



Failover Manager

リリース **5.2.0**

EnterpriseDB Corporation

2026 年 02 月 09 日

目次

第 1 章 Introduction	3
1.1 Release notes	3
1.1.1 Version 5.2	3
1.1.2 Version 5.1	4
1.1.3 Version 5.0	5
1.1.4 Version 4.10	7
1.1.5 Version 4.9	8
1.1.6 Version 4.8	8
1.1.7 Failover Manager 4.7 release notes	10
1.1.8 Failover Manager 4.6 release notes	10
1.1.9 Failover Manager 4.5 release notes	11
1.1.10 Failover Manager 4.4 release notes	12
1.1.11 Failover Manager 4.3 release notes	12
1.1.12 Failover Manager 4.2 release notes	13
1.1.13 Failover Manager 4.1 release notes	14
1.1.14 Failover Manager 4.0 release notes	15
1.2 Supported platforms	16
第 2 章 Planning	17
2.1 Architecture	17
2.2 Choosing a deployment architecture	18
2.2.1 Failover Manager with virtual IP	19
2.2.2 Failover Manager with client connect failover	20
2.2.3 Failover Manager with EDB PgBouncer	22
2.2.4 Failover Manager with EDB Pgpool-II	28
第 3 章 Installing and Upgrading	35
3.1 Installing Failover Manager on Linux	35
3.1.1 Linux x86-64 (amd64)	35
3.1.2 Linux Installing Failover Manager on Linux IBM Power (ppc64le)	35
3.1.3 Linux Installing Failover Manager on Linux AArch64 (ARM64)	36
3.2 Upgrading Failover Manager	57
3.2.1 オプションの-source フラグ	58
第 4 章 Configuring	61
4.1 Configuring streaming replication	61
4.1.1 カスケードレプリケーションの限定的なサポート	61
4.2 Configuring Failover Manager	62
4.2.1 The cluster properties file	62
4.2.2 Encrypting your database password	100

4.2.3	The cluster members file	102
4.2.4	Extending Failover Manager permissions	103
4.2.5	Using Failover Manager with virtual IP addresses	106
4.2.6	Configuring for Eager Failover	110
4.3	Configuring SSL authentication on a Failover Manager cluster	113
第 5 章	Using	115
5.1	Using Failover Manager	115
5.1.1	フェールオーバーマネージャークラスターの管理	115
5.1.2	単一のノードで複数のエージェントを実行する	120
5.2	Monitoring a Failover Manager cluster	123
5.2.1	クラスターステータスレポートの確認	123
5.2.2	クラスター内の障害が発生したノードの表示	125
5.2.3	Postgres Enterprise Manager を使用したストリーミングレプリケーションのモニタリング	126
5.3	Using the efm utility	128
5.3.1	efm allow-node	128
5.3.2	efm disallow-node	128
5.3.3	efm cluster-status	128
5.3.4	efm cluster-status-json	128
5.3.5	efm create-standby	130
5.3.6	efm encrypt	131
5.3.7	efm promote	131
5.3.8	efm remote-members	132
5.3.9	efm resume	132
5.3.10	efm set-priority	132
5.3.11	efm stop-cluster	133
5.3.12	efm upgrade-conf	133
5.3.13	efm node-status-json	133
5.3.14	efm -help	134
5.4	Controlling the Failover Manager service	134
5.4.1	RHEL/Rocky Linux/AlmaLinux 8.x 以降での systemctl ユーティリティの使用	134
5.5	Controlling logging	135
5.5.1	syslog ログファイルエントリの有効化	137
5.6	Notifications	138
5.7	Supported failover and failure scenarios	147
5.7.1	プライマリデータベースがダウンしています	147
5.7.2	ノードをクラスターに戻す	149
5.7.3	ノードをプライマリのロールに戻す	149
5.7.4	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	149
5.7.5	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	151
5.7.6	専用の監視エージェントが終了/ノードに障害が発生	152
5.7.7	ノードがクラスターから分離される	152
5.8	Troubleshooting	153
5.8.1	Failover Manager エージェントの起動に失敗する	153
5.8.2	承認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?	153

5.8.3	このコマンドを実行する権限がありません。ユーザー ' <code><os user></code> ' は ' <code>efm</code> ' グループのメンバーではありません。	153
5.8.4	通知; 予期しないエラーメッセージ	154
5.8.5	OpenJDK バージョンの確認	154
5.8.6	クラスター外部からの予期しない接続試行	155
第 6 章	Tutorial	157
6.1	Creating a Failover Manager cluster	157
6.1.1	スイッチオーバーを実行する	160
6.1.2	オンラインヘルプにアクセスします	160

フェールオーバーマネージャー 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た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有効にします。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をスタンバイデータベースノードに自動フェールオーバーを提供します。PostgreSQL または EDB Postgres Advanced Server でフェールオーバーマネージャーを使用できます。

第1章 Introduction

1.1 Release notes

Failover Manager のドキュメントでは、Failover Manager の最新バージョンについて説明しています。これらのリリースノートでは、各リリースの新機能をカバーしています。各リリースの標準のサポート終了日は、次のリリースから 18 か月後です。

Version	Release Date
<i>efm reset-members</i>	02 Dec 2025
<i>efm reset-members</i>	30 Sep 2025
<i>efm reset-members</i>	28 Mar 2025
<i>efm reset-members</i>	21 Aug 2024
<i>efm reset-members</i>	14 May 2024
<i>efm reset-members</i>	15 Nov 2023
<i>efm reset-members</i>	20 Jun 2023
<i>efm reset-members</i>	14 Feb 2023
<i>efm reset-members</i>	30 Aug 2022
<i>efm reset-members</i>	05 Jan 2022
<i>efm reset-members</i>	18 Dec 2021
<i>efm reset-members</i>	19 Apr 2021
<i>efm reset-members</i>	11 Dec 2020
<i>efm reset-members</i>	02 Sep 2020

1.1.1 Version 5.2

EFM 5.2 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	フェールオーバーマネージャーは、pg_rewind または pg_basebackup を使用して、障害が発生したデータベースサーバーをスタンバイとして自動的に再構築をサポートするようになりました。	
Enhancement	パフォーマンスを向上させるために、エージェントは、分離されたと判断した場合、切断されたスタンバイデータベースの状態のチェックをスキップします。	

次のページに続く

表 2 – 前のページからの続き

Type	Description	Addresses
Enhancement	タイミングおよびスタックトレース情報を追加して、定期的な監視およびエージェントの最終的なヘルスチェック中にデータベース接続障害が発生した場合に出力の一貫性を高めます。	
Bug Fix	データベースがまだ初期化されていない場合に <code>`efm create-standby`</code> コマンドを使用できなかった 5.1 の回帰を修正しました。	
Bug Fix	特定の場合にローカルデータベースが初期化されていない場合、エージェントが起動できないバグを修正しました。	
Bug Fix	混乱を避けるために、無効なプロパティ名が使用されたときに出力されるエラーメッセージを変更しました。	
Bug Fix	VIP が使用されていない場合の VIP の解放に関する不要な出力を削除しました。	

1.1.2 Version 5.1

EFM 5.1 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	フェールオーバーマネージャーは、必要に応じて <code>pg_rewind</code> を使用して、障害が発生したプライマリデータベースサーバーをスタンバイとして再構成をサポートするようになりました。	
Enhancement	<code>`efm create-standby`</code> コマンドの実行にスーパーユーザー権限は必要ありません。	
Enhancement	<code>`efm create-standby`</code> コマンドは、元のデータベースが存在する場合の <code>`synchronous_standby_names`</code> 設定を保存するようになりました。	
Enhancement	<code>efm create-standby`</code> コマンドに <code>-waldir`</code> オプションを追加して、ログ先行書き込みディレクトリのデフォルト以外の場所を指定します。	
Enhancement	プロモーション中に VIP を使用する場合を待機する時間を制御する新しい <code>`check.vip.timeout`</code> プロパティを追加しました。	
Enhancement	プロモーション中に VIP が解放されるタイミングを制御するための新しいプロパティ <code>`release.vip.*`</code> を追加しました。これにより、データベース接続を妨げる特定のネットワークの問題を回避できます。スイッチオーバー中。	52388, 48276

次のページに続く

表 3 – 前のページからの続き

Type	Description	Addresses
Enhancement	データベース接続を作成しようとしたときに記録される詳細ログの量を制御するための新しい <code>`jdbc.loglevel`</code> プロパティを追加しました。	
Enhancement	内部クラスターの状態を転送する際に予期しないエラーが発生した場合に送信される通知を更新しました。	46361
Enhancement	フェールオーバーマネージャーは、起動時に不明なプロパティをチェックして、混乱を招く可能性のあるユーザーエラーをキャッチするようになりました。	
Enhancement	Bouncey Castle 暗号化ライブラリバージョン 2.1.1 を使用するようにフェールオーバーマネージャーがアップグレードされました。	
Bug Fix	プライマリがまだ実行中のスタンバイデータベースであるときに <code>`efm create-standby`</code> で作成された場合、データベースのフェイルオーバーを妨げる可能性があるバグを修正しました。	
Bug Fix	<code>`bind.address`</code> プロパティが適切な形式でなかった場合に、有用なエラーメッセージを提供しない回帰を修正しました。	
Bug Fix	多すぎるパラメーターを使用して <code>`efm cluster-status`</code> コマンドが実行された場合の、紛らわしいメッセージを削除しました。	

1.1.3 Version 5.0

EFM 5.0 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	起動ログは、フェールオーバーマネージャーエージェントによって使用されなくなりました。エージェントの起動情報は、エージェントが起動された方法に応じて systemd または標準出力に送られるようになりました。	
Enhancement	<code>`pg_basebackup`</code> を使用して既存のプライマリからスタンバイデータベースを作成するための新しい CLI コマンド <code>`efm create-standby`</code> が追加されました。	

次のページに続く

表 4 – 前のページからの続き

Type	Description	Addresses
Enhancement	フェールオーバーマネージャーは、クォーラムの目的でクラスターサイズの処理を簡素化するようになりました。障害が発生したノード/エージェントは、削除されるまでクラスターの一部です。	
Enhancement	再構成中にスタンバイ WAL をバックアップするかどうかを制御する新しい `backup.wal` プロパティを追加しました。	1277862, 91506, 101026
Enhancement	`auto.resume.period` プロパティは、2 つの新しいプロパティに分割されました。1 つはエージェントの起動用、もう 1 つはデータベース障害用です。	
Enhancement	プライマリデータベースが同期スタンバイについてチェックされる頻度を制御する新しい `check.num.sync.period` プロパティを追加しました。	1133047, 91506
Enhancement	sudoers 構成ファイルのデータベースオペレーティングシステムユーザーの不要なチェックを削除しました。	42594, 99957
Enhancement	古いオペレーティングシステム用に追加された不要なネットワークステータスチェックを削除しました。	
Enhancement	`efm cluster-status` 出力のテーブル情報は、長いホスト名を処理できるようにサイズが変更されます。	101878
Enhancement	代わりに「lsn」を使用するように `efm cluster-status-json` 出力の「xlog」参照を更新しました。	
Enhancement	提案された値と一致するようにデフォルトの `detach.on.agent.failure` 動作を変更しました。	
Enhancement	障害中のクラスターサイズチェックを追加して、段階的なノードの分離が発生するネットワークの停止中のスプリットブレインの安全性を向上させます。	
Enhancement	エージェントが接続できない後、紛らわしいログを生成したローカルデータベースチェックを削除しました。	
Enhancement	エージェントがデータベースに接続できない場合、詳細情報を追加しました。	
Enhancement	JGroups バージョン 5.4.4 を使用するようにフェールオーバーマネージャーがアップグレードされました。	

次のページに続く

表 4 – 前のページからの続き

Type	Description	Addresses
Enhancement	PostgreSQL JDBC ドライバーバージョン 42.7.3 を使用するようフェールオーバーマネージャーがアップグレードされました。	
Enhancement	Log4J バージョン 2.24.3 を使用するようフェールオーバーマネージャーがアップグレードされました。	
Enhancement	Jakarta Mail バージョン 2.0.1 を使用するよう Failover Manager がアップグレードされました。	
Enhancement	Json-Simple バージョン 1.1.1 を使用するよう Failover Manager がアップグレードされました。	
Bug Fix	エージェントがシャットダウンを完了する前に `systemctl stop` 呼び出しが終了し、エージェントプロセスが正常に終了しない原因となる可能性のあるエージェントシャットダウンコードを修正しました。	37270
Bug Fix	同じホストからの以前のエージェントがタイムアウトする前に、障害が発生した/リブートされたスタンバイノードのエージェントがクラスターに参加しようとした場合を処理します。	43162
Bug Fix	フェールオーバーマネージャーバージョン 4.0 以前のパスワードの `efm upgrade-conf` サポートを削除しました。まれに、このサポートがアップグレードの失敗を発生させていました。	
Bug Fix	失敗したユーザーフェンシングスクリプトが一時的な内部状態の不一致を発生させる可能性がある問題を修正しました。	
Bug Fix	クラスターステータス情報を収集するエージェントに返される場合、ノードのステータスに関する欠落している説明出力を追加しました。	
Bug Fix	`jdbc.properties` プロパティをプロパティテンプレートに追加して、構成のアップグレード中に失われるのを回避しました。	

1.1.4 Version 4.10

EFM 4.10 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	Bouncy Castle 暗号化ライブラリバージョン 1.0.2.5 を使用するようフェールオーバーマネージャーがアップグレードされました。	

[次のページに続く](#)

表 5 – 前のページからの続き

Type	Description	Addresses
Bug Fix	スイッチオーバー中にプロモートするスタンバイが使用できなくなる稀なケースの処理が向上しました。	37266
Bug Fix	<code>-source`と宛先ディレクトリが同じである場合、`efm upgrade-conf`コマンドは、.nodes`ファイル情報を失わないようになりました。</code>	37479
Bug Fix	すべてのデータベース接続が失敗した場合、 <code>`efm cluster-status`</code> コマンドが接続エラーを非表示にする問題を修正しました。	39108
Bug Fix	起動時に、プライマリデータベースを使用するエージェントが、クラスター内に既にプライマリが存在することを確認すると、データベースをフェンシングオフするときにVIPをドロップするようになりました該当する場合。	

1.1.5 Version 4.9

EFM 4.9 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	PostgreSQL JDBC ドライバーバージョン 42.7.2 を使用するようにフェールオーバーマネージャーがアップグレードされました。	
Bug Fix	スイッチオーバー中にプライマリデータベースの停止中にスレッドが中断されるというまれなケースを処理しました。	102362
Bug Fix	起動プロセスの終了を妨げる可能性のある不適切なプロパティ設定の問題を修正しました。	

1.1.6 Version 4.8

EFM 4.8 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	フェールオーバーマネージャーを実行するために必要な Java の最小バージョンは、8 ではなくバージョン 11 になりました。	

次のページに続く

表 7 – 前のページからの続き

Type	Description	Addresses
Enhancement	ノードがクラスターから削除された後に、キャッシュされたノードアドレスを削除するための新しい CLI コマンド <code>efm remote-members</code> が追加されました。	1047118
Enhancement	データベースパスワードの暗号化/復号化が FIPS 環境で機能するようになりました。	90803, 90805
Enhancement	クラスターステータスコール中にスタンバイから WAL 再生/受信クエリを取得する順序を変更しました。これは、データベースクラスターに負荷がかかっている場合の混乱を招く出力を回避するのに役立ちます。	
Enhancement	フェールオーバーマネージャーは、実行中のエージェントのアドレス/ポートに接続しようとしている外部プロセスに関する詳細情報をログに記録するようになりました。	
Enhancement	PostgreSQL JDBC ドライバーバージョン 42.6.0 を使用するようにフェールオーバーマネージャーがアップグレードされました。	
Enhancement	ローカルデータベースが初期化される前に、フェールオーバーマネージャーエージェントを起動できます。	
Enhancement	<code>ping.server.ip</code> プロパティが仮想 IP アドレス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ユーザーに警告し、起動されません。これらを同じ値に設定すると、スイッチオーバー中に VIP がドロップされたときにネットワーク接続チェックが中断されます。	
Bug Fix	クラスターが偶数に分割された場合たとえば、同じ数のノードを持つ 2 つのデータセンター間で、2 つのプライマリノードが発生する可能性のあるエッジケースを修正しました。	
Bug Fix	他のノードがクラスター通信の問題に関する警告をログに記録するエッジケースを防ぐため、クラスターのシャットダウン中に、コーディネーターは数秒待機してから終了します。	96840
Bug Fix	同時のクラスターステータスチェックのため、プライマリデータベースの障害が無視される場合がある、まれなタイミングケースを修正しました。	98428

1.1.7 Failover Manager 4.7 release notes

リリース日 2023 年 6 月 20 日

EFM 4.7 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	新しいプライマリに従うように再構成する場合、フェールオーバーマネージャーは pg_wal のバックアップを 1 つだけスタンバイに保持するようになりました。	1277862, 91506
Bug Fix	フェールオーバーマネージャー起動スクリプトが、小数点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。例、「1,8」。	89616
Bug Fix	フェールオーバーマネージャーが誤ってスタンバイの WAL 受信値を、フェールオーバーマネージャー 4.6 の 'efm' コマンドの再生値に使用した問題を修正しました。	

1.1.8 Failover Manager 4.6 release notes

リリース日 2023 年 2 月 14 日

EFM 4.6 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Security Fix	フェールオーバーマネージャーは、PostgreSQL JDBC ドライバーバージョン 42.4.3 を使用するようにアップグレードされ、セキュリティの脆弱性 CVE-2018-10936 に対処しました。注フェールオーバーマネージャーは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ません。	
Enhancement	フェールオーバーマネージャーは、カスタム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。	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ありません」通知の送信を続行しません。	87106

[次のページに続く](#)

表 9 – 前のページからの続き

Type	Description	Addresses
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が実行されないようにノードが過負荷になった場合、クラスターの内部状態が一貫せず、将来のフェールオーバーを妨げる可能性があります。この場合は、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ます。	87975
Bug Fix	フェールオーバーマネージャープライマリエージェントは、稀にデータベース障害とノード分離が発生した場合に VIP の解放に失敗しなくなりました。	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 つを超える同期スタンバイをサポートするようになりました。	

1.1.9 Failover Manager 4.5 release notes

リリース日 2022 年 8 月 30 日

EFM 4.5 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Security Fix	フェールオーバーマネージャーは、PostgreSQL JDBC ドライバーバージョン 42.4.1 を使用するようにアップグレードされ、セキュリティの脆弱性 CVE-2022-31197 に対処しました。注フェールオーバーマネージャーは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ません。	
Enhancement	プライマリノードがクラスターから分離され、再参加したときに、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。	937960, 968619, 81770, 81894
Bug Fix	フェールオーバーマネージャーは、まれに LSN 値が誤って比較される場合に、プロモートするスタンバイを適切に選択するようになりました。	83186

次のページに続く

表 10 – 前のページからの続き

Type	Description	Addresses
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に対して `efm` コマンドを実行するときに、プロパティファイルの余分な空白を処理するようになりました。	
Bug Fix	プライマリエージェントの実行が遅い場合でも、フェールオーバーマネージャーは昇格に失敗しなくなりました。	79847
Bug Fix	フェールオーバーマネージャーは、`efm` コマンドを介して手動でプロモーション可能になった後に再起動されるスタンバイの <code>promotable=false</code> 値を適切にに従います。	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に、スタンバイの LSN 値を進めるときに、スタンバイでスロットをドロップして再作成します。	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」ステータスを表示するようになりました。	

1.1.10 Failover Manager 4.4 release notes

リリース日 2022 年 1 月 5 日

EFM 4.4 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Security Fix	log4j バージョン 2.17.1 を使用するようにフェールオーバーマネージャーがアップグレードされ、セキュリティの脆弱性 CVE-2021-44832 に対処しました。	CVE-2021-44832

