



EDB Failover Manager

リリース **3.10**

EnterpriseDB

2020 年 06 月 28 日

目次

第 1 章	新機能	3
第 2 章	フェールオーバーマネージャーの概要	5
2.1	前提条件	7
2.2	チュートリアル-シンプルなフェールオーバーマネージャークラスターの構成	10
第 3 章	フェールオーバーマネージャーのインストール	15
3.1	RedHat、CentOS、または OEL ホストへのフェールオーバーマネージャーのインストール	15
3.1.1	設置場所	17
3.2	Debian または Ubuntu ホストへのフェールオーバーマネージャーのインストール	18
3.3	SLES ホストへのフェールオーバーマネージャーのインストール	19
第 4 章	フェールオーバーマネージャーの設定	21
4.1	クラスタプロパティファイル	21
4.1.1	クラスタープロパティの指定	22
4.2	クラスターメンバーファイル	45
4.3	フェールオーバーマネージャーの権限の拡張	46
4.3.1	sudo を使用せずにフェールオーバーマネージャーを実行する	48
4.4	仮想 IP アドレスでのフェールオーバーマネージャーの使用	50
第 5 章	フェールオーバーマネージャーの使用	53
5.1	フェールオーバーマネージャークラスターの管理	53
5.1.1	FailoverManager クラスターの起動	54
5.1.2	クラスターへのノードの追加	54
5.1.3	スタンバイの優先度の変更	55
5.1.4	フェールオーバーマネージャーノードの昇格	56
5.1.5	FailoverManager エージェントの停止	57
5.1.6	フェールオーバーマネージャークラスターの停止	58
5.1.7	クラスターからのノードの削除	58
5.2	単一ノードでの複数のエージェントの実行	59
5.2.1	RHEL6.x または CentOS6.x	61
5.2.2	RHEL/CentOS7.x または RHEL/CentOS8.x	61
第 6 章	フェールオーバーマネージャークラスターの監視	63

6.1	クラスタステータスレポートの確認	63
6.2	PostgresEnterpriseManager によるストリーミングレプリケーションの監視	66
第 7 章	efm ユーティリティの使用	69
第 8 章	フェールオーバーマネージャサービスの制御	73
8.1	RHEL6.x および CentOS6.x でのサービスユーティリティの使用	74
8.2	RHEL/CentOS7.x および RHEL/CentOS8.x での systemctl ユーティリティの使用	75
第 9 章	ロギングの制御	77
9.1	Syslog ログファイルエントリの有効化	79
第 10 章	通知	81
第 11 章	サポートされているフェイルオーバーと失敗のシナリオ	93
11.1	マスターデータベースがダウンしています	94
11.2	スタンバイデータベースがダウンしています	96
11.3	マスターエージェントの終了またはノードの失敗	97
11.4	スタンバイエージェントの終了またはノードの失敗	99
11.5	専用監視エージェントの終了/ノードが失敗する	100
11.6	ノードがクラスターから分離される	101
第 12 章	既存のクラスターのアップグレード	103
12.1	FailoverManager のアンインストール	105
12.2	データベース更新の実行（マイナーバージョン）	106
第 13 章	トラブルシューティング	107
第 14 章	ストリーミングレプリケーションの設定	109
14.1	カスケード型レプリケーションの限定サポート	110
第 15 章	フェイルオーバーマネージャクラスターでの SSL 認証の設定	113
第 16 章	結論	115
索引		117

EDB フェイルオーバーマネージャー™

EDBPostgresVariable_Prod_Name (EFM) は EnterpriseDB の高可用性モジュールであり、マスターでソフトウェアまたはハードウェアの障害が発生した場合に Postgres マスターノードがスタンバイノードに自動的にフェイルオーバーできるようにします。

このガイドでは、**variable_prod_name**のインストール、設定、使用について説明します。**バージョン**。 **variable_prod_name**でサポートされているプラットフォームとバージョンについての情報**version**、EnterpriseDB の Web サイトにアクセスしてください:

<https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms#efm>

このドキュメントでは、PostgreSQL または EDBPostgresAdvancedServer データベースを意味する Postgres を使用しています。

第 1 章

新機能

EDBPostgreslvariable_prod_name1に以下の変更が加えられました。バージョン 3.10 を作成するには:

- Postgres の物理レプリケーションスロット機能のサポート
- NAT アドレスのサポート
- 再起動のためにスタンバイをずらす
- ローカルノードのステータスをチェックする新しいコマンドの導入
- スタンバイ昇格の trigger_file を pg_ctl ユーティリティに置き換えます

第 2 章

フェールオーバーマネージャーの概要

EDBPostgresIvariable_prod_nameI (EFM) クラスターは、ネットワーク上の次のホストにある FailoverManager プロセスで構成されています。

- マスターノード-マスターノードは、データベースクライアントにサービスを提供するプライマリデータベースサーバーです。
- 1 つ以上のスタンバイノード-スタンバイノードは、マスターノードに関連付けられたストリーミングレプリケーションサーバーです。
- ウィットネスノード-ウィットネスノードは、フェイルオーバーシナリオでのマスターまたはスタンバイのアサーションを確認します。クラスターに 3 つ以上のノードが含まれている場合、クラスターには専用の監視ノードは必要ありません。データベースホストである 3 番目のクラスターメンバーがない場合は、専用の監視ノードを追加できます。クラスターには複数の監視ノードが含まれる場合があります。

伝統的に、*クラスター*は複数のデータベースを管理する Postgres の単一のインスタンスです。このドキュメントでは、クラスターという用語はIvariable_prod_nameIを指します。集まる。Ivariable_prod_nameIクラスターは、マスターエージェント、1 つ以上のスタンバイエージェント、およびクラウドまたは従来のネットワーク上のサーバーに常駐し、JGroups ツールキットを使用して通信するオプションの監視エージェントで構成されます。

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は、アイドルモードで起動します。
- データベースが回復中であることが判明した場合、エージェントはスタンバイの役割を担います。
- データベースがリカバリ中でない場合、エージェントはマスターの役割を引き受けます。

フェイルオーバーが発生した場合、Ivariable_prod_nameI昇格したスタンバイがクラスター内で最新のスタンバイであることを確認しようとします。スタンバイノードがマスターノードと同期していない場合、データが失われる可能性があることに注意してください。

*JGroups<<http://www.jgroups.org/>>*はIvariable_prod_nameIを可能にするテクノロジーを提供しますメンバーノードが相互に通信し、ノードの障害を検出できるクラスターを作成します。

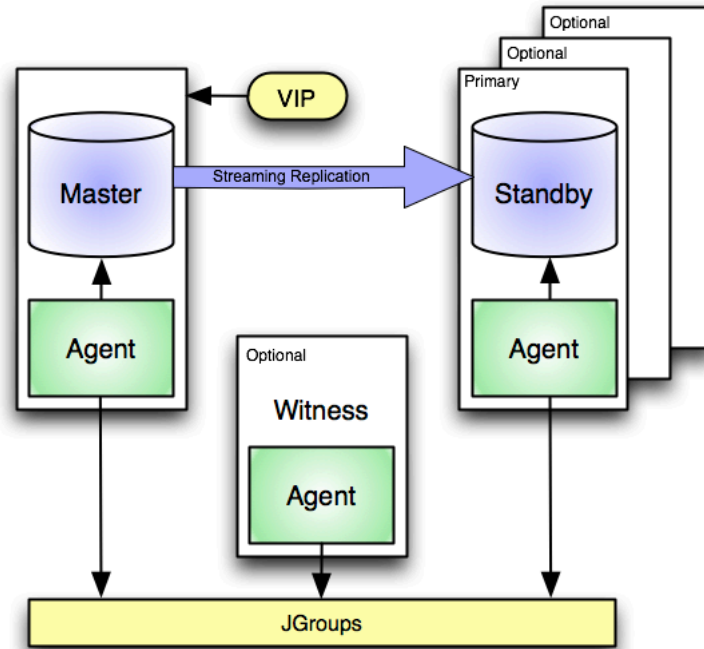


図1 仮想 IP アドレスを使用する EFM シナリオ。

上の図は、`variable_prod_name`を示しています。仮想 IP アドレスを使用するクラスター。:doc:`仮想 IP アドレスの代わりにロードバランサーを使用できます<using\_vip\_addresses>`独自の :doc:`script を提供する場合<cluster\_properties>`障害発生時にロードバランサーを再構成します。

2.1 前提条件

`variable_prod_name1`を設定する前にクラスターでは、以下に説明する前提条件を満たす必要があります。

Java 1.8 以降をインストールします

`variable_prod_name1`を使用する前に、まず 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。`variable_prod_name1`は OpenJDK でテストされており、そのバージョンの Java をインストールすることを強くお勧めします。‘Java のインストール手順’<<https://openjdk.java.net/install/>>‘__はプラットフォーム固有です。

SMTP サーバーを提供する

`variable_prod_name1`から通知を受け取ることができますユーザー定義の通知スクリプト、電子メール、またはその両方で指定されたとおり。

- 電子メール通知を使用している場合、SMTP サーバーが`variable_prod_name1`の各ノードで実行されている必要があります。シナリオ。
- `script.notification` プロパティに値を指定する場合、`user.email` フィールドを空白のままにすることができます。SMTP サーバーは必要ありません。

イベントが発生した場合、`variable_prod_name1`スクリプト（提供されている場合）を呼び出し、クラスターのプロパティファイルの `user.email` パラメータで指定されたメールアドレスに通知メールを送信します。SMTP サーバーの使用の詳細については、次の URL にアクセスしてください。

‘https://access.redhat.com/site/documentation/US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s1-email-mta.html’<https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s1-email-mta.html>‘__

ストリーミングレプリケーションの構成

`variable_prod_name1`マスターノードとスタンバイノード間で PostgreSQL ストリーミングレプリケーションを構成する必要があります。`variable_prod_name1`他のタイプのレプリケーションはサポートしていません。

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

データベースバージョン 12 では、`primary_conninfo` と `restore_command` のプロパティは、スイッチオーバー中に停止したマスターにコピーされます（`-sourcenode` オプションで特に指定されていない限り）。

pg_hba.conf ファイルを変更します

マスターノードとスタンバイノードの pg_hba.conf ファイルを変更し、クラスター内のすべてのノード間の通信を可能にするエントリを追加する必要があります。次の例は、マスターノードの pg_hba.conf ファイルに作成されるエントリを示しています。

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

どこ:

efm は有効なデータベースユーザーの名前を指定します。

fmdb は、efm ユーザーが接続できるデータベースの名前を指定します。

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

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

x は Postgres のバージョンを指定します。

データベースサーバーの自動起動の使用

If a Master node reboots, **variable_prod_name1** may detect the database is down on the Master node and promote a Standby node to the role of Master. If this happens, the **variable_prod_name1** agent on the (rebooted) Master node will not get a chance to write the recovery.conf file; the recovery.conf file prevents the database server from starting. If this happens, the rebooted Master node will return to the cluster as a second Master node.

これを防ぐには、**variable_prod_name1**を開始しますデータベースサーバーを起動する前のエージェント。エージェントはアイドルモードで起動し、クラスター内に既にマスターがあるかどうかを確認します。マスターノードがある場合、エージェントは recovery.conf または standby.signal ファイルが存在することを確認し、データベースは 2 番目のマスターとして起動しません。

ファイアウォールを介した通信を保証する

Linux ファイアウォール (iptables など) がフェイ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は、ファイアウォール構成にルールを追加して、**variable_prod_name1**間の TCP 通信を許可する必要がある場合があります。クラスター内のプロセス。例えば:

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

上記のコマンドは、狭い範囲のポート（7800 から 7810）を開きます。lvariable_prod_name クラスタープロパティファイルで指定されたポートに対応するポートを介して接続します。

データベースユーザーに十分な権限があることを確認してください

efm.properties ファイルの db.user プロパティで指定されたデータベースユーザーは、lvariable_prod_name に代わって次の関数を呼び出すための十分な権限を持っている必要があります:

```
pg_current_wal_lsn()

pg_last_wal_replay_lsn()

pg_wal_replay_resume()

pg_reload_conf()
```

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

ユーザーには、構成変数の値を読み取るための権限も必要です。データベースのスーパーユーザーは PostgreSQL の GRANT コマンドを使用して必要な権限を与えることができます:

```
GRANT pg_read_all_settings TO user_name;
```

pg_read_all_settings の詳細については、**‘PostgreSQL コア のドキュメントを参照してください’**
<<https://www.postgresql.org/docs/12/default-roles.html>>’。

2.2 チュートリアル-シンプルなフェールオーバーマネージャークラスターの構成

このチュートリアルでは、`lvariable_prod_name`の迅速な設定について説明しますテスト環境のクラスター。このガイドの他のセクションでは、`lvariable_prod_name`を構成する前に読んで理解しておくべき重要な情報を提供しています。本番デプロイメント用。

このチュートリアルでは、次のことを前提としています。

- データベースサーバーが実行中で、マスターと 1 つまたは 2 つのスタンバイノードの間にストリーミングレプリケーションが設定されています。
- `lvariable_prod_name`をインストールしました各ノード。

次の例では、`efm` という名前のクラスターを作成します。

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

ステップ 1:動作する構成ファイルを作成する

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

```
cd /etc/edb/efm-|variable_prod_version|

cp efm.properties.in efm.properties

cp efm.nodes.in efm.nodes

chown efm:efm efm.properties

chown efm:efm efm.nodes
```

ステップ 2:暗号化されたパスワードを作成する

暗号化されたパスワードを作成します（プロパティファイルに必要）。

```
/usr/edb/efm-|variable_prod_version|/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されたバージョンを生成します。

ステップ 3:efm.properties ファイルを更新する

`efm.properties` ファイルには、`lvariable_prod_name`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集まる。プロパティ設定の変更は、フェイルオーバーマネージャの起動時に適用されます。

次のプロパティは、`lvariable_prod_name`を構成するために必要な最小限のプロパティです。集まる。本番システムを設定している場合は、`:doc:` クラスタプロパティファイルをご覧ください。`<cluster_properties>` プロパティの

完全なリストについては。

データベース接続プロパティ（必要に応じて他のデータベースに接続できるように、ウィットネスでも必要）：

```
db.user

db.password.encrypted

db.port

db.database
```

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

```
db.service.owner
```

EFM uses the `db.service.name` and `db.bin` properties when restarting the server. The service name provided with the `db.service.name` property is used when restarting the server with `service` or `systemctl`; the value you provide in the `db.bin` property (the path to the Postgres bin directory), will be used for calls to `pg_ctl`. Please note that `db.bin` is a required field. `db.service.name` is required if you are running the database as a service.

```
db.service.name

db.bin
```

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

```
db.data.dir
```

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

```
user.email
```

これは、EFM に使用するノードとポートのローカルアドレスです。他のノードはこのアドレスを使用してエージェントに到達し、エージェントはこのアドレスを使用して（localhost に接続するのではなく）ローカルデータベースに接続します。フォーマットの例を以下に示します。

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

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

```
is.witness
```

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

```
ping.server.ip=8.8.8.8
```

実稼働クラスターを構成する場合、システムの構成と使用法に応じて、以下のプロパティが `true` または `false` になる可能性があります。EFM テストクラスターを構成している場合は、両方を `true` に設定して起動を簡略化します。

```
auto.allow.hosts=true

stable.nodes.file=true
```

ステップ 4:efm.nodes ファイルを更新する

起動時に `efm.nodes` ファイルが読み込まれ、残りのクラスターを見つける方法がエージェントに通知されます。最初のノードが起動された場合は、後続のノードの認証を簡素化するために使用できます。

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

```
1.2.3.4:7800

1.2.3.5:7800

1.2.3.6:7800
```

Please note that the `|variable_prod_name|` agent will not verify the content of the `efm.nodes` file; the agent expects that some of the addresses in the file cannot be reached (e.g. that another agent hasn't been started yet). For more information about the `efm.nodes` file, see [The Cluster Members File](#)

ステップ 5:他のノードを構成する

Copy the `efm.properties` and `efm.nodes` files to the `/etc/edb/efm-3.10` directory on the other nodes in your sample cluster. After copying the files, change the file ownership so the files are owned by `efm:efm`. The `efm.properties` file can be the same on every node, except for the following properties:

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

ステップ 6:EFM クラスターを開始する

任意のノードで、`|variable_prod_name|`を開始しますエージェント。エージェントの名前は `edb-efm-3.10` です。プラットフォーム固有の `service` コマンドを使用して、サービスを制御できます。たとえば、`RHEL/CentOS7.x` または `RHEL/CentOS8.x` ホストでは、次のコマンドを使用します。

```
systemctl start edb-efm-|variable_prod_version|
```

CentOS または RHEL6.x ホストでは、次のコマンドを使用します。

```
service edb-efm-|variable_prod_version| start
```

エージェントが起動したら、次のコマンドを実行して、単一ノードクラスターのステータスを確認します。[許可されたノードのホスト] リストに他のノードのアドレスが表示されます。

```
/usr/edb/efm-|variable_prod_version|/bin/efm cluster-status efm
```

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

起動に失敗したエージェントがある場合は、起動ログで問題の原因を確認してください。

```
cat /var/log/efm-|variable_prod_version|/startup-efm.log
```

スイッチオーバーの実行

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

```
/usr/edb/efm-|variable_prod_version|/bin/efm promote efm -switchover
```

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

ヘルプにすばやくアクセスするには、次のコマンドを呼び出します。

```
/usr/edb/efm-|variable_prod_version|/bin/efm --help
```

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

第 3 章

フェイルオーバーマネージャーのインストール

EnterpriseDB リポジトリへのアクセスを許可する資格情報をリクエストするには、EDBWeb サイトにアクセスしてください。

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

3.1 RedHat、CentOS、または OEL ホストへのフェイルオーバーマネージャーのインストール

資格情報を受け取ったら、クラスタの各ノードで EnterpriseDB リポジトリ構成ファイルを作成し、ファイルを変更してアクセスを有効にする必要があります。次の手順では、EnterpriseDB リポジトリへのアクセスに関する詳細情報を提供します。手順は、クラスターの各ノードで実行する必要があります。

リポジトリ構成ファイルを作成するには、スーパーユーザー特権を想定して、次のコマンドを呼び出します。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

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

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

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

変更を構成ファイルに保存した後、次のコマンドを使用してフェイルオーバーマネージャーをインストールできます。

- RHEL または CentOS7 の場合:

```
yum install edb-efm310
```

- RHEL または CentOS8 の場合:

