Contents

0 EDB Postgres Failover Manager . . . 2
1.0 EDB Failover Manager . . . 2
1.1 What's New . . . 2
1.2.0 Failover Manager Overview . . . 2
1.2.1 Supported Platforms . . . 3
1.2.2 Prerequisites . . . 4
1.2.3 Tutorial - Configuring a Simple Failover Manager Cluster . . . 5
1.3 Installing Failover Manager . . . 8
Installing Failover Manager on a RedHat, CentOS, or OEL Host . . . 8
Installation Locations . . . 9
Installing Failover Manager on a Debian or Ubuntu Host . . . 9
Installing Failover Manager on a SLES Host . . . 10
1.4.0 Configuring Failover Manager . . . 10
1.4.1.0 The Cluster Properties File . . . 11
Specifying Cluster Properties . . . 11
1.4.1.1 Encrypting Your Database Password . . . 24
1.4.2 Encrypting Your Database Password . . . 25
1.4.3 The Cluster Members File . . . 26
1.4.4 Extending Failover Manager Permissions . . . 27
Running Failover Manager without sudo . . . 28
1.4.5 Using Failover Manager with Virtual IP Addresses . . . 28
1.5 Using Failover Manager . . . 30
Managing a Failover Manager Cluster . . . 31
Starting the Failover Manager Cluster . . . 31
Adding Nodes to a Cluster . . . 31
Changing the Priority of a Standby . . . 32
Promoting a Failover Manager Node . . . 33
Stopping a Failover Manager Agent . . . 34
Stopping a Failover Manager Cluster . . . 34
Removing a Node from a Cluster . . . 34
Running Multiple Agents on a Single Node . . . 34
RHEL 6.x or CentOS 6.x . . . 35
RHEL 7.x or CentOS 7.x . . . 36
1.6 Monitoring a Failover Manager Cluster . . . 36
Reviewing the Cluster Status Report . . . 36
Monitoring Streaming Replication with Postgres Enterprise Manager . . . 38
1.7 Using the efm Utility . . . 41
1.8 Controlling the Failover Manager Service . . . 43
Using the service Utility on RHEL 6.x and CentOS 6.x . . . 43
Using the systemctl Utility on RHEL 7.x and CentOS 7.x . . . 44
1.9 Controlling Logging . . . 44
Enabling syslog Log File Entries . . . 45
1.10 Notifications . . . 46
1.11 Supported Failover and Failure Scenarios . . . 48
Master Database is Down . . . 48
Standby Database is Down . . . 50
Master Agent Exits or Node Fails . . . 51
Standby Agent Exits or Node Fails . . . 52
Dedicated Witness Agent Exits / Node Fails . . . 52
Nodes Become Isolated from the Cluster . . . 53
1.12 Upgrading an Existing Cluster . . . 54
Un-installing Failover Manager . . . 55
Performing a Database Update (Minor Version) . . . 55
1.13 Troubleshooting . . . 56
1.14 Configuring Streaming Replication . . . 56
Limited Support for Cascading Replication . . . 56
1.15 Configuring SSL Authentication on a Failover Manager Cluster . . . 57

0 EDB Postgres Failover Manager

1.0 EDB Failover Manager

EDB Failover Manager

EDB Postgres Failover Manager (EFM) is a high-availability module from EnterpriseDB that enables a Postgres Master node to automatically failover to a Standby node in the event of a software or hardware failure on the Master.

This guide provides information about installing, configuring and using Failover Manager 3.7.

This document uses Postgres to mean either the PostgreSQL or EDB Postgres Advanced Server database. For more information about EDB Postgres products, please visit the EnterpriseDB website .

1.1 What's New

The following changes have been made to EDB Postgres Failover Manager to create version 3.7:

- Failover Manager now supports Advanced Server and PostgreSQL version 12. Please note that the database user must have sufficient permissions to read the values of configuration variables; a database superuser can use the PostgreSQL `GRANT` command to provide the permissions needed:

  ```
  GRANT pg_read_all_settings TO user_name;
  ```

For more information about `pg_read_all_settings` , please see the PostgreSQL core documentation

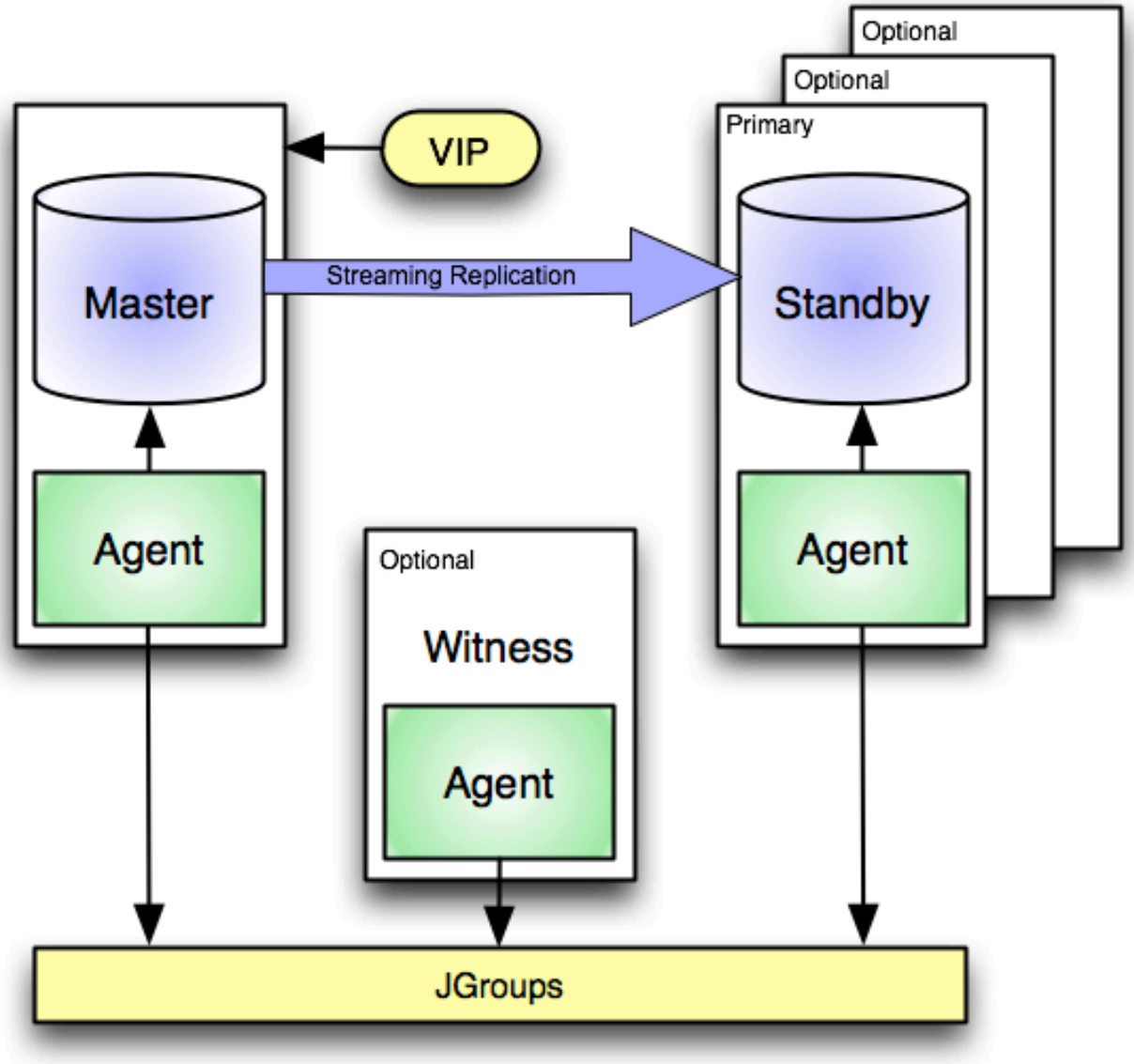
- The `application.name` property behavior is changed. For more information about `application.name` , see `The Cluster Properties File <cluster_properties>` .
- The `db.recovery.conf.dir` property has been replaced by the `db.recovery.dir` property. For more information about the `db.config.dir` property, see `The Cluster Properties File <cluster_proper` .

1.2.0 Failover Manager Overview

An EDB Postgres Failover Manager (EFM) cluster is comprised of Failover Manager processes that reside on the following hosts on a network:

- A Master node - The Master node is the primary database server that is servicing database clients.
- One or more Standby nodes - A Standby node is a streaming replication server associated with the Master node.
- A Witness node - The Witness node confirms assertions of either the Master or a Standby in a failover scenario. A cluster does not need a dedicated witness node if the cluster contains three or more nodes; if you do not have a third cluster member that is a database host, you can add a dedicated Witness node.

Traditionally, a *cluster* is a single instance of Postgres managing multiple databases. In this document, the term cluster refers to a Failover Manager cluster. A Failover Manager cluster consists of a Master agent, one or more Standby agents, and an optional Witness agent that reside on servers in a cloud or on a traditional network and communicate using the JGroups toolkit.



When a non-witness agent starts, it connects to the local database and checks the state of the database:

- If the agent cannot reach the database, it will start in idle mode.
- If it finds that the database is in recovery, the agent assumes the role of standby;
- If the database is not in recovery, the agent assumes the role of master.

In the event of a failover, Failover Manager attempts to ensure that the promoted standby is the most up-to-date standby in the cluster; please note that data loss is possible if the standby node is not in sync with the master node.

JGroups provides technology that allows Failover Manager to create clusters whose member nodes can communicate with each other and detect node failures.

The figure shown above illustrates a Failover Manager cluster that employs a virtual IP address. You can use a load balancer in place of a `virtual IP address <using_vip_addresses>` if you provide your own `fencing script <cluster_properties>` to re-configure the load balancer in the event of a failure.

1.2.1 Supported Platforms

Failover Manager 3.7 is supported on EDB Postgres Advanced Server or PostgreSQL (version 9.4 through version 12) installations running on:

- CentOS 6.x and 7.x
- Red Hat Enterprise Linux 6.x and 7.x

- Oracle Enterprise Linux 6.x and 7.x
- Red Hat Enterprise Linux (IBM Power8 Little Endian or ppc64le) 7.x
- Debian 9
- SLES 12
- Ubuntu 18.04

1.2.2 Prerequisites

Before configuring a Failover Manager cluster, you must satisfy the prerequisites described below.

Install Java 1.8 (or later)

Before using Failover Manager, you must first install Java (version 1.8 or later). Failover Manager is tested with OpenJDK, and we strongly recommend installing that version of Java. Installation instructions for Java are platform specific.

Provide an SMTP Server

You can receive notifications from Failover Manager as specified by a user-defined notification script, by email, or both.

- If you are using email notifications, an SMTP server must be running on each node of the Failover Manager scenario.
- If you provide a value in the script.notification property, you can leave the user.email field blank; an SMTP server is not required.

If an event occurs, Failover Manager invokes the script (if provided), and sends a notification email to any email addresses specified in the user.email parameter of the cluster properties file. For more information about using an SMTP server, visit:

https://access.redhat.com/site/documentation

Configure Streaming Replication

Failover Manager requires that PostgreSQL streaming replication be configured between the Master node and the Standby node or nodes. Failover Manager does not support other types of replication.

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.

On database version 12, the `primary_conninfo` , `restore_command` , and `promote_trigger_file` properties are copied to the stopped master during switchover (unless otherwise specified with the `-sourcenode` option.

Please note that Failover Manager does not support automatic reconfiguration of the standby databases after a failover if you use replication slots to manage your WAL segments. If you use replication slots, you should set the auto.reconfigure parameter to false, and manually reconfigure the standby servers in the event of a failover.

Modify the pg_hba.conf File

You must modify the `pg_hba.conf` file on the Master and Standby nodes, adding entries that allow communication between the all of the nodes in the cluster. The following example demonstrates entries that might be made to the pg_hba.conf file on the Master node:

```
# 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
```

Where:

`efm` specifies the name of a valid database user.

`fmdb` specifies the name of a database to which the efm user may connect.

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
```

Where `x` specifies the Postgres version.

Using Autostart for the Database Servers

If a Master node reboots, Failover Manager may detect the database is down on the Master node and promote a Standby node to the role of Master. If this happens, the Failover Manager 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.

To prevent this, start the Failover Manager 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.

Ensure Communication Through Firewalls

If a Linux firewall (i.e. iptables) is enabled on the host of a Failover Manager node, you may need to add rules to the firewall configuration that allow tcp communication between the Failover Manager processes in the cluster. For example:

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

The command shown above opens a small range of ports (7800 through 7810). Failover Manager will connect via the port that corresponds to the port specified in the cluster properties file.

Ensure that the Database user has Sufficient Privileges

The database user specified by the `db.user` property in the `efm.properties` file must have sufficient privileges to invoke the following functions on behalf of Failover Manager:

`pg_current_wal_lsn()`

`pg_last_wal_replay_lsn()`

`pg_wal_replay_resume()`

For detailed information about each of these functions, please see the PostgreSQL core documentation

The user must also have permissions to read the values of configuration variables; a database superuser can use the PostgreSQL `GRANT` command to provide the permissions needed:

```
GRANT pg_read_all_settings TO user_name;
```

For more information about `pg_read_all_settings` , please see the PostgreSQL core documentation

---

1.2.3 Tutorial - Configuring a Simple Failover Manager Cluster

This tutorial describes quickly configuring a Failover Manager cluster in a test environment. Other sections in this guide provide key information that you should read and understand before configuring Failover Manager for a production deployment.

This tutorial assumes that:

- A database server is running and streaming replication is set up between a master and one or two standby nodes.
- You have installed Failover Manager on each node.

The example that follows creates a cluster named `efm` .

You should start the configuration process on a master or standby node. Then, copy the configuration files to other nodes to save time.

Step 1: Create Working Configuration Files

Copy the provided sample files to create EFM configuration files, and correct the ownership:

```
cd /etc/edb/efm-3.7
```

```
cp efm.properties.in efm.properties
```

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

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

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

Step 2: Create an Encrypted Password

Create the encrypted password (needed for the properties file):

```
/usr/edb/efm-3.7/bin/efm encrypt efm
```

Follow the onscreen instructions to produce the encrypted version of your database password.

Step 3: Update the efm.properties File

The `cluster_name.properties` file contains parameters that specify connection properties and behaviors for your Failover Manager cluster. Modifications to property settings are applied when Failover Manager starts.

The following properties are the minimal properties required to configure a Failover Manager cluster. If you are configuring a production system, please see `The Cluster Properties File <cluster_properties>` for a complete list of properties.

Database connection properties (needed even on the witness so it can connect to other databases when needed):

```
db.user
```

```
db.password.encrypted
```

```
db.port
```

```
db.database
```

Owner of the data directory (usually postgres or 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
```

The data directory in which EFM will find or create the `recovery.conf` file or the `standby.signal` file:

```
db.recovery.dir
```

Set to receive email notifications (the notification text is also included in the agent log):

```
user.email
```

This is the local address of the node and the port to use for EFM. Other nodes will use this address to reach the agent, and the agent will also use this address for connecting to the local database (as opposed to connecting to localhost). An example of the format is included below:

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

Set this property to `true` on a witness node and `false` if it is a master or standby:

```
is.witness
```

If you are running on a network without access to the Internet, change this to an address that is available on your network:

```
pingServerIp=8.8.8.8
```

When configuring a production cluster, the following properties can be either true or false depending on your system configuration and usage. Set them both to true to simplify startup if you're configuring an EFM test cluster.

```
auto.allow.hosts=true
```

```
stable.nodes.file=true
```

Step 4: Update the efm.nodes File

The `cluster_name.nodes` file is read at startup to tell an agent how to find the rest of the cluster or, in the case of the first node started, can be used to simplify authorization of subsequent nodes.

