Failover Manager

Release 4.7

EnterpriseDB Corporation

Jul 19, 2023

CONTENTS

1 Release notes **3**
- 1.1 Version 4.7 . . . 3
- 1.2 Version 4.6 . . . 4
- 1.3 Version 4.5 . . . 4
- 1.4 Version 4.4 . . . 5
- 1.5 Version 4.3 . . . 5
- 1.6 Version 4.2 . . . 5
- 1.7 Version 4.1 . . . 6
- 1.8 Version 4.0 . . . 6
  - 1.8.1 EFM properties changes . . . 7

2 Supported platforms **9**

3 Architecture **11**

4 Choosing a deployment architecture **13**
- 4.1 Failover Manager with virtual IP . . . 14
  - 4.1.1 Using Failover Manager with VIP . . . 14
- 4.2 Failover Manager with client connect failover . . . 16
  - 4.2.1 Using Failover Manager with Client Connection failover . . . 16
- 4.3 Failover Manager with EDB PgBouncer . . . 17
  - 4.3.1 Failover Manager with EDB PgBouncer on premises . . . 18
  - 4.3.2 Failover Manager with EDB PgBouncer in the cloud . . . 18
  - 4.3.3 Using Failover Manager with PgBouncer . . . 20
- 4.4 Failover Manager with EDB Pgpool-II . . . 24
  - 4.4.1 Failover Manager with EDB Pgpool-II on premises . . . 24
  - 4.4.2 Failover Manager with EDB Pgpool-II in the cloud . . . 24
  - 4.4.3 Using Failover Manager with EDB Pgpool-II . . . 27

5 Installing Failover Manager on Linux **33**
- 5.1 Linux x86-64 (amd64) . . . 33
  - 5.1.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 33
  - 5.1.2 SUSE Linux Enterprise (SLES) . . . 33
  - 5.1.3 Debian and derivatives . . . 33
- 5.2 Linux Installing Failover Manager on Linux IBM Power (ppc64le) . . . 34
  - 5.2.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 34
  - 5.2.2 SUSE Linux Enterprise (SLES) . . . 34

6 Upgrading Failover Manager **53**
- 6.1 Uninstalling Failover Manager . . . 54
- 6.2 Performing a maintenance task . . . 55

7 Configuring streaming replication **57**
7.1 Limited support for cascading replication . . . 57

8 Configuring Failover Manager **59**
8.1 The cluster properties file . . . 59
8.1.1 Specifying cluster properties . . . 60
8.1.2 Cluster properties . . . 63
8.2 Encrypting your database password . . . 91
8.2.1 Using the EFMPASS environment variable . . . 92
8.3 The cluster members file . . . 93
8.4 Extending Failover Manager permissions . . . 93
8.4.1 Running Failover Manager without sudo . . . 95
8.5 Using Failover Manager with virtual IP addresses . . . 96
8.5.1 Testing the VIP . . . 96
8.6 Configuring for Eager Failover . . . 98
8.6.1 Enabling Eager Failover . . . 98
8.6.2 Disabling Eager Failover . . . 99
8.6.3 Upgrading Failover Manager in Eager Failover mode . . . 99
8.6.4 Important notes . . . 99

9 Configuring SSL authentication on a Failover Manager cluster **101**

10 Using Failover Manager **103**
10.1 Managing a Failover Manager cluster . . . 103
10.1.1 Starting the Failover Manager cluster . . . 104
10.1.2 Adding nodes to a cluster . . . 104
10.1.3 Changing the priority of a standby . . . 105
10.1.4 Promoting a Failover Manager node . . . 105
10.1.5 Stopping a Failover Manager agent . . . 106
10.1.6 Stopping a Failover Manager cluster . . . 107
10.1.7 Removing a node from a cluster . . . 107
10.2 Running multiple agents on a single node . . . 107
10.2.1 RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x . . . 109

11 Using the efm utility **111**
11.1 efm allow-node . . . 111
11.2 efm disallow-node . . . 111
11.3 efm cluster-status . . . 111
11.4 efm cluster-status-json . . . 111
11.5 efm encrypt . . . 112
11.6 efm promote . . . 113
11.7 efm resume . . . 113
11.8 efm set-priority . . . 113
11.9 efm stop-cluster . . . 113
11.10 efm upgrade-conf . . . 114
11.11 efm node-status-json . . . 114
11.12 efm –help . . . 114

12 Monitoring a Failover Manager cluster **115**
12.1 Reviewing the cluster status report . . . 115
12.1.1 Exit codes . . . 117
12.2 Monitoring streaming replication with Postgres Enterprise Manager . . . 117

13 Controlling the Failover Manager service **121**
13.1 Using the systemctl utility on RHEL/CentOS 7.x and RHEL/Rocky Linux/AlmaLinux 8.x . . . 121

14 Controlling logging **123**
14.1 Enabling syslog log file entries . . . 124

15 Notifications **127**

16 Supported failover and failure scenarios **137**
16.1 Primary database is down . . . 137
16.2 Return the node to the cluster . . . 139
16.3 Return the node to the role of primary . . . 139
16.4 Standby database is down . . . 139
16.5 Primary agent exits or node fails . . . 139
16.6 Standby agent exits or node fails . . . 142
16.7 Dedicated witness agent exits/node fails . . . 143
16.8 Nodes become isolated from the cluster . . . 143

17 Troubleshooting **145**
17.1 Authorization file not found. Is the local agent running? . . . 145
17.2 Not authorized to run this command. User '<os user>' is not a member of the `efm` group. . . . 145
17.3 Notification; Unexpected error message . . . 145
17.4 Confirming the OpenJDK version . . . 146

18 Creating a Failover Manager cluster **147**
18.1 Perform a switchover . . . 149
18.2 Access online help . . . 149

Failover Manager (EFM) is a tool for managing Postgres database clusters, enabling high availability of primary-standby deployment architectures using streaming replication. Failover Manager provides a Postgres primary database node automatic failover to a standby database node in the event of a software or hardware failure. You can use Failover Manager with PostgreSQL or EDB Postgres Advanced Server.

CHAPTER

ONE

RELEASE NOTES

The Failover Manager documentation describes the latest version of Failover Manager 4, including minor releases and patches. These release notes cover what was new in each release. For new functionality introduced in a minor or patch release, there are also indicators in the content about the release that introduced the feature.

Version	Release Date
Version 4.7	2023 Jun 20
Version 4.6	2023 Feb 14
Version 4.5	2022 Aug 30
Version 4.4	2022 Jan 5
Version 4.3	2021 Dec 18
Version 4.2	2021 Apr 19
Version 4.1	2021 Dec 11
Version 4.0	2021 Sep 2

1.1 Version 4.7

Enhancements, bug fixes, and other changes in EFM 4.7 include:

Type	Description
Enhancement	Failover Manager now only keeps one backup of pg_wal on standby when reconfiguring to follow a new primary. [Support tickets: #1277862, #91506]
Bug Fix	Fixed an issue where the Failover Manager startup script didn't work in locales that use commas for decimals; for example, "1,8". [Support ticket #89616]
Bug Fix	Fixed an issue where Failover Manager mistakenly used a standby's WAL receive value for replay value in the 'efm' command in Failover Manager 4.6.

