機能しない可能性があります。                                                      |       |
| Standby custom monitor failure for cluster <i>cluster_name</i> | 次のカスタム監視スクリプトは、スタンバイノードで失敗しました。エージェントはローカルデータベースの監視を停止します。スクリプトの場所: <i>script_name</i> スクリプト出力: <i>script_results</i>                                                             |       |
| Primary custom monitor failure for cluster <i>cluster_name</i> | 次のカスタム監視スクリプトは、プライマリノードで失敗しました。フェールオーバーマネージャは、スタンバイを昇格しようとしています。スクリプトの場所: <i>script_name</i> スクリプト出力: <i>script_results</i>                                                       |       |

次のページに続く

表 3 – 前のページからの続き

| Subject                                                           | Description                                                                       | Notes                                        |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------------|
| Loopback address set for ping.server.ip                           | ping.server.ip プロパティにループバックアドレスを設定しています。この設定はネットワーク分離の検出を妨げる可能性があるため、変更する必要があります。 |                                              |
| Load balancer attach script error                                 | ロードバランサーのアタッチスクリプト 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  |                                              |
| Load balancer detach script error                                 | ロードバランサーデタッチスクリプト 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   |                                              |
| Pgpool attach node error                                          | フェールオーバーマネージャーはpgpool ノードをアタッチできませんでした。終了値： exit_code。結果： script_results          | Available in Failover Manager 4.1 and later. |
| Pgpool detach node error                                          | フェールオーバーマネージャーは、pgpool ノードのデタッチに失敗しました。終了値： exit_code。結果： script_results          |                                              |
| Not enough synchronous standbys available in cluster cluster_name | クラスター内の同期スタンバイノードの数がカウントされました。十分な同期スタンバイノードが追加されるまで、プライマリ上のすべての書き込みクエリはブロックされます。  |                                              |
|                                                                   | 該当なし                                                                              |                                              |

## 第16章 Supported failover and failure scenarios

フェールオーバーマネージャーは、フェールオーバーを引き起こす可能性の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。フェールオーバーが発生する可能性があります。

- プライマリデータベースがクラッシュまたはシャットダウンした場合。
- プライマリデータベースをホストしているノードがクラッシュまたは到達不能になった場合。

フェールオーバーマネージャーは、これらの条件の正確性を検証しようとするあらゆる試みを行います。プライマリデータベースまたはノードに障害が発生したことをエージェントが確認できない場合、フェールオーバーマネージャーはクラスターでフェ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したいが、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、*noauto-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たときに通知が管理者に送信されます。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* パラメーターを **false** に設定します。

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プライマリに昇格することに値しない状況について、管理者に警告します。

### 16.1 プライマリデータベースがダウンしています

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

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

エージェントが仮想 IP アドレスまたはデータベースサーバ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。最新のノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをプライマリデータベースに昇格させ、仮想 IP アドレスをスタンバイノードに割り当てます。*auto.reconfigure* が **false** に設定されていない限り、追加のスタンバイノードは新しいプライマリから複製するように構成されます。該当する場合、エージェントはプロモーション後スクリプトを実行します。