Add the addresses and ports of each node in the cluster to this file. One node will act as the membership coordinator; the list should include at least the membership coordinator's address; for example:

```
1.2.3.4:7800
```

```
1.2.3.5:7800
```

```
1.2.3.6:7800
```

Please note that the Failover Manager 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 <cluster_members>`

Step 5: Configure the Other Nodes

Copy the `efm.properties` and `efm.nodes` files to the `/etc/edb/efm-3.7` 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:

- Modify the `bind.address` property to use the node's local address.
- Set `is.witness` to `true` if the node is a witness node. If the node is a witness node, the properties relating to a local database installation will be ignored.

Step 6: Start the EFM Cluster

On any node, start the Failover Manager agent. The agent is named `efm-3.7` ; you can use your platform-specific service command to control the service. For example, on a CentOS or RHEL 7.x host use the command:

```
systemctl start efm-3.7
```

On a a CentOS or RHEL 6.x host use the command:

```
service efm-3.7 start
```

After the agent starts, run the following command to see the status of the single-node cluster. You should see the addresses of the other nodes in the Allowed node host list.

```
/usr/edb/efm-3.7/bin/efm cluster-status efm
```

Start the agent on the other nodes. Run the `efm cluster-status efm` command on any node to see the cluster status.

If any agent fails to start, see the startup log for information about what went wrong:

```
cat /var/log/efm-3.7/startup-efm.log
```

Performing a Switchover

If the cluster status output shows that the master and standby(s) are in sync, you can perform a switchover with the following command:

```
/usr/edb/efm-3.7/bin/efm promote efm -switchover
```

The command will promote a standby and reconfigure the master database as a new standby in the cluster. To switch back, run the command again.

For quick access to online help, you can invoke the following command:

```
/usr/edb/efm-3.7/bin/efm promote efm --help
```

For detailed information about using the efm command line tool, see `Using the EFM Utility <using_efm_utility>` .

1.3 Installing Failover Manager

To request credentials that allow you to access an EnterpriseDB repository, visit the EDB website at:

https://info.enterprisedb.com/rs/069-ALB-3images/Repository%20Access%2004-09-2019.pdf

Installing Failover Manager on a RedHat, CentOS, or OEL Host

After receiving your credentials, you must create the EnterpriseDB repository configuration file on each node of the cluster, and then modify the file to enable access. The following steps provide detailed information about accessing the EnterpriseDB repository; the steps must be performed on each node of the cluster:

To create the repository configuration file, assume superuser privileges, and invoke the following command:

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

The repository configuration file is named `edb.repo` . The file resides in `/etc/yum.repos.d` .

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
```

After saving your changes to the configuration file, you can use the `yum install` command to install Failover Manager:

```
yum install edb-efm37
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y` , and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

Failover Manager must be installed by root. During the installation process, the installer will also create a user named `efm` that has sufficient privileges to invoke scripts that control the Failover Manager service for clusters owned by `enterprisedb` or `postgres` .

If you are using Failover Manager to monitor a cluster owned by a user other than `enterprisedb` or `postgres` , see `Extending Failover Manager Permissions <extending_efm_permissions>` .

After installing Failover Manager on each node of the cluster, you must:

1. Modify the `cluster properties file <cluster_properties>` on each node.

1. Modify the `cluster members file <cluster_members>` on each node.
2. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
3. Start the Failover Manager agent on each node of the cluster. For more information about controlling the Failover Manager service, see Section 5.

Installation Locations

Failover Manager components are installed in the following locations:

Component	Location
Executables	/usr/edb/efm-3.7/bin
Libraries	/usr/edb/efm-3.7/lib
Cluster configuration files	/etc/edb/efm-3.7
Logs	/var/log/efm-3.7
Lock files	/var/lock/efm-3.7
Log rotation file	/etc/logrotate.d/efm-3.7
sudo configuration file	/etc/sudoers.d/efm-37
Binary to access VIP without sudo	/usr/edb/efm-3.7/bin/secure

Installing Failover Manager on a Debian or Ubuntu Host

To install Failover Manager, you must also have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, visit the EnterpriseDB website at:

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

Follow the links in the EDB Failover Manager table to request credentials.

The following steps will walk you through using the EnterpriseDB apt repository to install Failover Manager. When using the commands, replace the *username* and *password* with the credentials provided by EnterpriseDB.

1. Assume superuser privileges:

   ```
   sudo su -
   ```

2. Configure the EnterpriseDB apt repository:

```
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'
```

1. Add support to your system for secure APT repositories:

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

2. Add the EDB signing key:

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

1. Update the repository meta data:

   ```
   apt-get update
   ```

2. Install Failover Manager:

   ```
   apt-get install edb-efm37
   ```

Installing Failover Manager on a SLES Host

To install Failover Manager, you must also have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, visit the EnterpriseDB website at:

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

You can use the zypper package manager to install a Failover Manager agent on an SLES 12 host. zypper will attempt to satisfy package dependencies as it installs a package, but requires access to specific repositories that are not hosted at EnterpriseDB.

You must assume superuser privileges and stop any firewalls before installing Failover Manager. Then, use the following commands to add EnterpriseDB repositories to your system:

```
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
```

The commands create the repository configuration files in the /etc/zypp/repos.d directory. Then, use the following command to refresh the metadata on your SLES host to include the EnterpriseDB repository:

```
zypper refresh
```

When prompted, provide credentials for the repository, and specify a to always trust the provided key, and update the metadata to include the EnterpriseDB repository.

You must also add SUSEConnect and the SUSE Package Hub extension to the SLES host, and register the host with SUSE, allowing access to SUSE repositories. Use the commands:

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

Then, you can use the zypper utility to install a Failover Manager agent:

```
zypper install edb-efm37
```

For detailed information about registering a SUSE host, visit:

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

1.4.0 Configuring Failover Manager

Configurable Failover Manager properties are specified in two user-modifiable files:

- `efm.properties <cluster_properties>`
- `efm.nodes <cluster_members>`

1.4.1.0 The Cluster Properties File

Each node in a Failover Manager cluster has a properties file (by default, named `efm.properties`) that contains the properties of the individual node on which it resides. The Failover Manager installer creates a file template for the properties file named `efm.properties.in` in the `/etc/edb/efm-3.7` directory.

After completing the Failover Manager installation, you must make a working copy of the template before modifying the file contents:

```
# cp /etc/edb/efm-3.7/efm.properties.in /etc/edb/efm-3.7/efm.properties
```

After copying the template file, change the owner of the file to `efm` :

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

Please note: By default, Failover Manager 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 Failover Manager to use a different name.

After creating the cluster properties file, add (or modify) configuration parameter values as required. For detailed information about each property, see Specifying Cluster Properties.

The property files are owned by `root` . The Failover Manager service script expects to find the files in the `/etc/edb/efm-3.7 directory` . If you move the property file to another location, you must create a symbolic link that specifies the new location.

Please note that all user scripts referenced in the properties file will be invoked as the Failover Manager user.

Specifying Cluster Properties

You can use the properties listed in the cluster properties file to specify connection properties and behaviors for your Failover Manager cluster. Modifications to property settings will be applied when Failover Manager starts. If you modify a property value you must restart Failover Manager to apply the changes.

Property values are case-sensitive. Note that while Postgres uses quoted strings in parameter values, Failover Manager does not allow quoted strings in property values. For example, while you might specify an IP address in a Postgres configuration parameter as:

```
listen_addresses='192.168.2.47'
```

Failover Manager requires that the value *not* be enclosed in quotes:

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

Use the properties in the `efm.properties` file to specify connection, administrative, and operational details for Failover Manager.

Use the following properties to specify connection details for the Failover Manager cluster:

```
# 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=
```

The `db.user` specified must have sufficient privileges to invoke selected PostgreSQL commands on behalf of Failover Manager. For more information, please see `Prerequisites <prerequisites>` .

For information about encrypting the password for the database user, see `Encrypting Your Database Password <enc`
.

Use the `db.service.owner` property to specify the name of the operating system user that owns the cluster that is being managed by Failover Manager. This property is not required on a dedicated witness node.

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

Specify the name of the database service in the `db.service.name` property if you use the service or systemctl command when starting or stopping the service.

```
# 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=
```

You should use the same service control mechanism (pg_ctl, service, or systemctl) each time you start or stop the database service. If you use the `pg_ctl` program to control the service, specify the location of the `pg_ctl` program in the `db.bin` property.

```
# 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=
```

Use the `db.recovery.dir` property to specify the location to which a recovery file will be written on the Master node of the cluster, and a trigger file is written on a Standby. This property required on master and standby nodes; it is not required on a dedicated witness node.

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

Use the `db.config.dir` property to specify the location of database configuration files if they are not stored in the same directory as the `recovery.conf` or `standby.signal` file. This should be the value specified by the `config_file` parameter directory of your Advanced Server or PostgreSQL installation. This value will be used as the location of the Postgres `data` directory when stopping, starting, or restarting the database.

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

```
# db.recovery.dir property will be used.
db.config.dir=
```

For more information about database configuration files, visit the PostgreSQL website

Use the `jdbc.sslmode` property to instruct Failover Manager 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
```

For information about configuring and using SSL, please see:

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

and

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

Use the `user.email` property to specify an email address (or multiple email addresses) that will receive any notifications sent by Failover Manager.

```
# 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=
```

The `from.email` property specifies the value that will be used as the sender's address on any email notifications from Failover Manager. You can:

- leave `from.email` blank to use the default value (`efm@localhost`).
- specify a custom value for the email address.
- specify a custom email address, using 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 Linux hostname utility.

For more information about notifications, see `Notifications <notifications>` .

```
# 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=
```

Use the `notification.level` property to specify the minimum severity level at which Failover Manager will send user notifications or when a notification script is called. For a complete list of notifications, please see `Notifications <notifications>` .

```
# Minimum severity level of notifications that will be sent by
# the agent. The minimum level also applies to the notification
# script (below). Valid values are INFO, WARNING, and SEVERE.
# A list of notifications is grouped by severity in the user's
# guide.
notification.level=INFO
```

Use the `script.notification` property to specify the path to a user-supplied script that acts as a notification service; the script will be passed a message subject and a message body. The script will be invoked each time Failover Manager generates a user notification.

```
# 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=
```

The `bind.address` property specifies the IP address and port number of the agent on the current node of the Failover Manager cluster.

```
# 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=
```

Use the `admin.port` property to specify a port on which Failover Manager listens for administrative commands.

```
# 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
```

Set the `is.witness` property to true to indicate that the current node is a witness node. If is.witness is true, the local agent will not check to see if a local database is running.

```
# 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.

Note

If `is.witness` is `true` , Failover Manager will not check the recovery state.

The following properties specify properties that apply to the local server:

- The `local.period` property specifies how many seconds between attempts to contact the database server.
- The `local.timeout property` specifies how long an agent will wait for a positive response from the local database server.
- The `local.timeout.final` property specifies how long an agent will wait after the final attempt to contact the database server on the current node. If a response is not received from the database within the number of seconds specified by the local.timeout.final property, the database is assumed to have failed.

For example, given the default values of these properties, a check of the local database happens once every 10 seconds. If an attempt to contact the local database does not come back positive within 60 seconds,

Failover Manager makes a final attempt to contact the database. If a response is not received within 10 seconds, Failover Manager declares database failure and notifies the administrator listed in the user.email property. These properties are not required on a dedicated witness node.

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

If necessary, you should modify these values to suit your business model.

Use the `remote.timeout property` to specify how many seconds an agent waits for a response from a remote database server (i.e., how long a standby agent waits to verify that the master database is actually down before performing failover).

```
# 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
```

Use the `node.timeout` property to specify the number of seconds that an agent will wait for a response from a node when determining if a node has failed. The node.timeout property value specifies a timeout value for agent-to-agent communication; other timeout properties in the cluster properties file specify values for agent-to-database communication.

```
# 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 Failover Manager to shut down the database if a master agent detects that it is isolated. When true (the default), Failover Manager 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 Failover Manager to attempt to shut down a master database if it can not reach the database. If true, Failover Manager 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
```

Use the `master.shutdown.as.failure` parameter to indicate that any shutdown of the Failover Manager agent on the master node should be treated as a failure. If this parameter is set to true and the master agent stops (for any reason), the cluster will attempt to confirm if the database on the master node is running:

- If the database is reached, a notification will be sent informing you of the agent status.
- If the database is not reached, a failover will occur.

```
# 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
```

The `master.shutdown.as.failure` property is meant to catch user error, rather than failures such as the accidental shutdown of a master node. The proper shutdown of a node can appear to the rest of the cluster like a user has stopped the master Failover Manager agent (for example to perform maintenance on the master database). If you set the `master.shutdown.as.failure` property to `true` , care must be taken when performing maintenance.

To perform maintenance on the master database when master.shutdown.as.failure is true, you should stop the master agent and wait to receive a notification that the master agent has failed but the database is still running. Then it is safe to stop the master database. Alternatively, you can use the efm stop-cluster command to stop all of the agents without failure checks being performed.

Use the `pingServer` property to specify the IP address of a server that Failover Manager 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.
pingServerIp=8.8.8.8
```

Use the `pingServerCommand` property to specify the command used to test network connectivity.

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

Use the `auto.allow.hosts` property to instruct the server to use the addresses specified in the .nodes file of the first node started to update the allowed host list. Enabling this property (setting auto.allow.hosts to true) can simplify cluster start-up.

```
# Have the first node started automatically add the addresses
# from its .nodes file to the allowed host list. This will make
```

```
# it faster to start the cluster when the initial set of hosts
# is already known.
auto.allow.hosts=false
```

Use the `stable.nodes.file` property to instruct the server to not rewrite the nodes file when a node joins or leaves the cluster. This property is most useful in clusters with unchanging IP addresses.

```
# 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
```

The `db.reuse.connection.count` property allows the administrator to specify the number of times Failover Manager reuses the same database connection to check the database health. The default value is 0, indicating that Failover Manager will create a fresh connection each time. This property is not required on a dedicated witness node.

```
# 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
```

The `auto.failover` property enables automatic failover. By default, auto.failover is set to true.

```
# 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 Failover Manager 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. Failover
# manager will back up recovery.conf, change the host parameter
# of the primary_conninfo entry, and restart the database. The
# restart command is contained in either the efm_db_functions or
# efm_root_functions file; default when not running db as an os
# service is:
# "pg_ctl restart -m fast -w -t <timeout> -D <directory>"
# where the timeout is the local.timeout property value and the
# directory is specified by db.recovery.dir. To turn off
# automatic reconfiguration, set this property to false.
auto.reconfigure=true
```

Please note: primary_conninfo is a space-delimited list of keyword=value pairs.

Please note: If you are using replication slots to manage your WAL segments, automatic reconfiguration is not supported; you should set `auto.reconfigure` to `false` . In the event of a failover, you must manually reconfigure the standby servers.

Use the `promotable` property to indicate that a node should not be promoted. 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 <using_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
```

You can use the `application.name` property to provide the name of an application that will be copied to the `primary_conninfo` parameter before restarting an old master node as a standby.