```
dnf install edb-efm310
```

システムで認識されないソースによって署名された RPM パッケージをインストールすると、yum はローカルサーバーにキーをインポートする許可を要求する場合があります。プロンプトが表示され、パッケージが信頼できるソースからのものであることに満足したら、y を入力し、Return を押して続行します。

インストール中に、yum は解決できない依存関係に遭遇する可能性があります。含まれている場合は、手動で解決する必要がある必要な依存関係のリストが表示されます。

!variable_prod_name!ルートでインストールする必要があります。インストールプロセス中に、インストーラーは、enterprisedb または postgres が所有するクラスターの FailoverManager サービスを制御するスクリプトを呼び出すための十分な権限を持つ efm という名前のユーザーも作成します。

!variable_prod_name!を使用している場合 enterprisedb または postgres 以外のユーザーが所有するクラスターを監視するには、[フェイルオーバーマネージャーのアクセス許可の拡張を参照してください](#)。

!variable_prod_name!をインストールした後クラスターの各ノードで、次のことを行う必要があります。

1. **!doc:cluster** プロパティファイルを変更する<cluster_properties>‘各ノード上。
2. **!doc:clustermembers** ファイルを変更します<cluster_members>‘各ノード上。
3. 該当する場合は、仮想 IP アドレスの設定と、クラスタープロパティファイルで識別されるスクリプトを構成してテストします。
4. **!variable_prod_name!**を開始しますクラスターの各ノードのエージェント。**!variable_prod_name!**の制御の詳細についてはサービス、**‘セクション 5<lt; # controlling-the-failover-manager-service>>‘__**を参照してください。

3.1.1 設置場所

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

Component	ロケーション
実行可能ファイル	/usr/edb/efm-3.10/bin
図書館	/usr/edb/efm-3.10/lib
クラスター構成ファイル	/etc/edb/efm-3.10
ログ	/Var/log/efm- バージョン
ファイルをロックする	/var/lock/efm-3.10
ログローテーションファイル	/etc/logrotate.d/efm-3.10
sudo 設定ファイル	/etc/sudoers.d/efm-310
sudo なしで VIP にアクセスするためのバイナリ	/usr/edb/efm-3.10/bin/secure

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

`lvariable_prod_name1`をインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報も必要です。リポジトリの認証情報をリクエストするには、次の EnterpriseDBWeb サイトにアクセスしてください。

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

EDB のリンクをたどってください `lvariable_prod_name1` 資格情報を要求するテーブル。

次の手順では、EnterpriseDBapt リポジトリを使用して `lvariable_prod_name1` をインストールする手順を示します。コマンドを使用するときは、`*username*` および `*password*` を EnterpriseDB によって提供される資格情報に置き換えます。

1. スーパーユーザー権限を想定します。

```
sudo su -
```

2. EnterpriseDBapt リポジトリを構成します。

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

3. 安全な APT リポジトリのサポートをシステムに追加します。

```
apt-get install apt-transport-https
```

4. EDB 署名キーを追加します。

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

5. リポジトリのメタデータを更新します。

```
apt-get update
```

6. `lvariable_prod_name1` をインストールします:

```
apt-get install edb-efm310
```

3.3 SLES ホストへのフェイルオーバーマネージャーのインストール

`!variable_prod_name!`をインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報も必要です。リポジトリの認証情報をリクエストするには、次の EnterpriseDBWeb サイトにアクセスしてください。

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

zypper パッケージマネージャを使用して、`!variable_prod_name!`をインストールできます。SLES12 ホスト上のエージェント。zypper は、パッケージをインストールするときにパッケージの依存関係を満たすよう試みますが、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。

`!variable_prod_name!`をインストールする前に、スーパーユーザー権限を引き受け、ファイアウォールを停止する必要があります。次に、次のコマンドを使用して、EnterpriseDB リポジトリをシステムに追加します。

```
zypper addrepo http://zypp.enterprisedb.com/suse/epas96-sles.repo
zypper addrepo http://zypp.enterprisedb.com/suse/epas-sles-tools.repo
zypper addrepo http://zypp.enterprisedb.com/suse/epas-sles-dependencies.repo
```

コマンドはリポジトリ設定ファイルを `/etc/zypp/repos.d` ディレクトリに作成します。次に、次のコマンドを使用して SLES ホストのメタデータを更新し、EnterpriseDB リポジトリを含めます。

```
zypper refresh
```

プロンプトが表示されたら、リポジトリの資格情報を入力し、提供されたキーを常に信頼するようにを指定し、EnterpriseDB リポジトリを含めるようにメタデータを更新します。

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

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

次に、zypper ユーティリティを使用して、`!variable_prod_name!`をインストールします。エージェント:

```
zypper install edb-efm310
```

SUSE ホストの登録の詳細については、次の Web サイトにアクセスしてください。

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

第 4 章

フェイルオーバーマネージャーの設定

構成可能 `variable_prod_name` プロパティは、2 つのユーザー変更可能なファイルで指定されます。

- `efm.properties`
- `efm.nodes`

4.1 クラスタプロパティファイル

`variable_prod_name` の各ノードクラスターには、それが存在する個々のノードのプロパティを含むプロパティファイル（デフォルトでは、`efm.properties` という名前）があります。`variable_prod_name` インストーラは、`/etc/edb/efm-3.10` ディレクトリに `efm.properties.in` という名前のプロパティファイルのファイルテンプレートを作成します。

`variable_prod_name` の完了後インストールでは、ファイルの内容を変更する前に、テンプレートの作業用コピーを作成する必要があります。

```
# cp /etc/edb/efm-variable_prod_version/efm.properties.in /etc/edb/efm-variable_
prod_version/efm.properties
```

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

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

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

クラスタプロパティファイルを作成した後、必要に応じて構成パラメーター値を追加（または変更）します。各プロパティの詳細については、“[クラスタプロパティを指定する](#)”を参照してください。

プロパティファイルは `root` が所有しています。`variable_prod_name` サービススクリプトは `/etc/edb/efm-3.10 directory` でファイルを見つけることを期待しています。プロパティファイルを別の場所に移

動する場合は、新しい場所を指定するシンボリックリンクを作成する必要があります。

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

4.1.1 クラスタプロパティの指定

クラスタ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、`!variable_prod_name!`の接続プロパティと動作を指定できます。集まる。プロパティ設定への変更は、`!variable_prod_name!`のときに適用されます開始します。プロパティ値を変更した場合は、`!variable_prod_name!`を再起動する必要があります変更を適用します。

プロパティ値では大文字と小文字が区別されます。Postgres は引用符で囲まれた文字列をパラメータ値で使用しますが、`!variable_prod_name!`プロパティ値に引用符で囲まれた文字列は許可されません。たとえば、次のように Postgres 構成パラメータで IP アドレスを指定する場合があります。

```
listen_addresses='192.168.2.47'
```

`!variable_prod_name!`値を引用符で囲む必要はありません。

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

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

以下のプロパティを使用して、`!variable_prod_name!`の接続の詳細を指定します集まる:

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

指定された `db.user` には、`!variable_prod_name!`に代わって選択された PostgreSQL コマンドを呼び出すための十分な権限が必要です。詳細については、[前提条件をご覧ください。](#)。

データベースユーザーのパスワードの暗号化については、“[データベースパスワードの暗号化](#)”を参照してください。。

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

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

サービスを開始または停止するときに `service` コマンドまたは `systemctl` コマンドを使用する場合は、データベースサービスの名前を `db.service.name` プロパティに指定します。

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

データベースサービスを開始または停止するたびに、同じサービス制御メカニズム (`pg_ctl`、`service`、または `systemctl`) を使用する必要があります。 `pg_ctl` プログラムを使用してサービスを制御する場合は、 `db.bin` プロパティで `pg_ctl` プログラムの場所を指定します。

```
# Specify the directory containing the pg_controldata/pg_ctl commands,
# for example:
# /usr/edb/as11/bin. Unless the db.service.name property is used, the
# pg_ctl command is used to start/stop/restart databases as needed
# after a failover or switchover. This property is required.
db.bin=
```

`db.data.dir` プロパティを使用して、昇格中にクラスターのマスターノードでリカバリファイルが書き込まれる場所を指定します。このプロパティは、マスターノードとスタンバイノードで必要です。専用の監視ノードでは必要ありません。

```
# For database version 12 and up, this is the directory where a
# standby.signal file will exist for a standby node. For previous
# versions, this is the location of the db recovery.conf file on
# the node.
# After a failover, the recovery.conf files on remaining standbys are
# changed to point to the new master db (a copy of the original is made
# first). On a master node, a recovery.conf file will be written during
# failover and promotion to ensure that the master node can not be
```

(次のページに続く)

(前のページからの続き)

```
# restarted as the master database.
# This corresponds to database environment variable PGDATA and should
# be same as the output of query 'show data_directory;' on respective
# database.
db.data.dir=
```

データベース構成ファイルが `recovery.conf` または `standby.signal` ファイルと同じディレクトリに保存されていない場合は、`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、AdvancedServer 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ある必要があります。この値は、データベースを停止、起動、または再起動するときに Postgres の data ディレクトリの場所として使用されます。

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

データベース構成ファイルの詳細については、**‘PostgreSQLWeb サイトにアクセスしてください’**
<<https://www.postgresql.org/docs/12/runtime-config-file-locations.html>>‘。

Use the `jdbc.sslmode` property to instruct **|variable_prod_name|** to use SSL connections; by default, SSL is disabled.

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

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

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

SSL の構成と使用については、以下を参照してください。

<https://www.postgresql.org/docs/12/static/ssl-tcp.html>

そして

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

`user.email` プロパティを使用して、フェイルオーバーマネージャーから送信された通知を受信するメールアドレス（または複数のメールアドレス）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=
```

`from.email` プロパティは、`!variable_prod_name!`からのメール通知で送信者のアドレスとして使用される値を指定します。あなたはできる:

- 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` プレースホルダーを使用して、カスタムのメールアドレスを指定します（例:`example@%h`）。プレースホルダーは、Linux ホスト名ユーティリティによって返されるホストの名前に置き換えられます。

通知の詳細については、[通知を参照してください](#)。

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

`notification.level` プロパティを使用して、`!variable_prod_name!`が実行される最小の重大度レベルを指定します。ユーザー通知または通知スクリプトが呼び出されたときに送信されます。通知の完全なリストについては、[通知を参照してください](#)。

```
# Minimum severity level of notifications that will be sent by
# the agent. The minimum level also applies to the notification
# script (below). Valid values are INFO, WARNING, and SEVERE.
```

(次のページに続く)

(前のページからの続き)

```
# A list of notifications is grouped by severity in the user's
# guide.
notification.level=INFO
```

script.notification プロパティを使用して、通知サービスとして機能するユーザー提供のスクリプトへのパスを指定します。スクリプトには、メッセージの件名とメッセージ本文が渡されます。スクリプトは毎回呼び出されます|variable_prod_name|ユーザー通知を生成します。

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

bind.address プロパティは、|variable_prod_name|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集まる。

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

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

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

admin.port プロパティを使用して、|variable_prod_name|が設定されているポートを指定します管理コマンドを待機します。

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

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

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

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

注釈: `is.witness` が `true` の場合、`!variable_prod_name!` リカバリ状態はチェックしません。

次のプロパティは、ローカルサーバーに適用されるプロパティを指定します。

- `local.period` プロパティは、データベースサーバーへの接続を試行する間隔を秒数で指定します。
- `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ーからの肯定的な応答を待つ時間を指定します。
- `local.timeout.final` プロパティは、現在のノード上のデータベースサーバーへの接続を最後に試行した後、エージェントが待機する時間を指定します。 `local.timeout.final` プロパティで指定された秒数以内にデータベースから応答が受信されない場合、データベースは失敗したと見なされます。

たとえば、これらのプロパティのデフォルト値が指定されている場合、ローカルデータベースのチェックは 10 秒に 1 回行われます。ローカルデータベースへの接続試行が 60 秒以内に成功しない場合、`!variable_prod_name!` 最後にデータベースへの接続を試みます。10 秒以内に応答がない場合、`!variable_prod_name!` 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
```

(次のページに続く)

(前のページからの続き)

```
# 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、これらの値をビジネスモデルに合わせて変更する必要があります。

`remote.timeout` プロパティを使用して、エージェントがリモートデータベースサーバーからの応答を待機する秒数（つまり、スタンバイエージェントがマスターデータベースが実際にダウンしていることを確認してからフェイルオーバーを実行する前に待機する時間）を指定します。

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

`node.timeout` プロパティを使用して、ノードが失敗したかどうかを判断するときにエージェントがノードからの応答を待機する秒数を指定します。`node.timeout` プロパティ値は、エージェント間の通信のタイムアウト値を指定します。クラスタープロパティファイルの他のタイムアウトプロパティは、エージェントとデータベース間の通信の値を指定します。

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

Use the `stop.isolated.master` property to instruct **`variable_prod_name1`** to shut down the database if a master agent detects that it is isolated. When true (the default), **`variable_prod_name1`** will stop the database before invoking the script specified in the `script.master.isolated` property.

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

Use the `stop.failed.master` property to instruct **`variable_prod_name1`** to attempt to shut down a master database if it can not reach the database. If true, **`variable_prod_name1`** will run the script specified in the `script.db.failure` property after attempting to shut down the database.

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

`master.shutdown.as.failure` パラメーターを使用して、`variable_prod_name1`のシャットダウンを示します。マスターノードのエージェントは障害として扱われるべきです。このパラメーターが `true` に設定されていて、マスターエージェントが何らかの理由で停止した場合、クラスターはマスターノードのデータベースが実行されているかどうかを確認しようとします。

- データベースに到達すると、エージェントのステータスを通知する通知が送信されます。
- データベースに到達しない場合、フェイルオーバーが発生します。

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

`master.shutdown.as.failure` プロパティは、マスターノードの偶発的なシャットダウンなどの障害ではなく、ユーザーエラーをキャッチするためのものです。ユーザーがマスターを停止したように、ノードの適切なシャットダウンは、残りのクラスターに表示されます。`variable_prod_name1` エージェント（たとえば、マスターデータベースのメンテナンスを実行するため）。`master.shutdown.as.failure` プロパティを `true` に設定した場合、メンテナンスの際には注意が必要です。

`master.shutdown.as.failure` が `true` のときにマスターデータベースのメンテナンスを実行するには、マスターエージェントを停止し、マスターエージェントが失敗したがデータベースがまだ実行中であるという通知を受け取るまで待つ必要があります。その後、`master` データベースを停止しても安全です。または、`efm stop-cluster` コマンドを使用して、障害チェックを実行せずにすべてのエージェントを停止できます。

`update.physical.slots.period` プロパティを使用して、データベースバージョン 12 以降のスロットアドバンス周波数を定義します。`update.physical.slots.period` がゼロ以外の値に設定されている場合、マスターエージェントは `update.physical.slots.period` 秒ごとに物理レプリケーションスロットの現在の `restart_lsn` を読み取り、この情報を `pg_current_wal_lsn` と `primary_slot_name` (`postgresql.conf` ファイルで設定されている場合) をスタンバイに設定します。物理スロットがまだ存在しない場合、このパラメーターをゼロ以外の値に設定すると、スロットが作成され、これらのスロットの `restart_lsn` parameter が更新されます。昇格できないスタンバイは新しいスロットを作成しませんが、存在する場合は更新します。

```
# Period in seconds between having the master 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
```

Use the `ping.server.ip` property to specify the IP address of a server that `|variable_prod_name|` can use to confirm that network connectivity is not a problem.

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

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

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

許可されたホストリストの更新を開始する最初のノードの `.nodes` ファイルで指定されたアドレスを使用するようにサーバーに指示するには、`auto.allow.hosts` プロパティを使用します。このプロパティを有効にする (`auto.allow.hosts` を `true` に設定する) と、クラスターの起動が簡単になります。

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

(次のページに続く)

(前のページからの続き)

```
# is already known.
auto.allow.hosts=false
```

`stable.nodes.file` プロパティを使用して、ノードがクラスターに参加または離脱した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 アドレスが変化しないクラスターで最も役立ちます。

```
# When set to true, EFM will not rewrite the .nodes file whenever
# new nodes join or leave the cluster. This can help starting a
# cluster in the cases where it is expected for member addresses
# to be mostly static, and combined with 'auto.allow.hosts' makes
# startup easier when learning failover manager.
stable.nodes.file=false
```

`db.reuse.connection.count` プロパティを使用すると、管理者は回数を指定できます`|variable_prod_name|` 同じデータベース接続を再利用して、データベースの状態をチェックします。デフォルト値は 0 で、`|variable_prod_name|` 毎回新しい接続を作成します。このプロパティは、専用の監視ノードでは必要ありません。

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

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

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

Use the `auto.reconfigure` property to instruct `|variable_prod_name|` to enable or disable automatic reconfiguration of remaining Standby servers after the primary standby is promoted to Master. Set the property to `true` to enable automatic reconfiguration (the default) or `false` to disable automatic reconfiguration. This property is not required on a dedicated witness node. If you are using Advanced Server or PostgreSQL version 11 or earlier, the `recovery.conf` file will be backed up during the reconfiguration process.

```
# After a standby is promoted, Failover Manager will attempt to
# update the remaining standbys to use the new master. For database
# versions before 12, Failover Manager will back up recovery.conf.
# Then it will change the host parameter of the primary_conninfo entry
# in recovery.conf or postgresql.auto.conf, and restart the database.
```

(次のページに続く)

(前のページからの続き)

```
# The restart command is contained in either the efm_db_functions or
# efm_root_functions file; default when not running db as an os
# service is: "pg_ctl restart -m fast -w -t <timeout> -D <directory>"
# where the timeout is the local.timeout property value and the
# directory is specified by db.data.dir. To turn off
# automatic reconfiguration, set this property to false.
auto.reconfigure=true
```

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

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

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

同じ量のデータが複数のスタンバイノードに書き込まれ、フェイルオーバーが発生した場合、`use.replay.tiebreaker` の値により、変数の方法が決定されます。置換マスターを選択します。`!variable_prod_name!`に指示するには、`use.replay.tiebreaker` プロパティを `true` に設定しますログシーケンス番号によって決定されるように、回復からより早く出るノードにフェイルオーバーする。ログシーケンス番号を無視して、ユーザー設定に基づいてノードを昇格するには、`use.replay.tiebreaker` を `false` に設定します。