1.2 Version 4.6

Enhancements, bug fixes, and other changes in EFM 4.6 include:

Type	Description
Security Fix	Failover Manager has been upgraded to use PostgreSQL JDBC Driver version 42.4.3 to address security vulnerability CVE-2018-10936. Note: Failover Manager is not impacted by the vulnerability, but no longer contains the impacted version of the library.
Enhancement	Failover Manager now supports creating the agent pidfiles in a custom location. [Support tickets #88686 #89016]
Bug Fix	Failover Manager will not continue to send "primary node missing" notifications after a primary has joined the cluster in rare cases. [Support ticket #87106]
Bug Fix	In cases where a node is overloaded such that the Failover Manager agent is prevented from running, without failover or the overloaded node being stopped, it was possible for the cluster's internal state to become inconsistent, preventing future failovers. This case will now be handled so that the internal state is fixed after such an event. [Support ticket #87975]
Bug Fix	Failover Manager primary agents will no longer fail to release a VIP in rare cases of a simultaneous database failure and node isolation. [Support ticket #79280]
Bug Fix	A misleading notification subject was fixed for the cases of a standby node failure in a cluster.
Bug Fix	Failover Manager now supports more than nine synchronous standbys in a cluster.

1.3 Version 4.5

Enhancements, bug fixes, and other changes in EFM 4.5 include:

Type	Description
Security Fix	Failover Manager has been upgraded to use PostgreSQL JDBC Driver version 42.4.1 to address security vulnerability CVE-2022-31197. Note: Failover Manager is not impacted by the vulnerability, but no longer contains the impacted version of the library.
Enhancement	When a primary node has been isolated from the cluster and rejoins, if no other node has been promoted, Failover Manager will restart the isolated database and resume monitoring as the primary node in the cluster. [Support tickets #937960, #968619, #81770, #81894]
Bug Fix	Failover Manager will now properly choose a standby to promote in rare cases of LSN values being compared incorrectly. [Support ticket #83186]
Bug Fix	Failover Manager now handles extra whitespace in the properties file when running the *efm* command for cluster status.
Bug Fix	Failover Manager will no longer fail to promote in rare cases where the primary agent is running slowly. [Support ticket #79847]
Bug Fix	Failover Manager properly follows the promotable=false value of a standby that is restarted after being made promotable manually through the *efm* command. [Support ticket #83192]
Bug Fix	When using physical replication slots, Failover Manager will now drop and recreate slots on standbys when advancing the LSN values on them if they have an *xmin* value set. [Support ticket #83910]

continues on next page

Table 4 – continued from previous page

Type	Description
Bug Fix	Systemd status will now show "inactive" status instead of "failed" after being used to stop a Failover Manager agent.

1.4 Version 4.4

Enhancements, bug fixes, and other changes in EFM 4.4 include:

Type	Description
Security Fix	Failover Manager has been upgraded to use log4j version 2.17.1 to address security vulnerability CVE-2021-44832.

1.5 Version 4.3

Enhancements, bug fixes, and other changes in EFM 4.3 include:

Type	Description
Security Fix	Failover Manager has been upgraded to use log4j version 2.17.0 to address security vulnerabilities CVE-2021-44228, CVE-2021-45046, and CVE-2021-45105. As CVE-2021-44228 is considered a critical vulnerability with a CVSS score of 10.0 this update should be applied as soon as possible.

1.6 Version 4.2

Enhancements, bug fixes, and other changes in EFM 4.2 include:

Type	Description
Enhancement	A new Eager Failover setup was added to move primary resources from an unmonitored and unhealthy node to a fully operational and healthy node. With this setup, a failover is exercised when the primary EFM agent is stopped. Without Eager Failover, the primary Postgres instance is left available but unmanaged.
Enhancement	The EDB Postgres High Availability and Horizontal Read Scaling Architecture is certified with PgPool-II 4.2 and is supported with non-sudo mode.
Enhancement	Promotion priority for standbys can now be defined per primary. [Support Ticket #1092697]
Enhancement	The stop-cluster command can now be disabled in EFM configuration.
Enhancement	In the case that the primary agent leaves the cluster but the primary database is still running, Failover Manager sends a warning notification and not failover. Because this means there is no failover protection, an enhancement to version 4.2 is to repeat the warning notification until the primary agent rejoins the cluster.
Bug Fix	Standbys configured with archiving are now properly reconfigured to follow the new primary. [Support Ticket #1127627, #1182654]

continues on next page

Table 7 – continued from previous page

Type	Description
Bug Fix	A new option can disable the LoadBalancer detach script when the primary EFM agent fails but Postgres is still running. [Support Ticket #1096007]
Bug Fix	Added validation in the agent startup to check if the parent directory of pg_wal/pg_xlog is writable. The parent directory needs to be writable so EFM can create a backup of it and later restore it. If the new validation fails, an error is logged and the agent refuses to start.

See *The cluster properties file* for more information on the new properties.

1.7 Version 4.1

Enhancements, bug fixes, and other changes in EFM 4.1 include:

Type	Description
Enhancement	Support added for HA architecture with EFM and Pgpool deployment with Azure Network Load Balancer.
Enhancement	Standby agents attempt to resume health monitoring when local database connectivity is recovered.
Enhancement	New option automatically increases *num_sync* when synchronous standbys are added to a cluster or they rejoin a cluster.
Bug Fix	Can increase *num_sync* when synchronous standys are added. [Support Ticket #1092053]
Bug Fix	Can't start agent with certain properties not present. [Support Ticket #1087362]
Bug Fix	Upgraded JDBC driver due to security vulnerabilities. [Support Ticket #1052383]
Bug Fix	'arping' now called after acquiring VIP in non-sudo mode. [Support Ticket #1039625]

See *The cluster properties file* for more information on the new properties.

1.8 Version 4.0

Enhancements, bug fixes, and other changes in EFM 4.0 include:

Type	Description
Enhancement	Properties were renamed. See the table that follows for details. If you use the upgrade utility to upgrade your Failover Manager installation, they are automatically updated.
Enhancement	The property value for the encrypted database password needs to be updated to meet the new hashing algorithm. The upgrade tool will make the required changes. When creating the properties file manually, you can use the standard procedure for encrypting the password instead.
Enhancement	Encryption for database password was improved. Encryption was also enabled for communication between the agents.
Enhancement	Standby servers are no longer stopped while selecting the new primary. This enhancement significantly speeds up the promotion process.

continues on next page

Table 9 – continued from previous page

Type	Description
Enhancement	To be consistent with community naming guidelines, the term "master" was replaced with "primary" in the Failover Manager product and documentation. The upgradeconf tool handles the task of renaming the impacted properties post upgrade. The load balancer scripts such as *script.load.balancer.attach*, *script.load.balancer.detach* now accept the character *p* instead of *m* as an argument.
Enhancement	Support was added to delay the restart of standbys after a promotion. You can increase the availability by staggering the restart of standbys.
Bug Fix	Documentation was updated to aid users who use tmpfs mount points for */var/run* and */var/lock* [Support Ticket #977437]
Bug Fix	An IDLE agent will attempt to resume monitoring when other agents can reach its database. [Support Ticket #1001533]

1.8.1 EFM properties changes

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

If you're using a load balancer script, the '%t' placeholder is replaced with `p` for primary rather than `m` .

CHAPTER

TWO

SUPPORTED PLATFORMS

For information about the platforms and versions supported by Failover Manager, see Platform Compatibility .

Note: A mixed mode for CPU architecture is supported where the primary and standby nodes are on Linux on IBM Power and witness node is on x86-64 Linux.

CHAPTER

THREE

ARCHITECTURE

Failover Manager is a high-availability tool that monitors the health of a Postgres streaming replication cluster and verifies failures quickly. When a database failure occurs, Failover Manager can automatically promote a streaming replication standby node into a writable primary node. This capability ensures continued performance and protects against data loss with minimal service interruption.