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

Please note: You should set the `application.name` property on the master and any promotable standby; in the event of a failover/switchover, the master node could potentially become a standby node again.

Use the `minimum.standbys` property to specify the minimum number of standby nodes that will be retained on a cluster; if the standby count drops to the specified minimum, a replica node will not be promoted in the event of a failure of the master node.

```
# 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
```

Use the `recovery.check.period` property to specify the number of seconds that Failover Manager will wait before checks to see if a database is out of recovery.

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

Use the `restart.connection.timeout` property to specify the number of seconds that Failover Manager will attempt to connect to a newly reconfigured master or standby node while the database on that node prepares to accept connections.

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

Use the `auto.resume.period` property to specify the number of seconds (after a monitored database fails, and an agent has assumed an idle state, or when starting in IDLE mode) during which an agent will attempt to resume monitoring that database.

```
# 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
```

Failover Manager 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 `virtualIp` property; specify the corresponding prefix in the virtualIp.prefix property. If virtualIp is left blank, virtual IP support is disabled.

Use the `virtualIp.interface` property to provide the network interface used by the VIP.

The specified virtual IP address is assigned only to the master node of the cluster. If you specify virtualIp.single=true, the same VIP address will be used on the new master in the event of a failover. Specify a value of false to provide a unique IP address for each node of the cluster.

For information about using a virtual IP address, see `Using Failover Manager with Virtual IP Addresses <usin`
.

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

Please Note: If a master agent is started and the node does not currently have the VIP, the EFM agent will acquire it. Stopping a master agent does not drop the VIP from the node.

Set the `check.vip.before.promotion` property to false to indicate that Failover Manager will not check to see if a VIP is in use before assigning it to a a new master in the event of a failure. Please note that this could result in multiple nodes broadcasting on the same VIP address; unless the master node is isolated or can be shut down via another process, you should set this property to true.

```
# 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
```

Use the following properties to provide paths to scripts that reconfigure your load balancer in the event of a switchover or master failure scenario. The scripts will also be invoked in the event of a standby failure. If you are using these properties, they should be provided on every node of the cluster (master, standby, and witness) to ensure that if a database node fails, another node will call the detach script with the failed node's address.

Provide a script name after the `script.load.balancer.attach` property to identify a script that will be invoked when a node should be attached to the load balancer. Use the `script.load.balancer.detach` property to specify the name of a script that will be invoked when a node should be detached from the load

balancer. Include the `%h` placeholder to represent the IP address of the node that is being attached or removed from the cluster. Include the `%t` placeholder to instruct Failover Manager to include an m (for a master node) or an s (for a standby node) in the string.

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

`script.fence` specifies the path to an optional user-supplied script that will be invoked during the promotion of a standby node to master node.

```
# 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=
```

Use the `script.post.promotion` property to specify the path to an optional user-supplied script that will be invoked after a standby node has been promoted to master.

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

Use the `script.resumed property` to specify an optional path to a user-supplied script that will be invoked when an agent resumes monitoring of a database.

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

Use the `script.db.failure` property to specify the complete path to an optional user-supplied script that Failover Manager will invoke if an agent detects that the database that it monitors has failed.

```
# 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=
```

Use the `script.master.isolated` property to specify the complete path to an optional user-supplied script that Failover Manager will invoke if the agent monitoring the master database detects that the master is isolated from the majority of the Failover Manager cluster. This script is called immediately after the VIP is released (if a VIP is in use).

```
# 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=
```

Use the `script.remote.pre.promotion` property to specify the path and name of a script that will be invoked on any agent nodes not involved in the promotion when a node is about to promote its database to master.

Include the %p placeholder to identify the address of the new primary node.

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

Use the `script.remote.post.promotion` property to specify the path and name of a script that will be invoked on any non-master nodes after a promotion occurs.

Include the %p placeholder to identify the address of the new primary node.

```
# 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=
```

Use the `script.custom.monitor` property to provide the name and location of an optional script that will be invoked on regular intervals (specified in seconds by the `custom.monitor.interval` property).

Use `custom.monitor.timeout` to specify the maximum time that the script will be allowed to run; if script execution does not complete within the time specified, Failover Manager will send a notification.

Set `custom.monitor.safe.mode` to true to instruct Failover Manager to report non-zero exit codes from the script, but not promote a standby as a result of an exit code.

```
# Absolute path to a custom monitoring script.
#
# Use script.custom.monitor to specify the location and name of
# an optional user-supplied script that will be invoked
# periodically to perform custom monitoring tasks. A non-zero
# exit value means that a check has failed; this will be treated
# as a database failure. On a master node, script failure will
# cause a promotion. On a standby node script failure will
# generate a notification and the agent will become IDLE.
#
# The custom.monitor.\* properties are required if a custom
# monitoring script is specified:
#
# custom.monitor.interval is the time in seconds between executions of the script.
#
# custom.monitor.timeout is a timeout value in seconds for how
# long the script will be allowed to run. If script execution
# exceeds the specified time, the task will be stopped and a
# notification sent. Subsequent runs will continue.
#
# If custom.monitor.safe.mode is set to true, non-zero exit codes
# from the script will be reported but will not cause a promotion
# or be treated as a database failure. This allows testing of the
# script without affecting EFM.
#
script.custom.monitor=
custom.monitor.interval=
custom.monitor.timeout=
custom.monitor.safe.mode=
```

Use the `sudo.command` property to specify a command that will be invoked by Failover Manager when performing tasks that require extended permissions. Use this option to include command options that might be specific to your system authentication.

Use the `sudo.user.command` property to specify a command that will be invoked by Failover Manager when executing commands that will be performed by the database owner.

```
# 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
```

Use the `lock.dir` property to specify an alternate location for the Failover Manager lock file; the file prevents Failover Manager from starting multiple (potentially orphaned) agents for a single cluster on the node.

```
# 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=
```

Use the `log.dir` property to specify the location to which agent log files will be written; Failover Manager will attempt to create the directory if the directory does not exist.

```
# 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=
```

After enabling the UDP or TCP protocol on a Failover Manager host, you can enable logging to syslog. Use the `syslog.protocol` parameter to specify the protocol type (UDP or TCP) and the `syslog.port` parameter to specify the listener port of the syslog host. The `syslog.facility` value may be used as an identifier for the process that created the entry; the value must be between LOCAL0 and LOCAL7.

```
# 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
```

For more information about configuring syslog logging, see Enabling syslog Log File Entries .

Use the `jgroups.loglevel` and `efm.loglevel` parameters to specify the level of detail logged by Failover Manager. The default value is INFO. For more information about logging, see `Controlling Logging <controlling_logging>` .

```
# 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
```

Use the `jvm.options` property to pass JVM-related configuration information. The default setting specifies the amount of memory that the Failover Manager agent will be allowed to use.

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

```
# the agent.
jvm.options=-Xmx128m
```

1.4.1.1 Encrypting Your Database Password

Failover Manager requires you to encrypt your database password before including it in the cluster properties file. Use the efm utility (located in the `/usr/edb/efm-3.7/bin` directory) to encrypt the password. When encrypting a password, you can either pass the password on the command line when you invoke the utility, or use the `EFMPASS` environment variable.

To encrypt a password, use the command:

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

Where <cluster_name> specifies the name of the Failover Manager cluster.

If you include the `--from-env` option, you must export the value you wish to encrypt before invoking the encryption utility. For example:

```
export EFMPASS=password
```

If you do not include the `--from-env` option, Failover Manager will prompt you to enter the database password twice before generating an encrypted password for you to place in your cluster property file. When the utility shares the encrypted password, copy and paste the encrypted password into the cluster property files.

Please note: Many Java vendors ship their version of Java with full-strength encryption included, but not enabled due to export restrictions. If you encounter an error that refers to an illegal key size when attempting to encrypt the database password, you should download and enable a Java Cryptography Extension (JCE) that provides an unlimited policy for your platform.

The following example demonstrates using the encrypt utility to encrypt a password for the acctg cluster:

```
# efm encrypt acctg
This utility will generate an encrypted password for you to place in
 your EFM cluster property file:
/etc/edb/efm-3.7/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

Please note: the utility will notify you if a properties file does not exist.

After receiving your encrypted password, paste the password into the properties file and start the Failover Manager service. If there is a problem with the encrypted password, the Failover Manager service will not start:

```
[witness@localhost ~]# service efm-3.7 start
Starting local efm-3.7 service: [FAILED]
```

If you receive this message when starting the Failover Manager service, please see the startup log (located in /var/log/efm-3.7/startup-efm.log) for more information.

If you are using RHEL 7.x or CentOS 7.x, startup information is also available with the following command:

```
systemctl status efm-3.7
```

To prevent a cluster from inadvertently connecting to the database of another cluster, the cluster name is incorporated into the encrypted password. If you modify the cluster name, you will need to re-encrypt the database password and update the cluster properties file.

Using the EFMPASS Environment Variable

The following example demonstrates using the --from-env environment variable when encrypting a password. Before invoking the `efm encrypt` command, set the value of `EFMPASS` to the password (`1safepassword`):

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

Then, invoke `efm encrypt` , specifying the `--from-env` option:

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

The encrypted password (`7ceecd8965fa7a5c330eaa9e43696f83`) is returned as a text value; when using a script, you can check the exit code of the command to confirm that the command succeeded. A successful execution returns `0` .

1.4.2 Encrypting Your Database Password

Failover Manager requires you to encrypt your database password before including it in the cluster properties file. Use the efm utility (located in the `/usr/edb/efm-3.7/bin` directory) to encrypt the password. When encrypting a password, you can either pass the password on the command line when you invoke the utility, or use the `EFMPASS` environment variable.

To encrypt a password, use the command:

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

Where <cluster_name> specifies the name of the Failover Manager cluster.

If you include the `--from-env` option, you must export the value you wish to encrypt before invoking the encryption utility. For example:

```
export EFMPASS=password
```

If you do not include the `--from-env` option, Failover Manager will prompt you to enter the database password twice before generating an encrypted password for you to place in your cluster property file. When the utility shares the encrypted password, copy and paste the encrypted password into the cluster property files.

Please note: Many Java vendors ship their version of Java with full-strength encryption included, but not enabled due to export restrictions. If you encounter an error that refers to an illegal key size when attempting to encrypt the database password, you should download and enable a Java Cryptography Extension (JCE) that provides an unlimited policy for your platform.

The following example demonstrates using the encrypt utility to encrypt a password for the acctg cluster:

```
# efm encrypt acctg
This utility will generate an encrypted password for you to place in
 your EFM cluster property file:
/etc/edb/efm-3.7/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

Please note: the utility will notify you if a properties file does not exist.

After receiving your encrypted password, paste the password into the properties file and start the Failover Manager service. If there is a problem with the encrypted password, the Failover Manager service will not start:

```
[witness@localhost ~]# service efm-3.7 start
Starting local efm-3.7 service: [FAILED]
```

If you receive this message when starting the Failover Manager service, please see the startup log (located in /var/log/efm-3.7/startup-efm.log) for more information.

If you are using RHEL 7.x or CentOS 7.x, startup information is also available with the following command:

```
systemctl status efm-3.7
```

To prevent a cluster from inadvertently connecting to the database of another cluster, the cluster name is incorporated into the encrypted password. If you modify the cluster name, you will need to re-encrypt the database password and update the cluster properties file.

Using the EFMPASS Environment Variable

The following example demonstrates using the --from-env environment variable when encrypting a password. Before invoking the `efm encrypt` command, set the value of `EFMPASS` to the password (`1safepassword`):

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

Then, invoke `efm encrypt` , specifying the `--from-env` option:

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

The encrypted password (`7ceecd8965fa7a5c330eaa9e43696f83`) is returned as a text value; when using a script, you can check the exit code of the command to confirm that the command succeeded. A successful execution returns `0` .

1.4.3 The Cluster Members File

Each node in a Failover Manager cluster has a cluster members file (by default, named efm.nodes) that contains a list of the current Failover Manager cluster members. When an agent starts, it uses the file to locate other cluster members. The Failover Manager installer creates a file template for the cluster members file named `efm.nodes.in` in the `/etc/edb/efm-3.7` directory.

After completing the Failover Manager installation, you must make a working copy of the template:

```
# cp /etc/edb/efm-3.7/efm.nodes.in /etc/edb/efm-3.7/efm.nodes
```

After copying the template file, change the owner of the file to `efm` :

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

By default, Failover Manager 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 Failover Manager service script to instruct Failover Manager to use the new name.

The cluster members file on the first node started can be empty; this node will become the Membership Coordinator. On each subsequent node, the cluster member file must contain the address and port number of the Membership Coordinator. Each entry in the cluster members file must be listed in an address:port format, with multiple entries separated by white space.

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, Failover Manager does not modify the file.

If the Membership Coordinator leaves the cluster, another node will assume the role. You can use the efm cluster-status command to find the address of the Membership Coordinator. If a node joins or leaves a cluster while an agent is down, you must manually ensure that the file includes at least the current Membership Coordinator.

If you know the IP addresses and ports of the nodes that will be joining the cluster, you can include the addresses in the cluster members file at any time. At startup, any addresses that do not identify cluster members will be ignored unless the `auto.allow.hosts` property (in the cluster properties file) is set to `true` .

If the `stable.nodes.file` property (located in the cluster properties file) is set to `true` , the Membership Coordinator will not update the `.nodes` file when cluster members join or leave the cluster; this behavior is most useful when the IP addresses of cluster members do not change often.

1.4.4 Extending Failover Manager Permissions

During the Failover Manager 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.

- When performing management functions requiring database superuser privileges, `efm` invokes the `efm_db_functions` script.
- When performing management functions requiring operating system superuser privileges, `efm` invokes the `efm_root_functions` script.
- When assigning or releasing a virtual IP address, efm invokes the `efm_address script`.

The `efm_db_functions` or `efm_root_functions` scripts perform management functions on behalf of the `efm` user.

The sudoers file contains entries that allow the user efm to control the Failover Manager 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.

The `efm-37` file is located in `/etc/sudoers.d`, and contains the following entries:

```
# Copyright EnterpriseDB Corporation, 2014-2019. All Rights Reserved.
#
# Do not edit this file. Changes to the file may be overwritten
# during an upgrade.
#
# This file assumes you are running your efm cluster as user
# 'efm'. If not, then you will need to copy this file.
# Allow user 'efm' to sudo efm_db_functions as either 'postgres'
# or 'enterprisedb'. If you run your db service under a
# non-default account, you will need to copy this file to grant
# the proper permissions and specify the account in your efm
# cluster properties file by changing the 'db.service.owner'
# property.

efm ALL=(postgres) NOPASSWD: /usr/edb/efm-3.7 /bin/efm_db_functions
efm ALL=(enterprisedb) NOPASSWD: /usr/edb/efm-3.7
/bin/efm_db_functions

# Allow user 'efm' to sudo efm_root_functions as 'root' to
# write/delete the PID file, validate the db.service.owner
# property, etc.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-3.7 /bin/efm_root_functions
# Allow user 'efm' to sudo efm_address as root for VIP tasks.
efm ALL=(ALL) NOPASSWD: /usr/edb/efm-3.7 /bin/efm_address
# relax tty requirement for user 'efm'
Defaults:efm !requiretty
```

If you are using Failover Manager to monitor clusters that are owned by users other than `postgres` or `enterprisedb`, make a copy of the `efm-37` file, and modify the content to allow the user to access the `efm_functions` script to manage their clusters.