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

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

```
# During a switchover, recovery settings are copied from a standby
# to the original master. If the application.name property is set,
# Failover Manager will replace the application_name portion of the
# primary_conninfo entry with this property value before starting
# the original master database as a standby. If this property is
```

(次のページに続く)

(前のページからの続き)

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

注意: マスターと昇格可能なスタンバイで `application.name` プロパティを設定する必要があります。フェイルオーバー/スイッチオーバーが発生した場合、マスターノードが再びスタンバイノードになる可能性があります。

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

‘restore_command’ の使用の詳細については、PostgreSQL のドキュメントを参照してください<<https://www.postgresql.org/docs/12/runtime-config-wal.html#RUNTIME-CONFIG-WAL-ARCHIVE-RECOVERY>>‘。

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

マスターノードのデータベースパラメーター `synchronous_standby_names` は、マスターノードが書き込みトランザクションを確実に受け入れることができるように、データの受信を確認する同期スタンバイサーバーの名前と数を指定します。 `reconfigure.num.sync` プロパティが `true` に設定されている場合、`|variable_prod_name|` 同期スタンバイサーバーの数を減らし、マスター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 master database. If set to true,
# Failover Manager will reduce the number of standbys needed
# in the master's synchronous_standby_names property and reload
# the master configuration.
# Failover Manager will not reduce the number below 1, taking
# the master out of synchronous replication, unless the
# reconfigure.sync.master property is also set to true.
reconfigure.num.sync=false
```

スタンバイノードの数が必要なレベルを下回った場合にマスターデータベースを同期レプリケーションモードから外すには、`reconfigure.sync.master` プロパティを `true` に設定します。`reconfigure.sync.master` を `false` に設定すると、スタンバイカウントが低下した場合に通知が送信されますが、同期レプリケーションは中断されません。

```
# Take the master database out of synchronous replication mode when
# needed. If set to true, Failover Manager will clear the
# synchronous_standby_names configuration parameter on the master
# if the number of synchronous standbys drops below the required
# level for the master to accept writes.
# If set to false, Failover Manager will detect the situation but
# will only send a notification if the standby count drops below the
# required level.
#
# CAUTION: TAKING THE MASTER 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.master=false
```

`minimum.standbys` プロパティを使用して、クラスターに保持されるスタンバイノードの最小数を指定します。スタンバイカウントが指定された最小数に減少した場合、マスターノードに障害が発生した場合、レプリカノードは昇格されません。

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

`recovery.check.period` プロパティを使用して、`!variable_prod_name!`を実行する秒数を指定しますデータベースが復旧していないかどうかを確認する前に待機します。

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

`restart.connection.timeout` プロパティを使用して、`!variable_prod_name!`を実行する秒数を指定しますそのノード上のデータベースが接続を受け入れる準備をしている間、新しく再構成されたマスターまたはスタンバイノードへの接続を試みます。

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

(次のページに続く)

(前のページからの続き)

```
# reconfigured for a new master, and to master databases that
# are stopped and started as standbys during a switchover.
# This retry mechanism is unrelated to the auto.resume.period
# parameter.
restart.connection.timeout=60
```

auto.resume.period プロパティを使用して、エージェントがそのデータベースの監視を再開しようとする秒数（監視対象のデータベースに障害が発生し、エージェントがアイドル状態になった後、または IDLE モードで開始した後）を指定します。

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

variable_prod_name1 provides support for clusters that use a virtual IP. If your cluster uses a virtual IP, provide the host name or IP address in the virtual.ip property; specify the corresponding prefix in the virtual.ip.prefix property. If virtual.ip is left blank, virtual IP support is disabled.

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

指定された仮想 IP アドレスは、クラスターのマスターノードにのみ割り当てられます。virtual.ip.single=true を指定すると、フェイルオーバー時に新しい VIP で同じ VIP アドレスが使用されます。クラスターの各ノードに固有の IP アドレスを提供するには、値 false を指定します。

仮想 IP アドレスの使用については、“[仮想 IP アドレスでのフェイルオーバーマネージャーの使用](#)”を参照してください。

```
# These properties specify the IP and prefix length that will be
# remapped during failover. If you do not use a VIP as part of
# your failover solution, leave the 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
```

(次のページに続く)

(前のページからの続き)

```
# false, you can specify unique values for virtual.ip and
# virtual.ip.prefix on each node.
#
# If you are using an IPv4 address, the virtual.ip.interface value
# should not contain a secondary virtual ip id (do not include
# ":1", etc).
virtual.ip=
virtual.ip.interface=
virtual.ip.prefix=
virtual.ip.single=true
```

注意: マスターエージェントが開始され、ノードに現在 VIP がない場合、EFM エージェントはそれを取得します。マスターエージェントを停止しても、ノードから VIP は削除されません。

`check.vip.before.promotion` プロパティを `false` に設定して、`!variable_prod_name!` は、障害発生時に VIP を新しいマスターに割り当てる前に、VIP が使用中かどうかを確認しません。これにより、複数のノードが同じ VIP アドレスでブロードキャストする可能性があることに注意してください。マスターノードが分離されているか、別のプロセスでシャットダウンできる場合を除き、このプロパティを `true` に設定する必要があります。

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

次のプロパティを使用して、スイッチオーバーまたはマスターの障害シナリオでロードバランサーを再構成するスクリプトへのパスを提供します。スクリプトは、スタンバイ障害が発生した場合にも呼び出されます。これらのプロパティを使用している場合は、クラスターのすべてのノード（マスター、スタンバイ、ウィットネス）でこれらのプロパティを指定して、データベースノードに障害が発生した場合に、別のノードが切り離されたスクリプトを、障害が発生したノードのアドレスで呼び出すようにします。

ノードをロードバランサーに接続する必要があるときに呼び出されるスクリプトを識別するために、`script.load.balancer.attach` プロパティの後にスクリプト名を指定します。ノードをロードバランサーから切り離す必要があるときに呼び出されるスクリプトの名前を指定するには、`script.load.balancer.detach` プロパティを使用します。クラスターに接続またはクラスターから削除されるノードの IP アドレスを表す `%h` プレースホルダーを含めます。`!variable_prod_name!` に指示する `%t` プレースホルダーを含めます文字列に `m`（マスターノードの場合）または `s`（スタンバイノードの場合）を含める。

```
# 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
```

(次のページに続く)

(前のページからの続き)

```
# when a database has failed or is about to be stopped. Use %h to
# represent the IP/hostname of the node that is being
# attached/detached. Use %t to represent the type of node being
# attached or detached: the letter m will be passed in for master nodes
# and the letter s for standby nodes.
#
# Example:
# script.load.balancer.attach=/somepath/attachscript %h %t
script.load.balancer.attach=
script.load.balancer.detach=
```

script.fence は、スタンバイノードからマスターノードへの昇格中に呼び出されるオプションのユーザー指定スクリプトへのパスを指定します。

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

script.post.promotion プロパティを使用して、スタンバイノードがマスターに昇格した後に呼び出されるオプションのユーザー指定スクリプトへのパスを指定します。

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

(次のページに続く)

(前のページからの続き)

```
# should be the same as the node's efm binding address.
#
# Example:
# script.post.promotion=/somepath/myscript %f %p
script.post.promotion=
```

script.resumed property を使用して、エージェントがデータベースの監視を再開したときに呼び出されるユーザー指定のスクリプトへのオプションのパスを指定します。

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

script.db.failure プロパティを使用して、`!variable_prod_name!`である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呼び出されます。

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

script.master.isolated プロパティを使用して、`!variable_prod_name!`である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マスターデータベースを監視しているエージェントが、マスターが`!variable_prod_name!`の大部分から分離されていることを検出した場合に呼び出されます集まる。このスクリプトは、VIP が解放された直後に呼び出されます (VIP が使用されている場合)。

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

script.remote.pre.promotion プロパティを使用して、ノードがデータベースをマスターに昇格させようとしているときに、昇格に関与していないエージェントノードで呼び出されるスクリプトのパスと名前を指定します。

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

```
# Absolute path to script invoked on non-promoting agent nodes
# before a promotion.
#
# This optional user-supplied script will be invoked on other
```

(次のページに続く)

(前のページからの続き)

```
# agents when a node is about to promote its database. The exit
# code from this script has no effect on Failover Manager, but
# will be included in a notification sent after the script
# executes.
#
# Pass a parameter (%p) with the script to identify the new
# primary node address.
#
# Example:
# script.remote.pre.promotion=/path_name/script_name %p
script.remote.pre.promotion=
```

`script.remote.post.promotion` プロパティを使用して、昇格の発生後に非マスターノードで呼び出されるスクリプトのパスと名前を指定します。

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

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

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

`custom.monitor.timeout` を使用して、スクリプトの実行を許可する最大時間を指定します。指定された時間内にスクリプトの実行が完了しない場合、`!variable_prod_name!`通知を送信します。

`!variable_prod_name!`に指示するには、`custom.monitor.safe.mode` を `true` に設定しますスクリプトからゼロ以外の終了コードを報告しますが、終了コードの結果としてスタンバイを昇格させません。

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

(次のページに続く)

(前のページからの続き)

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

sudo.command プロパティを使用して、`variable_prod_name1`によって呼び出されるコマンドを指定します拡張権限が必要なタスクを実行するとき。このオプションを使用して、システム認証に固有のコマンドオプションを含めます。

sudo.user.command プロパティを使用して、`variable_prod_name1`によって呼び出されるコマンドを指定しますデータベース所有者が実行するコマンドを実行するとき。

```
# Command to use in place of 'sudo' if desired when efm runs
# the efm_db_functions or efm_root_functions, or efm_address
# scripts.
# Sudo is used in the following ways by efm:
#
# sudo /usr/edb/efm-<version>/bin/efm_address <arguments>
# sudo /usr/edb/efm-<version>/bin/efm_root_functions <arguments>
# sudo -u <db service owner> /usr/edb/efm-<version>/bin/efm_db_functions <arguments>
#
# 'sudo' in the first two examples will be replaced by the value
# of the sudo.command property. 'sudo -u <db service owner>' will
# be replaced by the value of the sudo.user.command property.
# The '%u' field will be replaced with the db owner.
sudo.command=sudo
sudo.user.command=sudo -u %u
```

`lock.dir` プロパティを使用して、`lvariable_prod_name1`の代替の場所を指定しますロックファイル; ファイルが `lvariable_prod_name1`を妨げているノード上の単一クラスターに対して複数の（孤立している可能性がある）エージェントを開始しない。

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

`log.dir` プロパティを使用して、エージェントログファイルが書き込まれる場所を指定します。`lvariable_prod_name1`ディレクトリが存在しない場合は、ディレクトリの作成を試みます。

```
# 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=
```

`lvariable_prod_name1`で UDP または TCP プロトコルを有効にした後ホストでは、`syslog` へのロギングを有効にできます。プロトコルタイプ (UDP または TCP) を指定するには `syslog.protocol` パラメータを使用し、`syslog` ホストのリスナーポートを指定するには `syslog.port` パラメータを使用します。 `syslog.facility` の値は、エントリを作成したプロセスの識別子として使用できます。値は `LOCAL0` と `LOCAL7` の間でなければなりません。

```
# Syslog information. The syslog service must be listening on
# the port for the given protocol, which can be UDP or TCP.
# The facilities supported are LOCAL0 through LOCAL7.
syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

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

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

syslog ロギングの構成の詳細については、[syslog ログファイルエントリの有効化を参照してください](#)。。

jgroups.loglevel および efm.loglevel パラメータを使用して、`!variable_prod_name!`によってログに記録される詳細レベルを指定します。デフォルト値は INFO です。ロギングの詳細については、[ロギングの制御を参照してください](#)。

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

jvm.options プロパティを使用して、JVM 関連の構成情報を渡します。デフォルトの設定では、`!variable_prod_name!`が使用するメモリの量を指定します。エージェントの使用が許可されます。

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

データベースのパスワードの暗号化

`!variable_prod_name!`クラスのプロパティファイルに含める前に、データベースパスワードを暗号化する必要があります。efm ユーティリティを使用する`<efm_encrypt>`（`/usr/edb/efm-3.10/bin/` ディレクトリにあります）パスワードを暗号化します。パスワードを暗号化する場合、ユーティリティを呼び出すときにコマンドラインでパスワードを渡すか、EFMPASS 環境変数を使用できます。

パスワードを暗号化するには、次のコマンドを使用します。

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

ここで、`<cluster_name>` は `!variable_prod_name!` の名前を指定します集まる。

`--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。例えば:

```
export EFMPASS=password
```

`--from-env` オプションを含めない場合、`!variable_prod_name!` クラスのプロパティファイルに配置するための暗号化されたパスワードを生成する前に、データベースパスワードを 2 回入力するように求められます。ユーティリティが暗号化されたパスワードを共有する場合、暗号化されたパスワードをコピーしてクラスのプロパティファイルに貼り付けます。

ご注意ください: 多くの Java ベンダーは、フル強度の暗号化が含まれているバージョンの Java を出荷していますが、輸出規制により有効にされていません。データベースのパスワードを暗号化しようとしたときに不正なキーサイズを参照するエラーが発生した場合は、プラットフォームに無制限のポリシーを提供する Java 暗号化拡張機能 (JCE) をダウンロードして有効にする必要があります。

次の例は、`encrypt` ユーティリティを使用して `acctg` クラスタのパスワードを暗号化する方法を示しています。

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

注意: プロパティファイルが存在しない場合、ユーティリティから通知されます。

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、`|variable_prod_name|`を開始します。サービス。暗号化されたパスワードに問題がある場合、`|variable_prod_name|`サービスは開始されません:

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

`|variable_prod_name|`の開始時にこのメッセージを受け取った場合サービス、詳細は起動ログ (`/var/log/efm-3.10/startup-efm.log` にあります) を参照してください。

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

```
systemctl status edb-efm-|variable_prod_version|
```

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

EFMPASS 環境変数の使用

次の例は、パスワードを暗号化するときに `--from-env` 環境変数を使用する方法を示しています。 `efm encrypt` コマンドを呼び出す前に、EFMPASS の値をパスワード (`1safepassword`) に設定します:

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

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

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

暗号化されたパスワード（0）はテキスト値として返されます。スクリプトを使用する場合、コマンドの終了コードをチェックして、コマンドが成功したことを確認できます。実行に成功すると 0 が返されます。

4.2 クラスターメンバーファイル

`lvariable_prod_name`の各ノード `cluster` には、現在の FailoverManager クラスターメンバーのリストを含むクラスターメンバーファイル（デフォルトでは、`efm.nodes` という名前）があります。エージェントが起動すると、その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。`lvariable_prod_name`インストーラーは、`/etc/edb/efm-3.10` ディレクトリに `efm.nodes.in` という名前のクラスターメンバーファイルのファイルテンプレートを作成します。

`lvariable_prod_name`の完了後インストールでは、テンプレートの作業用コピーを作成する必要があります。

```
# cp /etc/edb/efm-lvariable_prod_version/efm.nodes.in /etc/edb/efm-lvariable_prod_
↪version/efm.nodes
```

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

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

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

起動した最初のノードのクラスターメンバーファイルは空にできます。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、複数のエントリを空白で区切って、`address:port` の形式でリストする必要があります。

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

メンバーシップコーディネーターがクラスターを離れると、別のノードが役割を引き継ぎます。:ref:`efmcluster-status` を使用できます<efm_cluster_status>‘メンバーシップコーディネーターのアドレスを検索するコマンド。エージェントが停止しているときにノードがクラスターに参加または離脱する場合、ファイルに少なくとも現在のメンバーシップコーディネーターが含まれていることを手動で確認する必要があります。

クラスターに参加するノードの IP アドレスとポートがわかっている場合は、いつでもクラスターメンバーファイルにアドレスを含めることができます。起動時に、クラスターメンバーを識別しないアドレスは、（クラスタープロパティファイル内の）“`auto.allow.hosts`” プロパティでない限り無視されます<auto.allow.hosts>‘）は `true` に設定されます。

`stable.nodes.file` プロパティ（`cluster` プロパティファイルにあります）が `true` に設定されている場合、メンバーシップコーディネーターは、クラスターメンバーがクラスターに参加または離脱するときに `.nodes` ファイルを更新しません。この動作は、クラスターメンバーの IP アドレスが頻繁に変更されない場合に最も役立ちます。

4.3 フェールオーバーマネージャーの権限の拡張

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

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

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

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

`efm-310` ファイルは `/etc/sudoers.d` にあり、以下のエントリを含んでいます:

```
# Copyright EnterpriseDB Corporation, 2014-2020. All Rights Reserved.
#
# Do not edit this file. Changes to the file may be overwritten
# during an upgrade.
#
# This file assumes you are running your efm cluster as user
# 'efm'. If not, then you will need to copy this file.
# Allow user 'efm' to sudo efm_db_functions as either 'postgres'
# or 'enterprisedb'. If you run your db service under a
# non-default account, you will need to copy this file to grant
# the proper permissions and specify the account in your efm
# cluster properties file by changing the 'db.service.owner'
# property.

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

(次のページに続く)

(前のページからの続き)

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

variable_prod_name1を使用している場合 postgres または enterprisedb 以外のユーザーが所有するクラスターを監視し、efm-310 ファイルのコピーを作成し、ユーザーがefm_functions にアクセスできるようにコンテンツを変更するクラスターを管理するスクリプト。

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

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

4.3.1 sudo を使用せずにフェイルオーバーマネージャーを実行する

デフォルトでは、`|variable_prod_name|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。`|variable_prod_name|`を設定する場合 `sudo` アクセスなしで実行するには、`root` アクセスが次のことを行うために必要であることに注意してください。

- `|variable_prod_name|`をインストールします RPM。
- `|variable_prod_name|`を実行するセットアップタスク。

`|variable_prod_name|`を実行するには `sudo` を使用しない場合、`|variable_prod_name|`に代わって管理機能を実行する権限を持つデータベースプロセスの所有者を選択する必要があります。ユーザーは、デフォルトのデータベーススーパーユーザー（たとえば、`enterprisedb` や `postgres`）または別の特権ユーザーである可能性があります。ユーザーを選択した後:

1. 次のコマンドを使用して、ユーザーを `efm` グループに追加します。

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

これにより、ユーザーは `/var/run/efm-3.10` および `/var/lock/efm-3.10` に書き込むことができます。

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

```
su - enterprisedb
```

```
cp /etc/edb/efm-|variable_prod_version|/efm.properties.in <directory/cluster_
↪name>.properties
```