1.1.11 Failover Manager 4.3 release notes

リリース日 2021 年 12 月 18 日

EFM 4.3 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Security Fix	フェールオーバーマネージャーは、log4j バージョン 2.17.0 を使用するようにアップグレードされ、セキュリティ脆弱性 CVE-2021-44228、CVE-2021-45046、および CVE-2021-45105 に対処します。CVE-2021-44228 は、CVSS スコア 10.0 の重大な脆弱性と考えられているため、この更新はできるだけ早く適用する必要があります。	CVE-2021-44228, CVE-2021-45046, CVE-2021-45105

1.1.12 Failover Manager 4.2 release notes

リリース日 2021 年 4 月 19 日

EFM 4.2 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	プライマリリソースを監視されていない異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に、新しいイーガーフェールオーバー設定が追加されました。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。イーガーフェールオーバーがないと、プライマリ Postgres インスタンスは利用可能なままですが、管理されません。	
Enhancement	EDB Postgres の高可用性とホリゾンタル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。	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脱しても、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 の機能強化は、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を繰り返すことです。	
Bug Fix	アーカイブで構成されたスタンバイが、新しいプライマリに従うように適切に再構成されます。	1127627, 1182654

次のページに続く

表 13 – 前のページからの続き

Type	Description	Addresses
Bug Fix	新しいオプションは、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、LoadBalancer 切断スクリプトを無効にできます。	1096007
Bug Fix	エージェントの起動に検証を追加して、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します。親ディレクトリは、EFM がそのバックアップを作成し、後で復元できるように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。

1.1.13 Failover Manager 4.1 release notes

リリース日 2021 年 12 月 11 日

EFM 4.1 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
Enhancement	EFM を使用した HA アーキテクチャ、および Azure ネットワークロードバランサーを使用した Pg プール展開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接続が回復すると、ヘルスマonitoringを再開しようとします。	
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、`num_sync` を自動的に増加させます。	
Bug Fix	同期スタンディが追加されるときに `num_sync` を増加できます。	1092053
Bug Fix	存在しない特定のプロパティではエージェントを起動できません。	1087362
Bug Fix	セキュリティの脆弱性のため、JDBC ドライバーがアップグレードされました。	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。	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。

1.1.14 Failover Manager 4.0 release notes

リリース日 2021 年 9 月 2 日

EFM 4.0 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description	Addresses
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ーインストールをアップグレードする場合、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に合わせて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の暗号化も有効になりました。	
Enhancement	新しいプライマリの選択中に、スタンバイサーバーが停止されなくなりました。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。	
Enhancement	コミュニティの命名ガイドラインと一致するように、フェールオーバーマネージャー製品とドキュメントでは、用語「マスター」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。 <code>script.load.balancer.attach</code> 、 <code>script.load.balancer.detach</code> などのロードバランサースクリプトは、引数として`m`の代わりに文字`p`を受け入れるようになりました。	
Enhancement	昇格後のスタンバイの再起動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遅らせることにより、可用性を向上できます。	
Bug Fix	<code>/var/run</code> および <code>/var/lock</code> 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	977437
Bug Fix	IDLE エージェントは、他のエージェントがデータベースに到達できるときに監視を再開しようとしています。	1001533

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- 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、プライマリの%t プレースホルダーは、m ではなく p に置き換えられます。

1.2 Supported platforms

フェールオーバーマネージャーがサポートするプラットフォームとバージョンの詳細については、[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は、プライマリノードとスタンバイノードが IBM Power の Linux であり、監視ノードが x86-64 Linux である場合にサポートされます。

第2章 Planning

2.1 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、障害を迅速に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プロモートすることができます。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の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ー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するプライマリデータベースサーバーです。
- 1つ以上のスタンバイノードは、プライマリノードに関連付けられたストリーミングレプリケーションサーバーです。
- 監視ノードは、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アサーションを確認します。障害が発生しているときに、プライマリがノードの半分以上を含むパーティションにある場合、プライマリはプライマリのままです。そのため、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備えたクラスターでの実行をサポートしています。

非 Witness 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、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。
- データベースが復旧していることが判明した場合、エージェントはスタンバイのロールを引き継ぎます。
- 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ロールを引き継ぎます。

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確認しようとします。スタンバイノードがプライマリノードと同期していない場合、データ損失の可能性があります。

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相互に通信し、ノード障害を検出できるクラスターを作成でき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ロードバランサーを再構成する独自の *efm_address スクリプト* を提供する場合、*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せん。高可用性を実現するためにネイティブ 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。

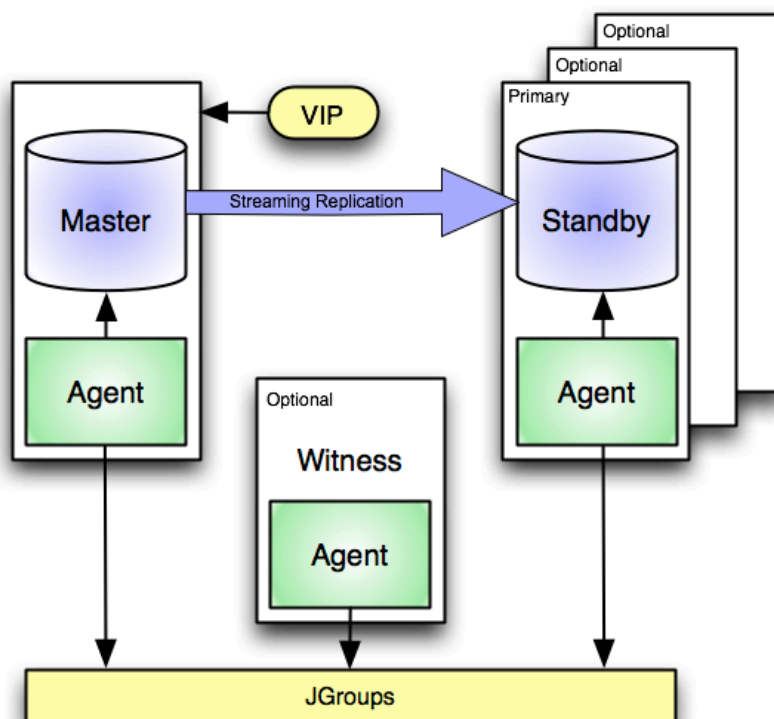


図 1: A Failover Manager scenario employing a virtual IP address

2.2 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 の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を与えます。

データベースの高可用性を保証には、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。

フェールオーバーマネージャーの機能を使用して、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. *Using Failover Manager with virtual IP addresses* 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 アドレスが利用可能な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 with client connect failover* libpq や jdbc のような PostgreSQL クライアントライブラリでは、クライアント接続フェールオーバーが可能です。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=svr1,svr2` が含まれ、クライアントライブラリは使用可能なホストをループして、使用可能な読み取り/書き込み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追跡できます。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. *Failover Manager with EDB PgBouncer* 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てクラスターモードで実行、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

2.2.1 Failover Manager with virtual IP

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

VIP 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 3 つのサーバーに Advanced Server データベースとフェールオーバーマネージャーをインストールおよび構成します。

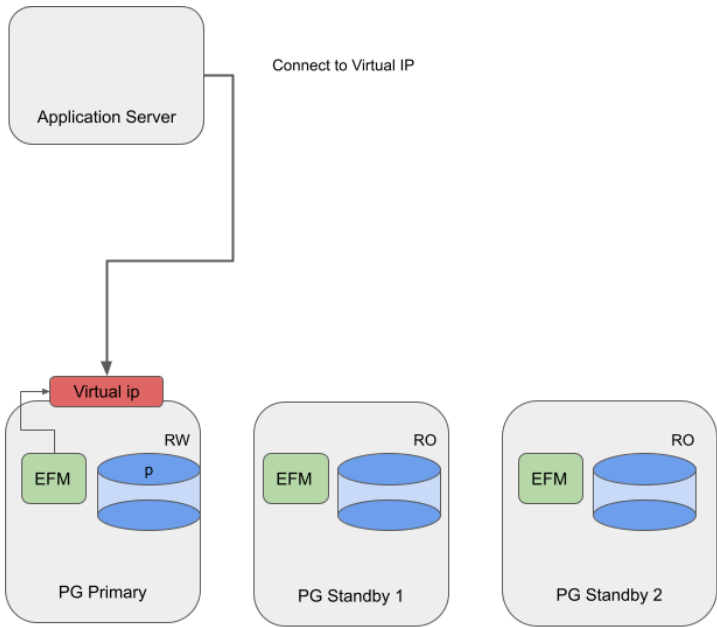


図 2: Failover Manager with VIP

Systems	Components
PG Primary, Standby1, and Standby2	PG Primary / standby nodes running Advanced Server and Failover Manager.

VIP の指定

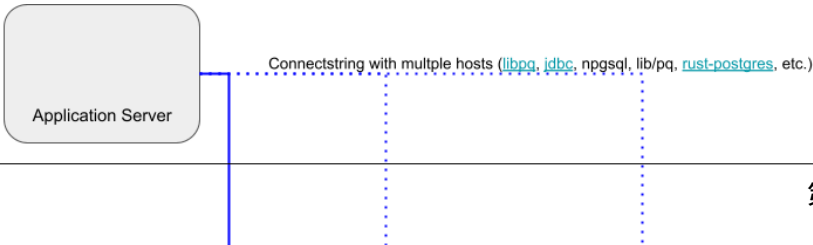
クラスタープロパティファイルの `virtual.ip` プロパティにホスト名または IP アドレスを指定します。 `virtual.ip.prefix` プロパティで、対応する接頭辞を指定します。 `virtual.ip.interface` プロパティを使用して、VIP が使用するネットワークインターフェイスを提供します。デフォルトでは、`virtual.ip` および `virtual.ip.prefix` 値はすべてのエージェントで同じです。

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

2.2.2 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ています。最初の接続試行または接続が失われると、フェールオーバー中に発生します。クライアントドライバは、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。新しいマスターへの再接続にかかる時間は、ドライバおよび 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。



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

次のとおりに 3 つのサーバーに Advanced Server と Failover Manager をインストールおよび構成します。

Systems	Components.
PG Primary, PG Standby1, and PG Standby2	Primary or standby nodes running Advanced Server and Failover Manager.

`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	Version supported	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: https://jdbc.postgresql.org/documentation/head/connect.html#connection-failover
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: https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING
.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;Target Session Attributes=primary</code> . More information: https://www.npgsql.org/doc/failover-and-load-balancing.html
go-jackc/pgx	Yes	5 and above	Use <code>libpq</code> syntax as shown above. More information: https://pkg.go.dev/github.com/jackc/pgx#ConnConfig
psycopyg3	Yes	All supported versions	Use <code>libpq</code> syntax as shown above. More information: https://www.psycopg.org/psycopg3/docs/api/connections.html
OCL	Yes	10 and above	OCL is based on <code>libpq</code> , hence check <code>libpq</code> for details.
ODBC	No		

2.2.3 Failover Manager with EDB PgBouncer

フェールオーバーマネージャーと 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セットアップで高可用性を提供できます。EDB PgBouncer は人気の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、自分自身で PostgreSQL の高可用性を達成するには十分ではありません。

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することにより高可用性を提供します。

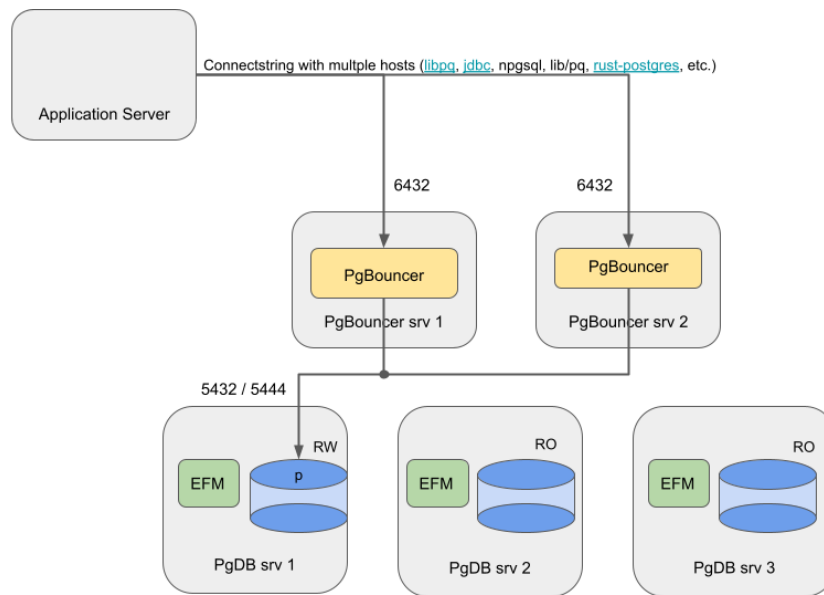


図 4: Failover Manager using pgBouncer on-premises architecture diagram

クラウドでの EDB PgBouncer を使用したフェールオーバーマネージャー

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のバランスをとります。

EDB は、EDB PgBouncer と Failover Manager / PostgreSQL を同じマシンで実行している場合、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の制限 [Azure](#)

ソースと宛先が同じマシンにある場合、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、場合によってはネットワーク。
- EDB PgBouncer と PostgreSQL がリソースをめぐる競争します。

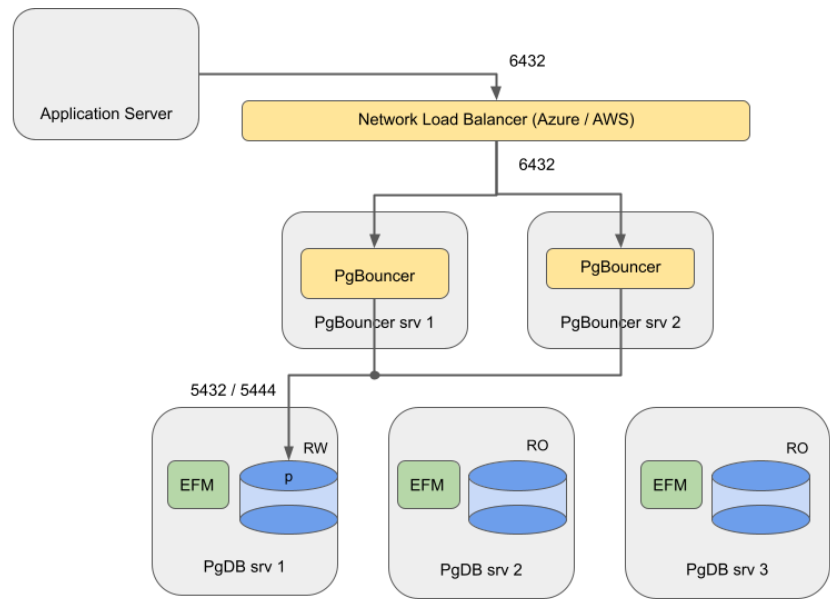


図 5: Failover Manager with PgBouncer cloud architecture diagram

- これらの2つのコンポーネントが同じマシンで組み合わせられている場合、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を与えます。

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、フェールオーバーマネージャー、および EDB PgBouncer をインストールおよび構成します。

Systems	Components
PgDB srv 1, 2, 3	Primary / standby node running Advanced Server and Failover Manager
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ます

をクリックして、フェールオーバーマネージャーを構成します。これらの指示に加えて、次の手順を実行します。

- 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-5.<x>/bin/efm_pg_bouncer_functions でスクリプトを見つけます。ユーザー efm が次の内容を含むスクリプトを実行できることを確認します。

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

(次のページに続く)

(前のページからの続き)

```
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、efm プロパティファイルのカスタムスクリプトに `script.load.balancer.attach` を設定します。

```
script.load.balancer.attach=/usr/edb/efm-5.<x>/bin/efm_pgbouncer_functions attach %h
↪ %t <pgbs1>,<pgbs2>
```

<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では、トラフィックは両方の PgBouncer インスタンス、および PostgreSQL への両方の開いた接続にわたってバランスがとれます。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` パラメーターが補正されていることを確認します。

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 を構成するため。これらの指示に加えて、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、このセクションを別のファイル `/etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini` で構成します。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。

次に、ファイルを作成する bash コマンドの例を示します。

```
echo "[databases]" > /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
echo "edb= host=srv1" >> /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ルートであり、そのユーザー/グループ/その他 0755 に読み取りおよび実行アクセスがあることを確認します。スクリプトには次の内容が含まれます。

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

パスワードなし ssh の構成

EDB PgBouncer 統合の場合、パスワードなしの ssh アクセスが必要です。ssh を構成するには複数の方法があります。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 接続できる必要があります。例、enterprisedb。

EDB PgBouncer ホストでの構成

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、一時的なパスワードを 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。

```
grep ^PasswordAuthentication /etc/ssh/sshd_config
```

yes に設定されていることを確認します。必要に応じて、変更して ssh をリスタートします。

フェールオーバーマネージャー/PostgreSQL ホストでの構成

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、次のコマンドを使用して ssh キーをコピーします。

```
ssh-copy-id enterprisedb@<pgbouncerhost>
```

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 ホストでの一時パスワードのリセット

root として次のコマンドを実行して、すべての EDB PgBouncer ホストで enterprisedb ユーザーの一時パスワードをリセットできます。

```
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 および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

次のページに続く

表 7 – 前のページからの続き

Type	Protocol	Port range	Source	Description
Custom TCP	TCP	5444	Entire Subnet	Postgres
Custom TCP	TCP	22	Entire Subnet	ssh

これらのルールは、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 を実行するために必要なポートが、トラフィックルーティングとヘルスマonitoringのためのノードとロードバランサー間の通信用に開いていることを保証します。

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 での NLB の構成

AWS を使用している場合、[AWS での NLB の構成](#) 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する 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。
- 正常性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できる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 のロードバランシングルールを追加します。このルールにより、そのポートに着信する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すると、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プロモートし、EDB PgBouncer を再構成してトラフィックを再分散します。トラフィックを受け入れることができない EDB PgBouncer プロセスがある場合、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。フェールオーバーが発生した場合に備えて、接続数の増加を補償するように max_client_conn パラメーターが調整されていることを確認します。

AWS での NLB の構成

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は同じサブネットに展開されます。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、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 ドキュメントに従ってください

- 次の詳細を使用して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 **Advanced health check** 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。

- 次の詳細を使用してロードバランサーを作成します。

Type	VPC	Listener
<i>Public or 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 <i>6432</i> , and forward it to the target group <i>pg-bouncer</i> .

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プロモートし、EDB PgBouncer を再構成してトラフィックを再分散します。トラフィックを受け入れることができない EDB PgBouncer プロセスがある場合、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。フェイルオーバーが発生した場合に備えて、接続数の増加を補償するように `max_client_conn` パラメーターが調整されていることを確認します。

2.2.4 Failover Manager with EDB Pgpool-II

Pgpool-II は、読み取り専用のトラフィックロードバランシング、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人気の接続プーラーです。可用性を管理するフェールオーバーマネージャーと、正しいプライマリにプロキシへの EDB Pgpool-II を再構成すると、セットアップはオンプレミスの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。

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利用可能な EDB Pgpool-II インスタンスにルーティングできます。この設定では、EDB Pgpool-II の自動フェイルオーバーが無効になり、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ます。

クラウドでの 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のある環境クラウド環境などの場合、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 インスタンスでトラフィックを分散できます。

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

次のとおりに、Advanced Server 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 をインストールおよび構成します。

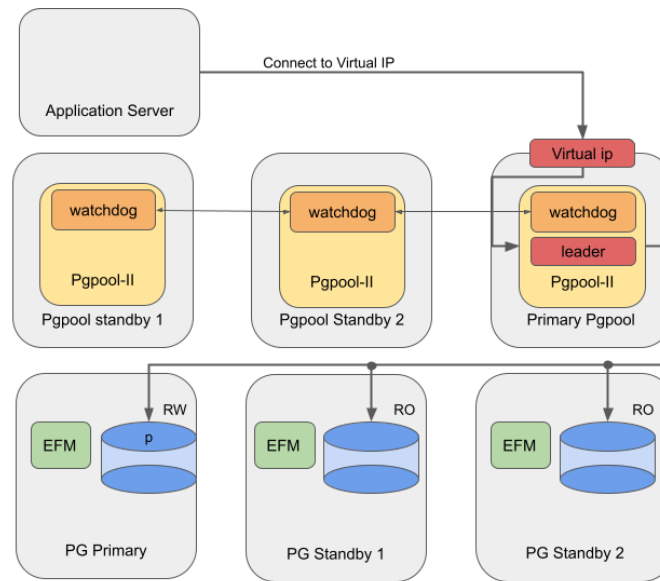


図 6: Failover Manager with Pgpool on-premises

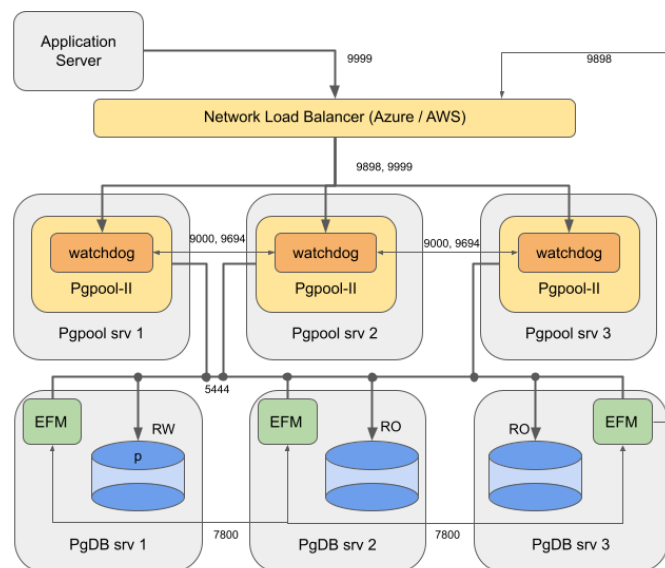


図 7: Failover Manager with Pgpool in cloud

Systems	Components
PgDB server 1, server2, and server 3	Primary / standby node running Advanced Server and Failover Manager
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、次の理由から、EDB Pgpool-II と同じマシンで実行されている Failover Manager / PostgreSQL での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の制限は、ソースと宛先が同じマシンにある場合、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、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 はリソースをめぐって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 ロードバランシングから削除できます。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 に再接続することもできます。EDB Pgpool-II を使用して高可用性を備えた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 と Failover Manager を統合できます。

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。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等しく分散されます。