If an agent cannot start because of permission problems, make sure the default /etc/sudoers file contains the following line at the end of the file:

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

#includedir /etc/sudoers.d
```

Running Failover Manager without sudo

By default, Failover Manager uses sudo to securely manage access to system functionality. If you choose to configure Failover Manager to run without sudo access, please note that root access is still required to:

- install the Failover Manager RPM.
- perform Failover Manager setup tasks.

To run Failover Manager without sudo, you must select a database process owner that will have privileges to perform management functions on behalf of Failover Manager. The user could be the default database superuser (for example, enterprisedb or postgres) or another privileged user. After selecting the user:

1. Use the following command to add the user to the efm group:

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

   This should allow the user to write to `/var/run/efm-3.7` and `/var/lock/efm-3.7` .

2. If you are reusing a cluster name, remove any previously created log files; the new user will not be able to write to log files created by the default (or other) owner.

3. Copy the cluster properties template file and the nodes template file:

   ```
   su - enterprisedb
   ```

   ```
   cp /etc/edb/efm-3.7/efm.properties.in <directory/cluster_name>.properties
   ```

   ```
   cp /etc/edb/efm-3.7/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.

After modifying the configuration, the new user can control Failover Manager with the following command:

> `/usr/edb/efm-3.7/bin/runefm.sh start|stop <directory/cluster_name>.properties`

Where `<directory/cluster_name.properties>` specifies the full path and name of the cluster properties file. Please note that the user must ensure that the full path to the properties file must be provided whenever the non-default user is controlling agents or using the efm script.

To allow the new user to manage Failover Manager as a service, you must provide a custom script or unit file.

Failover Manager uses a binary named manage-vip that resides in `/usr/edb/efm-3.7/bin/secure/` to perform VIP management operations without sudo privileges. This script uses setuid to acquire with the privileges needed to manage Virtual IP addresses.

- This directory is only accessible to root and users in the `efm` group.
- The binary is only executable by root and the `efm` group.

For security reasons, we recommend against modifying the access privileges of the `/usr/edb/efm-3.7/bin/secure/` directory or the `manage-vip` script.

For more information about using Failover Manager without sudo, visit:

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

---

1.4.5 Using Failover Manager with Virtual IP Addresses

Failover Manager uses the `efm_address` script to assign or release a virtual IP address.

Please note that virtual IP addresses are not supported by many cloud providers. In those environments, another mechanism should be used (such as an Elastic IP Address on AWS), which can be changed when needed by a fencing or post-promotion script.

By default, the script resides in:

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

Where `x.x` specifies the EFM version.

Use the following command variations to assign or release an IPv4 or IPv6 IP address.

To assign a virtual IPv4 IP address:

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

To assign a virtual IPv6 IP address:

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

To release a virtual address:

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

Where:

`<interface_name>` matches the name specified in the `virtualIp.interface` property in the cluster properties file.

`<IPv4_addr>` or `<IPv6_addr>` matches the name specified in the `virtualIp` property in the cluster properties file.

`prefix` matches the value specified in the `virtualIp.prefix` property in the cluster properties file.

For more information about properties that describe a virtual IP address, see The Cluster Properties File.

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 <extending_efm_permissions>` .

Please note: If a VIP address (or any address other than the `bind.address`) is assigned to a node, the operating system can choose the source address used when contacting the database. Be sure that you modify the pg_hba.conf file on all monitored databases to allow contact from all addresses within your replication scenario.

Testing the VIP

When using a virtual IP (VIP) address with Failover Manager, it is important to test the VIP functionality manually before starting failover manager. This will catch any network-related issues before they cause a problem during an actual failover. The following steps test the actions that failover manager will take. The example uses the following property values:

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

Please note: the virtualIp.prefix specifies the number of significant bits in the virtual Ip address.

When instructed to ping the VIP from a node, use the command defined by the `pingServerCommand` property.

1. Ping the VIP from all nodes to confirm that the address is not already in use:

```
# /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
```

You should see 100% packet loss.

2. Run the `efm_address add4` command on the Master node to assign the VIP and then confirm with ip address:

```
# 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. Ping the VIP from the other nodes to verify that they can reach the VIP:

```
# /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
```

You should see no packet loss.

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
...
```

The output from this step should not show an eth0 interface

5. Repeat step 3, this time verifying that the Standby and Witness do not see the VIP in use:

```
# /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
```

You should see 100% packet loss. Repeat this step on all nodes.

6. Repeat step 2 on all Standby nodes to assign the VIP to every node. You can ping the VIP from any node to verify that it is in use.

```
# 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
...
```

After the test steps above, release the VIP from any non-Master node before attempting to start Failover Manager.

Please note: the network interface used for the VIP does not have to be the same interface used for the Failover Manager agent's `bind.address` value. The master agent will drop the VIP as needed during a failover, and Failover Manager will verify that the VIP is no longer available before promoting a standby. A failure of the bind address network will lead to master isolation and failover.

If the VIP uses a different interface, you may encounter a timing condition where the rest of the cluster checks for a reachable VIP before the master agent has dropped it. In this case, EFM will retry the VIP check for the number of seconds specified in the node.timeout property to help ensure that a failover happens as expected.

1.5 Using Failover Manager

Failover Manager offers support for monitoring and failover of clusters with one or more Standby servers. You can add or remove nodes from the cluster as your demand for resources grows or shrinks.

If a Master node reboots, Failover Manager may detect the database is down on the Master node and promote a Standby node to the role of Master. If this happens, the Failover Manager 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); the rebooted Master node will return to the cluster as a second Master node. To prevent this, start the Failover Manager 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.

Managing a Failover Manager Cluster

Once configured, a Failover Manager cluster requires no regular maintenance. The following sections provide information about performing the management tasks that may occasionally be required by a Failover Manager Cluster.

By default, `some of the efm commands <using_efm_utility>` 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:

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

Starting the Failover Manager Cluster

You can start the nodes of a Failover Manager cluster in any order.

To start the Failover Manager cluster on RHEL 6.x or CentOS 6.x, assume superuser privileges, and invoke the command:

```
service efm-3.7 start
```

To start the Failover Manager cluster on RHEL 7.x or CentOS 7.x, assume superuser privileges, and invoke the command:

```
systemctl start efm-3.7
```

If the cluster properties file for the node specifies that `is.witness` is `true` , the node will start as a Witness node.

If the node is not a dedicated Witness node, Failover Manager will connect to the local database and invoke the `pg_is_in_recovery()` function. If the server responds `false` , the agent assumes the node is a Master node, and assigns a virtual IP address to the node (if applicable). If the server responds `true` , the Failover Manager agent assumes that the node is a Standby server. If the server does not respond, the agent will start in an idle state.

After joining the cluster, the Failover Manager agent checks the supplied database credentials to ensure that it can connect to all of the databases within the cluster. If the agent cannot connect, the agent will shut down.

If a new master or standby node joins a cluster, all of the existing nodes will also confirm that they can connect to the database on the new node.

Adding Nodes to a Cluster

You can add a node to a Failover Manager cluster at any time. When you add a node to a cluster, you must modify the cluster to allow the new node, and then tell the new node how to find the cluster. The following steps detail adding a node to a cluster:

1. Unless `auto.allow.hosts` is set to `true`, use the `efm allow-node` command, to add the IP address of the new node to the Failover Manager allowed node host list. When invoking the command, specify the cluster name and the IP address of the new node:

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

   For more information about using the `efm allow-node` command or controlling a Failover Manager service, see Using the EFM Utility.

   Install a Failover Manager agent and configure the cluster properties file on the new node. For more information about modifying the properties file, see `The Cluster Properties File <cluster_properties>`.

2. Configure the cluster members file on the new node, adding an entry for the Membership Coordinator. For more information about modifying the cluster members file, see `The Cluster Members File <cluster_members>`.

3. Assume superuser privileges on the new node, and start the Failover Manager agent. To start the Failover Manager cluster on RHEL 6.x or CentOS 6.x, assume superuser privileges, and invoke the command:

   ```
   service efm-3.7 start
   ```

   To start the Failover Manager cluster on RHEL 7.x or CentOS 7.x, assume superuser privileges, and invoke the command:

   ```
   systemctl start efm-3.7
   ```

When the new node joins the cluster, Failover Manager will send a notification to the administrator email provided in the user.email property, and/or will invoke the specified notification script.

Please Note: To be a useful Standby for the current node, the node must be a standby in the PostgreSQL Streaming Replication scenario.

Changing the Priority of a Standby

If your Failover Manager cluster includes more than one Standby server, you can use the `efm set-priority` command to influence the promotion priority of a Standby node. Invoke the command on any existing member of the Failover Manager cluster, and specify a priority value after the IP address of the member.

For example, the following command instructs Failover Manager that the acctg cluster member that is monitoring `10.0.1.9` is the primary Standby `(1)`:

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

You can set the priority of a standby to `0` to make the standby non-promotable. Setting the priority of a standby to a value greater than `0` overrides a property value of `promotable=false`.

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
```

In the event of a failover, Failover Manager will first retrieve information from Postgres streaming replication to confirm which Standby node has the most recent data, and promote the node with the least chance of data loss. If two Standby nodes contain equally up-to-date data, the node with a higher user-specified priority value will be promoted to Master. To check the priority value of your Standby nodes, use the command:

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

Please note: The promotion priority may change if a node becomes isolated from the cluster, and later re-joins the cluster.

Promoting a Failover Manager Node

You can invoke `efm promote` on any node of a Failover Manager cluster to start a manual promotion of a Standby database to Master database.

Manual promotion should only be performed during a maintenance window for your database cluster. If you do not have an up-to-date Standby database available, you will be prompted before continuing. To start a manual promotion, assume the identity of efm or the OS superuser, and invoke the command:

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

Where:

`<cluster_name>` is the name of the Failover Manager cluster.

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.

Include the `-sourcenode` keyword to specify the node from which the recovery settings will be copied to the master.

Include the `-quiet` keyword to suppress notifications during switchover.

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

During switchover:

- For server versions 11 and prior, the `recovery.conf` file is copied from an existing standby to the master node. For server version 12, the `primary_conninfo`, `restore_command`, and `promote_trigger_file` parameters are copied and stored in memory.
- The master database is stopped.
- If you are using a VIP, the address is released from the master node.
- A standby is promoted to replace the master node, and acquires the VIP.
- The address of the new master node is added to the `recovery.conf` file or the `primary_conninfo` details are stored in memory.
- If the `application.name` property is set for this node, the application_name property will be added to the `recovery.conf` file or the `primary_conninfo` information will be stored in memory.
- If you are using server version 12, the recovery settings that have been stored in memory are written to the `postgres.auto.conf` file.
- The old master is started; the agent will resume monitoring it as a standby.

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.recovery.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.

The Master agent remains running, and assumes a status of `Idle`.

The Standby agent confirms that the virtual IP address is no longer in use before pinging a well-known address to ensure that the agent is not isolated from the network. The Standby agent runs the fencing script and promotes the Standby database to Master. The Standby agent then assigns the virtual IP address to the Standby node, and runs the post-promotion script (if applicable).

Please note that this command instructs the service to ignore the value specified in the `auto.failover` parameter in the cluster properties file.

To return a node to the role of master, place the node first in the promotion list:

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

Then, perform a manual promotion:

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

For more information about the efm utility, see `Using the EFM Utility <using_efm_utility>` .

Stopping a Failover Manager Agent

When you stop an agent, Failover Manager will remove the node's address from the cluster members list on all of the running nodes of the cluster, but will not remove the address from the Failover Manager Allowed node host list.

To stop the Failover Manager agent on RHEL 6.x or CentOS 6.x, assume superuser privileges, and invoke the command:

```
service efm-3.7 stop
```

To stop the Failover Manager agent on RHEL 7.x or CentOS 7.x, assume superuser privileges, and invoke the command:

```
systemctl stop efm-3.7
```

Until you invoke the efm disallow-node command (removing the node's address of the node from the Allowed node host list), you can use the `service efm-3.7 start` command to restart the node at a later date without first running the `efm allow-node` command again.

Please note that stopping an agent does not signal the cluster that the agent has failed unless the master.shutdown.as.failure property is set to `true` .

Stopping a Failover Manager Cluster

To stop a Failover Manager cluster, connect to any node of a Failover Manager cluster, assume the identity of `efm` or the OS superuser, and invoke the command:

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

The command will cause *all* Failover Manager agents to exit. Terminating the Failover Manager agents completely disables all failover functionality.

Please Note: when you invoke the `efm stop-cluster` command, all authorized node information is lost from the Allowed node host list.

Removing a Node from a Cluster

The `efm disallow-node` command removes the IP address of a node from the Failover Manager 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>
```

The `efm disallow-node` command will not stop a running agent; the service will continue to run on the node until you stop the agent. If the agent or cluster is subsequently stopped, the node will not be allowed to rejoin the cluster, and will be removed from the failover priority list (and will be ineligible for promotion).

After invoking the `efm disallow-node` command, you must use the efm allow-node command <efm_allow_node> to add the node to the cluster again .

Running Multiple Agents on a Single Node

You can monitor multiple database clusters that reside on the same host by running multiple Master or Standby agents on that Failover Manager node. You may also run multiple Witness agents on a single node. To configure

Failover Manager to monitor more than one database cluster, while ensuring that Failover Manager agents from different clusters do not interfere with each other, you must:

1. Create a cluster properties file for each member of each cluster that defines a unique set of properties and the role of the node within the cluster.
2. Create a cluster members file for each member of each cluster that lists the members of the cluster.
3. Customize the service script (on a RHEL or CentOS 6.x system) or the unit file (on a RHEL or CentOS 7.x system) for each cluster to specify the names of the cluster properties and the cluster members files.
4. Start the services for each cluster.

The examples that follow uses two database clusters (acctg and sales) running on the same node:

- Data for `acctg` resides in `/opt/pgdata1` ; its server is monitoring port `5444` .
- Data for `sales` resides in `/opt/pgdata2` ; its server is monitoring port `5445` .

To run a Failover Manager agent for both of these database clusters, use the `efm.properties.in` template to create two properties files. Each cluster properties file must have a unique name. For this example, we create `acctg.properties` and `sales.properties` to match the `acctg` and `sales` database clusters.

The following parameters must be unique in each cluster properties file:

`admin.port` `bind.address` `db.port` `db.recovery.dir` `virtualIp` (if used) `virtualIp.interface` (if used)

Within each cluster properties file, the `db.port` parameter should specify a unique value for each cluster, while the `db.user` and `db.database` parameter may have the same value or a unique value. For example, the `acctg.properties` file may specify:

```
db.user=efm_user
db.password.encrypted=7c801b32a05c0c5cb2ad4ffbda5e8f9a
db.port=5444
db.database=acctg_db
```

While the `sales.properties` file may specify:

```
db.user=efm_user
db.password.encrypted=e003fea651a8b4a80fb248a22b36f334
db.port=5445
db.database=sales_db
```

Some parameters require special attention when setting up more than one Failover Manager cluster agent on the same node. If multiple agents reside on the same node, each port must be unique. Any two ports will work, but it may be easier to keep the information clear if using ports that are not too close to each other.