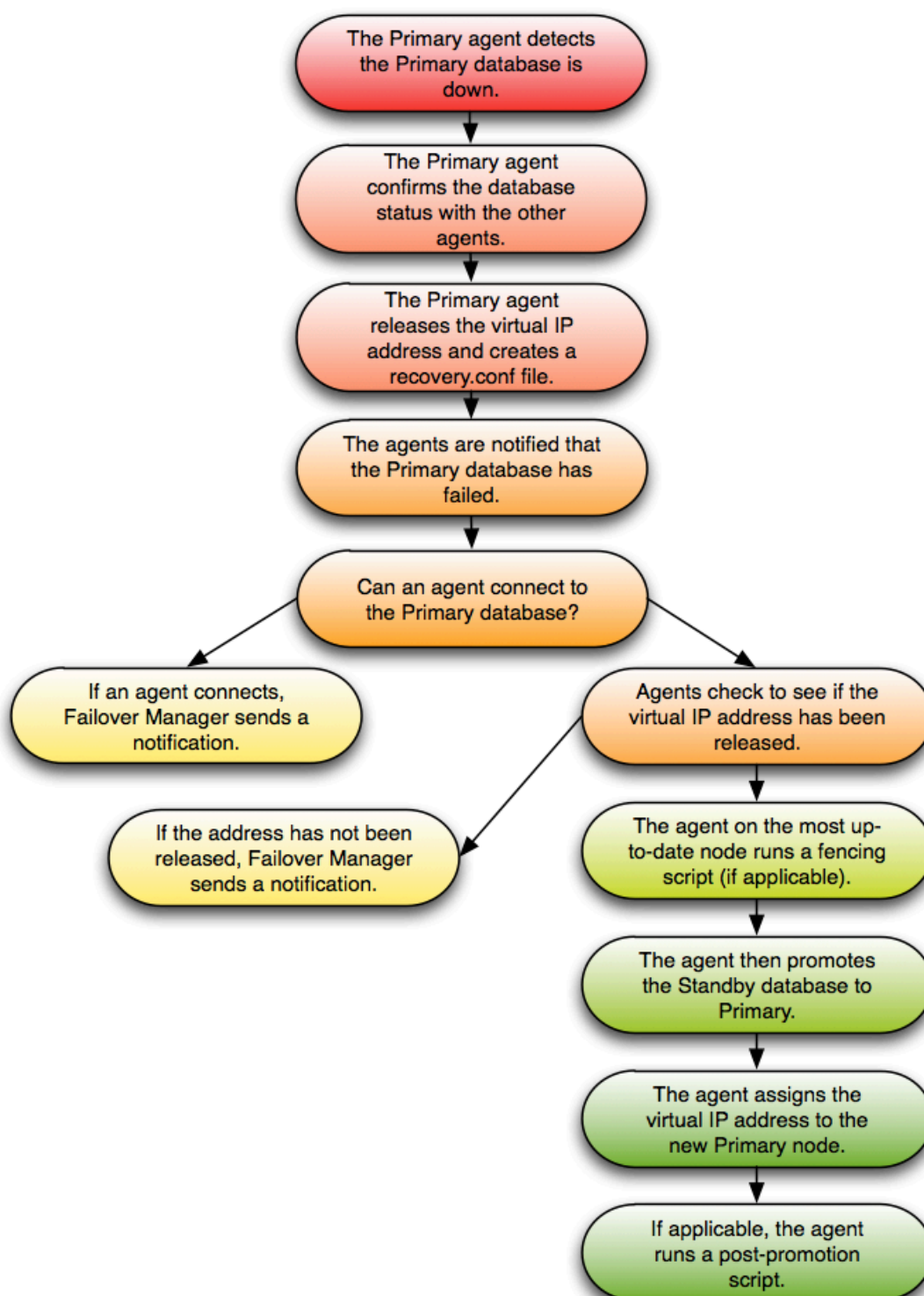


図 1: Confirming the failure of the primary database.

## 16.2 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：

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

## 16.3 ノードをプライマリの役割に戻します

ノードをスタンバイとしてクラスターに戻した後、ノードを簡単にプライマリのロールに戻すことができます。

1. クラスターに複数のスタンバイノードがある場合、`efm set-priority` コマンドを使用して、ノードのフェイルオーバー優先度を 1 に設定します。
2. `efm promote -switchover` コマンドを呼び出して、ノードをプライマリノードの元のロールに昇格させます。

---

**注釈：** フェールオーバーマネージャーは、障害が発生したプライマリデータベースを再構築してスタンバイにしません。再構築する前に、プライマリに障害が発生した理由を特定し、新しいプライマリですべてのデータを利用できることを確認することが重要です。サーバーをスタンバイとして復元する準備ができたなら、古いデータディレクトリを削除して、サーバーを復元できます。詳細については、[Setting up a standby server](#) の PostgreSQL ドキュメントを参照してください。場合によっては、`pg_rewrite` を使用してサーバーを復元することもできます。