```
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
```

- EDB Pgpool-II バージョン 4.2 より前のバージョンの場合

```
master_slave_mode = on

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フェールオーバーマネージャーによって処理されるため、EDB Pgpool-II 側でそれらが無効にします。EDB Pgpool-II 側でヘルスチェックとフェールオーバーが無効にするには、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ときに以下を確認します。

- pgpool.conf の wd_priority の値をノードごとに異なるままにします。最も高い値を持つノードが最も高い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ールオーバーマネージャー用語であり、すべての EDB Pgpool-II ノードの pgpool.conf ファイルに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 ファイルに、ノードの数に応じて、プロパティ heartbeat_destination0、heartbeat_destination1、heartbeat_destination2 などを追加します。ここでは、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 のセットアップ

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

PCP を設定するには、[Configuring pcp.conf](#) を参照してください。

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

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保証します。

フェールオーバーマネージャーがヘルスチェックを実行し、フェールオーバーをトリガーするため、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っています。この場合、フェールオーバーマネージャーとの競合、または時期尚早に実行されるフェールオーバーを回避するために、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 は低い値に設定されて、フェールオーバーマネージャーがトリガーしたフェールオーバーが発生したときに EDB Pgpool-II サービスの迅速なリカバリーを保証します。

仮想 IP アドレスの使用

EDB Pgpool-II と Failover Manager は、両方ともシームレスなフェイルオーバーのために仮想 IP を使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。このデザインと同様に、このような仮想 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 はこのデザイン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 サーバーに障害が発生すると、次の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従って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フェールオーバーマネージャー/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 が使用するセキュリティグループのルール。

タイプ	プロトコル	ポートレンジ	ソース	説明
カスタム	TCP	TCP 9000	サブネット全体	ウォッチドッグカスタム TCP TCP 9694
サブネット全体	ハートビート	カスタム TCP TCP 9898	サブネット全体	pcp カスタム TCP TCP 9999
サブネット全体				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

タイプ	プロトコル	ポートレンジ	ソース	説明
カスタム	TCP	TCP 7800	サブネット全体	フェールオーバーマネージャーカスタム TCP TCP 5444
サブネット全体				Postgres

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 を実行するために必要なポートが、トラフィックルーティングとヘルスマonitoringのためのノードとロードバランサー間の通信のためにオープンされるようになります。

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 での NLB の構成

AWS を使用する場合、[AWS での NLB の構成](#) を参照してください。

[ネットワークロードバランサーの構成](#) 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。
- 正常性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。

- 2つのロードバランシングルールを追加します。ポート 9898 およびポート 9999 に1つずつです。これらのルールにより、そのポートに着信する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すると、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プロモートし、トラフィックを再分散するように EDB Pgpool-II を再構成します。トラフィックを受け入れることができない EDB Pgpool-II プロセスがある場合、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。フェールオーバーが発生した場合に備えて、接続数の増加を補償するように `listen_backlog_multiplier` パラメーターが調整されていることを確認します。

AWS での NLB の構成

サンプル構成では、次の仮定が採用されています。

- すべての EC2 インスタンスとロードバランサーは同じサブネットに展開されます。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、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。

ネットワークロードバランサーの構成 で説明しているセキュリティグループルールを構成した後、AWS ドキュメントに従って、次のことを行います。

- 次の詳細を使用して2つのターゲットグループを作成します。

名前 | タイプ | プロトコル | ポート | VPC ----- |----- -|----- -|----- -|-----
 ----- pcpl | インスタンス | TCP | 9898 | インスタンスが接続されている VPC を選択します。 pgpool
 | インスタンス | TCP | 9999 | インスタンスが接続されている VPC を選択します。残りの設定ヘルスチェック TCP および **Advanced health check** 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。

- 次の詳細を使用してロードバランサーを作成します。

タイプ | VPC | リスナー | ----- -|----- -|----- -|----- -|-----
 ----- -|----- -|----- -|----- -|-----
 ----- | Public または Internal 。EDB
 は、内部ロードバランサーを使用することをお勧めします。| VPC を選択し、目的のゾーンにマッピングします。| TCP を 9898 に設定してリスナーを作成し、ターゲットグループ pcpl に転送します。TCP を 9999 に設定して別のリスナーを作成し、ターゲットグループ pgpool に転送します。

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プロモートし、EDB Pgpool-II を再構成してトラフィックを再分散します。トラフィックを受け入れることができない EDB Pgpool-II プロセスがある場合、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。フェールオーバーが発生した場合に備えて、接続数の増加を補うように `listen_backlog_multiplier` が調整されていることを確認します。

第3章 Installing and Upgrading

3.1 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。

3.1.1 Linux x86-64 (amd64)

Red Hat Enterprise Linux RHEL とその派生

- *Installing Failover Manager on RHEL 9 or OL 9 arm64*、*Installing Failover Manager on RHEL 8 ppc64le*
- *Installing Failover Manager on RHEL 9 or OL 9 arm64*、*Installing Failover Manager on RHEL 8 ppc64le*
- *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*、*Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86_64*

SUSE Linux Enterprise SLES

- *Installing Failover Manager on SLES 15 ppc64le*

Debian とデリバティブ

- *Installing Failover Manager on Ubuntu 22.04 x86_64*
- *Installing Failover Manager on Debian 12 arm64*、*Installing Failover Manager on Debian 11 x86_64*

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

Red Hat Enterprise Linux RHEL とその派生

- *Installing Failover Manager on RHEL 9 or OL 9 arm64*、*Installing Failover Manager on RHEL 8 ppc64le*

SUSE Linux Enterprise SLES

- *Installing Failover Manager on SLES 15 ppc64le*

3.1.3 Linux Installing Failover Manager on Linux AArch64 (ARM64)

Red Hat Enterprise Linux RHEL とその派生

- [Installing Failover Manager on RHEL 9 or OL 9 arm64](#)
- [Installing Failover Manager on RHEL 9 or OL 9 arm64](#)

Debian とデリバティブ

- [Installing Failover Manager on Debian 12 arm64](#)

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、次の前提条件を満たす必要があります。

Java 11 以降のインストール

フェールオーバーマネージャークラスターを使用する前に、最初に Java バージョン 11 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。

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が使用している Java のバージョンを更新または変更する前に、エージェントを停止し、更新後に再度起動する必要があります。これは OS のアップグレードにも適用され、エージェントが使用している Java インストールが変更される可能性があります。[Configuring for Eager Failover](#) を使用しない限り、エージェントを停止しても実行中のデータベースサーバーに影響はありません。

注釈

RHEL およびその派生の OpenJDK バージョン 11 に一時的な問題があります。フェールオーバーマネージャークラスターを起動すると、次のようなエラーが表示される場合があります。

```
java.lang.Error: java.io.FileNotFoundException: /usr/lib/jvm/java-11-openjdk-11.0.20.0.8-2.el8.x86_64/lib/tzdb.dat (No such file or directory)
```

このメッセージが表示された場合、回避策は、`sudo dnf install tzdata-java` コマンドを使用して、不足しているパッケージを手動でインストールすることです。

SMTP サーバーを提供します

ユーザー定義の通知スクリプト、電子メール、またはその両方で指定されたとおりに、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、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 SMTP サーバーが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することができます。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の構成

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。

`primary_conninfo` および `restore_command` プロパティは、`-sourcenode` オプションで特に指定しない限り、スイッチオーバー中にランダムな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。この例は、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役割に昇格させる場合があります。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。この場合、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、データベースサーバーの前に Failover Manager エージェントが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存在す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

Linux ファイアウォールつまり iptables がフェールオーバーマネージャーノードのホストで有効になっている場合、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の **bind.address** 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。**admin.port** で指定されたポートを開く必要はありません。efm ユーティリティが実行されるときに、ローカル通信にのみ使用されます。

データベースユーザーが十分な権限を持ってい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、Failover Manager に代わって次の機能呼び出すための十分な権限が必要です。

pg_current_wal_lsn()

pg_last_wal_replay_lsn()

pg_wal_replay_resume()

pg_wal_replay_pause()

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

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

update.physical.slots.period プロパティを使用する場合、**db.user** には **REPLICATION** 特権が必要です。データベーススーパーユーザーは、必要な権限を提供できます。

```
ALTER USER <user_name> REPLICATION;
```

ユーザーは、構成変数の値を読み取る権限も必要です。データベーススーパーユーザーは、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)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Failover Manager on RHEL 9 or OL 9 arm64*
- *Installing Failover Manager on RHEL 8 ppc64le*
- *Installing Failover Manager on RHEL 9 or OL 9 arm64*
- *Installing Failover Manager on RHEL 8 ppc64le*
- *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*
- *Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86_64*

SUSE Linux Enterprise SLES

- *Installing Failover Manager on SLES 15 ppc64le*

Debian とデリバティブ

- *Installing Failover Manager on Ubuntu 22.04 x86_64*
- *Installing Failover Manager on Debian 12 arm64*
- *Installing Failover Manager on Debian 11 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

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。
 - 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<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は **edb-efm52** です。

インストールプロセスは、**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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。

2. 各ノードの *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

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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 crb
```

パッケージをインストールする

```
sudo dnf -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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

- 同じホストに `Postgres` を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 リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。
 - 必要な SUSE モジュールをアクティブにします。

```
sudo SUSEConnect -p PackageHub/15.7/x86_64
```

- メタデータを更新します。

```
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 24.04 x86_64

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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 リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。

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 4.10 をインストールする場合、パッケージ名は **edb-efm410** です。

インストールプロセスは、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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 12 x86_64

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 11 x86_64

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの [The cluster properties file](#) を変更します。
2. 各ノードの [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)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL

- *Installing Failover Manager on RHEL 9 or OL 9 arm64*
- *Installing Failover Manager on RHEL 8 ppc64le*

SUSE Linux Enterprise SLES

- *Installing Failover Manager on SLES 15 ppc64le*

Installing Failover Manager on RHEL 9 ppc64le

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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
```

- キャッシュを更新します。

```
sudo dnf makecache
```

パッケージをインストールする

```
sudo dnf -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 ppc64le

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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
```

- キャッシュを更新します。

```
sudo dnf makecache
```

パッケージをインストールする

```
sudo dnf -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 ppc64le

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 必要な SUSE モジュールをアクティブにします。

```
sudo SUSEConnect -p PackageHub/15.7/ppc64le
```

- メタデータを更新します。

```
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 AArch64 (ARM64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Failover Manager on RHEL 9 or OL 9 arm64*
- *Installing Failover Manager on RHEL 9 or OL 9 arm64*

Debian とデリバティブ

- *Installing Failover Manager on Debian 12 arm64*

Installing Failover Manager on RHEL 9 or OL 9 arm64

前提条件

インストールプロセスを開始する前に

- 同じホストに Postgres を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 リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
 3. 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。
 4. EDB リポジトリのセットアップの指示に従います。
- 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<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は `edb-efm52` です。

インストールプロセスは、`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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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 12 arm64

前提条件

インストールプロセスを開始する前に

- 同じホストに `Postgres` を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 リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、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、次のように入力します。

```
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ます。

EDB リポジトリを設定するには

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。
3. 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。
4. EDB リポジトリのセットアップの指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-efm<5x>
```

<5x> 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 5.2 をインストールする場合、パッケージ名は **edb-efm52** です。

インストールプロセスは、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。

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、[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. 各ノードの *The cluster properties file* を変更します。
2. 各ノードの *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。

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。5.x はマイナーリリースを示します。

Component	Location
Executables	/usr/edb/efm-5.x/bin
Libraries	/usr/edb/efm-5.x/lib
Cluster configuration files	/etc/edb/efm-5.x
Logs	/var/log/efm-5.x
Lock files	/var/lock/efm-5.x
Log rotation file	/etc/logrotate.d/efm-5.x
sudo configuration file	/etc/sudoers.d/efm-5.x
Binary to access VIP without sudo	/usr/edb/efm-5.x/bin/secure

3.2 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、次のことを行う必要があります。

1. クラスターの各ノードに Failover Manager 5.2 をインストールします。Failover Manager のインストールの詳細については、[Installing Failover Manager on Linux](#) を参照してください。
2. Failover Manager をインストールした後、`efm upgrade-conf` ユーティリティを呼び出して、Failover Manager 5.2 の `.properties` および `.nodes` ファイルを作成します。Failover Manager インストーラーは、アップグレードユーティリティ `efm upgrade-conf` を `/usr/edb/efm-5.2/bin` ディレクトリにインストールします。ユーティリティを起動するには、root 権限を引き受けて、コマンドを呼び出します。

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

`efm upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、フェールオーバーマネージャーが使用する新しい構成ファイルにパラメーター値をコピーします。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-5.2` ディレクトリに保存します。

3. Failover Manager 5.2 の `.properties` および `.nodes` ファイルを変更し、新しい設定を指定します。そのノードのサービスを開始する前に、自分の使いやすいエディターを使って、プロパティファイル `/etc/edb/efm-5.2` ディレクトリにある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。
4. イーガフェールオーバーを使用している場合、フェールオーバーマネージャークラスターを停止する前にそれを無効にする必要があります。詳細については、[Eager フェールオーバーの無効化](#) 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 5.1 クラスターを停止できます。

```
/usr/edb/efm-5.1/bin/efm stop-cluster efm
```

注釈

プライマリエージェントは、停止時に仮想 IP アドレスを使用している場合をドロップしません。EFM のアップグレード中に、VIP でデータベースは稼働したままでアクセスできます。[Using Failover Manager with virtual IP addresses](#) も参照してください。

6. クラスターの各ノードで新しい [Controlling the Failover Manager service](#) `edb-efm-5.2` 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-5.2/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-5.1
Processing efm.properties file
```

(次のページに続く)

(前のページからの続き)

```
The following properties were added in addition to those in previous installed.
→version:
    auto.basebackup

Checking directory /etc/edb/efm-5.1
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 ~]#
```

3.2.1 オプションの `-source` フラグ

`-source` フラグを使用して、処理するファイルを含むディレクトリを明示的に指定できます。ディレクトリがフェールオーバーマネージャーの構成の場所つまり `/etc/edb/efm-<earlier_version>` の場合、ユーティリティはデフォルトの構成ディレクトリに新しいファイルを書き込みます。この動作により、必要に応じて特定の以前のバージョンからアップグレードできます。

ソースディレクトリが他のディレクトリの場合、ユーティリティはコマンドが呼び出されたディレクトリに新しいファイルを作成します。ファイルの所有者は、コマンドを実行したユーザーです。このアプローチは、通常、*using a Failover Manager configuration without sudo* の場合に使用され、root 権限は必要ありません。

概要

- `--source` フラグは使用されません。ユーティリティは、以前のインストールディレクトリで構成ファイルを検索します。新しいファイルは、現在のデフォルト構成ディレクトリに生成され、efm ユーザーが所有します。root 権限が必要です。
- `--source` フラグは、以前のインストールの構成ディレクトリに設定されています。ユーティリティは、指定されたディレクトリでのみ構成ファイルを検索します。新しいファイルは、現在のデフォルト構成ディレクトリに生成され、efm ユーザーが所有します。root 権限が必要です。
- `--source` フラグは他のディレクトリに設定されています。ユーティリティは、指定されたディレクトリでのみ構成ファイルを検索します。新しいファイルは、コマンドが呼び出されたディレクトリに生成され、コマンドを呼び出したユーザーが所有します。root 権限は必要ありません。

注釈

すべての場合、`<cluster_name>.properties` または `<cluster_name>.nodes` ファイルがターゲットディレクトリに既に存在する場合、新しいファイルが保存される前にタイムスタンプで名前が変更されます。

フェールオーバーマネージャー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、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ーの `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ー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、古いバージョンのフェールオーバーマネージャースクリプトをアンインストールする前に、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ースクリプトを使用するようにカスタムスクリプトを更新します。

Failover Manager 5.2 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ーを使用して、以前にインストールした Failover Manager を削除できます。たとえば、次のコマンドを使用して、フェールオーバーマネージャー 5.1 および不要な依存関係を削除します。

- RHEL/Rocky Linux/AlmaLinux 8.x 以降

```
dnf remove edb-efm51
```

- Debian または Ubuntu の場合

```
apt-get remove edb-efm51
```

- SLES の場合

```
zypper remove edb-efm51
```


第4章 Configuring

4.1 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。

i 注釈

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プロモーションのために `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。

4.1.1 カスケードレプリケーションの限定的なサポート

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シナリオでの単純なフェールオーバーの限定的なサポートを提供します。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。

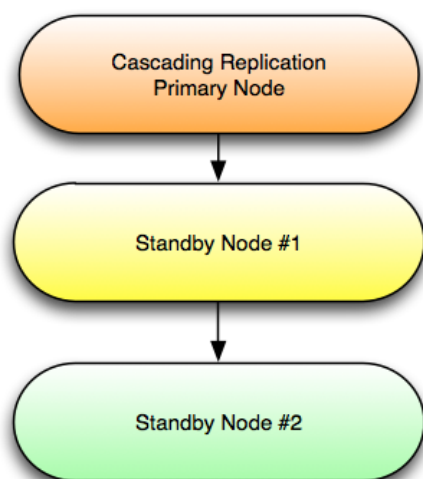


図 1: Cascading replication.

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。スタンバイノード 2 は、3 つのノードを含むようにレプリケーションシナリオを手動で再構成するアクションを実行するまで、新しいプライマリノードの読み取り専用レプリカとして機能し続けます。

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

i 注釈

物理レプリケーションスロットを使用する場合、状況はより複雑です。上の図のスタンバイノード 2 が物理レプリケーションスロットを使用しており、スタンバイノード 1 がプライマリに昇格すると、新しいプライマリエージェント構成されている場合、レプリケーションスロットをクラスター内の他のプロモートブルデータベースノードにコピーします。その後、スタンバイが昇格した場合、スタンバイノード 2 で設定されたスロット名を昇格させず、他のスタンバイエージェントにスロットを進めるように信号を送りません。スタンバイノード 1 を昇格させないように注意するか、他のスタンバイを提供し、1 番をプロモーション不可としてマークするか、Postgresql `max_slot_wal_keep_size` 設定を使用して保持される wal の量を制限する手順を実行するか、非アクティブなノードを手動でドロップする必要があります他のデータベースノードのスロット。このスロットの削除は、[script.post.promotion](#) および [script.remote.post.promotion](#) プロパティを使用して実行できます。物理レプリケーションスロットとフェールオーバーマネージャーでは、カスケードレプリケーションの使用を避けることをお勧めします。

4.2 Configuring Failover Manager

フェールオーバーマネージャーの構成方法については、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](#)

4.2.1 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、それが存在する個々のノードのプロパティを含むプロパティファイルデフォルトでの名前付け `efm.properties` があります。Failover Manager インストーラーは、`/etc/edb/efm-5.<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コピーを作成します。

```
# cp /etc/edb/efm-5.2/efm.properties.in /etc/edb/efm-5.2/efm.properties
```

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

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

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` と命名されます。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後、必要に応じて構成パラメーター値を追加または変更します。各プロパティの詳細については、[クラスタープロパティの指定](#) を参照してください。

Failover Manager サービススクリプトは、`/etc/edb/efm-5.<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は、新しい場所を指定するシンボリックリンクを作成する必要があります。

注釈

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

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する場合は、フェールオーバーマネージャーを再起動して変更を適用する必要があります。

プロパティ値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ではプロパティ値で引用符で囲まれた文字列を使用できません。たとえば、Postgres 構成パラメーターで IP アドレスを次のように指定できます。

```
listen_addresses='192.168.2.47'
```

フェールオーバーマネージャーを使用する場合、値を引用符で囲まないでください。

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

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

凡例 次の表において

- A プライマリまたはスタンバイノードで必要
- W 監視ノードで必要
- S すべてのノードで同じである必要があります
- Y はい

クラスタープロパティ

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、Failover Manager に代わって選択した PostgreSQL コマンドを呼び出すための十分な権限が必要です。詳細については、[前提条件](#) を参照してください。

データベースユーザーのパスワードの暗号化については、[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
```

(次のページに続く)

(前のページからの続き)

```
# 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、standby.signal または recovery.conf ファイルを作成する場所を指定します。このプロパティは、プライマリノードとスタンバイノードが必要です。専用の監視ノードでは必要ありません。

```
# This is the directory where a standby.signal file will exist for a standby node.

