



EDB

Failover Manager

リリース 4.4.0

EnterpriseDB

2022 年 01 月 24 日

目次

第 1 章	Release Notes	3
1.1	Version 4.4	3
1.2	Version 4.3	3
1.3	Version 4.2	4
1.4	Version 4.1	4
1.5	Version 4.0	5
1.5.1	EFM プロパティの変更	6
第 2 章	Choosing a deployment architecture	7
2.1	Failover Manager with virtual IP	8
2.1.1	VIP でのフェールオーバーマネージャーの使用	8
2.2	Failover Manager with client connect failover	10
2.2.1	クライアント接続フェイルオーバーでのフェイルオーバーマネージャーの使用	10
2.3	Failover Manager with PgBouncer	10
2.3.1	オンプレミスの PgBouncer を使用したフェールオーバーマネージャー	12
2.3.2	クラウド内の PgBouncer を使用したフェールオーバーマネージャー	12
2.3.3	PgBouncer でフェールオーバーマネージャーを使用する	13
2.4	Failover Manager with Pgpool	14
2.4.1	オンプレミスの Pgpool を使用したフェールオーバーマネージャー	14
2.4.2	クラウド内の Pgpool を使用したフェールオーバーマネージャー	15
2.4.3	Pgpool でのフェールオーバーマネージャーの使用	15
第 3 章	Prerequisites	17
3.1	Java 1.8（またはそれ以降）をインストールします	17
3.2	SMTP サーバーを提供する	17
3.3	ストリーミングレプリケーションを構成する	17
3.4	pg_hba.conf の変更	18
3.5	データベースサーバーの自動起動の使用	18
3.6	ファイアウォールを介した通信を確保する	19
3.7	データベースユーザに十分な権限があることを確認します	19
第 4 章	Installing Failover Manager	21
4.1	RedHat または CentOS ホスト	21
4.1.1	RHEL または CentOS 7 PPCLE ホスト	21
4.1.2	RHEL または CentOS 7 ホスト	22
4.1.3	RHEL または CentOS 8 ホスト	23
4.2	Debian または Ubuntu ホスト	23
4.2.1	Debian ホスト	23
4.2.2	Ubuntu ホスト	24
4.3	SLES ホスト	24

4.4	インストール後のタスクの実行	25
4.4.1	インストール場所	25
第 5 章	Upgrading an existing cluster	27
5.1	フェイルオーバーマネージャーのアンインストール	28
5.2	データベースの更新の実行（マイナーバージョン）	29
第 6 章	Configuring streaming replication	31
6.1	カスケーディングレプリケーションの限定サポート	31
第 7 章	Configuring Failover Manager	33
7.1	The cluster properties file	33
7.1.1	クラスタープロパティの指定	34
7.1.2	クラスターのプロパティ	41
7.2	Encrypting your database password	67
7.2.1	EFMPASS 環境変数を使用する	68
7.3	The cluster members file	68
7.4	Extending Failover Manager permissions	69
7.4.1	sudo なしでフェールオーバーマネージャーを実行する	71
7.5	Using Failover Manager with virtual IP addresses	72
7.5.1	VIP のテスト	73
7.6	Configuring for Eager Failover	75
7.6.1	Eager Failover を有効にする	75
7.6.2	Eager Failover の無効化	76
7.6.3	Eager Failover モードでの Failover Manager のアップグレード	76
7.6.4	重要な注意事項	76
第 8 章	Configuring SSL authentication on a Failover Manager cluster	79
第 9 章	Using Failover Manager	81
9.1	フェールオーバーマネージャークラスターの管理	81
9.1.1	Failover Manager クラスターの起動	82
9.1.2	クラスターにノードを追加する	82
9.1.3	スタンバイの優先度を変更する	83
9.1.4	フェールオーバーマネージャークラスターの昇格	84
9.1.5	Failover Manager エージェントの停止	85
9.1.6	Failover Manager クラスターの停止	86
9.1.7	クラスターからノードを削除する	86
9.2	単一ノードでマルチプルのエージェントを実行する	86
9.2.1	RHEL / CentOS 7.x または RHEL / CentOS 8.x	88
第 10 章	Using the efm utility	89
10.1	efm allow-node	89
10.2	efm disallow-node	89
10.3	efm cluster-status	89
10.4	efm cluster-status-json	90
10.5	efm 暗号化	91
10.6	efm プロモート	91
10.7	efm 履歴書	92

10.8	efm set-priority	92
10.9	efm stop-cluster	92
10.10	efm upgrade-conf	93
10.11	efm node-status-json	93
10.12	efm-ヘルプ	93
第 11 章	Monitoring a Failover Manager cluster	95
11.1	クラスターステータスレポートの確認	95
11.1.1	終了コード	97
11.2	Postgres Enterprise Manager を使用したストリーミングレプリケーションの監視	97
第 12 章	Controlling the Failover Manager service	101
12.1	RHEL / CentOS 7.x および RHEL / CentOS 8.x で systemctl ユーティリティを使用する	101
第 13 章	Controlling logging	103
13.1	syslog ログファイルエントリの有効化	104
第 14 章	Notifications	105
第 15 章	Supported failover and failure scenarios	121
15.1	プライマリデータベースがダウンしています	121
15.2	ノードをクラスターに戻す	123
15.3	ノードをプライマリのロールに戻す	123
15.4	スタンバイデータベースがダウンしています	123
15.5	プライマリエージェントが終了するか、ノードに障害が発生する	125
15.6	スタンバイエージェントが終了するか、ノードに障害が発生する	126
15.7	専用の監視エージェントが終了/ノードが失敗する	126
15.8	ノードはクラスターから分離されます	128
第 16 章	Troubleshooting	129
16.1	認証ファイルが見つかりません。ローカルエージェントは実行されていますか？	129
16.2	このコマンドを実行する権限がありません。ユーザー ‘& lt; os user & gt;’ efm\グループのメンバではありません。	129
16.3	通知; 予期しないエラーメッセージ	129
16.4	OpenJDK バージョンの確認	130
第 17 章	Creating a Failover Manager cluster	131
17.1	スイッチオーバーを実行する	133
17.2	オンラインヘルプにアクセスする	134

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

従来、クラスターはマルチプルのデータベースを管理する Postgres の単一インスタンスです。この文書では、クラスターという用語はフェールオーバーマネージャークラスターを指します。フェールオーバーマネージャークラスターは、クラウド内のサーバーまたは従来のネットワーク上に存在し、JGroups ツールキットを使用して通信する以下のノード（またはエージェント）で構成されます。

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

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

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

フェイルオーバーが発生したイベント、フェイルオーバーオーバーマネージャは、昇格したスタンバイがクラスター内で最新のスタンバイであることを保証しようとします。スタンバイノードがプライマリノードと同期していない場合、データが失われる可能性があります。

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

この図は、仮想 IP アドレスを使用する Failover Manager クラスターを示しています。データベースを追加または削除するたびにロードを再構成するために独自のロードバランサーを提供する場合は、代わりにロードを使用できます。また、高可用性のためにネイティブ EFM-Pgpool 統合を有効にすることもできます。

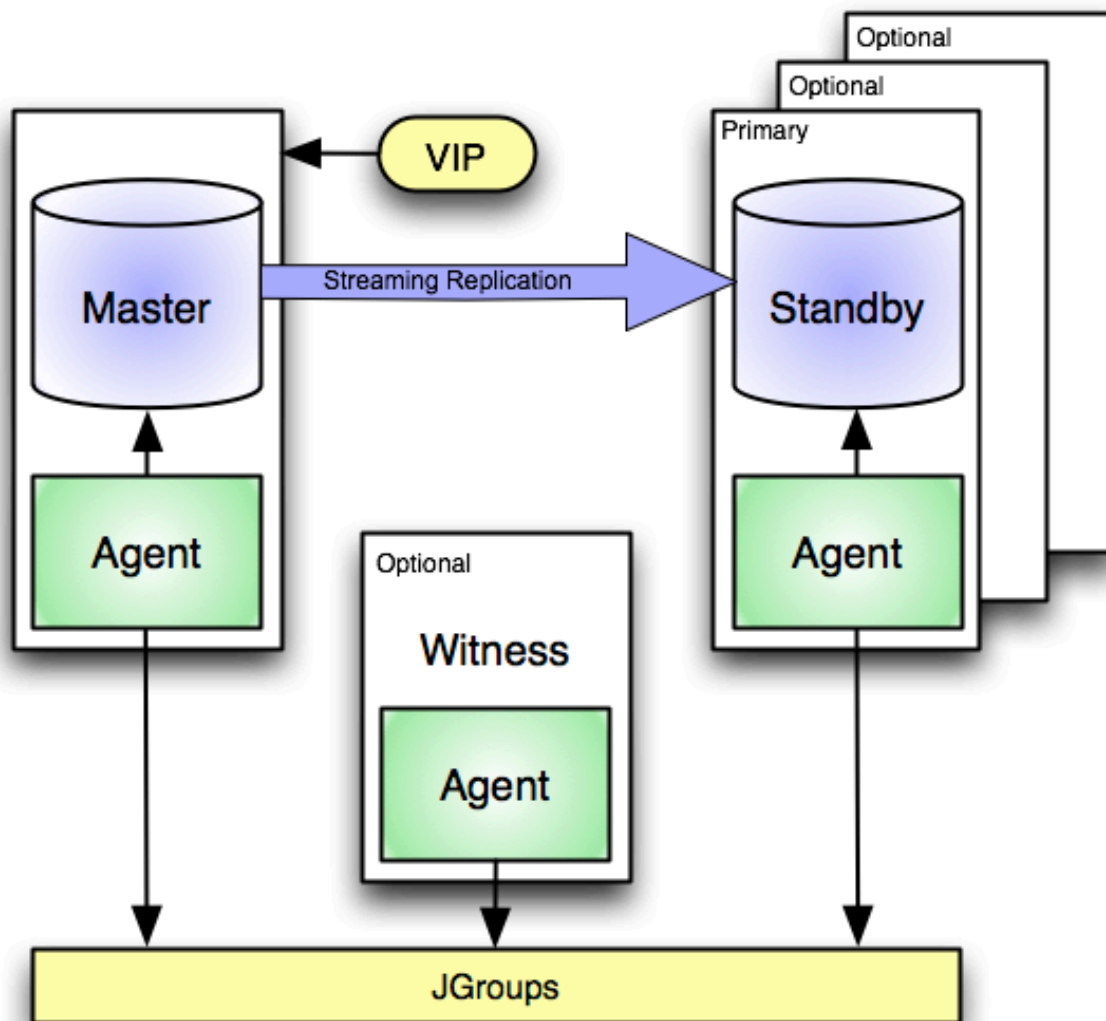


図 1: A Failover Manager scenario employing a virtual IP address

第1章 Release Notes

Version	Release Date
[4.4](06_efm_44_rel_notes)	2022 Jan 5
[4.3](07_efm_43_rel_notes)	2021 Dec 18
[4.2](08_efm_42_rel_notes)	2021 Apr 19
[4.1](09_efm_41_rel_notes)	2021 Dec 11
[4.0](10_efm_40_rel_notes)	2021 Sep 2

1.1 Version 4.4

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

Type	Description
Security Fix	Failover Manager は、log4j バージョン 2.17.1 を使用してセキュリティの脆弱性 CVE-2021-44832 に対処するようにアップグレードされました。

1.2 Version 4.3

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

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

1.3 Version 4.2

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

Type	Description
Enhancement	プライマリリソースを監視されていない正常でないノードから完全に動作可能で正常なノードに移動するために、新しい Eager Failover セットアップが追加されました。この設定では、プライマリ EFM エージェントが停止したときにフェイルオーバーが実行されます。Eager Failover がない場合、プライマリ Postgres インスタンスは使用可能なままになりますが、管理されません。
Enhancement	EDB Postgres の高可用性および水平読み取りスケーリングアーキテクチャは、PgPool-II 4.2 で認定されており、非 sudo モードでサポートされています。
Enhancement	スタンバイのプロモーション優先度をプライマリごとに定義できるようになりました。[サポートチケット # 1092697]
Enhancement	stop-cluster コマンドを EFM 構成で無効にできるようになりました。
Bug Fix	アーカイビング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るようになりました。[サポートチケット # 1127627、# 1182654]
Bug Fix	新しいオプションでは、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に、LoadBalancer デタッチスクリプトを無効にできます。【サポートチケット # 1096007】
Bug Fix	エージェントのスタートアップに、pg_wal / pg_xlog の親ディレクトリが書き込み可能かどうかを確認するバリデーションを追加しました。EFM がバックアップを作成し、後で復元できるように、親ディレクトリは書き込み可能である必要があります。新しいバリデーションが失敗すると、エラーがログに記録され、エージェントはスタートを拒否します。
	NA

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。

1.4 Version 4.1

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

Type	Description
Enhancement	EFM を使用した HA アーキテクチャと AzureNetwork LoadBalancer を使用した Pgpool 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の接続が回復すると、ヘルスモニタリングを再開しようとします。

次のページに続く

表 5 – 前のページからの続き

Type	Description
Enhancement	新しいオプションは、同期的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` を自動的に増やします。
Bug Fix	同期的スタンバイが追加されたときに `num_sync` を増やすことができます。【サポートチケット # 1092053】
Bug Fix	特定のプロパティが存在しない状態でエージェントをスタートできません。[サポートチケット # 1087362]
Bug Fix	セキュリティの脆弱性のためにアップグレードされた JDBC ドライバ。[サポートチケット # 1052383]
Bug Fix	非 sudo モードで VIP を取得した後に「arping」が呼び出されるようになりました。[サポートチケット # 1039625]

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

1.5 Version 4.0

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

Type	Description
Enhancement	プロパティの名前が変更されました。詳細については、次の表を参照してください。アップグレードユーティリティを使用して Failover Manager インストールをアップグレードすると、それらは自動的に更新されます。
Enhancement	暗号化されたデータベースパスワードのプロパティ値は、新しいハッシュアルゴリズムを満たすように更新するニーズがあります。アップグレードツールは make 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は、代わりにパスワードを暗号化するための標準的なプロシージャを使用できます。
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でも暗号化が有効になりました。
Enhancement	新しいプライマリを選択している間、スタンバイサーバーが停止しなくなりました。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します。
Enhancement	コミュニティの命名ガイドラインとの一貫性を保つため、Failover Manager 製品および文書では、 *マスタ* という用語が *プライマリ* に置き換えられました。 upgrade-conf ツールは、アップグレード後に影響を受けるプロパティの名前を変更するタスクを処理します。 script.load.balancer.attach、script.load.balancer.detach などのロードスクリプトは、引数として `m` ではなく文字 `p` を受け入れるようになりました。
Enhancement	プロモーション後のスタンバイのリスタートを遅らせるサポートが追加されました。スタンバイのリスタートをずらすことで、可用性を高めることができます。

次のページに続く

表 6 – 前のページからの続き

Type	Description
Bug Fix	<code>/var/run</code> および <code>/var/lock</code> に tmpfs マウントポイントを使用するユーザーを支援するためにドキュメントが更新されました [サポートチケット # 977437]
Bug Fix	IDLE エージェントは、他のエージェントがデータベースに到達できるときにモニタリングを再開しようとします。【サポートチケット # 1001533】
	NA

1.5.1 EFM プロパティの変更

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

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

第2章 Choosing a deployment 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モジュールです。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プライマリノードに自動的にプロモートさせることができます。この機能により、継続的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ロスを防ぐことができます。

フェールオーバーマネージャークラスターは、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ます。

- プライマリノードは、サービスを提供しているプライマリデータベースサーバです。データベースクライアント。
- 1つ以上のスタンバイノードがストリーミングレプリケーションしているプライマリノードに関連しています。
- 監視ノードは、プライマリまたはアサーションのいずれかのアサーションを確認しますフェールオーバーシナリオでのスタンバイ。障害シチュエーション中に、プライマリは、半分以上のノードがあるパーティションにあります。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持つクラスター。

Failover Manager は、Postgres 接続プラーと接続ライブラリを使用して、EDBPostgres Advanced Server にさまざまな高可用性オプションを提供します。これらのオプションは、高可用性アーキテクチャに影響を与えます。

データベースの高可用性を保証に、Failover Manager のコア機能を Postgres 接続ライブラリ（クライアント接続フェールオーバー）および接続プラーと組み合わせることができます。

Failover Manager の機能により、EDB は、高可用性環境を実行するための4つの基本アーキテクチャを設計しました。

1. **Failover Manager using VIP (virtual IP)**：フェールオーバーマネージャーには、VIP アドレスをすぐに管理できるキー機能があります。VIP アドレスにより、アプリケーションは、プライマリデータベースサーバにルーティングされている単一の IP アドレスに接続できます。このアーキテクチャは、VIP アドレスが環境で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.: libpq や jdbc などの PostgreSQL クライアントライブラリでは、クライアント接続の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、接続文字列にマルチプルサーバ（ホスト= srv1、srv2）が含まれ、クライアントライブラリが使用可能なホストをループして、使用可能で読み書き操作が可能な接続を見つけます。この機能により、クライアントはスイッチオーバー中にマスタを追跡できます。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 PgBouncer**：PgBouncer は、コネクション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。また、クライアントと Postgres データベースサーバ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 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にルーティングできます。

4. **Failover Manager with Pgpool** : Pgpool-II は、クライアントと Postgres データベースサーバ間のプロキシとして使用される別のツール。Pgpool-II は、Watchdog を使用したクラスターモードでの実行、VIP の管理、読み取り専用のスケーラビリティなどの機能を追加します。フェールオーバーマネージャーには、Pgpool-II と統合してデータベースフェイルオーバー操作中に別のプライマリにトラフィックを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 PgBouncer	Failover Manager with Pgpool
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

2.1 Failover Manager with virtual IP

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

2.1.1 VIP でのフェールオーバーマネージャーの使用

インストール

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

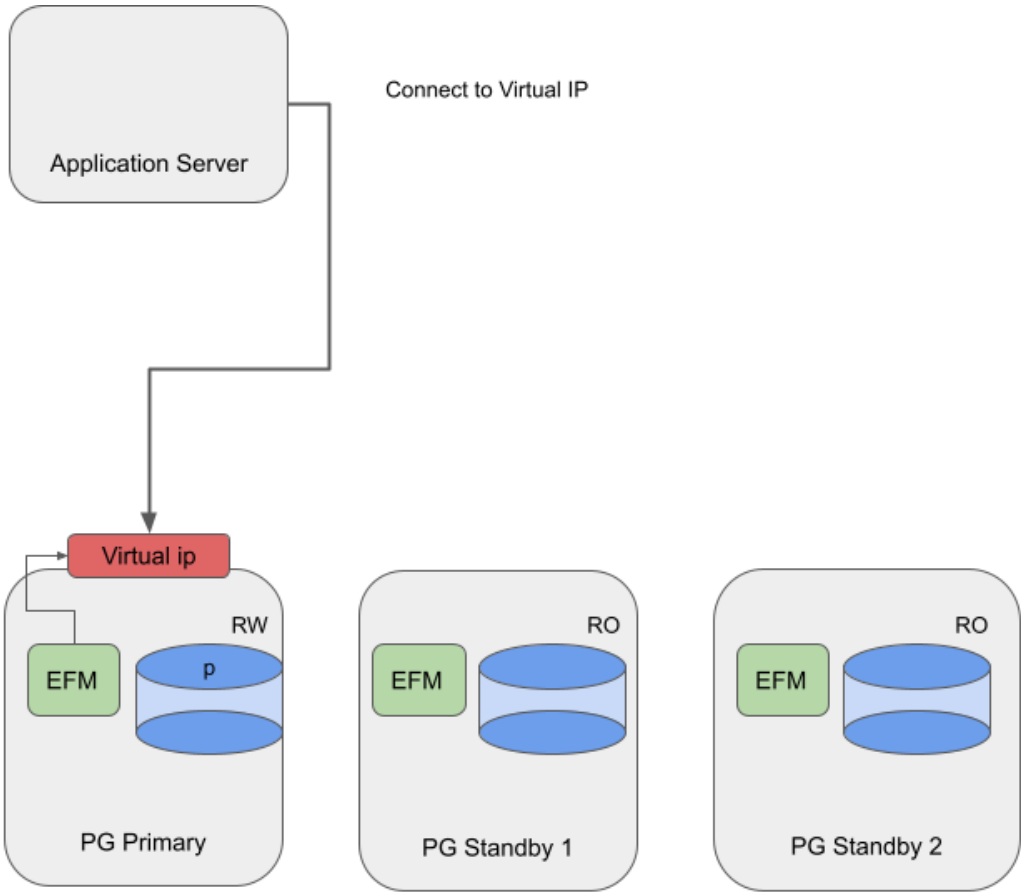


図 1: Failover Manager with VIP

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

VIP を指定する

クラスタプロパティファイルで、`virtual.ip` プロパティにホスト名または IP アドレスを指定します。`virtual.ip.prefix` プロパティで対応するプレフィックスを指定します。`virtual.ip.interface` プロパティを使用して、VIP が使用するネットワークインタフェースを提供します。デフォルトでは、`virtual.ip` と `virtual.ip.prefix` の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、クラスタのプライマリノードにのみ割り当てられます。プライマリを指定すると、フェイルオーバー中に新しい VIP 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 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との接続文字列をサポートしています。最初の接続試行時または接続が失われたとき（フェイルオーバー時にも発生します）、クライアントドライバは読み書き接続を見つけるまで提供されたホストに 1 つずつ接続します。`newmaster` への再接続の時間は接続によって異なりますドライバおよび TCP レイヤーで構成されたタイムアウト。