A Failover Manager cluster is made up of Failover Manager processes that reside on the following hosts on a network:

- A primary node is the primary database server that is servicing database clients.
- One or more standby nodes are streaming replication servers associated with the primary node.
- The witness node confirms assertions of either the primary or a standby in a failover scenario. If, during a failure situation, the primary is in a partition with half or more of the nodes, it stays primary. As such, Failover Manager supports running in a cluster with an even number of agents.

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

- If the agent can't reach the database, it starts in idle mode.
- If it finds that the database is in recovery, the agent assumes the role of standby.
- If the database isn't in recovery, the agent assumes the role of primary.

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. Data loss is possible if the standby node is not in sync with the primary node.

JGroups provides technology that allows Failover Manager to create

clusters whose member nodes can communicate with each other and detect node failures.

The figure illustrates a Failover Manager cluster that uses a virtual IP address. You can use a load balancer in place of a *Using Failover Manager with virtual IP addresses* if you provide your own *script* to reconfigure the load balancer whenever databases are added or removed. You can also choose to enable native EFM-Pgpool integration for high availability. See *Choosing a deployment architecture* for more information.

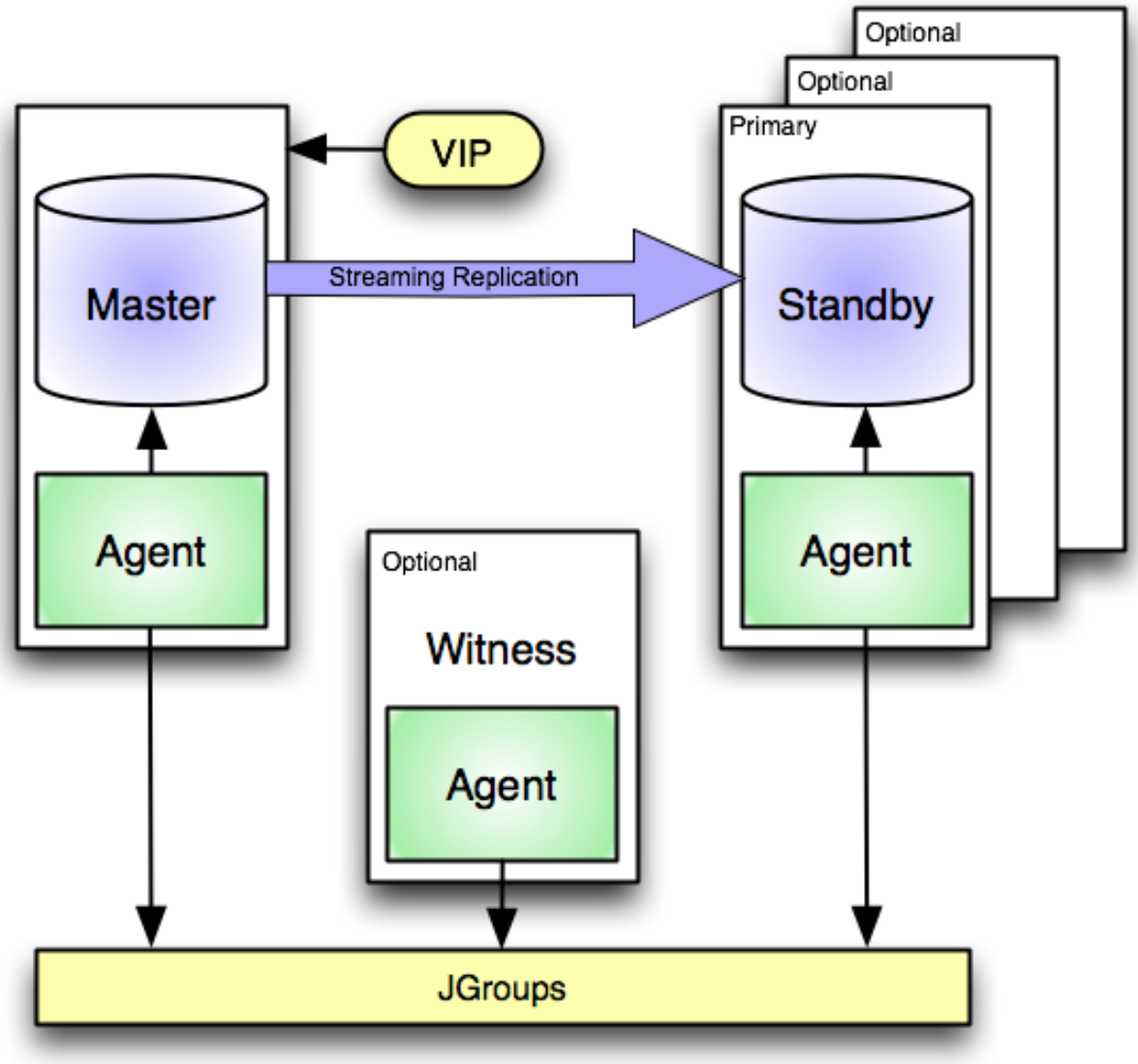

Fig. 1: A Failover Manager scenario employing a virtual IP address

CHAPTER

FOUR

CHOOSING A DEPLOYMENT ARCHITECTURE

Failover Manager provides various high availability options for EDB Postgres Advanced Server using the Postgres connection poolers and connection libraries. These options have implications for a high-availability architecture.

To ensure the high availability of your database, you can combine core features of Failover Manager with the Postgres connection libraries (client connect failover) and connection poolers.

With the capabilities of Failover Manager, EDB has designed four basic architectures to run a high-availability environment:

1. *Failover Manager using VIP (virtual IP)* : Failover Manager has a key capability to manage VIP

   addresses out of the box. VIP addresses allow applications to connect to a single IP address that is being routed to the primary database server. This architecture is the most basic solution to run when VIP addresses are available in your environment.

2. *Failover Manager using client connect failover* : PostgreSQL client libraries like libpq and jdbc allow

   for client connection failover. With client connection failover, the connection string contains multiple servers (host=srv1,srv2) and the client library loops over the available hosts to find a connection that is available and capable of read-write operations. This capability allows clients to follow the master during a switchover. This solution doesn't rely on virtual IP addresses. You can use it in every environment where such client configurations can be set.

3. *Using Failover Manager with PgBouncer* : PgBouncer adds capabilities such as connection

   pooling and the option to halt traffic. You can also use it as a proxy between the client and the Postgres database server. By leveraging the integration options in Failover Manager to run reconfiguration of PgBouncer during a failover, you can use PgBouncer to route the traffic to the correct primary database server.

4. *Failover Manager with EDB Pgpool-II* : EDB Pgpool-II is another tool used as a proxy between

   the client and the Postgres database server. EDB Pgpool-II adds capabilities such as running in cluster mode with a Watchdog, managing VIPs, and read-only scalability. Failover Manager has native capabilities to integrate with EDB Pgpool-II to redirect traffic to another primary during Database failover operations.

These features are supported in each of the architectures:

Clientconnectfailover

Features	Failover Manager with VIP	Failover Manager with client connect failover	Failover Manager with EDB PgBouncer	Failover Manager with EDB Pgpool-II
Connection pooling			Yes	Yes

continues on next page

Table 1 – continued from previous page

Features	Failover Manager with VIP	Failover Manager with client connect failover	Failover Manager with EDB PgBouncer	Failover Manager with EDB Pgpool-II
Runs on cloud (no VIP)		Yes	Yes	Yes
Halt traffic option			Yes	
Read-only scalability		Yes (using multiple connection factories)		Yes
Clustered proxy				Yes
Proxy integration			ssh	PCP
Minimum servers required	3	3	5	6
Complexity	Low	Low	Medium	High
Network hops	1	1		
Failover duration	Low	Medium	Low	Low

4.1 Failover Manager with virtual IP

Failover Manager provides support for clusters that use a virtual IP (VIP).