When creating the cluster properties file for each cluster, the `db.recovery.dir` parameters must also specify values that are unique for each respective database cluster.

The following parameters are used when assigning the virtual IP address to a node. If your Failover Manager cluster does not use a virtual IP address, leave these parameters blank.

`virtualIp` `virtualIp.interface` `virtualIp.prefix`

This parameter value is determined by the virtual IP addresses being used and may or may not be the same for both acctg.properties and sales.properties.

After creating the `acctg.properties` and `sales.properties` files, create a service script or unit file for each cluster that points to the respective property files; this step is platform specific. If you are using RHEL 6.x or CentOS 6.x, see RHEL 6.x or CentOS 6.x; if you are using RHEL 7.x or CentOS 7.x, see RHEL 7.x or CentOS 7.x.

Please note: If you are using a custom service script or unit file, you must manually update the file to reflect the new service name when you upgrade Failover Manager.

RHEL 6.x or CentOS 6.x

If you are using RHEL 6.x or CentOS 6.x, you should copy the `efm-3.7` service script to new file with a name that is unique for each cluster. For example:

```
# cp /etc/init.d/efm-3.7 /etc/init.d/efm-acctg
```

```
# cp /etc/init.d/efm-3.7 /etc/init.d/efm-sales
```

Then edit the `CLUSTER` variable, modifying the cluster name from `efm` to `acctg` or `sales` .

After creating the service scripts, run:

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

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

Then, use the new service scripts to start the agents. For example, you can start the `acctg` agent with the command:

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

RHEL 7.x or CentOS 7.x

If you are using RHEL 7.x or CentOS 7.x, you should copy the `efm-3.7` unit file to new file with a name that is unique for each cluster. For example, if you have two clusters (named acctg and sales), the unit file names might be:

```
/etc/systemd/system/efm-acctg.service
```

```
/etc/systemd/system/efm-sales.service
```

Then, edit the `CLUSTER` variable within each unit file, changing the specified cluster name from `efm` to the new cluster name. For example, for a cluster named `acctg` , the value would specify:

```
Environment=CLUSTER=acctg
```

You must also update the value of the `PIDfile` parameter to specify the new cluster name. For example:

```
PIDFile=/var/run/efm-3.7/acctg.pid
```

After copying the service scripts, use the following commands to enable the services:

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

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

Then, use the new service scripts to start the agents. For example, you can start the `acctg` agent with the command:

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

For information about customizing a unit file, please visit:

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

1.6 Monitoring a Failover Manager Cluster

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

Reviewing the Cluster Status Report

The efm cluster-status cluster properties file command returns a report that contains information about the status of the Failover Manager cluster. To invoke the command, enter:

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

The following status report is for a cluster named edb that has four nodes running:

```
efm cluster-status efm
Cluster Status: efm
Agent Type Address Agent DB VIP
-----------------------------------------------------
Witness 172.19.12.170 UP N/A
Master 172.19.13.105 UP UP 172.19.13.107\*
Standby 172.19.13.113 UP UP 172.19.13.106
Standby 172.19.14.106 UP UP 172.19.13.108
Allowed node host list:
172.19.12.170 172.19.13.113 172.19.13.105 172.19.14.106
Membership coordinator: 172.19.12.170
Standby priority host list:
172.19.13.113 172.19.14.106
Promote Status:
DB Type Address WAL LSN Info
-----------------------------------------------------
Master 172.19.13.105 0/31000140
Standby 172.19.13.113 0/31000140
Standby 172.19.14.106 0/31000140
Standby database(s) in sync with master. It is safe to promote.

[root@FOUR efm-3.5]}:
```

The Cluster Status section provides an overview of the status of the agents that reside on each node of the cluster:

```
Cluster Status: efm
Agent Type Address Agent DB VIP
-----------------------------------------------------
Witness 172.19.12.170 UP N/A
Master 172.19.13.105 UP UP 172.19.13.107\*
Standby 172.19.13.113 UP UP 172.19.13.106
Standby 172.19.14.106 UP UP 172.19.13.108
```

The asterisk (*) after the VIP address indicates that the address is available for connections. If a VIP address is not followed by an asterisk, the address has been associated with the node (in the properties file), but the address is not currently in use.

Failover Manager agents provide the information displayed in the Cluster Status section.

The `Allowed node host list` and `Standby priority host list` provide an easy way to tell which nodes are allowed to join the cluster, and the promotion order of the nodes. The IP address of the Membership coordinator is also displayed in the report:

```
Allowed node host list:
172.19.12.170 172.19.13.113 172.19.13.105 172.19.14.106
Membership coordinator: 172.19.12.170
Standby priority host list:
172.19.13.113 172.19.14.106
```

The `Promote Status` section of the report is the result of a direct query from the node on which you are invoking the cluster-status command to each database in the cluster; the query also returns the transaction log location of each database.

```
Promote Status:
DB Type Address WAL LSN Info
-----------------------------------------------------
Master 172.19.13.105 0/31000140
Standby 172.19.13.113 0/31000140
Standby 172.19.14.106 0/31000140
Standby database(s) in sync with master. It is safe to promote.
```

If a database is down (or if the database has been restarted, but the resume command has not yet been invoked), the state of the agent that resides on that host will be Idle. If an agent is idle, the cluster status report will include a summary of the condition of the idle node:

```
Agent Type Address Agent DB VIP
-----------------------------------------------------
Idle 172.19.18.105 UP UP 172.19.13.105
```

Exit Codes

The cluster status process returns an exit code that is based on the state of the cluster:

- An exit code of `0` indicates that all agents are running, and the databases on the Master and Standby nodes are running and in sync.
- A non-zero exit code indicates that there is a problem. The following problems can trigger a non-zero exit code:

  A database is down or unknown (or has an idle agent).

  Failover Manager cannot decrypt the provided database password.

  There is a problem contacting the databases to get xlog locations.

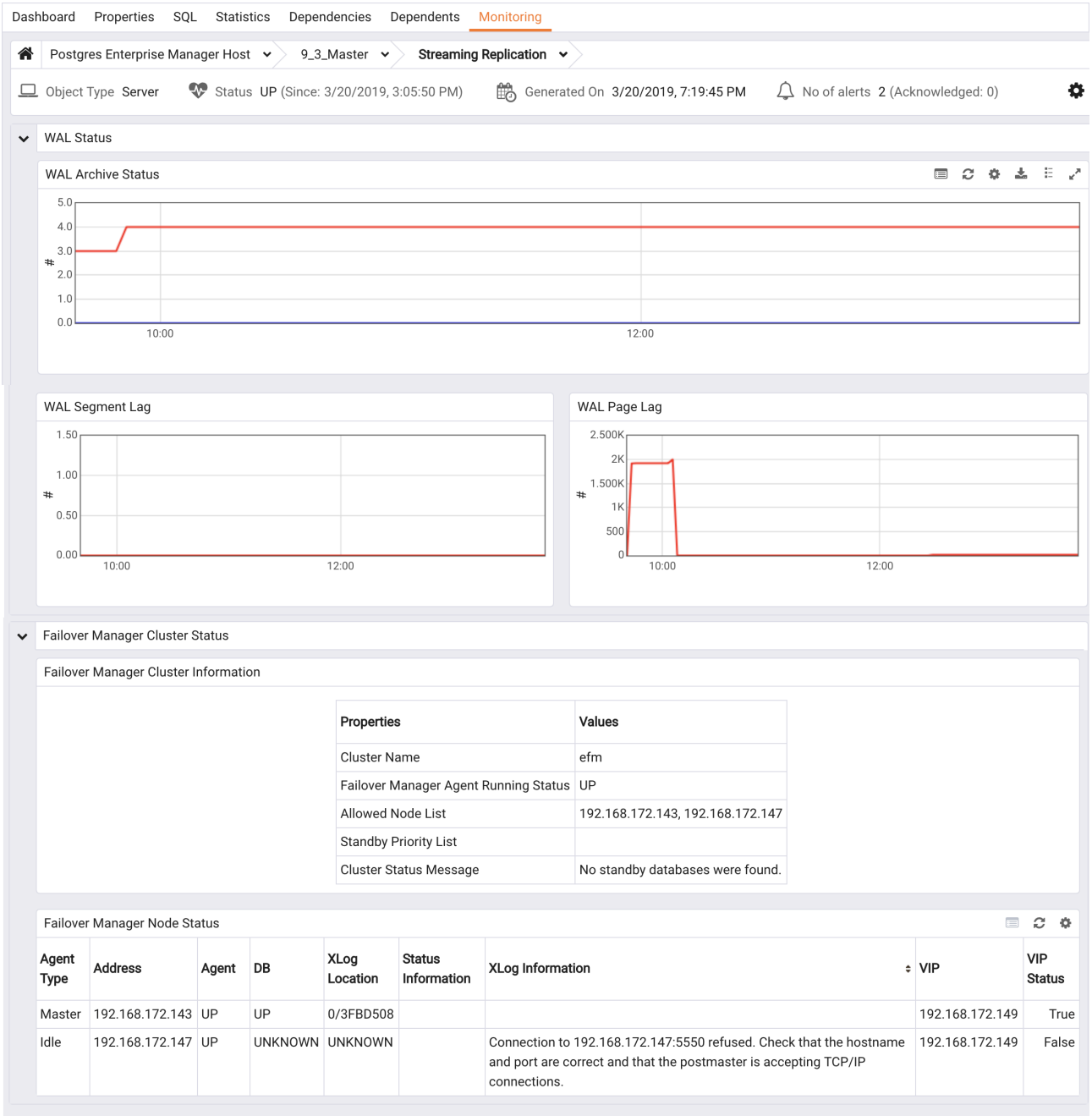
  There is no Master agent.

  There are no Standby agents.

  One or more Standby nodes are not in sync with the Master.

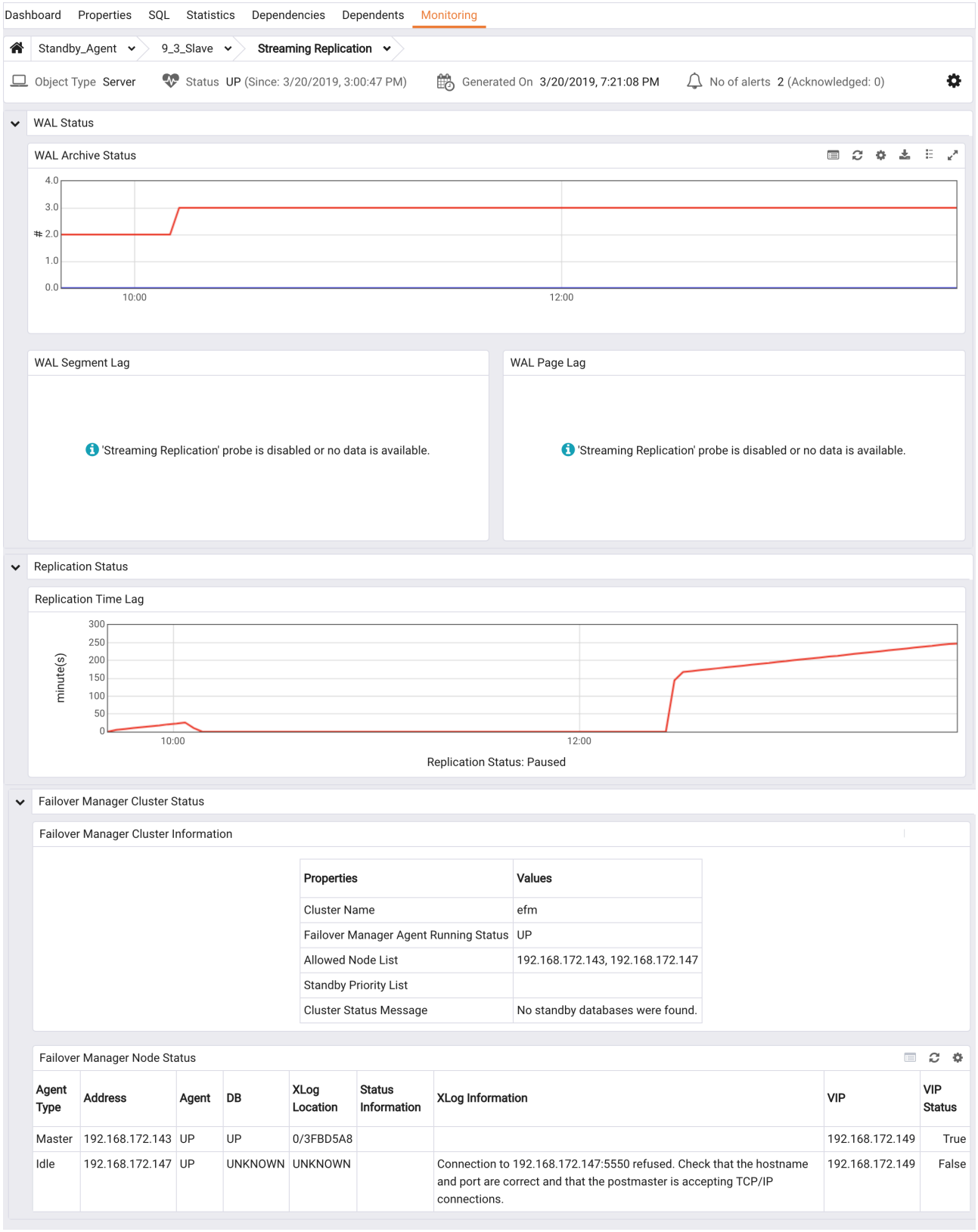
Monitoring Streaming Replication with Postgres Enterprise Manager

If you use Postgres Enterprise Manager (PEM) to monitor your servers, you can configure the Streaming Replication Analysis dashboard (part of the PEM graphical interface) to display the state of a Master or Standby node that is part of a Streaming Replication scenario.



The Streaming Replication Analysis Dashboard displays statistical information about activity for any monitored server on which streaming replication is enabled. The dashboard header identifies the status of the monitored server (either Replication Master or Replication Slave), and displays the date and time that the server was last started, the date and time that the page was last updated, and a current count of triggered alerts for the server.

When reviewing the dashboard for a Replication Slave (a Standby node), a label at the bottom of the dashboard confirms the status of the server.



By default, the PEM replication probes that provide information for the Streaming Replication Analysis dashboard are disabled.

To view the Streaming Replication Analysis dashboard for the Master node of a replication scenario, you must enable the following probes:

- Streaming Replication
- WAL Archive Status

To view the Streaming Replication Analysis dashboard for the Standby node of a replication scenario, you must enable the following probes:

- Streaming Replication Lag Time

For more information about PEM, please visit the EnterpriseDB website at:

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

---

1.7 Using the efm Utility

Failover Manager provides the efm utility to assist with cluster management. The RPM installer adds the utility to the `/usr/edb/efm-3.7/bin` directory when you install Failover Manager.

`efm allow-node <cluster_name>`

Invoke the `efm allow-node` command to allow the specified node to join the cluster. When invoking the command, provide the name of the cluster and the IP address of the joining node.

This command must be invoked by efm, a member of the efm group, or root.

`efm cluster-status <cluster_name>`

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

`efm cluster-status-json <cluster_name>`

Invoke the `efm cluster-status-json` command to display the status of a Failover Manager cluster in json format. While the format of the displayed information is different than the display generated by the efm cluster-status command, the information source is the same.