```
cp /etc/edb/efm-|variable_prod_version|/efm.nodes.in <directory>/<cluster_
↪name>.nodes
```

Then, modify the cluster properties file, providing the name of the user in the `db.service.owner` property. You must also ensure that the `db.service.name` property is blank; without `sudo`, you cannot run services without root access.

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

```
/usr/edb/efm-|variable_prod_version|/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完全パスと名前

を指定します。デフォルト以外のユーザーがエージェントを制御している場合、または efm スクリプトを使用している場合は常に、ユーザーがプロパティファイルへのフルパスを提供する必要があることに注意してください。

新しいユーザーが`|variable_prod_name|`を管理できるようにするにはサービスとして、カスタムスクリプトまたはユニットファイルを提供する必要があります。

|variable_prod_name| `/usr/edb/efm-3.10/bin/secure/` に常駐する `manage-vip` という名前のバイナリを使用して、`sudo` 権限なしで VIP 管理操作を実行します。このスクリプトは、`setuid` を使用して、仮想 IP アドレスの管理に必要な特権を取得します。

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

セキュリティ上の理由から、`/usr/edb/efm-3.10/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。

|variable_prod_name|の使用に関する詳細情報 `sudo` なしで、訪問:

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

4.4 仮想 IP アドレスでのフェイルオーバーマネージャーの使用

variable_prod_name efm_address スクリプトを使用して、仮想 IP アドレスを割り当てまたは解放します。

注意: 仮想 IP アドレスは、多くのクラウドプロバイダーでサポートされていません。それらの環境では、別のメカニズム（AWS の ElasticIP アドレスなど）を使用する必要があります。これは、フェンシングまたはプロモーション後のスクリプトで必要に応じて変更できます。

デフォルトでは、スクリプトは次の場所にあります。

```
/usr/edb/efm-3.10/bin/efm_address
```

次のコマンドバリエーションを使用して、IPv4 または IPv6 IP アドレスを割り当てまたは解放します。

仮想 IPv4IP アドレスを割り当てるには:

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

仮想 IPv6IP アドレスを割り当てるには:

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

仮想アドレスを解放するには:

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

どこ:

<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、[クラスタープロパティファイルを参照してください](#)。。

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

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

VIP のテスト

`!variable_prod_name!`で仮想 IP (VIP) アドレスを使用する場合、フェイ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、実際のフェイ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が検出されます。次の手順では、`!variable_prod_name!`を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ています。

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

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

ノードから VIP に ping するように指示された場合、`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 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
```

パケット損失が 100 %になるはずですが、

2. マスターノードで `efm_address add4` コマンドを実行して VIP を割り当て、IP アドレスで確認します。

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

3. 他のノードから VIP に ping して、それらが 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
```

パケット損失はないはずですが、

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

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

このステップの出力には **eth0** インターフェイスが表示されないはずです

5. 手順 3 を繰り返します。今回は、スタンバイおよび監視に、使用中の **VIP** が表示されないことを確認します。

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

パケット損失が 100 % になるはずです。すべてのノードでこの手順を繰り返します。

6. すべてのスタンバイノードで手順 2 を繰り返し、すべてのノードに **VIP** を割り当てます。任意のノードから **VIP** に ping して、**VIP** が使用中であることを確認できます。

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

上記のテスト手順の後、**lvariable_prod_name1**を開始する前に、マスター以外のノードから **VIP** を解放します。

注意: **VIP** に使用されるネットワークインターフェースは、**lvariable_prod_name1**に使用されるインターフェースと同じである必要はありません。エージェントの `bind.address` 値。マスターエージェントはフェイルオーバー中に必要に応じて **VIP** をドロップし、**lvariable_prod_name1**スタンバイを昇格する前に、**VIP** が使用できないことを確認します。バインドアドレスネットワークに障害が発生すると、マスターの分離とフェイルオーバーが発生します。

VIP が別のインターフェースを使用している場合、マスターエージェントがドロップする前に、クラスターの残りの部分が到達可能な **VIP** をチェックするというタイミング条件が発生する可能性があります。この場合、EFM は `node.timeout` プロパティで指定された秒数の間 **VIP** チェックを再試行し、期待どおりにフェイルオーバーが発生することを確認します。

第 5 章

フェールオーバーマネージャーの使用

`|variable_prod_name|`は、1 つ以上のスタンバイサーバーを持つクラスターの監視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縮小するときに、クラスターからノードを追加または削除できます。

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

5.1 フェールオーバーマネージャークラスターの管理

設定したら、`|variable_prod_name|`クラスターは定期的なメンテナンスを必要としません。次のセクションでは、フェールオーバーマネージャークラスターで必要になる場合がある管理タスクの実行について説明します。

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

- `efmallow-node`
- `efmdisallow-node`
- `efm` プロモート
- `efm` 再開
- `efmset-priority`
- `efmstop-cluster`
- `efmupgrade-conf`

5.1.1 FailoverManager クラスターの起動

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

`|variable_prod_name|`を開始するには RHEL6.x または CentOS6.x でクラスター化し、スーパーユーザー権限を引き受け、次のコマンドを呼び出します。

```
service edb-efm-|variable_prod_version| start
```

`|variable_prod_name|`を開始するには RHEL/CentOS7.x または RHEL/CentOS8.x でクラスターを作成し、スーパーユーザー特権を引き受けて、次のコマンドを呼び出します。

```
systemctl start edb-efm-|variable_prod_version|
```

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ます。

ノードが専用の Witness ノードでない場合、`|variable_prod_name|`ローカルデータベースに接続して `pg_is_in_recovery()` 関数を呼び出します。サーバーが `false` に応答すると、エージェントはノードがマスターノードであると想定し、仮想 IP アドレスをノードに割り当てます（該当する場合）。サーバーが `true` と応答した場合、`|variable_prod_name|`エージェントは、ノードがスタンバイサーバーであると想定します。サーバーが応答しない場合、エージェントはアイドル状態で起動します。

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

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

5.1.2 クラスターへのノードの追加

`|variable_prod_name|`にノードを追加できますいつでもクラスター。クラスターにノードを追加するときは、新しいノードを許可するようにクラスターを変更してから、新しいノードにクラスターの検索方法を伝える必要があります。次の手順では、クラスターへのノードの追加について詳しく説明します。

1. `auto.allow.hosts` が `true` に設定されていない限り、`efm allow-node` コマンドを使用して、新しいノードの IP アドレスをフェイルオーバーマネージャーの許可ノードホストリストに追加します。コマンドを呼び出すときに、新しいノードのクラスター名と IP アドレスを指定します。

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

`efm allow-node` コマンドの使用または `|variable_prod_name|` の制御の詳細についてはサービス、[EFM ユーティリティの使用を参照](#)。

`lvariable_prod_name`をインストールしますエージェントと新しいノードの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、“[クラスタープロパティファイル](#)”を参照してください。

2. 新しいノードでクラスターメンバーファイルを構成し、MembershipCoordinator のエントリを追加します。クラスターメンバーファイルの変更の詳細については、“[クラスターメンバーファイル](#)”を参照してください。
3. 新しいノードでスーパーユーザー権限を引き受け、フェイルオーバーマネージャーエージェントを起動します。`lvariable_prod_name`を開始するには RHEL6.x または CentOS6.x でクラスター化し、スーパーユーザー権限を引き受け、次のコマンドを呼び出します。

```
service edb-efm-lvariable_prod_version start
```

`lvariable_prod_name`を開始するには RHEL/CentOS7.x または RHEL/CentOS8.x でクラスターを作成し、スーパーユーザー特権を引き受けて、次のコマンドを呼び出します。

```
systemctl start edb-efm-lvariable_prod_version
```

新しいノードがクラスターに参加すると、`lvariable_prod_name` `user.email` プロパティで提供される管理者の電子メールに通知を送信するか、指定された通知スクリプトを呼び出します。

注意: 現在のノードで有効なスタンバイになるためには、ノードが PostgreSQL ストリーミングレプリケーションシナリオでスタンバイである必要があります。

5.1.3 スタンバイの優先度の変更

`lvariable_prod_name`クラスターに複数のスタンバイサーバーが含まれている場合、`efm set-priority` コマンドを使用して、スタンバイノードの昇格の優先度に影響を与えることができます。`lvariable_prod_name`の既存のメンバーでコマンドを呼び出すクラスター、およびメンバーの IP アドレスの後に優先度の値を指定します。

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

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

スタンバイの優先度を 0 に設定して、スタンバイを非昇格にすることができます。スタンバイの優先度を 0 より大きい値に設定すると、プロパティ値 `promotable=false` が上書きされます。

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

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

フェイルオーバーが発生した場合、`variable_prod_name`最初に Postgres ストリーミングレプリケーションから情報を取得して、最新のデータがあるスタンバイノードを確認し、データ損失の可能性が最も低いノードを昇格させます。2つのスタンバイノードに等しく最新のデータが含まれている場合、`use.replay.tiebreaker`を除き、ユーザー指定の優先度の値が高いノードがマスターに昇格されます。`<use.replay.tiebreaker>`は `‘false’` に設定されません。スタンバイノードの優先度の値を確認するには、次のコマンドを使用します。

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

注意: ノードがクラスターから分離され、後でクラスターに再度参加した場合、プロモーションの優先度が変わる可能性があります。

5.1.4 フェイルオーバーマネージャーノードの昇格

`variable_prod_name`の任意のノードで `efm promote` を呼び出すことができます。クラスターを使用して、スタンバイデータベースからマスターデータベースへの手動昇格を開始します。

手動昇格は、データベースクラスターのメンテナンス期間中のみ実行する必要があります。最新のスタンバイデータベースがない場合は、続行する前にプロンプトが表示されます。手動昇格を開始するには、`efm` または `OS` スーパーユーザーの ID を想定して、次のコマンドを呼び出します。

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

どこ:

`<cluster_name>` は `variable_prod_name` の名前です。集まる。

Include the `-switchover` option to reconfigure the original Master as a Standby. If you include the `-switchover` keyword, the cluster must include a master node and at least one standby, and the nodes must be in sync.

リカバリ設定のマスターからのコピー元のノードを指定するには、`-sourcenode` キーワードを含めます。

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

Include the `-noscripts` keyword to prevent instruct `variable_prod_name` to not invoke fencing and post-promotion scripts.

スイッチオーバー中:

- サーバーバージョン 11 以前の場合、`recovery.conf` ファイルは既存のスタンバイからマスターノードにコピーされます。サーバーバージョン 12 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。
- マスターデータベースが停止します。
- VIP を使用している場合、アドレスはマスターノードから解放されます。

- マスター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。
- 新しいマスターノードのアドレスが `recovery.conf` ファイルに追加されるか、`primary_conninfo` の詳細がメモリに保存されます。
- このノードに `application.name` プロパティが設定されている場合、`application_name` プロパティが `recovery.conf` ファイルに追加されるか、または `primary_conninfo` 情報がメモリに格納されます。
- サーバーバージョン 12 以降を使用している場合、メモリに保存されているリカバリ設定は `postgresql.auto.conf` ファイルに書き込まれます。
- 古いマスターが開始されます。エージェントはそれをスタンバイとして監視を再開します。

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

マスターエージェントは実行中のままで、ステータスは `Idle` になります。

スタンバイエージェントは、既知のアドレスに `ping` する前に仮想 IP アドレスが使用されていないことを確認して、エージェントがネットワークから分離されていないことを確認します。スタンバイエージェントはフェンシングスクリプトを実行し、スタンバイデータベースをマスターに昇格させます。次に、スタンバイエージェントは仮想 IP アドレスをスタンバイノードに割り当て、プロモーション後のスクリプトを実行します（該当する場合）。

このコマンドは、クラスタープロパティファイルの `auto.failover` パラメーターで指定された値を無視するようにサービスに指示することに注意してください。

ノードをマスターの役割に戻すには、ノードをプロモーションリストの最初に配置します。

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

次に、手動プロモーションを実行します。

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

`efm` ユーティリティの詳細については、[EFM ユーティリティの使用を参照してください](#)。。

5.1.5 FailoverManager エージェントの停止

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

`!variable_prod_name!` を停止するには RHEL6.x または CentOS6.x のエージェントで、スーパーユーザー権限を引き受け、次のコマンドを実行します。

```
service edb-efm-|variable_prod_version| stop
```

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

```
systemctl stop edb-efm-|variable_prod_version|
```

efmdisallow-node コマンドを呼び出す（許可されたノードのホストリストからノードのアドレスを削除する）まで、service edb-efm-3.10 start コマンドを使用して、最初に efm allow-node を実行せず、後でノードを再起動できます。コマンドをもう一度。

master.shutdown.as.failure でない限り、エージェントを停止しても、エージェントが失敗したことをクラスタに通知しないことに注意してください。<master_shutdown_as_failure> プロパティは ‘true’ に設定されます。

5.1.6 フェールオーバーマネージャークラスターの停止

|variable_prod_name|を停止するにはクラスター、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、ID が efm または OS スーパーユーザーであると想定して、次のコマンドを呼び出します：

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

コマンドは *all*|variable_prod_name|を引き起こします終了するエージェント。|variable_prod_name|を終了していますエージェントはすべてのフェイルオーバー機能を完全に無効にします。

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

5.1.7 クラスターからのノードの削除

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

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

efm disallow-node コマンドは実行中のエージェントを停止しません。エージェントを停止するまで、サービスはノードで実行され続けます。その後エージェントまたはクラスターが停止した場合、ノードはクラスターに再参加することができず、フェイルオーバー優先順位リストから削除されます（昇格の対象外になります）。

efm disallow-node コマンドを呼び出した後、:ref:`efmallow-node` を使用する必要があります <efm_allow_node> ‘ノードをクラスターに再度追加するコマンド。

5.2 単一ノードでの複数のエージェントの実行

`lvariable_prod_name1`で複数のマスターまたはスタンバイエージェントを実行することにより、同じホスト上にある複数のデータベースクラスターを監視できます。ノード。単一のノードで複数の監視エージェントを実行することもできます。`lvariable_prod_name1`を設定するには`lvariable_prod_name1`を確認しながら、複数のデータベースクラスターを監視する異なるクラスターのエージェントは相互に干渉しません。

1. 各クラスターのメンバーごとに、固有の一連のプロパティとクラスター内のノードの役割を定義するクラスタープロパティファイルを作成します。
2. クラスターのメンバーをリストする各クラスターのメンバーごとにクラスターメンバーファイルを作成します。
3. 各クラスターのサービススクリプト（RHEL または CentOS6.x システム）またはユニットファイル（RHEL/CentOS7.x または RHEL/CentOS8.x システム）をカスタマイズして、クラスタープロパティの名前とクラスターメンバーファイル。
4. 各クラスターのサービスを開始します。

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

- `acctg` のデータは `/opt/pgdata1` にあります; そのサーバーはポート 5444 を監視しています。
- `sales` のデータは `/opt/pgdata2` にあります; そのサーバーはポート 5445 を監視しています。

`lvariable_prod_name1`を実行するにはこれらの両方のデータベースクラスターのエージェントでは、`efm.properties.in` テンプレートを使用して 2 つのプロパティファイルを作成します。各クラスタープロパティファイルには一意の名前を付ける必要があります。この例では、`acctg.properties` と `sales.properties` を作成して、`acctg` と `sales` のデータベースクラスターに一致させます。

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

```
admin.port

bind.address

db.port

db.data.dir

virtual.ip （使用する場合）

virtual.ip.interface （使用する場合）
```

各クラスタープロパティファイル内で、`db.port` パラメータは各クラスターの一意の値を指定する必要がありますが、`db.user` および `db.database` パラメータは同じ値または一意の値を持つことができます。例えば、`acctg.properties` ファイルは以下を指定するかもしれません:

```
db.user=efm_user
```

```
db.password.encrypted=7c801b32a05c0c5cb2ad4ffbdba5e8f9a
```

```
db.port=5444
```

```
db.database=acctg_db
```

sales.properties ファイルは以下を指定できます:

```
db.user=efm_user
```

```
db.password.encrypted=e003fea651a8b4a80fb248a22b36f334
```

```
db.port=5445
```

```
db.database=sales_db
```

`!variable_prod_name!`を複数設定する場合、一部のパラメーターには特別な注意が必要です。同じノード上のクラスターエージェント。複数のエージェントが同じノードに存在する場合、各ポートは一意である必要があります。どの2つのポートでも機能しますが、相互に近すぎないポートを使用する場合は、情報を明確にしておく方が簡単な場合があります。

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

ノードに仮想 IP アドレスを割り当てるときに、次のパラメーターが使用されます。`!variable_prod_name!`クラスターは仮想 IP アドレスを使用しないため、これらのパラメーターは空白のままにします。

```
virtual.ip
```

```
virtual.ip.interface
```

```
virtual.ip.prefix
```

このパラメータ値は、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` の両方で同じである場合とそうでない場合があります。

`acctg.properties` と `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。このステップはプラットフォーム固有です。RHEL6.x または CentOS6.x を使用している場合は、[RHEL6.x](#) または [CentOS6.x](#) を参照してください。
;RHEL/CentOS7.x または RHEL/CentOS8.x を使用している場合は、[RHEL/CentOS7.x](#) または [RHEL/CentOS8.x](#) を参照してください。。

注意: カスタムサービススクリプトまたはユニットファイルを使用している場合、`!variable_prod_name!`をアップグレードするときに、新しいサービス名を反映するようにファイルを手動で更新する必要があります。

5.2.1 RHEL6.x または CentOS6.x

RHEL6.x または CentOS6.x を使用している場合は、`edb-efm-3.10` サービススクリプトを、クラスターごとに一意の名前で新しいファイルにコピーする必要があります。例えば:

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

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

サービススクリプトを作成したら、次のコマンドを実行します。

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

次に、新しいサービススクリプトを使用してエージェントを起動します。たとえば、次のコマンドで `acctg` エージェントを起動できます:

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

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

RHEL/CentOS7.x または RHEL/CentOS8.x を使用している場合は、`edb-efm-3.10` ユニットファイルを、クラスターごとに一意の名前で新しいファイルにコピーする必要があります。たとえば、2つのクラスター (`acctg` と `sales` という名前) がある場合、ユニットファイル名は次のようになります。

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

次に、各ユニットファイル内の CLUSTER 変数を編集して、指定したクラスター名を `efm` から新しいクラスター名に変更します。たとえば、`acctg` という名前のクラスターの場合、値は次のように指定します。

```
Environment=CLUSTER=acctg
```

