EDB Postgres Failover Manager Guide

EDB Postgres Failover Manager Version 2.1.1

February 24, 2017

EDB Postgres Failover Manager Guide, Version 2.1.1
by EnterpriseDB Corporation

EnterpriseDB Corporation, 34 Crosby Drive Suite 100, Bedford, MA 01730, USA
T +1 781 357 3390 **F** +1 978 589 5701 **E** info@enterprisedb.com **www**.enterprisedb.com

Table of Contents

1 Introduction

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 2.1.

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

http://www.enterprisedb.com/documentation

1.1 What's New

The following changes have been made to EDB Postgres Failover Manager 2.0 to create EDB Postgres Failover Manager 2.1:

- Failover Manager now supports use of a notification script for user notifications. For more information, see Section 3.2.1.1.
- Failover Manager allows you to pass in values to scripts that are invoked during Failover. For more information, see Section 3.2.1.1.
- Failover Manager now simplifies cluster startup with the `auto.allow.hosts` property. For more information, see Section 3.2.1.1.
- Failover Manager now includes the `minimum.standbys` parameter, allowing you to specify how many standbys must remain in the replication scenario in the event of failover. For more information, see Section 3.2.1.1.
- Failover Manager includes the `auto.resume.period` property; the property instructs Failover Manager to recheck the health of a failed database. For more information, see Section 3.2.1.1.
- Failover Manager can use either pg_ctl or the operating system service script to restart the server. For more information, see Section 3.2.1.1.
- The Failover Manager Membership Coordinator simplifies adding a new standby node to an existing cluster. For more information, see Section 3.2.2.
- `efm promote` now includes a `-switchover` option; the `-switchover` option instructs Failover Manager to perform a failover, promoting a Standby to Master, and then, return the Master node to the cluster as a Standby node. For more information, see Section 5.3.
- The `efm add-node` command has been deprecated; while the syntax is still supported, the command is now replaced by `efm allow-node`. For more information, see Section 5.3.
- The `efm remove-node` command has been deprecated; while the syntax is still supported, the command is now replaced by `efm disallow-node`. For more information, see Section 5.3.
- The `efm set-priority` command allows you to set a failover priority for a Standby node. For more information, see Section 5.3.

- Failover Manager provides an upgrade utility that copies configuration information from an existing EFM 2.0 installation into EFM 2.1 configuration files. For more information, see Section 9.

1.2 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the proceeding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

2 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.

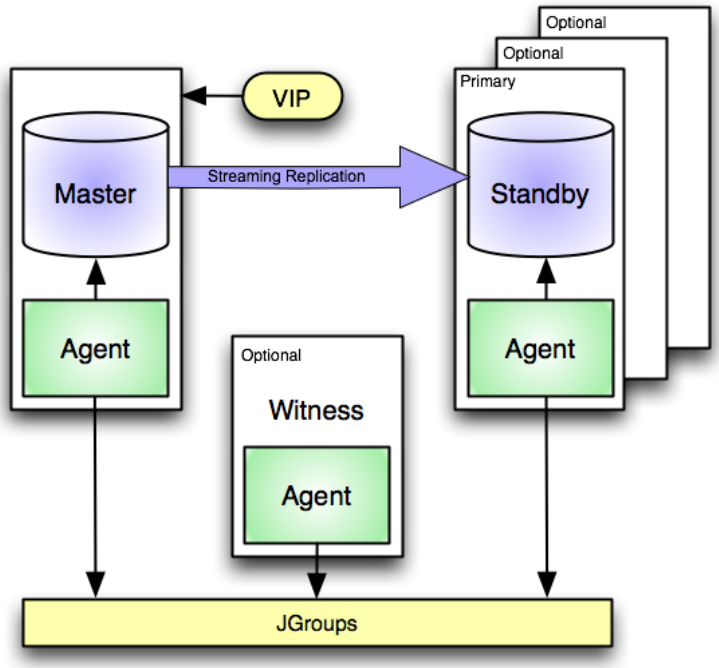


Figure 2.1 - A FM scenario employing a Virtual IP address.

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. For more information about JGroups, visit the official project site at:

http://www.jgroups.org

Figure 2.1 illustrates a Failover Manager cluster that employs a virtual IP address. You can use a load balancer in place of a virtual IP address if you provide your own fencing script to re-configure the load balancer in the event of a failure. For more information about using Failover Manager with a virtual IP address, see Section 3.3. For more information about using a fencing script, see Section 3.2.

2.1 Supported Platforms

Failover Manager 2.1 is supported on EDB Postgres Advanced Server or PostgreSQL (version 9.2 and higher) 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

2.2 Prerequisites

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

Provide an SMTP Server

You can receive notifications from Failover Manager as specified by a user-defined notification script, by email, or both. 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. If you are using email notifications, an SMTP server must be running on each node of the Failover Manager scenario. For more information about using an SMTP server, visit:

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

For more information, see Section 3.2.1.1.

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. For more information about streaming replication, see Appendix A - Configuring Streaming Replication.

During switchover, a `recovery.conf` file is copied from a random standby node to the stopped master. You should ensure that the paths within the `recovery.conf` files on your standby nodes are consistent.

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.

For more information about the `properties` file, see Section 3.2.

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:

```
# /etc/init.d/ppas-9.x reload
```

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 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` 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.

Install Java 1.6 (or later)

Before using Failover Manager, you must first install Java (version 1.6 or later). You can use the Yum package manager to install Java. For example, to install Java 1.7, open a terminal window, assume superuser privileges, and enter:

```
# yum install java-1.7.0
```

Ensure that the db.user has Sufficient Privileges

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

```
version()
pg_is_in_recovery()
pg_current_xlog_location()
pg_last_xlog_replay_location()
pg_xlog_replay_pause()
pg_is_xlog_replay_paused()
pg_xlog_replay_resume()
```

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

http://www.enterprisedb.com/products-services-training/products/documentation/enterpriseedition

2.3 Failover Manager Product Keys

The initial installation of Failover Manager does not require a product key. Upon installation you are granted full access to the cluster monitoring and failover features under a Limited Use License for evaluation purposes for a 60-day trial period.

At the end of the trial period, you are required to either un-install the software or purchase a valid subscription. In addition, at the end of the trial period, the Failover Manager agents will exit making cluster monitoring and failover non-operational. Failover Manager will send multiple email alerts alerting your system administrator of the approaching end of trial period.

To use Failover Manager beyond the evaluation period, you are required to purchase a database server subscription (i.e. Standard Edition with the Failover Manager option or Enterprise Edition) from EnterpriseDB. With the purchase of a subscription you will receive a product key, which will re-enable Failover Manager's cluster monitoring and failover operations. The product key is applied to the Failover Manager configuration file and will restore full functionality for the term of your subscription.

After purchasing a product key, edit the cluster properties file, adding the value to the right of the `efm.license` parameter:

```
efm.license=license_key
```

Where *`license_key`* specifies the product key you received from EnterpriseDB.

You do not need to restart the agents after adding the product key. Every 6 hours, Failover Manager agents check for and validate the product key, so the parameter update will be detected dynamically. For more information about editing the properties file, see Section 3.2.1.

To purchase a database subscription that includes Failover Manager, contact one of the EnterpriseDB offices listed at:

http://www.enterprisedb.com/company/offices

or contact EnterpriseDB at:

sales@enterprisedb.com

3 Installing and Configuring Failover Manager

Creating a Failover Manager cluster is a relatively simple process. After creating a Postgres streaming replication scenario and ensuring that the nodes have sufficient permissions to communicate with each other, you must:

1. Use Yum to install a Failover Manager agent on each node in the cluster.

   Failover Manager is packaged and delivered as an RPM. To install Failover Manager, copy the RPM package to the Master, Standby and Witness systems. After copying the archive to each system, use `yum` to install the package:

   ```
   # yum install efm21-x.x.x-x.distribution.rpm
   ```

   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 Section 3.1, *Extending Failover Manager Permissions*.

2. Modify the cluster properties file on each node. For detailed information about modifying the cluster properties file, see Section 3.2.1.

3. Modify the cluster members file on each node. For more information about the cluster members file, see Section 3.2.2.

4. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.

5. Start the Failover Manager agent on each node of the cluster. For more information about using the Failover Manager service, see Section 5.

Failover Manager File Locations

Failover Manager components are installed in the following locations:

- Executables: `/usr/efm-2.1/bin`
- Libraries: `/usr/efm-2.1/lib`
- Cluster configuration files: `/etc/efm-2.1`
- Logs: `/var/log/efm-2.1`
- Lock files: `/var/lock/efm-2.1`
- Log rotation file: `/etc/logrotate.d/efm-2.1`
- sudo configuration file: `/etc/sudoers.d/efm-21`

3.1 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-21` file is located in `/etc/sudoers.d`, and contains the following entries:

```
# Copyright EnterpriseDB Corporation, 2014-2017. 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/efm-2.1/bin/efm_db_functions
efm ALL=(enterprisedb) NOPASSWD: /usr/efm-2.1/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/efm-2.1/bin/efm_root_functions

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

efm   ALL=(ALL)   NOPASSWD:   /usr/efm-2.1/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-21` 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
```

3.2 Configuring Failover Manager

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

- efm.properties
- efm.nodes

The efm.properties file contains the properties of the individual node on which it resides, while the efm.nodes file contains a list of the current Failover Manager cluster members.

By default, the installer places the files in the /etc/efm-2.1 directory.