4.1.1 Using Failover Manager with VIP

Installing

Install and configure the Advanced Server database and Failover Manager on three servers as following:

Systems |Components ——————————————————|——————————————————————————————
PG Primary, PG Standby1, and PG Standby2 | Primary / standby nodes running Advanced Server 13 and Failover Manager 4.2

Specifying VIP

In the cluster properties file, provide the hostname or IP address in the `virtual.ip` property. Specify the corresponding prefix in the `virtual.ip.prefix` property. Use the `virtual.ip.interface` property to provide the network interface used by the VIP. By default, the `virtual.ip` and `virtual.ip.prefix` values are the same across all the agents.

The specified virtual IP address is assigned only to the primary node of the cluster. If you specify `virtual.ip.single=true` , the same VIP address is used on the new primary during a failover. Specify a value of `false` to provide a unique IP address for each node of the cluster.

For information about using a virtual IP address, see *Using Failover Manager with virtual IP addresses* .

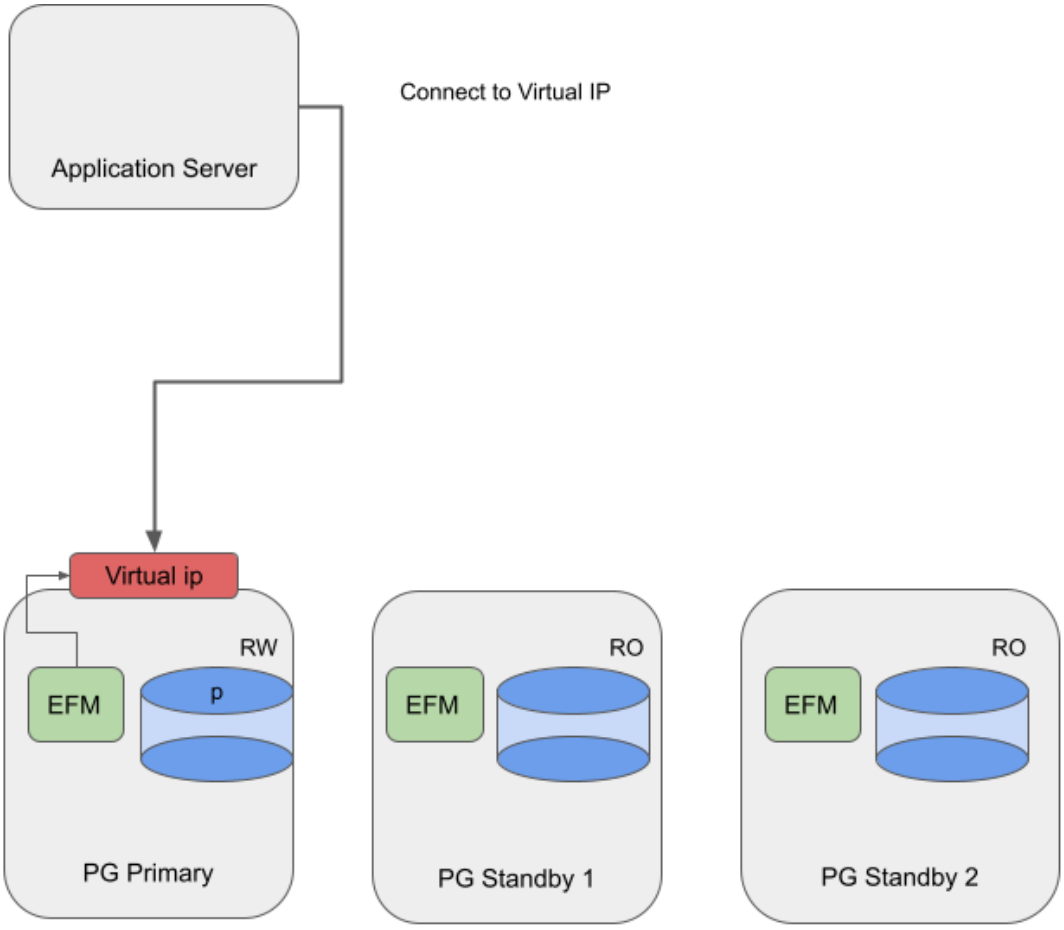

Fig. 1: Failover Manager with VIP

4.2 Failover Manager with client connect failover

Most of the PostgreSQL connection libraries support client connection failover. These libraries support connection strings with more than one database server. On first connection attempt or when the connection is lost (which also occurs during a failover), the client driver connects to the supplied hosts one by one until it finds a read-write connection. The time for reconnecting to the new master depends on the connection timeouts as configured in the driver and TCP layer.

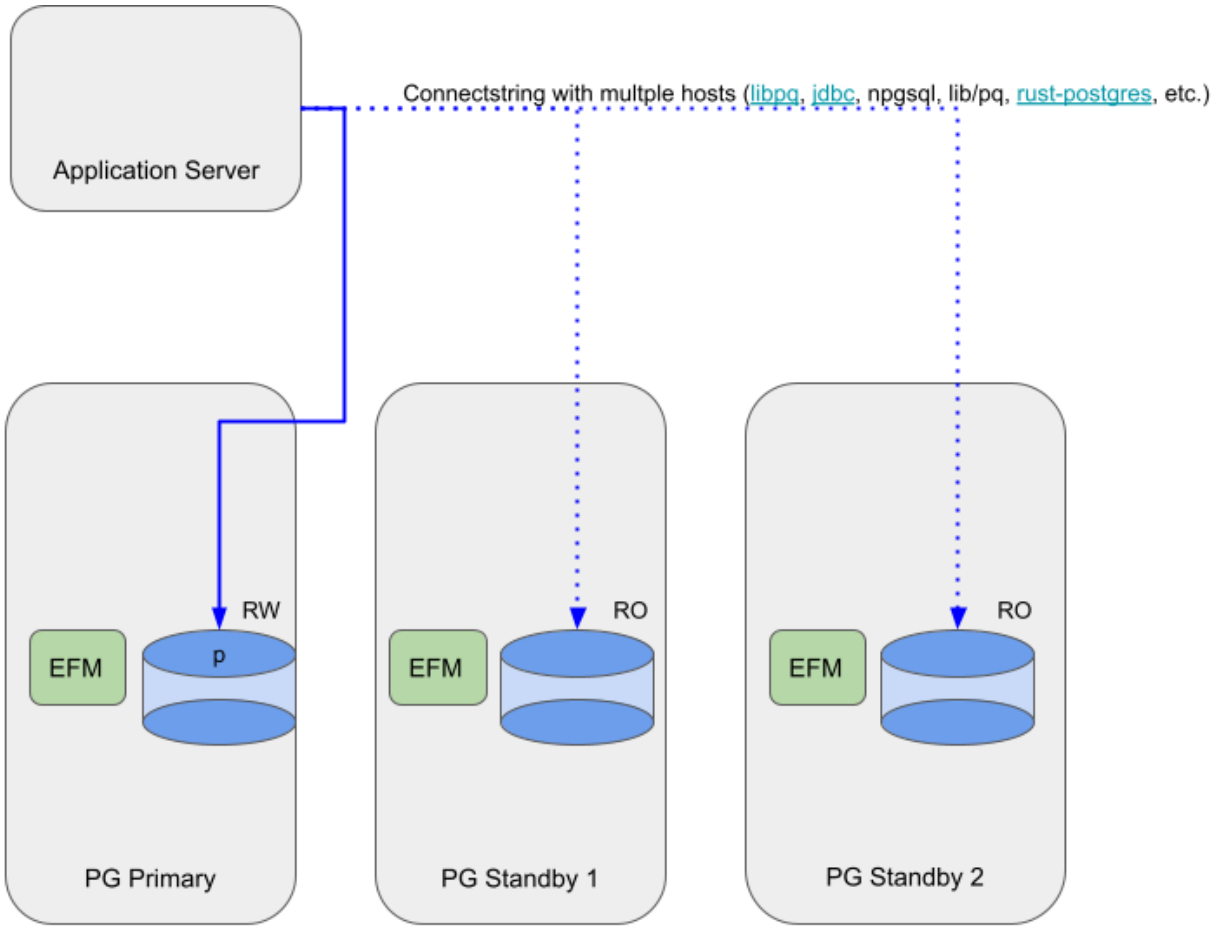

Fig. 2: Failover Manager traffic routing diagram for client connect failover

Figure 2: Failover Manager's traffic routing using client connect failover

4.2.1 Using Failover Manager with Client Connection failover

Installing

Install and configure Advanced Server and Failover Manager on three servers as follows:

Systems	Components
PG Primary, PG Standby1, and PG Standby2	Primary or standby nodes running Advanced Server 13 and Failover Manager 4.2

You don't need to configure the virtual IP configuration in `efm.properties` (`virtual.ip` , `virtual.ip.interface` , `virtual.ip.prefix` , and `virtual.ip.single`).

Configuring Client Connection failover

Driver	Client Connection failover support	Version supported	Configuration
JDBC	Yes	All supported versions	Supply multiple hosts in the connection string, and set the *targetServerType* attribute as primary. Example: *jdbc:postgresql://host1:5444,host2:5444/accounting?targetServerType=pr* More information: https://jdbc.postgresql.org/documentation/head/connect.html#connection-failover
libpq	Yes	10 and above	Supply multiple hosts in the connection string, and set the *target_session_attrs* attribute as read-write. Example: *postgresql://host1:5444,host2:5444/edb?target_session_attrs=read-write*. More information: https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING
.NET	Yes	6 and above	Supply multiple hosts in the connection string, and set the Target Session Attributes attribute as primary. Example: *Host=host1,host2;Username=test;Password=test;Target Session Attributes=primary*. More information: https://www.npgsql.org/doc/failover-and-load-balancing.html
OCL	Yes	10 and above	OCL is based on *libpq*, hence check *libpq* for details.
ODBC	No		

4.3 Failover Manager with EDB PgBouncer

You can use Failover Manager and EDB PgBouncer to provide high availability in an on-premises setup as well as in a cloud setup. EDB PgBouncer is a popular connection pooler, but it is not enough to achieve PostgreSQL high availability by itself as it doesn't have multi-host configuration, failover, or detection.

4.3.1 Failover Manager with EDB PgBouncer on premises

For an on-premises setup, use the connection libraries to provide high availability by using a connection string with multiple hosts.

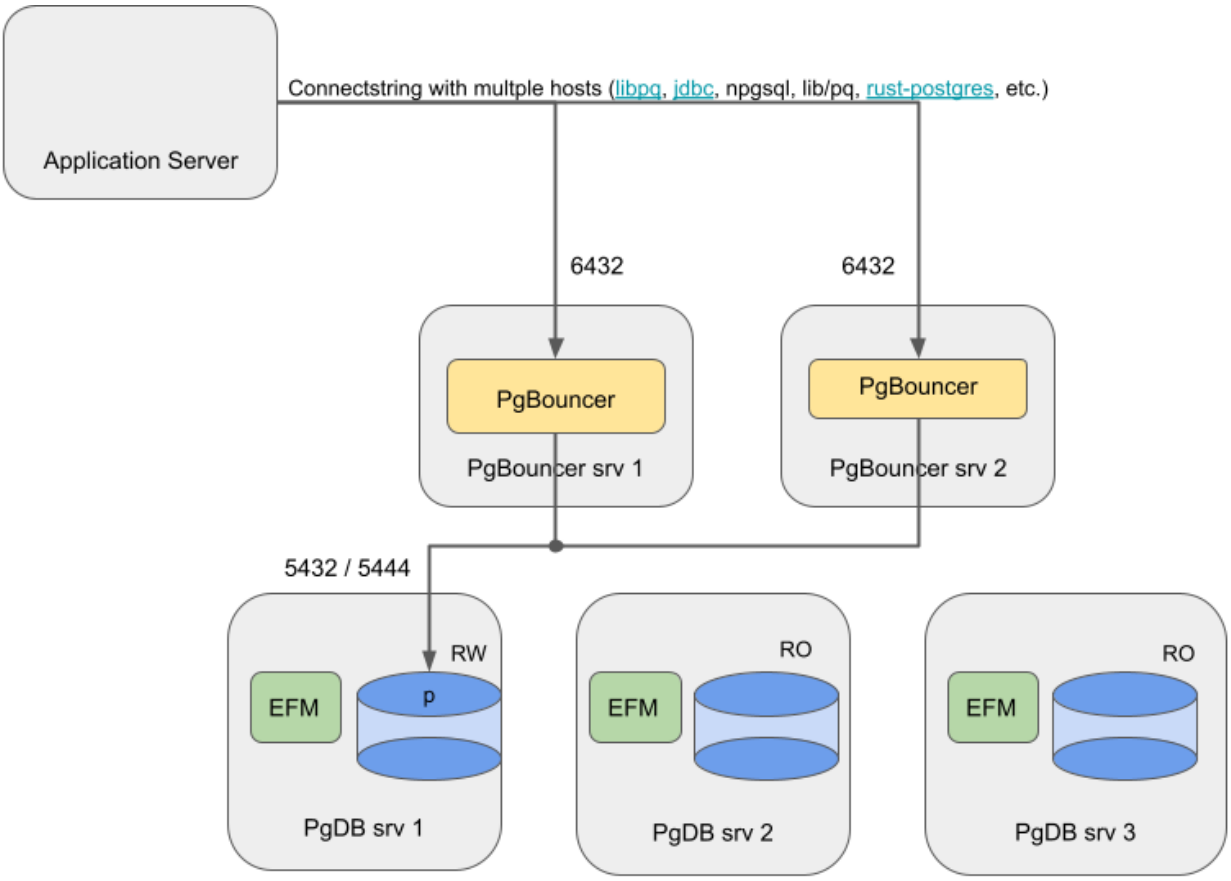

Fig. 3: Failover Manager using pgBouncer on-premises architecture diagram

Figure 3: Failover Manager's traffic routing using EDB PgBouncer on-premises

4.3.2 Failover Manager with EDB PgBouncer in the cloud

For a cloud setup, use a network load balancer (NLB) to balance the traffic on both instances of EDB PgBouncer.

Figure 4: Failover Manager's traffic routing using EDB PgBouncer in cloud

EDB does not support this architecture with EDB PgBouncer and Failover Manager/PostgreSQL running on the same machines:

- A restriction with cloud network load balancers *Configuring NLB in Azure*

doesn't route traffic properly when source and destination reside on the same machines.

- In a mixed architecture, traffic between EDB PgBouncer and Postgres can become unbalanced (sometimes local, sometimes networked).
- EDB PgBouncer and PostgreSQL compete for resources.