# If a primary database fails, a recovery.conf file will be written in this

# location to ensure that the failed database 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 standby.signal

# file. This is most likely the case for Debian installations. The location

# specified will be used as the -D value (the location of the data directory

# for the cluster) when calling pg_ctl to start or stop the database.

# If this property is blank, the db.data.dir location specified by the

# db.data.dir property will be used.

# This corresponds to the output of query show config_file; on respective database.

db.config.dir=
```

データベース構成ファイルの詳細については、PostgreSQL website にアクセスしてください。

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、次の値を設定する必要があります。この行は、クラスタープロパティファイルのどこにでも追加できます。

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

SSL の構成と使用については、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、フェールオーバーマネージャー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. This is required if not using a script.notification
# script. Either/both can be used.
user.email=
```

`from.email` プロパティは、フェールオーバーマネージャーからの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。
- 電子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たとえば `example@%h` などを表すカスタムメールアドレスを指定します。プレースホルダーは、Linux ホスト名ユーティリティによって返された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
# 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.text.prefix` プロパティを使用して、すべての通知の先頭に追加するテキストを指定します。

```
# Text to add to the beginning of every notification. This could
# be used to help identify what the cluster is used for, the role
```

(次のページに続く)

(前のページからの続き)

```
# of this node, etc. To use multiple lines, add a backslash \ to
# the end of a line of text. To include a newline use \n.
# Example:
# notification.text.prefix=Development cluster for Example dept.\n
# Used by Dev and QA \
# See Example group for questions.
notification.text.prefix=
```

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

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

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

```
# This property specifies the ip address and port that jgroups
# will bind to on this node. The value is of the form
# <ip>:<port>.
# Note that the port specified here is used for communicating
```

(次のページに続く)

(前のページからの続き)

```
# 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が管理コマンドをリッスンするポートを指定します。

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

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

```
# Specifies whether or not this is a witness node. Witness nodes
```

(次のページに続く)

(前のページからの続き)

```
# do not have local databases running.

is.witness=
```

EDB Postgres Advanced Server `pg_is_in_recovery()` ファンクションは、データベースの復旧状態を報告するブールファンクションです。このファンクションは、データベースが復旧している場合は `true` を、データベースが復旧していない場合は `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()` ファンクションを呼び出します。

- サーバーが `true` で応答した場合、エージェントはスタンバイのロールを引き継ぎます。
- 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ロールを引き継ぎます。
- ローカルデータベースがない場合、エージェントはアイドル状態になります。

注釈

`is.witness` が `'true'` の場合、フェールオーバーマネージャーは復旧状態をチェックしません。

次のプロパティは、エージェントのローカルデータベース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秒数を指定します。
- `local.timeout` プロパティは、障害チェックを続行する前にデータベースサーバーからの肯定的な応答をエージェントが待機する時間秒単位を制限します。
- `local.timeout.final` プロパティは、以前のチェックが失敗した後の最後の接続試行中にエージェントが待機する時間秒単位を制限します。最後の接続試行が失敗するか、指定された時間内にデータベースが応答しない場合、Failover Manager エージェントはデータベースに到達不能であると考えます。接続が確立されると、エージェントは通知を送信し、監視を再開します。

例、デフォルト値の場合

1. ローカルデータベース `local.period` のチェックは 10 秒ごとに発生します。
2. ローカルデータベースに接続しようとして、60 秒以内に正の結果が返されなかった場合 `local.timeout`、フェールオーバーマネージャーは最後の接続試行を開始します。
3. 最後の接続試行が失敗した場合、または応答が 10 秒以内に受信されなかった場合 `local.timeout.final`、フェールオーバーマネージャーは次のステップに進み、データベースに到達可能かどうかをクラスタの残りの部分に尋ねます。

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
```

(次のページに続く)

(前のページからの続き)

```
# 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。

- エージェントは、ローカルで到達不能になった後、データベースに接続できるかどうかを他のエージェントに尋ねます。
- 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復旧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
```

(次のページに続く)

(前のページからの続き)

```
# 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
```

!!!note タイムアウトプロパティの概要/比較 - `local.*` プロパティは、エージェントのローカルデータベースの障害検出用です。・`node.timeout` プロパティは、他ノードの障害検出用です。 - `remote.timeout` プロパティは、エージェントが他のエージェントからの応答を待機する時間を制限します。

```
# 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、意図せず呼び出されると問題が発生する可能性があります。イーガーフェールオーバーモードでは、この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分離されたことを検出した場合に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
```

(次のページに続く)

(前のページからの続き)

```
# 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、プライマリデータベースがデータベースに到達できない場合にシャットダウンを試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。`true` の場合、フェールオーバーマネージャーは、データベースをシャットダウンしようとした後、`script.db.failure` プロパティで指定されたスクリプトを実行します。

```
# Attempt to shut down a failed 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` に設定されており、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障害として扱います。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ど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すると、残りのクラスターは、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 を使用す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  
  
# cause an unwanted promotion in the case of performing primary
```

(次のページに続く)

(前のページからの続き)

```
# 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誤った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、クラスタの残りの部分には、ユーザーがプライマリデータベースのメンテナンスを実行するなど、プライマリフェールオーバーマネージャエージェントを停止したように見える場合があります。`primary.shutdown.as.failure` プロパティを `true` に設定する場合、メンテナンスの実行には注意してください。

`primary.shutdown.as.failure` が `true` のときにプライマリデータベースのメンテナンスを実行するには、プライマリエージェントを停止し、プライマリエージェント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まで待ちます。次に、プライマリデータベースを停止しても安全です。または、`stop-cluster` コマンドを使用して、障害チェックを実行せずにすべてのエージェントを停止できます。

`update.physical.slots.period` プロパティを使用して、スロットの前進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` を読み取ります。この情報は、`pg_current_wal_lsn` および `primary_slot_name` `postgresql.conf` ファイルで設定されている場合スタンバイに送信します。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既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。スロットに `xmin` 値が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ドロップおよび再作成します。

注必要に応じて現在のプライマリに設定されたスロ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.

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

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

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

(次のページに続く)

(前のページからの続き)

```
#  
  
# Make sure this command returns reasonably quickly - test it  
  
# from a shell command line first to make sure it works properly.  
  
ping.server.command=/bin/ping -q -c3 -w5
```

`auto.allow.hosts` プロパティを使用して、許可ホストリストの設定を開始する最初のノードの `.nodes` ファイルで指定されたアドレスを使用するようにサーバーに指示します。`auto.allow.hosts` を `true` に設定してこのプロパティを有効にすると、クラスターの起動が簡素化されます。

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

`stable.nodes.file` プロパティを使用して、ノードがクラスターに参加または離脱するときにノードファイルを書き換えないようにサーバーに指示します。このプロパティは、IP アドレスが変更されない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
```

(次のページに続く)

(前のページからの続き)

```
# 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。自動再構成を有効にする場合はプロパティを `true` デフォルトに設定し、自動再構成を無効にする場合は `false` に設定します。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。

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

(次のページに続く)

(前のページからの続き)

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

`auto.rewind` および/または `auto.basebackup` プロパティが `true` に設定されている場合、エージェントは `pg_rewind` または `pg_basebackup` を使用して、障害が発生したまたは置換されたプライマリデータベースをスタンバイとして再構成しようとします。適用されるいくつかのケース

- プライマリデータベースの障害新しいプライマリ用に再構成するようにエージェントが通知されると、元のプライマリエージェントは、再構築する必要があるかどうかを確認します。
- 分離されたプライマリノードがクラスターに再接続されると、プライマリエージェントは、新しいプライマリに置き換えられたかどうかを確認し、リビルドまたはプロモーションがなかった場合はプライマリとして再開する必要があるかどうかを確認します。
- 起動時エージェントがクラスター内に既にプライマリデータベースが存在し、ローカルデータベースがスタンバイとして構成されていないことを確認した場合、再構築する必要があるかどうかを確認します。

エージェントは、再構築の必要があることを確認した場合、現在のデータベース構成設定を収集し、以下を実行します。

- `auto.rewind` が `true` に設定されている場合、エージェントは `--dry-run` オプションを指定して `pg_rewind` を実行して、巻き戻しが必要かどうかを確認し、指示されている場合は巻き戻し、監視を再開する前にデータベースをスタンバイとして再構成します。`pg_rewind` の実行中にエラーが発生し、`auto.basebackup` が `true` に設定されている場合、エージェントは `pg_basebackup` でリビルドします。
- `auto.rewind` が `false` に設定され、`auto.basebackup` が `true` に設定されている場合、エージェントは `pg_basebackup` を使用してデータベースを再構築し、監視を再開しようとします。

i 注釈

この機能は、障害が発生したノードをクラスターに自動的に戻す必要があり、障害の原因が既知および予測可能なユースケースを対象としているため、注意して使用します。EFMが障害が発生したプライマリを再構築できず、手動介入がまだ必要である状況が発生する場合があります。

```
# Set either or both of these properties to true to have this agent
# attempt to reconfigure a failed primary database as a standby after
# failover. If both properties are set to true, the agent will attempt
```

(次のページに続く)

(前のページからの続き)

```
# to use pg_rewind first, and then pg_basebackup if the rewind fails.

# See the users guide for more information.

auto.rewind=false
auto.basebackup=false
```

i 注釈

auto.rewind は内部で pg_rewind を使用するため、このパラメーターを設定する前に pg_rewind のすべての前提条件を満たす必要があります。これは、on に set wal_log_hints を設定するか、data_checksums を手動で有効にする必要があることを意味します。

promotable プロパティを使用して、ノードを昇格させないことを示します。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イルオーバーの後に、元のプライマリへのスイッチングが簡単になります。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置換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まり、より早く復旧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プロモートするには、use.replay.tiebreaker を false に設定します。

```
# Use replay LSN value for tiebreaker when choosing a standby to
```

(次のページに続く)

(前のページからの続き)

```
# promote before using failover priority. Set this property to true to

# consider replay location as more important than failover priority

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

# standby to promote.

use.replay.tiebreaker=true
```

`standby.restart.delay` プロパティを使用して、プロモーション後に新しいプライマリに従うように再構成 `stopstarts` 前にスタンバイが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提供できます。

```
# 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.
```

(次のページに続く)

```
application.name=
```

注釈

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

`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。プロパティが空白のままの場合、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 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=
```

`backup_wal` プロパティを使用して、新しいプライマリに従うように再構成中にスタンバイでローカル WAL

のバックアップを作成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。

```
# When a standby is reconfigured to follow a new primary, its local WAL
# is removed. Set this property to true to create a backup of the WAL first.
# This is generally not necessary; the property has been added to allow
# backwards compatibility with the behavior of versions of Failover Manager
# before version 5. If not specified defaults to false.

backup.wal=false
```

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 primaries
# 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
```

`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.
```

(次のページに続く)

(前のページからの続き)

```
# 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=
```

スタンバイノードの数が必要なレベルを下回った場合に、プライマリデータベースを同期レプリケーションモードから解除するには、`reconfigure.sync.primary` プロパティを `true` に設定します。同期レプリケーションを中断せずにスタンバイ数が低下した場合に通知を送信するように `reconfigure.sync.primary` ~ `false` を設定します。

```
# Take the primary database out of synchronous replication mode when

# needed. If set to true, Failover Manager will clear the

# synchronous_standby_names configuration parameter on the primary

# if the number of synchronous standbys drops below the required

# level for the primary to accept writes.

# If set to false, Failover Manager will detect the situation but

# will only send a notification if the standby count drops below the

# required level.

#

# CAUTION: TAKING THE PRIMARY DATABASE OUT OF SYNCHRONOUS MODE MEANS

# THERE MAY ONLY BE ONE COPY OF DATA. DO NOT MAKE THIS CHANGE UNLESS

# YOU ARE SURE THIS IS OK.

reconfigure.sync.primary=false
```

注釈

5.0 より前のフェールオーバーマネージャーバージョンの場合 `reconfigure.num.sync` または ``reconfig-`

ure.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` 値より少なくとも 10 秒小さく設定されていることを確認します。Failover Manager バージョン 5.0 以降、num_sync チェックはクラスターの変更の結果ではなく、定期的に発生します。

check.num.sync.period プロパティを使用して、プライマリデータベースに必要な数より多いまたは少ない同期スタンバイがあるかどうかを確認する秒単位の期間を定義します。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、フェールオーバーマネージャーは必要に応じてプライマリの synchronous_standby_names を変更します。両方のプロパティが false の場合、プライマリに必要な数の同期スタンバイが存在しない場合、フェールオーバーマネージャーは通知を送信します。

```
# Period in seconds between checking if the primary database has more or fewer
# synchronous standbys than it requires. If the reconfigure.num.sync or
# reconfigure.sync.primary properties are used, Failover Manager will change
# synchronous_standby_names on the primary as needed. If the primary
# database is not in synchronous mode, this property has no effect.

check.num.sync.period=30
```

注釈

エージェントログがいっぱいになるのを回避するために、フェールオーバーマネージャーは DEBUG レベルでプライマリノードの同期スタンバイに関する情報をログに記録します。この情報は、任意のノードで`efm cluster-status` コマンドを実行することにより、必要に応じて INFO レベルでログに記録できます。定期的なバックグラウンドステータスチェックによっても、情報がログに記録されます。

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。スタンバイ数が指定された最小値まで低下すると、プライマリノードの障害が発生した場合に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、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、ノード上のデータベースが接続を受け入れる準備をしているときに、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に接続を試行する秒数を指定します。

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

(次のページに続く)

(前のページからの続き)

```
restart.connection.timeout=60
```

`auto.resume.startup.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する秒数を指定します。このプロパティは、IDLE モードで起動するときに適用されます。

```
# Period in seconds for agents to try to resume monitoring 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 up).

auto.resume.startup.period=0
```

`auto.resume.failure.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に適用されます。

```
# Period in seconds for IDLE agents to try to resume monitoring after

# a database failure. 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.failure.period=0
```

!!!注 Failover Manager バージョン 4.x では、両方の動作を制御する単一のプロパティ `auto.resume.period` がありました。構成を 4.x から 5.x にアップグレードする場合、アップグレードユーティリティは `auto.resume.startup.period` およびデフォルトの `auto.resume.failure.period` の元の値をゼロに維持します。

フェールオーバーマネージャーは、仮想 IP を使用するクラスターのサポートを提供します。クラスターが仮想 IP を使用する場合、`virtual.ip` プロパティにホスト名または IP アドレスを指定します。`virtual.ip.prefix` プロパティで、対応する接頭辞を指定します。`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、フェールオーバーが発生した場合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
```

(次のページに続く)

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

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

i 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 エージェントがそれを取得します。プライマリエージェントを停止しても、ノードから VIP はドロップされません。

`check.vip.before.promotion` プロパティを `false` に設定して、フェールオーバーマネージャーが、障害が発生した場合に新しいプライマリに割り当てる前に 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 `to true` に設定します。

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

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

VIP が到達不能になるまでにデフォルトの 60 秒より長くかかる可能性がある場合は、`check.vip.timeout` プロパティを使用して、フェールオーバーマネージャーが待機する最大時間を増やします。

```
# If check.vip.before.promotion is set to true, use this property to set
# the total amount of time in seconds to wait for the VIP (when used) to
# become unreachable.

check.vip.timeout=60
```

`release.vip.*` プロパティは、VIP がノードから解放されるタイミングを制御するために使用できます。詳細については、*Using Failover Manager with virtual IP addresses* を参照してください。

```
# In certain networks, there can be errors trying to connect to remote databases
# at the same time the VIP is being released (i.e. on the primary node during a
# switchover). Set the release.vip.background property to false to have the agent
# pause while the VIP is being released. The pre and post wait periods can add
```

(次のページに続く)

(前のページからの続き)

```
# time (in seconds) to wait before and after the VIP is released in case there  
  
# are other network effects that require them.  
  
release.vip.background=true  
release.vip.pre.wait=0  
release.vip.post.wait=0
```

`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 Pg プール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 Pg プール統合を有効にする場合、PCPPASS ファイルは DB 所有者オペレーティングシステムユーザーが所有し、ファイル許可を 600 に設定する必要があります。

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

次のパラメーターを使用して、Pgpool 統合に使用する値を指定します。

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

(次のページに続く)

(前のページからの続き)

```
# pgpool.bin - Absolute path of pgpool bin directory

# These properties are required if pgpool.enable is set to true.

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

次のプロパティを使用して、スイッチオーバーまたはプライマリ障害シナリオ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、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すると、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、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` プロパティにちなんで名前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指定します。`script.load.balancer.detach` プロパティを使用して、ロードバランサーからノードを切断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。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するには、`%t` プレースホルダーを含めます。

```
# Absolute path to load balancer scripts

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

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

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

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

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

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

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

# and the letter s for standby nodes.

#

# Example:
```

(次のページに続く)

(前のページからの続き)

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

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

5.0 以降のバージョンでは、この値はデフォルトで **false** に設定されます。これは、プライマリエージェントに障害が発生した場合でも、フェールオーバーマネージャーはロードバランサーからノードを切断しないが、データベースにはまだ到達可能であることを意味します。以前のバージョンでは、デフォルトは **true** に設定されていました。値を初期のフェールオーバーマネージャー動作との下位互換性が必要な場合にのみ **true** に設定します。

```
# 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.
```

(次のページに続く)

(前のページからの続き)

```
#
# 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.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
```

(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.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、監視対象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、プライマリがフェールオーバーマネージャークラスターの大部分から分離されていることをプライマリデータベースを監視する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。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
```

(次のページに続く)

(前のページからの続き)

```
# majority of the efm cluster.

script.primary.isolated=
```

`script.remote.pre.promotion` プロパティを使用して、ノードがデータベースをプライマリに昇格させようとしているときに、プロモーションに関係し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指定します。

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

# before a promotion.

#
# This optional user-supplied script will be invoked on other
# agents when a node is about to promote its database. The exit
# code from this script has no effect on Failover Manager, but
# will be included in a notification sent after the script
# executes.