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

2.2.1 クライアント接続フェイルオーバーでのフェイルオーバーマネージャの使用

インストール

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

2.3 Failover Manager with PgBouncer

フェールオーバーマネージャと PgBouncer を使用して、オンプレミスのセットアップとクラウドのセットアップで高可用性を実現できます。PgBouncer は一般的な接続プーラーですが、マルチホスト構成、フェイルオーバー、または検出がないため、PostgreSQL の高可用性を自分自身で達成するには不十分です。

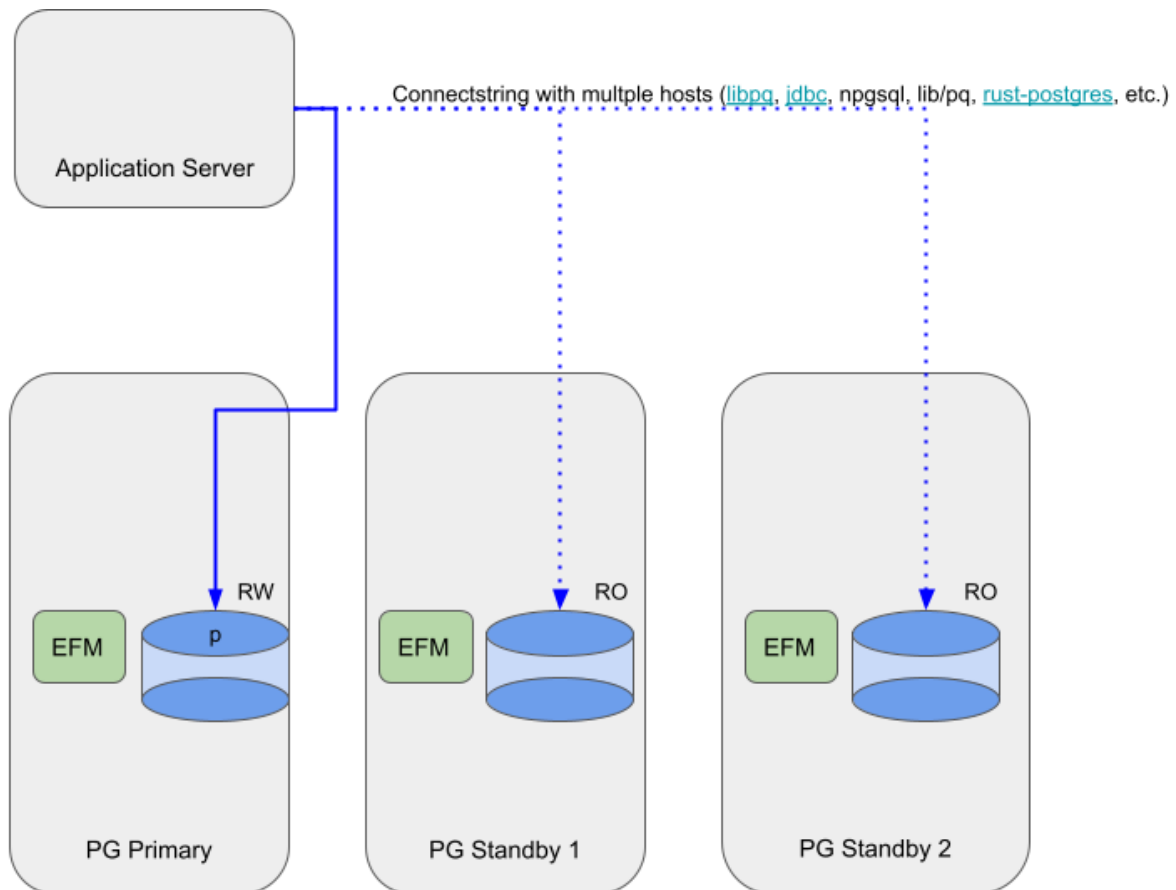


図 2: Failover Manager traffic routing diagram for client connect failover

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

オンプレミスのセットアップでは、接続ライブラリを使用して、マルチプルのホストで接続文字列を使用することで高可用性を提供します。

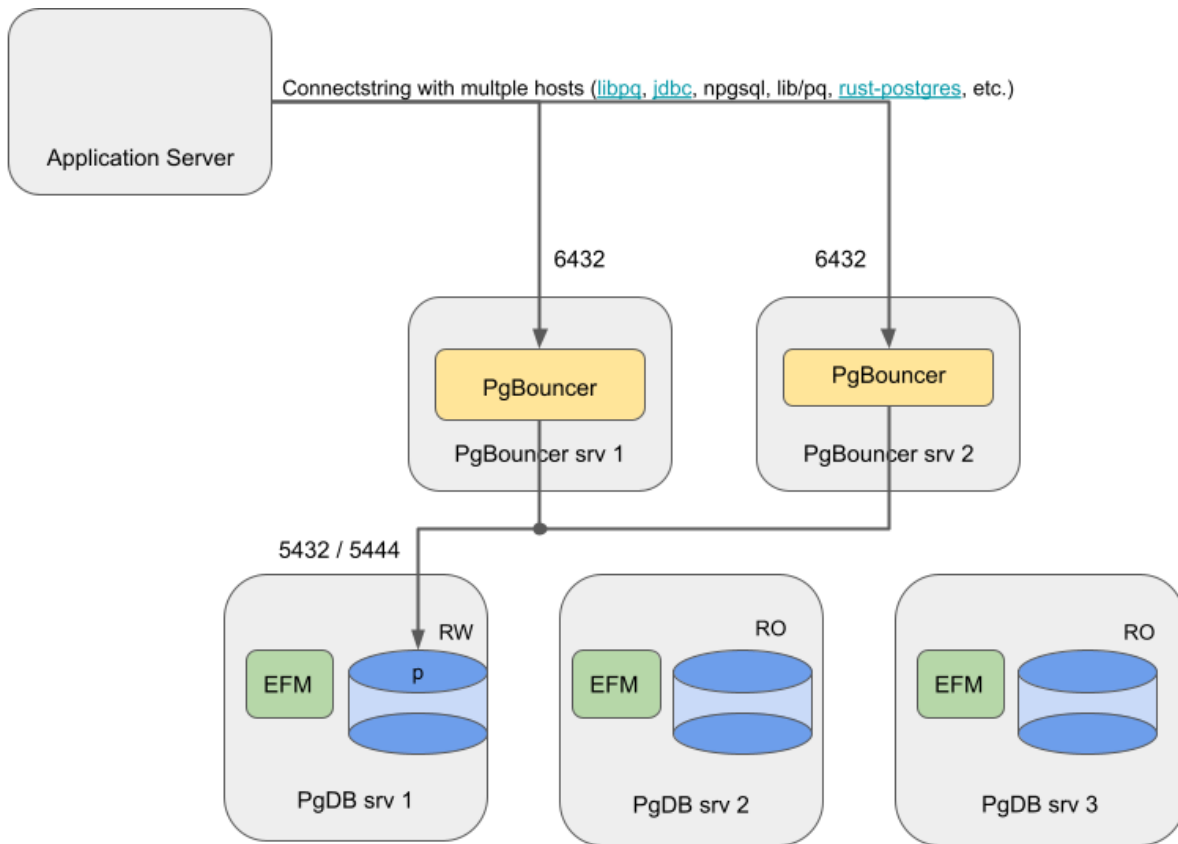


図 3: Failover Manager using pgBouncer on-premises architecture diagram

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

2.3.2 クラウド内の PgBouncer を使用したフェールオーバーマネージャー

クラウドのセットアップでは、ネットワークロード（NLB）を使用して、PgBouncer の両方のインスタンスでトラフィックのバランスを取ります。

図 4：クラウドで PgBouncer を使用したフェールオーバーマネージャーのトラフィックルーティング

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

- クラウドネットワークロード [Azure](#) の制約ソースと宛先が存在する場合、トラフィックを適切にルーティングしません同じマシン上で。
- 混在アーキテクチャでは、PgBouncer と Postgres 間のトラフィックは不均衡になります（時にはローカル、時にはネットワーク化）。

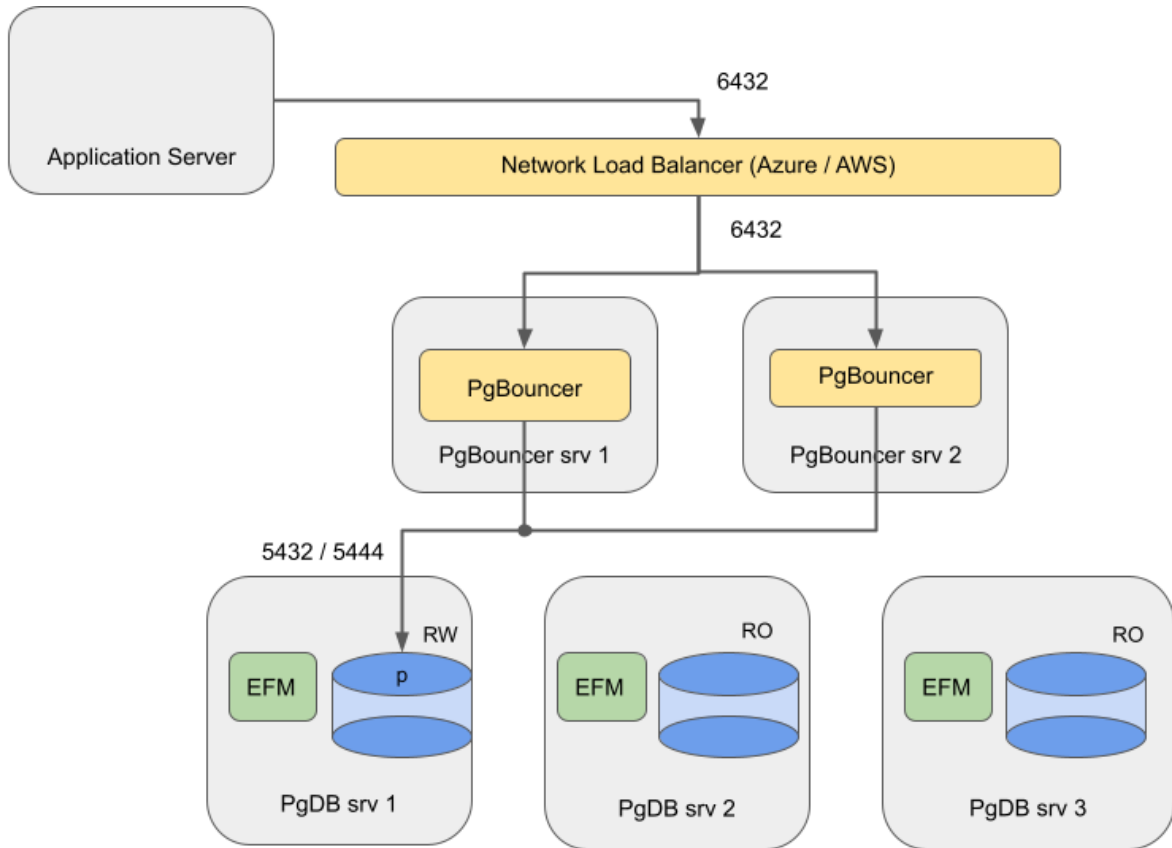


図 4: Failover Manager with PgBouncer cloud architecture diagram

- PgBouncer と PostgreSQL はリソースをめぐって競合します。
- マスタ障害はルーティング (PgBouncer) とデータベースの両方に影響しますこれらの 2 つのコンポーネントが同じマシン上で組み合わせられる場合。

2.3.3 PgBouncer でフェールオーバーマネージャーを使用する

インストール

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

構成が完了したら、ポート 6432 でネットワークロードの IP アドレスのデータベースに接続できます。プライマリデータベースサーバで障害が発生した場合、フェールオーバーマネージャーは新しいプライマリを昇格し、PgBouncer を再構成してトラフィックを再分配します。PgBouncer プロセスのいずれかがトラフィックを受け入れることができない場合、ネットワークロードバランサーはすべてのトラフィックを残りの PgBouncer プロセスに再配布します。`max_client_conn` パラメータが、フェールオーバーの場合の接続数の増加を補うように調整されていることを確認してください。

2.4 Failover Manager with Pgpool

Pgpool-II は、読み取り専用のトラフィック負荷分散、コネクションプーリング、クラスター化されたプロキシとしての実行など、多くの機能を提供できる一般的なコネクションプーラです。Failover Manager を使用して可用性を管理し、Pgpool-II を正しいプライマリにプロキシするように再構成すると、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のセットアップで高可用性を実現できます。

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

オンプレミスのセットアップでは、VIP を使用して、利用可能な Pgpool-II インスタンスにトラフィックをルーティングできます。このセットアップでは、Pgpool-II の自動フェイルオーバーが無効になり、Failover Manager が Pgpool-II を管理するように構成されます。

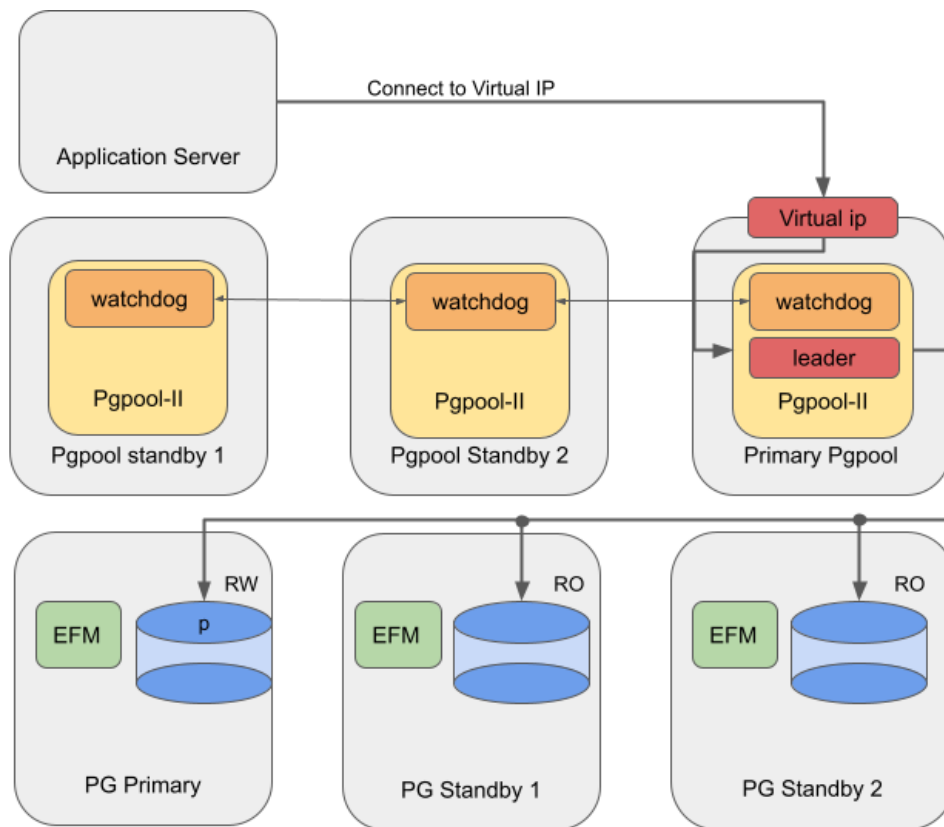


図 5: Failover Manager with Pgpool on-premises

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

2.4.2 クラウド内の Pgpool 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のある環境（クラウド環境など）では、ネットワークロード（NLB）を使用して、VIP を必要とせずに利用可能なすべての Pgpool-II インスタンスでトラフィックのバランスを取ることができます。

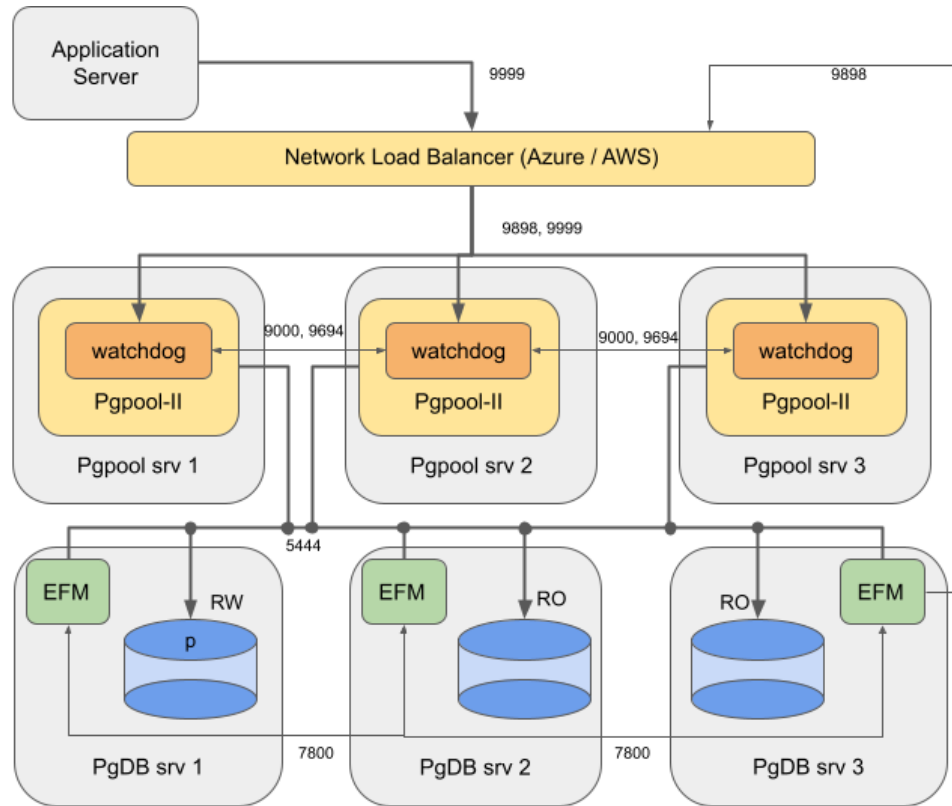


図 6: Failover Manager with Pgpool in cloud

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

2.4.3 Pgpool 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

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

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で障害が発生した場合、フェールオーバーマネージャーは新しいプライマリを昇格し、トラフィックを再配布するように Pgpool を再構成します。いずれかの Pgpool プロセスがトラフィックを受け入れることができない場合、ネットワークロードは残りの 2 つの Pgpool プロセスにすべてのトラフィックを再分配します。フェイルオーバーの場合に、より多くの接続を補償するために、`listen_backlog_multiplier` が調整されていることを確認してください。

第3章 Prerequisites

</div>

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

3.1 Java 1.8（またはそれ以降）をインストールします

Failover Manager を使用する前に、最初に Java（バージョン 1.8 以降）をインストールする必要があります。フェールオーバーマネージャは 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。

3.2 SMTP サーバーを提供する

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

- メール通知を使用している場合、フェールオーバーマネージャシナリオの各ノードで SMTP サーバーが実行されている必要があります。
- `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` パラメータで指定された任意のメールアドレスに通知メールを送信することもできます。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。

3.3 ストリーミングレプリケーションを構成する

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

データベースバージョン 11（またはそれ以前）では、`-sourcenode` オプションで指定しない限り、スイッチオーバー中に `recovery.conf` ファイルがランダムスタンバイノードから停止したプライマリにコピーされます。スイッチオーバーを実行する前に、スタンバイノードの `recovery.conf` ファイルの経路が一貫して

いることを確認してください。-sourcenode オプションの詳細については、[Promoting a Failover Manager node](#) を参照してください。

データベースバージョン 12 以降では、-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 primary_conninfo および restore_command プロパティがランダムスタンバイノードから停止したプライマリにコピーされます。