---

## 16.4 スタンバイデータベースがダウンしています

スタンバイエージェントがデータベースの障害を検出すると、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。

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。

## 16.5 プライマリエージェントが終了するか、ノードに障害が発生しました

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

プライマリエージェントが去ったことをエージェントが検出した場合、すべてのエージェントはプライマリデータベースに直接接続しようとします。データベースに接続できるエージェントがある場合、エージェ

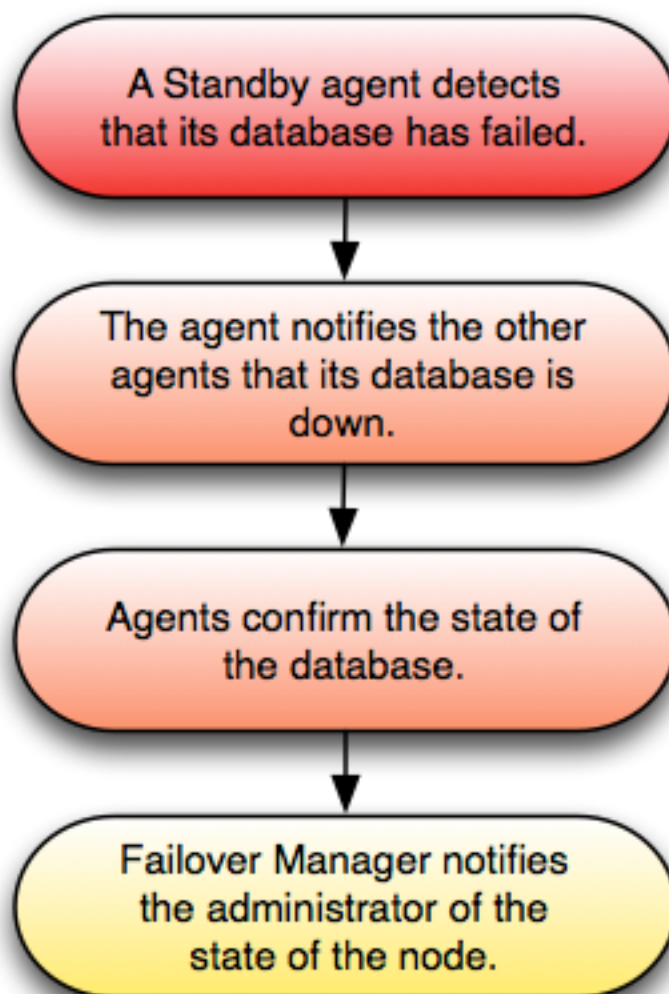


図 2: Confirming the failure of a standby database.