#
# Pass a parameter (%p) with the script to identify the new
# primary node address.

#
# Example:

# script.remote.pre.promotion=/path_name/script_name %p

script.remote.pre.promotion=
```

`script.remote.post.promotion` プロパティを使用して、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指定します。

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

# after a promotion.
```

(次のページに続く)

(前のページからの続き)

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

`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る定期的呼び出すオプション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、スクリプトの実行の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` から `true` を設定して、スクリプトからゼロ以外の終了コードを報告するようにフェールオーバーマネージャーに指示しますが、終了コードの結果としてスタンバイをプロモートしません。

```
# Absolute path to a custom monitoring script.
#
#
# Use script.custom.monitor to specify the location and name of
#
# an optional user-supplied script that will be invoked
#
# periodically to perform custom monitoring tasks. A non-zero
#
# exit value means that a check has failed; this will be treated
#
# as a database failure. On a primary node, script failure will
```

(次のページに続く)

(前のページからの続き)

```
# cause a promotion. On a standby node script failure will
# generate a notification and the agent will become IDLE.
#
# The custom.monitor.* properties are required if a custom
# monitoring script is specified:
#
# custom.monitor.interval is the time in seconds between executions
# of the script.
#
# custom.monitor.timeout is a timeout value in seconds for how
# long the script will be allowed to run. If script execution
# exceeds the specified time, the task will be stopped and a
# notification sent. Subsequent runs will continue.
#
# If custom.monitor.safe.mode is set to true, non-zero exit codes
# from the script will be reported but will not cause a promotion
# or be treated as a database failure. This allows testing of the
# script without affecting EFM.
#
script.custom.monitor=
custom.monitor.interval=
custom.monitor.timeout=
custom.monitor.safe.mode=
```

`sudo.command` プロパティを使用して、拡張されたアクセス許可を必要とする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した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>

# 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複数の孤立する可能性のあるエージェントを起動するのを防ぎます。

```
# 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。ディレクトリが存在しない場合、フェールオーバーマネージャーはディレクトリを作成しようとします。`efm.properties`

ファイルで定義された `log.dir` パラメーターは、EFM ログが保存されるディレクトリパスを決定します。このパラメーターは EFM ログにのみ適用され、他のコンポーネントまたはサービスのロギング構成には影響しません。

EFM バージョン 4.x の起動ログの場所を変更するには、EFM bin ディレクトリにある `runefm.sh` スクリプトを変更します。このスクリプトで `LOG` パラメーターを設定して、目的のログファイルの場所を定義します。

```
# 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>.

# 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 値は、エントリーを作成したプロセスの識別子として使用できます。LOCAL0~LOCAL7 の値を使用します。

```
# 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。ファイルへのロギングを有効にするには、file.log.enabled を true に設定します。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`、`efm.loglevel`、および `jdbc.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 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.

# If there are issues with creating database connections, increasing
# the jdbc.loglevel level will show more information.

# TRACE level logging should be used for diagnosing problems only.

# It is not supported for production use.

jgroups.loglevel=INFO
efm.loglevel=INFO
jdbc.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`

4.2.2 Encrypting your database password

フェールオーバーマネージャーでは、データベースのパスワードをクラスタープロパティファイルに含める前に暗号化する必要があります。`/usr/edb/efm-5.<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、ユーティリティを起動するときにコマンドラインでパスワードを渡すか、`EFMPASS` 環境変数を使用できます。

パスワードを暗号化するには、次のコマンドを使用します。

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

<cluster_name> は、フェールオーバーマネージャークラスターの名前を指定します。

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

```
export EFMPASS=password
```

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

```
# efm encrypt acctg
```

```
This utility will generate an encrypted password for you to place in
your Failover Manager cluster property file:
/etc/edb/efm-5.2/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

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-5.2
```

```
Job for edb-efm-5.2.service failed because the control process exited with error code.
```

(次のページに続く)

(前のページからの続き)

```
See "systemctl status edb-efm-5.2.service" and "journalctl -xe" for details.
```

5. x ではなくバージョン 4.x で Failover Manager サービスを開始したとき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.<x>/startup-efm.log` を参照してください。

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

EFMPASS 環境変数を使用する

この例は、パスワードを暗号化するときには `--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。

4.2.3 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の名前付け `efm.nodes` 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。Failover Manager インストーラーは、`/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。

フェールオーバーマネージャークラスターのインストールが完了したら、テンプレートの作業コピーを作成します。

```
cp /etc/edb/efm-5.2/efm.nodes.in /etc/edb/efm-5.2/efm.nodes
```

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

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

既定では、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` と命名されます。クラスターメンバーファイルに `efm.nodes` 以外の名前を付ける場合、Failover Manager サービススクリプトを変更して、新しい名前を使用するように Failover Manager に指示します。

最初に起動したノードのクラスターメンバーファイルは空でもかまいません。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、少なくともメン

バーシッPCODE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: ポート形式でリストされている必要があります。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、他のエージェント上の `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を反映します。 `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自分の場所を必要としないため、自分自身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脱すると、別のノードがロールを引き継ぎます。 `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。エージェントがダウンしているときにノードがクラスターに参加または離脱する場合、そのエージェントを起動する前に、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。

クラスターに参加するノードのアドレスとポートがわかっている場合は、いつでもクラスターメンバーファイルにアドレスを含めることができます。起動時に、クラスターメンバーを識別しないアドレスは、 *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、エージェントは、クラスターメンバーがクラスターに参加または離脱するときに `.nodes` ファイルを更新しません。この動作は、クラスターメンバーの IP アドレスが頻繁に変更されない場合に最も役立ちます。

4.2.4 Extending Failover Manager permissions

フェールオーバーマネージャー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` 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。

- 仮想 IP アドレスの割り当てまたは解放時に、 `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権限を必要とする管理機能を実行する場合、 `efm` は `efm_root_functions` スクリプトを呼び出します。
- 仮想 IP アドレスを割り当てまたは解放するときに、 `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-50` 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, 2014-2025. All Rights Reserved.

#
```

(次のページに続く)

```
# Do not edit this file. Changes to the file may be overwritten

# during an upgrade.

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

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

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

efm    ALL=(postgres)      NOPASSWD:  /usr/edb/efm-5.2/bin/efm_db_functions
efm    ALL=(enterprisedb)  NOPASSWD:  /usr/edb/efm-5.2/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-5.2/bin/efm_root_functions

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

efm    ALL=(ALL)           NOPASSWD:  /usr/edb/efm-5.2/bin/efm_address

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

efm    ALL=(ALL)           NOPASSWD:  /usr/edb/efm-5.2/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、efm-50 ファイルのコピーを作成します。次に、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。

権限の問題のためエージェントが起動できない場合は、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ます。

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

# includedir /etc/sudoers.d
```

sudo なしでフェールオーバーマネージャーを実行する

既定では、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` を使用せずにフェールオーバーマネージャーを実行するには、フェールオーバーマネージャーに代わって管理機能を実行する権限を持つデータベースプロセスの所有者を選択する必要があります。ユーザーは、デフォルトのデータベーススーパーユーザーたとえば、`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-5.<x>` および `/var/lock/efm-5.<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が作成したログファイルに書き込むことができません。
3. クラスタープロパティテンプレートファイルとノードテンプレートファイルをコピーします。

```
su - enterprisedb

cp /etc/edb/efm-5.2/efm.properties.in <directory>/cluster_name.properties

cp /etc/edb/efm-5.2/efm.nodes.in <directory>/<cluster_name>.nodes
```

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

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

```
/usr/edb/efm-5.2/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しているとき、または `efm` スクリプトを使用しているときは常に、プロパティファイルへのフルパスを提供します。

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-5.<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 アドレスの管理に必要な特権を取得します。

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

セキュリティ上の理由から、`/usr/edb/efm-5.<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` なしでフェールオーバーマネージャーを使用する方法の詳細については、次の Web サイトをご覧ください。

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

4.2.5 Using Failover Manager with virtual IP addresses

フェールオーバーマネージャーは、現在のプライマリノードに要求をルーティングするために仮想 IP アドレス VIP とともに使用できます。

EFM エージェントのシャットダウン中の動作

フェールオーバーマネージャーは、プライマリノードのエージェントがシャットダウンしても、プライマリノードから仮想 IP アドレスをドロップしません。テストの便宜として、プライマリノードのエージェントは、ノードが VIP をまだ持っていない場合は、起動中に VIP を取得しますが、それ以外の場合、フェールオーバーマネージャーの起動と停止は、ノードが仮想 IP アドレスを保持するかどうかに影響を与えません。

これにより、データベースへのアクセスを中断せずに EFM サービスのアップグレードとメンテナンスを実行できます。

ノードがスタンバイからプライマリに昇格する際の動作

VIP は、最初にプライマリノードに割り当てる必要があります。EFM は、プライマリノードのデータベースの障害を検出すると、VIP を解放し、スタンバイノードが新しいプライマリに昇格するときに割り当てます。

EFM は、`ping.server.command` クラスタープロパティで構成されたコマンドを使用して、スタンバイの昇格中に VIP が現在使用されていないことを確認し、`ping` が VIP が到達不能であることを示すまで、新しいプライマリノードをプロモーションしません。`check.vip.before.promotion` クラスタープロパティを使用してこの動作を無効にするか、`check.vip.timeout` クラスタープロパティで待機する時間を増やすことができます。

!!!tip 「ping コマンドの終了コードの意味」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。ゼロの終了コードは、アドレスに到達可能であることを示しますこのコンテキストでは、これは VIP が割り当てられていることを意味します。ゼロ以外の終了コードは、アドレスに到達できないことを示しますこのコンテキストでは、これは VIP が割り当てられていないことを意味します。

これは、標準の `ping(8)` コマンドの動作と一致します。`ping.server.command` クラスタープロパティを介して別のコマンドを構成する場合、それもこの動作に準拠する必要があります。

VIP を解放するときのプライマリエージェントの動作

プライマリエージェントのデフォルト動作は、他のタスクの実行中にバックグラウンドで VIP を解放することです。一部の環境では、これにより、同時に作成されているエージェントからの新しいデータベース接続が妨げられる場合があります。これにより、たとえば、プライマリエージェントが、スイッチオーバー中にプロモートブルなスタンバイデータベースが複数ある場合、プロモートするスタンバイデータベースを選択できなくなります。 `release.vip.*` プロパティを使用してこの動作を変更するには 2 つの方法があります。

1. `release.vip.background=true` のデフォルト値をそのまま使用し、`release.vip.pre.wait` プロパティをゼロ以外の値たとえば 5 秒に設定します。この設定では、VIP をドロップする前にスタンバイが選択されるまでエージェントを待機させることにより、この問題を回避できます。
2. `release.vip.background` のデフォルト値を `false` に変更し、`release.vip.post.wait` プロパティをゼロ以外の値 1 または 2 秒などに設定します。これらの設定により、エージェントは VIP をドロップし、プロモートするスタンバイの選択を続行する前に待機します。

これらのプロパティの効果は、スイッチオーバーを実行することによりテストできます。これにより、プライマリは VIP をドロップし、昇格したノードがそれを取得します。

ノードに複数のアドレスを使用する場合の Postgres の構成

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。

複数のインターフェイスを使用する

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、プライマリの分離とフェールオーバーにつながります。

VIP が `bind.address` とは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 VIP チェックを再試行して、フェールオーバーが予想通りに発生することを確認します。

efm_address スクリプト

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。

スクリプトは次の場所にあります: `/usr/edb/efm-5.<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。

VIP のテスト

フェールオーバーマネージャーで仮想 IPVIP 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。

!!!重要 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てください。

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

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

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、適切なロールプライマリ/セカンダリ/監視用に EFM で構成されたマシンから実行します。

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
__OUTPUT__
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
```

アドレスが未使用の場合、100 のパケット損失が表示されます。

!!!tip 到達不能アドレスの ping コマンド終了コードの意味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は、ゼロ以外の終了コードを返す必要があります。

2. プライマリノードとして構成されたマシンで `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__
<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
__OUTPUT__
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
```

0%のパケット損失が表示され、IP がプライマリノードとして構成されたマシンに到達したことを示します。

!!!tip 到達可能なアドレスの ping コマンド終了コードの意味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ときにゼロ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、VIP が `ip address` コマンド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
__OUTPUT__
```

(次のページに続く)

(前のページからの続き)

```
eth0 Link encap:Ethernet HWaddr 22:00:0A:89:02:8E
inet addr:10.137.2.142 Bcast:10.137.2.191
...
```

この手順からの出力には、eth0 インターフェイスの VIP アドレスは表示されません。

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

```
/bin/ping -q -c3 -w5 172.24.38.239
__OUTPUT__
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 のパケット損失が発生します。すべてのノードでこの手順を繰り返します。

6. すべてのスタンバイノードで手順2、3、および4を繰り返し、VIP をすべてのノードに正常に割り当ておよび解放できることを確認します。任意のノードから VIP に ping を実行して、使用されていることを確認できます。

```
efm_address add4 eth0 172.24.38.239/24
ip address
__OUTPUT__
<output truncated>
eth0 Link encap:Ethernet HWaddr 36:AA:A4:F4:1C:40
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。

4.2.6 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に障害が発生した場合、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。この場合を回避するには、systemd を介してプライマリノードを設定して、プライリエージェントの終了時にフェールオーバーが発生させることができます。これは、イーガーフェールオーバーと呼ばれます。

次の手順を実行して、イーガーフェールオーバーを設定できます。この例では、EDB Postgres Advanced Server を使用します。以下の<x> を適切なバージョン番号に置き換えます。

Eager フェイルオーバーの有効化

- フェールオーバーマネージャーエージェントが停止または障害が発生するとすぐにデータベースサーバーが停止するため、フェールオーバーマネージャーを開始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開始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 が有効になっている場合、`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。エージェントは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できます。

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

- データベースサーバーとローカルフェールオーバーマネージャー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、次のコマンドを使用してサービスの `edb-as-<x>.service` ファイルを編集します。

```
systemctl edit edb-as-<x>.service
```

- テキストエディタに次の行を追加し、保存します。

```
[Unit]
```

```
BindsTo=edb-efm-5.<x>.service
```

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、残りのクラスターはこの状況を障害として扱い、フェールオーバーを試行します。

Eager フェイルオーバーの無効化

- 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-<x>.service` の次の行をコメントアウトします。

```
BindsTo=edb-efm-5.<x>.service
```

- 次のコマンドを実行して、構成ファイルをリロードします。

```
systemctl daemon-reload
```

Eager Failover モードでの 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

重要な注意事項

- `systemd` コマンドは、非 `sudo` 設定でフェールオーバーマネージャーの実行中、データベースの管理に使用されないため、Eager フェールオーバーは `sudo` モードでのみサポートされています。非 `sudo` モードではサポートされていません。
- Eager Failover は、VIP が古いプライマリによって解放されない状況には適していません。
- イーガーフェールオーバーは、次の状況に適しています。
- EDB Postgres Advanced Server の高可用性 setup.
- `jdbc` または `libpq` とのクライアント接続フェイルオーバーを使用するセットアップ
- `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をフェンスする場合 (STONITH)。いくつかの例は、VMWare vCenter 統合、openstack 統合、またはライトアウト管理を使用して 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。

i 注釈

古いプライマリが解放する前に、新しいプライマリが VIP を接続できるようにするには、`check.vip.before.promotion=false` 設定が必要です。

- `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ダウン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、プライマリは障害が発生したプライマリになり、操作可能なスタンバイとして自動的に復旧しません。したがって、クラスターに複数のプロモーション可能なスタンバイが含まれ、スタンバイの合計数が `minimum.standbys` プロパティに指定された値より少なくとも 2 つ多いことを確認します。これは一般的な推奨事項ですが、イーガーフェールオーバーを使用する場合、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フェールオーバーマネージャーが起動します。

i 注釈

`l_blank_9` - 不正なプロパティ値など、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度起動およびシャットダウンします。
`l_blank_8` - Failover Manager プロセスが以前に終了している場合、ロックファイルはまだ存在します、エージェントは自動的に再起動できません。

• `stop-cluster` コマンドを実行した結果、すべてのノードでフェールオーバーマネージャーが停止します。イーガーフェールオーバーモードでは、`stop-cluster` コマンドはフェールオーバーなしで EDB Postgres Advanced Server 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。

4.3 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
```

`$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、次を入力してください。

```
man keytool
```

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

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

第5章 Using

5.1 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。リソースの需要の増加または縮小に応じて、クラスターから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役割に昇格させる場合があります。これが発生した場合、リブートされた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存在す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、データベースがセカンダリプライマリとして起動しないようにします。

5.1.1 フェールオーバーマネージャークラスターの管理

フェールオーバーマネージャークラスターは、一度構成すると、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で時折必要にな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 promote*
- *efm remote-members*
- *efm resume*
- *efm set-priority*
- *efm stop-cluster*
- *efm upgrade-conf*

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/Rocky Linux/AlmaLinux 8.x 以降で 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-5.<x>
```

注釈

バージョン 4.x を使用し、エージェントが起動に失敗した場合、詳細については、起動ログ `/var/log/efm-4.<x>/startup-efm.log` を参照してください。それ以外の場合は、サービスステータスを参照します。

```
systemctl status edb-efm-5.<x>
```

ノードのクラスタープロパティファイルで `is.witness` が `true` が指定されている場合、ノードは監視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()` ファンクションを呼び出します。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、必要に応じて仮想 IP アドレスをノードに割り当てます。サーバーが `true` に応答した場合、Failover Manager エージェントはノードがスタンバイサーバーであると想定します。サーバーが応答しない場合、エージェントはアイドル状態で起動します。

クラスターに参加した後、Failover Manager エージェントは、指定されたデータベース資格情報を確認して、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場合、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ます。

クラスターへのノードの追加

ノードはフェールオーバーマネージャークラスターにいつでも追加できます。クラスターにノードを追加するときは、クラスターを変更して新しいノードを許可し、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サービスの制御の詳細については、[efm allow-node](#) を参照してください。

新しいノードに Failover Manager エージェントをインストールし、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/Rocky Linux/AlmaLinux 8.x 以降で 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-5.<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、ノードは 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。

スタンバイの優先度の変更

フェールオーバーマネージャ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クラスターの既存のメンバーでコマンドを呼び出し、メンバーの IP アドレスの後に優先度の値を指定します。

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

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

スタンバイの優先順位を `0` に設定して、スタンバイをプロモーション不可にすることができます。スタンバイの優先度を `0` より大きい値に設定すると、プロパティ値 `promotable=false` がオーバーライドされます。

たとえば、ノード `10.0.1.10` 上のプロパティファイルが `promotable=false` の設定を含み、`efm set-priority` を使用して `10.0.1.10` の昇格プライオリティをフェールオーバーの際に使用されるスタンバイに設定している場合、`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すると、フェールオーバーマネージャは最初に 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ことを確認し、データ損失の可能性が最も低いノードをプロモートします。2つのスタンバイノードに同等の最新データが含まれて

いる場合、*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優先度値を確認するには、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が変更されます。

efm set-priority コマンドを使用してスタンバイがプロモーション可能かどうかを変更した場合、プロモーションまたはクラスターの分割と再参加を介して、スタンバイのプロパティファイルの値にリセットされる場合があります。エージェントが再起動すると、プロモーション可能ステータスはプロパティファイルの値に戻ります。

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時間帯にのみ手動プロモーションを実行します。使用可能な最新のスタンバイデータベースがない場合、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 を引き継ぎ、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に

- **primary_conninfo** および **restore_command** パラメーターは、既存のスタンバイからプライマリノードにコピーされ、メモリに保存されます。
- プライマリデータベースが停止している。
- VIP を使用している場合、アドレスはプライマリノードから解放されます。
- スタンバイがプライマリノードを置き換えるように昇格し、VIP を取得します。
- 新しいプライマリノードのアドレスは、メモリに保存されている **primary_conninfo** の詳細に追加されます。