新しいクラスター名を指定するには、PIDfile パラメーターの値も更新する必要があります。例えば:

```
PIDFile=/var/run/efm-|variable_prod_version|/acctg.pid
```

サービススクリプトをコピーした後、次のコマンドを使用してサービスを有効にします。

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

次に、新しいサービススクリプトを使用してエージェントを起動します。たとえば、次のコマンドで acctg エージェントを起動できます:

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

ユニットファイルのカスタマイズについては、次の URL をご覧ください。

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

第 6 章

フェールオーバーマネージャークラスターの監視

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

6.1 クラスタステータスレポートの確認

`efmcluster-status :ref:‘クラスタープロパティファイル<efm_cluster_status>’` コマンドは `variable_prod_name!` のステータスに関する情報を含むレポートを返します。コマンドを呼び出すには、次のように入力します。

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

次のステータスレポートは、3 つのノードが実行されている `edb` という名前のクラスターに関するものです。

Agent	Type	Address	Agent	DB	VIP
Standby		172.19.10.2	UP	UP	
Standby		172.19.12.163	UP	UP	
Master		172.19.14.9	UP	UP	
Allowed node host list:					
172.19.14.9 172.19.12.163 172.19.10.2					
Membership coordinator: 172.19.14.9					
Standby priority host list:					
172.19.12.163 172.19.10.2					

(次のページに続く)

(前のページからの続き)

Promote Status:

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

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

クラスター状況セクションには、クラスターの各ノードに常駐するエージェントの状況の概要が表示されます。

Agent Type	Address	Agent	DB	VIP
Standby	172.19.10.2	UP	UP	
Standby	172.19.12.163	UP	UP	
Master	172.19.14.9	UP	UP	

VIP アドレスの後のアスタリスク (*) は、そのアドレスが接続に使用できることを示します。VIP アドレスの後にアスタリスクが付いていない場合、そのアドレスは（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、そのアドレスは現在使用されていません。

`|variable_prod_name|` エージェントは、[ClusterStatus] セクションに表示される情報を提供します。

Allowed node host list と Standby priority host list は、どのノードがクラスターへの参加を許可されているか、およびノードの昇格順序を簡単に知る方法を提供します。メンバーシップコーディネーターの IP アドレスもレポートに表示されます。

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

レポートの Promote Status セクションは、`cluster-status` コマンドを呼び出しているノードからクラスター内の各データベースへの直接クエリの結果です。クエリは、各データベースのトランザクションログの場所も返します。

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

データベースが停止している場合（またはデータベースが再起動されているが、再開コマンドがまだ呼び出されていない場合）、そのホストに存在するエージェントの状態はアイドルになります。エージェントがアイドル状態の場合、クラスタ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。例えば:

Agent	Type	Address	Agent	DB	VIP

Idle		172.19.18.105	UP	UP	172.19.13.105

終了コード

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

- 終了コード 0 は、すべてのエージェントが実行中で、マスターノードとスタンバイノードのデータベースが実行中で同期していることを示します。
- ゼロ以外の終了コードは、問題があることを示します。次の問題により、ゼロ以外の終了コードがトリガーされる可能性があります。

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

`|variable_prod_name|`提供されたデータベースパスワードを復号化できません。

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。

マスターエージェントはありません。

スタンバイエージェントはありません。

1 つ以上のスタンバイノードがマスターと同期していません。

6.2 PostgresEnterpriseManager によるストリーミングレプリケーションの監視

PostgresEnterpriseManager (PEM) を使用してサーバーを監視する場合、ストリーミングレプリケーション分析ダッシュボード (PEM グラフィカルインターフェイスの一部) を構成して、ストリーミングレプリケーションシナリオの一部であるマスターノードまたはスタンバイノードの状態を表示できます。

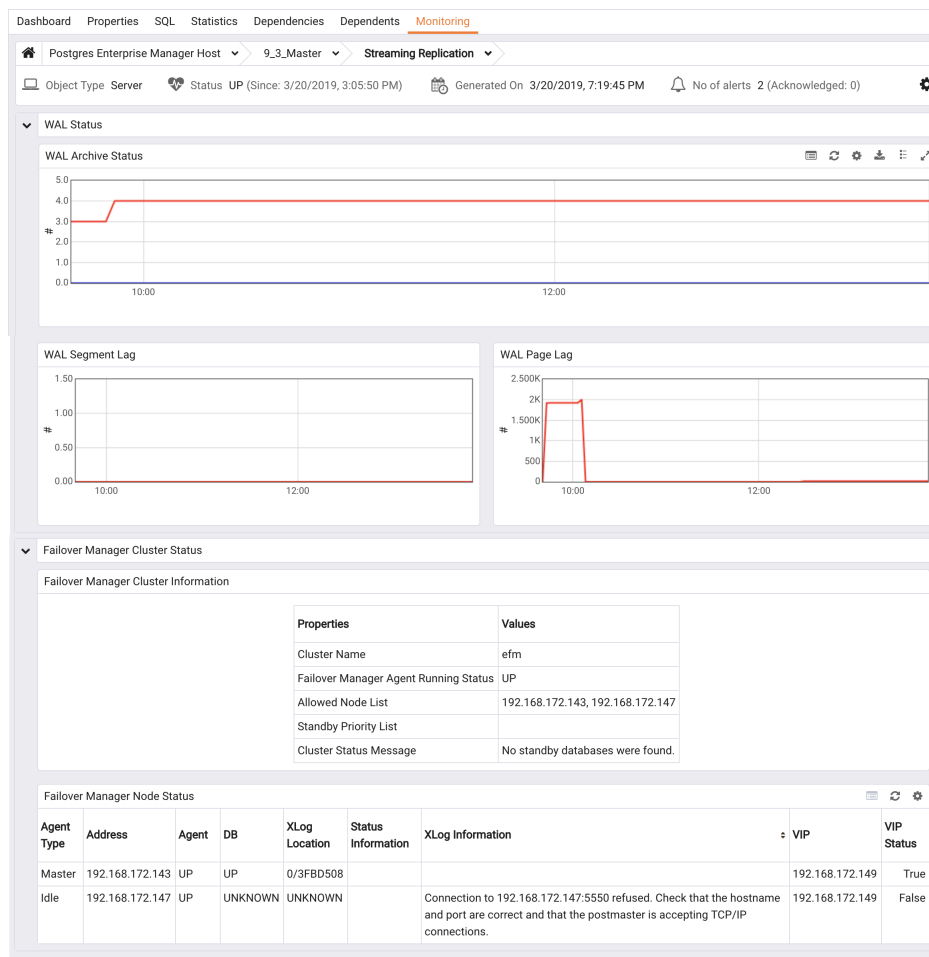


図 1 ストリーミングレプリケーションダッシュボード (マスターノード)

ストリーミングレプリケーション分析ダッシュボードには、ストリーミングレプリケーションが有効になっている監視対象サーバーのアクティビティに関する統計情報が表示されます。ダッシュボードヘッダーは、監視対象サーバー (レプリケーションマスターまたはレプリケーションスレーブ) のステータスを識別し、サーバーが最後に起動された日時、ページが最後に更新された日時、およびトリガーされたアラートの現在の数を表示しますサーバー用。

レプリケーションスレーブ (スタンバイノード) のダッシュボードを確認すると、ダッシュボードの下部にあるラベルでサーバーのステータスを確認できます。

デフォルトでは、ストリーミングレプリケーション分析ダッシュボードの情報を提供する PEM レプリケーション

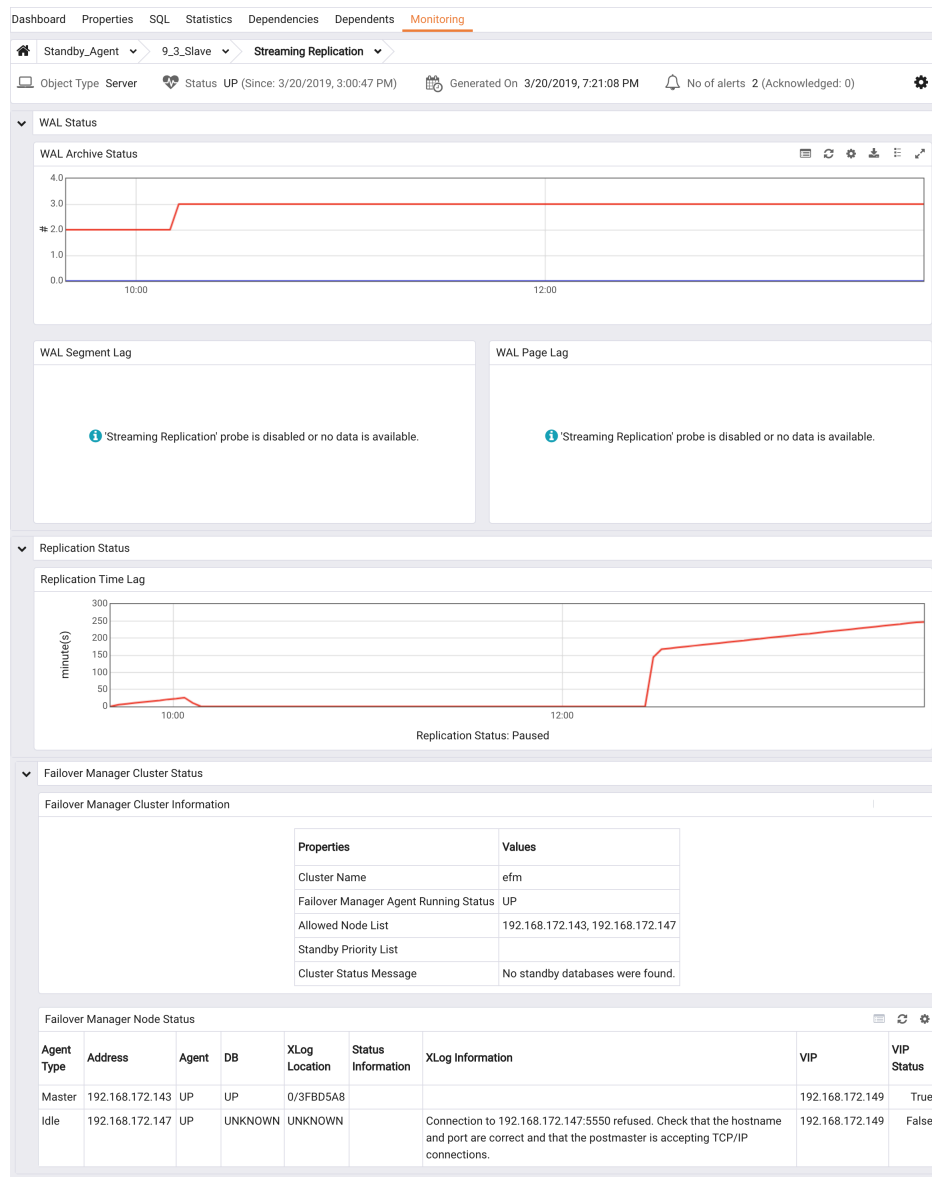


図 2 ストリーミングレプリケーションダッシュボード（スタンバイノード）

プローブは無効になっています。

レプリケーションシナリオのマスターノードのストリーミングレプリケーション分析ダッシュボードを表示するには、以下のプローブを有効にする必要があります。

- ストリーミングレプリケーション
- WAL アーカイブステータス

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

- ストリーミングレプリケーションラグタイム

PEM の詳細については、次の EnterpriseDBWeb サイトにアクセスしてください。

<http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager>

第 7 章

efm ユーティリティの使用

`!variable_prod_name!` 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。`!variable_prod_name!` をインストールすると、RPM インストーラーはユーティリティを `/usr/edb/efm-3.10/bin` ディレクトリに追加します。

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

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

このコマンドは、efm、efm グループのメンバー、または root が呼び出す必要があります。

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

Invoke the `efm cluster-status` command to display the status of a `!variable_prod_name!` cluster. For more information about the 3.10 status report, see *Monitoring a Failover Manager Cluster*.

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

`efm cluster-status-json` コマンドを呼び出して、`!variable_prod_name!` のステータスを表示します。json 形式のクラスター。表示される情報の形式は、`efmcluster-status` コマンドによって生成される表示とは異なりますが、情報ソースは同じです。

次の例は、2 つのノードを持つ正常なクラスターのステータスをクエリすることによって生成されます。

```
{
  "nodes": {
    "172.16.144.176": {
      "type": "Witness",
      "agent": "UP",
      "db": "N\A",
      "vip": "",
      "vip_active": false
    },
    "172.16.144.177": {
```

(次のページに続く)

(前のページからの続き)

```

        "type": "Master",
        "agent": "UP",
        "db": "UP",
        "vip": "",
        "vip_active      :      false"
        "xlogReceive     :      0/14001478"
        "xlog      :      0/14001478"
        "xloginfo      :      "
    },
    "172.16.144.180": {
        "type": "Standby",
        "agent": "UP",
        "db": "UP",
        "vip": "",
        "vip_active      :      false"
        "xlogReceive     :      0/14001478"
        "xlog      :      0/14001478"
        "xloginfo      :      "
    }
},
"allowednodes": [
    "172.16.144.177",
    "172.16.144.160",
    "172.16.144.180",
    "172.16.144.176"
],
"membershipcoordinator": "172.16.144.177",
"failoverpriority": [
    "172.16.144.180"
],
"minimumstandbys": 0,
"missingnodes": [],
"messages": []
}

```

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

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

```
efm 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。`variable_prod_name`に指示する `--from-env` オプションを含めます `EFMPASS` 環境変数で指定された値を使用し、ユーザー入力なしで実行します。詳細については、[データ](#)

ベースパスワードの暗号化を参照してください。。

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

efm promote コマンドは `variable_prod_name1` に指示しますスタンバイからマスターへの手動フェイルオーバーを実行します。

手動昇格は、マスターに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 コマンドによって報告された場合にのみ試行してください。最新のスタンバイがない場合、`variable_prod_name1` 続行する前にプロンプトが表示されます。

`-switchover` 句を含めてスタンバイノードを昇格させ、マスターノードをスタンバイノードとして再構成します。`-sourcenode` キーワードを含め、ノードアドレスを指定して、`recovery.conf` ファイルを古いマスターノードにコピーする（スタンバイにする）ノードを指定します。スイッチオーバープロセス中の通知を抑制するには、`-quiet` キーワードを含めます。`variable_prod_name1` に指示する `-noscripts` キーワードを含めますフェンシングスクリプトやプロモーション後のスクリプトを呼び出さないようにします。

このコマンドは、efm、efm グループのメンバー、または root が呼び出す必要があります。

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

```
efm resume <cluster_name>
```

efm resume コマンドを呼び出して、以前に停止したデータベースの監視を再開します。このコマンドは、efm、efm グループのメンバー、または root が呼び出す必要があります。

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

efm set-priority コマンドを呼び出して、スタンバイノードにフェイルオーバー優先度を割り当てます。この値は、フェイルオーバー時に新しいノードが使用される順序を指定します。このコマンドは、efm、efm グループのメンバー、または root が呼び出す必要があります。

priority は 1 ~ ***n*** の整数値で、***n*** はリスト内のスタンバイノードの数です。値 1 を指定して、新しいノードがプライマリスタンバイであり、フェイルオーバー時に最初に昇格するノードであることを示します。優先度値 0 は `variable_prod_name1` に指示しますスタンバイを昇格させないため。

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

efm stop-cluster コマンドを呼び出して停止します `variable_prod_name1` すべてのノード。このコマンドは `variable_prod_name1` に指示しますクラスターの各ノードに接続し、既存のメンバーにシャットダウンするように指示します。このコマンドは実行中のデータベースには影響しませんが、コマンドが完了すると、フェイルオーバー保護は適用されません。

注意: `efmstop-cluster` コマンドを呼び出すと、許可されたすべてのノード情報が許可ノードホストリストから削除されます。

このコマンドは、efm、efm グループのメンバー、または root が呼び出す必要があります。

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

efmupgrade-conf コマンドを呼び出して、既存の`variable_prod_name1`から構成ファイルをコピーしますインストール、`variable_prod_name1`で必要なパラメーターを追加インストール。ユーティリティを呼び出すときに、前のクラスターの名前を指定します。このコマンドは、root 権限で呼び出す必要があります。

`variable_prod_name1`からアップグレードする場合 `sudo` を使用しない設定には、`-source` フラグを含め、`upgrade-conf` を呼び出すときに設定ファイルが存在する*ディレクトリ*の名前を指定します。

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

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

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

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

```
efm --help
```

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

第 8 章

フェールオーバーマネージャサービスの制御

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

- `!variable_prod_name!`で使用するプロパティを含む `efm.properties` という名前の設定ファイルサービス。レプリケーションシナリオの各ノードには、ノードに関する情報を提供するプロパティファイルが含まれている必要があります。
- クラスターメンバーのリストを含む `efm.nodes` という名前のクラスターメンバーファイル。レプリケーションシナリオの各ノードには、クラスターメンバーリストが含まれている必要があります。

単一ノードで複数のクラスターを実行している場合は、クラスター固有の名前で構成ファイルを手動で作成し、対応するクラスターのサービススクリプトを変更する必要があることに注意してください。

`!variable_prod_name!`を制御するコマンドサービスはプラットフォーム固有です。

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

RHEL6.x および CentOS6.x では、`|variable_prod_name|`（デフォルトで `edb-efm-|variable_prod_version|`）という名前の Linux サービスとして実行されます `/etc/init.d` にあります。`|variable_prod_name|`によって監視される各データベースクラスターレプリケーションクラスターの各ノードでサービスのコピーを実行します。

次のサービスコマンドを使用して、`|variable_prod_name|`を制御します。RHEL6.x または CentOS6.x ホストにあるエージェント:

```
service edb-efm-|variable_prod_version| start
```

`start` コマンドは `|variable_prod_name|`を開始します現在のノードのエージェント。ローカルの `|variable_prod_name|` エージェントはローカルデータベースを監視し、`|variable_prod_name|`と通信します他のノードで。`|variable_prod_name|`でノードを開始できます任意の順序でクラスター化します。このコマンドは `root` が呼び出す必要があります。

```
service edb-efm-|variable_prod_version| stop
```

`|variable_prod_name|`を停止します現在のノード。このコマンドは `root` が呼び出す必要があります。

```
service edb-efm-|variable_prod_version| status
```

`status` コマンドは、`|variable_prod_name|`のステータスを返します。呼び出されるエージェント。`|variable_prod_name|`に指示するために、任意のノードで `status` コマンドを呼び出すことができます。ステータス情報を返します。例えば:

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