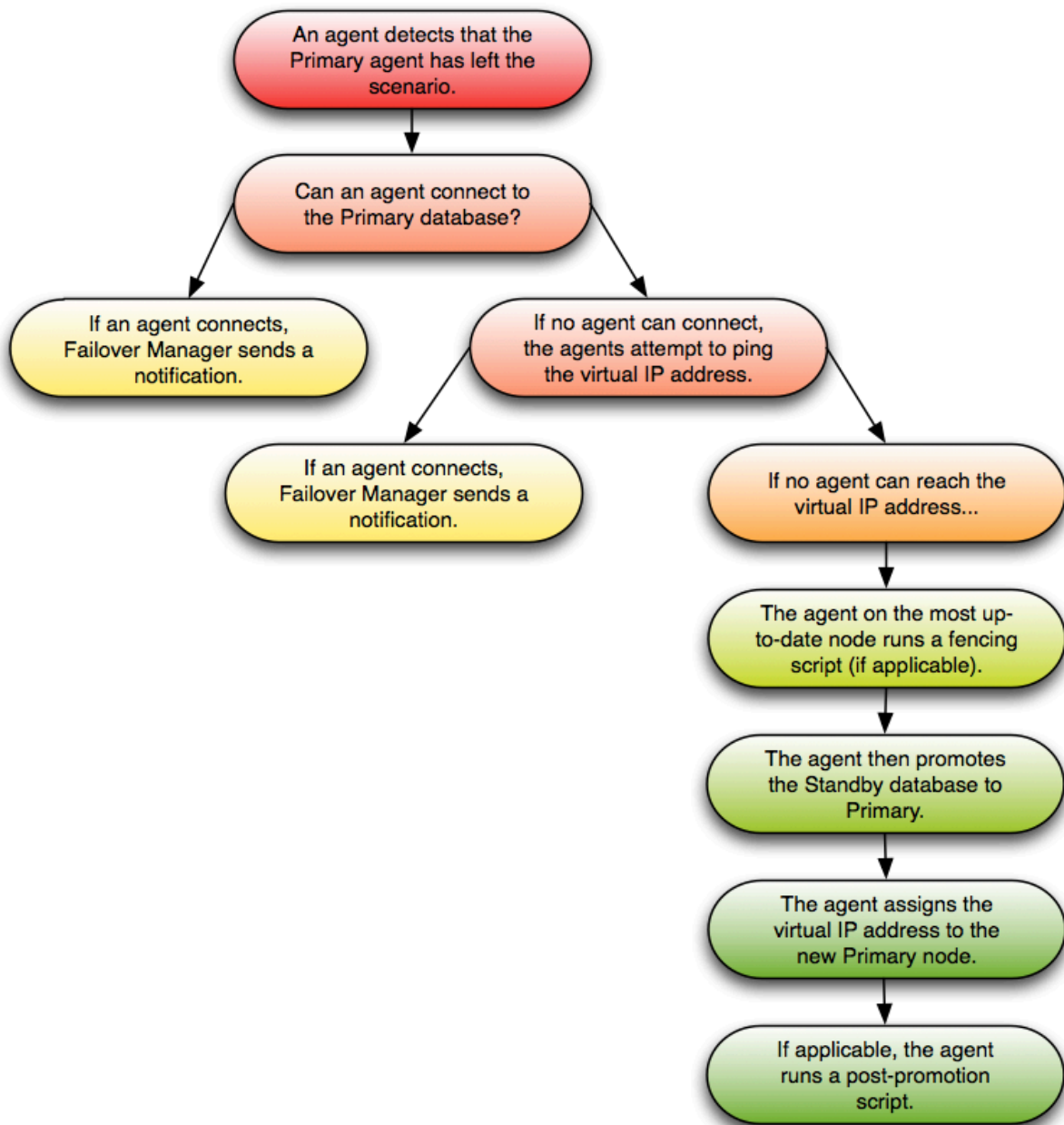


図 3: Confirming the failure of the primary agent.

ントはプライマリエージェントの障害に関する通知を送信します。接続できるエージェントがない場合、エージェントは仮想 IP アドレス（該当する場合）に 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。最新のノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをプライマリデータベースに昇格させ、仮想 IP アドレスをスタンバイノードに割り当てます。該当する場合、エージェントはプロモーション後スクリプトを実行します。`auto.reconfigure` が `false` に設定されていない限り、追加のスタンバイノードは新しいプライマリから複製するように構成されます。

プライマリがネットワークから分離されたためにこのシナリオが発生した場合、プライマリエージェントは分離を検出し、仮想 IP アドレスを解放して、`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、クラスタの残りのノードで同じ手順を実行します。

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：

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

プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動されるまでフェールオーバー保護はありません。このケースを回避するには、`systemd` を介してプライマリノードを設定し、プライマリエージェントの終了時にフェイルオーバーが発生させることができます。詳細については、

*[Configuring for Eager Failover](#)* を参照してください。

## 16.6 スタンバイエージェントが終了するか、ノードに障害が発生します

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。

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

プライマリが1つだけ、スタンバイが1つだけ残っている場合、プライマリノードに障害が発生した場合のフェールオーバー保護はありません。プライマリデータベースに障害が発生した場合、プライマリエージェントとスタンバイエージェントは、データベースに障害が発生したことに同意し、フェールオーバーを続行できます。