- このノードに `application.name` プロパティが設定されている場合、メモリに格納されている `primary_conninfo` 情報に `application_name` プロパティが追加されます。
- メモリに保存されたリカバリ設定は、`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の監視を再開します。

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はない場合、`db.data.dir` プロパティで指定されたディレクトリに `recovery.conf` ファイルが作成されます。`recovery.conf` ファイルは、ファイルが削除されるまで古いプライマリデータベースが起動しないようにするために使用され、ノードがクラスター内の 2 番目のプライマリとして起動しないようにします。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、エージェントがネットワークから分離されていないことを確認する前に、仮想 IP アドレスが使用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。

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、クラスターの実行中のすべてのノードのクラスターメンバーリストからノードのアドレスを削除しますが、フェールオーバーマネージャーの許可ノードホストリストからアドレスは削除されません。

RHEL/Rocky Linux/AlmaLinux 8.x 以降で 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-5.<x>
```

`efm disallow-node` または `efm reset-members` 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、`efm allow-node` コマンドを再度実行せずに、`service edb-efm-5.<x> start` コマンドを使用してエージェントを再起動できます。エージェントをクラスターに追加する予定がない場合は、このエージェントが停止した後に `efm remote-members` コマンドを使用することをお勧めします。

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

フェールオーバーマネージャークラスターの停止

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、次の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のエージェントを終了すると、すべてのフェールオーバー機能が完全に無効になります。

注釈

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。

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 を想定しています。次に、ノードのクラスター名と 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た場合、ノードはクラスターに再参加することはできず、フェールオーバー優先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。

5.1.2 単一のノードで複数のエージェントを実行する

そのフェールオーバーマネージャークラスター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。別の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エージェントが相互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/Rocky Linux/AlmaLinux 8.x 以降のシステムでカスタマイズして、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* および *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 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 監視しています。

これらのデータベースクラスターの両方で 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、一意の名前が必要です。この例では、`acctg` および `sales` データベースクラスターと一致する `acctg.properties` および `sales.properties` を作成します。

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

`admin.port`

`bind.address`

`db.port`

`db.data.dir`

`virtual.ip` 使用する場合

`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` ファイルでは次を指定できます。

`db.user=efm_user`

`db.password.encrypted=7c801b32a05c0c5cb2ad4ffbdba5e8f9a`

`db.port=5444`

`db.database=acctg_db`

`sales.properties` ファイルでは次を指定できます。

`db.user=efm_user`

`db.password.encrypted=e003fea651a8b4a80fb248a22b36f334`

`db.port=5445`

`db.database=sales_db`

一部のパラメーターは、同じノードで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、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ある場合、各ポートは一意である必要があります。任意の 2 つのポートが機能しますが、互いに近すぎないポートを使用する場合、情報を明確に保つことが簡単です。

クラスターごとにクラスタープロパティファイルを作成するときは、`db.data.dir` パラメーターで、各データベースクラスターごとに一意の値を指定する必要があります。

仮想 IP アドレスをノードに割り当てるときに、次のパラメーターを使用します。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、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、各クラスターのプロパティファイルをポイントする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/Rocky Linux/AlmaLinux 8.x 以降を使用している場合、[RHEL/Rocky Linux/AlmaLinux 8.x 以降](#) 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。

RHEL/Rocky Linux/AlmaLinux 8.x 以降

RHEL/Rocky Linux/AlmaLinux 8.x 以降を使用している場合、クラスターごとに一意の新しい名前でサービスファイル `/usr/lib/systemd/system/edb-efm-5.<x>.service` `/etc/systemd/system` にコピーします。

たとえば、フェールオーバーマネージャー 5.2 によって管理される `acctg` および `sales` という名前の 2 つのクラスターがある場合、ユニットファイル名は `efm-acctg.service` および `efm-sales.service` になります。次を使用して作成できます。

```
cp /usr/lib/systemd/system/edb-efm-5.2.service /etc/systemd/system/efm-acctg.service
cp /usr/lib/systemd/system/edb-efm-5.2.service /etc/systemd/system/efm-sales.service
```

次に、`systemctl edit` を使用して、各ユニットファイルの `CLUSTER` 変数を編集し、指定されたクラスター名を `efm` から新しいクラスター名に変更します。また、新しいクラスター名と一致するように `PIDfile` パラメーターの値を更新します。

この例では、`systemctl edit efm-acctg.service` を実行して `acctg` クラスターを編集し、次のように書き込みます。

```
[Service]
Environment=CLUSTER=acctg
PIDFile=/run/efm-5.2/acctg.pid
```

`systemctl edit efm-sales.service` を実行して `sales` クラスターを編集し、次のように書き込みます。

```
[Service]
Environment=CLUSTER=sales
PIDFile=/run/efm-5.2/sales.pid
```

!!!注 `/etc/systemd/system` のファイルを直接編集することもできますが、`systemctl daemon-reload` を実行する必要があります。 `systemd edit` を使用してオーバーライドファイルを変更する場合、この手順は必要ありません。

変更を保存した後、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。例、`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については、

[Understanding and administering systemd](#) を参照してください。

5.2 Monitoring a Failover Manager cluster

Failover Manager `efm cluster-status` コマンドまたは PEM クライアント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。

5.2.1 クラスターステータスレポートの確認

`efm cluster-status` *efm cluster-status* 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ステータスに関する情報を含むレポートを結果ます。

```
# 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用です。

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

(次のページに続く)

(前のページからの続き)

```
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.

cluster status 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	DB	VIP
Standby	172.19.10.2	UP	192.168.225.190
Standby	172.19.12.163	UP	192.168.225.190
Primary	172.19.14.9	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続いていない場合、アドレスはプロパティファイルでノードに関連付けられましたが、アドレスは現在使用されていません。

Failover Manager エージェントは、Cluster Status 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 は、クラスターに参加できるノードとノードのプロモーション順序を確認する簡単な方法を提供します。メンバーシップコーディネーターの IP アドレスもレポートに表示されます。

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

レポートの Promote Status セクションには、クラスター内のプロモーションに関連する情報が含まれています。プロモートするスタンバイを選択するとき、LSN 情報は Standby priority host list とともに使用されます。リプレイ LSN に不一致がある場合、フェールオーバーマネージャーはスイッチオーバーを許可しません。ただし、スタンバイの昇格は常に許可されます。

LSN 情報は、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照会の結果です。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結果です。各データベースへのクエリーは異なるタイミングで返されるため、ストリーミングレプリケーションがクラスターで正常に動作している場合でも、LSN は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      0/4000638          0/4000638
Standby  172.19.12.163    0/4000638          0/4000638
```

(次のページに続く)

(前のページからの続き)

Standby	172.19.10.2	0/4000638	0/4000638
---------	-------------	-----------	-----------

データベースがダウンしている場合、またはデータベースが再起動されたが 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アイドルです。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、アイドルノードの状態の概要が含まれます。例

Agent Type	Address	DB	VIP
Idle	172.19.18.105	UP	172.19.13.105

終了コード

クラスターステータスプロセスは、クラスターの状態に基づいて終了コードを結果ます。

- 0 の終了コードは、すべてのエージェントが実行中であり、プライマリノードとスタンバイノードのデータベースが実行され同期していることを示します。
- ゼロ以外の終了コードは、次の 1 つ以上を示します。
 - クラスターステータスを取得するためのローカルエージェントへの接続に問題があります。
 - データベースがダウンしている、不明な、またはアイドル状態のエージェントがあります。
 - エージェントが DOWN 状態にある。
 - フェールオーバーマネージャーは、提供されたデータベースパスワードを復号できません。
 - WAL の場所を取得するためのデータベースに接続して問題があります。
 - プライマリエージェントがありません。
 - スタンバイエージェントがありません。
 - 1 つ以上のスタンバイノードが同期していません。プライマリ。

!!!注リストの最後にある項目は、レプリケーションに問題があることを示しているわけではありません。これらは、クラスターがフェールオーバーマネージャーがプロモーション/スイッチオーバーを実行するための安全な状態にないことを意味します。

5.2.2 クラスター内の障害が発生したノードの表示

フェールオーバーマネージャーバージョン 5.0 以降、障害が発生したノードはクラスターから自動的に削除されません。エージェントがクラスターから消えると、クラスターステータスレポートには DOWN として含まれます。この場合、データベース情報は提供されませんノードが以前にデータベースノードであった場合。

次のステータスレポートは、障害が発生したノードを含むクラスターを示しています。上部のみが表示されています。DOWN アドレスがスタンバイプライオリティまたはプロモートステータスセクションにないことを除き、残りの情報は変更されません。

Cluster Status: efm

Agent Type	Address	DB	VIP
Primary	172.17.0.2	UP	
Standby	172.17.0.3	UP	
Standby	172.17.0.4	UP	
Standby	172.17.0.5	UP	
DOWN	172.17.0.6	N/A	

DOWN ノードまたはそのノード上の障害が発生したエージェントを再起動しない場合、`efm remote-members` コマンドを使用してノードをクラスターから削除できます。

5.2.3 Postgres Enterprise Managerを使用したストリーミングレプリケーションのモニタリング

Postgres Enterprise ManagerPEM を使用してサーバーをモニタする場合、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するように Streaming Replication Analysis ダッシュボード PEM インターフェイスの一部を構成できます。

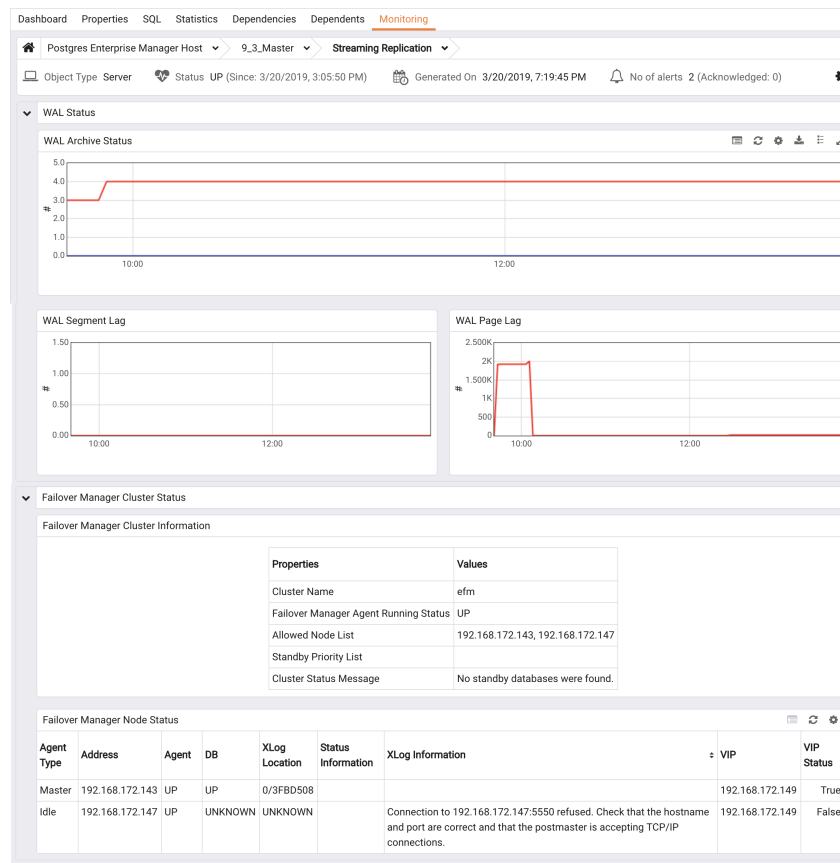


図 1: The Streaming Replication dashboard (Primary node)

Streaming Replication Analysis 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。

レプリケーションスレーブスタンバイノードのダッシュボードを確認すると、ダッシュボードの下部のラベルがサーバーのステータスを確認します。

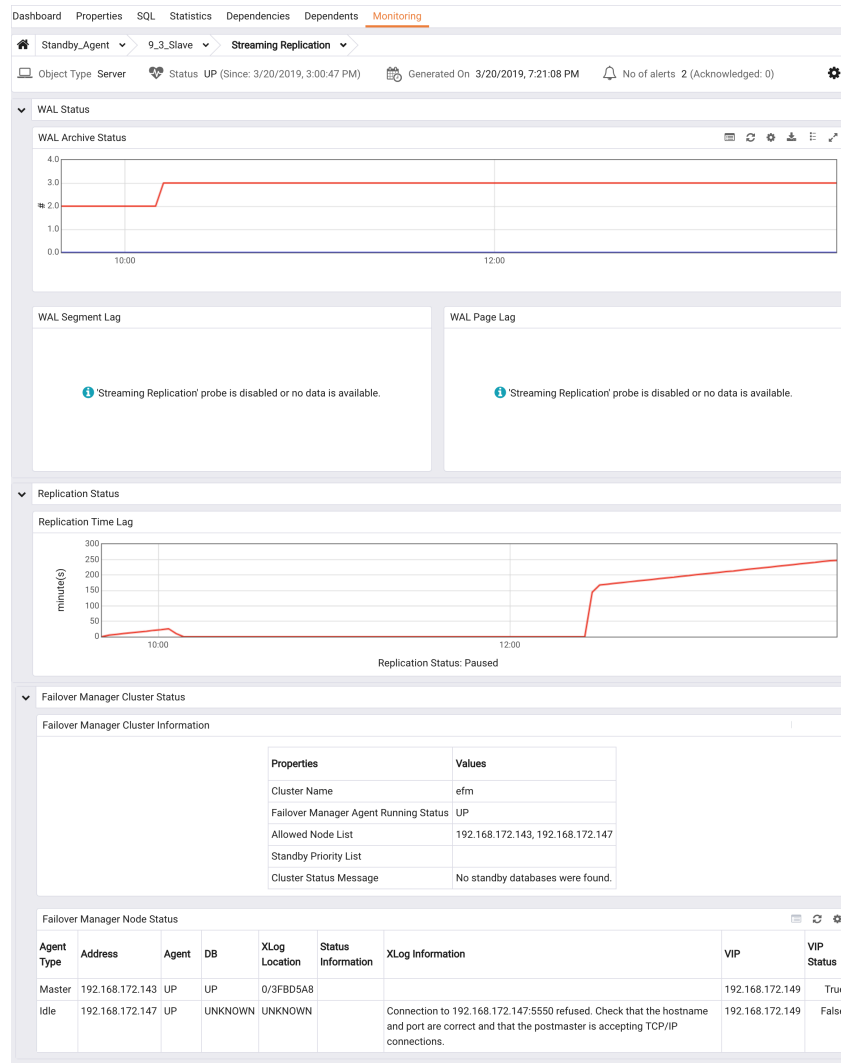


図 2: The Streaming Replication dashboard (Standby node)

デフォルトでは、Streaming Replication Analysis ダッシュボードの情報を提供する 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

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

5.3 Using the efm utility

Failover Manager 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-5.<x>/bin` ディレクトリに追加します。

5.3.1 efm allow-node

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

efm allow-node コマンドを呼び出して、指定されたノードをクラスターに参加させます。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 root によって呼び出される必要があります。

5.3.2 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 root によって呼び出される必要があります。

注釈

クラスターからノードを削除し、再度追加する予定がない場合、代わりに *efm remote-members* コマンドを使用できます。

5.3.3 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状態を表示します。ステータスレポートの詳細については、[Monitoring a Failover Manager cluster](#) を参照してください。

5.3.4 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状態を JSON 形式で表示します。表示される情報の形式は **efm cluster-status** コマンドで生成された表示とは異なりますが、情報ソースは同じです。

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

```

{
  "nodes": {
    "172.16.144.176": {
      "type": "Witness",
      "db": "N\\A",
      "vip": "",
      "vip_active": false
    },
    "172.16.144.177": {
      "type": "Primary",
      "db": "UP",
      "vip": "",
      "vip_active : false"
      "lsnReceive : 0/14001478"
      "lsn : 0/14001478"
      "lsnInfo : "
    },
    "172.16.144.180": {
      "type": "Standby",
      "db": "UP",
      "vip": "",
      "vip_active : false"
      "lsnReceive : 0/14001478"
      "lsn : 0/14001478"
      "lsnInfo : "
    }
  },
  "allowednodes": [
    "172.16.144.177",
    "172.16.144.160",
    "172.16.144.180",
    "172.16.144.176"
  ],
  "membershipcoordinator": "172.16.144.177",
  "failoverpriority": [
    "172.16.144.180"
  ],
  "minimumstandbys": 0,
  "missingnodes": [],
  "messages": []
}

```

5.3.5 efm create-standby

!!!注 バージョン 5.0 のみの重要な情報このコマンドがスタンバイノードで実行される場合ローカルデータベースが実行され、監視されている場合、コマンドが完了した後にエージェントを再起動する必要があります。エージェントが誤った内部状態のままになる可能性があるという既知の問題があります。このノードが後でプライマリに昇格し、データベースに障害が発生した場合、この状態によりフェールオーバーが発生しない場合があります。これは、「アイドル」ノードローカルデータベースが実行されていない、または監視されていないノードには影響しません。5.1 リリースで修正されました。

efm create-standby <cluster_name> [-prompt] [-slot <slot_name>] [-waldir <directory>]

efm create-standby コマンドを呼び出して、このノードに新しいスタンバイデータベースを作成します。ローカルエージェントが実行されており、クラスター内にプライマリエージェントが存在する必要があります。コマンドプロセスは次のことを行います。

- ローカルエージェントに接続して、現在のプライマリデータベースを見つけます。
- ローカルデータベースを停止し、ローカルエージェントをアイドル状態にします必要な場合。
- 現在の `synchronous_standby_names` 構成設定を読み取ります。
- `slot` パラメーターが指定された場合、プライマリデータベースに接続して、その名前のスロットが存在する場合は削除し、スタンバイが使用するこの名前新しいスロットを作成します。
- ローカルデータディレクトリを削除します。 `waldir` パラメーターが指定された場合、現在のログ先行書き込みディレクトリも削除されます。
- `pg_basebackup` を実行して、スタンバイデータベースを作成します。必要に応じて `pg_basebackup` の `--waldir` パラメーターを使用します。
- 必要に応じて、 `synchronous_standby_names` 構成設定を設定します。
- スタンバイデータベースを起動し、監視を再開します。

`-prompt` オプションが指定されている場合、コマンドは続行する前に、生成された `pg_basebackup` コマンドを含む実行されるステップを出力します。次のエージェントプロパティが使用されます。

- `db.service.owner` は、 `pg_basebackup` コマンドを実行するユーザーを指定します。
- `sudo.user.command` は、上記のユーザーとしてコマンドを実行する方法を指定します。
- `db.bin` は、 `pg_basebackup` の場所を指定します。
- `db.data.dir` は、ターゲットディレクトリを指定します。
- `db.port` は、プライマリデータベースへのアクセスに使用されるポートを指定します。
- `application.name` は、使用する `application_name` を指定します設定されている場合。

!!!注 このコマンドは、フェールオーバーマネージャー 5.0 で導入されました。5.0 のみで、コマンドを実行するにはスーパーユーザー権限が必要です。

次の例は、 `-prompt` および `-slot` オプションの両方が指定された場合のコマンド出力を示しています。

```
# /usr/edb/efm-5.2/bin/efm create-standby efm -slot s2 -prompt
```

(次のページに続く)

(前のページからの続き)

```

Found primary node1 from cluster status.
Verify primary address node1 does not match this agents bind address node2 or
↳external address .
Will signal local agent to run database stop command and become idle if not already.
Will connect to primary on node1 to drop slot s2 if it exists.
Will remove the /opt/postgres/data/pg_wal and /opt/postgres/data directories, and run
↳pg_basebackup using the following parameters:
- R -D /opt/postgres/data -X stream -S s2 -C
...with connection string: host=node1 port=5432 application_name=node2
Will set synchronous_standby_names to: any 2 ("node1", "node3", "node4")

Do you want to continue? [y/N]:y
Signalling local agent to stop db and become idle if needed.
Replication slot s2 does not exist on primary node1.
Removing directories/files and running pg_basebackup.
Starting database.
Waiting briefly for database to finish startup.
Attempting to resume local efm agent monitoring.
Resume command successful on local agent.