3.4 pg_hba.conf の変更

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

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

どこで：

efm は、有効なデータベースユーザの名前を指定します。

fmdb は、efm ユーザが接続できるデータベースの名前を指定します。

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

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

x は Postgres バージョンを指定します。

3.5 データベースサーバーの自動起動の使用

プライマリノードが再起動すると、フェールオーバーマネージャーは、プライマリノードでデータベースがダウンしていることを検出し、スタンバイノードをプライマリのロールにプロモート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ず、recovery.conf ファイルはデータベースサーバの起動を妨げます。これが発生した場合、再起動されたプライマリノードは、2 番目のプライマリノードとしてクラスタに戻ります。

この状態を防ぐには、データベースサーバの前に Failover Manager エージェントが自動的に起動保証にします。エージェントはアイドルモードで起動します。クラスター内に既にプライマリがあるかどうかを確認してください。プライマリノードがある場合、エージェントは recovery.conf または standby.signal ファイルが存在することを確認し、データベースは 2 番目のプライマリとしてスタートしません。

3.6 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で指定されたポートに対応するポート経由で接続します。

3.7 データベースユーザに十分な権限があることを確認します

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

pg_current_wal_lsn()

pg_last_wal_replay_lsn()

pg_wal_replay_resume()

pg_wal_replay_pause()

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

- データベースバージョン 9.6 の場合、db. ユーザはスーパーユーザでなければなりません。
- バージョン 9.6 以降のデータベースバージョンの場合、db. ユーザは、pg_reload_conf() を実行するために pg_read_all_stats 権限と許可が必要です。

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

ユーザには、構成変数の値を読み取るためのアクセス許可も必要です。データベーススーパーユーザは、PostgreSQL GRANT コマンドを使用して、必要な権限を提供できます。

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

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

第4章 Installing Failover Manager

</ div>

</ div>

Failover Manager がサポートするプラットフォームとバージョンについては、[Platform Compatibility](#) を参照してください。

!!! Note * CPU アーキテクチャの混在モードは、プライマリノードとスタンバイノードが IBM Power 上の Linux 上にあり、監視ノードが x86-64 Linux 上にある場合にサポートされます。

EnterpriseDB リポジトリへのアクセスを許可する資格情報を要求するには、[EDB Repository Access instructions](#) を参照してください。

4.1 RedHat または CentOS ホスト

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

インストール中に、yum は解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決するために必要な依存関係のリストを提供します。

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

資格情報を受け取ったら、クラスターの各ノードで EnterpriseDB リポジトリ構成ファイルを作成し、ファイルを変更してアクセスを有効にする必要があります。

4.1.1 RHEL または CentOS 7 PPCLE ホスト

1. 構成ファイルを作成し、Advance Toolchain をインストールします。

```
* rpm --import https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/
↪redhat/RHEL7/gpg-pubkey-6976a827-5164221b
* cat > /etc/yum.repos.d/advance-toolchain.repo <<EOF
* # Begin of configuration file
* [advance-toolchain]
```

(次のページに続く)

(前のページからの続き)

```
* name=Advance Toolchain IBM FTP
* baseurl=https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/
↳redhat/RHEL7
* failovermethod=priority
* enabled=1
* gpgcheck=1
* gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/
↳RHELX/gpg-pubkey-6976a827-5164221b
* # End of configuration file
* EOF
```

2. EDB リポジトリをインストールします。

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

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

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

4. EPEL リポジトリをインストールします。

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

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

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

6. 選択したパッケージをインストールします。

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

4.1.2 RHEL または CentOS 7 ホスト

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

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

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

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

3. EPEL リポジトリをインストールします。

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

4. RHEL 7 で、追加のリポジトリを有効にして依存関係を解決します。

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

5. 選択したパッケージをインストールします。

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

4.1.3 RHEL または CentOS 8 ホスト

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

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

3. EPEL リポジトリをインストールします。

4. 追加のリポジトリを有効にして、依存関係を解決します。

```
* ARCH=$( /bin/arch ) subscription-manager repos --enable "codeready-builder-for-
→rhel-8-${ARCH}-rpms"
```

5. ビルトインの PostgreSQL モジュールを無効にします。

4.2 Debian または Ubuntu ホスト

Failover Manager をインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格情報を要求するには、[EnterpriseDB website](#) を参照してください。

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

4.2.1 Debian ホスト

```
* * sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)-
→edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
* * sh -c 'echo "machine apt.enterprisedb.com login <username> password <password>
→" > /etc/apt/auth.conf.d/edb.conf'
```

4.2.2 Ubuntu ホスト

```
* * sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)-
→edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
* * sh -c 'echo "machine apt.enterprisedb.com login <username> password <password>
→" > /etc/apt/auth.conf.d/edb.conf'
```

4.3 SLES ホスト

Failover Manager をインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格情報を要求するには、[access EDB software repositories](#) の手順を参照してください。

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

1. フェールオーバーマネージャをインストールする前に、スーパーユーザ特権を想定し、ファイアウォールを停止します。次に、以下のコマンドを使用して、EnterpriseDB リポジトリをシステムに追加します。

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

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

```
* zypper refresh
```

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

3. SUSEConnect と SUSE Package Hub 拡張を SLES ホストに追加し、ホストを SUSE に登録して、SUSE リポジトリへのアクセスを許可します。

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

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

```
* zypper addrepo https://download.opensuse.org/repositories/Apache:/Modules/SLE_12_
→SP4/Apache:Modules.repo
```

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

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

6. zypper ユーティリティを使用して、Failover Manager エージェントをインストールします。

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

4.4 インストール後のタスクの実行

Failover Manager を使用して、enterprisedb または postgres 以外のユーザが所有するクラスタをモニタしている場合は、[Extending Failover Manager permissions](#) を参照してください。

クラスターの各ノードにインストールした後：

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

4.4.1 インストール場所

コンポーネントは次の場所にインストールされます。

Component	Location
Executables	<i>/usr/edb/efm-<4.4>/bin</i>
Libraries	<i>/usr/edb/efm-<4.4>/lib</i>
Cluster configuration files	<i>/etc/edb/efm-<4.4></i>
Logs	<i>/var/log/efm-<4.4></i>
Lock files	<i>/var/lock/efm-<4.4></i>
Log rotation file	<i>/etc/logrotate.d/efm-<4.4></i>
sudo configuration file	<i>/etc/sudoers.d/efm-<4.4></i>
Binary to access VIP without sudo	<i>/usr/edb/efm-<4.4>/bin/secure</i>

第5章 Upgrading an existing cluster

</div>

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

1. クラスターの各ノードにフェールオーバーマネージャー 4.4 をインストールします。Failover Manager のインストールの詳細については、[Installing Failover Manager](#) を参照してください。
2. Failover Manager をインストールした後、`efm upgrade-conf` ユーティリティを呼び出して、Failover Manager 4.4 用の `.properties` および `.nodes` ファイルを作成します。Failover Manager インストーラは、アップグレードユーティリティ () を `/usr/edb/efm-4.4/bin` ディレクトリにインストールします。ユーティリティを呼び出すには、ルート権限を引き受けて、コマンドを呼び出します。

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

`efm upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、Failover Manager で使用するためにパラメータ値を新しい構成ファイルにコピーします。ユーティリティは、構成ファイルの更新されたコピーを `/etc/edb/efm-4.4` ディレクトリに保存します。

3. 新しい設定を指定して、Failover Manager 4.4 の `.properties` および `.nodes` ファイルを変更します。エディタの選択を使用して、そのノードのサービスを開始する前に、プロパティ・ファイル (`/etc/edb/efm-4.4` ディレクトリにある) の追加プロパティを変更します。プロパティ設定の詳細については、[The cluster properties file](#) を参照してください。

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

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

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

1. クラスターの各ノードで新しい () を開始します。

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

```
[root@k8s-worker ~]# /usr/edb/efm-4.4/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-4.1
```

(次のページに続く)

```
Processing efm.properties file
```

```
The following properties were added in addition to those in the previous installed_
↳version:
```

```
priority.standbys
```

```
detach.on.agent.failure
```

```
Checking directory /etc/edb/efm-4.1
```

```
Processing efm.nodes file
```

```
Upgrade of files is finished. The owner and group for properties and nodes files have_
↳been set as 'efm'.
```

```
[root@k8s-worker ~]#
```

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

!!! Note * ユニットファイルを使用している場合は、アップグレードを実行するときに、新しい Failover Manager サービス名前を反映するようにファイルを手動で更新します。

5.1 フェイルオーバーマネージャーのアンインストール

Failover Manager 4.4 にアップグレード処理した後、ネイティブパッケージマネージャを使用して、Failover Manager の以前のインストールを削除できます。例、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 4.1 および不要な依存関係を削除します。

- RHEL または CentOS 7.x の場合：

```
yum remove edb-efm41
```

- RHEL または CentOS 8.x の場合：

```
dnf remove edb-efm41
```

- Debian または Ubuntu の場合：

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

- SLES の場合：

```
zypper remove edb-efm41
```

5.2 データベースの更新の実行（マイナーバージョン）

迅速なマイナーデータベースバージョンアップグレードを実行できます。1つのマイナーバージョンから別のマイナーバージョン（10.1.5 からバージョン 10.2.7 例）にアップグレードするか、バージョンにパッチリリースを適用できます。

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

クラスターの各ノードで、次の手順を実行してデータベースサーバを更新します。

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

EDB Postgres Advanced Server サービスの制御または EDB Postres Advanced Server のバージョンのアップグレード処理の詳細については、[EDB Postgres Advanced Server Guide](#) を参照してください。

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

第6章 Configuring streaming replication

</ div>

レプリケーションシナリオの構成は複雑になる場合があります。構成オプションの詳細については、[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の md5 認証を有効にすることができます。これは、環境にとって最も安全な認証メソッドではない可能性があります。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。

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

6.1 カスケーディングレプリケーションの限定サポート

フェールオーバーマネージャーはカスケーディングレプリケーションの完全なサポートを提供しませんが、カスケーディングレプリケーションシナリオでのシンプルフェイルオーバーの限定的なサポートを提供します。カスケードレプリケーションにより、スタンバイノードが別のスタンバイノードにストリームできるため、プライマリノードへの接続数（および処理オーバーヘッド）が削減されます。

カスケーディング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。

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

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

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

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

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

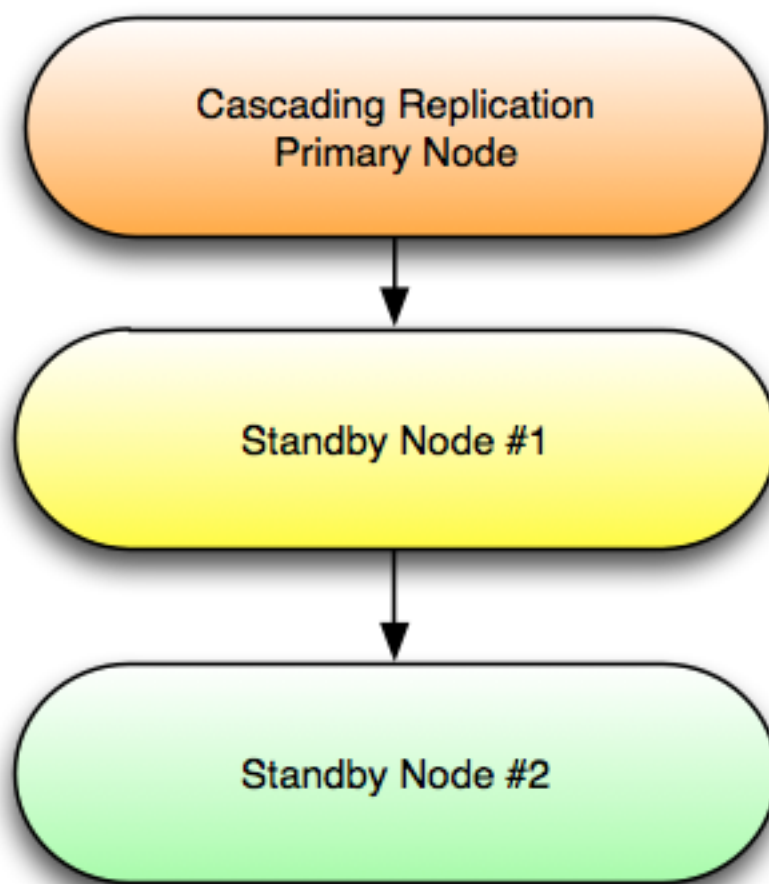


図 1: Cascading replication.

第7章 Configuring Failover Manager

</div>

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](#)

7.1 The cluster properties file

</div>

フェールオーバーマネージャークラスターの各ノードには、それが存在する個々のノードのプロパティを含むプロパティファイル（デフォルトでは `efm.properties` という名前）があります。Failover Manager インストーラは、`/etc/edb/efm-4.<x>`ディレクトリに `efm.properties.in` 名前付けのプロパティファイルのファイルテンプレートを作成します。

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

```
# cp /etc/edb/efm-4.4/efm.properties.in /etc/edb/efm-4.4/efm.properties
```

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

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

!!! Note * デフォルトでは、フェールオーバーマネージャーはクラスタープロパティファイルの名前付けが `efm.properties` であると想定しています。プロパティファイルに `efm.properties` 以外の名前を付ける場合は、サービススクリプト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、必要に応じて設定パラメータ値を追加または変更します。各プロパティの詳細については、*Specifying cluster properties* を参照してください。

プロパティファイルの所有者はルートです。Failover Manager サービススクリプトは、`/etc/edb/efm-4.<x>`ディレクトリ内のファイルを見つけることを想定しています。プロパティファイルを別の場所に移動する場合は、新しい場所を指定するシンボリックリンクを作成する必要があります。

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

</div>

7.1.1 クラスタープロパティの指定

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

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

`listen_addresses='192.168.2.47'`

フェールオーバーマネージャーでは、値を引用符で囲まないでください。

`bind.address=192.168.2.54:7800`

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

凡例：次の表：

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

Property name	A	W	Default value	Comments
[db.user](#db_user)	Y	Y		データベースのユーザー名。
[db.password.encrypted](#db_password_encrypted)		Y		'efmencrypt' を使用してパスワードを暗号化します。
[db.port](#db_port)	Y	Y		この値は、すべてのエージェントで同じである必要があります。
[db.database](#db_database)	Y	Y		データベース名前。

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[db.service.owner](#db_service_owner)	Y			db の \$PGDATA ディレクトリの所有者。データベース。
[db.service.name](#db_service_name)				データベースをサービスとして実行する場合に必要です。
[db.bin](#db_bin)	Y			'/ usr / edb / as * nn * / bin' などの pg_controldata / pg_ctl コマンドを含むディレクトリ。
[db.data.dir](#db_data_dir)	Y			クエリー の 出力 と 同じ 'show data_directory;'
[db.config.dir](#db_config_dir)				クエリー 'show-config_file;' の 出力 と 同じです。 * db.data.dir * と同じでない場合は、指定する必要があります。
[jdbc.sslmode](#jdbc_sslmode)	Y	Y	disable	[note](#jdbc_note) を参照してください。
[user.email](#user_email)				この値は、すべてのエージェントで同じである必要があります。通知スクリプトを使用する場合は空白のままにすることができます。
[from.email](#from_email).			efm@localhost	メールの efm@localhost を使用するには、空白のままにします。
[notification.level](#notification_level)	Y	Y	INFO	[list of notifications](../10_notifications/#notifications) を参照してください。
[notification.text.prefix](#notification_text_prefix)				NA

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[script.notification](#script_notification)				ユーザの場合は必須。メールプロパティは使用されません。両方のパラメーターを一緒に使用できます。
[bind.address](#bind_address)	Y	Y		例： <lt; ip_address> : <lt; port>
[external.address](#external_address)				例： <lt; ip_address / hostname>
[admin.port](#admin_port)	Y	Y	7809	デフォルトポートがすでに使用されている場合は変更します。
[is.witness](#is_witness)	Y	Y		説明を参照してください。
[local.period](#local_period)	Y		10	NA
[local.timeout](#local_timeout)	Y		60	NA
[local.timeout.final](#local_timeout_final)	Y		10	NA
[remote.timeout](#remote_timeout)	Y	Y	10	NA
[node.timeout](#node_timeout)	Y	Y	50	この値は、すべてのエージェントで同じである必要があります。
[encrypt.agent.messages](#encrypt_agent_messages)		Y	false	この値は、すべてのエージェントで同じである必要があります。
[enable.stop.cluster](#enable_stop_cluster)			true	この値は、すべてのエージェントで同じである必要があります。 ** Failover Manager 4.2 以降で利用可能です。 **
[stop.isolated.primary](#stop_isolated_primary)	Y		true	この値は、すべてのエージェントで同じである必要があります。
[stop.failed.primary](#stop_failed_primary)	Y		true	NA
[primary.shutdown.as.failure](#primary_shutdown_as_failure)	Y	Y	false	NA
[update.physical.slots.period](#update_physical_slots_period)			0	NA
[ping.server.ip](#ping_server_ip)	Y	Y	8.8.8.8	NA

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[ping.server.command](#ping_server_command)	Y	Y	/bin/ping -q -c3 -w5	NA
[auto.allow.hosts](#auto_allow_hosts)	Y	Y	false	NA
[stable.nodes.file](#stable_nodes_file)	Y	Y	false	NA
[db.reuse.connection.count](#db_reuse_connection_count)	Y		0	NA
[auto.failover](#auto_failover)	Y	Y	true	NA
[auto.reconfigure](#auto_reconfigure)	Y		true	この値は、すべてのエージェントで同じである必要があります。
[promotable](#promotable)	Y		true	NA
[use.replay.tiebreaker](#use_replay_tiebreaker)	Y	Y	true	この値は、すべてのエージェントで同じである必要があります。
[standby.restart.delay](#standby_restart_delay)			0	NA
[application.name](#application_name)				オリジナルのプライマリデータベースをスタンバイとして開始する前に、primary_conninfo エントリーの application_name 部分をこのプロパティ値に置き換えるように設定します。
[restore.command](#restore_command)				例：restore.command = scp < db_service_owner> @% h : < archive_path> /% f % p
[reconfigure.num.sync](#reconfigure_num_sync)	Y		false	Failover Manager 4.1 を使用している場合は、reconfigure_num_sync_max を参照して num_sync を上げます。

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[reconfigure.num.sync.max](#reconfigure_num_sync_max)				** Failover Manager4.1 以降で利用可能です。 **
[reconfigure.sync.primary](#reconfigure_sync_primary)			false	NA
[minimum.standbys](#minimum_standbys)	Y	Y	0	この値は、すべてのノードで同じである必要があります。
[priority.standbys](#priority_standbys)				** Failover Manager4.2 以降で利用可能です。 **
[recovery.check.period](#recovery_check_period)	Y		1	NA
[restart.connection.timeout](#restart_connection_timeout)			60	NA
[auto.resume.period](#auto_resume_period)	Y		0	NA
[virtual.ip](#virtual_ip)			(see virtual.ip.single)	VIP を指定しない場合は空白のままにします。
[virtual.ip.interface](#virtual_ip)				VIP を指定する場合に必要です。
[virtual.ip.prefix](#virtual_ip)				VIP を指定する場合に必要です。
[virtual.ip.single](#virtual_ip)	Y	Y	Yes	この値は、すべてのノードで同じである必要があります。
[check.vip.before.promotion](#check_vip_before_promotion)	Y	Y	Yes	NA
[pgpool.enable](#pgpool_enable)			false	** Failover Manager4.1 以降で利用可能です。 **
[pcp.user](#pgpool_enable)				pgpool.enable が true に設定されている場合に必要です。 ** Failover Manager4.1 以降で利用可能です。 **
[pcp.host](#pgpool_enable)				pgpool.enable が true に設定されている場合は必須です。この値は、すべてのエージェントで同じである必要があります。 ** Failover Manager4.1 以降で利用可能です。 **

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[pcp.port](#pgpool_enable)				pgpool.enable が true に設定されている場合は必須です。この値は、すべてのエージェントで同じである必要があります。 ** Failover Manager4.1 以降で利用可能です。 **
[pcp.pass.file](#pgpool_enable)				pgpool.enable が true に設定されている場合に必要です。 ** Failover Manager4.1 以降で利用可能です。 **
[pgpool.bin](#pgpool_enable)				pgpool.enable が true に設定されている場合に必要です。 ** Failover Manager4.1 以降で使用できます。 **
[script.load.balancer.attach](#script_load_balancer)				例：スクリプト。ロード.balancer.attach = /<パス> /< attach_script>% h % t
[script.load.balancer.detach](#script_load_balancer)				例：スクリプト。ロード.balancer.detach = /<パス> /< detach_script>% h % t
[detach.on.agent.failure](#detach_on_agent_failure)			true	実行中のプライマリデータベースを接続したままにする場合は、false に設定します。 ** Failover Manager4.2 以降で利用可能です。 **