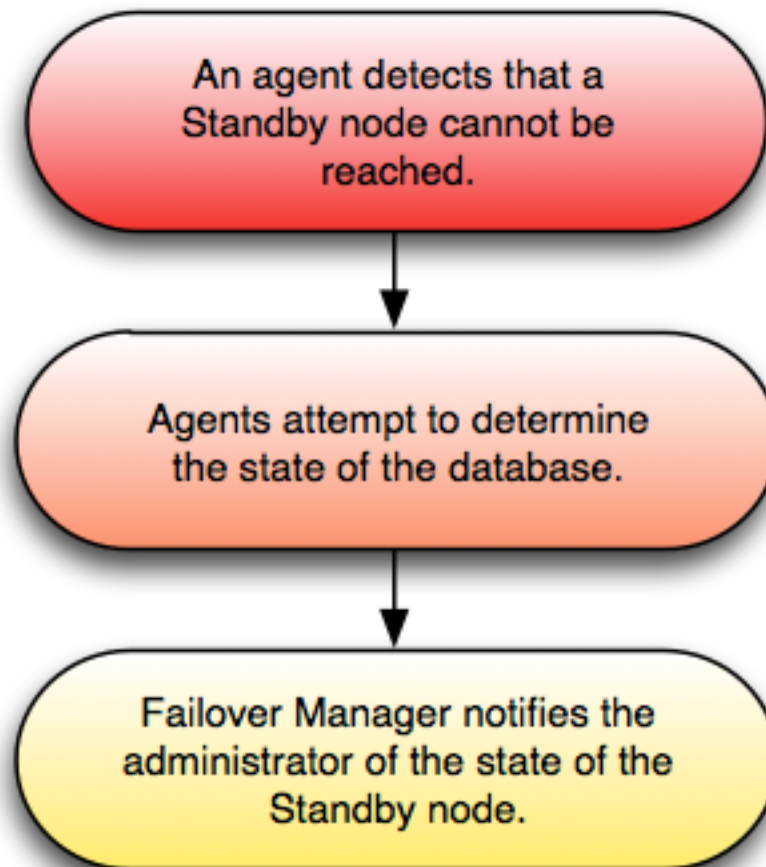


図 4: Failure of standby agent.

## 16.7 専用の証人エージェントが終了/ノードに障害が発生する

このシナリオでは、専用の監視（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。

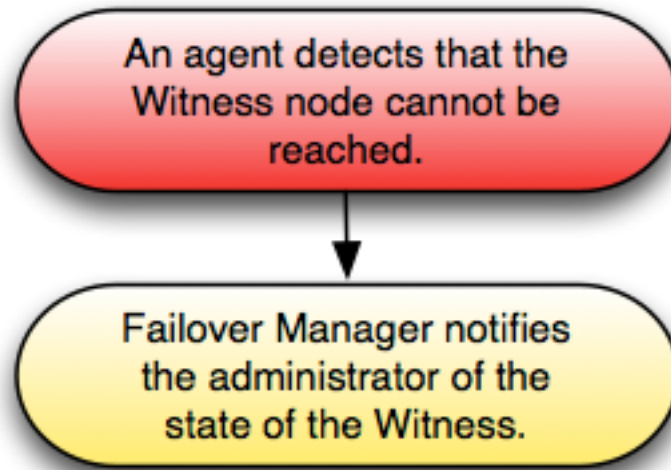


図 5: Confirming the failure of a dedicated witness.