```

5.3.6 efm encrypt

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

クラスタプロパティファイルにパスワードを含める前に、`efm encrypt` コマンドを呼び出してデータベースパスワードを暗号化します。--from-env オプションを含めて、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。

5.3.7 efm promote

```
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のメンテナンス時間帯に、クラスタ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` コマンドで報告された場合にのみ、手動プロモーションを試行します。

-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、スタンバイにする古いプライマリノードにリカバリ設定をコピー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 root によって呼び出される必要があります。

i 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。

5.3.8 efm remote-members

`efm reset-members <cluster_name>`

`efm reset-members` コマンドを呼び出して、キャッシュされたノードアドレスをフェールオーバーマネージャークラスターから削除します。ノードがクラスターから完全に削除された後にこのコマンドを実行して、クラスターが削除されたノードのアドレスに接続しようとしないようにします。これにより、DOWN ノードに障害が発生した後、このクラスターから切断された後も削除されます。

このコマンドを実行すると、クラスター内の各ノードで次のことが行われます。

1. `.nodes` ファイルのアドレスを現在のクラスターメンバーにリセットします。これは、`stable.nodes.file` プロパティが `true` に設定されている場合でも発生します。
2. 許可ノードホストリストを更新して、現在のメンバーのみを含めます。
3. すべてのエージェントを互いに一時的に切断し、再接続します。

実行中のデータベースは、この操作の影響を受けません。操作が完了したら、スタンバイ優先度リストの更新が必要になる場合があります。詳細については、[*efm set-priority*](#) コマンドを参照してください。

5.3.9 efm resume

`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 root によって呼び出される必要があります。

5.3.10 efm set-priority

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

`efm set-priority` コマンドを呼び出して、フェールオーバー優先度をスタンバイノードに割り当てます。値は、フェールオーバー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 root 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、優先度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、フェー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 の優先度値は、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。

5.3.11 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ールオーバーマネージャーを停止します。このコマンドは、フェールオーバーマネージャーに、クラスター内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します。このコマンドは実行中のデータベースには影響しませんが、コマンドが完了すると、フェールオーバー保護は実施されません。

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 root によって呼び出される必要があります。

5.3.12 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。デフォルトモードでフェールオーバーマネージャーを実行している場合、このコマンドはスーパーユーザー特権で呼び出す必要があります。

オプションの `-source` フラグの詳細について、または `sudo` を使用しないフェー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、[Eager Failover モードでの Failover Manager のアップグレード](#) を参照してください。

5.3.13 efm node-status-json

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

`efm node-status-json` コマンドを呼び出して、ローカルノードの状態を 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するか、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。

次に、`efm node-status-json` コマンドの出力例を示します。

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

5.3.14 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ー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。

5.4 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エージェントをホストします。デフォルトでは、サービススクリプトは以下を検出することを想定しています。

- Failover Manager サービス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、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。

5.4.1 RHEL/Rocky Linux/AlmaLinux 8.x 以降での `systemctl` ユーティリティの使用

RHEL/Rocky Linux/AlmaLinux 8.x 以降では、Failover Manager は、`/usr/lib/systemd/system` にあるデフォルトで **edb-efm-5.<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ーによって監視される各データベースクラスターは、レプリケーションクラスターの各ノードでサービスのコピーを実行します。

次の `systemctl` コマンドを使用して、RHEL/Rocky Linux/AlmaLinux 8.x 以降のホストにある Failover Manager エージェントを制御します。

```
systemctl start edb-efm-5.<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エージェントはローカルデータベースを監視し、他のノード上のフェールオーバーマネージャーと通信します。フェールオーバーマネージャークラスター内のノードは、任意の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-5.<x>
```

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

```
systemctl status edb-efm-5.<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結果ます。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-5.2
● edb-efm-5.2.service - EnterpriseDB Failover Manager 5.2
   Loaded: loaded (/usr/lib/systemd/system/edb-efm-5.2.service; disabled; preset:↵
   ↵disabled)
   Active: active (running) since Mon 2025-11-24 16:00:08 UTC; 10s ago
   Process: 11755 ExecStart=/bin/bash -c /usr/edb/efm-5.2/bin/runefm.sh start $
   ↵{CLUSTER} (code=exited, status=0/SUCCESS)
   Main PID: 11837 (java)
   Tasks: 46 (limit: 79998)
   Memory: 195.6M
   CPU: 2.989s
   CGroup: /docker/7ce08d0a35648d9156b56ef27a1bfced6af013a3e174f756681612002ac85961/
   ↵system.slice/edb-efm-5.2.service
           └─ 11837 /usr/lib/jvm/java-11-openjdk-11.0.25.0.9-7.el9.aarch64/bin/
   ↵java -cp /usr/edb/efm-5.2/lib/EFM-5.2.jar -Xmx128m com.enterprisedb.efm.main.
   ↵ServiceCommand __int_start /etc/edb/efm-5.2/efm.properties

Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 -----
   ↵-----
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Checking shared properties↵
   ↵with node2-51572(node2)
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Getting remote database↵
   ↵addresses to check.
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Addresses to check after↵
   ↵adding standbys: [node2-51572(node2), node3-8868(node3), node4-56220(node4)]
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Testing remote database↵
   ↵connections.
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Checking host node2 for↵
   ↵address: node2-51572(node2)
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Checking host node3 for↵
   ↵address: node3-8868(node3)
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Checking host node4 for↵
   ↵address: node4-56220(node4)
Nov 24 16:00:08 node1 bash[11820]: 2025-11-24 16:00:08 Now monitoring database.
Nov 24 16:00:08 node1 systemd[1]: Started EnterpriseDB Failover Manager 5.2.
```

5.5 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つのログファイルを/var/log/efm-<version>/に書き込み、保存します。

The cluster properties file の `jgroups.loglevel`、`efm.loglevel`、および `jdbc.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.
```

(次のページに続く)

```
# 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.

# If there are issues with creating database connections, increasing
# the jdbc.loglevel level will show more information.

# TRACE level logging should be used for diagnosing problems only.

# It is not supported for production use.

jgroups.loglevel=INFO
efm.loglevel=INFO
jdbc.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。

!!!重要 TRACE レベルのログは、特定の問題を診断する場合にのみ使用してください。これにより、Failover Manager エージェントの作業が増加し、このレベルのロギングでの動作は定義されていません。実稼働使用はサポートされていません。

デフォルトでは、フェールオーバーマネージャーログファイルは毎日ローテーション、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-5.<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
```

5.5.1 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 を設定して UDP または TCP 接続を許可する必要があります。

syslog への接続を許可するには、`/etc/rsyslog.conf` ファイルを編集し、使用するプロトコルのコメントを外します。また、プロトコルに関連付けられている `UDPServerRun` または `TCPServerRun` エントリに、ログエントリの送信先のポート番号が含まれていることを確認します。たとえば、次の構成ファイルエントリは、ポート 514 への UDP 接続を有効にします。

```
# Provides UDP syslog reception

$ModLoad imudp
$UDPServerRun 514
```

次の構成ファイルエントリは、ポート 514 への TCP 接続を有効にします。

```
# Provides TCP syslog reception

$ModLoad imtcp
$InputTCPServerRun 514
```

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

```
systemctl restart rsyslog.service
```

フェールオーバーマネージャーホストで `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プロパティを変更してロギングを有効にします。*modify the properties file* `/etc/edb/efm-5.<x>/efm.properties.in` のエディターの選択を使用し、実装するロギングのタイプを指定します。

```
# Which logging is enabled.

file.log.enabled=true
syslog.enabled=false
```

システムの *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 ホストのリスナーポートを指定します。syslog.facility 値は、エントリーを作成したプロセスの識別子として使用できます。LOCAL0 から LOCAL7 までの値を使用します。

```
# 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.
```

(次のページに続く)

```
# 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。

man syslog

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