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[script.fence](#script_fence)				例 : script.fence = / & lt; パス> /& lt; script_name>% p % f
[script.post.promotion](#script_post_promotion)				例 : script.post.promotion = /& lt; パス> /& lt; script_name>% f % p
[script.resumed](#script_resumed)				例 : script.resumed = /& lt; パス> /& lt; script_name>
[script.db.failure](#script_db_failure)				例 : script.db.failure = /& lt; パス> /& lt; script_name>
[script.primary.isolated](#script_primary_isolated)				例 : スクリプト。プ ライマリ.isolated = / & lt; パス> /& lt; script_name>
[script.remote.pre.promotion](#script_remote_pre_promotion)				例 : スクリプト。リ モート.pre.promotion = /& lt; パス> /& lt; script_name>% p
[script.remote.post.promotion](#script_remote_post_promotion)				例 : ス ク リ プ ト 。 リ モ ー ト.post.promotion = /& lt; パス> /& lt; script_name>% p
[script.custom.monitor](#script_custom_monitor)				例 : script.custom。モ ニタ= /& lt; パス> / & lt; script_name>
[custom.monitor.interval](#script_custom_monitor)				カスタムモニタリン グスクリプトが指定 されている場合に必 要です。
[custom.monitor.timeout](#script_custom_monitor)				カスタムモニタリン グスクリプトが指定 されている場合に必 要です。
[custom.monitor.safe.mode](#script_custom_monitor)				カスタムモニタリン グスクリプトが指定 されている場合に必 要です。

次のページに続く

表 1 – 前のページからの続き

Property name	A	W	Default value	Comments
[sudo.command](#sudo_command)	Y	Y	sudo	NA
[sudo.user.command](#sudo_command)	Y	Y	sudo -u %u	NA
[lock.dir](#lock_dir)				指定しない場合、デフォルトは /var / ロック / efm- & lt; バージョン> '
[log.dir](#log_dir)				指定しない場合、デフォルトは /var / ログ / efm- & lt; バージョン> '
[syslog.host](#syslog_logging)			localhost	NA
[syslog.port](#syslog_logging)			514	NA
[syslog.protocol](#syslog_logging)				NA
[syslog.facility](#syslog_logging)			UDP	NA
[file.log.enabled](#logtype_enabled)	Y	Y	true	NA
[syslog.enabled](#logtype_enabled)	Y	Y	false	NA
[jgroups.loglevel](#loglevel)			info	NA
[efm.loglevel](#loglevel)			info	NA
[jvm.options](#jvm_options)			- Xmx128m	NA

7.1.2 クラスターのプロパティ

</ div>

</ div>

</ div>

</ div>

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

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

(次のページに続く)

(前のページからの続き)

```
db.port=
db.database=
```

指定された `db.user` には、フェールオーバーマネージャーに代わって選択された PostgreSQL コマンドを呼び出すのに十分な権限が必要です。詳細については、[Prerequisites](#) を参照してください。

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

```
</div>
```

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

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

```
</div>
```

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

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

```
</div>
```

データベースサービスをスタート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は、`db.bin` プロパティで `pg_ctl` プログラムの場所を指定します。

```
# Specify the directory containing the pg_controldata/pg_ctl commands,
# for example:
```

(次のページに続く)

(前のページからの続き)

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

</ div>

db.data.dir プロパティを使用して、昇格中にクラスターのプライマリノードにリカバリファイルを書き込む場所を指定します。このプロパティは、プライマリノードとスタンバイノードで必要です。専用の監視ノードでは必要ありません。

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

</ div>

db.data.dir プロパティを使用して、昇格中にクラスターのプライマリノードにリカバリファイルを書き込む場所を指定します。このプロパティは、プライマリノードとスタンバイノードで必要です。専用の監視ノードでは必要ありません。

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

</ div>

`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` パラメータディレクトリで指定された値です。この値は、データベースを停止、起動、または再起動するときに、EDB Postgres Advanced Server data ディレクトリの場所として使用されます。

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

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

</ div>

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

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

</ div>

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

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

SSL の構成および使用については、PostgreSQL の文書を参照してください。

</ div>

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

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

</div>

`from.email` プロパティは、Failover Manager からのメール通知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。

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

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

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

</div>

`notification.level` プロパティを使用して、フェールオーバーマネージャーがユーザ通知を送信するとき、または通知スクリプトが呼び出されるとき、最小の重大度レベルを指定します。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 user's
# guide.
notification.level=INFO
```

</div>

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

```
# Text to add to the beginning of every notification. This could
# be used to help identify what the cluster is used for, the role
# of this node, etc. To use multiple lines, add a backslash \ to
```

(次のページに続く)

(前のページからの続き)

```
# the end of a line of text. To include a newline use \n.
# Example:
# notification.text.prefix=Development cluster for Example dept.\n\
# Used by Dev and QA \
# See Example group for questions.
notification.text.prefix=
```

</div>

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

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

</div>

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

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

</div>

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

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

(次のページに続く)

(前のページからの続き)

```
# an IP address other than the bind.address to the external world.
external.address=
```

</div>

`admin.port` プロパティを使用して、フェールオーバーマネージャー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
```

</div>

`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ノード。`is.witness` が `true` の場合、ローカルエージェントはローカルデータベースが実行されているかどうかを確認しません。

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

EDB Postgres Advanced Server の `pg_is_in_recovery()` ファンクションは、データベースのリカバリ状態を報告するブールファンクションです。このファンクションは、データベースがリカバリ中の場合は `true` を返し、データベースがリカバリ中でない場合は `false` を返します。エージェントが起動すると、ローカルデータベースに接続し、`pg_is_in_recovery()` ファンクションを呼び出します。

- サーバーが `true` と応答した場合、エージェントはスタンバイのロールを引き受けます。
- サーバーが `false` と応答した場合、エージェントはプライマリのロールを引き受けます。
- ローカルデータベースがない場合、エージェントはアイドル状態になります。

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

</div>

</div>

</div>

次のプロパティはローカルサーバに適用されます。

- `local.period` プロパティは、データベースサーバへの接続を試行する間隔を秒数で指定します。
- `local.timeout` プロパティは、エージェントがローカルデータベースサーバからの肯定的な応答を待つ秒数を指定します。
- `local.timeout.final` プロパティは、以前のチェックが現在のノード上のデータベースサーバへの接続に失敗した後、エージェントが待機する秒数を指定します。`local.timeout.final` プロパティ

で指定された秒数内にデータベースから応答を受信しなかった場合、データベースは失敗したと見なされます。

例、これらのプロパティのデフォルト値を指定すると、ローカルデータベースのチェックは 10 秒ごとに 1 回行われます。ローカルデータベースへの接続の試行が 60 秒以内に帰ってこない場合、Failover Manager はデータベースへの接続の最終試行を行います。応答が 10 秒以内に受信されない場合、Failover Manager はデータベース障害を宣言し、`user.email` プロパティにリストされている管理者に通知します。これらのプロパティは、専用の監視ノードでは必要ありません。

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

必要に応じて、ビジネスモデルに合わせてこれらの値を変更します。

</ div>

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

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

</ div>

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

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

</div>

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

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

</div>

`enable.stop.cluster` プロパティを使用して、`stop-cluster` コマンドを有効または無効にします。このコマンドは一部の環境では便利ですが、意図せずに起動すると問題が発生する可能性があります。Eager Failover モードでは、コマンドにより、フェイルオーバーなしで EDB Postgres Advanced Server が停止します。

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

</div>

`stop.isolated.primary` プロパティを使用して、プライマリエージェントがデータベースの分離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、Failover Manager は `script.primary.isolated` プロパティで指定されたスクリプトを呼び出す前にデータベースを停止します。

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

</div>

`stop.failed.primary` プロパティを使用して、フェールオーバーマネージャーに、プライマリデータベース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指示します。`true` の場合、Failover Manager は、データベースのシャットダウンを試行した後、`script.db.failure` プロパティで指定されたスクリプトを実行します。

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

(次のページに続く)

(前のページからの続き)

```
# If specified, a 'script.db.failure' script is run after this attempt.
stop.failed.primary=true
```

</div>

`primary.shutdown.as.failure` プロパティを使用して、プライマリノード上の Failover Manager エージェントのシャットダウンを障害として扱います。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、クラスターの残りの部分は、エージェントに障害が発生した場合に発生する通常のフェイルオーバー手順を実行します。通常のフェイルオーバー手順には、プライマリデータベースへの接続の試行、VIP が使用可能な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、フェイルオーバーが発生します。

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

`primary.shutdown.as.failure` プロパティは、プライマリノードの偶発的なシャットダウンなどの障害ではなく、ユーザエラーをキャッチするためのものです。ノードの適切なシャットダウンは、例プライマリデータベースのメンテナンスを実行するために、ユーザがプライマリフェールオーバーマネージャエージェントを停止したかのように、クラスターの残りの部分に表示されます。`primary.shutdown.as.failure` プロパティを `true` に設定した場合は、メンテナンスを実行するときに注意してください。

`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が失敗したがデータベースがまだ実行されているという通知を受信するまで待ちます。その後、プライマリデータベースを停止しても安全です。または、`stop-cluster` コマンドを使用して、障害チェックを実行せずにすべてのエージェントを停止できます。

</div>

`update.physical.slots.period` プロパティを使用して、データベースバージョン 12 以降のスロットアドバンス頻度を定義します。`update.physical.slots.period` が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` 秒ごとに物理的レプリケーションスロットの現在の `restart_lsn` を読み取り、`pg_current_wal_lsn` および `primary_slot_name` (`postgresql.conf` ファイルで設定されている場合) でこの情報を送信しますスタンバイ。物理的スロットがまだ存在しない場合、このパラメータを正の整数値に設定するとスロットが作成され、これらのスロットの `restart_lsn parameter` が更新されます。プロモーション不可のスタンバイでは、新しいスロットは作成されませんが、存在する場合は更新されます。

注：必要に応じて、現在のプライマリで設定されたもの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
```

</div>

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

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

</div>

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

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

</ div>

`auto.allow.hosts` プロパティを使用して、最初の `nostarts` の `.nodes` ファイルで指定されたアドレスを使用して許可ホストリストを更新するようにサーバーに指示します。`auto.allow.hosts` を `true` に設定してこのプロパティを有効にすると、クラスターのスタートアップが簡単になります。

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

</ div>

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

</ div>

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

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

</ div>

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

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

</ div>

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

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

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

</div>

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

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

</div>

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

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

</ div>

`standby.restart.delay` プロパティを使用して、昇格後に新しいプライマリを追跡するためにスタンバイが再構成 (stopstarts) されるまでスタンバイする時間を秒単位で指定します。

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

</ div>

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

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

!!! Note * プライマリおよびプロモーション可能なスタンバイで `application.name` プロパティを設定します。フェイルオーバー/スイッチオーバーが発生したイベント、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。

</ div>

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

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

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

</div>

プライマリノードのデータベースパラメータ `synchronous_standby_names` は、データの受信を保証してプライマリノードが書き込みトランザクションを受け入れることができるようにする同期的スタンバイサーバ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的スタンバイサーバの数を減らし、プライマリノードの構成を再読み込みして現在の値を反映します。

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

!!! Note * `reconfigure.num.sync` プロパティを使用している場合は、プライマリデータベースの `wal_sender_timeout` の値が `efm.node.timeout` の値より小さい設定されていることを確認してください。

</div>

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

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

(次のページに続く)

(前のページからの続き)

```
# Manager will not reconfigure the primary database if standbys are
# added to the cluster.
reconfigure.num.sync.max=
```

</div>

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

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

!!! Note * `reconfigure.sync.primary` プロパティを使用している場合は、プライマリデータベースの `wal_sender_timeout` の値が `efm.node.timeout` の値より小さい設定されていることを保証してください。

</div>

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

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

</div>

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

```
# Space-separated list of standby addresses that are high priority for
```

(次のページに続く)

(前のページからの続き)

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

</div>

`recovery.check.period` プロパティを使用して、フェールオーバーマネージャーがデータベースのリカバリ状態を確認する前に待機する秒数を指定します。

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

</div>

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

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

</div>

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

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

</div>

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

`prefix` プロパティで対応するプレフィックスを指定します。仮想 IP サポートを無効にするには、`virtual.ip` のままにします。

`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フェースを提供します。

指定された仮想 IP アドレスは、クラスターのプライマリノードにのみ割り当てられます。`virtual.ip.single=true` を指定すると、フェイルオーバーが発生した場合に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
# remapped during failover. If you do not use a VIP as part of
# your failover solution, leave the virtual.ip property blank to
# disable Failover Manager support for VIP processing (assigning,
# releasing, testing reachability, etc).
#
# If you specify a VIP, the interface and prefix are required.
#
# If you specify a host name, it will be resolved to an IP address
# when acquiring or releasing the VIP. If the host name resolves
# to more than one IP address, there is no way to predict which
# address Failover Manager will use.
#
# By default, the virtual.ip and virtual.ip.prefix values must be
# the same across all agents. If you set virtual.ip.single to
# false, you can specify unique values for virtual.ip and
# virtual.ip.prefix on each node.
#
# If you are using an IPv4 address, the virtual.ip.interface value
# should not contain a secondary virtual ip id (do not include
# ":1", etc).
virtual.ip=
virtual.ip.interface=
virtual.ip.prefix=
virtual.ip.single=true
```

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

</div>

`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、VIP が使用中かどうかを確認してから、障害が発生した場合に新しいプライマリに割り当てるようにします。これにより、マルチプルのノードが同じ VIP アドレスでブロードキャストする可能性があります。プライマリノードが分離されているか、別のプロセスでシャットダウンできない限り、このプロパティ `to true` を設定します。

```
# 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
```

</div>

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

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

以下のパラメーターを使用して、Pgpool 統合に使用する値を指定します。

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

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

</div>

次のプロパティを使用して、スイッチオーバーまたはプライマリ障害シナリオの場合にロードを再構成するスクリプトへの経路を提供します。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 witness) でそれらを保証し、データベースノードに障害が発生した場合、別のノードが障害ノードのア

ドレスで detach スクリプトを呼び出すようにします。

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

ノードをロードに接続するときに呼び出すスクリプトを識別するために、`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ます。`script.load.balancer.detach` プロパティを使用して、ロードからノードをデタッチするときに呼び出すスクリプトの名前を指定します。クラスターに接続または削除されるノードの IP アドレスを表す `%h` プレースホルダを含めます。文字列に `ap` (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` プレースホルダを含めます。

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

</ div>

`detach.on.agent.failure` プロパティを使用して、プライマリエージェントに障害が発生してもデータベースにアクセスできるシナリオで、ロードからノードを切り離したくないことを示します。デフォルト値は `true` です

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

</ div>

`script.fence` プロパティは、スタンバイノードをプライマリノードに昇格するときに呼び出すオプションのユーザー指定スクリプトへのパスを指定します。

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

</ div>

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

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

</ div>

`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モニタリングを再開したときに呼び出すユーザー指定のスクリプトへのオプションのパスを指定します。

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

</ div>

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

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

</ div>

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

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

</ div>

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

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

(次のページに続く)

(前のページからの続き)

```
# Pass a parameter (%p) with the script to identify the new
# primary node address.
#
# Example:
# script.remote.pre.promotion=/path_name/script_name %p
script.remote.pre.promotion=
```

</div>

`script.remote.post.promotion` プロパティを使用して、プロモーションの発生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 プレースホルダを含めて、新しいプライマリノードのアドレスを識別します。

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

</div>

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

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

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

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

(次のページに続く)

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

</div>

`sudo.command` プロパティを使用して、拡張アクセス許可が必要なタスクを実行するときに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したコマンドを実行するときに呼び出すフェールオーバーマネージャーのコマンドを指定します。

```
# Command to use in place of 'sudo' if desired when efm runs
# the efm_db_functions or efm_root_functions, or efm_address
# scripts.
# Sudo is used in the following ways by efm:
#
# sudo /usr/edb/efm-<version>/bin/efm_address <arguments>
# sudo /usr/edb/efm-<version>/bin/efm_root_functions <arguments>
# 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.
```

(次のページに続く)

(前のページからの続き)

```
sudo.command=sudo
sudo.user.command=sudo -u %u
```

</div>

`lock.dir` プロパティを使用して、Failover Manager 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、ノード上の単一クラスターに対してマルチプルの（潜在的に孤立した）エージェントを開始するのを防ぎます。

```
# 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=
```

</div>

`log.dir` プロパティを使用して、エージェントログファイルを書き込む場所を指定します。フェールオーバーマネージャーは、ディレクトリを作成しようとします。

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

</div>

Failover Manager ホストで UDP または TCP プロトコルを有効にした後、syslog へのロギングを有効にできます。`syslog.protocol` パラメータを使用してプロトコルタイプ (UDP または TCP) を指定し、`syslog.port` パラメータを使用して syslog ホストのリスナポートを指定します。`syslog.facility` 値を、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 と LOCAL7 の間の値を使用します。

```
# 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
```

</div>

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

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

`syslog` ログギングの構成の詳細については、[Enabling syslog log file entries](#) を参照してください。

</ div>

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

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

</ div>

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

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

`encrypting_database_password`

</ div>

7.2 Encrypting your database password

</div>

フェールオーバーマネージャーでは、データベースのパスワードをクラスタープロパティファイルに含める前に暗号化する必要があります。/usr/edb/efm-4.<x>/bin ディレクトリにある [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、Failover Manager クラスターの名前を指定します。

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

```
export EFMPASS=password
```

--from-env オプションを含めない場合、フェールオーバーマネージャーは、クラスタープロパティファイルに配置するための暗号化パスワードを生成する前に、データベースパスワードを2回入力するように求めます。ユーティリティが暗号化されたパスワードを共有したら、暗号化されたパスワードをコピーしてクラスタープロパティファイルに貼り付けます。

!!! Note * 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、エクスポート制限のために有効化されていません。データベースパスワードを暗号化しようとしたときに不正なキーサイズを参照するエラーが発生した場合は、プラットフォームに制限のないポリシーを提供する Java 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ことを示しています。

```
# efm encrypt acctg
This utility will generate an encrypted password for you to place in
your Failover Manager cluster property file:
/etc/edb/efm-4.4/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

!!! Note * 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 サービスをスタートします。暗号化されたパスワードに問題がある場合、Failover Manager サービスはスタートされません。

```
[witness@localhost ~]# systemctl start edb-efm-4.4
Job for edb-efm-4.4.service failed because the control process exited with error code.
See "systemctl status edb-efm-4.4.service" and "journalctl -xe" for details.
```

Failover Manager サービスの開始時にこのメッセージを受け取った場合、詳細についてはスタートアップログ `/var/log/efm-4.4/startup-efm.log` を参照してください。

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

```
systemctl status edb-efm-4.4
```

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

7.2.1 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 `--from-env` 環境変数を使用することを示しています。efm encrypt コマンドを呼び出す前に、EFMPASS の値をパスワード `1safepassword` に設定します。

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

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

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

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

7.3 The cluster members file

</div>

Failover Manager クラスターの各ノードには、現在の Failover Manager クラスターメンバーのリストを含むクラスターメンバーファイル（デフォルトでは `efm.nodes` 名前付け）があります。エージェントが起動すると、ファイルを使用して他のクラスターメンバーを見つけます。フェールオーバーマネージャーインストーラは、`/etc/edb/efm-4.<x>` ディレクトリに `efm.nodes.in` 名前付けのクラスターメンバーファイルのファイルテンプレートを作成します。

Failover Manager のインストールが完了したら、テンプレートの作業用コピーを make します。

```
cp /etc/edb/efm-4.4/efm.nodes.in /etc/edb/efm-4.4/efm.nodes
```

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

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

デフォルトでは、フェールオーバーマネージャーはクラスターメンバーファイルの名前付けが `efm.nodes` であると想定しています。クラスターメンバーファイルに `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。

開始された最初のノードのクラスターメンバーファイルは空にすることができます。このノードがメンバーシップコーディネーターになります。後続の各ノードでは、クラスターメンバーファイルにメンバーシップコーディネーターのアドレスとポート番号が含まれている必要があります。クラスターメンバーファイルの各エントリは、アドレス：ポートフォーマットでリストする必要があります。マルチプル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にマッチするように `efm.nodes` ファイルの内容を更新します。エージェントがクラスターに結合または離脱すると、他のエージェントの `efm.nodes` ファイルが更新され、現在のクラスターメンバーシップが反映されます。`efm stop-cluster <../07\_using\_efm\_utility/#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。

メンバーシップコーディネーターがクラスターを離れると、別のノードがロールになります。`efm cluster-status <../07\_using\_efm\_utility/#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。エージェントが停止している間にノードがクラスターに参加または離脱する場合、そのエージェントを開始する前に、ファイルに少なくとも現在のメンバーシップコーディネーターのアドレスとポートが含まれていることを手動で保証してください。

クラスターに参加しているノードのアドレスとポートがわかっている場合は、いつでもクラスターメンバーファイルにアドレスを含めることができます。スタートアップに、`cluster properties file <01\_cluster\_properties/#auto\_allow\_hosts>` __ の `auto.allow.hosts` プロパティが `true` に設定されていない限り、クラスターメンバーを識別しないアドレスは無視されます。

) の `stable.nodes.file` プロパティが `true` に設定されている場合、クラスターメンバーがクラスターに結合または離脱してもエージェントは `.nodes` ファイルを更新しません。この動作は、クラスターメンバーの IP アドレスが頻繁に変更されない場合に最も役立ちます。

7.4 Extending Failover Manager permissions

</div>

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

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

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

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

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

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

```
# Copyright EnterpriseDB Corporation, 2014-2021. All Rights Reserved.
#
# Do not edit this file. Changes to the file may be overwritten
# during an upgrade.
#
# This file assumes you are running your efm cluster as user 'efm'. If not,
# then you will need to copy this file.