3.2.1 The Cluster Properties File

The Failover Manager installer creates a file template for the cluster properties file named efm.properties.in in the /etc/efm-2.1 directory. After completing the Failover Manager installation, you must make a working copy of the template before modifying the file contents.

The following command copies the efm.properties.in file, creating a properties file named efm.properties:

```
# cp /etc/efm-2.1/efm.properties.in /etc/efm-2.1/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 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 parameter, see Section 3.2.1.1.

The property files are owned by root. The Failover Manager service script expects to find the files in the /etc/efm-2.1 directory. If you move the property file to another location, you must create a symbolic link that specifies the new location.

Note that you must use the efm encrypt command to encrypt the value supplied in the db.password.encrypted parameter. For more information about encrypting a password, see Section 3.2.1.2.

3.2.1.1 Specifying Cluster Properties

You can use the parameters listed in the cluster properties file to specify connection properties and behaviors for your Failover Manager cluster. Modifications to configuration parameter settings will be applied when Failover Manager starts. If you modify a parameter value (with the exception of the `efm.license` parameter) 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 the parameter 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 parameters in the `efm.properties` file to specify connection, administrative, and operational details for Failover Manager.

Use the `efm.license` parameter to provide the Failover Manager product key:

```
# The full license to run failover manager.

efm.license=
```

The trial period is 60 days. When there are five (or fewer) days left in the trial period, Failover Manager will send an email warning you that it is time to provide a valid license number. If you have not provided a product key before the trial period expires, all Failover Manager agents will exit.

You do not need to restart the agents after adding the product key to the properties file. Every six hours the Failover Manager agent will attempt to locate and validate the product key.

Use the following parameters to specify connection properties for each node of 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 Section 2.2.

For information about encrypting the password for the database user, see 3.2.1.2.

Use the db.service.owner parameter 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.conf.dir' specified below.

db.service.owner=
```

Specify the name of the database server in the db.service.name parameter 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 instance, 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 unless db.bin is set.

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 parameter.

```
# Specify the directory containing the pg_ctl command, for
# example: /usr/pgsql-9.3/bin. The pg_ctl command is used to
# restart standby databases after a failover so that they are
# streaming from the new master node. 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 unless
# db.service.name is set.

db.bin=
```

Use the `db.recovery.conf.dir` parameter 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 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.

db.recovery.conf.dir=
```

Use the `jdbc.ssl` parameter to instruct Failover Manager to use SSL connections. If you have enabled SSL, use the `jdbc.ssl.mode` parameter to specify behaviors related to server certificates.

```
# Use the jdbc.ssl property to enable ssl for EFM connections.
# Setting this property to true will force the agents to use
# 'ssl=true' for all JDBC database connections (to both local
# and remote databases).
# When jdbc.ssl is true (and ssl is enabled), the jdbc.ssl.mode
# property will determine how server certificates are handled.
# Valid values are:
#
# verify-ca - EFM will perform CA verification before allowing
#             the certificate. This is the default value in case
#             ssl is used and the mode property is not set.
# require   - Verification will not be performed on the server
#             certificate.

jdbc.ssl=false
jdbc.ssl.mode=verify-ca
```

For information about configuring and using SSL, please see:

https://www.postgresql.org/docs/9.5/static/ssl-tcp.html
and
https://jdbc.postgresql.org/documentation/94/ssl.html

Use the `user.email` parameter 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=
```

Use the `script.notification` parameter 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` parameter 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.

bind.address=
```

Use the `admin.port` parameter 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).

admin.port=
```

Set the `is.witness` parameter 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.

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

The `local.period` parameter specifies how many seconds between attempts to contact the database server.
The `local.timeout` parameter specifies how long an agent will wait for a positive response from the local database server.
The `local.timeout.final` parameter 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` parameter, the database is assumed to have failed.

For example, given the default values of these parameters, 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` parameter. 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` parameter 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` parameter 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` parameter value specifies a timeout value for agent-to-agent communication; other timeout parameters 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 `pingServer` parameter 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 parameter 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 parameter 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 parameter (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
```

The db.reuse.connection.count parameter 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 parameter 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` parameter to instruct Failover Manager to enable or disable automatic reconfiguration of remaining Standby servers after the primary standby is promoted to Master. Set the parameter to `true` to enable automatic reconfiguration (the default) or `false` to disable automatic reconfiguration. This property is not required on a dedicated witness node.

```
# 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.conf.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`. For more information, see Section 2.2.

Use the `promotable` parameter 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 Section 5.3.

```
# 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
```

Use the `minimum.standbys` parameter 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` parameter 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 `auto.resume.period` parameter to specify the number of seconds (after a monitored database fails, and an agent has assumed an IDLE state) that an agent will attempt to resume monitoring that database.

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

auto.resume.period=0
```

Use the `virtualIp` parameter to specify virtual IP address information for the Failover Manager cluster.

Use the `virtualIp.interface` parameter to specify an alias for your network adaptor (for example, `eth0:1` specifies an alias for the adaptor, `eth0`). You might create multiple aliases for each adaptor on a given host; for more information about running multiple agents on a single node, please see Section 4.3.

Use the `virtualIp.netmask` parameter to specify which bits in the virtual IP address refer to the network address (as opposed to the host address).

For information about using a virtual IP address, see Section 3.3.

```
# This is the IP and netmask/prefix length that will be remapped
# during failover.  If you do not use a VIP as part of your
# failover solution, then leave these properties blank to disable
# EFM's support for VIP processing (assigning, releasing, testing
# reachability, etc).
#
# If you enable VIP, then all three properties are required.
#
# The virtualIP and virtualIP.netmask values must be the same
# across all agents. If using an IPv6 address, the
# virtualIp.netmask value must be the prefix length instead. If
```

```
# using an IPv4 address, the virtualIp.interface value must
# contain the secondary virtual ip id (ie ":1", etc).

virtualIp=
virtualIp.interface=
virtualIp.netmask=
```

`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. The script is run as the
# efm user.
#
# 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=
```

Please note that the fencing script runs as the `efm` user; you must ensure that the `efm` user has sufficient privileges to invoke any commands included in the fencing script. For more information about Failover Manager permissions, please see Section 3.1.

Use the `script.post.promotion` parameter 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.  The script is run
# as the efm user.
#
# 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` parameter 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 whenever an IDLE agent is resumed and starts
# monitoring its local database.

script.resumed=
```

Use the `script.db.failure` parameter 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. The script
# is run as the efm user.

script.db.failure=
```

Use the `script.master.isolated` parameter 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. The script is run as the efm user.

script.master.isolated=
```

Use the `sudo.command` parameter 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 parameter 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/efm-<version>/bin/efm_address <arguments>
# sudo /usr/efm-<version>/bin/efm_root_functions <arguments>
# sudo -u <db service owner>
           /usr/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 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 Section 6, *Controlling Logging*.

```
# Logging levels for JGroups and EFM.
# Valid values are: FINEST, FINER, FINE, CONFIG, INFO, WARNING,
# SEVERE
# 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 FINER 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 parameter 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=-Xmx32m
```

3.2.1.2 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/efm-2.1/bin` directory) to encrypt the password; open a command line, and enter the command:

```
# efm encrypt cluster_name
```

Where *`cluster_name`* specifies the name of the Failover Manager cluster.

The Failover Manager service 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.

Please enter the password and hit enter:
Please enter the password again to confirm:

The encrypted password is: 835fb18954f198e94fd3d6f4b070350b

Please paste this into your cluster properties file.
    db.password.encrypted=835fb18954f198e94fd3d6f4b070350b
```

After receiving your encrypted password, paste the password into the cluster 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-2.1 start
Starting local efm-2.1 service:      [FAILED]
```

If you receive this message when starting the Failover Manager service, please see the startup log (located in `/var/log/efm-2.1/startup-efm.log`) for more information.

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

```
systemctl status efm-2.1
```

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.

3.2.2 The Cluster Members File

Each node in a Failover Manager cluster has a cluster members file. 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/efm-2.1` directory. After completing the Failover Manager installation, you must make a working copy of the template:

```
# cp /etc/efm-2.1/efm.nodes.in /etc/efm-2.1/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.

If the Membership Coordinator leaves the cluster, another node will assume the role of Membership Coordinator. You can use the `efm cluster-status` command to find the address of the Membership Coordinator.

Please note: 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` parameter (in the cluster properties file) is set to `true`. For more information, see Section 4.1.2.

A running agent 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 cluster member file.

If nodes join or leave a cluster while an agent is down, you must manually ensure that the file includes at least the current Membership Coordinator.

3.3 Using Failover Manager with Virtual IP Addresses

Failover Manager uses the `efm_address` script to assign or release a virtual IP address. By default, the script resides in:

```
/usr/efm-2.1/bin/efm_address
```

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

To assign a virtual IPv4 IP address:

```
# efm_address assign interface_name IPv4_addr netmask
```

To assign a virtual IPv6 IP address:

```
# efm_address assign6 interface_name IPv6_addr prefix_len
```

To release a virtual IPv4 address:

```
# efm_address release interface_name
```

To release a virtual IPv6 address:

```
# efm_address release6 interface_name IPv6_addr prefix_len
```

Where:

interface_name matches the name specified in the `virtualIp.interface` parameter in the cluster properties file.

IPv4_addr or *IPv6_addr* matches the name specified in the `virtualIp` parameter in the cluster properties file.

netmask or *prefix_len* matches the value specified in the `virtualIp.netmask` parameter in the cluster properties file.

For more information about properties that describe a virtual IP address, see Section 3.2.1.

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 Section 3.1, *Extending Failover Manager Permissions*.