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

**注釈:** 監視が失敗し、クラスターに2つのノードしかない場合、フェールオーバー保護はありません。スタンバイノードは、プライマリに障害が発生したか切断されたかを知ることができません。2ノードクラスターでは、プライマリデータベースに障害が発生してもノードがまだ接続されている場合、フェールオーバーは引き続き発生します。スタンバイはプライマリデータベースの状態を確認できます。

## 16.8 ノードはクラスターから分離されます

このシナリオでは、1つ以上のノード（クラスターのマイノリティ）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。

1つ以上のノード（ただし、クラスターの半分未満）がクラスターの残りの部分から分離されると、残りのクラスターは、ノードに障害が発生したかのように動作します。エージェントは、プライマリノードが隔離されたノードの中にあるかどうかを識別しようとします。そうである場合、プライマリはクラスターから自分自身をフェンスオフし、クラスターマジョリティ内のスタンバイノードがそれを置き換えるように昇格されます。`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プライマリ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。

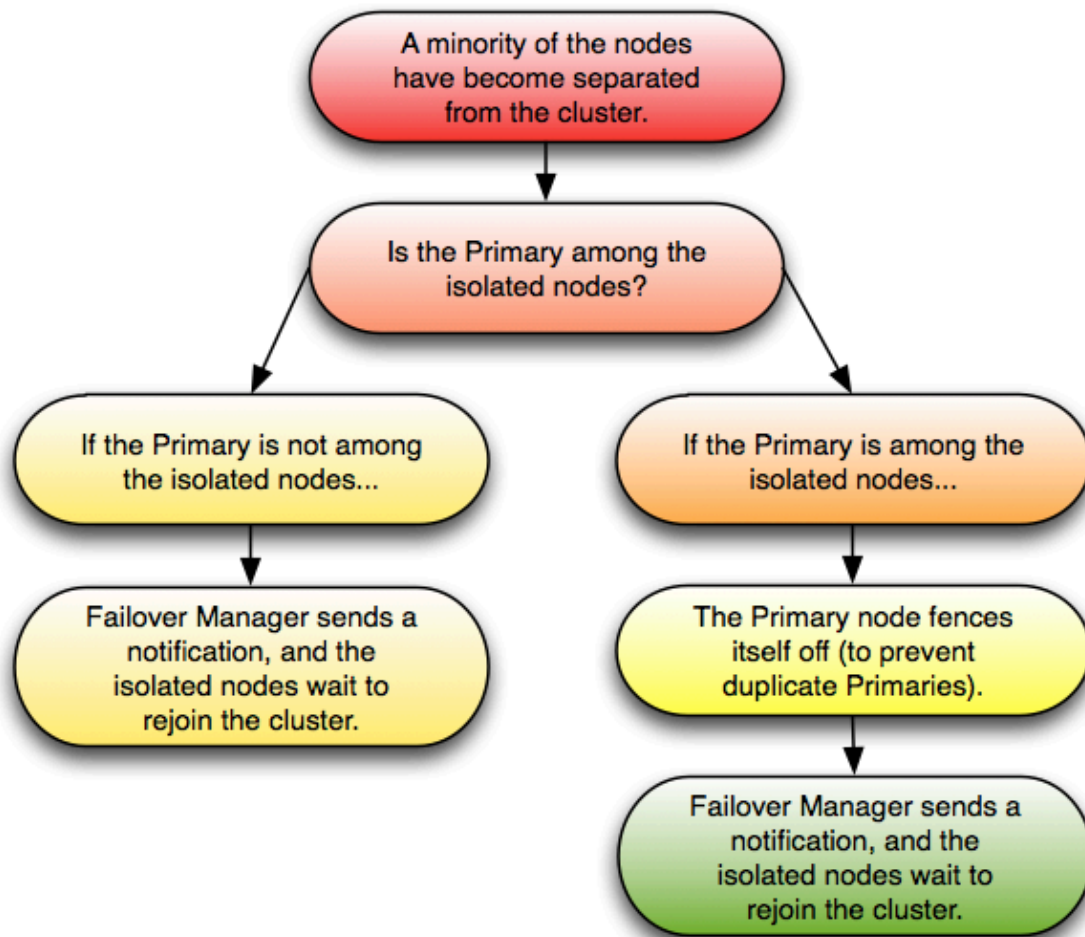


図 6: If members of the cluster become isolated.