# Allow user 'efm' to sudo efm_db_functions as either 'postgres' or 'enterprisedb'.
# If you run your db service under a non-default account, you will need to copy
# this file to grant the proper permissions and specify the account in your efm
# cluster properties file by changing the 'db.service.owner' property.
efm    ALL=(postgres)      NOPASSWD:    /usr/edb/efm-4.4/bin/efm_db_functions
efm    ALL=(enterprisedb)  NOPASSWD:    /usr/edb/efm-4.4/bin/efm_db_functions

# Allow user 'efm' to sudo efm_root_functions as 'root' to write/delete the PID file,
# validate the db.service.owner property, etc.
efm    ALL=(ALL)           NOPASSWD:    /usr/edb/efm-4.4/bin/efm_root_functions

# Allow user 'efm' to sudo efm_address as root for VIP tasks.
efm    ALL=(ALL)           NOPASSWD:    /usr/edb/efm-4.4/bin/efm_address

# Allow user 'efm' to sudo efm_pgpool_functions as root for pgpool tasks.
efm    ALL=(ALL)           NOPASSWD:    /usr/edb/efm-4.4/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モニタしている場合は、`efm-42` ファイルのコピーを `make` し、ユーザが `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コンテンツを変更します。

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

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

# includedir /etc/sudoers.d
```

</div>

7.4.1 sudo なしでフェールオーバーマネージャーを実行する

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

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

`sudo` なしで Failover Manager を実行するには、Failover Manager に代わって管理機能を実行する権限を持つデータベースプロセス所有者を選択する必要があります。ユーザは、デフォルトデータベーススーパーユーザ（`enterprisedb` や `postgres` 例）または別の特権ユーザになります。ユーザを選択した後：

1. 次のコマンドを使用して、ユーザを `efm` グループに追加します。

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

- このコマンドにより、ユーザは `/var/run/efm-4.<x>` および `/var/lock/efm-4.<x>` に書き込むことができます。

2. クラスター名前を再利用する場合は、以前に作成したログファイルを削除します。新しいユーザは、デフォルトまたは他の所有者が作成したログファイルに書き込むことができません。
3. クラスタープロパティテンプレートファイルとノードテンプレートファイルをコピーします。

```
* su - enterprisedb
* cp /etc/edb/efm-4.4/efm.properties.in <directory/>cluster_name>.properties
* cp /etc/edb/efm-4.4/efm.nodes.in <directory/><cluster_name>.nodes
```

次に、クラスタープロパティファイルを変更し、`db.service.owner` プロパティでユーザの名前を指定します。また、`db.service.name` プロパティが空白である `make` も確認してください。`sudo` がないと、ルートアクセスなしでサービスを実行できません。

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

```
/usr/edb/efm-4.4/bin/runefm.sh start|stop <directory/>cluster_name>.properties
```

ここで、`<directory/>cluster_name.properties>` はクラスタープロパティファイルのフルパスを指定します。デフォルト以外のユーザ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いつでも、ユーザはプロパティファイルへのフルパスを提供します。

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

Failover Manager は、`/usr/edb/efm-4.<x>/bin/secure/` にある `manage-vip` 名前付けのバイナリを使用して、`sudo` 権限なしで VIP 管理操作を実行します。このスクリプトは、`setuid` を使用して、仮想 IP アドレスの管理に必要な特権を取得します。

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

セキュリティ上の理由から、`/usr/edb/efm-4.<x>/bin/secure/`ディレクトリまたは `manage-vip` スクリプトのアクセス権限を変更しないことをお勧めします。

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

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

7.5 Using Failover Manager with virtual IP addresses

</div>

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

!!! Note * 仮想 IP アドレスは、多くのクラウドプロバイダーでサポートされていません。これらの環境では、AWS の Elastic IP アドレスなどの別のメカニズムを使用します。これは、フェンシングまたはポストプロモーションスクリプトで必要に応じて変更できます。

デフォルトでは、スクリプトは次の場所にあります。

`/usr/edb/efm-4.<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リリースします。

仮想 IPv4 IP アドレスを割り当てるには：

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

仮想 IPv6 IP アドレスを割り当てるには：

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

仮想アドレスをリリースするには：

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

どこで：

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

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

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

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

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

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

7.5.1 VIP のテスト

フェールオーバーマネージャーで仮想 IP (VIP) アドレスを使用する場合、フェールオーバーマネージャーを起動する前に VIP 機能を手動でテストすることが重要です。これにより、実際のフェールオーバー中に問題が発生する前に、ネットワーク関連の問題をキャッチします。VIP のテスト中に、Failover Manager 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
```

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

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

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

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

100 %のパケット損失が見られます。

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

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

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

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

パケット損失は発生しません。

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

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

このステップの出力には、`eth0` インタフェースは表示されません。

5. ステップ 3 を繰り返します。今回は、使用中の VIP がスタンバイとミラーリング監視サーバーに表示されないことを確認します。

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

100 % のパケット損失が発生します。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 を繰り返し、VIP をすべてのノードに割り当てます。任意のノードから VIP をピングして、使用中であることを確認できます。

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

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

!!! Note * VIP に使用されるネットワークインタフェースは、Failover Manager エージェントの `bind.address` 値に使用されるインタフェースと同じである必要はありません。プライマリエージェントは、フェ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、プライマリ分離とフェールオーバーにつながります。

VIP が別のインタフェースを使用する場合、プライマリエージェントがドロップする前に、クラスタの残りが到達可能な VIP をチェックするタイミング条件が発生する可能性があります。この場合、フェールオーバーマネージャーは、ヘルププロパティで指定された秒数の VIP チェックを再試行して、フェイルオーバーが期待どおりに行われる保証にします。

7.6 Configuring for Eager Failover

デフォルトの実行モードでは、プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動するまでフェイルオーバー保護はありません。このケースを回避するには、`systemd` を使用してプライマリノードをセットアップし、プライマリエージェントが終了したときにフェイルオーバーが発生するようにします。これは、Eager Failover と呼ばれます。

次の手順を実行して、Eager Failover をセットアップできます。この例では、EDB Postgres Advanced Server バージョン 12 および Failover Manager バージョン 4.2 を使用しています。

</div>

7.6.1 Eager Failover を有効にする

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

- フェールオーバーマネージャーを起動する前にこのプロパティを設定しない場合、フェールオーバーマネージャーエージェントをシャットダウンすると、フェイルオーバーなしでデータベースがシャットダウンされます。
- Eager Failover を有効にして、`efm stop-cluster` コマンドを使用すると、すべての 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します。エージェントが実行されていないため、フェイルオーバーはありません。このシナリオを回避するには、`enable.stop.cluster` プロパティを使用してコマンドを無効にします。

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

- データベースサーバとローカルフェ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。

ルートとして、`/etc/systemd/system/edb-as-12.service` ファイルを作成し、以下を含めます。

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

- 次のコマンドを実行して、構成ファイルをリロードします。

```
* systemctl daemon-reload
```

これらの変更により、Failover Manager エージェントが停止または終了すると、クラスターの残りの部分はこのシチュエーションを障害として扱い、フェイルオーバーを試行します。

</div>

7.6.2 Eager Failover の無効化

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

EDB Postgres Advanced Server を停止せずに Failover Manager をアップグレードするには、Eager Failover モードを一時的に無効にします。

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

7.6.4 重要な注意事項

- `systemd` コマンドは、`sudo` 以外のセットアップでフェールオーバーマネージャーを実行している間はデータベースの管理に使用されないため、Eager Failover は `sudo` モードでのみサポートされます。非 `sudo` モードではサポートされていません。
- Eager Failover は、VIP が古いプライマリによってリリースされない状況には適していません。
- Eager Failover は、次の状況に適しています。

EDB Postgres Advanced Server の高可用性セットアップ。-または `libpq` を使用したクライアント接続フェイルオーバーを使用したセットアップ。-`script.fence` によってトリガーされたカスタムスクリプトが古いプライマリサーバ (STONITH) をフェンスする場合。いくつかの例は、VMWare vCenter 統合、OpenStack 統合、または Lights Out 管理で VM をシャットダウンすることです。-`script.fence` によってトリガーされた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。

- **!!! Note**
- 古いプライマリがリリースする前に新しいプライマリが VIP を接続できるようにするには、`check.vip.before.promotion=false` の設定が必要です。
- `primary.shutdown.as.failure=true` を使用する場合は注意してください。必要に応じてデータベースを安全に停止する方法については、`primary.shutdown.as.failure` プロパティの説明を参照してください。
- すべてのフェイルオーバーで、プライマリは失敗したプライマリになりますが、運用スタンバイとして自動的に回復しません。したがって、クラスターにマルチプルの昇格可能なスタンバイが含まれ `make` いること、および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てください。これは一般的な推奨事項ですが、Eager Failover を使用する場合はより緊急になります。

- データベースサーバが停止している場合、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も起動します。
 - !!! Note
 - -悪いプロパティ値など、Failover Manager の起動に問題がある場合、データベースサーバは起動していないことを示すワーニングを表示することなく、再びシャットダウンします。-Failover Manager プロセスが以前に終了した場合、ロックファイルはまだ存在し、エージェントは自動的にリスタートできません。-データベースサーバまたは Failover Manager エージェントの起動時に問題が発生した場合は、Failover Manager のスタートアップログで情報を確認してください。
- **stop-cluster** コマンドを実行した結果、Failover Manager はすべてのノードで停止します。 Eager Failover モードでは、**stop-cluster** コマンドは、フェイルオーバーなしで EDB Postgres Advanced Server も停止します。**stop-cluster** を意図せずに起動できない make に、**enable.stop.cluster=false** を設定します。

第8章 Configuring SSL authentication on a Failover Manager cluster

</div>

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。権限によって署名された証明書を購入するか、独自の自己署名証明書を作成できます。自己署名証明書の作成については、[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の各データベースで `postgresql.conf` ファイルを変更し、SSL を有効にします。

```
*    ssl=on
```

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

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

```
*    hostnossl all all all reject
```

- この行は、SSL 認証を使用していない接続を拒否するようサーバーに指示します。これにより、接続しているクライアントに SSL 認証が強制されます。`pg_hba.conf` ファイルの変更については、[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 が理解できるフォームに変換します。次のコマンドを使用できます。

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

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

1. 証明書を Java 信頼証明書ファイルに追加します。

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

- どこで：
- `$JAVA_HOME` は、Java インストールのホームディレクトリです。
- `alias_name` には任意の文字列を指定できますが、証明書ごとに一意である必要があります。
- `keytool` コマンドを使用して、使用可能な証明書のリストを確認したり、特定の証明書に関する情報を取得したりできます。`keytool` コマンドの使用の詳細については、次のように入力してください。

* <code>man keytool</code>

- 各データベースサーバからの証明書は、各エージェントの信頼できる証明書ファイルにインポートする必要があります。`cacerts` ファイルの場所は、システムごとに異なる場合があります。詳細については、[Postgres JDBC documentation](#) を参照してください。
1. `jdbc.sslmode` プロパティを設定して、クラスター内の各ノードで `efm.properties file` を変更します。

第9章 Using Failover Manager

</ div>

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

プライマリノードが再起動すると、フェールオーバーマネージャーは、プライマリノードでデータベースがダウンしていることを検出し、スタンバイノードをプライマリのロールにプロモート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、Postgres が二次的なプライマリとしてスタートしないように `recovery.conf` ファイルを書き込め `make` とします。したがって、データベースサーバを起動する前に Failover Manager エージェントをスタート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、クラスターに既にプライマリがあるかどうかを確認します。プライマリノードがある場合、エージェントは `recovery.conf` または `standby.signal` ファイルが存在することを確認するか、必要に応じて `recovery.conf` を作成して、データベースが 2 番目のプライマリとして起動しないようにします。

9.1 フェールオーバーマネージャークラスターの管理

構成が完了すると、Failover Manager クラスターは定期的なメンテナンスを必要としません。ただし、Failover Manager クラスターで必要になることがあ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 スーパーユーザによって呼び出される必要があります。管理者は、ユーザを `efm` グループに追加することにより、ユーザーがこれらのコマンドを呼び出すことを選択的に許可できます。コマンドは次のとおりです。

- `efm allow-node`
- `efm disallow-node`
- `efm promote`
- `efm resume`
- `efm set-priority`
- `efm stop-cluster`
- `efm upgrade-conf`

</ div>

9.1.1 Failover Manager クラスターの起動

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

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

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

ノードのクラスタープロパティファイルで `is.witness` が `true` であると指定されている場合、ノードは監視ノードとして起動しノード。

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

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

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

</div>

!!! Note * `tmpfs` (Temporary File System) で `make` または `/var/run` を実行している場合、Failover Manager の `systemd` サービスファイルが `systemd-tmpfiles-setup.service` に依存していることを確認してください。

9.1.2 クラスターにノードを追加する

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

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前とアドレスを指定します。

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

2. 新しいノードでクラスターメンバーファイルを構成し、メンバーシップコーディネーターのエントリーを追加します。クラスターメンバーファイルの変更の詳細については、[The cluster members file](#) を参照してください。

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

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

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

</ div>

!!! Note * 現在のノードの便利なスタンバイになるには、PostgreSQL Streaming Replication シナリオでノードがスタンバイである必要があります。

9.1.3 スタンバイの優先度を変更する

フェールオーバーマネージャークラスターに複数のスタンバイサーバが含まれている場合は、`efm set-priority` コマンドを使用して、スタンバイノードのプロモーションの優先度に影響を与えることができます。Failover Manager クラスターの既存のメンバでコマンドを呼び出し、メンバの IP アドレスの後に優先度の値を指定します。

例、次のコマンドは、フェールオーバーマネージャーに、`10.0.1.9` をモニタリングしている `acctg` クラスターメンバがプライマリスタンバイ (1) であることを指示します。

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

スタンバイをスタンバイ不可にすることができます。スタンバイの優先度を 0 より大きい値に設定すると、`promotable=false` のプロパティ値がオーバーライドされます。

例、ノード `10.0.1.10` のプロパティファイルに `promotable=false` の設定が含まれており、`efm set-priority` を使用して `10.0.1.10` のプロモーション優先度をフェールオーバーのイベントで使用されるスタンバイに設定する場合、`efm set-priority` コマンドで指定された値は、プロパティファイルの値をオーバーライドします。

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

フェールオーバーが発生したイベント、フェールオーバーマネージャーはまず Postgres ストリーミングレプリケーションから情報を取得して、どのスタンバイノードに最新のデータがあるかを確認し、データロスの可能性が最も低いノードをプロモートさせます。2つのスタンバイノードに等しく最新のデータが含まれている場合、`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順位値が高いノードが `primary` に昇格されます。スタンバイノードの優先度の値を確認するには、次のコマンドを使用します。

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

</ div>

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

- `efm set-priority` コマンドを使用してスタンバイが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の分割と再結合により、スタンバイのプロパティファイルの値にこれをリセットできます。

9.1.4 フェールオーバーマネージャノードの昇格

フェールオーバーマネージャクラスタの任意のノードで `efm promote` を呼び出して、スタンバイデータベースのプライマリデータベースへのマニュアル昇格をスタートできます。

データベースクラスタのメンテナンスウィンドウの間にのみマニュアル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は、続行する前にプロンプトが表示されます。マニュアルプロモーションをスタートするには、`efm` または OS スーパーユーザの ID を想定して、コマンドを呼び出します。

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

どこで：

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

`-switchover` オプションを含めて、オリジナルのプライマリをスタンバイとして再構成します。`-switchover` キーワードを含める場合、クラスタ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、フェンシングスクリプトとポストプロモーションスクリプトを呼び出さないようにフェールオーバーマネージャに指示します。

スイッチオーバー中：

- サーババージョン 11 以前の場合、`recovery.conf` ファイルは既存のスタンバイからプライマリノードにコピーされます。サーババージョン 12 以降では、`primary_conninfo` および `restore_command` パラメーターがコピーされ、メモリに保存されます。
- プライマリデータベースは停止しています。
- VIP を使用している場合、アドレスはプライマリノードから解放されます。
- プライマリノードを置き換えるためにスタンバイが昇格され、VIP を取得します。
- 新しいプライマリノードのアドレスが `recovery.conf` ファイルに追加されるか、`primary_conninfo` の詳細がメモリに保存されます。
- `application.name` プロパティがこのノードに設定されている場合、`application_name` プロパティが `recovery.conf` ファイルに追加されるか、`primary_conninfo` 情報がメモリに保存されます。
- サーババージョン 12 以降を使用している場合、メモリに保存されたりリカバリ設定は `postgresql.auto.conf` ファイルに書き込まれます。`standby.signal` ファイルが作成されます。

- 古いプライマリが開始されます。エージェントは、スタンバイとしてのモニタリングを再開します。

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

プライマリエージェントは実行されたままで、アイドル状態になります。

スタンバイエージェントは、仮想 IP アドレスが使用されていないことを保証してから、既知のアドレスに ping を送信して、エージェントがネットワークから隔離されないようにします。スタンバイエージェントはフェンシングスクリプトを実行し、スタンバイデータベースをプライマリに昇格させます。次に、スタンバイエージェントは仮想 IP アドレスをスタンバイノードに割り当て、ポストプロモーションスクリプトを実行します（該当する場合）。

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

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

次に、マニュアル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。

</div>

9.1.5 Failover Manager エージェントの停止

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

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

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

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

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

9.1.6 Failover Manager 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 スーパーユーザの ID を想定して、コマンドを呼び出します。

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

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

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

9.1.7 クラスターからノードを削除する

efm disallow-node コマンドは、ノードの IP アドレスを Failover Manager の許可ノードホストリストから削除します。現在実行中のクラスターのパートである既存のノード上の efm または OS スーパーユーザの ID を想定します。次に、ノードのクラスター名前と 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した場合、ノードはクラスターへの再参加を許可されず、フェイルオーバー優先順位リストから削除されます。プロモーションの対象外となります。

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

</ div>

9.2 単一ノードでマルチプルのエージェントを実行する

フェールオーバーマネージャークラスターでマルチプルのプライマリエージェントまたはスタンバイエージェントを実行することにより、同じホストにあるマルチプルのデータベースクラスターをモニタできます。単一のノードでマルチプルの監視エージェントを実行することもできます。異なるクラスターからのフェールオーバーマネージャークラスター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モニタ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。クラスターのメンバーをリストする各クラスターのメンバごとにクラスターメンバーファイルを作成します。各クラスターのユニットファイル (RHEL / CentOS 7.x または RHEL / CentOS 8.x システム) をカスタマイズして、クラスタープロパティとクラスターメンバーファイルの名前を指定します 4. 各クラスターのサービスを開始します。

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

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

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

各クラスタープロパティファイルで、`db.port` パラメータは各クラスターの一意の値を指定します。`db.user` および `db.database` パラメータは、同じ値または一意の値を持つことができます。例、`acctg.properties` ファイルでは次を指定できます。

`db.user=efm_user`

`db.password.encrypted=7c801b32a05c0c5cb2ad4ffbd5e8f9a`

`db.port=5444`

`db.database=acctg_db`

`sales.properties` ファイル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`

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

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

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

`virtual.ip`

`virtual.ip.interface`

`virtual.ip.prefix`

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

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

!!! Note * ユニットファイルを使用している場合、Failover Manager のアップグレード時に新しいサービス名前を反映するようにファイルを手動で更新します。

9.2.1 RHEL / CentOS 7.x または RHEL / CentOS 8.x

RHEL / CentOS 7.x または RHEL / CentOS 8.x を使用している場合は、`edb-efm-4.<x>` ユニットファイルを、クラスターごとに一意の名前を持つ新しいファイルにコピーします。例、`acctg` と `sales` 名前付けの 2 つのクラスターがある場合、ユニットファイル名は次のようになります。