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=eth0:0
virtualIp.netmask=255.255.255.0
pingServerCommand=/bin/ping -q -c3 -w5
```

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 assign` command on the Master node to assign the VIP and then confirm with `ifconfig`:

```
# efm_address assign eth0:0 172.24.38.239 255.255.255.0
# ifconfig
<output truncated>
eth0:0    Link encap:Ethernet  HWaddr 36:AA:A4:F4:1C:40
          inet addr:172.24.38.239  Bcast:172.24.38.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          Interrupt:247
```

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 release` command to release the address on the master node and confirm the node has been released with `ifconfig`:

```
# efm_address release eth0:0
# ifconfig
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 eth0:0 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 assign eth0:0 172.24.38.239 255.255.255.0
# ifconfig
<output truncated>
eth0:0    Link encap:Ethernet  HWaddr 36:AA:A4:F4:1C:40
          inet addr:172.24.38.239  Bcast:172.24.38.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          Interrupt:247
```

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

4 Using Failover Manager

Failover Manager 2.1 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.

For failover protection, the initial cluster should have a master and one or more standby nodes. If the cluster contains more than one Standby node, a dedicated Witness node is not required. If there is only one Standby node, the cluster should include a dedicated Witness node.

Before configuring and starting Failover Manager, configure PostgreSQL Streaming Replication between the Master node and one or more Standby nodes on an Advanced Server or PostgreSQL installation. For more information about implementing streaming replication, see the PostgreSQL core documentation at

http://www.enterprisedb.com/warm-standby.html.

Before starting a Failover Manager agent, you must install Failover Manager (see Section 3), and specify your configuration preferences in the cluster properties file (see Section 3.2.1) and the cluster members file (see Section 3.2.2) on the server on which the agent will be running.

> **Using WAL Archiving on Advanced Server 9.2 or PostgreSQL 9.2 Instances**
>
> If you use WAL archiving on your Postgres 9.2 instance, and a failover occurs, Failover Manager does not automatically reconfigure WAL archiving on the new master node. You must manually configure WAL archiving on the new master node of your cluster to ensure that it will work properly during the next failover.

Please Note: 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 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` file exists, and the database will not start as a second master.

4.1 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 commands listed below must be invoked by `efm` or by 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 stop-cluster`
- `efm promote`
- `efm resume`
- `efm upgrade-conf`

4.1.1 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-2.1 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-2.1
```

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.

4.1.2 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 Section 5.

2. Install a Failover Manager agent and configure the cluster properties file on the new node. For more information about modifying the properties file, see Section 3.2.1.

3. 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 Section 3.2.2.

4. 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-2.1 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-2.1
   ```

When the new node joins the cluster, Failover Manager will send a notification to the administrator email provided in the `user.email` parameter, 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.

4.1.3 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 `10.0.1.9:7800` is the primary Standby (`1`):

```
efm set-priority acctg 10.0.1.9:7800 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.

4.1.4 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. Include the `–switchover` option to reconfigure the original Master as a Standby.

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]
```

Please note that 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. During switchover:

- The master database is stopped.
- A `recovery.conf` file is copied from an existing standby to the master node.
- The address of the new master node is added to the `recovery.conf` file.
- The old master is restarted; 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.conf.dir` parameter. 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 using the efm utility, see Section 5.3.

4.1.5 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-2.1 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-2.1
```

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-2.1 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.

4.1.6 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`. For detailed information about restarting a Failover Manager cluster, please see Section 4.1.7.

4.1.7 Restarting a Failover Manager Cluster

To restart a cluster, you must:

1. Restart the Failover Manager agent on any node of the cluster.

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

   ```
   efm allow-node cluster_name ip_address
   ```

3. Start a Failover Manager agent on each node of the cluster. When you start the agent, the node will join the cluster.

If a node's address is not in the allowed node host list before you start the Failover Manager agent on a member node, Failover Manager will write the following error to the `startup-efm.log` file:

```
There was an error starting service: authentication failed
If other nodes are already running in the cluster, please verify
that the address for this node is on the allowed node host list.
```

4.1.8 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 (for information about controlling the agent, see Section 5). 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 to add the node to the cluster again. For more information about using the efm utility, see Section 5.3.

4.2 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.

4.2.1 Reviewing the Cluster Status Report

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

```
# efm cluster-status acctg
```

The following status report is for a cluster with five nodes running:

```
Cluster Status: acctg
VIP:

 Agent Type  Address              Agent  DB       Info
 --------------------------------------------------------
 Idle        172.24.38.106        UP     UNKNOWN
 Standby     172.24.38.123        UP     UP
 Standby     172.24.38.103        UP     UP
 Idle        172.24.38.152        UP     UNKNOWN
 Master      172.24.38.163        UP     UP

Allowed node host list:
 172.24.38.106 172.24.38.163 172.24.38.103 172.24.38.123
172.24.38.152

Membership coordinator: 172.24.38.123

Standby priority host list:
 172.24.38.103 172.24.38.123

Promote Status:

 DB Type     Address              XLog Loc         Info
 --------------------------------------------------------
 Master      172.24.38.163        2/35000230
 Standby     172.24.38.103        2/35000230
 Standby     172.24.38.123        2/35000230

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

Idle Node Status (idle nodes ignored in XLog location
comparisons):

 Address              XLog Loc         Info
```

```
 ------------------------------------------------------
 172.24.38.152        2/35000160       DB is not in
recovery.
 172.24.38.106        UNKNOWN          Connection refused.
Check that the hostname and port are correct and that the
postmaster is accepting TCP/IP connections.
```

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

```
Cluster Status: acctg

 Agent Type  Address              Agent  DB       Info
 ------------------------------------------------------
 Idle        172.24.38.106        UP     UNKNOWN
 Standby     172.24.38.123        UP     UP
 Standby     172.24.38.103        UP     UP
 Idle        172.24.38.152        UP     UNKNOWN
 Master      172.24.38.163        UP     UP
```

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 standby nodes:

```
Allowed node host list:
 172.24.38.106 172.24.38.163 172.24.38.103 172.24.38.123
172.24.38.152

Standby priority host list:
 172.24.38.103 172.24.38.123
```

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              XLog Loc         Info
 ------------------------------------------------------
 Master      172.24.38.163        2/35000230
 Standby     172.24.38.103        2/35000230
 Standby     172.24.38.123        2/35000230

 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 node.

```
Idle Node Status (idle nodes ignored in XLog location
comparisons):

 Address              XLog Loc       Info
 ------------------------------------------------------
 172.24.38.152        2/35000160     DB is not in recovery.
 172.24.38.106        UNKNOWN        Connection refused.
Check that the hostname and port are correct and that the
postmaster is accepting TCP/IP connections.
```

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.

4.2.2 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 client graphical interface) to display the state of a Master or Standby node that is part of a Streaming Replication scenario.

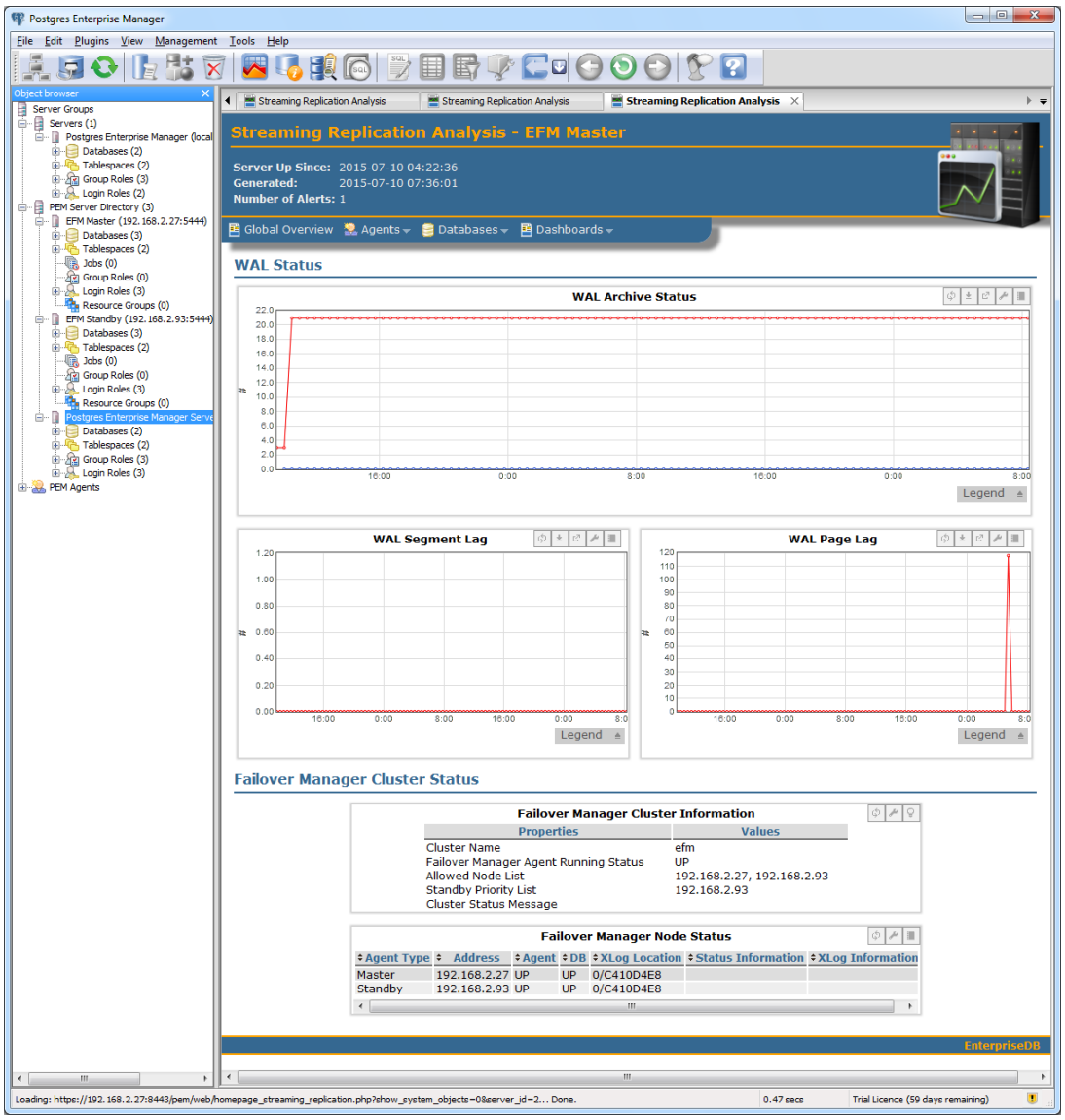


Figure 4.1 - The Streaming Replication dashboard (Master node)

The Streaming Replication Analysis Dashboard (shown in Figure 4.1) 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 (see Figure 4.2).