## 第17章 Troubleshooting

### 17.1 認証ファイルが見つかりません。ローカルエージェントは実行されていますか？

フェールオーバーマネージャクラスター管理コマンドを呼び出し、フェールオーバーマネージャがノードで実行されていない場合、efm コマンドはエラーを表示します。

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

### 17.2 このコマンドを実行する権限がありません。ユーザー「<os user>」`efm` グループのメンバーではありません。

*Using the efm utility* に記載されている efm

コマンドの一部を呼び出すには、特別な特権が必要です。これらのコマンドが、実行を許可されていないユーザーによって呼び出された場合、efm コマンドはエラーを表示します。

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

### 17.3 通知。予期しないエラーメッセージ

予期しないエラーメッセージに関する通知メッセージを受け取った場合は、

*Failover Manager log file* で OutOfMemory

メッセージを確認してください。フェールオーバーマネージャは、このプロパティによって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
```

128 メガバイト未満の割り当てで実行している場合は、値を増やして Failover Manager エージェントを再起動します。

## 17.4 OpenJDK バージョンの確認

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 の使用を強くお勧めします。次のコマンドを使用して、Java インストールの種類を確認できます。

```
# 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18章 Creating a Failover Manager cluster

フェールオーバーマネージャーは、プライマリノードでソフトウェアまたはハードウェアに障害が発生した場合に、Postgres プライマリノードがスタンバイノードに自動的にフェールオーバーできる高可用性ツールです。

このチュートリアルでは、テスト環境での Failover Manager クラスターの構成について説明します。運用展開用にフェールオーバーマネージャーを構成する前に、残りのフェールオーバーマネージャードキュメントを読んで理解してください。

EDB Postgres Advanced Server を例として使用します（フェールオーバーマネージャーは PostgreSQL でも動作します）。チュートリアルを開始する前に、次の基本的なインストールと構成の手順に従います。

- 1つのプライマリノードと1つまたは2つのスタンバイノードにデータベースサーバをインストールおよび初期化します。インストールの詳細については、  
EDB Postgres Advanced Server ドキュメントを参照してください。
- Postgres ストリーミングレプリケーションを構成し、プライマリノードとスタンバイノード間で実行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構成の詳細については、  
*Configuring streaming replication* を参照してください。
- 各プライマリノードとスタンバイノードに Failover Manager をインストールします。EDB Postgres Advanced Server のインストール中に、各データベースホストで EDB リポジトリを構成しました。EDB リポジトリと `yum install` コマンドを使用して、クラスターの各ノードに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-efm47
```

インストールプロセス中に、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特権を持つ `efm` という名前のユーザーを作成します。次の例では、`efm` という名前のクラスターを作成します。

プライマリまたはスタンバイノードで構成プロセスを開始します。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。

1. 作業用構成ファイルを作成します。提供されたサンプルファイルをコピーしてフェールオーバーマネージャー構成ファイルを作成し、別のバージョンをインストールする場合は所有権とバージョン番号を修正します。

```
cd /etc/edb/efm-4.7

cp efm.properties.in efm.properties

cp efm.nodes.in efm.nodes
```

(次のページに続く)

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

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

1. プロパティファイルに必要な *encrypted password* を作成します。

```
/usr/edb/efm-4.7/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生成します。

1. `efm.properties` を更新します。 `<cluster_name>.properties` ファイル（この例では `efm.properties`）には、Failover Manager 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、フェールオーバーマネージャーの起動時に適用されます。

このチュートリアルで説明されているプロパティは、フェールオーバーマネージャークラスターを構成するために必要な最小限のプロパティです。実稼働システムを構成している場合、フェールオーバーマネージャーのオプションの詳細については、