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

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

次に、各ユニットファイルの `CLUSTER` 変数を編集し、指定されたクラスター名前を `efm` から新しいクラスター名前に変更します。例、`acctg` 名前付けのクラスターの場合、値は次を指定します。

```
Environment=CLUSTER=acctg
```

また、`PIDfile` パラメータの値を更新して、新しいクラスター名前を指定します。例：

```
PIDFile=/var/run/efm-4.4/acctg.pid
```

サービススクリプトをコピーした後、サービスを有効にします。

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

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

次に、新しいサービススクリプトを使用してエージェントをスタートします。例、`acctg` エージェントをスタートするには：

```
# 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。

第10章 Using the efm utility

</div>

フェールオーバーマネージャーは、クラスター管理を支援する **efm** ユーティリティを提供します。RPM インストーラは、Failover Manager のインストール時に、ユーティリティを `/usr/edb/efm-4.<x>/bin` ディレクトリに追加します。

10.1 efm allow-node

</div>

efm allow-node <cluster_name>

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

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

10.2 efm disallow-node

</div>

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できないようにします。**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、またはルートによって呼び出される必要があります。

10.3 efm cluster-status

</div>

efm cluster-status <cluster_name>

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

10.4 efm cluster-status-json

</div>

`efm cluster-status-json <cluster_name>`

`efm cluster-status-json` コマンドを呼び出して、フェールオーバーマネージャークラスターのステータスを JSON フォーマットで表示します。表示される情報のフォーマットは `efm cluster-status` コマンドによって生成される表示とは異なりますが、情報ソースは同じです。

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

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

(次のページに続く)

(前のページからの続き)

```

*      "172.16.144.180"
*    ],
*    "minimumstandbys": 0,
*    "missingnodes": [],
*    "messages": []
}

```

10.5 efm 暗号化

</ div>

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

クラスタプロパティファイルにパスワードを含める前に、`efm encrypt` コマンドを呼び出してデータベースパスワードを暗号化します。--from-env オプションを含めて、フェールオーバーマネージャーに `EFMPASS` 環境変数で指定された値を使用し、ユーザ入力なしで実行ように指示します。詳細については、[Encrypting your database password](#) を参照してください。

10.6 efm プロモート

</ div>

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

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

プライマリコマンドで最新のスタンバイノードがクラスターに含まれていることが `status` コマンドから報告された場合にのみ、マニュアル昇格を試みます。最新のスタンバイがない場合、フェールオーバーマネージャーは続行する前にプロンプトを表示します。

-switchover 句を含めて、スタンバイノードをプロモートさせ、プライマリノードをスタンバイノードとして再構成します。-sourcenode キーワードを含め、ノードアドレスを指定して、リカバリ設定が古いプライマリノードにコピーするノードを指定します。これにより、スタンバイになります。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。フェンダーまたはプロモーション後のスクリプトを呼び出さないようにフェールオーバーマネージャーに指示するには、-noscripts キーワードを含めます。

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

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

10.7 efm 履歴書

</ div>

efm resume <cluster_name>

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

10.8 efm set-priority

</ div>

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

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

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

10.9 efm stop-cluster

</ div>

efm stop-cluster <cluster_name>

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

!!! Note * **efm stop-cluster** コマンドを呼び出すと、許可された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、またはルートによって呼び出される必要があります。

10.10 efm upgrade-conf

</ div>

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

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

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

10.11 efm node-status-json

</ div>

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

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

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

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

10.12 efm-ヘルプ

</ div>

```
efm --help
```

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

第11章 Monitoring a Failover Manager cluster

</div>

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

11.1 クラスターステータスレポートの確認

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

```
# 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 という名前付けのクラスターに関するものです。

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

Allowed node host list:

172.19.14.9 172.19.12.163 172.19.10.2

Membership coordinator: 172.19.14.9

Standby priority host list:

172.19.12.163 172.19.10.2

Promote Status:

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

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

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

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

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

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

Allowed node host list と Standby priority host list を使用すると、クラスターに結合できるノードとノードの昇格オーダーを簡単に確認できます。メンバーシップコーディネーターの IP アドレスもレポートに表示されます。

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

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

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

データベースがダウンしているか、データベースが再起動されたが、再開コマンドがまだ呼び出されていない場合、そのホストに存在するエージェントの状態はアイドル状態です。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	Agent	DB	VIP
Idle	172.19.18.105	UP	UP	172.19.13.105

11.1.1 終了コード

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

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

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

- フェールオーバーマネージャーは、指定されたデータベースパスワードを解読できません。
- データベースに接続して WAL の場所を取得する際に問題が発生しました。

プライマリエージェントはありません。

スタンバイエージェントはありません。

- 1 つ以上のスタンバイノードがプライマリと同期していません。

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

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

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

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

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

- ストリーミングレプリケーション
- WAL アーカイブステータス

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

- ストリーミングレプリケーションの遅延時間

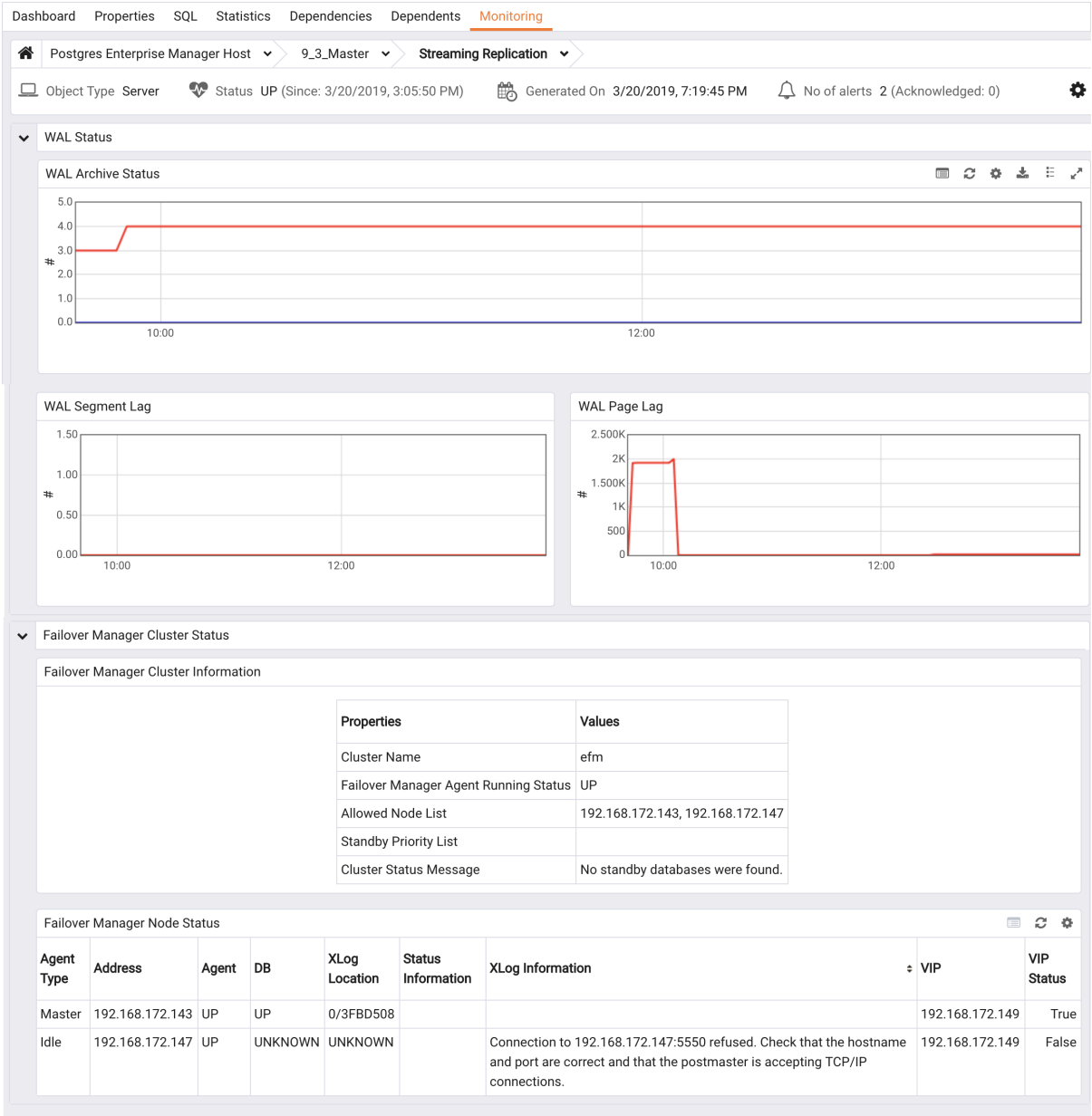
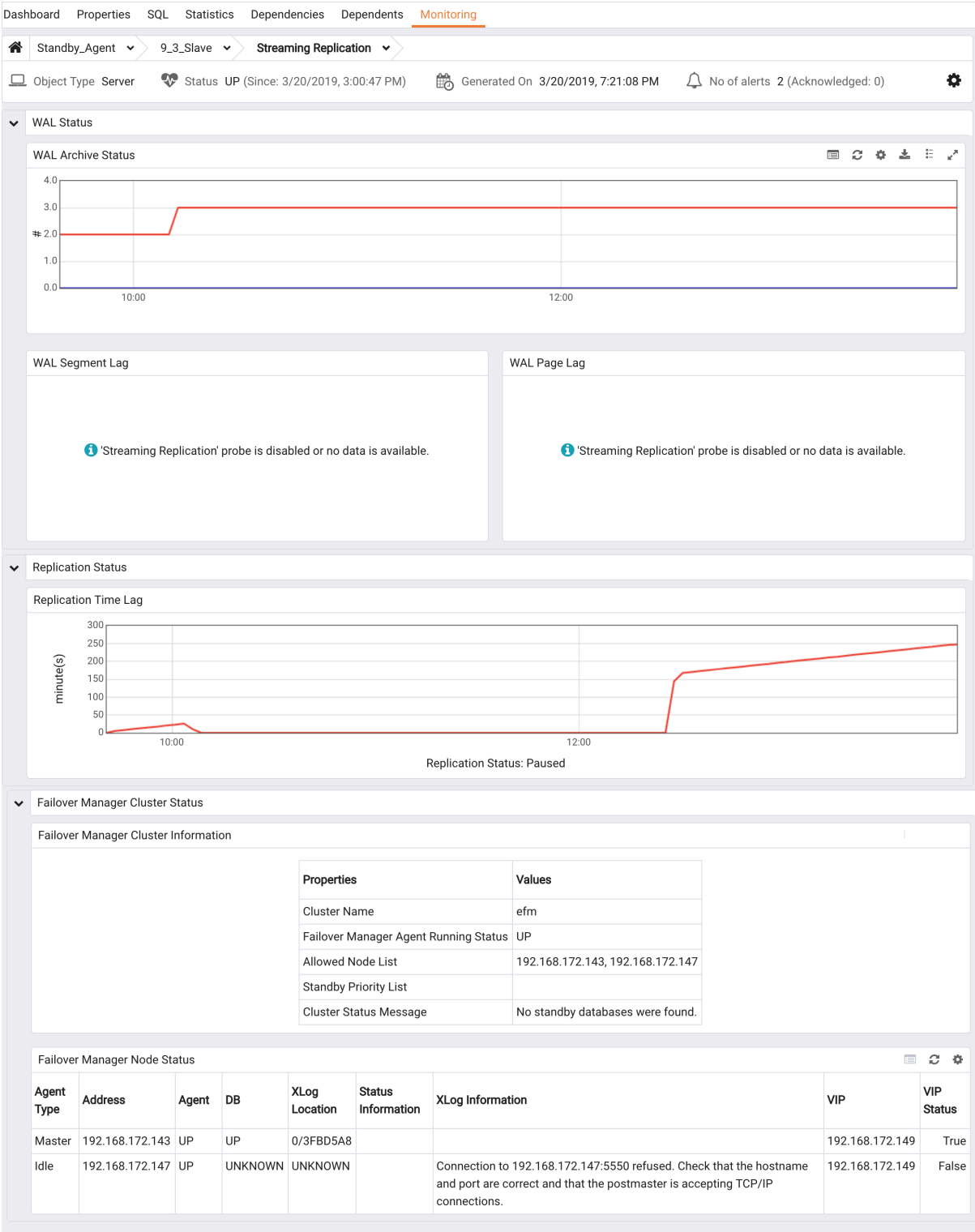


図 1: The Streaming Replication dashboard (Primary node)



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

第12章 Controlling the Failover Manager service

</ div>

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

- Failover Manager サービスで使用されるプロパティを含む **efm.properties** という名前付けの構成ファイル。レプリケーションシナリオの各ノードには、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** という名前付け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。

単一のノードでマルチプルのクラスターを実行している場合、クラスター固有の名前で構成ファイルを手動で作成し、対応するクラスターのサービススクリプトを変更する必要があります。

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。

</ div>

12.1 RHEL / CentOS 7.x および RHEL / CentOS 8.x で **systemctl** ユーティリティを使用する

RHEL / CentOS 7.x および RHEL / CentOS 8.x では、Failover Manager は、**/usr/lib/systemd/system** にある（デフォルトで）**edb-efm-4.<x>.service** 名前付けの Linux サービスとして実行されます。フェールオーバーマネージャによって監視される各データベースクラスターは、レプリケーションクラスターの各ノードでサービスのコピーを実行します。

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

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

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

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

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

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

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

```
[root@ONE ~]]> systemctl status edb-efm-4.4
* edb-efm-4.4.service - EnterpriseDB Failover Manager 4.4
*   Loaded: loaded (/usr/lib/systemd/system/edb-efm-4.4.service; disabled; vendor
→preset: disabled)
*   Active: active (running) since Wed 2013-02-14 14:02:16 EST; 4s ago
*   Process: 58125 ExecStart=/bin/bash -c /usr/edb/edb-efm-4.4/bin/runefm.sh start $
→{CLUSTER} (code=exited, status=0/SUCCESS)
Main PID: 58180 (java)
*   CGroup: /system.slice/edb-efm-4.4.service
*           └─ 58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/
→jre/bin/java -cp /usr/edb/edb-efm-4.4/lib/EFM-4.4.0.jar -Xmx128m...
```

第13章 Controlling logging

</ div>

フェールオーバーマネージャーは、エージェントごとに1つのログファイルを書き込み、`/var/log/<cluster_name>-4`のエージェントごとに1つのスタートアップログを保存します。

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

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

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

- TRACE
- DEBUG
- INFO
- WARN
- ERROR

例、`efm.loglevel` パラメータを `WARN` に設定すると、Failover Manager 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。

デフォルトでは、フェールオーバーマネージャーのログファイルは毎日ローテーションされ、圧縮され、1週間保存されます。ログローテーションファイル (`/etc/logrotate.d/efm-4.<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
```

</ div>

13.1 syslog ログファイルエントリの有効化

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

syslog への接続を許可するには、`/etc/rsyslog.conf` ファイルを編集し、使用するプロトコルのコメントを解除します。また、プロトコルに関連付けられた `UDPServerRun` または `TCPServerRun` エントリーに、ログエントリの送信先のポート番号が含まれていることを確認してください。例、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の `rsyslog.conf` ファイルを変更した後、フェールオーバーマネージャーのプロパティを変更してロギングを有効にします。エディタの選択を [modify the properties file](#) `/etc/edb/efm-4.<x>/efm.properties.in` に使用して、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
```

また、システムにも必要です。パラメータを使用してプロトコルタイプ（UDP または TCP）を指定し、`syslog.port` パラメータを使用して syslog ホストのリスナポートを指定します。`syslog.facility` 値を、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` と `LOCAL7` の間の値を使用します。

```
# 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
```

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

```
syslog man
```

第14章 Notifications

</div>

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

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

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

通知の本文には、通知をトリガーしたイベントとクラスターの現在の状態に関する詳細が含まれます。例：

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

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

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

- **INFO** は、エージェントに関する情報メッセージを示しており、マニュアル介入は必要ありません（例、フェールオーバーマネージャーの起動または停止）。*List of INFO level notifications* を参照
- **WARNING** は、管理者がシステムをチェックする必要があるイベントが発生したことを示します（例、フェールオーバーが発生した）。*List of WARNING level notifications* を参照
- **SEVERE** は、重大なイベントが発生し、管理者の即時の注意が必要であることを示します（例、フェールオーバーが試行されましたが完了できません）。*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。重大度レベルが **SEVERE** の通知はすぐにユーザの注意を必要としますが、重大度レベルが **INFO** の通知はユーザのアクションを必要としないクラスターに関する操作情報に注意を喚起します。通知の重大度レベルはロギングレベルとは関係ありません。構成ファイルで指定されたログレベルの詳細に関係なく、すべての通知が送信されます。

``notification.level <04_configuring_efm/01_cluster_properties/#notification_level>`` プロパティを使用して、通知をトリガーする最小の重大度レベルを指定できます。

!!! Note * 管理用メールアドレスへの通知の送信に加えて、すべての通知はエージェントログファイル (/var/log/efm-4.<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 レベル通知を引き起こします：

</div>

Subject	Description
Executed fencing script	実行されたフェンシングスクリプト* script_name 結果： script_results *
Executed post-promotion script	実行されたプロモーション後のスクリプト* script_name 結果： script_results *
Executed remote pre-promotion script	実行されたリモート事前プロモーションスクリプト* script_name 結果： script_results *
Executed remote post-promotion script	実行されたリモートプロモーション後スクリプト* script_name 結果： script_results *
Executed post-database failure script	実行されたデータベース障害後のスクリプト* script_name 結果： script_results *
Executed primary isolation script	実行されたプライマリ分離スクリプト* script_name 結果： script_results *
Witness agent running on node_address for cluster cluster_name	監視エージェントが実行されています。
Primary agent running on node_address for cluster cluster_name	プライマリエージェントが実行されており、データベースの状態が監視されています。
Standby agent running on node_address for cluster cluster_name	スタンバイエージェントが実行されており、データベースの状態が監視されています。
Idle agent running on node node_address for cluster cluster_name	アイドル状態のエージェントが実行されています。ローカルデータベースを起動した後、エージェントを再開できます。
Assigning VIP to node node_address	VIP * VIP_address をノード node_address に割り当てます結果： script_results *
Releasing VIP from node node_address	ノード* node_address から VIP * VIP_address *を解放する結果： script_results *
Starting auto resume check for cluster cluster_name	このノードのエージェントは、* auto.resume.period *秒ごとにチェックして、障害が発生したデータベースのモニタリングを再開できるかどうかを確認します。この間にクラスターを確認し、データベースが再度起動されない場合はエージェントを停止します。詳細については、エージェントログを参照してください。
Executed agent resumed script	実行されたエージェントはスクリプトを再開しました script_name 結果： script_results
WAL logs backed up during promotion	新しいプライマリに従うようにこのスタンバイを再構成すると、pg_xlog または pg_wal の内容が pgdata ディレクトリにバックアップされました。ディスクスペースをフリーするのに都合がよい場合は、このバックアップを削除してください。

以下のこの表にリストされている条件は、警告レベルの通知をトリガーします。

</ div>

Subject	Description	Notes
Witness agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	監視エージェントが終了しました。	
Primary agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。	
Cluster <i>cluster_name</i> notified that primary agent has left	プライマリエージェントが再起動されるまで、クラスタのフェールオーバーは無効になります。	
Standby agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。	
Agent exited during promotion on <i>node_address</i> for cluster <i>cluster_name</i>	データベースの状態は監視されていません。	
Agent exited on <i>node_address</i> for cluster <i>cluster_name</i>	エージェントが終了しました。これは、アイドル状態のエージェントによって生成されます。	
Agent exited for cluster <i>cluster_name</i>	エージェントが終了しました。この通知は通常、スタートアップ中にエージェントがスタートアップが完了する前に終了したときに生成されます。	
Virtual IP address assigned to non-primary node	仮想 IP アドレスが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、FailoverManager は VIP をリリースします。VIP がプライマリノードに割り当てられていることを確認し、割り当てられていない場合は手動でアドレスを再割り当てします。	
Virtual IP address not assigned to primary node.	仮想 IP アドレスがプライマリノードに割り当てられていないようです。フェールオーバーマネージャーは VIP の再取得を試みます。	
No standby agent in cluster for cluster <i>cluster_name</i>	cluster_name * のスタンバイがクラスターを離れました。	

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Standby agent failed for cluster <i>cluster name</i>	cluster_name * のスタンバイエージェントがクラスターを離れましたが、コーディネーターはスタンバイデータベースがまだ実行中であることを検出しました。	
Standby database failed for cluster <i>cluster name</i>	スタンバイエージェントは、データベースに障害が発生したことを通知しました。他のノードもスタンバイデータベースに到達できません。	
Standby agent cannot reach database for cluster <i>cluster name</i>	スタンバイエージェントはデータベース障害を通知しましたが、他のノードはスタンバイデータベースがまだ実行中であることを検出しました。	
Cluster <i>cluster name</i> has dropped below three nodes	完全なフェイルオーバー保護には、少なくとも3つのノードが必要です。監視ノードまたはエージェントノードをクラスターに追加します。	
Subset of cluster <i>cluster name</i> disconnected from primary	このノードは、クラスター*クラスター名前*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パートであるため、フェイルオーバーは試行されません。可視されている現在のノードは次のとおりです：*node_address*。	
Promotion has started on cluster <i>cluster name</i> .	スタンバイの昇格がクラスター*cluster_name*で開始されました。	
Witness failure for cluster <i>cluster_name</i>	node_address * で実行されている Witness がクラスターを離れました。	

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Idle agent failure for cluster <i>cluster_name</i> .	• <i>node_address</i> *で実行されているアイドルエージェントがクラスターを離れました。	
One or more nodes isolated from network for cluster <i>cluster_name</i>	このノードはネットワークから分離されているようです。クラスターに表示される他のメンバーは次のとおりです。 <i>node_name</i>	
Node no longer isolated from network for cluster <i>cluster_name</i> .	このノードはネットワークから分離されなくなりました。	
Failover Manager tried to promote, but primary DB is still running	Failover Manager 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*アドレス*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 Failover Manager 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、フェイルオーバー保護はありません。	
Primary agent missing for cluster <i>cluster_name</i>	プライマリエージェントは以前にクラスターを離れました。プライマリエージェントがクラスターに参加するまで、フェイルオーバー保護はありません。	Available in Failover Manager 4.2 and later.
Standby agent started to promote, but primary has rejoined.	スタンバイ Failover Manager エージェントは自分自身のプロモートを開始しましたが、プライマリエージェントがクラスターに再参加していることがわかりました。フェイルオーバーは発生していません。	
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プロモートしようとしたましたが、プライマリ DB が* <i>node_address</i> *でまだ実行されているかどうかを検出できませんでした。フェイルオーバーは発生していません。	Description applicable for Failover Manager 4.2 and later.