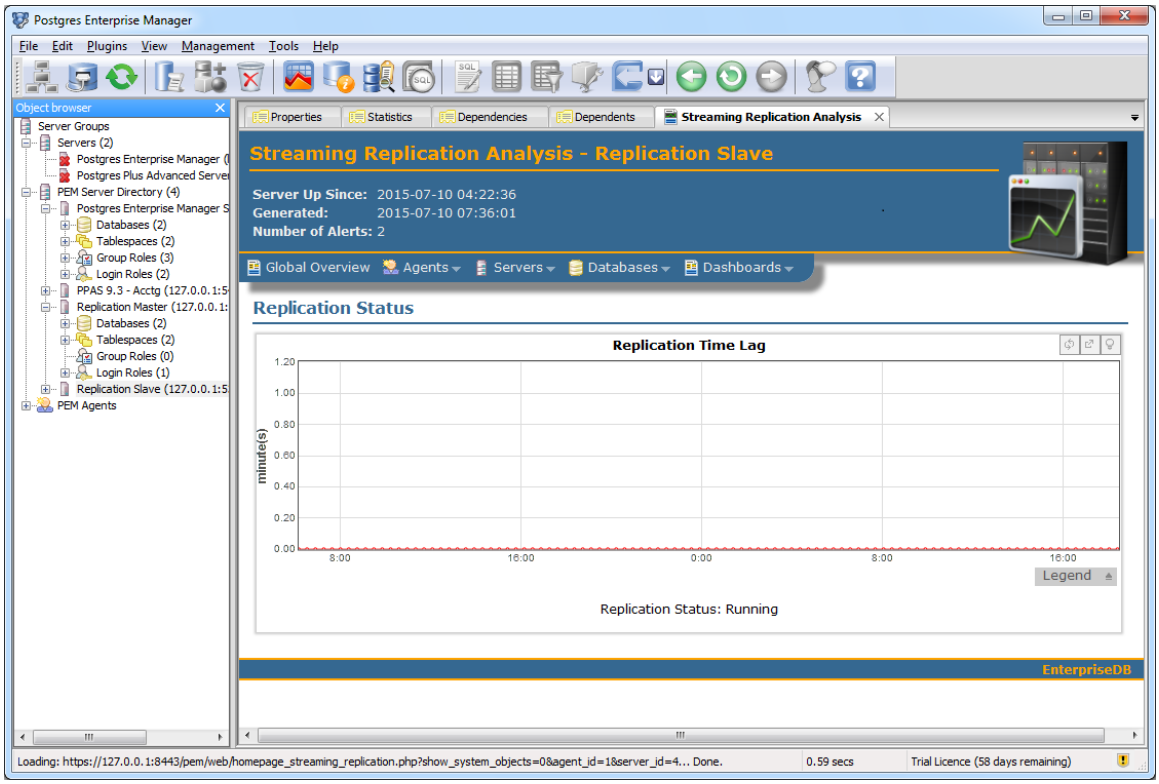


Figure 4.2 - The Streaming Replication dashboard (Standby node)

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

To enable a probe, highlight the name of the replication master in the PEM client `Object browser` pane, and select `Probe Configuration...` from the `Management` menu. Use the `Probe Configuration` dialog to enable each probe.

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

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

4.3 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 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.conf.dir
script.fence (if used)
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.conf.dir` parameters must also specify values that are unique for each respective database cluster.

If you are using a fencing script, use the `script.fence` parameter to identify a fencing script that is unique for each 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`

> You must specify a unique virtual IP address for each cluster. If the same address is used, a failure of one database cluster would cause the address to be released from the master, breaking existing connections to the remaining database cluster.

`virtualIp.interface`

> You must specify a unique interface name for each cluster. For example, `acctg.properties` might include a value of `eth0:0`, while `sales.properties` might specify `eth0:1`.

`virtualIp.netmask`

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 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 Section 4.3.1; if you are using RHEL 7.x or CentOS 7.x, see Section 4.3.2.

4.3.1 RHEL 6.x or CentOS 6.x

If you are using RHEL 6.x or CentOS 6.x, you should copy the `efm-2.1` service script to new file with a name that is unique for each cluster. For example:

```
# cp /etc/init.d/efm-2.1 /etc/init.d/efm-acctg

# cp /etc/init.d/efm-2.1 /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
```

4.3.2 RHEL 7.x or CentOS 7.x

If you are using RHEL 7.x or CentOS 7.x, you should copy the `efm-2.1` service script to new file with a name that is unique for each cluster. For example:

```
# cp /usr/lib/systemd/system/efm-2.1.service
     /usr/lib/systemd/system/efm-acctg.service

# cp /usr/lib/systemd/system/efm-2.1.service
     /usr/lib/systemd/system/efm-sales.service
```

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

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
```

5 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; for information about controlling Failover Manager on a RHEL 6.x or CentOS 6.x host, see Section 5.1. If you are using RHEL 7.x or CentOS 7.x, see Section 5.2.

5.1 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-2.1` 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-2.1 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-2.1 stop`

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

`service efm-2.1 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-2.1 status
efm-2.1 (pid  50836) is running...
```

`service efm-2.1 help`

Display online help for the Failover Manager service script.

5.2 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-2.1.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-2.1
```

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-2.1
```

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

```
systemctl status efm-2.1
```

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-2.1
efm-2.1.service - EnterpriseDB Failover Manager 2.1
   Loaded: loaded (/usr/lib/systemd/system/efm-2.1.service;
disabled)
   Active: active (running) since Tue 2015-04-07 06:05:49
PDT; 25s ago
  Process: 28446 ExecStart=/bin/java -cp /usr/efm-
2.1/lib/EFM-2.1.0.jar
com.enterprisedb.hal.main.ServiceCommand start /etc/efm-
2.1/${CLUSTER}.properties (code=exited, status=0/SUCCESS)
 Main PID: 28456 (java)
   CGroup: /system.slice/efm-2.1.service
           → java -cp /usr/efm-2.1/lib/EFM-2.1.0.jar
com.enterprisedb.hal.main.ServiceCommand __int_start
/etc/efm-2.1/efm
....
```

5.3 Using the efm Utility

Failover Manager provides the `efm` utility to assist with cluster management. The RPM installer adds the utility to the `/usr/efm-2.1/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 Section 4.2.1.

`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": {
        "172.16.144.161": {
            "type": "Idle",
            "agent": "UP",
            "db": "UNKNOWN",
            "info": " ",
            "xlog": "UNKNOWN",
            "xloginfo": "Connection to 172.16.144.161:5432
refused. Check that the hostname and port are correct and that
the postmaster is accepting TCP\/IP connections."
        },
        "172.16.144.160": {
            "type": "Standby",
            "agent": "UP",
            "db": "UP",
            "info": " ",
            "xlog": "3\/80001A18",
            "xloginfo": ""
        },
```

```
            "172.16.144.159": {
                "type": "Master",
                "agent": "UP",
                "db": "UP",
                "info": " ",
                "xlog": "3\/80001A18",
                "xloginfo": ""
            },
            "172.16.144.172": {
                "type": "Standby",
                "agent": "UP",
                "db": "UP",
                "info": " ",
                "xlog": "3\/80001A18",
                "xloginfo": ""
            }
        },
        "allowednodes": [
            "172.16.144.160",
            "172.16.144.159",
            "172.16.144.161",
            "172.16.144.172"
        ],
        "membershipcoordinator": "172.16.144.160",
        "failoverpriority": [
            "172.16.144.160",
            "172.16.144.172"
        ],
        "VIP": "",
        "minimumstandbys": 0,
        "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`*

Invoke the `efm encrypt` command to encrypt the database password before include the password in the cluster properties file.

`efm promote` *`cluster_name`* `[-switchover]`

The `promote` command instructs Failover Manager to perform a manual failover of standby to master. Include the `-switchover` option to promote a standby node, and reconfigure a master node as a standby node.

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.

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 prop-check` *`cluster_name`*

The `efm prop-check` command invokes a utility that may help to identify configuration problems caused by mismatched property files.

When invoked on the Witness node with a cluster named `employees`, the `prop-check` command might display:

```
# efm prop-check employees
Agents: 172.24.38.107:7800
Binding address: 172.24.38.107
I am witness node: true
Cluster name: employees
User email: [user.name@example.com]
Notification script: null
VIP: 172.24.38.239
Automatic failover set to: true
Network adapters: eth0
fe80:0:0:0:10a8:bff:fe7c:70cf%2
172.24.38.185
lo
0:0:0:0:0:0:0:1%1
127.0.0.1
```

`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* 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. For detailed information about restarting a Failover Manager cluster, please see Section 4.1.7.

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

```
efm upgrade-conf cluster_name
```

Invoke the efm upgrade-conf command to copy the configuration files from a Failover Manager 2.0 installation, and add parameters required by a Failover Manager 2.1 installation. Provide the name of the 2.0 cluster when invoking the utility. This command must be invoked with root privileges.