The following example is generated by querying the status of a healthy cluster with two nodes:

```
{
    "nodes": {
        "172.16.144.176": {
            "type": "Witness",
            "agent": "UP",
            "db": "N\/A",
            "vip": "",
            "vip_active": false
        },
        "172.16.144.177": {
            "type": "Master",
            "agent": "UP",
            "db": "UP",
            "vip": "",
            "vip_active": false,
            "xlog": "2\/77000220",
            "xloginfo": ""
        },
        "172.16.144.180": {
            "type": "Standby",
            "agent": "UP",
            "db": "UP",
            "vip": "",
            "vip_active": false,
            "xlog": "2\/77000220",
            "xloginfo": ""
        }
    },
    "allowednodes": [
        "172.16.144.177",
        "172.16.144.160",
```

```
        "172.16.144.180",
        "172.16.144.176"
    ],
    "membershipcoordinator": "172.16.144.177",
    "failoverpriority": [
        "172.16.144.180"
    ],
    "minimumstandbys": 0,
    "missingnodes": [],
    "messages": []
}
```

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

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]`

Invoke the `efm encrypt` command to encrypt the database password before include the password in the cluster properties file. Include the --from-env option to instruct Failover Manager to use the value specified in the EFMPASS environment variable, and execute without user input. For more information, see `Encrypting Your Database Password <encrypting_database_password>` .

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

The `efm promote` command instructs Failover Manager to perform a manual failover of standby to master.

Manual promotion should only be attempted if the status command reports that the cluster includes a Standby node that is up-to-date with the Master. If there is no up-to-date Standby, Failover Manager will prompt you before continuing.

Include the `-switchover` clause to promote a standby node, and reconfigure a master node as a standby node. Include the -sourcenode keyword, and specify a node address to indicate the node whose recovery.conf file will be copied to the old master node (making it a standby). Include the `-quiet` keyword to suppress notifications during the switchover process. Include the `-noscripts` keyword to instruct Failover Manager to not invoke fencing or post-promotion scripts.

This command must be invoked by efm, a member of the efm group, or root.

Please note that this command instructs the service to ignore the value specified in the auto.failover parameter in the cluster properties file.

`efm resume <cluster_name>`

Invoke the `efm resume` command to resume monitoring a previously stopped database. This command must be invoked by efm, a member of the efm group, or root.

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

Invoke the `efm set-priority` command to assign a failover priority to a standby node. The value specifies the order in which the new node will be used in the event of a failover. This command must be invoked by efm, a member of the efm group, or root.

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

`efm stop-cluster <cluster_name>`

Invoke the `efm stop-cluster` command to stop Failover Manager on all nodes. This command instructs Failover Manager to connect to each node on the cluster and instruct the existing members to shut down. The command has no effect on running databases, but when the command completes, there is no failover protection in place.

Please note: when you invoke the efm stop-cluster command, all authorized node information is removed from the Allowed node host list.

This command must be invoked by efm, a member of the efm group, or root.

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

Invoke the efm upgrade-conf command to copy the configuration files from an existing Failover Manager installation, and add parameters required by a Failover Manager installation. Provide the name of the previous cluster when invoking the utility. This command must be invoked with root privileges.

If you are upgrading from a Failover Manager configuration that does not use sudo, include the -source flag and specify the name of the *directory* in which the configuration files reside when invoking upgrade-conf.

`efm --help`

Invoke the `efm --help` command to display online help for the Failover Manager utility commands.

1.8 Controlling the Failover Manager Service

Each node in a Failover Manager cluster hosts a Failover Manager agent that is controlled by a service script. By default, the service script expects to find:

- A configuration file named `efm.properties` that contains the properties used by the Failover Manager service. Each node of a replication scenario must contain a properties file that provides information about the node.
- A cluster members file named `efm.nodes` that contains a list of the cluster members. Each node of a replication scenario must contain a cluster members list.

Note that if you are running multiple clusters on a single node you will need to manually create configuration files with cluster-specific names and modify the service script for the corresponding clusters.

The commands that control the Failover Manager service are platform-specific.

Using the service Utility on RHEL 6.x and CentOS 6.x

On RHEL 6.x and CentOS 6.x, Failover Manager runs as a Linux service named (by default) efm-3.7 that is located in /etc/init.d. Each database cluster monitored by Failover Manager will run a copy of the service on each node of the replication cluster.

Use the following service commands to control a Failover Manager agent that resides on a RHEL 6.x or CentOS 6.x host:

`service efm-3.7 start`

The start command starts the Failover Manager agent on the current node. The local Failover Manager agent monitors the local database and communicates with Failover Manager on the other nodes. You can start the nodes in a Failover Manager cluster in any order.

This command must be invoked by root.

`service efm-3.7 stop`

Stop the Failover Manager on the current node. This command must be invoked by root.

`service efm-3.7 status`

The status command returns the status of the Failover Manager agent on which it is invoked. You can invoke the status command on any node to instruct Failover Manager to return status information. For example:

[witness@localhost ~]# service efm-3.7 status efm-3.7 (pid 50836) is running...

`service efm-3.7 help`

Display online help for the Failover Manager service script.

Using the systemctl Utility on RHEL 7.x and CentOS 7.x

On RHEL 7.x and CentOS 7.x, Failover Manager runs as a Linux service named (by default) `efm-3.7.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.

Use the following systemctl commands to control a Failover Manager agent that resides on a RHEL 7.x or CentOS 7.x host:

`systemctl start efm-3.7`

The start command starts the Failover Manager agent on the current node. The local Failover Manager agent monitors the local database and communicates with Failover Manager on the other nodes. You can start the nodes in a Failover Manager cluster in any order.

This command must be invoked by root.

`systemctl stop efm-3.7`

Stop the Failover Manager on the current node. This command must be invoked by root.

`systemctl status efm-3.7`

The status command returns the status of the Failover Manager agent on which it is invoked. You can invoke the status command on any node to instruct Failover Manager to return status and server startup information.

[root@ONE ~]}> systemctl status efm-3.7
efm-3.7.service - EnterpriseDB Failover Manager 3.7
Loaded: loaded (/usr/lib/systemd/system/efm-3.7.service; disabled; vendor preset: disabled) Active: active (running) since Wed 2013-02-14 14:02:16 EST; 4s ago

Process: 58125 ExecStart=/bin/bash -c /usr/edb/efm-3.7/bin/runefm.sh start ${CLUSTER} (code=exited, status=0/SUCCESS) Main PID: 58180 (java) CGroup: /system.slice/efm-3.7.service └─58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/jre/bin/java -cp /usr/edb/efm-3.7/lib/EFM-3.7.0.jar -Xmx128m -agentlib:jdwp=transport...

1.9 Controlling Logging

Failover Manager writes and stores one log file per agent and one startup log per agent in `/var/log/<cluster_name>-3.7` (where `<cluster_name>` specifies the name of the cluster).

You can control the level of detail written to the agent log by modifying the `jgroups.loglevel` and `efm.loglevel` parameters in the cluster properties file:

```
# 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
```

The logging facilities use the Java logging library and logging levels. The log levels (in order from most logging output to least) are:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

For example, if you set the `efm.loglevel` parameter to `WARN`, Failover Manager will only log messages at the `WARN` level and above (`WARN` and `ERROR`).

By default, Failover Manager log files are rotated daily, compressed, and stored for a week. You can modify the file rotation schedule by changing settings in the log rotation file (`/etc/logrotate.d/efm-3.7`). For more information about modifying the log rotation schedule, consult the logrotate man page:

```
$ man logrotate
```

Enabling syslog Log File Entries

Failover Manager supports for syslog logging. To implement syslog logging, you must configure syslog to allow UDP or TCP connections.

To allow a connection to syslog, edit the `/etc/rsyslog.conf` file and uncomment the protocol you wish to use. You must also ensure that the `UDPServerRun` or `TCPServerRun` entry associated with the protocol includes the port number to which log entries will be sent.

For example, the following configuration file entries enable UDP connections to port 514:

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

The following configuration file entries enable TCP connections to port 514:

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

After modifying the syslog configuration file, restart the rsyslog service to enable the connections:

```
systemctl restart rsyslog.service
```

After modifying the `rsyslog.conf` file on the Failover Manager host, you must modify the Failover Manager properties to enable logging. Use your choice of editor to modify the properties file (`/etc/edb/efm-3.7/efm.properties`) specifying the type of logging that you wish to implement:

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

You must also specify syslog details for your system. Use the `syslog.protocol` parameter to specify the protocol type (UDP or TCP) and the `syslog.port` parameter to specify the listener port of the syslog host. The `syslog.facility` value may be used as an identifier for the process that created the entry; the value must be between LOCAL0 and LOCAL7.

```
# 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
```

For more information about syslog, please see the syslog man page:

```
syslog man
```

1.10 Notifications

Failover Manager will send e-mail notifications and/or invoke a notification script when a notable event occurs that affects the cluster. If you have configured Failover Manager to send an email notification, you must have an SMTP server running on port 25 on each node of the cluster. Use the following parameters to configure notification behavior for Failover Manager:

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

For more information about editing the configuration properties, see `Specifying Cluster Properties <cluster_pro`
.

The body of the notification contains details about the event that triggered the notification, and about the current state of the cluster. For example:

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

The VIP field displays the IP address and state of the virtual IP if implemented for the node.

Failover Manager assigns a severity level to each notification. The following levels indicate increasing levels of attention required:

`INFO` indicates an informational message about the agent and does not require any manual intervention (for example, Failover Manager has started or stopped).

`WARNING` indicates that an event has happened that requires the administrator to check on the system (for example, failover has occurred).

`SEVERE` indicates that a serious event has happened and requires the immediate attention of the administrator (for example, failover was attempted, but was unable to complete).

The severity level designates the urgency of the notification. A notification with a severity level of `SEVERE` requires user attention immediately, while a notification with a severity level of `INFO` will call your attention to operational information about your cluster that does not require user action. Notification severity levels are not related to logging levels; all notifications are sent regardless of the log level detail specified in the configuration file.

You can use the notification.level property to specify the minimum severity level that will trigger a notification.

Please note: In addition to sending notices to the administrative email address, all notifications are recorded in the cluster log file (`/var/log/efm-3.7/cluster_name.log`).

The conditions listed in the table below will trigger an `INFO` level notification:

Subject	Description
Executed fencing script	Executed fencing script *script_name* Results: *script_*
Executed post-promotion script	Executed post-promotion script *script_name* Results:
Executed remote pre-promotion script	Executed remote pre-promotion script *script_name* R
Executed remote post-promotion script	Executed remote post-promotion script *script_name*
Executed post-database failure script	Executed post-database failure script *script_name* Re
Executed master isolation script	Executed master isolation script *script_name* Results

Subject	Description
Witness agent running on *node_address* for cluster *cluster_name*	Witness agent is running.
Master agent running on *node_address* for cluster *cluster_name*	Master agent is running and database health is being
Standby agent running on *node_address* for cluster *cluster_name*	Standby agent is running and database health is beir
Idle agent running on node *node_address* for cluster *cluster_name*	Idle agent is running. After starting the local database
Assigning VIP to node *node_address*	Assigning VIP *VIP_address* to node *node_address* F
Releasing VIP from node *node_address*	Releasing VIP *VIP_address* from node *node_addres*
Starting auto resume check for cluster *cluster_name*	The agent on this node will check every *auto.resume*
Executed agent resumed script	Executed agent resumed script *script_name* Results:
WAL logs backed up during promotion	When reconfiguring this standby to follow the new pri

The conditions listed in the table below will trigger a *WARNING* level notification:

Subject	Description
Witness agent exited on *node_address* for cluster *cluster_name*	Witness agent has exited.
Master agent exited on *node_address* for cluster *cluster_name*	Database health is not being monitored.
Cluster *cluster_name* notified that master node has left	Failover is disabled for the cluster until the mas
Standby agent exited on *node_address* for cluster *cluster_name*	Database health is not being monitored.
Agent exited during promotion on *node_address* for cluster *cluster_name*	Database health is not being monitored.
Agent exited on *node_address* for cluster *cluster_name*	The agent has exited. This is generated by an
Agent exited for cluster *cluster_name*	The agent has exited. This notification is usual
Virtual IP address assigned to non-master node	The virtual IP address appears to be assigned
Virtual IP address not assigned to master node.	The virtual IP address appears to not be assig
No standby agent in cluster for cluster *cluster_name*	The standbys on *cluster_name* have left the cl
Standby agent failed for cluster *cluster_name*	A standby agent on *cluster_name* has left the
Standby database failed for cluster *cluster_name*	A standby agent has signaled that its database
Standby agent cannot reach database for cluster *cluster_name*	A standby agent has signaled database failure
Cluster *cluster_name* has dropped below three nodes	At least three nodes are required for full failove
Subset of cluster *cluster_name* disconnected from master	This node is no longer connected to the majori
Promotion has started on cluster *cluster_name*.	The promotion of a standby has started on clus
Witness failure for cluster *cluster_name*	Witness running at *node_address* has left the
Idle agent failure for cluster *cluster_name*.	Idle agent running at *node_address* has left th
One or more nodes isolated from network for cluster *cluster_name*	This node appears to be isolated from the netw
Node no longer isolated from network for cluster *cluster_name*.	This node is no longer isolated from the netwo
Standby agent tried to promote, but master DB is still running	The standby EFM agent tried to promote itself,
Standby agent started to promote, but master has rejoined.	The standby EFM agent started to promote itse
Standby agent tried to promote, but could not verify master DB	The standby EFM agent tried to promote itself,
Standby agent tried to promote, but VIP appears to still be assigned	The standby EFM agent tried to promote itself,
Standby agent tried to promote, but appears to be orphaned	The standby EFM agent tried to promote itself,
Failover has not occurred	An agent has detected that the master databas
Potential manual failover required on cluster *cluster_name*.	A potential failover situation was detected for c
Failover has completed on cluster *cluster_name*	Failover has completed on cluster *cluster_nam*
Lock file for cluster *cluster_name* has been removed	The lock file for cluster *cluster_name* has been
recovery.conf file for cluster *cluster_name* has been found	A recovery.conf file for cluster *cluster_name* ha
recovery_target_timeline is not set to latest in recovery settings	The recovery_target_timeline parameter is not
trigger_file path given in recovery.conf is not writable	The path provided for the trigger_file paramete
Promotion has not occurred for cluster *cluster_name*	A promotion was attempted but there is alread
Standby not reconfigured after failover in cluster *cluster_name*	The auto.reconfigure property has been set to
Could not resume replay for cluster *cluster_name*	Could not resume replay for standby being pro
Could not resume replay for standby *standby_id*.	Could not resume replay for standby. Manual i
Possible problem with database timeout values	Your remote.timeout value (*value*) is higher tha
No standbys available for promotion in cluster *cluster_name*	The current number of standby nodes in the cl
Custom monitor timeout for cluster *cluster_name*	The following custom monitoring script has tim
Custom monitor 'safe mode' failure for cluster *cluster_name*	The following custom monitor script has failed,

The conditions listed in the table below will trigger a *SEVERE* notification:

1.11 Supported Failover and Failure Scenarios

Failover Manager monitors a cluster for failures that may or may not result in failover.