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
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プロモートしようとしたが、プライマリ DB がまだ* node_address *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 Failover Manager 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。	Description applicable for Failover Manager 4.1 and earlier.
Standby agent tried to promote, but VIP appears to still be assigned	スタンバイ Failover Manager エージェントは自分自身をプロモートさせようとしたが、仮想 IP アドレス (VIP_address) がまだ別のノードに割り当てられているように見えるため、昇格できませんでした。このような状況でプロモートすると、データが破損する可能性があります。フェイルオーバーは発生していません。	
Standby agent tried to promote, but appears to be orphaned	スタンバイ Failover Manager エージェントは自分自身をプロモートしようとしたが、既知のサーバー (server_address) に到達できなかったため、プロモートできませんでした。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェイルオーバーは発生していません。	
Potential manual failover required on cluster <i>cluster_name</i> .	クラスター* cluster_name *で潜在的なフェイルオーバーシチュエーションが検出されました。このクラスターでは自動フェイルオーバーが無効になっているため、マニュアル介入が必要です。	
Failover has completed on cluster <i>cluster_name</i>	クラスター* cluster_name *でフェイルオーバーが完了しました。	

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Lock file for cluster <i>cluster_name</i> has been removed	クラスター* <i>cluster_name</i> のロックファイルは、ノード <i>node_address</i> *の <i>path_name</i> から削除されました。このロックは、マルチプルのエージェントが同じノード上の同じクラスターをモニタリングするのを防ぎます。クラスターの別のエージェントを誤って起動しないように、このファイルを復元してください。	
A recovery file for cluster <i>cluster_name</i> has been found on primary node	クラスター* <i>cluster_name</i> * <i>hwas</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_name</i> にあります。このノードで DB をリスタートしようとする、問題が発生する可能性があります。	
recovery_target_timeline is not set to latest in recovery settings	リカバリ設定で <i>recovery_target_timeline</i> パラメータが latest に設定されていません。スタンバイサーバは、新しいプライマリが昇格されたときに発生するタイムラインの変更に従うことができません。	
Promotion has not occurred for cluster <i>cluster_name</i>	昇格が試行されましたが、昇格されているノード <i>ip_address</i> がすでに存在します。	
Standby will not be reconfigured after failover in cluster cluster_name	このノードの 'auto.reconfigure' プロパティは false に設定されているため、昇格後に新しいプライマリノードに従うように再構成されません。	Subject and Description applicable to Failover Manager 4.2 and later.
Standby not reconfigured after failover in cluster cluster_name	このノードの 'auto.reconfigure' プロパティは false に設定されました。フェイルオーバー後に新しいプライマリノードに従うようにノードが再構成されていません。	Subject and Description applicable to Failover Manager 4.1 and earlier.
Could not resume replay for standby <i>standby_id</i> .	スタンバイのリプレイを再開できませんでした。手動による介入が必要になる場合があります。エラー: <i>error_message</i> 。	

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Possible problem with database timeout values	`remote.timeout` の値 (value) が `local.timeout` の値 (value) よりも高くなっています。ローカルデータベースの応答に時間がかかりすぎる場合、他のエージェントは接続できますが、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見なす場合があります。これによってフェイルオーバーが発生することはありませんが、ローカルエージェントがモニタリングを停止し、フェイルオーバー保護が失われる可能性があります。	
No standbys available for promotion in cluster <i>cluster_name</i>	クラスター内の現在のスタンバイノードの数は、最小数: <i>number</i> に減少しました。別のスタンバイノードを追加するか、昇格可能にしない限り、フェイルオーバーは発生しません。	
No promotable standby for cluster <i>cluster_name</i>	クラスター内の現在のフェイルオーバー優先度リストは空です。クラスター* <i>cluster_name</i> *の唯一の昇格可能なスタンバイを削除しました。別の昇格可能なスタンバイノードを追加するか、フェイルオーバー優先度リストに追加して昇格しない限り、フェイルオーバーは発生しません。	
Synchronous replication has been reconfigured for cluster <i>cluster_name</i>	クラスター内の同期的スタンバイノードの数が数を下回りました。プライマリの同期的スタンバイ名は、新しい <i>synchronous_standby_names</i> 値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的スタンバイノードの数がカウントを下回りました。プライマリは同期レプリケーションモードから外されました。	

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Could not reload database configuration.	データベース構成をリロードできませんでした。手動による介入が必要です。エラー：error_message。	
Custom monitor timeout for cluster <i>cluster_name</i>	次のカスタムモニタリングスクリプトがタイムアウトしました：script_name	
Custom monitor 'safe mode' failure for cluster <i>cluster_name</i>	次のカスタムモニタスクリプトは失敗しましたが、*モード*で実行されています：script_name。 出力：script_results	
primary.shutdown.as.failure set to true for primary node	プライマリ。このクラスターのシャットダウン.as.failure プロパティは true に設定されています。クラスタ全体を停止せずにプライマリエージェントを停止すると、クラスタの残りの部分によって、即時のプライマリエージェント障害として扱われます。プライマリデータベースのメンテナンスが必要な場合は、プライマリエージェントをシャットダウンし、フェイルオーバーが発生しないという残りのノードからの通知を待ちます。	
Primary_or_Standby cannot ping local database for cluster <i>cluster_name</i>	Primary_or_Standby エージェントは、* node_address *で実行されているローカルデータベースに到達できなくなりました。他のノードはデータベースにリモートでアクセスできるため、エージェントは IDLE になり、データベースのモニタリングを再開しようとします。	Subject and Description applicable to Failover Manager 4.1 and later.
PPrimary cannot ping local database for cluster <i>cluster_name</i>	プライマリエージェントは、node_address で実行されているローカルデータベースに到達できなくなります。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、プライマリは IDLE になり、データベースのモニタリングを再開しようとします。	Subject and Description applicable to Failover Manager 4.0.

次のページに続く

表 2 – 前のページからの続き

Subject	Description	Notes
Standby cannot resume monitoring local database for cluster <i>cluster_name</i>	スタンバイエージェントは、* node_address *で実行されている ローカルデータベースに到達で きなくなります。他のノードは データベースにリモートでアク セスできます。スタンバイエー ジェントは、`resume` コマンドを 実行してデータベースのモニタ リングを再開するまで IDLE の ままです。	Available in Failover Man- ager 4.1 and later.
Primary agent left the cluster and node detached from load balancer	スタンバイエージェントは、* node_address *で実行されている ローカルデータベースに到達で きなくなります。他のノードは データベースにリモートでアク セスできます。スタンバイエー ジェントは、`resume` コマンドを 実行してデータベースのモニタ リングを再開するまで IDLE の ままです。	Available in Failover Man- ager 4.2 and later.
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 num_seconds 秒後まで、新しい プライマリに従うように再構成 されません。場合によっては、 マニュアルの介入なしに新しい プライマリを追跡できない場合 があります。	Available in Failover Man- ager 4.2 and later.

</div>

この表にリストされている条件は重大な通知を引き起こします：

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

次のページに続く

表 3 – 前のページからの続き

Subject	Description	Notes
Unable to connect to DB on <i>node_address</i>	最大接続制限に達しました。	
Unable to connect to DB on <i>node_address</i>	db のパスワードが無効です。 ユーザ= <i>user_name</i> 。	
Unable to connect to DB on <i>node_address</i>	権限指定が無効です。	
Primary cannot resume monitoring local database for cluster <i>cluster_name</i>	プライマリエージェントは、* <i>node_address</i> *で実行されている ローカルデータベースに到達で 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、プライマリは VIP をリリースしたり、`recov- ery.conf` ファイルを作成したりし ません。プライマリエージェント は、`resume` コマンドを実行して データベースのモニタリングを 再開するまで 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> フェイル オーバーは発生していません。	
Post-promotion script failed	プロモーション後のスクリプト <i>script_name</i> を正常に実行できま せんでした。終了値: <i>exit_code</i> 結 果: <i>script_results</i>	
Remote post-promotion script failed	リモートのプロモーション 後 スクリプト <i>script_name</i> を正常に実行できませんで した 終了値: <i>exit_code</i> 結 果: <i>script_results</i> Node: * <i>node_address</i> *	
Remote pre-promotion script failed	リモート事前プロモーション スクリプト <i>script_name</i> は正常に実行できませんで した 終了値: <i>exit_code</i> 結 果: <i>script_results</i> Node: * <i>node_address</i> *	
Post-database failure script error	データベース障害後のスクリプ 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>	

次のページに続く

表 3 – 前のページからの続き

Subject	Description	Notes
Agent resumed script error	エージェントが再開したスクリプト <code>script_name</code> は正常に実行できませんでした。結果： <code>script_results</code>	
Primary isolation script failed	一次分離スクリプト <code>script_name</code> は正常に実行できませんでした。終了値： <code>exit_code</code> 結果： <code>script_results</code>	
Could not promote standby	ノードでプロモートコマンドが失敗しました。スタンバイをプロモートできませんでした。エラーの詳細： <code>error_details</code>	
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> *にリカバリ.conf ファイルを作成中にエラーが発生しました。プロモーションは継続されてい保証が、古いプライマリノードを再起動できないようにマニュアルで介入する必要があります。エラーの詳細： <code>message_details</code>	
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： <code>error_details</code>	
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ています。クラスターは、 <code>ip_address</code> のプライマリエージェントに、フェイルオーバーマネージャクラスターの残りの部分がスタンバイをプロモートするときに2つのプライマリを防ぐために、プライマリデータベースをフェンスオフするように指示しています。	
Isolated primary database shutdown.	分離されたプライマリデータベースは、フェイルオーバーマネージャによってシャットダウンされました。	

次のページに続く

表 3 – 前のページからの続き

Subject	Description	Notes
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、スタンバイが昇格され、クラスター内のこのノードに再参加しました。このノードは、複数のプライマリデータベースを回避するために自分自身を分離しています。	
Could not assign VIP to node <i>node_address</i>	フェイルオーバーマネージャは、何らかの理由で VIP アドレスを割り当てることができませんでした。	
primary_or_standby database failure for cluster <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アクセスしようとしてタイムアウトしました。タイムアウト後、エージェントはデータベースに正常に ping を実行し、モニタリングを再開しました。ただし、データベースまたはエージェントの障害を防ぐために、ノードをチェックして正常に実行されていることを確認してください。	
Resume timed out for cluster <i>cluster_name</i>	このエージェントは、ローカルデータベースを再構成して再起動した後、モニタリングを再開できませんでした。詳細については、エージェントログを参照してください。	

次のページに続く

表 3 – 前のページからの続き

Subject	Description	Notes
Internal state mismatch for cluster <i>cluster_name</i>	Failover Manager クラスターの内部状態が、クラスターメンバーの実際の状態とマッチしませんでした。これはまれであり、ノードがクラスターに参加したり、ノードの状態を変更したりするタイミングの問題が原因である可能性があります。問題は解決する必要がありますが、クラスターのステータスも確認して確認してください。不一致の詳細は、エージェントログファイルに記載されています。	
Failover has not occurred	エージェントは、プライマリデータベースがクラスター* <i>cluster_name</i> *で使えなくなったことを検出しましたが、フェイルオーバーに使用できるスタンバイノードがありません。	
Failover has not occurred	エージェントは、プライマリデータベースがクラスター* <i>cluster_name</i> *で使えなくなったことを検出しましたが、フェイルオーバーに使用できるスタンバイノードが十分にありません。	
Database in wrong state on <i>node_address</i>	スタンバイエージェントは、ローカルデータベースがリカバリ中ではなくなったことを検出しました。エージェントは IDLE になります。手動による介入が必要です。	
Database in wrong state on <i>node_address</i>	プライマリエージェントは、ローカルデータベースがリカバリ中であることを検出しました。エージェントは IDLE になります。手動による介入が必要です。	
Database 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、フェイルオーバーが正しく機能しない可能性があります。	

次のページに続く

表 3 – 前のページからの続き

Subject	Description	Notes
Standby custom monitor failure for cluster <i>cluster_name</i>	次のカスタムモニタスクリプトは、スタンバイノードで失敗しました。エージェントはローカルデータベースのモニタリングを停止します。スクリプトの場所: <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>	次のカスタムモニタスクリプトは、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のプロモートを試みます。スクリプトの場所: <i>script_name</i> スクリプト出力: <i>script_results</i>	
Loopback address set for ping.server.ip	ループバックアドレスはピングプロパティに設定されます。この設定はネットワーク分離の検出に干渉する可能性があるため、変更する必要があります。	
Load balancer attach script error	ロードバランサー接続スクリプト <i>script_name</i> を正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i>	
Load balancer detach script error	ロードバランサーのデタッチスクリプト <i>script_name</i> を正常に実行できませんでした。終了値: <i>exit_code</i> 結果: <i>script_results</i>	
Pgpool attach node error	フェールオーバーマネージャーは pgpool ノードの接続に失敗しました。終了値: <i>exit_code</i> 。結果: <i>script_results</i>	Available in Failover Manager 4.1 and later.
Pgpool detach node error	フェールオーバーマネージャーは、pgpool ノードのデタッチに失敗しました。終了値: <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的スタンバイノードの数がカウントダウンしました。プライマリでのすべての書き込みクエリは、十分な同期的スタンバイノードが追加されるまでブロックされます。	

第15章 Supported failover and failure scenarios

</ div>

フェールオーバーマネージャーは、フェイルオーバーにつながる可能性のある障害についてクラスターを監視します。

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

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

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

フェールオーバーマネージャーは、フェールオーバーマネージャーでフェイルオーバー条件をモニタおよび検出したいが、スタンバイへの自動フェイルオーバーを実行しない場合に、**自動フェールオーバーなし**モードもサポートします。このモードでは、フェイルオーバー条件が満たされると管理者に通知が送信されます。自動フェイルオーバーを無効にするには、`auto.failover` パラメータを `false` に設定して、クラスタープロパティファイルを変更します。

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メリットない状況について、管理者に警告します。

</ div>

15.1 プライマリデータベースがダウンしています

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

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

エージェントが仮想 IP アドレスまたはデータベースサーバに到達できない場合、フェールオーバーマネージャーはフェイルオーバープロセスを開始します。最新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。追加のスタンバイノードは、`auto.reconfigure` が

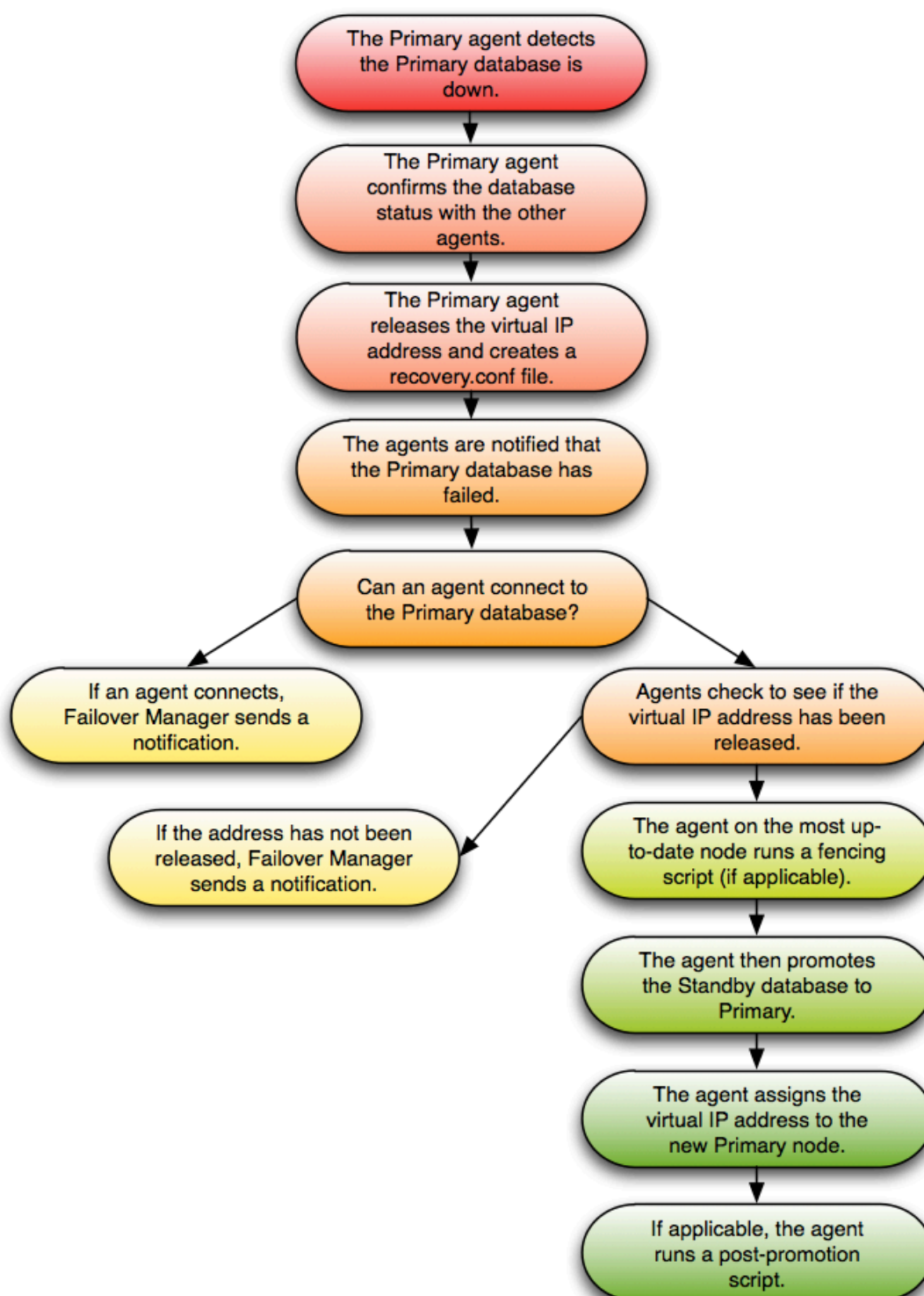


図 1: Confirming the failure of the primary database.