```
efm --help
```

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

6 Controlling Logging

Failover Manager writes and stores one log file per agent and one startup log per agent in `/var/log/efm-2.1`. 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: FINEST, FINER, FINE, CONFIG, INFO,
# WARNING,SEVERE
# 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
# FINER 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:

```
FINEST
FINER
FINE
CONFIG
INFO
WARNING
SEVERE
```

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

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-2.1`). For more information about modifying the log rotation schedule, consult the `logrotate man` page:

```
$ man logrotate
```

7 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
```

For more information about editing the configuration properties, see Section 3.2.1.1.

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 support:   DISABLED

Database health is not being monitored.
```

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.

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_results*
Executed post-promotion script	Executed post-promotion script *script_name* Results: *script_results*
Executed post-database failure script	Executed post-database failure script *script_name* Results: *script_results*
Executed master isolation script	Executed master isolation script *script_name* Results: *script_results*
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 monitored.
Standby agent running on *node_address* for cluster *cluster_name*	Standby agent is running and database health is being monitored.
Idle agent running on node *node_address* for cluster *cluster_name*	Idle agent is running. After starting the local database, the agent can be resumed.
Assigning VIP to node *node_address*	Assigning VIP *VIP_address* to node *node_address* Results: *script_results*
Releasing VIP from node *node_address*	Releasing VIP *VIP_address* from node *node_address* Results: *script_results*
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 master agent is restarted.
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 in the Idle state.
Starting auto resume check for cluster *cluster_name*	The agent on this node will check every *auto.resume.period* seconds to see if it can resume monitoring the failed database. The cluster should be checked during this time and the agent stopped if the database will not be started again. See the agent log for more details.
Executed agent resumed script	Executed agent resumed script *script_name* Results: *script_results*
Agent exited for cluster *cluster_name*	The agent has exited. This notification is usually generated during startup when an agent exits before startup has completed.

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

Subject	Description
Virtual IP address assigned to non-master node	The virtual IP address appears to be assigned to a non-master node. To avoid any conflicts, Failover Manager will release the VIP. You should confirm that the VIP is assigned to your master node and manually reassign the address if it is not.
No standby agent in cluster for cluster *cluster_name*	The standbys on *cluster_name* have left the cluster.
Standby agent failed for cluster *cluster_name*	A standby agent on *cluster_name* has left the cluster, but the coordinator has detected that the standby database is still running.
Standby database failed for cluster *cluster_name*	A standby agent has signaled that its database has failed. The other nodes also cannot reach the standby database.
Standby agent cannot reach database for cluster *cluster_name*	A standby agent has signaled database failure, but the other nodes have detected that the standby database is still running.
Cluster *cluster_name* has dropped below three nodes	At least three nodes are required for full failover protection. Please add witness or agent node to the cluster.
Subset of cluster *cluster_name* disconnected from master	This node is no longer connected to the majority of the cluster *cluster_name*. Because this node is part of a subset of the cluster, failover will not be attempted. Current nodes that are visible are: *node_address*
Promotion has started on cluster *cluster_name*.	The promotion of a standby has started on cluster *cluster_name*.
Witness failure for cluster *cluster_name*	Witness running at *node_address* has left the cluster.
Idle agent failure for cluster *cluster_name*.	Idle agent running at *node_address* has left the cluster.
One or more nodes isolated from network for cluster *cluster_name*	This node appears to be isolated from the network. Other members seen in the cluster are: *node_name*
Node no longer isolated from network for cluster *cluster_name*.	This node is no longer isolated from the network.
Standby agent tried to promote, but master DB is still running	The standby EFM agent tried to promote itself, but detected that the master DB is still running on *node_address*. This usually indicates that the master EFM agent has exited. Failover has NOT occurred.
Standby agent started to promote, but master has rejoined.	The standby EFM agent started to promote itself, but found that a master agent has rejoined the cluster. Failover has NOT occurred.
Standby agent tried to promote, but could not verify master DB	The standby EFM agent tried to promote itself, but could not detect whether or not the master DB is still running on *node_address*. Failover has NOT occurred.
Standby agent tried to promote, but VIP appears to still be assigned	The standby EFM agent tried to promote itself, but could not because the virtual IP address (*VIP_address*) appears to still be assigned to another node. Promoting under these circumstances could cause data corruption. Failover has NOT occurred.

Standby agent tried to promote, but appears to be orphaned	The standby EFM agent tried to promote itself, but could not because the well-known server (*server_address*) could not be reached. This usually indicates a network issue that has separated the standby agent from the other agents. Failover has NOT occurred.
Failover has not occurred	An agent has detected that the master database is no longer available in cluster *cluster_name*, but there are no standby nodes available for failover.
Potential manual failover required on cluster *cluster_name*.	A potential failover situation was detected for cluster *cluster_name*. Automatic failover has been disabled for this cluster, so manual intervention is required.