Failover Manager supports a very specific and limited set of failover scenarios. Failover can occur:

- if the Master database crashes or is shutdown.
- if the node hosting the Master database crashes or becomes unreachable.

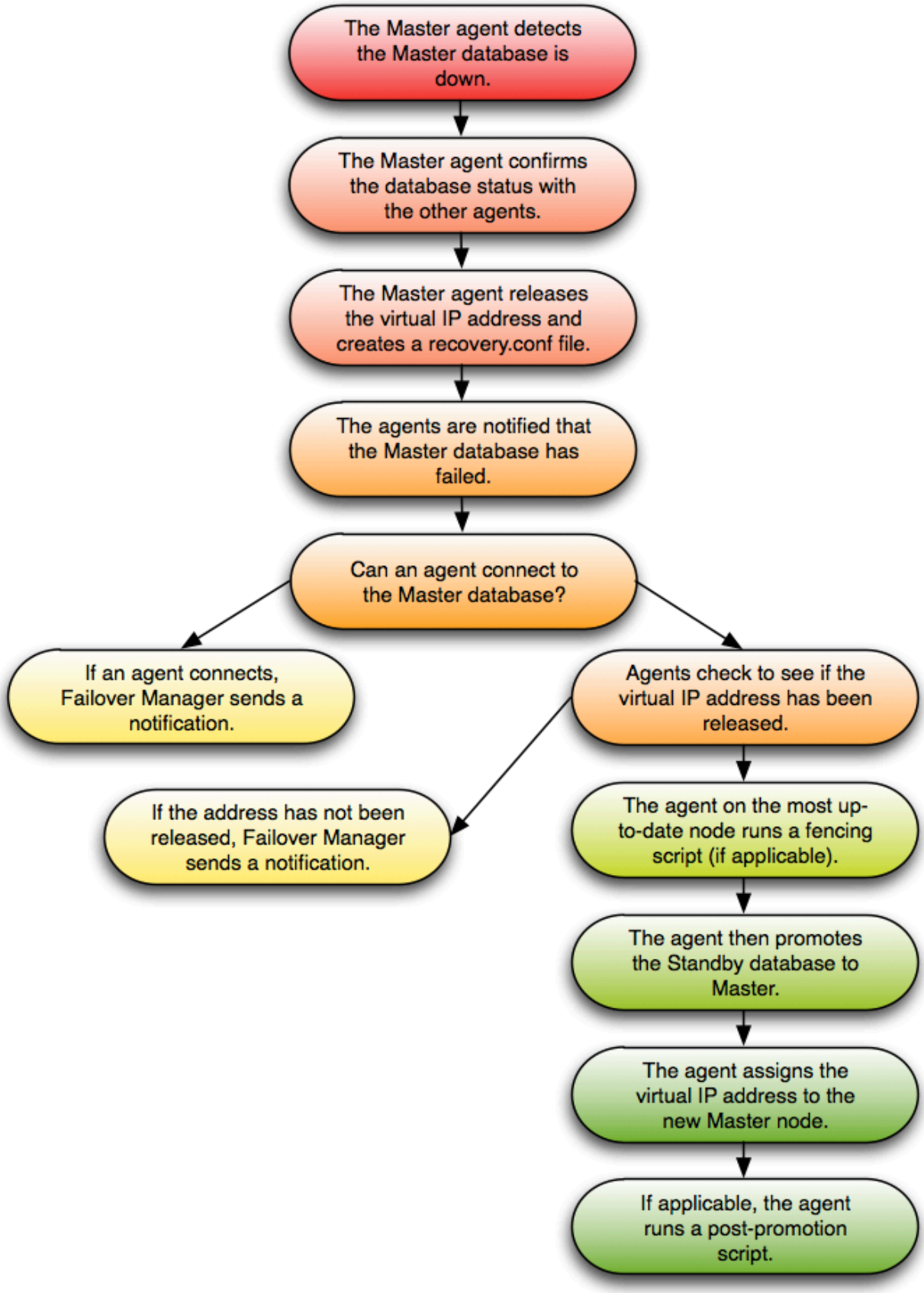
Failover Manager makes every attempt to verify the accuracy of these conditions. If agents cannot confirm that the Master database or node has failed, Failover Manager will not perform any failover actions on the cluster.

Failover Manager also supports a *no auto-failover* mode for situations where you want Failover Manager to monitor and detect failover conditions, but not perform an automatic failover to a Standby. In this mode, a notification is sent to the administrator when failover conditions are met. To disable automatic failover, modify the cluster properties file, setting the auto.failover parameter to false.

Failover Manager will alert an administrator to situations that require administrator intervention, but that do not merit promoting a Standby database to Master.

Master Database is Down

If the agent running on the Master database node detects a failure of the Master database, Failover Manager begins the process of confirming the failure.



If the agent on the Master node detects that the Master database has failed, all agents attempt to connect directly to the Master database. If an agent can connect to the database, Failover Manager sends a notification about the state of the Master node. If no agent can connect, the Master agent declares database failure and releases the VIP (if applicable).

If no agent can reach the virtual IP address or the database server, Failover Manager starts the failover process. The Standby agent on the most up-to-date node runs a fencing script (if applicable), promotes the Standby database to Master database, and assigns the virtual IP address to the Standby node. Any additional Standby nodes are configured to replicate from the new master unless auto.reconfigure is set to false. If applicable, the agent runs a post-promotion script.

Returning the Node to the Cluster

To recover from this scenario without restarting the entire cluster, you should:

1. Restart the database on the original Master node as a Standby database.
2. Invoke the efm resume command on the original Master node.

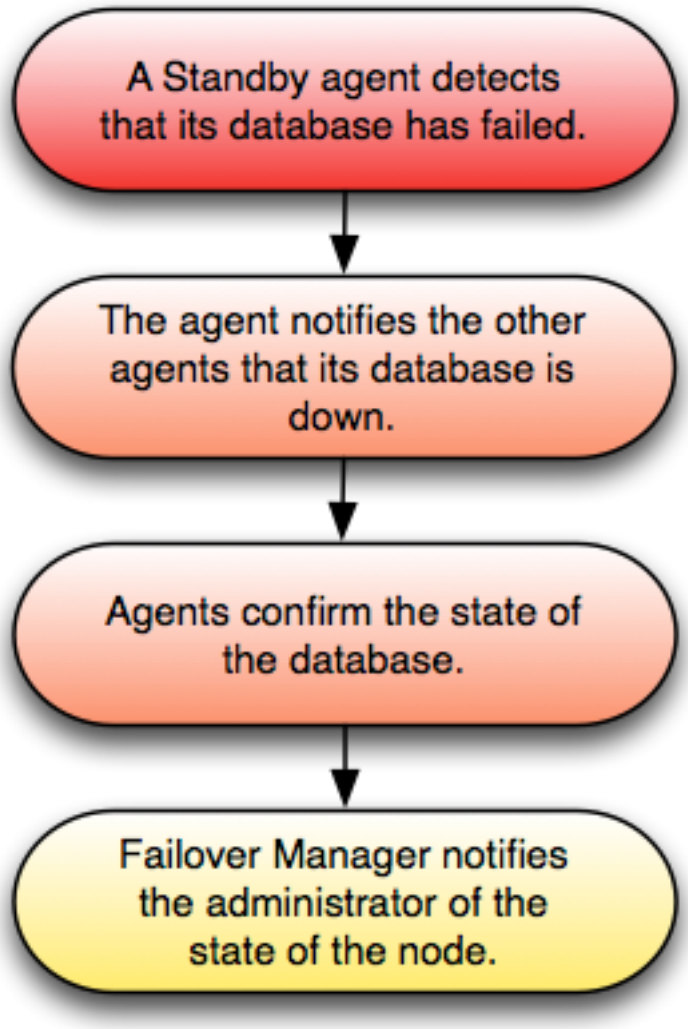
Returning the Node to the Role of Master

After returning the node to the cluster as a Standby, you can easily return the node to the role of Master:

1. If the cluster has more than one Standby node, use the efm allow-node command to set the node's failover priority to 1.
2. Invoke the efm promote -switchover command to promote the node to its original role of Master node.

Standby Database is Down

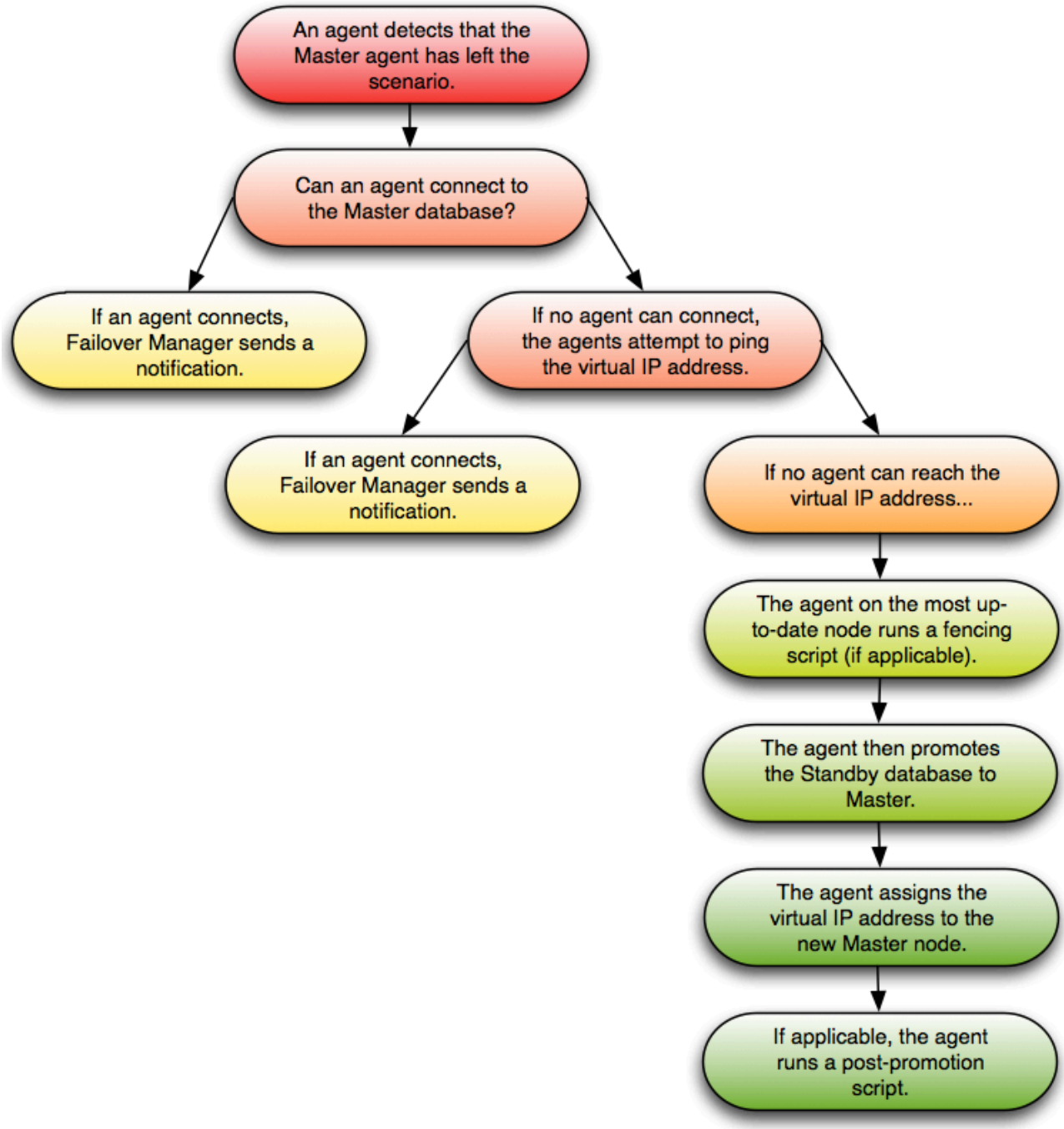
If a Standby agent detects a failure of its database, the agent notifies the other agents; the other agents confirm the state of the database.



After returning the Standby database to a healthy state, invoke the efm resume command to return the Standby to the cluster.

Master Agent Exits or Node Fails

If the Failover Manager Master agent crashes or the node fails, a Standby agent will detect the failure and (if appropriate) initiate a failover.



If an agent detects that the Master agent has left, all agents attempt to connect directly to the Master database. If any agent can connect to the database, an agent sends a notification about the failure of the Master agent. If no agent can connect, the agents attempt to ping the virtual IP address to determine if it has been released.

If no agent can reach the virtual IP address or the database server, Failover Manager starts the failover process. The Standby agent on the most up-to-date node runs a fencing script (if applicable), promotes the Standby database to Master database, and assigns the virtual IP address to the Standby node; if applicable, the agent runs a post-promotion script. Any additional Standby nodes are configured to replicate from the new master unless auto.reconfigure is set to false.

If this scenario has occurred because the master has been isolated from network, the Master agent will detect the isolation and release the virtual IP address and create the recovery.conf file. Failover Manager will perform the previously listed steps on the remaining nodes of the cluster.

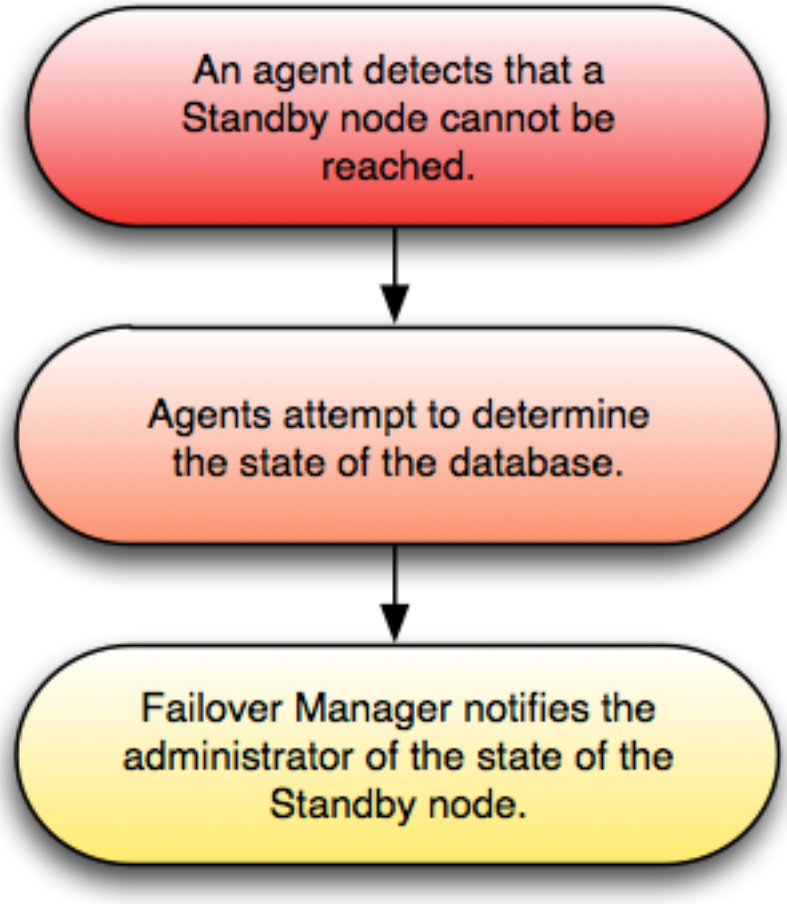
To recover from this scenario without restarting the entire cluster, you should:

1. Restart the original Master node.
2. Bring the original Master database up as a Standby node.
3. Start the service on the original Master node.