`false` に設定されていない限り、新しいプライマリから複製するように構成されます。該当する場合、エージェントはポストプロモーションスクリプトを実行します。

15.2 ノードをクラスターに戻す

クラスター全体を再起動せずにこのシナリオから回復するには：

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

15.3 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、簡単にノードをプライマリのロールに結果ことができます。

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

!!! Note * Failover Manager は、障害が発生したプライマリデータベースを再構築してスタンバイになりません。再構築する前に、プライマリが失敗した理由を特定し、すべてのデータが新しいプライマリで利用可能であることを保証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れば、古いデータディレクトリを削除して、サーバーを復元できます。詳細については、PostgreSQL の文書を参照してください。場合によっては、を使用してサーバーを復元することもできます。

</ div>

15.4 スタンバイデータベースがダウンしています

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

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結果ます。

</ div>

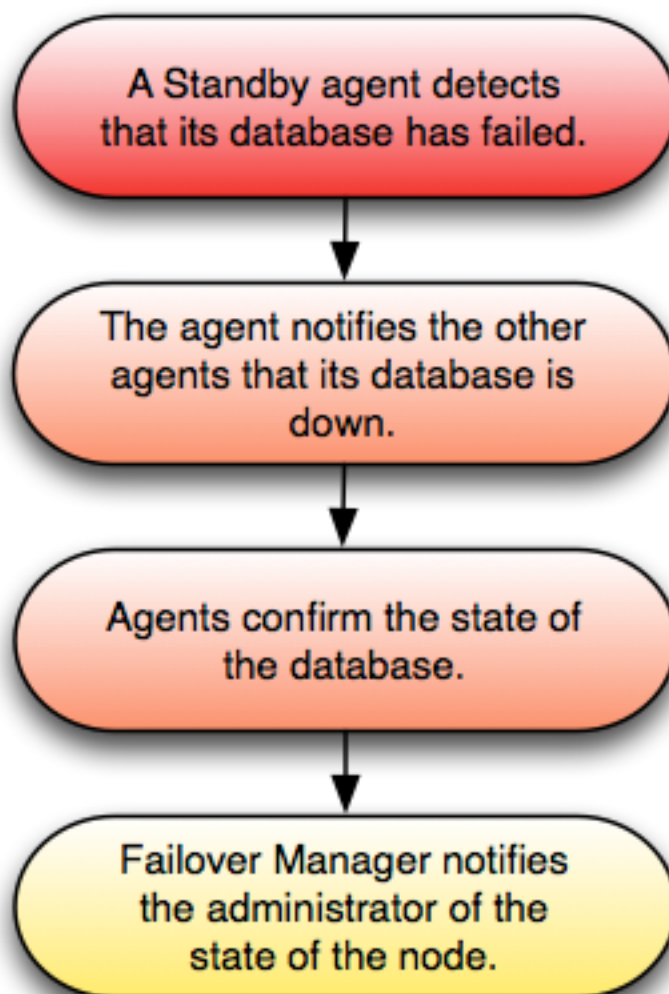


図 2: Confirming the failure of a standby database.

15.5 プライマリエージェントが終了するか、ノードに障害が発生する

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

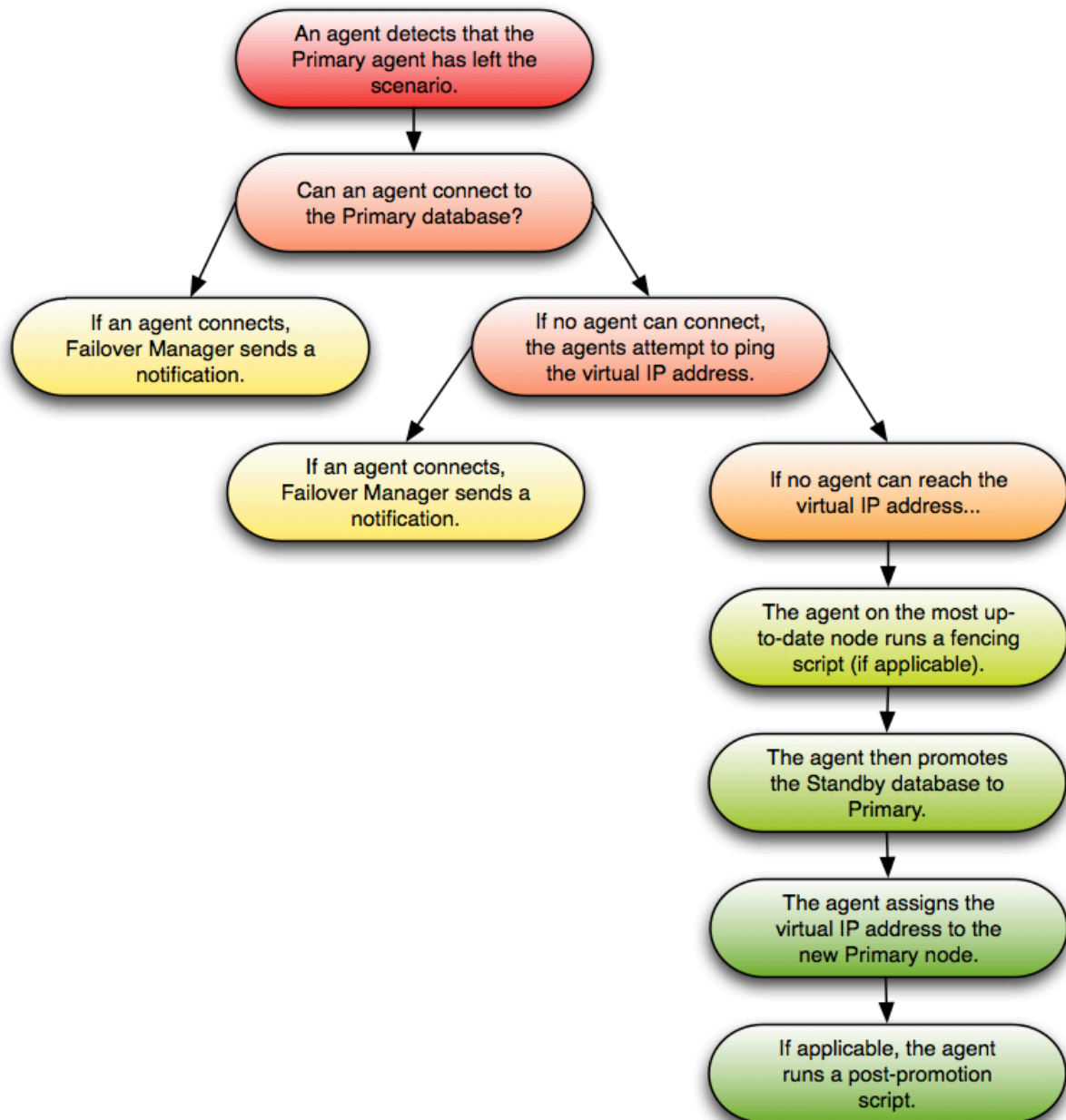


図 3: Confirming the failure of the primary agent.

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、エージェントはプライマリエージェントの障害に関する通知を送信します。接続できるエージェントがない場合、エージェントは仮想 IP アドレス（該当する場合）に対してピングを試行し、解放されたかどうかを判断します。

エージェントが仮想 IP アドレスまたはデータベースサーバに到達できない場合、フェールオーバーマネージャーはフェイルオーバープロセスを開始します。最新ノードのスタンバイエージェントは、フェンシン

グスクリプト（該当する場合）を実行し、スタンバイデータベースをプライマリデータベースに昇格させ、仮想 IP アドレスをスタンバイノードに割り当てます。該当する場合、エージェントはポストプロモーションスクリプトを実行します。追加のスタンバイノードは、`auto.reconfigure` が `false` に設定されていない限り、新しいプライマリから複製するように構成されます。

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同じ手順を実行します。

クラスター全体を再起動せずにこのシナリオから回復するには：

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

!!! Note: * エージェントを停止しても、エージェントに障害が発生したことをクラスターにシグナルしません。

プライマリフェールオーバーマネージャープロセスが失敗した場合、エージェントが再起動されるまでフェイルオーバー保護はありません。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セットアップし、プライマリエージェントが終了したときにフェイルオーバーが発生させることができます。詳細については、[Configuring for Eager Failover](#) を参照してください。

</div>

15.6 スタンバイエージェントが終了するか、ノードに障害が発生する

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、クラスターに接続されなくなったことを検出します。

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

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イルオーバー保護はありません。プライマリデータベースに障害が発生した場合、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、フェイルオーバーを続行できます。

</div>

15.7 専用の監視エージェントが終了/ノードが失敗する

このシナリオでは、専用ミラーリング監視（データベースをホスティングしていないノード）が失敗した場合に実行されるアクションを詳しく説明します。

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管理者に監視の状態を通知します。

!!! Note * ミラーリング監視が失敗し、クラスターに 2 つのノードしかない場合、フェイルオーバー保護はありません。スタンバイノードは、プライマリが失敗したか切断されたかを知ることができません。2 ノー

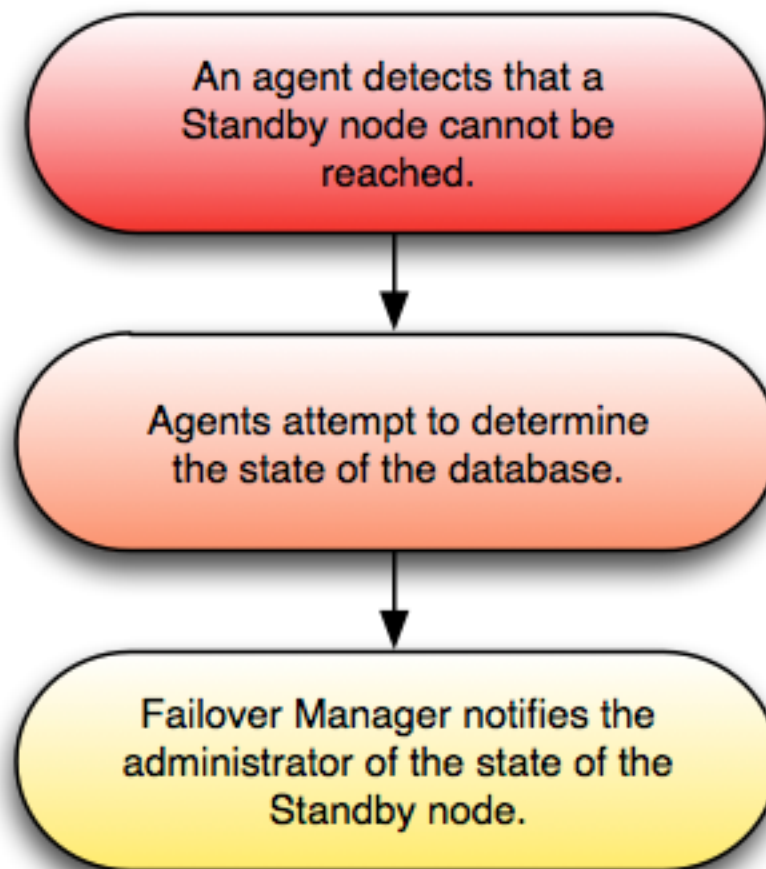


図 4: Failure of standby agent.

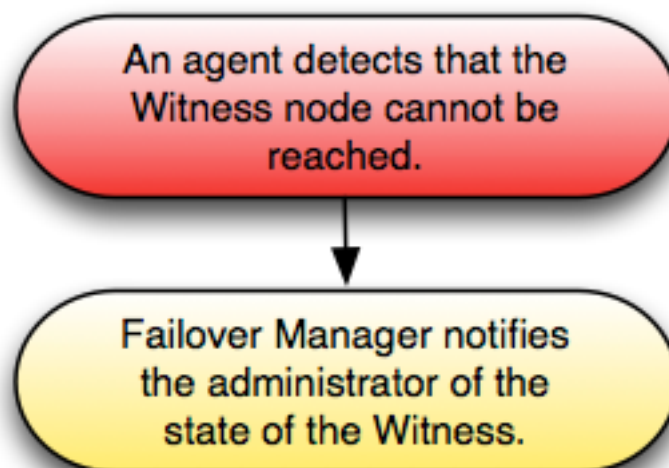


図 5: Confirming the failure of a dedicated witness.

ドクラスタでは、プライマリデータベースに障害が発生しても、ノードが接続されたままの場合、フェイルオーバーが発生します。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。

</div>

15.8 ノードはクラスターから分離されます

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

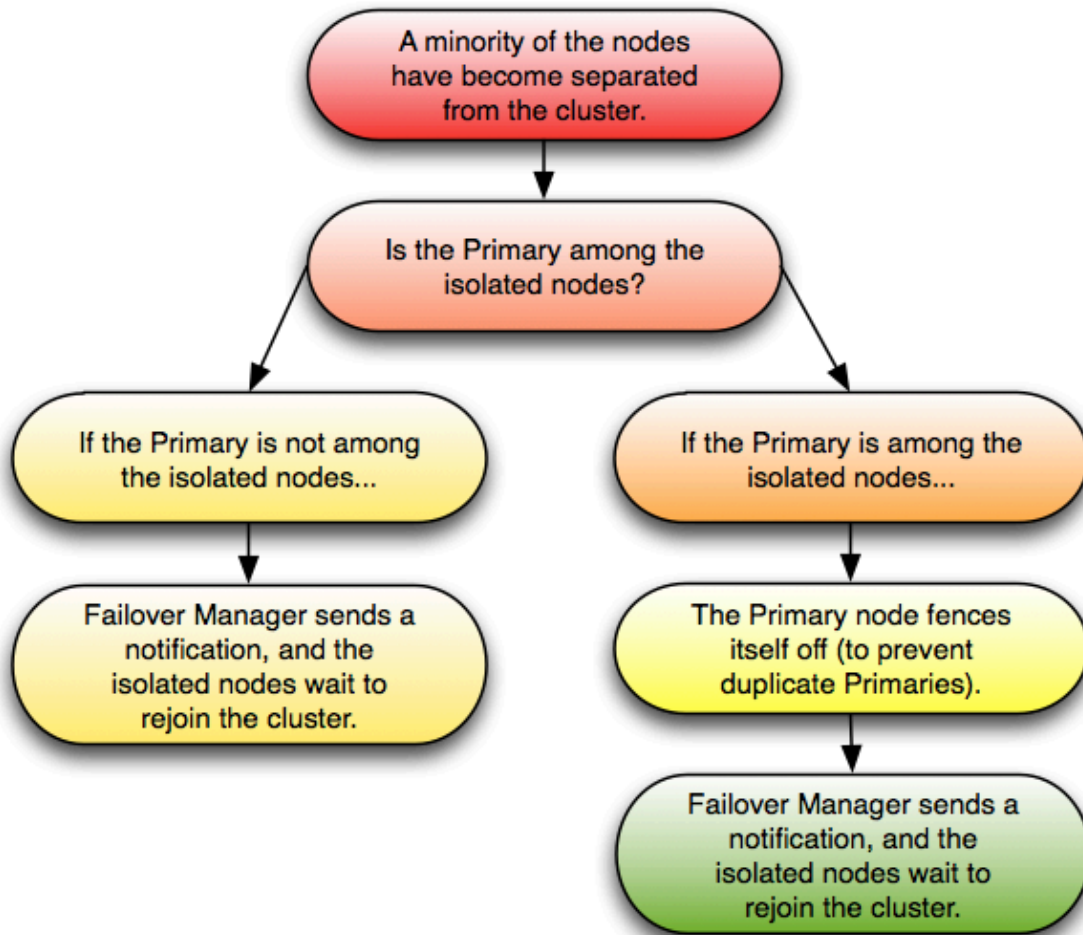


図 6: If members of the cluster become isolated.

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

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

第16章 Troubleshooting

</div>

16.1 認証ファイルが見つかりません。ローカルエージェントは実行されていますか？

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?
```

16.2 このコマンドを実行する権限がありません。ユーザー '`<os user>`' `efm` グループのメンバではありません。

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

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

16.3 通知; 予期しないエラーメッセージ

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

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

128 MB より小さい割り当てで実行している場合は、値を増やして Failover Manager エージェントをリスタートします。

16.4 OpenJDK バージョンの確認

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

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

第17章 Creating a Failover Manager cluster

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

このクイックスタートガイドでは、テスト環境でのフェールオーバーマネージャークラスターの構成について説明します。稼動展開用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前に、[Failover Manager User Guide](#) を読んで理解してください。

チュートリアルを開始する前に、これらの基本的なインストールおよび構成手順を実行します。

- 1つのプライマリノードと1つまたは2つのスタンバイノードにデータベースサーバをインストールして初期化します。Advanced Server のインストールについては、[EDB Postgres Advanced Server](#) の文書を参照してください。

Postgres ストリーミングレプリケーションを構成し、プライマリノードとスタンバイノード間で実行する必要があります。ストリーミングレプリケーションの構成の詳細については、[PostgreSQL Streaming Replication documentation](#) を参照してください。

- フェールオーバーマネージャークラスターを各プライマリノードとスタンバイノードにインストールします。Advanced Server のインストール中に、各データベースホストで EnterpriseDB リポジトリを構成しました。EnterpriseDB 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クラスターをインストールできます。

```
* yum install edb-efm43
```

インストールプロセス中に、インストーラは、enterprisedb または postgres が所有するクラスターの Failover Manager サービスを制御するスクリプトを呼び出す特権を持つ `efm` という名前付けのユーザを作成します。次の例では、`efm` 名前付けのクラスターを作成します。

プライマリノード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 Failover Manager 構成ファイルを作成し、所有権を修正します。

```
* cd /etc/edb/efm-4.4
* cp efm.properties.in efm.properties
* 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-4.4/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言及されているプロパティは、Failover Manager クラスターの構成に必要な最小限のプロパティです。稼動システムを構成している場合は、[Failover Manager User Guide](#) でフェールオーバーマネージャーオプションの詳細を確認してください。

- すべてのクラスターノードで次のプロパティの値を指定します。

• |プロパティ|説明| |-----| |-----|

 | db.user |データベースユーザの名前。|| db.password.encrypted |データベースユーザの暗号化されたパスワード。|| db.port |データベースによって監視されるポート。|| db.database |データベースの名前。|| db.service.owner | data ディレクトリの所有者（通常は postgres または enterprisedb）。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。|| db.service.name |データベースサービスの名前（サーバーのリスタートに使用）。データベースがサービスとして実行されている場合にのみ必要です。|| db.bin | bin ディレクトリへのパス（pg_ctl の呼び出しに使用）。|| db.data.dir | EFM が recovery.conf ファイルまたは standby.signal ファイルを検索または作成する data ディレクトリ。|| user.email |メール通知を受信するメールアドレス（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）。|| bind.address |フェールオーバーマネージャーに使用するノードとポートのローカルアドレス。フォーマットは次のとおりです。bind.address=1.2.3.4:7800|| is.witness |ミラーリング監視ノードの true およびプライマリまたはスタンバイの場合は false || ping.server.ip |インターネットにアクセスできないネットワークで実行している場合は、ping.server.ip をネットワークで使用可能なアドレスに設定します。|| auto.allow.hosts |テストクラスターでは、true に設定してスタートアップを簡素化します。稼動での使用については、ガイドを参照してください。|| stable.nodes.file |テストクラスターでは、true に設定してスタートアップを簡素化します。稼動環境での使用については、ガイドを参照してください。|

2. **efm.nodes** を更新します。<cluster_name>.nodes ファイル（この例ではノード）は、スタートアップ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。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。少なくともメンバーシップコーディネーターの住所をリストに含めてください。例：

```
1.2.3.4:7800
```

```
1.2.3.5:7800
```

```
1.2.3.6:7800
```

- Failover Manager エージェントは、**efm.nodes** ファイルのアドレスを検証しません。エージェントは、ファイル内のアドレスの一部に到達できないことを予期しています（例、別のエージェントがまだ開始されていないなど）。

1. 他のノードを構成します。サンプルクラスターの他のノードの **efm.properties** および **efm.nodes** ファイル

ルを/etc/edb/efm-4.4 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。

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

1. フェールオーバーマネージャクラスターを起動します。任意のノードで、Failover Manager エージェントをスタートします。エージェントの名前付けは edb-efm-4.4 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。例、CentOS / RHEL 7.x または CentOS / RHEL 8.x ホストでは、次のコマンドを使用します。

```
* systemctl start edb-efm-4.4
```

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

```
* service edb-efm-4.4 start
```

1. エージェントの起動後、次のコマンドを実行して、シングルノードクラスターのステータスを確認します。他のノードのアドレスは、Allowed node host リストに表示されます。

```
* /usr/edb/efm-4.4/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、クラスターの状態を確認します。

- いずれかのエージェントがスタートに失敗した場合は、スタートアップログを参照して、問題の詳細を確認してください。

```
* cat /var/log/efm-4.4/startup-efm.log
```

17.1 スイッチオーバーを実行する

クラスタステータスの出力に、プライマリノードとスタンバイノードが同期していることが示されている場合、スイッチオーバーを実行できます。

```
* /usr/edb/efm-4.4/bin/efm promote efm -switchover
```

このコマンドは、スタンバイをプロモートし、プライマリデータベースをクラスター内の新しいスタンバイとして再構成します。元に戻すには、コマンドを再度実行しスイッチ。

17.2 オンラインヘルプにアクセスする

オンラインヘルプにすばやくアクセスするには、次を使用します。

```
* /usr/edb/efm-4.4/bin/efm --help
```