Failover has completed on cluster *cluster_name*	Failover has completed on cluster *cluster_name*.
Lock file for cluster *cluster_name* has been removed	The lock file for cluster *cluster_name* has been removed from: *path_name* on node *node_address*. This lock prevents multiple agents from monitoring the same cluster on the same node. Please restore this file to prevent accidentally starting another agent for cluster.
`recovery.conf` file for cluster *cluster_name* has been found	A recovery.conf file for cluster *cluster_name* has been found at: *path_name* on master node *node_address*. This may be problematic should you attempt to restart the DB on this node.
Promotion has not occurred for cluster *cluster_name*	A promotion was attempted but there is already a node being promoted: *ip_address*.
Standby not reconfigured after failover in cluster *cluster_name*	The `auto.reconfigure` property has been set to false for this node. The node has not been reconfigured to follow the new master node after a failover.
Could not resume replay for cluster *cluster_name*	Could not resume replay for standby being promoted. Manual intervention may be required. Error: *error_decription* This error is returned if the server encounters an error when invoking replay during the promotion of a standby.
Could not resume replay for standby *standby_id*.	Could not resume replay for standby. Manual intervention may be required. Error: *error_message*.
Possible problem with database timeout values	Your `remote.timeout` value (*value*) is higher than your `local.timeout` value (*value*). If the local database takes too long to respond, the local agent could assume that the database has failed though other agents can connect. While this will not cause a failover, it could force the local agent to stop monitoring, leaving you without failover protection.
Trial license expiring soon	Your trial license for EDB Postgres Failover Manager will expire on *expiration_date*. A valid product key is required for continued operation once the initial trial period has ended. Without a valid product key, at the end of the trial, Failover Manager will exit and no longer run. Please contact your EnterpriseDB account manager to purchase a license for Failover Manager.

Full license expiring soon	Your Full Use license for EDB Postgres Failover Manager will expire on *expiration_date*. A valid product key is required is required for continued operation once your current license subscription ends. Without a valid product key, at the end of your subscription, Failover Manager will exit and no longer run. Please contact your EnterpriseDB account manager to renew your subscription for Failover Manager.
License has expired	Your license subscription to run this product expired on *expiration_date*. Failover Manager is continuing to run under the trial period. Please contact your EnterpriseDB Account Manager to purchase a new license subscription for Failover Manager.
License is invalid	There is a problem with the Full Use License for EDB Postgres Failover Manager that was provided in the cluster properties file. Please check to be sure it was entered correctly. Failover Manager will continue to run for the duration of the trial period. If this problem persists, please contact your EnterpriseDB Account Manager.

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

Subject	Description
Unable to connect to DB on *node_address*	The maximum connections limit has been reached.
Unable to connect to DB on *node_address*	Invalid password for db.user=*user_name*.
Unable to connect to DB on *node_address*	Invalid authorization specification.
Master cannot ping local database for cluster *cluster_name*	The master agent can no longer reach the local database running at *node_address*. Other nodes are able to access the database remotely, so the master will not release the VIP and/or create a recovery.conf file. The master agent will become IDLE until the resume command is run to resume monitoring the database.
Communication error for cluster *cluster_name*	This node has connected to the cluster, but cannot resolve the IP address for one or more cluster members. Please stop the agent running on *node_address* and verify that all the existing cluster members' addresses are in the .nodes file.
Fencing script error	Fencing script *script_name* failed to execute successfully. Exit Value: *exit_code* Results: *script_results* Failover has NOT occurred.
Post-promotion script failed	Post-promotion script *script_name* failed to execute successfully. Exit Value: *exit_code* Results: *script_results*
Post-database failure script error	Post-database failure script *script_name* failed to execute successfully. Exit Value: *exit_code*

	Results: *script_results*
Agent resumed script error	Agent resumed script *script_name* failed to execute successfully. Results: *script_results*
Master isolation script failed	Master isolation script *script_name* failed to execute successfully. Exit Value: *exit_code* Results: *script_results*
Could not promote standby	The trigger file *file_name* could not be created on node. Could not promote standby. Error details: *message_details*
Error creating `recovery.conf` file on *node_address* for cluster *cluster_name*	There was an error creating the recovery.conf file on master node *node_address* during promotion. Promotion has continued, but requires manual intervention to ensure that the old master node can not be restarted. Error details: *message_details*
An unexpected error has occurred for cluster *cluster_name*	An unexpected error has occurred on this node. Please check the agent log for more information. Error: *error_details*
Master database being fenced off for cluster *cluster_name*	The master database has been isolated from the majority of the cluster. The cluster is telling the master agent at *ip_address* to fence off the master database to prevent two masters when the rest of the failover manager cluster promotes a standby.
Master database being fenced off for cluster *cluster_name*	The master database has been isolated from the majority of the cluster. Before the master could finish detecting isolation, a standby was promoted and has rejoined this node in the cluster. This node is isolating itself to avoid more than one master database.