*Configuring Failover Manager* を確認してください。

すべてのクラスターノードで次のプロパティの値を指定します。

| Property                           | Description                                                                                                            |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <code>db.user</code>               | データベースユーザーの名前。                                                                                                         |
| <code>db.password.encrypted</code> | データベースユーザーの暗号化されたパスワード。                                                                                                |
| <code>db.port</code>               | データベースによって監視されるポート。                                                                                                    |
| <code>db.database</code>           | データベースの名前。                                                                                                             |
| <code>db.service.owner</code>      | <code>data</code> ディレクトリの所有者（通常は <code>postgres</code> または <code>enterprisedb</code> ）。データベースがサービスとして実行されている場合にのみ必要です。 |
| <code>db.service.name</code>       | データベースサービスの名前（サーバーの再起動に使用）。データベースがサービスとして実行されている場合にのみ必要です。                                                             |
| <code>db.bin</code>                | <code>bin</code> ディレクトリへのパス（ <code>pg_ctl</code> の呼び出しに使用）。                                                            |
| <code>db.data.dir</code>           | EFM が <code>recovery.conf</code> ファイルまたは <code>standby.signal</code> ファイルを検索または作成する <code>data</code> ディレクトリ。          |
| <code>user.email</code>            | 電子メール通知を受信する電子メールアドレス（通知テキストはエージェントログファイルにもあります）。                                                                      |
| <code>bind.address</code>          | フェールオーバーマネージャーに使用するノードとポートのローカルアドレス。フォーマットは次のとおりです。 <code>bind.address=1.2.3.4:7800</code>                             |
| <code>is.witness</code>            | 監視ノード上の <code>true</code> 、およびプライマリまたはスタンバイの場合は <code>false</code> 。                                                   |
| <code>ping.server.ip</code>        | インターネットにアクセスできないネットワークで実行している場合は、 <code>ping.server.ip</code> をネットワークで利用可能なアドレスに設定します。                                 |
| <code>auto.allow.hosts</code>      | テストクラスターでは、 <code>true</code> に設定して起動を簡素化します。実稼働での使用については、Failover Manager ユーザーガイドを参照してください。                           |
| <code>stable.nodes.file</code>     | テストクラスターでは、 <code>true</code> に設定して起動を簡素化します。実稼働での使用については、Failover Manager ユーザーガイドを参照してください。                           |

1. `efm.nodes` を更新します。 `<cluster_name>.nodes` ファイル（この例では `efm.nodes` ）は、起動時にエージェントに読み取られ、クラスターの残りの部分を見つける方法を指示します。または、最初に起動したノードの場合、後続のノードの承認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つのノードはメンバーシップコーディネーターとして機能します。少なくとも会員コーディネーターの住所をリストに入れてください。例：

```
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` ファイル内のアドレスを検証しません。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想定しています（たとえば、別のエージェントがまだ開始されていないことなど）。

1. 他のノードを構成します。 `efm.properties` および `efm.nodes` ファイルを、サンプルクラスター内の他のノードの `/etc/edb/efm-4.7` 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、次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。

- `bind.address` プロパティを変更して、ノードのローカルアドレスを使用します。
- ノードが witness ノードの場合、 `is.witness` を `true` に設定します。ノードが監視ノードの場合、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。

1. フェールオーバーマネージャークラスターを起動します。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 `edb-efm-4.7` 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、RHEL 7.x または Rocky Linux/AlmaLinux/RHEL 8.x ホストでは、次のコマンドを使用します。

```
systemctl start edb-efm-4.7
```

1. エージェント起動後、次のコマンドを実行してシングルノードクラスターの状態を確認します。他のノードのアドレスが `Allowed node host` リストに表示されます。

```
/usr/edb/efm-4.7/bin/efm cluster-status efm
```

1. 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、クラスターのステータスを確認します。

いずれかの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ます。

```
cat /var/log/efm-4.7/startup-efm.log
```

## 18.1 スイッチオーバーを実行する

クラスターステータスの出力に、プライマリノードとスタンバイノードが同期していることが示されている場合、スイッチオーバーを実行できます。

```
/usr/edb/efm-4.7/bin/efm promote efm -switchover
```

このコマンドは、スタンバイをプロモートし、プライマリ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。

## 18.2 オンラインヘルプにアクセスする

オンラインヘルプにすばやくアクセスするには、次を使用します。

```
/usr/edb/efm-4.7/bin/efm --help
```