Please note that stopping an agent does not signal the cluster that the agent has failed.

Standby Agent Exits or Node Fails

If a Standby agent exits or a Standby node fails, the other agents will detect that it is no longer connected to the cluster.

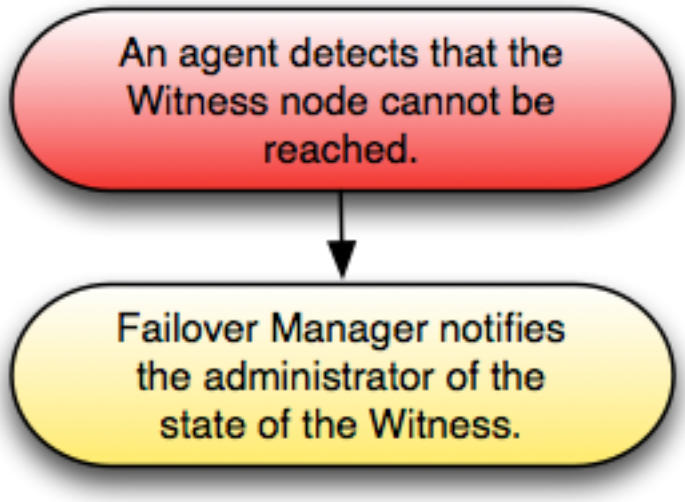


When the failure is detected, the agents attempt to contact the database that resides on the node; if the agents confirm that there is a problem, Failover Manager sends the appropriate notification to the administrator.

If there is only one Master and one Standby remaining, there is no failover protection in the case of a Master node failure. In the case of a Master database failure, the Master and Standby agents can agree that the database failed and proceed with failover.

Dedicated Witness Agent Exits / Node Fails

The following scenario details the actions taken if a dedicated Witness (a node that is not hosting a database) fails.

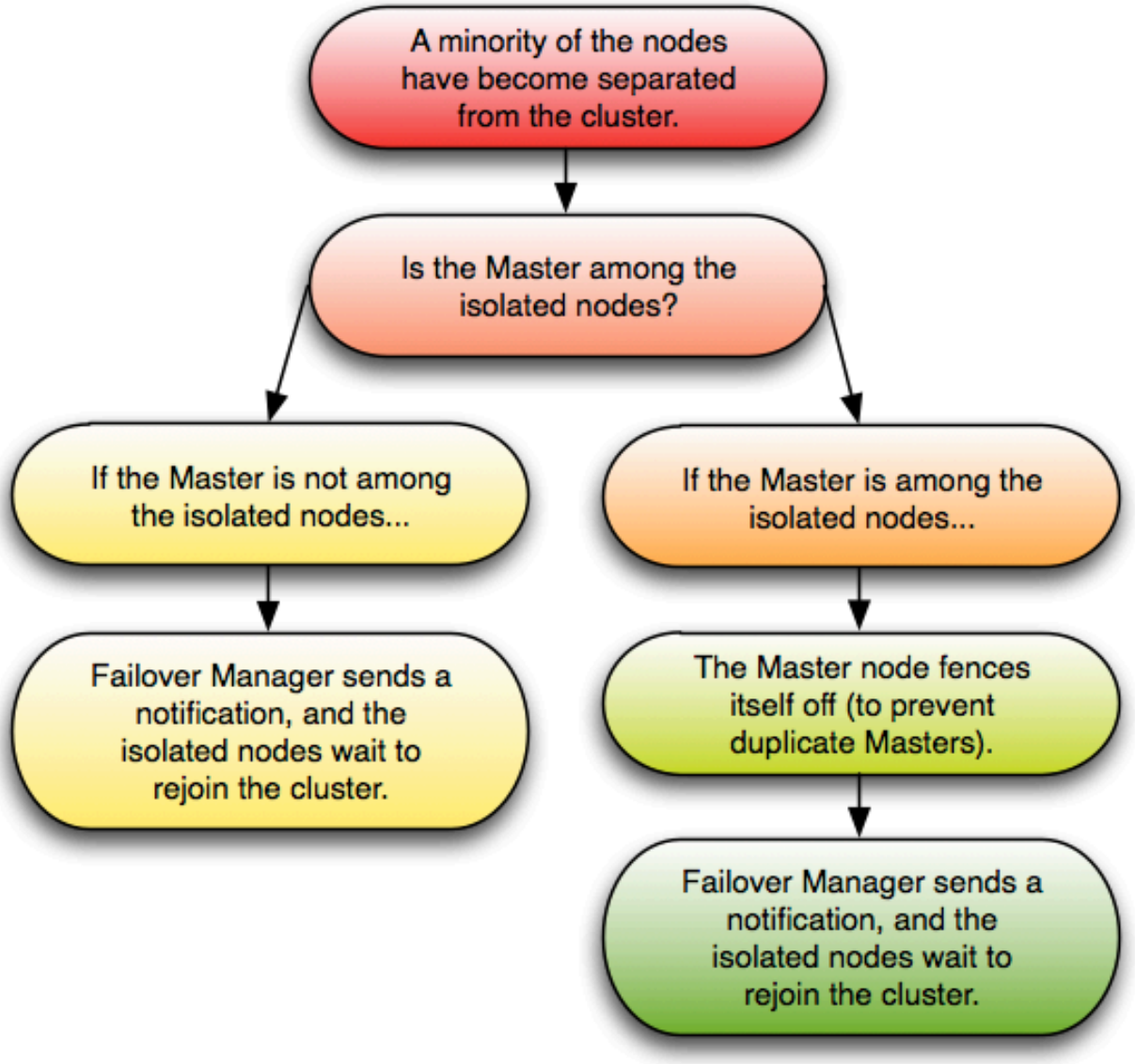


When an agent detects that the Witness node cannot be reached, Failover Manager notifies the administrator of the state of the Witness.

Note: If there is only one Master and one Standby remaining, there is no failover protection in the case of a Master node failure. In the case of a Master database failure, the Master and Standby agents can agree that the database failed and proceed with failover.

Nodes Become Isolated from the Cluster

The following scenario details the actions taken if one or more nodes (a minority of the cluster) become isolated from the majority of the cluster.



If one or more nodes (but less than half of the cluster) become isolated from the rest of the cluster, the remaining cluster behaves as if the nodes have failed. The agents attempt to discern if the Master node is among the isolated nodes; it is, the Master fences itself off from the cluster, while a Standby node (from within the cluster majority) is promoted to replace it. Other Standby nodes are configured to replicate from the new master unless auto.reconfigure is set to false.

Failover Manager then notifies an administrator, and the isolated nodes rejoin the cluster when they are able. When the nodes rejoin the cluster, the failover priority may change.

1.12 Upgrading an Existing Cluster

Failover Manager provides a utility to assist you when upgrading a Failover Manager cluster. To upgrade an existing cluster, you must:

1. Install Failover Manager 3.7 on each node of the cluster. For detailed information about installing Failover Manager, see `Installing Failover Manager <installing_efm>`
2. After installing Failover Manager, invoke the `efm upgrade-conf` utility to create the `.properties` and `.nodes` files for Failover Manager 3.7. The Failover Manager installer adds the upgrade utility (efm upgrade-conf) to the `/usr/edb/efm-3.7/bin directory` . To invoke the utility, assume root

privileges, and invoke the command:

```
efm upgrade-conf cluster_name
```

The efm `upgrade-conf` utility locates the `.properties` and `.nodes` files of pre-existing clusters and copies the parameter values to a new configuration file for use by Failover Manager. The utility saves the updated copy of the configuration files in the `/etc/edb/efm-3.7` directory.

3. Modify the `.properties` and `.nodes` files for EFM 3.7, specifying any new preferences.

   Use your choice of editor to modify any additional properties in the properties file (located in the `/etc/edb/efm-3.7` directory) before starting the service for that node. For detailed information about property settings, see `The Cluster Properties File <cluster_properties>` .

Note

As of version 3.6, `db.bin` is a required property. When modifying the properties file, ensure that the `db.bin` property specifies the location of the Postgres `bin` directory.

1. Use a version-specific command to stop the old Failover Manager cluster; for example, you can use the following command to stop a version 3.6 cluster:

   ```
   /usr/efm-3.6/bin/efm stop-cluster efm
   ```

2. Start the new `Failover manager service <controlling_efm_service>` (`efm-3.7`) on each node of the cluster.

The following example demonstrates invoking the upgrade utility to create the `.properties` and `.nodes` files for a Failover Manager installation:

```
# efm upgrade-conf example
Checking directory /etc/edb/efm-3.6
Processing example.properties file
The following properties were added in addition to those in previous installed version:
   db.config.dir
Checking directory /etc/edb/efm-3.6
Processing example.nodes file
Upgrade of files is finished. The owner and group for properties and nodes files have been set as 'e
```

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.

Please note: If you are using a custom service script or unit file, you must manually update the file to reflect the new Failover Manager service name when you perform an upgrade.

Un-installing Failover Manager

After upgrading to Failover Manager 3.7, you can use Yum to remove previous installations of Failover Manager. For example, use the following command to remove Failover Manager 3.6 and any unneeded dependencies:

```
yum remove edb-efm36
```

Performing a Database Update (Minor Version)

This section describes how to perform a quick minor database version upgrade. You can use the steps that follow to upgrade from one minor version to another (for example, from 10.1.5 to version 10.2.7), or to apply a patch release for a version.

You should first update the database server on each Standby node of the Failover Manager cluster. Then, perform a switchover, promoting a Standby node to the role of Master within the Failover Manager cluster. Then, perform a database update on the old master node.

On each node of the cluster you must perform the following steps to update the database server:

1. Stop the Failover Manager agent.
2. Stop the database server.
3. Update the database server.
4. Start the database service.
5. Start the Failover Manager agent.

For detailed information about controlling the Advanced Server service, or upgrading your version of Advanced Server, please see the EDB Postgres Advanced Server Guide, available at:

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

When your updates are complete, you can use the efm set-priority command to add the old master to the front of the standby list, and then switchover to return the cluster to its original state.

1.13 Troubleshooting

If you receive a notification message about an unexpected error message, check the `Failover Manager log file <con` for an `OutOfMemory` message. Failover Manager runs with the default memory value set by this property:

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

If you are running with less than 128 megabytes allocated, you should increase the value and restart the Failover Manager agent.

Failover Manager is tested with OpenJDK; we strongly recommend using OpenJDK. You can use the following command to check the type of your Java installation:

```
# 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)
```

1.14 Configuring Streaming Replication

Configuring a replication scenario can be complex; for detailed information about configuration options, please see the PostgreSQL core documentation, available at:

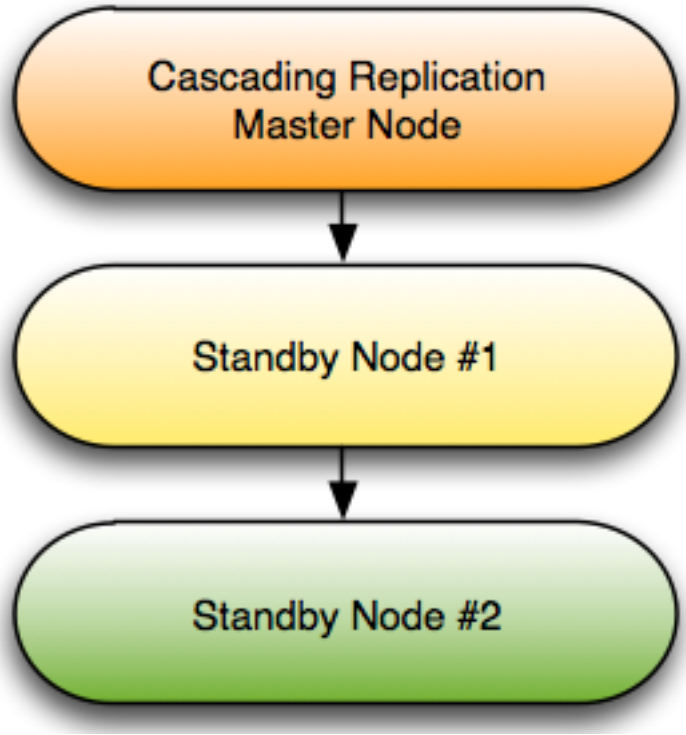
https://www.postgresql.org/docs/12/static/warm-standby.html#streaming-replication

You may want to use a `.pgpass` file to enable md5 authentication for the replication user – this may or may not be the safest authentication method for your environment. For more information about the supported authentication options, please see the PostgreSQL core documentation at:

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

Limited Support for Cascading Replication

While Failover Manager does not provide full support for cascading replication, it does provide limited support for simple failover in a cascading replication scenario. Cascading replication allows a Standby node to stream to another Standby node, reducing the number of connections (and processing overhead) to the master node.



For detailed information about configuring cascading replication, please see the PostgreSQL documentation at:

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

To use Failover Manager in a cascading replication scenario, you should modify the cluster properties file, setting the following property values on Standby Node #2:

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

In the event of a Failover, Standby Node #1 will be promoted to the role of Master node. Should failover occur, Standby Node #2 will continue to act as a read-only replica for the new Master node until you take actions to manually reconfigure the replication scenario to contain 3 nodes.

In the event of a failure of Standby Node #1, you will not have failover protection, but you will receive an email notifying you of the failure of the node.

Please note that performing a switchover and switch back to the original master may not preserve the cascading replication scenario.

1.15 Configuring SSL Authentication on a Failover Manager Cluster

The following steps enable SSL authentication for Failover Manager. Please note that all connecting clients will be required to use SSL authentication when connecting to any database server within the cluster; you will be required to modify the connection methods currently used by existing clients.

To enable SSL on a Failover Manager cluster, you must:

1. Place a `server.crt` and `server.key` file in the `data` directory (under your Advanced Server installation). You can purchase a certificate signed by an authority, or create your own self-signed certificate. For information about creating a self-signed certificate, see the PostgreSQL core documentation at:

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

2. Modify the `postgresql.conf` file on each database within the Failover Manager cluster, enabling SSL:

```
ssl=on
```

After modifying the postgresql.conf file, you must restart the server.

3. Modify the `pg_hba.conf` file on each node of the Failover Manager cluster, adding the following line to the beginning of the file:

   ```
   hostnossl all all all reject
   ```

   The line instructs the server to reject any connections that are not using SSL authentication; this enforces SSL authentication for any connecting clients. For information about modifying the pg_hba.conf file, see the PostgreSQL core documentation at:

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

4. After placing the server.crt and server.key file in the data directory, convert the certificate to a form that Java understands; you can use the command:

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

   For more information, visit:

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

1. Then, add the certificate to the Java trusted certificates file:

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

   Where

   `$JAVA_HOME` is the home directory of your Java installation.

   <alias_name> can be any string, but must be unique for each certificate.

   You can use the `keytool` command to review a list of the available certificates or retrieve information about a specific certificate. For more information about using the keytool command, enter:

   ```
   man keytool
   ```

   The certificate from each database server must be imported into the trusted certificates file of each agent. Note that the location of the cacerts file may vary on each system. For more information, visit:

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

1. Modify the efm.properties file on each node within the cluster, setting the `jdbc.sslmode` property.