```
user.email  
from.email  
notification.level  
notification.text.prefix  
script.notification
```

構成プロパティの編集の詳細については、[クラスタープロパティの指定](#)を参照してください。

通知の本文には、通知をトリガーしたイベント、およびクラスターの現在の状態に関する詳細が含まれています。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の増加を示します。

- **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、管理者がシステムを確認する必要があるイベントが発生したことを示しますたとえば、フェールオーバーが発生しました。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したたとえば、フェールオーバーが試行されましたが、完了できません。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必要としますが、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操作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。すべての通知は、構成ファイルで指定されたログレベルの詳細に関係なく送信されます。

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

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-5.<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
Executed fencing script	実行されたフェンシングスクリプト <i>script_name</i> 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 <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 <i>script_name</i> 結果 <i>script_results</i>
Executed remote post-promotion script	リモートポストプロモーションスクリプト <i>script_name</i> を実行しました 結果 <i>script_results</i>
Executed post-database failure script	実行されたデータベース障害後のスクリプト <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 <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、データベースの状態が監視されています。
Idle agent running on node <i>node_address</i> for cluster <i>cluster_name</i>	アイドルエージェントが実行されています。ローカルデータベースを起動した後、エージェントを再開できます。
Assigning VIP to node <i>node_address</i>	VIP <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> を解放します 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
Starting auto resume check for cluster <i>cluster_name</i>	このノードのエージェントは、`auto.resume.failure.period` 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はスクリプト <i>script_name</i> を再開しました 結果 <i>script_results</i>
WAL logs backed up during promotion	新しいプライマリに従ってこのスタンバイを再構成すると、pg_wal コンテンツが pgdata ディレクトリにバックアップされました。都合の良いときにこのバックアップを削除して、ディスク領域を解放します。
Reset members completed	クラスターメンバーをリセットする呼び出しの後、エージェントはクラスターに再参加しました。

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
Failed primary is being rebuilt with pg_basebackup in cluster <i>cluster_name</i>	このノードのプライマリデータベースは、 <i>node_address</i> の現在のプライマリに従って、pg_basebackup で再構築されています。
Witness agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	Witness エージェントが終了しました。
Primary agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。
Cluster <i>cluster_name</i> notified that primary agent has left	プライマリ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。

次のページに続く

表 2 – 前のページからの続き

Subject	Description
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、起動中に生成され、起動が完了する前にエージェントが終了します。
Virtual IP address assigned to non-primary node	仮想 IP アドレスは、非プライマリノードに割り当てられているようです。競合を回避するために、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 アドレスは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 を再取得しようとします。
Standby failed in cluster <i>cluster_name</i>	<i>node_address</i> のスタンバイがクラスターから離脱しました。
Standby agent failed for cluster <i>cluster_name</i>	<i>node_address</i> のスタンバイエージェントはクラスターから離脱しましたが、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。他のクラスター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を通知しましたが、他のノードは、スタンバイデータベースがまだ実行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>cluster_name</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。表示されている現在のノードは次のとおりです。 <i>node_list</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 がクラスターを離脱しました。
Idle agent failure for cluster <i>cluster_name</i> .	<i>node_address</i> で実行されているアイドルエージェントがクラスターから離脱しました。
One or more nodes isolated from network for cluster <i>cluster_name</i>	このノードはネットワークから分離されているようです。クラスター内で見られる他のメンバーは次のとおりです。 <i>node_list</i>
Node no longer isolated from network for cluster <i>cluster_name</i> .	このノードはネットワークから分離されません。

次のページに続く

表 2 – 前のページからの続き

Subject	Description
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プロモーション手順を開始しましたが、プライマリ DB がまだ <i>node_address</i> で実行されていることを検出しました。これは通常、プライマリフェールオーバーマネージャーエージェントが終了したことを示します。フェイルオーバーは発生していません。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脱しています。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。
Standby agent started to promote, but primary has rejoined.	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させ始めましたが、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を検出しました。フェイルオーバーは発生していません。
Standby agent tried to promote, but could not verify primary DB	スタンバイ Failover Manager エージェントは自分自身を昇格させようとしたが、プライマリ DB が ノードアドレス でまだ実行されているかどうかを検出できませんでした。フェイルオーバーは発生していません。
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であるため、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、データの破損が発生する可能性があります。フェイルオーバーは発生していません。
Standby agent tried to promote, but appears to be orphaned	スタンバイ Failover Manager エージェントは自分自身を昇格させようとしたが、既知のサーバー アドレス に到達できなかったため、できませんでした。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェイルオーバーは発生していません。
Potential manual failover required on cluster <i>cluster_name</i> .	潜在的なフェイルオーバー状況がクラスター <i>cluster_name</i> で検出されました。このクラスターでは自動フェイルオーバーが無効になっているため、手動介入が必要です。
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> のロックファイルが、ノード <i>node_address</i> の <i>path_name</i> から削除されました。このロックにより、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することを防ぎます。このファイルを復元して、クラスター用の別のエージェントを誤って起動するのを防ぎます。
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、プライマリノード <i>node_address</i> の <i>path</i> で見つかりました。これにより、このノードで DB を再起動しようすると、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	<i>recovery_target_timeline</i> パラメーターは、リカバリ設定で最新の状態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>	プロモーションが試行されましたが、既に昇格するノード <i>node_address</i> が存在します。

次のページに続く

表 2 – 前のページからの続き

Subject	Description
Standby will not be re-configured after failover in cluster <i>cluster_name</i>	このノードの `auto.reconfigure` プロパティは <code>false</code> 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に従うように再構成されません。
Could not resume replay for standby <i>node_address</i>	スタンバイの再生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。エラー出力 <i>error</i> 。
Possible problem with database timeout values	`remote.timeout` 値 <i>value</i> は、`local.timeout` 値 <i>value</i> より大きいです。ローカルデータベースの応答に時間がかかりすぎる場合、ローカルエージェントは、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他のエージェントは接続できます。これはフェールオーバーを発生させませんが、ローカルエージェントの監視を強制的に停止する場合があります、フェールオーバー保護がないままになります。
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は、最小数 <i>number</i> に減少しました。別のスタンバイノードが追加されるか、プロモーション可能にされない限り、フェールオーバーはできません。
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> の唯一のプロモーション可能なスタンバイを削除しました。別のプロモート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リストに追加することによりプロモーション可能にされない限り、フェールオーバーはできません。
Synchronous replication has been reconfigured for cluster <i>cluster_name</i>	クラスター内の同期スタンバイノードの数が <i>number</i> を下回りました。プライマリの同期スタンバイ名が <i>new_value</i> に再構成されました。
Synchronous replication has been reconfigured for cluster <i>cluster_name</i>	プライマリの <i>synchronous_standby_names</i> が <i>new_value</i> に再構成されました。
Synchronous replication has been disabled for cluster <i>cluster_name</i> .	クラスター内の同期スタンバイノードの数が <i>number</i> を下回りました。プライマリは同期レプリケーションモードから解除されました。
Could not reload database configuration.	データベース構成をリロードできませんでした。手動介入が必要です。エラー出力 <i>error</i> 。
Custom monitor timeout for cluster <i>cluster_name</i>	次のカスタム監視スクリプトは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> 。出力 <i>script_results</i>
<i>primary.shutdown.as.failure</i> set to true for primary node	このクラスターの `primary.shutdown.as.failure` プロパティは <code>true</code> に設定されています。クラスター全体を停止せずにプライマリエージェントを停止すると、クラスターの残りの部分によって、即時のプライマリエージェントの障害として扱われます。プライマリデータベースのメンテナンスが必要な場合は、プライマリエージェントをシャットダウンし、残りのノードからフェールオーバーが発生しないという通知を待ちます。

次のページに続く

表 2 – 前のページからの続き

Subject	Description
<i>Primary_or_Standby</i> cannot ping local database for cluster <i>cluster_name</i>	<i>Primary_or_Standby</i> 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す。他のノードはデータベースにリモートでアクセスできるため、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。
Standby cannot resume monitoring local database for cluster <i>cluster_name</i>	スタンバイエージェントは、 <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ません。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。`resume` コマンドが実行されてデータベースの監視が再開されるまで、スタンバイエージェントは IDLE のままです。
Primary agent left the cluster and node detached from load balancer	スタンバイエージェントは、 <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ません。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。`resume` コマンドが実行されてデータベースの監視が再開されるまで、スタンバイエージェントは IDLE のままです。
Standby database in cluster <i>cluster_name</i> not stopped before primary is promoted	このエージェントには `standby.restart.delay` プロパティが設定されているため、プロモーションが完了してから <i>num_seconds</i> 秒まで、新しいプライマリに従うように再構成されません。場合によっては、手動介入なしでは新しいプライマリを追跡できない場合があります。
Agent for cluster <i>cluster_name</i> could not load shared state	このエージェントは、コーディネーターから共有クラスターの状態をロードできませんでした。これは予期しないものであり、修正しない限りクラスターで不明な動作を発生させる可能性があります。このノードからクラスターステータスを確認して、予想どおりであることを確認することをお勧めします。そうでない場合、このエージェントを再起動して、正しい内部状態があることを確認できます。詳細については、エージェントログを確認してください。
Standby database restarted but EFM cannot connect	データベースの <i>start</i> または <i>restart</i> 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ません。EFM は、最大 <i>num_seconds</i> 秒間接続を試行し続けます。
Could not retrieve db settings for rebuild in cluster <i>cluster_name</i>	プライマリデータベースを再構築しようとしたエージェントは、以前の db 設定を取得できませんでした。デフォルト値で <i>pg_basebackup</i> を使用して、リビルドしようとします。エラーは以下です。詳細については、エージェントログを参照してください。エラー出力 <i>error</i>

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
<i>pg_basebackup</i> call failed in cluster <i>cluster_name</i>	障害が発生したプライマリをスタンバイとして再構築中に、 <i>pg_basebackup</i> への呼び出しがエラーコードで終了しました。詳細については、エージェントログを参照してください。エラー出力 <i>error</i>
A duplicate replication slot name is in use for cluster <i>cluster_name</i>	クラスター内のスタンバイは、プライマリデータベースと同じ物理レプリケーションスロット名 <i>slot_name</i> を使用しています。これはサポートされていないため、同じ名前を使用するスタンバイでスロットは作成されません。元のプライマリは、昇格した場合、このスタンバイに従うことはできません。
Unable to connect to DB on <i>node_address</i>	最大接続制限に到達しました。

次のページに続く

表 3 – 前のページからの続き

Subject	Description
Unable to connect to DB on <i>node_address</i>	db.user= <i>user_name</i> のパスワードが無効です。
Unable to connect to DB on <i>node_address</i>	無効な承認仕様です。
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ません。他のノードはデータベースにリモートでアクセスできるため、プライマリはVIPを解放せず、 <code>recovery.conf</code> ファイルを作成しません。 <code>resume</code> コマンドが実行されてデータベースの監視が再開されるまで、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	ポスト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>
Remote post-promotion script failed	リモートポストプロモーションスクリプト <i>script_name</i> 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> ノード <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> ノード <i>node_address</i>
Post-database failure script error	データベース障害後のスクリプト <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	エージェントが再開したスクリプト <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>
Could not promote standby	ノードで promote コマンドは失敗しました。スタンバイを昇格できませんでした。エラー出力 <i>error</i>
Error creating recovery.conf file on <i>node_address</i> for cluster <i>cluster_name</i>	プロモーション中にプライマリノード <i>node_address</i> で <code>recovery.conf</code> ファイルの作成中にエラーが発生しました。プロモーションは継続されますが、古いプライマリノードを再起動できないように手動介入が必要です。エラー出力 <i>error</i>
An unexpected error has occurred for cluster <i>cluster_name</i>	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。エラー出力 <i>error</i>
Primary database being fenced off for cluster <i>cluster_name</i>	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。クラスターは、 <i>ip_address</i> 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、フェールオーバーマネージャークラスターの残りの部分がスタンバイをプロモートするときに2つのプライマリを防ぐ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ークラスターによってシャットダウンされました。
Primary database being fenced off for cluster <i>cluster_name</i>	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。

次のページに続く

表 3 – 前のページからの続き

Subject	Description
Could not assign VIP to node <i>node_address</i>	フェールオーバーマネージャーは、何らかの理由で 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 <i>cluster_name</i>	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 <i>cluster_name</i>	このエージェントは、ローカルデータベースに到達しようとしてタイムアウトしました。タイムアウトの後、エージェントはデータベースを正常に ping し、監視を再開しました。ただし、ノードをチェックして、データベースまたはエージェントに障害が発生する可能性を防ぐために、通常に実行されていることを確認します。
Resume timed out for cluster <i>cluster_name</i>	このエージェントは、ローカルデータベースを再構成および再起動した後、監視を再開できませんでした。詳細については、エージェントログを参照してください。
Internal state mismatch for cluster <i>cluster_name</i>	フェールオーバーマネージャークラスターの内部状態は、クラスターメンバーの実際の状態と一致しませんでした。これはまれで、ノードがクラスターに参加するか、状態を変更するタイミングの問題が発生する可能性があります。問題は解決される必要がありますが、確認のためにクラスターステータスもチェックしてください。不一致の詳細は、エージェントログファイルで見つけることができます。
Failover has not occurred	エージェントは、プライマリデータベースがクラスター <i>cluster_name</i> で利用できない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> で利用できない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もはや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してい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、 <i>node_address</i> で実行されているデータベースに接続できません。これが修正されるまで、このノードはデータベースが実行されているかどうかを確認できないため、フェールオーバーが適切に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ーは、スタンバイを昇格させようとしています。スクリプトの場所 <i>script_name</i> スクリプト出力 <i>script_results</i>
Loopback address set for <i>ping.server.ip</i>	`ping.server.ip` プロパティにはループバックアドレスが設定されます。この設定はネットワーク分離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。

次のページに続く

表 3 – 前のページからの続き

Subject	Description
Load balancer attach script error	ロードバランサー接続スクリプト <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>
Load balancer detach script error	ロードバランサー切断スクリプト <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>
Pgpool attach node error	フェールオーバーマネージャーは pg プールノードの接続に失敗しました。終了値 <i>exit_code</i> 。結果 <i>script_results</i>
Pgpool detach node error	フェールオーバーマネージャーは pg プールノードの切断に失敗しました。終了値 <i>exit_code</i> 。結果 <i>script_results</i>
Not enough synchronous standbys available in cluster <i>cluster_name</i>	クラスター内の同期スタンバイノードの数は <i>number</i> に減少しました。十分な同期スタンバイノードが追加されるまで、プライマリ上のすべての書き込みクエリーはブロックされます。

5.7 Supported failover and failure scenarios

フェールオーバーマネージャーは、フェールオーバーの原因となる可能性のある障害が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、特定の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場合があります。

- プライマリデータベースがクラッシュするか、シャットダウンされた場合。
- プライマリデータベースをホストするノードがクラッシュするか、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、これらの条件の精度を確認するためにあらゆる試みを行います。プライマリデータベースまたはノードに障害が発生したことをエージェントが確認できない場合、フェールオーバーマネージャーはクラスターでフェ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がフェールオーバー条件を監視および検出するが、スタンバイへの自動フェールオーバーを実行しない場合に、*no auto- フェールオーバー* モードもサポートします。このモードでは、フェールオーバー条件が満たされると、管理者に通知が送信されます。自動フェールオーバーを無効にするには、クラスタープロパティファイルを変更し、*auto.failover* 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。

5.7.1 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、フェールオーバーマネージャーは障害を確認するプロセスを開始します。

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。

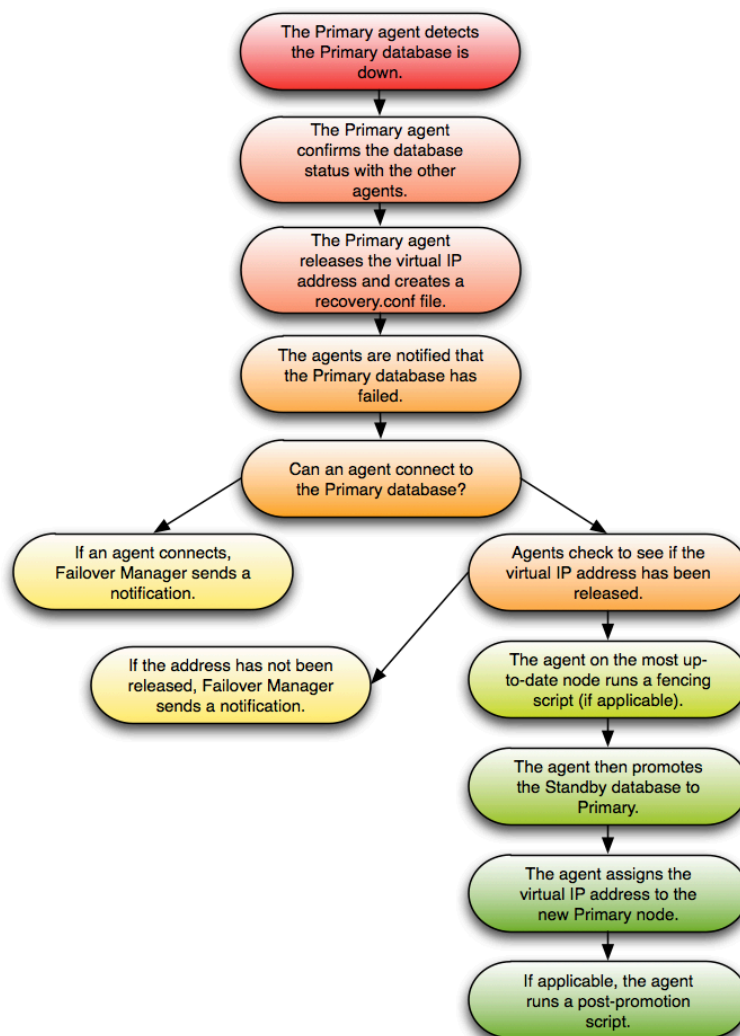


図 3: Confirming the failure of the primary database.

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。必要に応じて、エージェントはプロモーション後のスクリプトを実行します。

5.7.2 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復旧するには

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。

5.7.3 ノードをプライマリのロールに戻す

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

1. クラスターに複数のスタンバイノードがある場合、`efm set-priority` コマンドを使用してノードのフェールオーバー優先度を 1 に設定します。
2. `efm promote` コマンドを呼び出して、ノードをプライマリノードの元の役割に昇格させます。

!!!note デフォルトでは、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スタンバイになるように再構築しません。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、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_rewind` を使用してサーバーを復元することもできます。

バージョン 5.1 以降、フェールオーバーマネージャーが障害が発生したプライマリデータベースを再構築しようとする機能を有効にできます。この機能は、障害が発生したノードをクラスターに戻す必要があり、障害の原因が既知および予測可能なユースケースを対象としているため、注意して使用します。EFM が障害が発生したプライマリを再構築できない状況が発生する場合があります、引き続き手動介入が必要です。詳細については、[the auto.rewind cluster property](#) を参照してください。

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。

5.7.4 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

フェールオーバーマネージャー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

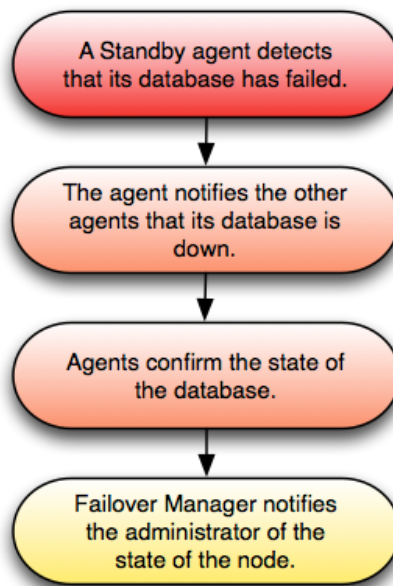


図 4: Confirming the failure of a standby database.

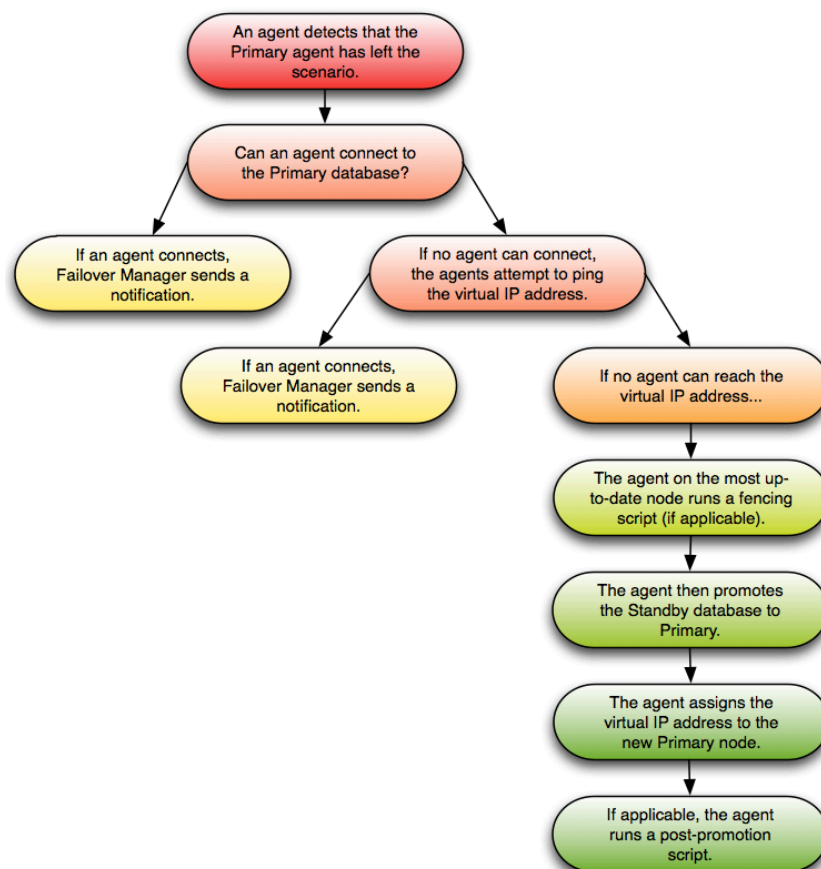


図 5: Confirming the failure of the primary agent

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。必要に応じて、エージェントはプロモーション後の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、`recovery.conf` ファイルを作成します。フェールオーバーマネージャーは、クラスターの残りのノードでこれらと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、次のことを行う必要があります。

クラスター全体を再起動せずにこのシナリオから復旧するには

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に障害が発生した場合、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。この場合を回避するには、`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の終了時にフェー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。

5.7.5 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

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

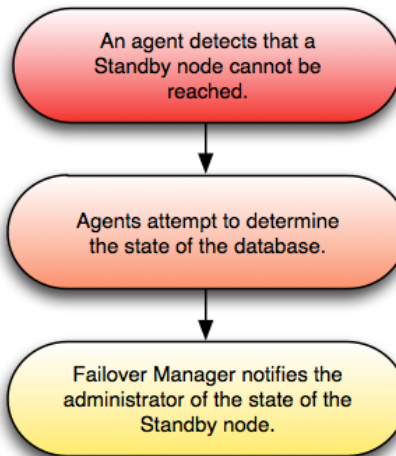


図 6: Failure of standby agent

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

1つのプライマリと1つのスタンバイのみが残っている場合、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。

5.7.6 専用の監視エージェントが終了/ノードに障害が発生

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。

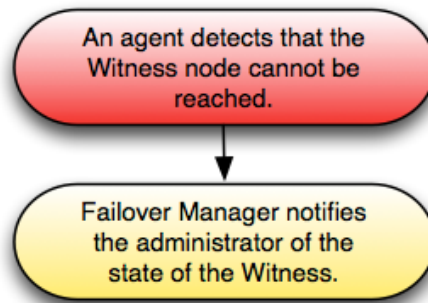


図 7: Confirming the failure of a dedicated witness

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。

i 注釈

監視に障害が発生し、クラスターに2つのノードのみがある場合、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。

5.7.7 ノードがクラスターから分離される

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分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。その場合、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ますが、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します。`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。

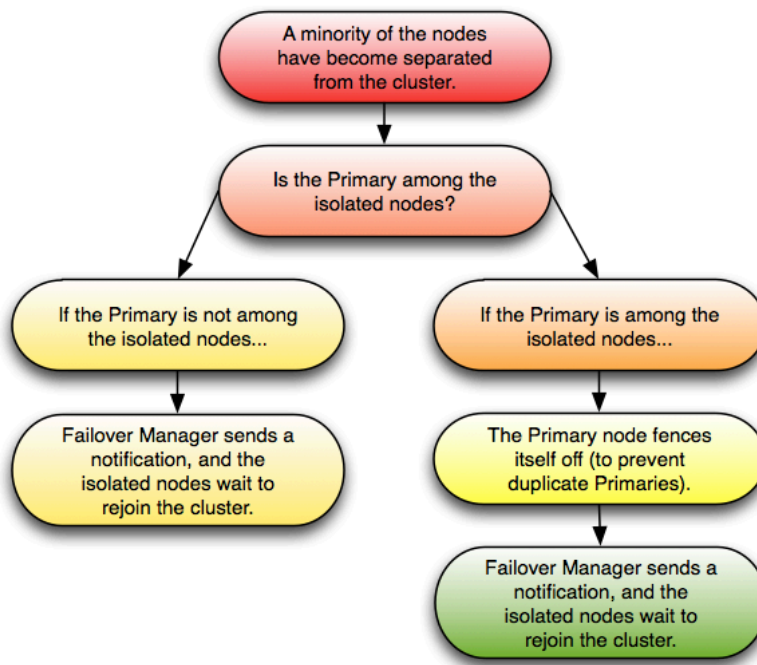


図 8: If members of the cluster become isolated

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。

5.8 Troubleshooting

5.8.1 Failover Manager エージェントの起動に失敗する

バージョン 5.x エージェントの起動に失敗した場合は、指示に従って `systemctl/journalctl` 出力を参照してください。エージェントがバージョン 4.x である場合、詳細については、起動ログ `/var/log/efm-<version>/startup-<cluster>.log` を参照してください。

5.8.2 承認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？

Failover Manager クラスター管理コマンドを呼び出し、Failover Manager 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?

5.8.3 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>`' は '`efm`' グループのメンバーではありません。

Using the efm utility に文書化されている一部の `efm` コマンドを呼び出すには、特別な権限が必要です。これらのコマンドを実行する権限を持たないユーザーが呼び出した場合、`efm` コマンドはエラーを表示します。

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

5.8.4 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ます。フェールオーバーマネージャーは、このプロパティで設定されたデフォルトのメモリ値で実行されます。

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

jvm.options=-Xmx128m
```

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。

5.8.5 OpenJDK バージョンの確認

Failover Manager は OpenJDK でテストされています。OpenJDK を使用することを強くお勧めします。次のコマンドを使用して、Java インストールのタイプを確認できます。

フェールオーバーマネージャーエージェントが使用している Java のバージョンを更新または変更する前に、エージェントを停止し、更新後に再度起動する必要があります。これは OS のアップグレードにも適用され、エージェントが使用している Java インストールが変更される可能性があります。*Configuring for Eager Failover* が使用されていない限り、エージェントを停止しても実行中のデータベースサーバーに影響はありません。

```
# java -version

openjdk version "11.0.20" 2023-07-18 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.20.0.8-1.el7_9) (build 11.0.20+8-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.20.0.8-1.el7_9) (build 11.0.20+8-LTS, mixed_
→mode, sharing)
```

注釈

RHEL およびその派生の OpenJDK バージョン 11 に一時的な問題があります。フェールオーバーマネージャーを起動すると、次のようなエラーが表示される場合があります。

```
java.lang.Error: java.io.FileNotFoundException: /usr/lib/jvm/java-11-openjdk-11.0.
20.0.8-2.el8.x86_64/lib/tzdb.dat (No such file or directory)
```

このメッセージが表示された場合、回避策は、`sudo dnf install tzdata-java` コマンドを使用して、不足しているパッケージを手動でインストールすることです。

5.8.6 クラスター外部からの予期しない接続試行

外部プロセスが `bind.address` ポートでエージェントに接続しようとする、フェールオーバーマネージャーは接続試行のソースを含む警告をログに記録します。これらの警告は、フェールオーバーマネージャークラスターには影響しません。ただし、ソースアドレスを使用して、外部プロセスを停止または構成して、フェールオーバーマネージャーエージェントに接続しようとしないようにできます。以下は、クラスターの外部の何かが `<source_address>` からエージェントプロセスに接続しようとしたときに表示されるメッセージの例です。

```
org.jgroups.protocols.TCP warn WARN: JGRP000006: failed accepting connection from_
→peer Socket[addr=/<source_address>,port=56046,localport=7800]: java.net.
→SocketTimeoutException: Read timed out
```

別のクラスターの一部であったアドレスでエージェントを実行している場合、元のクラスターはまだこのアドレスに接続してクラスターを再形成しようとしている可能性があります。この例では、クラスター `oldcluster` はまだ `newcluster` の一部になったアドレスに接続しようとしています。

```
org.jgroups.protocols.TCP warn WARN: JGRP000012: discarded message from different_
→cluster oldcluster (our cluster is newcluster). Sender was 93cb99c7-bf3f-4243-b582-
→faf25aced49e(<source_address>)
```

クラスター名前と `<source_address>` 情報を使用して、元のクラスターを見つけることができます。そのクラスターで `efm remote-members` コマンドを使用すると、キャッシュからアドレスがクリアされます。

第6章 Tutorial

6.1 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 プライマリノードを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、フェールオーバーマネージャーのドキュメントの残りの部分を読んで理解します。

例として EDB Postgres Advanced Server を使用して、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、次の手順に従って基本的なインストールと構成を行います。

- 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ーをインストールして初期化します。インストールについては、[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プライマリノードとスタンバイノードの間で構成および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フェールオーバーマネージャー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-efm50
```

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-5.2
cp efm.properties.in efm.properties
```

(次のページに続く)

```
cp efm.nodes.in efm.nodes

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-5.2/bin/efm encrypt efm
```

画面の指示に従って、データベースパスワードの暗号化バージョンを作成します。

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記載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する場合、Failover Manager 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提供します。

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を再起動するために使用されます。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> への呼び出しに使用されます。
<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>	ノードのローカルアドレスとフェールオーバーマネージャーに使用するポート。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>	監視ノードでは <code>true</code> 、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>	インターネットアクセスのないネットワークで実行している場合は、 <code>ping.server.ip</code> をネットワークで使用可能なアドレスに設定します。

次のページに続く

表 1 – 前のページからの続き

Property	Description
<i>auto.allow.hosts</i>	テストクラスターでは、起動を簡素化するために`true`に設定します。運用環境での使用については、フェールオーバーマネージャーユーザーガイドを参照してください。
<i>stable.nodes.file</i>	テストクラスターでは、起動を簡素化するために`true`に設定します。運用環境での使用については、フェールオーバーマネージャー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込まれ、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つのノードがメンバーシップコーディネーターとして機能します。リストには、少なくともメンバーシップコーディネーターのアドレスを含めます。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起動されていません。

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上の `/etc/edb/efm-5.2` 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き、すべてのノードで同じにすることができます。
 - ノードのローカルアドレスを使用するように **bind.address** プロパティを変更します。
 - ノードが監視ノードの場合は、**is.witness** を **true** に設定します。ノードが監視ノードの場合、ローカルデータベース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-5.2** 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-5.2
```

1. エージェントが起動したら、次のコマンドを実行して、シングルノードクラスターのステータスを確認します。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-5.2/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。

起動に失敗したエージェントがある場合は、指示に従って `systemctl/journalctl` を使用して、何が問題だったかに関する情報を取得します。フェールオーバーマネージャーバージョン 4.x を使用している場合、代わりに起動ログを参照してください。

```
cat /var/log/efm-4.<x>/startup-efm.log
```

6.1.1 スイッチオーバーを実行する

クラスターステータス出力が、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を示している場合、スイッチオーバーを実行できます。

```
/usr/edb/efm-5.2/bin/efm promote efm -switchover
```

このコマンドは、スタンバイを昇格させ、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。

6.1.2 オンラインヘルプにアクセスします

オンラインヘルプにすばやくアクセスするには、次を使用します。

```
/usr/edb/efm-5.2/bin/efm --help
```