```
service edb-efm-|variable_prod_version| help
```

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

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

On RHEL/CentOS 7.x and RHEL/CentOS 8.x, `|variable_prod_name|` runs as a Linux service named (by default) `edb-efm-3.10.service` that is located in `/usr/lib/systemd/system`. Each database cluster monitored by Failover Manager will run a copy of the service on each node of the replication cluster.

次の `systemctl` コマンドを使用して、`|variable_prod_name|`を制御します。RHEL/CentOS7.x および RHEL/CentOS8.x ホストにあるエージェント:

```
systemctl start edb-efm-|variable_prod_version|
```

`start` コマンドは`|variable_prod_name|`を開始します現在のノードのエージェント。ローカルの`|variable_prod_name|`エージェントはローカルデータベースを監視し、`|variable_prod_name|`と通信します他のノードで。`|variable_prod_name|`でノードを開始できます任意の順序でクラスター化します。このコマンドは `root` が呼び出す必要があります。

```
systemctl stop edb-efm-|variable_prod_version|
```

`|variable_prod_name|`を停止します現在のノード。このコマンドは `root` が呼び出す必要があります。

```
systemctl status edb-efm-|variable_prod_version|
```

`status` コマンドは、`|variable_prod_name|`のステータスを返します。呼び出されるエージェント。`|variable_prod_name|`に指示するために、任意のノードで `status` コマンドを呼び出すことができます。ステータスとサーバーの起動情報を返します。

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


第 9 章

ロギングの制御

`variable_prod_name` エージェントごとに 1 つのログファイルを書き込み、エージェントごとに 1 つの起動ログを `/var/log/<cluster_name>-3.10` に書き込み、保存します（`<cluster_name>` はクラスターの名前を指定します）。

クラスタープロパティファイルの `‘jgroups.loglevel’` および `efm.loglevel` パラメーターを変更することで、エージェントログに書き込まれる詳細のレベルを制御できます `<loglevel>‘`：

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

ロギング機能は、Java ロギングライブラリとロギングレベルを使用します。ログレベル（ほとんどのログ出力から最小の順に）は次のとおりです。

- TRACE
- DEBUG
- INFO
- WARN
- ERROR

たとえば、`efm.loglevel` パラメーターを `WARN` に設定した場合、フェイルオーバーマネージャーは `WARN` 以上のレベル（`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、`variable_prod_name` ログファイルは毎日ローテーションされ、圧縮され、1 週間保存されます。ログローテーションファイル（`/etc/logrotate.d/efm-3.10`）の設定を変更することで、ファイルローテーションスケジュールを変更できます。ログローテーションスケジュールの変更の詳細については、`logrotate` のマニュアルページを参照してください。

```
$ man logrotate
```

9.1 Syslog ログファイルエントリの有効化

`!variable_prod_name!syslog` ロギングをサポートします。syslog ロギングを実装するには、UDP または TCP 接続を許可するように syslog を構成する必要があります。

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

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

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

```
# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

syslog 構成ファイルを変更した後、rsyslog サービスを再起動して接続を有効にします。

```
systemctl restart rsyslog.service
```

`!variable_prod_name!` の `rsyslog.conf` ファイルを変更した後ホスト、`!variable_prod_name!` を変更する必要がありますログを有効にするプロパティ。選択したエディターを使用して、プロパティファイルを変更します `<logtype.enabled>` (`'/etc/edb/efm-3.10/efm.properties.in'`) は、実装したいロギングのタイプを指定します:

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

syslog の詳細も指定する必要があります `<syslog.logging>` システム用。プロトコルタイプ (UDP または TCP) を指定するには `'syslog.protocol'` パラメータを使用し、syslog ホストのリッスナーポートを指定するには `syslog.port` パラメータを使用します。 `syslog.facility` の値は、エントリを作成したプロセスの識別子として使用できます。値は `LOCAL0` と `LOCAL7` の間でなければなりません。

```
# Syslog information. The syslog service must be listening # on the
port for the given protocol, which can be UDP or
# TCP. The facilities supported are LOCAL0 through LOCAL7.
# syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

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

syslog man

第 10 章

通知

`variable_prod_name1` クラスターに影響を与える重要なイベントが発生すると、電子メール通知を送信したり、通知スクリプトを呼び出したりします。`variable_prod_name1` を設定した場合電子メール通知を送信するには、クラスターの各ノードのポート 25 で SMTP サーバーを実行している必要があります。次のパラメーターを使用して、`variable_prod_name1` の通知動作を構成します。

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

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

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

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

VIP フィールドは、ノードに実装されている場合、仮想 IP の IP アドレスと状態を表示します。

`variable_prod_name1` 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増加していることを示しています。

INFO はエージェントに関する情報メッセージを示し、手動による介入は必要ありません（たとえば、`variable_prod_name1` が開始または停止しました）。

WARNING は、管理者がシステムを確認する必要があるイベントが発生したことを示します（たとえば、フェイルオーバーが発生した）。

SEVERE は、重大なイベントが発生し、管理者の即時の注意が必要であることを示します（たとえば、フェイルオーバーが試行されたが、完了できなかった）。

重大度レベルは、通知の緊急度を指定します。重大度が SEVERE の通知は、ユーザーの注意をすぐに必要としますが、重大度が INFO の通知は、ユーザーのアクションを必要としないクラスターの運用情報に注意を喚起します。通知の重大度レベルは、ロギングレベルとは関係ありません。構成ファイルで指定されたログレベルの詳細に関係なく、すべての通知が送信されます。

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

注意: :管理用メールアドレスへの通知の送信に加えて、すべての通知はクラスターログファイル（ /var/log/efm-3.10/<cluster_name>.log ）に記録されます。

以下の表にリストされている条件は、 INFO レベルの通知をトリガーします:

件名	説明
実行されたフェンシングスクリプト	実行されたフェンシングスクリプト*script_name*結果:script_results
実行されたプロモーション後のスクリプト	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
実行されたりモート事前プロモーションスクリプト	実行されたりモート事前プロモーションスクリプト*script_name*結果:script_results
実行されたりモートポストプロモーションスクリプト	実行されたりモートポストプロモーションスクリプト*script_name*結果:script_results
実行されたデータベース障害スクリプト	実行されたデータベース障害スクリプト*script_name*結果:script_results
実行されたマスター分離スクリプト	実行されたマスター分離スクリプト*script_name*結果:script_results
クラスター*cluster_name*の*node_address*で実行されている監視エージェント	監視エージェントが実行されています。
クラスター*cluster_name*の*node_address*で実行されているマスターエージェント	マスターエージェントが実行され、データベースの状態が監視されています。
クラスター*cluster_name*の*node_address*で実行中のスタンバイエージェント	スタンバイエージェントが実行中で、データベースの状態が監視されています。
クラスター*cluster_name*のノード*node_address*で実行されているアイドルエージェント	アイドルエージェントが実行されています。ローカルデータベースを起動した後、エージェントを再開できます。
ノード*node_address*へのVIPの割り当て	VIP*VIP_address*をノード*node_address*に割り当てています結果:script_results
ノード*node_address*からVIPを解放しています	ノード*node_address*からVIP*VIP_address*を解放しています結果:script_results
クラスター*cluster_name*の自動再開チェックを開始しています	このノードのエージェントは、*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監視を再開できるかどうかを確認します。この間、クラスターを確認し、データベースを再起動しない場合はエージェントを停止する必要があります。詳細については、エージェントログを参照してください。
実行されたエージェントがスクリプトを再開しました	実行されたエージェントがスクリプトを再開しました*script_name*結果:script_results
プロモーション中にバックアップされたWALログ	新しいスタンバイ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。このバックアップは、ディスク領域を解放するのに都合がよいときに削除する必要があります。

以下の表にリストされている条件により、*警告*レベルの通知がトリガーされます。

件名	説明
監視エージェントは、クラスター*cluster_name*の*node_address*で終了しました	監視エージェントが終了しました。
クラスター*cluster_name*の*node_address*でマスターエージェントが終了しました	データベースの状態は監視されていません。
クラスター*cluster_name*はマスターノードが離れたことを通知しました	マスターエージェントが再起動されるまで、クラスターのフェイルオーバーは無効になります。
クラスター*cluster_name*の*node_address*でスタンバイエージェントが終了しました	データベースの状態は監視されていません。
クラスター*cluster_name*の*node_address*の昇格中にエージェントが終了しました	データベースの状態は監視されていません。
クラスター*cluster_name*の*node_address*でエージェントが終了しました	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。
クラスター*cluster_name*のエージェントが終了しました	エージェントが終了しました。通常、この通知は、起動が完了する前にエージェントが終了すると、起動中に生成されます。
非マスターノードに割り当てられた仮想 IP アドレス	仮想 IP アドレスが非マスターノードに割り当てられているようです。競合を回避するには、 <code>variable_prod_name/VIP</code> をリリースします。VIP がマスターノードに割り当てられていることを確認し、割り当てられていない場合は手動でアドレスを再割り当てする必要があります。
仮想 IP アドレスがマスターノードに割り当てられていません。	仮想 IP アドレスがマスターノードに割り当てられていないようです。EDB-Postgres <code>variable_prod_name/VIP</code> の再取得を試みます。
クラスター*cluster_name*のクラスターにスタンバイエージェントがありません	*cluster_name*のスタンバイがクラスターを離れました。
クラスター*cluster_name*のスタンバイエージェントが失敗しました	*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、コオ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。

次のページに続く

表 1 – 前のページからの続き

件名	説明
クラスター*cluster_name*のスタンバイデータベースが失敗しました	スタンバイエージェントが、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。
スタンバイエージェントがクラスター*cluster_name*のデータベースに到達できません	スタンバイエージェントがデータベース障害を通知しましたが、他のノードはスタンバイデータベースがまだ実行中であることを検出しました。
クラスター*cluster_name*が3つのノードを下回りました	完全なフェイルオーバー保護には、少なくとも3つのノードが必要です。クラスターに監視ノードまたはエージェントノードを追加してください。
クラスターのサブセット*cluster_name*がマスターから切断されました	このノードは、クラスター*cluster_name*の大部分に接続されていません。このノードはクラスターのサブセットの一部であるため、フェイルオーバーは試行されません。表示される現在のノードは次のとおりです: <i>node_address</i>
クラスター*cluster_name*でプロモーションが開始されました。	クラスター*cluster_name*でスタンバイの昇格が開始されました。
クラスター*cluster_name*の監視エラー	*node_address*で実行されている監視がクラスターを離れました。
クラスター*cluster_name*のアイドルエージェントエラー。	*node_address*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。
クラスター*cluster_name*のネットワークから分離された1つ以上のノード	このノードはネットワークから分離されているようです。クラスターに表示されるその他のメンバーは次のとおりです: <i>node_name</i>
クラスター*cluster_name*のノードはネットワークから分離されなくなりました。	このノードはネットワークから分離されていません。
スタンバイエージェントが昇格を試みましたが、マスター DB はまだ実行中です	スタンバイ EFM エージェントは自身を昇格させようとしたましたが、マスター 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、マスター EFM エージェントが終了したことを示します。フェイルオーバーは発生していません。
スタンバイエージェントが昇格を開始しましたが、マスターが再び参加しました。	スタンバイ EFM エージェントは自身の昇格を開始しましたが、マスターエージェントがクラスターに再参加したことを検出しました。フェイルオーバーは発生していません。
スタンバイエージェントが昇格を試みましたが、マスター DB を確認できませんでした	スタンバイ EFM エージェントは自身を昇格させようとしたましたが、マスター DB がまだ*node_address*で実行されているかどうかを検出できませんでした。フェイルオーバーは発生していません。

次のページに続く

表 1 – 前のページからの続き

件名	説明
スタンバイエージェントが昇格を試みましたが、VIP はまだ割り当てられているようです	スタンバイ EFM エージェントはそれ自体を昇格しようとしたましたが、仮想 IP アドレス (<i>VIP_address</i>) がまだ別のノードに割り当てられているように見えるため、失敗しました。このような状況で昇格すると、データが破損する可能性があります。フェイルオーバーは発生していません。
スタンバイエージェントが昇格を試みましたが、孤立しているようです	スタンバイ EFM エージェントはそれ自体を昇格しようとしたましたが、既知のサーバー (<i>server_address</i>) に到達できなかったため、失敗しました。これは通常、スタンバイエージェントを他のエージェントから分離したネットワークの問題を示しています。フェイルオーバーは発生していません。
フェイルオーバーは発生していません	マスターデータベースがクラスター* <i>cluster_name</i> *で使用できなくなったことをエージェントが検出しましたが、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。
クラスター* <i>cluster_name</i> *で潜在的な手動フェイルオーバーが必要です。	クラスター* <i>cluster_name</i> *の潜在的なフェイルオーバー状況が検出されました。このクラスターでは自動フェイルオーバーが無効になっているため、手動による介入が必要です。
クラスター* <i>cluster_name</i> *でフェイルオーバーが完了しました	クラスター* <i>cluster_name</i> *でフェイルオーバーが完了しました。
クラスター* <i>cluster_name</i> *のロックファイルが削除されました	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。このロックにより、複数のエージェントが同じノード上の同じクラスターを監視することが防止されます。クラスターの別のエージェントを誤って開始しないように、このファイルを復元してください。
マスターノードでクラスター* <i>cluster_name</i> *の <i>recovery.conf</i> ファイルが見つかりました	クラスター* <i>cluster_name</i> *の <i>recovery.conf</i> ファイルがマスターノード* <i>node_address</i> *の* <i>path_name</i> *で見つかりました。このノードで DB を再起動しようとする、問題が発生する可能性があります。
リカバリ設定で <i>recovery_target_timeline</i> が最新に設定されていない	リカバリ設定で、 <i>recovery_target_timeline</i> パラメーターが最新に設定されていません。スタンバイサーバーは、新しいマスターが昇格したときに発生するタイムラインの変更を追跡できません。
クラスター* <i>cluster_name</i> *の昇格は発生していません	昇格が試行されましたが、既に昇格されているノードがあります: <i>ip_address</i> 。
クラスター* <i>cluster_name</i> *でフェイルオーバー後にスタンバイが再構成されない	このノードの <i>auto.reconfigure</i> プロパティが <i>false</i> に設定されています。フェイルオーバー後、ノードは新しいマスターノードに追従するように再構成されていません。
クラスター* <i>cluster_name</i> *の再生を再開できませんでした	プロモートされているスタンバイの再生を再開できませんでした。手動による介入が必要になる場合があります。エラー: <i>*error_decription*</i> このエラーは、スタンバイの昇格中に再生を呼び出すときにサーバーでエラーが発生した場合に返されます。

次のページに続く

表 1 – 前のページからの続き

件名	説明
スタンバイ*standby_id*の再生を再開できませんでした。	スタンバイの再生を再開できませんでした。手動による介入が必要になる場合があります。エラー: <i>error_message</i> 。
データベースのタイムアウト値で起こりうる問題	<i>remote.timeout</i> の値 (<i>value</i>) が <i>local.timeout</i> の値 (<i>value</i>) より大きい。ローカルデータベースの応答に時間がかかりすぎる場合、ローカルエージェントは、他のエージェントが接続できてもデータベースに障害が発生したと見なす可能性があります。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強制的に監視を停止し、フェイルオーバー保護が失われる可能性があります。
クラスター*cluster_name*で昇格可能なスタンバイはありません	クラスター内のスタンバイノードの現在数が最小数*number*に減少しました。別のスタンバイノードが追加または昇格されない限り、フェイルオーバーは発生しません。
クラスター*cluster_name*の昇格可能なスタンバイはありません	クラスター内の現在のフェイルオーバー優先度リストが空です。クラスター*cluster_name*の唯一の昇格可能なスタンバイを削除しました。別の昇格可能なスタンバイノードが追加されるか、フェイルオーバー優先度リストに追加されて昇格可能にならない限り、フェイルオーバーはありません。
クラスター*cluster_name*の同期レプリケーションは無効になっています。	クラスター内の同期スタンバイノードの数が*カウント*を下回りました。マスターは同期複製モードから抜けています。
データベース構成を再ロードできませんでした。	データベース構成を再ロードできませんでした。手動による介入が必要です。エラー: <i>error_message</i> 。
クラスター*cluster_name*のカスタムモニタータイムアウト	次のカスタム監視スクリプトがタイムアウトしました: <i>script_name</i>
クラスター*cluster_name*のカスタムモニター“セーフモード”の失敗	次のカスタム監視スクリプトは失敗しましたが、“セーフモード”で実行されています: <i>script_name</i> 。出力: <i>script_results</i>

以下の表にリストされている条件により、***SEVERE***通知がトリガーされます。

件名	説明
lvariable_prod_name 必要に応じて、データベースを昇格できません。	
スタンバイデータベースが再起動されましたが、EFM は接続できません	データベースの <code>start</code> または <code>restart</code> コマンドは正常に実行されましたが、データベースは接続を受け入れていません。EFM は最大 <code>*restart.connection.timeout*</code> 秒接続を試み続けます。
node_address の DB に接続できません	最大接続数の制限に達しました。
node_address の DB に接続できません	<code>db.user=</code> *user_name* のパスワードが無効です。
node_address の DB に接続できません	無効な許可指定。
マスターはクラスター *cluster_name* のローカルデータベースに ping できません	マスターエージェントは *node_address* 。 * で実行されているローカルデータベースに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、マスターは VIP を解放したり、 <code>recovery.conf</code> ファイルを作成したりしません。マスターエージェントは、 <code>resume</code> コマンドを実行してデータベースの監視を再開するまでアイドル状態になります。
フェンシングスクリプトエラー	フェンシングスクリプト *script_name* を正常に実行できませんでした。 終了値: <code>exit_code</code> 結果: *script_results* フェイルオーバーは発生していません。
プロモーション後のスクリプトが失敗しました	プロモーション後のスクリプト *script_name* を正常に実行できませんでした。終了値: <code>exit_code</code> 結果: *script_results
リモートポストプロモーションスクリプトが失敗しました	リモートポストプロモーションスクリプト *script_name* を正常に実行できませんでした 終了値: <code>exit_code</code> 結果: <code>script_results</code> ノード: <code>node_address</code>
リモート事前プロモーションスクリプトが失敗しました	リモート事前プロモーションスクリプト *script_name* を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> ノード: <code>node_address</code>
データベース障害後のスクリプトエラー	データベース後障害スクリプト *script_name* を正常に実行できませんでした。終了値: <code>exit_code</code> 結果: *script_results

次のページに続く

表 2 – 前のページからの続き

件名	説明
エージェントがスクリプトエラーを再開しました	エージェントが再開したスクリプト*script_name*は正常に実行されませんでした。結果:script_results
マスター分離スクリプトが失敗しました	マスター分離スクリプト*script_name*を正常に実行できませんでした。終了値:exit_code*結果:*script_results
スタンバイを昇格できませんでした	ノードでのプロモートコマンドが失敗しました。スタンバイを昇格できませんでした。エラーの詳細:error_details
クラスター*cluster_name*の*node_address*に recovery.conf ファイルを作成中にエラーが発生しました	昇格中にマスターノード*node_address*で recovery.conf ファイルの作成中にエラーが発生しました。昇格は続行されましたが、古いマスターノードを再起動できないようにするには、手動による介入が必要です。エラーの詳細:message_details
クラスター*cluster_name*で予期しないエラーが発生しました	このノードで予期しないエラーが発生しました。詳細については、エージェントログを確認してください。エラー:error_details
クラスター*cluster_name*で隔離されているマスターデータベース	master データベースは、クラスターの大部分から分離されています。クラスターは、*ip_address*にあるマスターエージェントに、マスターデータベースを隔離して、フェイルオーバーマネージャークラスターの残りの部分がスタンバイを昇格したときに 2 つのマスターを防ぐように指示しています。
分離されたマスターデータベースのシャットダウン。	分離されたマスターデータベースは、フェイルオーバーマネージャーによってシャットダウンされました。
クラスター*cluster_name*で隔離されているマスターデータベース	master データベースは、クラスターの大部分から分離されています。マスターが分離の検出を完了する前に、スタンバイが昇格され、クラスター内のこのノードに再参加しました。このノードは、複数のマスターデータベースを回避するためにそれ自体を分離しています。
ノード*node_address*に VIP を割り当てられませんでした	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できませんでした。
クラスター*cluster_name*の*master_or_standby*データベース障害	指定されたノードでデータベースに障害が発生しました。
エージェントはクラスター*cluster_name*のタイムアウトです	このエージェントは、ローカルデータベースにアクセスしようとしてタイムアウトしました。タイムアウト後、エージェントはデータベースに正常に ping でき、監視を再開しました。ただし、ノードをチェックして、データベースまたはエージェントの障害の可能性を防ぐために、ノードが正常に動作していることを確認する必要があります。
クラスター*cluster_name*の再開がタイムアウトしました	このエージェントは、ローカルデータベースを再構成して再起動した後、監視を再開できませんでした。詳細については、エージェントログを参照してください。

次のページに続く

表 2 – 前のページからの続き

件名	説明
クラスター*cluster_name*の内部状態の不一致	フェールオーバーマネージャークラスターの内部状態が、クラスターメンバーの実際の状態と一致していませんでした。これはまれであり、クラスターに参加するノードのタイミングの問題や状態の変更によって発生する可能性があります。問題は解決するはずですが、クラスターのステータスも確認して確認する必要があります。不一致の詳細は、エージェントログファイルで確認できます。
フェイルオーバーは発生していません	エージェントがマスターデータベースを検出しました クラスター*cluster_name*では使用できなくなりましたが、フェイルオーバーに使用できるスタンバイノードが不足しています。
*node_address*のデータベースの状態が正しくありません	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なくなったことを検出しました。エージェントはアイドル状態になります。手動による介入が必要です。
*node_address*のデータベースの状態が正しくありません	マスターエージェントは、ローカルデータベースが回復中であることを検出しました。エージェントはアイドル状態になります。手動による介入が必要です。
クラスター*cluster_name*のデータベース接続障害	このノードは次で実行されているデータベースに接続できません: <i>node_address</i> これが修正されるまで、このノードはデータベースが実行されているかどうかを確認できないため、フェイルオーバーが正しく機能しない可能性があります。
クラスター*cluster_name*のスタンバイカスタムモニターの障害	次のカスタム監視スクリプトは、スタンバイノードで失敗しました。 エージェントはローカルデータベースの監視を停止します。 スクリプトの場所: <i>script_name</i> *スクリプトの出力:* <i>script_results</i>
クラスター*cluster_name*のマスターカスタムモニターの障害	次のカスタム監視スクリプトがマスターノードで失敗しました。 EFM はスタンバイの昇格を試みます。 スクリプトの場所: <i>script_name</i> スクリプト出力: <i>script_results</i>
*master.shutdown.as.failure*がマスターノードに対して true に設定されています	このクラスターでは、*master.shutdown.as.failure*プロパティが true に設定されています。クラスター全体を停止せずにマスターエージェントを停止すると、クラスターの残りの部分ではマスターエージェントの即時障害として扱われます。マスターデータベースでメンテナンスが必要な場合は、マスター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。

次のページに続く

表 2 – 前のページからの続き

件名	説明
*ping.server.ip*に設定されたループバックアドレス	ループバックアドレスは*ping.server.ip*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。
ロードバランサーの接続スクリプトエラー	ロードバランサー接続スクリプト*script_name*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>
ロードバランサのデタッチスクリプトエラー	ロードバランサーの切り離しスクリプト*script_name*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>
クラスター*cluster_name*で利用できる同期スタンバイが不十分です。	クラスター内の同期スタンバイノードの数が*count*に減少しました。マスター上のすべての書き込みクエリは、十分な同期スタンバイノードが追加されるまでブロックされます。

第 11 章

サポートされているフェイルオーバーと失敗のシナリオ

`variable_prod_name1` フェイルオーバーが発生する場合と発生しない場合があるクラスタの障害を監視します。

`variable_prod_name1` は、非常に限定的で限定的な一連のフェイルオーバーシナリオをサポートしています。フェイルオーバーが発生する可能性があります。

- Master データベースがクラッシュまたはシャットダウンした場合。
- Master データベースをホストしているノードがクラッシュまたは到達不能になった場合。

`variable_prod_name1` これらの条件の正確性を検証するためにあらゆる試みを行います。マスターデータベースまたはノードに障害が発生したことをエージェントが確認できない場合、`variable_prod_name1` クラスタでフェイルオーバーアクションを実行しません。

`variable_prod_name1` `variable_prod_name1` が必要な場合に、`no**auto-*failover*` モードもサポートしますフェイルオーバー状態を監視および検出しますが、スタンバイへの自動フェイルオーバーは実行しません。このモードでは、フェイルオーバー条件が満たされたときに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、クラスタープロパティファイルを変更し、`:ref: auto.failover` を設定します。`<auto.failover>` パラメータを `false` に。

`variable_prod_name1` は、管理者の介入が必要だが、スタンバイデータベースをマスターに昇格することに値しない状況について管理者に警告します。

11.1 マスターデータベースがダウンしています

マスターデータベースノードで実行されているエージェントがマスターデータベースの障害を検出した場合、`variable_prod_name`障害を確認するプロセスを開始します。

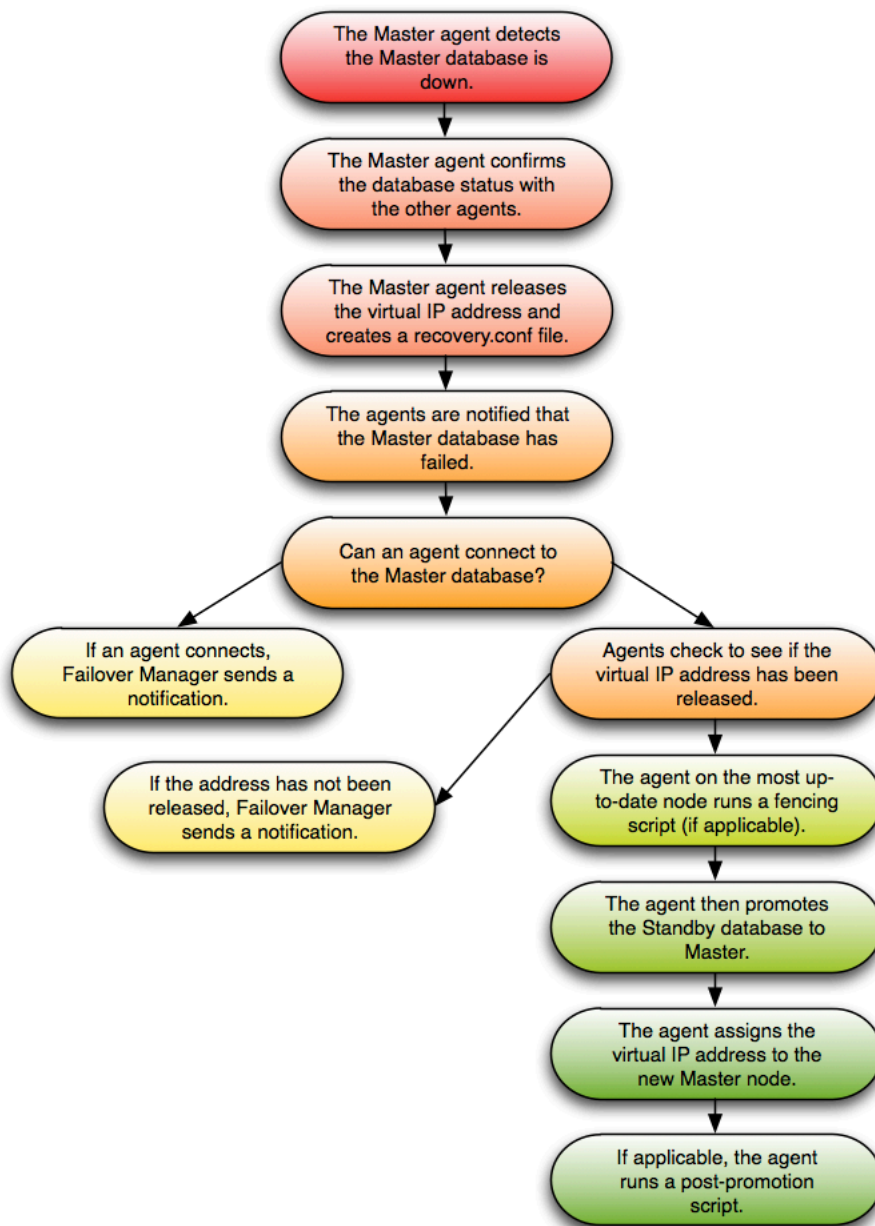


図 1 マスターデータベースの障害を確認しています。

マスターノードのエージェントがマスターデータベースの障害を検出すると、すべてのエージェントがマスターデータベースへの直接接続を試みます。エージェントがデータベースに接続できる場合、`variable_prod_name`マスターノードの状態に関する通知を送信します。エージェントが接続できない場合、マスターエージェントはデー

データベース障害を宣言し、VIP を解放します（該当する場合）。

エージェントが仮想 IP アドレスまたはデータベースサーバーに到達できない場合、`lvariable_prod_name1` フェイルオーバープロセスを開始します。最新のノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをマスターデータベースに昇格し、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 `false` に設定されていない限り、新しいマスターから複製するように構成されます。該当する場合、エージェントはプロモーション後のスクリプトを実行します。

ノードをクラスターに戻す

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

1. 元のマスターノードのデータベースをスタンバイデータベースとして再起動します。
2. 元のマスターノードで `efm resume` コマンドを呼び出します。

ノードをマスターの役割に戻す

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

1. クラスターに複数のスタンバイノードがある場合は、`efm set-priority` コマンドを使用してノードのフェイルオーバー優先度を 1 に設定します。
2. `efmserve-switchover` を呼び出します<efm_promote>ノードをマスターノードの元の役割に昇格させるコマンド。

11.2 スタンバイデータベースがダウンしています

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

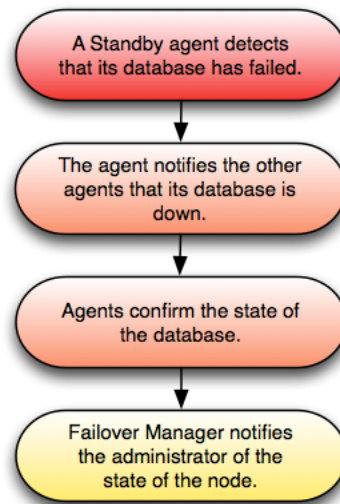


図 2 スタンバイデータベースの障害を確認しています。

スタンバイデータベースを正常な状態に戻した後、`efm resume` コマンドを呼び出してスタンバイをクラスターに戻します。

11.3 マスターエージェントの終了またはノードの失敗

`variable_prod_name1` マスターエージェントがクラッシュするか、ノードに障害が発生すると、スタンバイエージェントが障害を検出し、フェイルオーバーを開始します（該当する場合）。

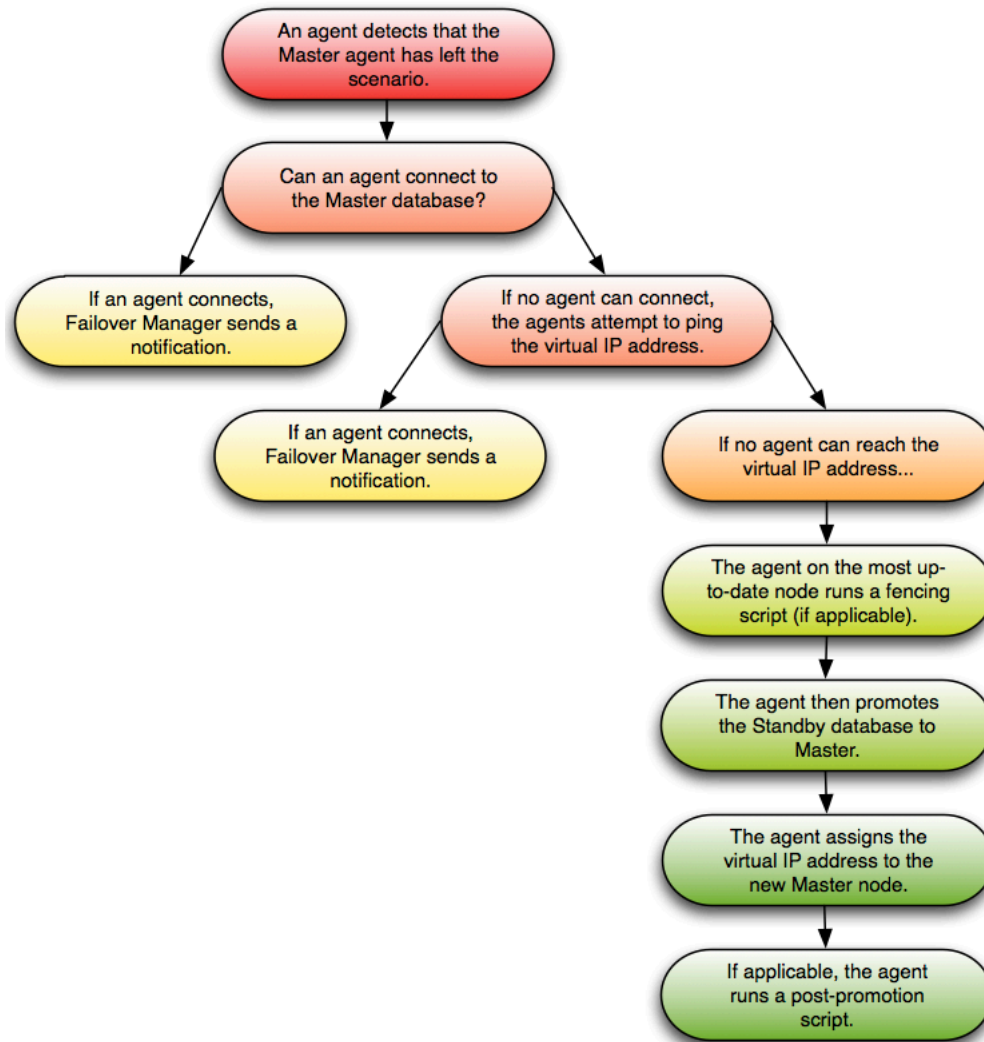


図3 マスターエージェントの障害を確認しています。

エージェントがマスターエージェントが離れたことを検出すると、すべてのエージェントがマスターデータベースへの直接接続を試みます。エージェントがデータベースに接続できる場合、エージェントはマスターエージェントの障害に関する通知を送信します。接続できるエージェントがない場合、エージェントは仮想 IP アドレスに ping を送信して、解放されているかどうかを確認します。

エージェントが仮想 IP アドレスまたはデータベースサーバーに到達できない場合、`variable_prod_name1` フェイルオーバープロセスを開始します。最新のノードのスタンバイエージェントは、フェンシングスクリプト（該当する場合）を実行し、スタンバイデータベースをマスターデータベースに昇格し、仮想 IP アドレスをスタンバイノードに割り当てます。該当する場合、エージェントはプロモーション後の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、新しいマスターから複製するように構成されます。

マスターがネットワークから分離されているためにこのシナリオが発生した場合、マスターエージェントは分離を検出して仮想 IP アドレスを解放し、`recovery.conf` ファイルを作成します。`!variable_prod_name` クラスターの残りのノードで前述の手順を実行します。

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

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

エージェントを停止しても、エージェントが失敗したことをクラスタに通知しないことに注意してください。

11.4 スタンバイエージェントの終了またはノードの失敗

スタンバイエージェントが存在するか、スタンバイノードに障害が発生した場合、他のエージェントは、それがクラスタに接続されていないことを検出します。

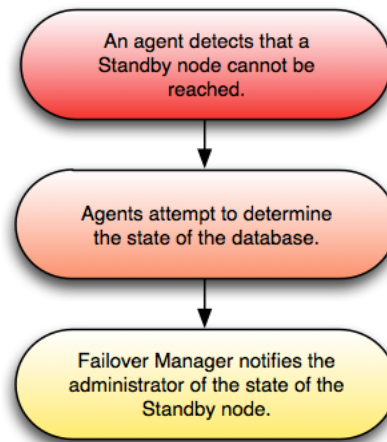


図 4 スタンバイエージェントの障害。

障害が検出されると、エージェントはノードにあるデータベースへの接続を試みます。エージェントが問題があることを確認した場合、`variable_prod_name`適切な通知を管理者に送信します。

マスターが 1 つとスタンバイが 1 つしか残っていない場合、マスターノードに障害が発生した場合のフェイルオーバー保護はありません。マスターデータベースに障害が発生した場合、マスターエージェントとスタンバイエージェントは、データベース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。

11.5 専用監視エージェントの終了/ノードが失敗する

次のシナリオでは、専用の監視（データベースをホストしていないノード）が失敗した場合に実行されるアクションについて詳しく説明します。

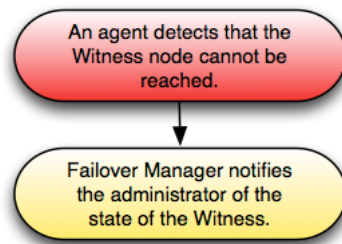


図 5 専用の証人の失敗を確認する。

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。

注:ミラーリング監視が失敗し、クラスターにノードが2つしかない場合、スタンバイノードはマスターに障害が発生したのか切断されたのかを知る方法がないため、フェイルオーバー保護はありません。2 ノードクラスターでは、マスターデータベースに障害が発生してもノードが接続されている場合、スタンバイがマスターデータベースの状態を確認できるため、フェイルオーバーが発生します。

11.6 ノードがクラスターから分離される

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

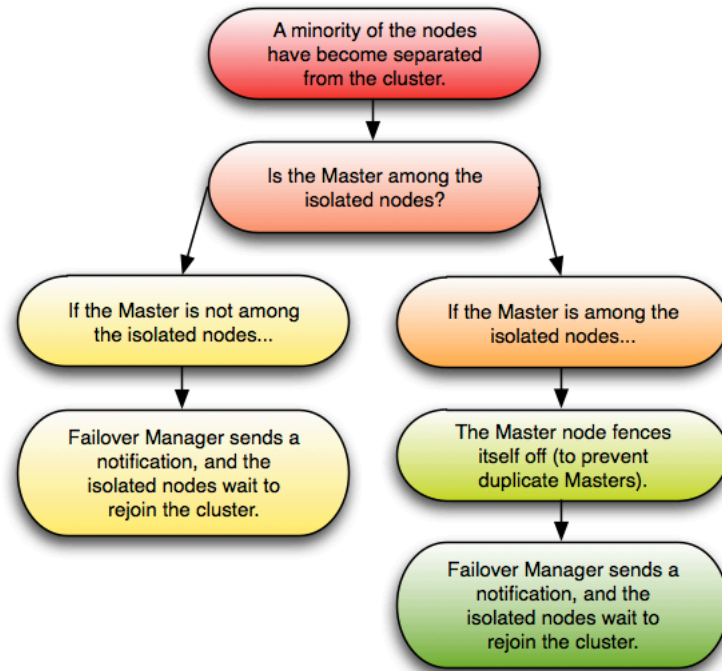


図 6 クラスターのメンバーが分離された場合。

1 つ以上のノード（ただし、クラスターの半分未満）が残りのクラスターから分離されると、残りのクラスターはノードに障害が発生したかのように動作します。エージェントは、マスターノードが分離されたノードの中にあるかどうかを識別しようとします。つまり、マスターは自身をクラスターから隔離し、スタンバイノード（クラスターの過半数内から）はそれを置き換えるように昇格されます。他のスタンバイノードは、`auto.reconfigure` が `false` に設定されていない限り、新しいマスターから複製するように構成されています。

`variable_prod_name`次に、管理者に通知し、分離されたノードは、可能な場合はクラスターに再参加します。ノードがクラスターに再度参加すると、フェイルオーバーの優先順位が変わる場合があります。

第 12 章

既存のクラスターのアップグレード

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

1. `lvariable_prod_name`をインストールします。インストールの各ノード。 `lvariable_prod_name`のインストールの詳細については、 [FailoverManager](#) のインストールを参照してください。
2. `lvariable_prod_name`をインストールした後、“`efm upgrade-conf`” ユーティリティを呼び出して、フェイルオーバーマネージャー用の “`.properties`” および “`.nodes`” ファイルを作成します。 `lvariable_prod_name`インストーラーはアップグレードユーティリティをインストールします (`efmupgrade-conf`) を `/usr/edb/efm-3.10/bin` directory に変更します。ユーティリティを呼び出すには、root 権限を引き受け、次のコマンドを呼び出します。

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

`efm upgrade-conf` ユーティリティは、既存のクラスターの `.properties` および `.nodes` ファイルを見つけ、パラメーター値を `lvariable_prod_name`で使用するために新しい構成ファイルにコピーします。ユーティリティは、設定ファイルの更新されたコピーを `/etc/edb/efm-3.10` ディレクトリに保存します。

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

任意のエディターを使用して、そのノードのサービスを開始する前に、プロパティファイル (`/etc/edb/efm-3.10` ディレクトリにあります) の追加のプロパティを変更します。プロパティ設定の詳細については、 [クラスタプロパティファイルを参照してください](#)。

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

4. バージョン固有のコマンドを使用して、古い `lvariable_prod_name`を停止します。たとえば、次のコマ

ンドを使用して、バージョン 3.9 クラスターを停止できます。