Could not assign VIP to node *node_address*	Failover manager could not assign the VIP address for some reason.
master_or_standby database failure for cluster *cluster_name*	The database has failed on the specified node.
Agent is timing out for cluster *cluster_name*	This agent has timed out trying to reach the local database. After the timeout, the agent could successfully ping the database and has resumed monitoring. However, the node should be checked to make sure it is performing normally to prevent a possible database or agent failure.
Resume timed out for cluster *cluster_name*	This agent could not resume monitoring after reconfiguring and restarting the local database. See agent log for details.
Internal state mismatch for cluster *cluster_name*	The failover manager cluster's internal state did not match the actual state of the cluster members. This is rare and can be caused by a timing issue of nodes joining the cluster and/or changing their state. The problem should be resolved, but you should check the cluster status as well to verify. Details of the mismatch can be found in the agent log file..
Failover has not occurred	An agent has detected that the master database is no longer available in cluster *cluster_name*, but there are not enough standby nodes available for failover..
Database in wrong state on *node_address*	The standby agent has detected that the local

	database is no longer in recovery. The agent will now become IDLE. Manual intervention is required.
Database in wrong state on *node_address*	The master agent has detected that the local database is in recovery. The agent will now become IDLE. Manual intervention is required.
Database connection failure for cluster *cluster_name*	This node is unable to connect to the database running on: *node_address* Until this is fixed, failover may not work properly because this node will not be able to check if the database is running or not.
License has expired	EDB Postgres Failover Manager has shutdown. Your license subscription to run this product expired on *expiration_date*. Please contact your EnterpriseDB Account Manager to purchase a new license subscription for Failover Manager.
License is invalid	EDB Postgres Failover Manager has shutdown. There is a problem with the Full Use License for EDB Postgres Failover Manager that was provided in the cluster properties file. Please check to be sure it was entered correctly. If this problem persists, please contact your EnterpriseDB Account Manager.

Please note: In addition to sending notices to the administrative email address, all notifications are recorded in the cluster log file (`/var/log/efm-2.1/`*`cluster_name`*`.log`).

8 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, reboots, or otherwise becomes unreachable due to network connectivity issues.

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` (see Section 3.2.1).

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

8.1 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 (see Figure 8.1).

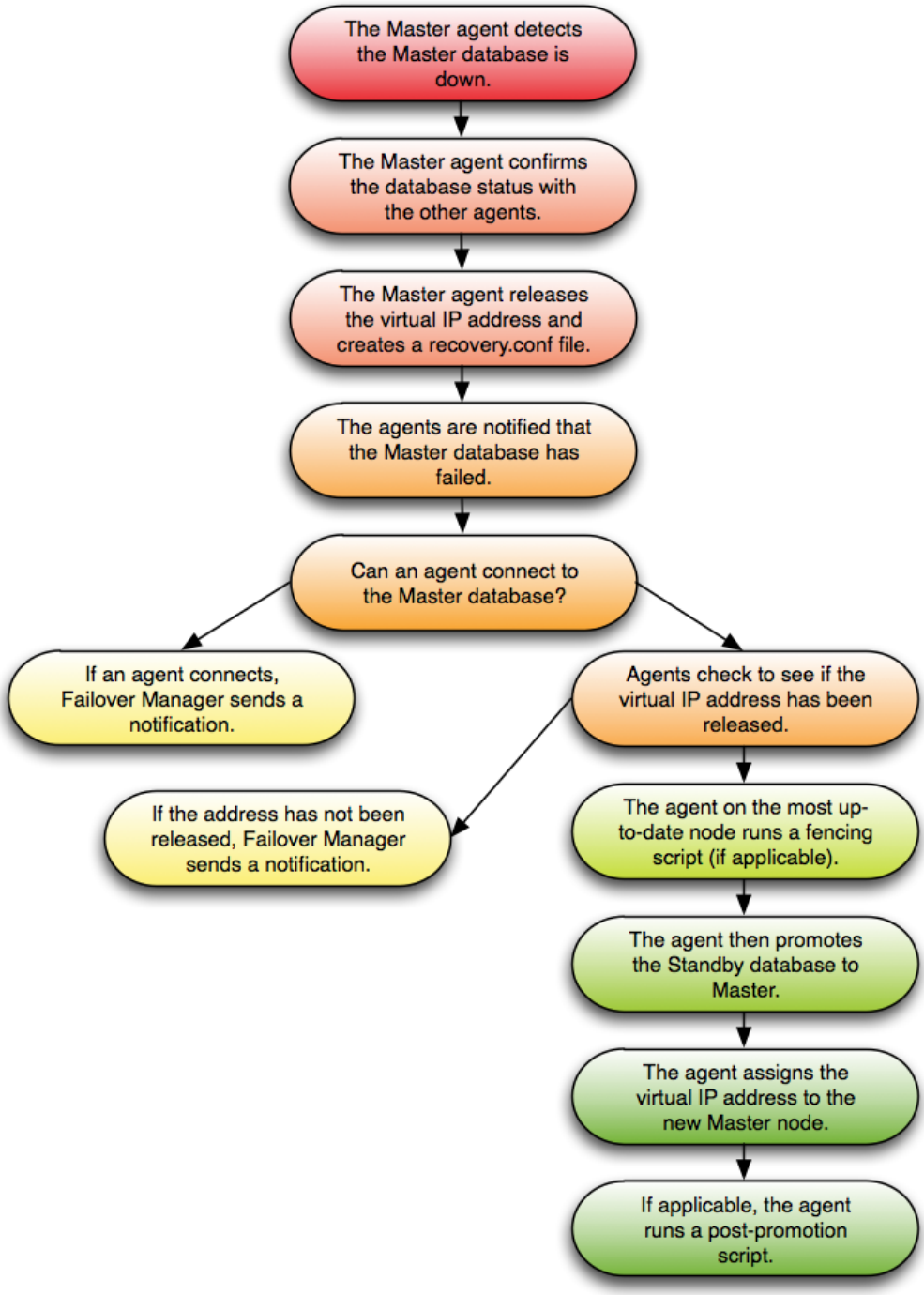


Figure 8.1 - Confirming the Failure of the Master Database.

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.

8.2 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 (see Figure 8.2).

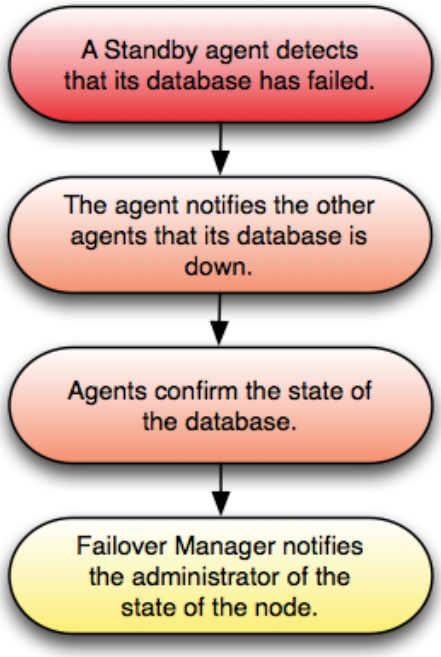


Figure 8.2 - Confirming the failure of a Standby Database.

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

8.3 Master Agent Exits or Node Fails

If the Failover Manager Master agent exits or the node fails, a Standby agent will detect the failure and (if appropriate) initiate a failover (see Figure 8.3).

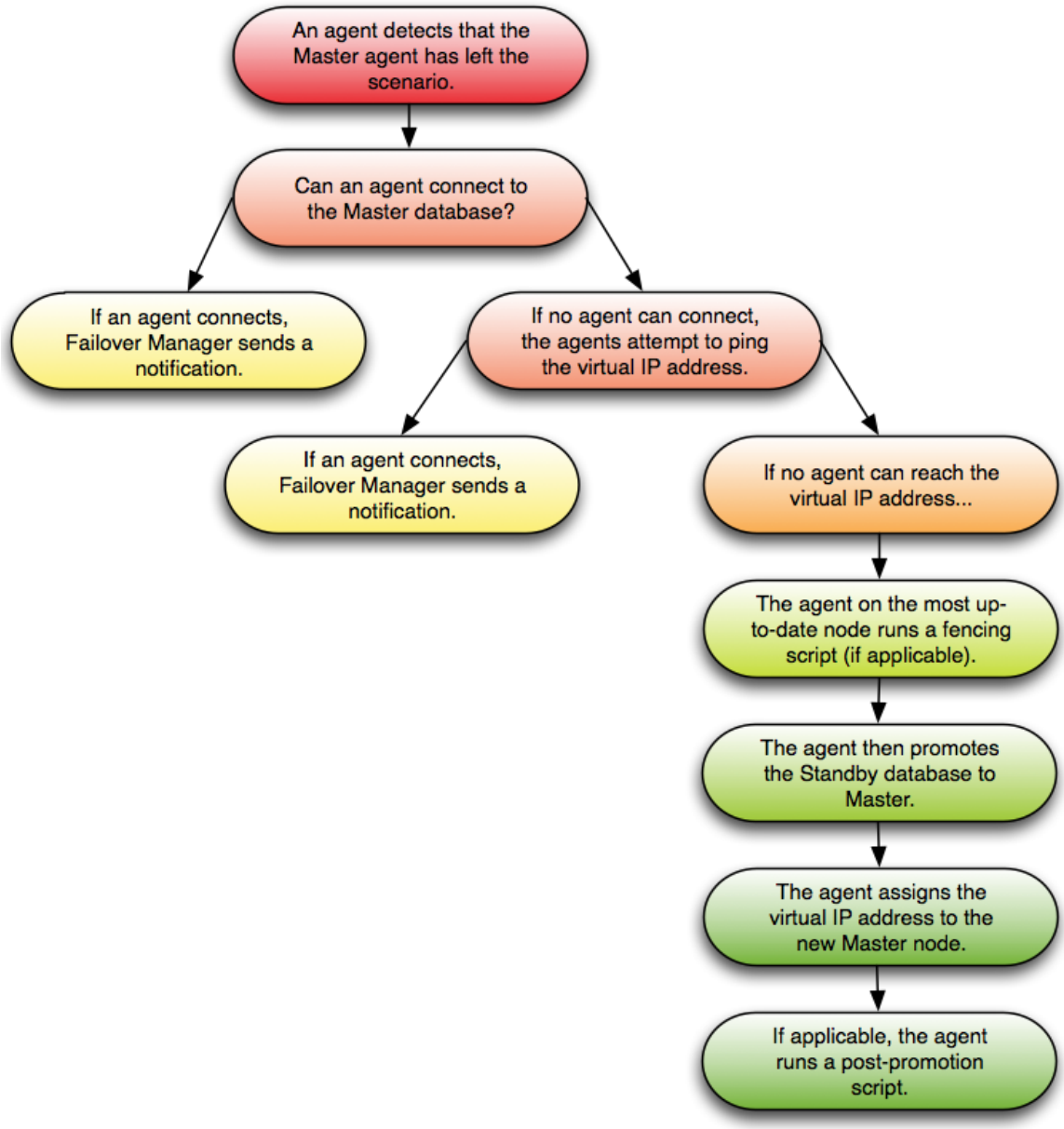


Figure 8.3 - Confirming the failure of the Master Agent.

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.

8.4 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.

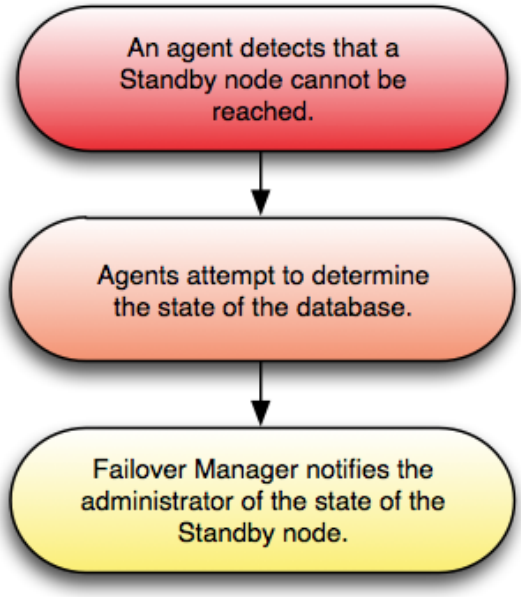


Figure 8.4 - Failure of Standby Agent.

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.

8.5 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.

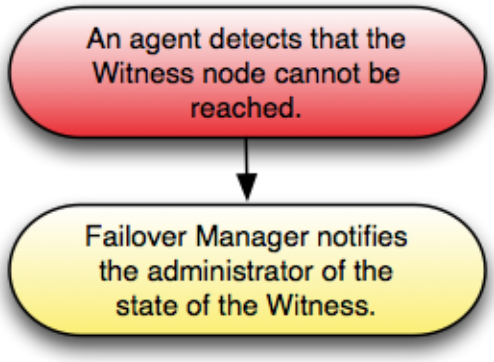


Figure 8.5 - Confirming the Failure of a dedicated Witness.

When an agent detects that the Witness node cannot be reached, Failover Manager notifies the administrator of the state of the Witness (see Figure 8.5).

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.

8.6 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.

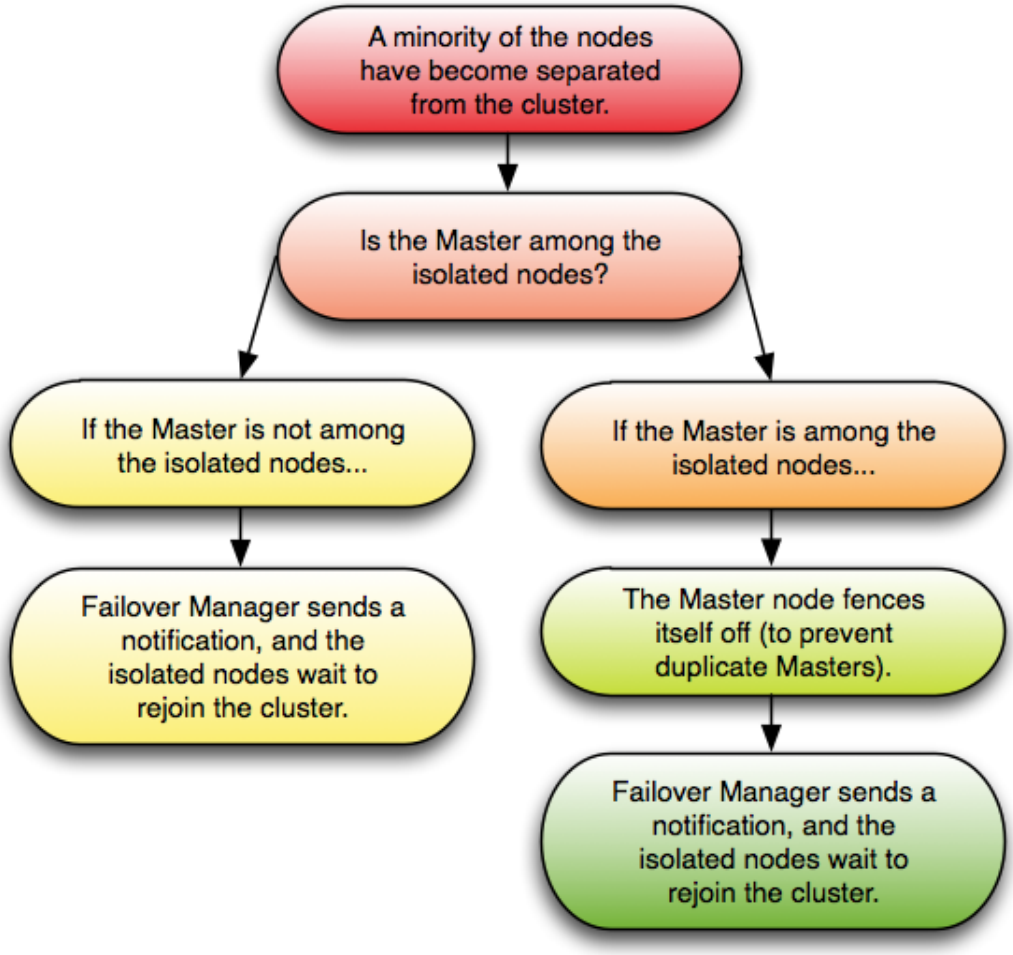


Figure 8.6 – If members of the cluster become isolated.

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.

9 Upgrading an Existing Cluster

Failover Manager 2.1 provides a utility to assist you when upgrading a Failover Manager 2.0 cluster to Failover Manager 2.1. To upgrade an existing cluster, you must:

1. Install Failover Manager 2.1 on each node of the cluster. For detailed information about installing Failover Manager, see Section 3.

   Invoke the `efm upgrade-conf` utility to create the `.properties` and `.nodes` files for Failover Manager 2.1. The Failover Manager installer adds the upgrade utility (`efm upgrade-conf`) to the `/usr/efm-2.1/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 a 2.0 cluster (stored in `/etc/efm-2.0`) and copies the parameter values to a configuration file for use by Failover Manager 2.1. The utility saves the updated copy of the configuration files in the `/etc/efm-2.1` directory for use by Failover Manager 2.1.

2. Modify the `.properties` and `.nodes` files for EFM 2.1, specifying any new preferences. Version 2.1 of Failover Manager adds the following configuration parameters:

   ```
   auto.allow.hosts
   auto.resume.period
   db.service.name
   jvm.options
   minimum.standbys
   node.timeout
   promotable
   recovery.check.period
   script.notification
   script.resumed
   ```

   Use your choice of editor to modify any additional properties in the properties file (located in the `/etc/efm-2.1` directory) before starting the service for that node.

   Please note: If you use an operating system service script (via the `service` or `systemctl` command) when starting and stopping the database on a non-witness node, you must provide the service name in the `db.service.name` parameter.

   The `jgroups.max.tries` and `jgroups.timeout` properties are replaced by the `node.timeout` property.

For detailed information about parameter settings, see Section 3.2.

3. Stop the Failover Manager 2.0 cluster; for example, you can use the command:

```
/usr/efm-2.0/bin/efm stop-cluster efm
```

4. Start the `efm-2.1` service on each node of the cluster. For more information about starting the service, see Section 4.1.1.

The following example demonstrates invoking the upgrade utility to create the .properties and .nodes files for an EFM 2.1 installation:

```
[root@localhost bin]# ./efm upgrade-conf efm
Processing efm.properties file.
Setting new property node.timeout to 40 (sec) based on existing
timeout 5000 (ms) and max tries 8.

Processing efm.nodes file.

Upgrade of files is finished. Please ensure that the new file
permissions match those of the template files before starting
EFM.
The db.service.name property should be set before starting a non-
witness agent.
```

9.1 Un-installing Failover Manager

After upgrading to Failover Manager 2.1, you can use Yum to remove Failover Manager 2.0. Use the following command to remove Failover Manager 2.0 and any unneeded dependencies:

```
yum remove efm20
```

10 Appendix A - Configuring Streaming Replication

The following section will walk you through the process of configuring a simple two-node replication scenario that uses streaming replication to replicate data from a Master node to a Standby node. The replication process for larger scenarios can be complex; for detailed information about configuration options, please see the PostgreSQL core documentation, available at:

http://www.enterprisedb.com/docs/en/9.5/pg/warm-standby.html#STREAMING-REPLICATION

In the example that follows, we will 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:

http://www.enterprisedb.com/docs/en/9.5/pg/client-authentication.html

The steps that follow configure a simple streaming replication scenario with one Master node and one Standby node, each running an installation of EDB Postgres Advanced Server. In the example:

- The Master node resides on `146.148.46.44`
- The Standby node resides on `107.178.217.178`
- The replication user name is `edbrepuser`.

The pathnames and commands referenced in the examples are for Advanced Server hosts that reside on a CentOS 6.5 host – you may have to modify paths and commands for your configuration.

Configuring the Master Node

Connect to the master node of the replication scenario, and modify the `pg_hba.conf` file (located in the `data` directory under your Postgres installation), adding connection information for the replication user (in our example, `edbrepuser`):

```
host replication edbrepuser 107.178.217.178/32 md5
```

The connection information should specify the address of the standby node of the replication scenario, and your preferred authentication method.

Modify the `postgresql.conf` file (located in the `data directory, under your Postgres installation`), adding the following replication parameter and values to the end of the file:

```
wal_level = hot_standby
max_wal_senders = 8
wal_keep_segments = 128
archive_mode = on
archive_command = 'cp %p /tmp/%f'
```

Save the configuration file and restart the server:

```
/etc/init.d/ppas-9.5 restart
```

Use the `sudo su -` command to assume the identity of the `enterprisedb` database superuser:

```
sudo su - enterprisedb
```

Then, start a `psql` session, connecting to the `edb` database:

```
psql -d edb
```

At the `psql` command line, create a user with the `replication` attribute:

```
CREATE ROLE edbrepuser WITH REPLICATION LOGIN PASSWORD
'password';
```

Configuring the Standby Node

Connect to the Standby server, and assume the identity of the database superuser (`enterprisedb`):

```
sudo su - enterprisedb
```

With your choice of editor, create a `.pgpass` file in the home directory of the `enterprisedb` user. The `.pgpass` file holds the password of the replication user in plain-text form; if you are using a `.pgpass` file, you should ensure that only trusted users have access to the `.pgpass` file:

Add an entry that specifies connection information for the replication user:

```
*:5444:*:edbrepuser:password
```

The server will enforce restrictive permissions on the `.pgpass` file; use the following command to set the file permissions:

```
chmod 600 .pgpass
```

Relinquish the identity of the database superuser:

```
exit
```

Then, assume superuser privileges:

```
sudo su -
```

You must stop the database server before replacing the data directory on the Standby node with the data directory of the Master node. Use the command:

```
/etc/init.d/ppas-9.5 stop
```

Then, delete the `data` directory on the Standby node:

```
rm -rf /opt/PostgresPlus/9.5AS/data
```

After deleting the existing `data` directory, use the `pg_basebackup` utility to copy the `data` directory of the Master node to the Standby:

```
./pg_basebackup -R -D /opt/PostgresPlus/9.5AS/data
--host=146.148.46.44 -port=5444
--username=edbrepuser --password
```

The call to `pg_basebackup` specifies the IP address of the Master node and the name of the replication user created on the Master node. For more information about the options available with the pg_basebackup utility, see the PostgreSQL core documentation at:

http://www.enterprisedb.com/docs/en/9.5/pg/app-pgbasebackup.html

When prompted by `pg_basebackup`, provide the password associated with the replication user.

After copying the `data` directory, change ownership of the directory to the database superuser (`enterprisedb`):

```
chown -R enterprisedb /opt/PostgresPlus/9.5AS/data
```

Navigate into the `data` directory:

```
cd /opt/PostgresPlus/9.5AS/data
```

With your choice of editor, create a file named `recovery.conf` (in the `/opt/PostgresPlus/9.xAS/data` directory) that includes:

```
standby_mode = on
primary_conninfo = 'host=146.148.46.44 port=5444 user=edbrepuser
sslmode=prefer sslcompression=1 krbsrvname=postgres'
trigger_file = '/opt/PostgresPlus/AS9.5/data/mytrigger'
restore_command = '/bin/true'
recovery_target_timeline = 'latest'
```

Please note: the `primary_conninfo` parameter specifies connection information for the replication user on the master node of the replication scenario.

Change ownership of the `recovery.conf` file to enterprisedb:

```
chown enterprisedb:enterprisedb recovery.conf
```

Modify the `postgresql.conf` file (located in `data directory`, under the Postgres installation), specifying the following values at the end of the file:

```
wal_level = hot_standby
max_wal_senders = 8
wal_keep_segments = 128
hot_standby = on
```

The data file has been copied from the Master node, and will contain the replication parameters specified previously.

Then, restart the server:

```
/etc/init.d/ppas-9.5 start
```

At this point, the Master node will be replicating data to the Standby node.

Confirming Replication from the Master to Standby

You can confirm that the server is running and replicating by entering the command:

```
ps -ef | grep postgres
```

If replication is running, the Standby server will echo:

```
501 42054 1 0 07:57 pts/1 00:00:00
/opt/PostgresPlus/9.2AS/bin/edb-postgres -D
/opt/PostgresPlus/9.2AS/data
501 42055 42054 0 07:57 ? 00:00:00 postgres: logger process
501 42056 42054 0 07:57 ? 00:00:00 postgres: startup
process recovering 000000010000000000000004
501 42057 42054 0 07:57 ? 00:00:00 postgres: checkpointer
process
501 42058 42054 0 07:57 ? 00:00:00 postgres: writer process
501 42059 42054 0 07:57 ? 00:00:00 postgres: stats
```

```
collector process
501 42060 42054 0 07:57 ? 00:00:00 postgres: wal receiver
process streaming 0/4000150
501 42068 42025 0 07:58 pts/1 00:00:00 grep postgres
```

If you connect to the Standby with the psql client and query the `pg_is_in_recovery()` function, the server will reply:

```
edb=# select pg_is_in_recovery();
pg_is_in_recovery
------------------
t
(1 row)
```

Any entries made to the Master node will be replicated to the Standby node. The Standby node will operate in read-only mode; while you can query the Standby server, you will not be able to add entries directly to the database that resides on the Standby node.

Manually Invoking Failover

To promote the Standby to become the Master node, assume the identity of the cluster owner (enterprisedb):

```
sudo su - enterprisedb
```

Then, invoke pg_ctl:

```
/opt/PostgresPlus/9.5AS/bin/pg_ctl promote -D /
opt/PostgresPlus/9.5AS /data/
```

Then, if you connect to the Standby node with psql, the server will confirm that it is no longer a standby node:

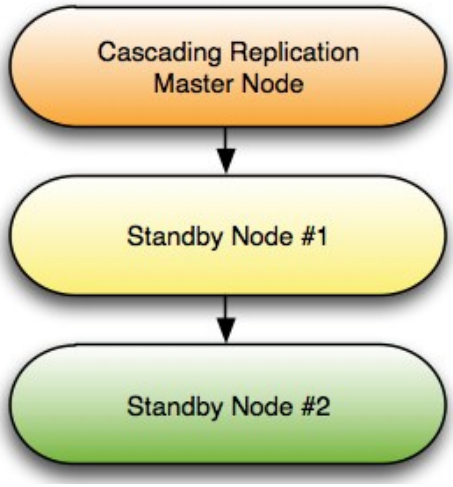
```
edb=# select pg_is_in_recovery();
pg_is_in_recovery
------------------
f
(1 row)
```

For more information about configuring and using streaming replication, please refer to Chapter 25 of the PostgreSQL core documentation, available at:

http://www.postgresql.org/docs/9.6/static/high-availability.html

10.1 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/9.6/static/warm-standby.html

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.

11 Appendix B - 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:

   http://www.enterprisedb.com/docs/en/9.5/pg/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:

   http://www.enterprisedb.com/docs/en/9.5/pg/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, see:

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

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

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

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

6. Modify the `efm.properties` file on each server within the cluster, enabling ssl:

```
jdbc.ssl=true
```

After modifying the `efm.properties` file, restart the Failover Manager agent. For more information about restarting the Failover Manager service, see Section 5.

12 Inquiries

If you have any questions regarding EDB Postgres Failover Manager, please contact EnterpriseDB at:

sales@enterprisedb.com