```
/usr/efm-3.9/bin/efm stop-cluster efm
```

5. 新しい フェイルオーバーマネージャーサービスを開始する<controlling_efm_service>クラスターの各ノードの ‘ (‘edb-efm-3.10‘)。

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

```
# /usr/edb/efm-3.10/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-3.9
Processing efm.properties file

The following properties were added in addition to those in previous installed version:
    external.address
    update.physical.slots.period

Checking directory /etc/edb/efm-3.9
Processing efm.nodes file

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

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

注意: カスタムサービススクリプトまたはユニットファイルを使用している場合は、ファイルを手動で更新して、新しい`variable_prod_name`を反映させる必要があります。アップグレードを実行するときのサービス名。

12.1 FailoverManager のアンインストール

|variable_prod_name|にアップグレードした後**|variable_prod_version|**、yum を使用して|variable_prod_name|の以前のインストールを削除できます。たとえば、次のコマンドを使用して、**|variable_prod_name|**を削除します。

3.9 および不要な依存関係:

- RHEL または CentOS6.x または 7.x の場合:

```
yum remove edb-efm39
```

- RHEL または CentOS8.x の場合:

```
dnf remove edb-efm39
```

- Debian または Ubuntu の場合:

```
apt-get remove edb-efm39
```

- SLES の場合:

```
zypper remove edb-efm39
```

12.2 データベース更新の実行（マイナーバージョン）

このセクションでは、データベースのマイナーバージョンのクイックアップグレードを実行する方法について説明します。次の手順を使用して、あるマイナーバージョンから別のマイナーバージョンにアップグレードする（たとえば、10.1.5 からバージョン 10.2.7 に）、またはバージョンのパッチリリースを適用することができます。

`!variable_prod_name!`の各スタンバイノードのデータベースサーバーを最初に更新する必要があります。集まる。次に、スイッチオーバーを実行して、`!variable_prod_name!`内でスタンバイノードをマスターの役割に昇格させます。集まる。次に、古いマスターノードでデータベースの更新を実行します。

クラスタの各ノードで、次の手順を実行してデータベースサーバーを更新する必要があります。

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

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

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

更新が完了したら、`!ref:efmset-priority`を使用できます<efm_set_priority>‘古いマスターをスタンバイリストの先頭に追加し、スイッチオーバーしてクラスタを元の状態に戻すコマンド。

第 13 章

トラブルシューティング

認証ファイルが見つかりません。ローカルエージェントは実行中ですか？

EFM クラスター管理コマンドを呼び出し、EFM 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?
```

このコマンドの実行は許可されていません。ユーザー`<osuser>`は`efm`グループのメンバーではありません。

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

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

通知; 予期しないエラーメッセージ

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

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

128 メガバイト未満を割り当てて実行している場合は、値を増やして、`|variable_prod_name|`を再起動する必要があります。エージェント。

OpenJDK バージョンの確認

`|variable_prod_name|`OpenJDK でテストされています。OpenJDK の使用を強くお勧めします。次のコマンドを使用して、Java インストールのタイプを確認できます。

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

第 14 章

ストリーミングレプリケーションの設定

レプリケーションシナリオの構成は複雑になる可能性があります。構成オプションの詳細については、次の場所にある PostgreSQL コアのドキュメントを参照してください。

<https://www.postgresql.org/docs/12/static/warm-standby.html#streaming-replication>

レプリケーションユーザーの md5 認証を有効にするために .pgpass ファイルを使用したい場合があります。これは、環境にとって最も安全な認証方法である場合とそうでない場合があります。サポートされている認証オプションの詳細については、次の場所にある PostgreSQL コアのドキュメントを参照してください。

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

注釈: バージョン 3.10 以降、EFM はスタンバイプロモーションに pg_ctl ユーティリティを使用します。スタンバイサーバーの昇格のために trigger_file または promote_trigger_file パラメータを設定する必要はありません。

14.1 カスケード型レプリケーションの限定サポート

`variable_prod_name1` カスケードレプリケーションの完全なサポートは提供していません。カスケードレプリケーションシナリオでの単純なフェイルオーバーのサポートは限定的です。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ミングできるようになり、マスターノードへの接続数（および処理オーバーヘッド）が減少します。

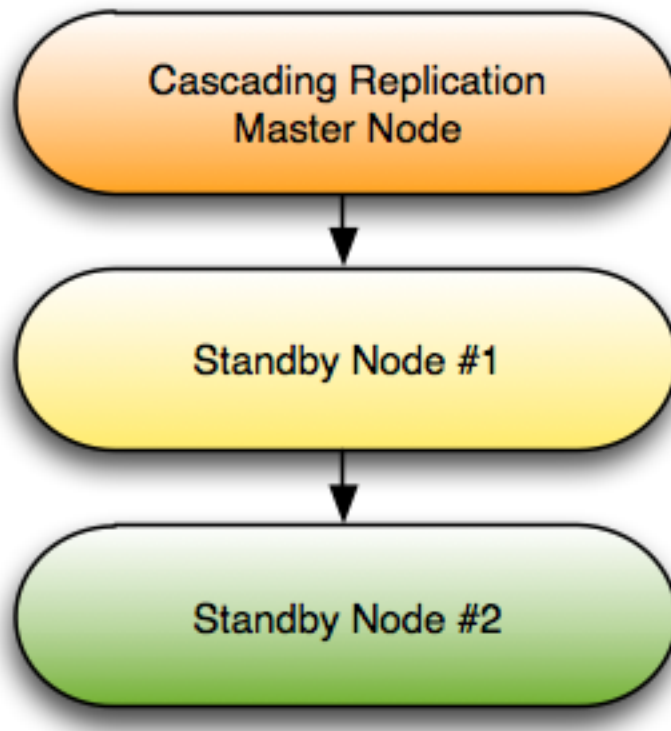


図 1 カスケードレプリケーション。

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

<https://www.postgresql.org/docs/12/static/warm-standby.html#cascading-replication>

`variable_prod_name1` を使用するにはカスケード型レプリケーションのシナリオでは、クラスタプロパティファイルを変更して、スタンバイノード # 2 で次のプロパティ値を設定する必要があります。

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

フェイルオーバーが発生すると、スタンバイノード # 1 がマスターノードの役割に昇格します。フェ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、新しいマスターノードの読み取り専用レプリカとして機能し続けます。

スタンバイノード # 1 に障害が発生した場合、フェ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電

子メールを受信します。

スイッチオーバーを実行して元のマスターに戻すと、カスケード複製シナリオが保持されない場合があることに注意してください。

第 15 章

フェイルオーバーマネージャークラスターでの SSL 認証の設定

次の手順で、`variable_prod_name1`の SSL 認証を有効にします。クラスタ内のデータベースサーバーに接続する場合、接続しているすべてのクライアントが SSL 認証を使用する必要があることに注意してください。既存のクライアントで現在使用されている接続方法を変更する必要があります。

`variable_prod_name1`で SSL を有効にするにはクラスタでは、次のことが必要です。

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

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

2. FailoverManager クラスター内の各データベースの `postgresql.conf` ファイルを変更して、SSL を有効にします。

```
ssl=on
```

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

3. `variable_prod_name1`の各ノードの `pg_hba.conf` ファイルを変更します。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ています。これにより、接続しているすべてのクライアントに SSL 認証が適用されます。`pg_hba.conf` ファイルの変更については、次の場所にある PostgreSQL コアのドキュメントを参照してください。

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

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

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

詳細については、以下をご覧ください。

<https://jdbc.postgresql.org/documentation/94/ssl-client.html>

5. 次に、証明書を Java の信頼できる証明書ファイルに追加します。

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

どこ

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

<alias_name> 任意の文字列を指定できますが、証明書ごとに一意である必要があります。

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

```
man keytool
```

各データベースサーバーからの証明書は、各エージェントの信頼できる証明書ファイルにインポートする必要があります。cacerts ファイルの場所は、システムによって異なる場合がありますことに注意してください。詳細については、以下をご覧ください。

<https://jdbc.postgresql.org/documentation/94/ssl-client.html>

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

第 16 章

結論

EDBPostgresFailoverManagerGuide

Copyright©2013-2020EnterpriseDBCorporation。

全著作権所有。

EnterpriseDB®Corporation

34CrosbyDrive、Suite201、Bedford、MA01730、USA

T+17813573390F+19784671307E

info@enterprisedb.com

www.enterprisedb.com

- EnterpriseDB および PostgresEnterpriseManager は、EnterpriseDBCorporation の登録商標です。EDB および EDBPostgres は EnterpriseDBCorporation の商標です。Oracle は Oracle、Inc. の登録商標です。その他の商標は、各所有者の商標です。
- EDB は、EDBPostgres 製品が存在する場合、ログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および検証を行います。EDB は、追加の入力コンポーネントについても同じアプローチに従いますが、製品の性質上、制限付きの検証が可能な信頼できるユーザーが入力および送信するフリーフォーム SQL、WMI、またはその他の文字列を受け入れることが必要になる場合があります。このような場合、ユーザーが誤った入力や危険な入力をするのを防ぐことはできません。
- EDB は、将来的に認証済みの信頼できるユーザーからのフリーフォーム SQL、WMI、またはその他の潜在的に危険な入力を受け入れる製品に機能を追加する権利を留保しますが、そのようなすべての機能が可能な限り最小限のリスクを提供するように設計およびテストされるようにします。可能、スーパーユーザーまたは同等の権限が必要です。
- EDB は、すべての潜在的な脅威を予測できること、または予測できることを保証するものではありません。したがって、私たちのプロセスでは、すべての潜在的な脆弱性が対処または検討されたことを完全に保証できません。

索引

Adding a Node to a Cluster, 54

admin.port, 26
application.name, 32
auto.allow.hosts, 30
auto.failover, 31
auto.reconfigure, 31
auto.resume.period, 35

bind.address, 26

Changing the priority of a Standby, 55

check.vip.before.promotion, 36
custom.monitor.interval, 39
custom.monitor.safe.mode, 39
custom.monitor.timeout, 39

db.bin, 23
db.config.dir, 24
db.data.dir, 23
db.database, 22
db.password.encrypted, 22
db.port, 22
db.reuse.connection.count, 31
db.service.name, 23
db.service.owner, 22
db.user, 22
Debian Installation, 17

efm -help, 72
efm allow-node, 69
efm cluster-status, 69
efm disallow-node, 70
efm encrypt, 70
efm node-status-json, 72
efm promote, 71
efm resume, 71
efm set-priority, 71
efm stop-cluster, 71
efm upgrade-conf, 72
efm.loglevel, 42
efm ユーティリティの使用, 69
enabling_syslog, 78

file.log.enabled, 41
from.email, 25

is.witness, 27

jdbc.sslmode, 24
jgroups.loglevel, 42
jvm.options, 42

Linux Installation, 15
local.period, 27
local.timeout, 27
local.timeout.final, 27
lock.dir, 41

log.dir, 41

Master Agent Exits, 96
Master Database is Down, 93
Master Node Fails, 96
master.shutdown.as.failure, 29
minimum.standbys, 34
Multiple agents on a single node, 58

Node Becomes Isolated, 100
node.timeout, 28
notification.level, 25

ping.server.command, 30
ping.server.ip, 30
PostgresEnterpriseManager によるストリーミングレプリ
ケーションの監視, 66
promotable, 32
Promoting a node, 56

reconfigure.num.sync, 33
reconfigure.sync.master, 34
recovery.check.period, 34
remote.timeout, 28
restart.connection.timeout, 34
restore.command, 33

script.custom.monitor, 39
script.db.failure, 38
script.fence, 37
script.load.balancer.attach, 36
script.load.balancer.detach, 36
script.master.isolated, 38
script.notification.property, 26
script.post.promotion, 37
script.remote.post.promotion, 39
script.remote.pre.promotion, 38
script.resumed, 38
SLES Installation, 18
stable.nodes.file, 31
Standby Agent Exits, 98
Standby Database is Down, 95
Standby Node Fails, 98
Starting the EFM Cluster, 53
stop.failed.master, 28
stop.isolated.master, 28
Stopping a Cluster, 58
Stopping an agent, 57
sudo.command, 40
sudo.user.command, 40
syslog.enabled, 41
syslog.facility, 41
syslog.host, 41
syslog.port, 41
syslog.protocol, 41

Ubuntu Installation, 17

`use.replay.tiebreaker`, 32
`user.email`, 25
Using the service utility, 73
Using the systemctl utility, 74

`virtual.ip`, 35
`virtual.ip.interface`, 35
`virtual.ip.prefix`, 35
`virtual.ip.single`, 35

Witness Agent Exits, 99
Witness Node Fails, 99

クラスターメンバーファイル, 45
クラスタプロパティファイル, 21

サポートされているフェイルオーバーと失敗のシナリオ, 93

ストリーミングレプリケーションの設定, 109

チュートリアル-シンプルなフェールオーバーマネージャークラスターの構成, 10

データベースのパスワードの暗号化, 42

トラブルシューティング, 107

フェイルオーバーマネージャークラスターでの SSL 認証の設定, 113
フェイルオーバーマネージャークラスターのインストール, 15
フェイルオーバーマネージャークラスターの設定, 21
フェイルオーバーマネージャークラスターの監視, 63
フェイルオーバーマネージャークラスターのサービスの制御, 73
フェイルオーバーマネージャークラスターの概要, 5
フェイルオーバーマネージャークラスターの権限の拡張, 46
フェイルオーバーマネージャークラスターの使用, 53

ロギングの制御, 77
仮想 IP アドレスでのフェイルオーバーマネージャークラスターの使用, 50

既存のクラスターのアップグレード, 103
結論, 115
新機能, 3
前提条件, 7
通知, 81