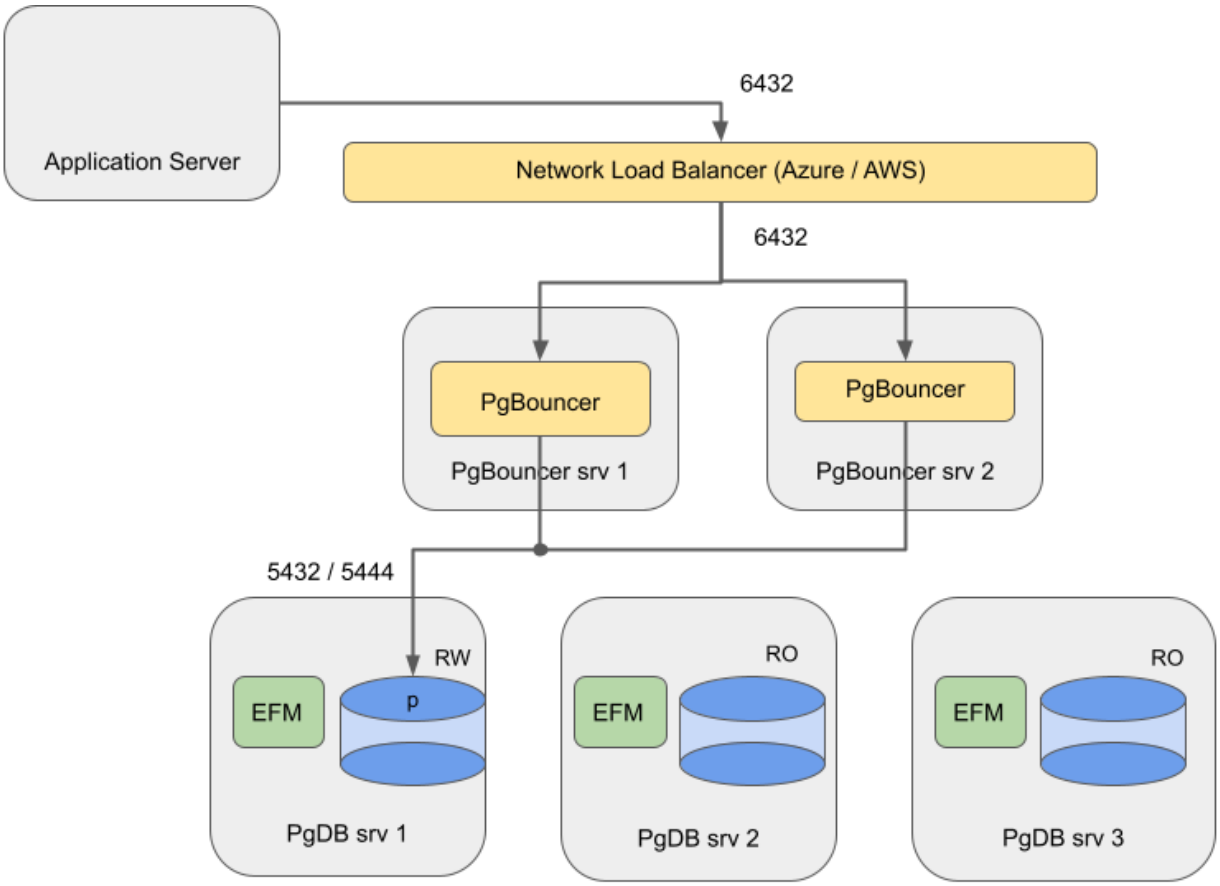

Fig. 4: Failover Manager with PgBouncer cloud architecture diagram

- A master failure impacts both routing (EDB PgBouncer) and database when these two components are combined on the same machines.

4.3.3 Using Failover Manager with PgBouncer

Installing

Install and configure Advanced Server database, Failover Manager, and EDB PgBouncer on AWS virtual machines as follows:

Systems	Components
PgDB srv 1, 2, 3	Primary / standby node running Advanced Server 13 and Failover Manager 4.2
PgBouncer srv 1, 2	PgBouncer node running EDB PgBouncer 1.15. Register these two nodes as targets in the target group. Two is the minimum and is sufficient for most cases.

Configuring Failover Manager

Use the instructions provided in the Failover Manager documentation

to configure Failover Manager. Perform the following steps in addition to those instructions:

1. Create an integration script that connects to every (remote) EDB PgBouncer host and runs the redirect script. Locate the script at `/usr/edb/efm-4.2/bin/efm_pgbouncer_functions` . Make sure the user efm can execute the script, which has the following contents:

```
#!/bin/bash -x
set -e
IFS=,  read -r -a PGB_HOSTS <<< "$4"
FAILED_PGB_HOST=
for PGB_HOST in "${PGB_HOSTS[@]}"; do
    echo "redirecting to $2 on enterprisedb@${PGB_HOST}"
    if [ "$3" == "p" ]; then
       ssh "enterprisedb@${PGB_HOST}" /usr/edb/pgbouncer1.15/bin/redirect.sh "$2" ||␣
↪FAILED_PGB_HOST="$FAILED_PGB_HOST $PGB_HOST" < /dev/null
    fi
done

# return exit code to inform EFM agent about failure. The agent would send a failure
# notification accordingly for manual intervention
if [ ! -z "$FAILED_PGB_HOST" ]; then
   echo "Failed to redirect to $2 on $FAILED_PGB_HOST"
   exit 1
fi
```

2. On each database node, set `script.load.balancer.attach` to the custom script in the `efm` properties file:

edb_notranlate_1 <`pgbs1`> is the hostname or IP address for PgBouncer server 1 and <`pgbs2`> is the hostname or IP address for PgBouncer server 2.

Configuring PostgreSQL

During normal operation, traffic is balanced across both PgBouncer instances, and both open connections to PostgreSQL. Therefore, make sure that in PostgreSQL the `max_connections` parameter is compensated to accept enough connections from both instances.

Configuring EDB PgBouncer

You can use the instructions provided in the EDB PgBouncer documentation

to configure EDB PgBouncer. Perform the following steps in addition to those instructions:

1. Append the following line to the `edb-pgbouncer-1.15.ini` file:

```
%include /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
```

2. In the `edb-pgbouncer-1.15.ini` file, set the value of `listen_addr` to *:

```
listen_addr = *
```

3. Leave the [databases] section empty in the `edb-pgbouncer-1.15.ini` file, and configure this section in a separate file `/etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini` . Ensure that this extra config file is readable and writable by enterprisedb.

The following is an example of the bash commands to create the file:

```
echo "[databases]" > /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
echo "edb= host=srv1" >> /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
chown enterprisedb: /etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini
```

4. Create a script `/usr/edb/pgbouncer1.15/bin/redirect.sh` to use to reconfigure the databases chapter and reload pgbouncer. Make sure the script is owned by root and that user/group/other (0755) has read and execute access. The script has the following content:

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

#Some defaults
PGBOUNCER_DATABASE_INI=/etc/edb/pgbouncer1.15/edb-pgbouncer-databases.ini

PGMSTR=${1:-localhost}

# enterprisedb user does not have permissions to write in folder directly, so `sed -i`␣
↪will not work
TMPFILE=$(mktemp)
sed "s/host=[A-Za-z0-9.]*/host=${PGMSTR}/" "${PGBOUNCER_DATABASE_INI}" > "${TMPFILE}"
if ! diff -q "${PGBOUNCER_DATABASE_INI}" "${TMPFILE}" >/dev/null; then
    cat "${TMPFILE}" > "${PGBOUNCER_DATABASE_INI}"
    pkill -SIGHUP pgbouncer
fi
```

Configuring passwordless ssh

For the EDB PgBouncer integration, passwordless `ssh` access is required. There are multiple ways to configure `ssh` . Follow your organization's recommended process to configure the passwordless `ssh` . For a quick start, you can also follow this example for configuring passwordless `ssh` . The user efm user must be able to ssh as the user running PgBouncer; for example, enterprisedb.

Configure on EDB PgBouncer hosts

1. On every EDB PgBouncer host, temporarily set a password for the enterprisedb user. As root, run `passwd enterprisedb` and enter the temporary password twice.
2. Make sure that passwordless `ssh` is enabled. You can check with the following command:

```
        grep ^PasswordAuthentication /etc/ssh/sshd_config
```  Make sure it is set to `yes` . If needed, change it and restart `ssh` .

####  Configure on Failover Manager/PostgreSQL hosts

On every Failover Manager/postgres host, as the efm user:

1.  Run the following command:

```shell
        ssh-keygen -P "" -f ~/.ssh/id_rsa
```

2. For every EDB PgBouncer host, copy the `ssh` key with the following command:

```
        ssh-copy-id enterprisedb@<pgbouncerhost>
```  The default home directory for the `enterprisedb`  user is  `/var/lib/edb` . If this directory is not already present, create it manually.  As a sudo user, run the following commands on each EDB PgBouncer host:

```shell
        mkdir -p /var/lib/edb
        chown -R enterprisedb:enterprisedb /var/lib/edb
```

Resetting temporary passwords on EDB PgBouncer hosts

You can reset the temporary password for the enterprisedb user on every EDB PgBouncer host by running the following command as root:

`shell passwd -d enterprisedb` ### Configuring the network load balancer

For the Failover Manager EDB PgBouncer integration using a network load balancer in AWS or Azure, you need to perform additional steps.

Add the following rules to the security groups to be used by the EDB PgBouncer and database instances.

- Rules for the security group to be used by the EDB PgBouncer instances (SG PgBouncer).

Type	Protocol	Port range	Source	Description
Custom TCP	TCP	6432	Entire Subnet	PgBouncer
Custom TCP	TCP	22	Entire Subnet	ssh

In addition to these rules, add the rules for SSH and Ping as per your requirement.

- Rules for the security group used by the database instances (SG DB):

Type	Protocol	Port range	Source	Description
Custom TCP	TCP	7800	Entire Subnet	Failover Manager
Custom TCP	TCP	5444	Entire Subnet	Postgres
Custom TCP	TCP	22	Entire Subnet	ssh

These rules ensure that the ports required to run the database, Failover Manager, and EDB PgBouncer are open for communication between the nodes and the load balancer for traffic routing and health monitoring.

In addition to these rules, add the rules for SSH and Ping as per your requirement.

Configuring NLB in Azure

If you are using AWS, see *Configuring NLB in AWS* .

After configuring the rules described in Creating rules for security groups , follow the Azure documentation to:

- Create a backend pool consisting of the two virtual machines running the EDB PgBouncer instances. Use the private IPs of the virtual machines to create the backend pool.
- Add a health probe to check if the EDB PgBouncer instance is available on the virtual machines. Select `TCP` as the protocol and `6432` as the port.
- Add a load balancing rule for port `6432` . This rule ensures that the network traffic coming toward that port is distributed evenly among all the virtual machines present in the backend pool. Select `Public` load balancer or `Internal` load balancer as the type.

After completing these configurations, you can connect to the database on the IP address of the network load balancer using port 6432. If a failure occurs on the primary database server, Failover Manager promotes a new primary and then reconfigures EDB PgBouncer to redistribute traffic. If any of the EDB PgBouncer processes is not available to accept traffic, the network load balancer redistributes all the traffic to the remaining EDB PgBouncer processes. Make sure that the `max_client_conn` parameter is tuned to compensate for the higher number of connections in case of failover.

Configuring NLB in AWS

The following sample configuration assumes:

- All the EC2 instances and the load balancer are deployed in the same subnet. If required, you can add the database nodes to another subnet, but that requires a more complex configuration and might have a performance impact.
- There's a security group for PgBouncer and a security group for the database instances.

After configuring the rules described in Creating rules for security groups , follow the AWS documentation to

- Create a target group with the following details:

Name	Type	Protocol	Port	VPC
pgbouncer	Instances	TCP	6432	Select the VPC to which the instances are connected.

Leave the rest of the settings (**HealthcheckTCP** and **Advancedhealthcheck** settings) as default.

Register the created target groups with the instances that are running EDB PgBouncer.

- Create a load balancer with the following details:

Type	VPC	Listener
Public or *Internal*. EDB recommends using an internal load balancer.	Choose a VPC and map it to the desired zones.	Create a listener with *TCP* as *6432*, and forward it to the target group pgbouncer.

After completing the configurations, you can connect to the database on the IP address of the network load balancer on port 6432. If a failure occurs on the primary database server, Failover Manager promotes a new primary and then reconfigures EDB PgBouncer to redistribute traffic. If any of the EDB PgBouncer processes is not available to accept traffic, the network load balancer redistributes all the traffic to the remaining EDB PgBouncer processes. Make sure that the `max_client_conn` parameter is tuned to compensate for the higher number of connections in case of failover.

4.4 Failover Manager with EDB Pgpool-II

Pgpool-II is a popular connection pooler that can provide many capabilities, like read-only traffic load balancing, connection pooling, and running as a clustered proxy. With Failover Manager to manage availability and reconfiguring EDB Pgpool-II to proxy to the correct primary, the setup can deliver high availability in an on-premises setup as well as in a cloud setup.

4.4.1 Failover Manager with EDB Pgpool-II on premises

For an on-premises setup, you can use a VIP to route the traffic to an available EDB Pgpool-II instance. In this setup, the automatic failover of EDB Pgpool-II is disabled, and Failover Manager is configured to manage EDB Pgpool-II.

Figure 5: Failover Manager's traffic routing using EDB Pgpool-II on-premises

4.4.2 Failover Manager with EDB Pgpool-II in the cloud

For environments with network load balancers (e.g., cloud environments), you can use a network load balancer (NLB) to balance the traffic over all available EDB Pgpool-II instances without requiring a VIP.

Figure 6: Failover Manager's traffic routing using EDB Pgpool-II in cloud

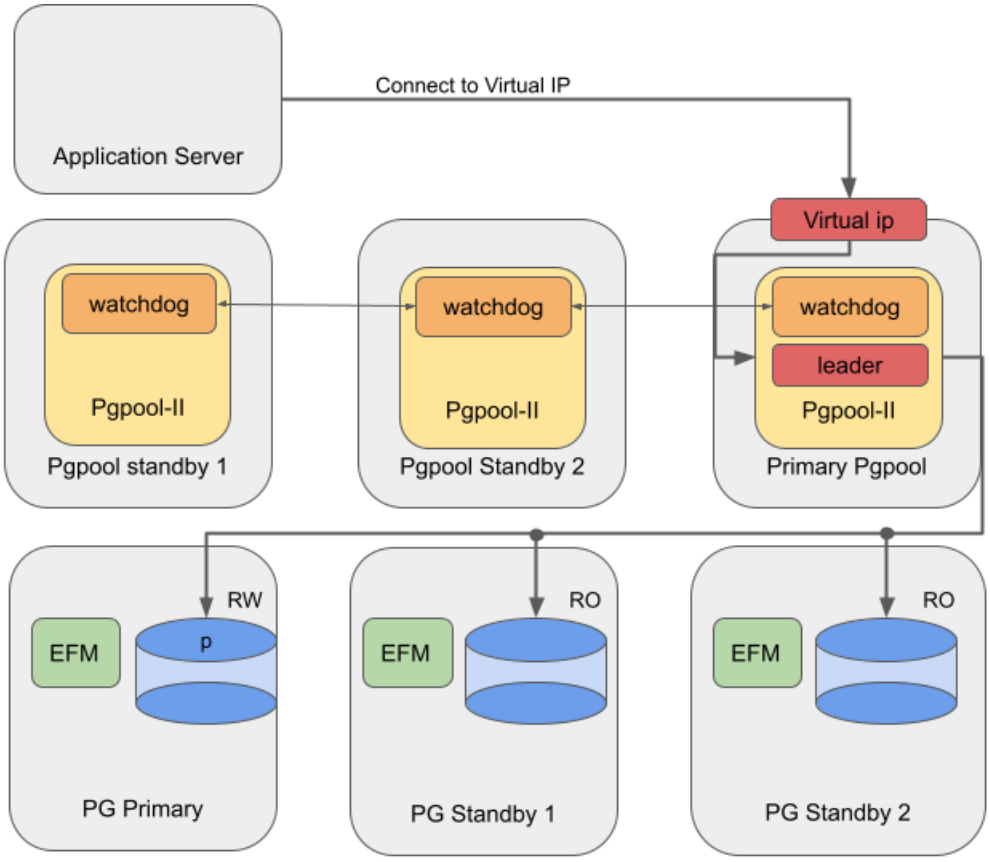

Fig. 5: Failover Manager with Pgpool on-premises

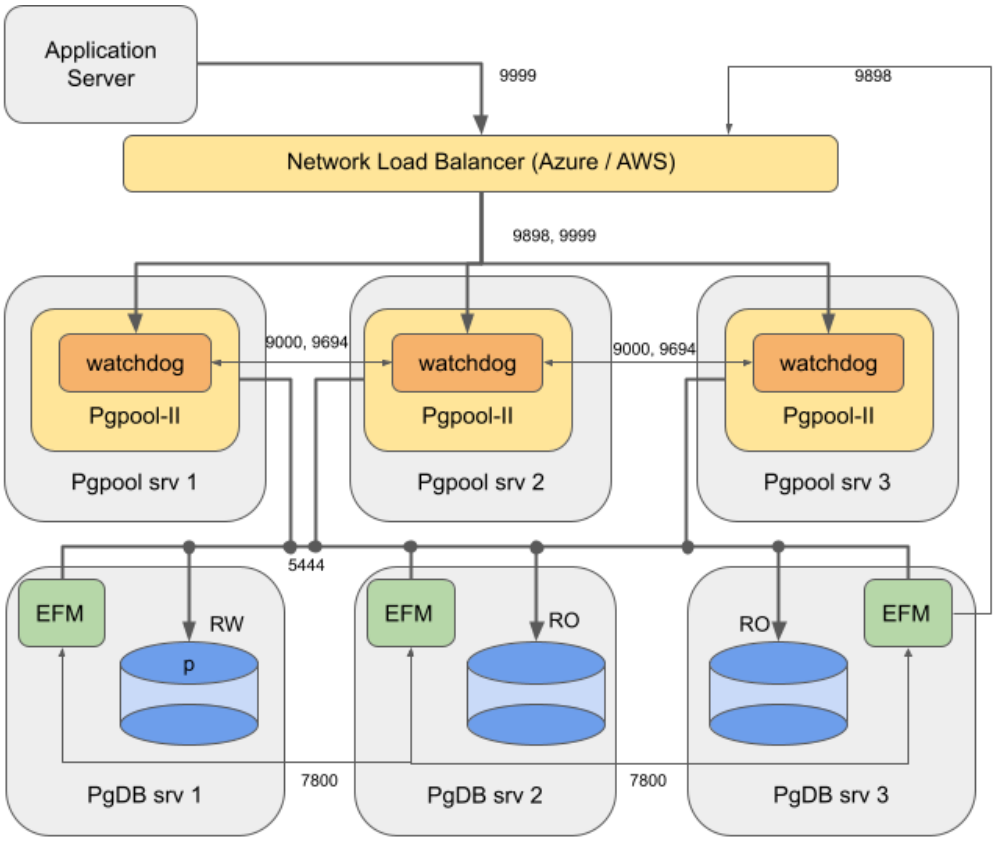

Fig. 6: Failover Manager with Pgpool in cloud

4.4.3 Using Failover Manager with EDB Pgpool-II

Installing

Install and configure Advanced Server database, Failover Manager, and EDB Pgpool-II as follows:

Systems	Components
PgDB server 1, server2, and server 3	Primary / standby node running Advanced Server 13 and Failover Manager 4.2
EDB Pgpool-II	Pgpool node running EDB Pgpool-II 4.2 in a watchdog configuration. Register these three nodes as targets in the target group. Three is the minimum and is sufficient for most cases.

EDB does not support this architecture with EDB Pgpool-II and Failover Manager/PostgreSQL running on the same machines for the following reasons:

- A restriction with cloud network load balancers doesn't route traffic properly when source and destination reside on the same machines.
- In mixed architecture, traffic between Pgpool and Postgres can become unbalanced.
- Pgpool and PostgreSQL might compete for resources.

Configuring Failover Manager

Failover Manager can remove failed database nodes from EDB Pgpool-II load balancing. It can also reattach nodes to EDB Pgpool-II when returned to the Failover Manager cluster. To configure Failover Manager for high availability using EDB Pgpool-II, you must set the following properties in the cluster properties file:

```
pgpool.enable = true

pcp.user = User invoking PCP commands

pcp.host = Virtual IP of EDB Pgpool-II or IP of NLB

pcp.port = 9898

pcp.pass.file = Absolute path of PCPPASSFILE

pgpool.bin = Absolute path of pgpool bin directory
```

Configuring EDB Pgpool-II

You can configure some important parameters in the `pgpool.conf` file to integrate EDB Pgpool-II with Failover Manager.

Backend node setting

There are three PostgreSQL backend nodes: one primary and two standby nodes. Configure using backend_* configuration parameters in `pgpool.conf` , and use the equal backend weights for all nodes. This makes the read queries distributed equally among all nodes.

```
backend_hostname0 = server1_IP

backend_port0 = 5444

backend_weight0 = 1

backend_flag0 = ALLOW_TO_FAILOVER

backend_hostname1 = server2_IP

backend_port1 = 5444

backend_weight1 = 1

backend_flag1 = ALLOW_TO_FAILOVER

backend_hostname2 = server3_IP

backend_port2 = 5444

backend_weight2 = 1

backend_flag2 = ALLOW_TO_FAILOVER
```

Enabling load balancing and streaming replication mode

Set the following configuration parameter in the `pgpool.conf` file to enable load balancing and streaming replication mode:

- For EDB Pgpool-II version 4.2:

```
backend_clustering_mode = streaming_replication

load_balance_mode = on
```

- For EDB Pgpool-II versions prior to 4.2:

```
master_slave_mode = on

master_slave_sub_mode = stream

load_balance_mode = on
```

Disabling health checking and failover

Health checking and failover are handled by Failover Manager, so disable them on the EDB Pgpool-II side. To disable the health check and failover on EDB Pgpool-II side, assign the following values:

```
health_check_period = 0

failover_on_backend_error = off

failover_if_affected_tuples_mismatch = off

failover_command =

failback_command =
```

Ensure the following while setting up the values in the pgpool.conf file:

- Keep the value of `wd_priority` in `pgpool.conf` different on each node. The node with the highest value gets the highest priority.
- The properties `backend_hostname0` , `backend_hostname1` , `backend_hostname2` , and so on are shared properties (in Failover Manager terms) and must hold the same value in the `pgpool.conf` file on all the EDB Pgpool-II nodes.
- Update the correct interface value in `if_ *` and `arping` cmd props in the `pgpool.conf` file.
- Add the properties `heartbeat_destination0` , `heartbeat_destination1` , `heartbeat_destination2` , and so on as per the number of nodes in the `pgpool.conf` file on every node. Here set `heartbeat_destination0` to the IP address or hostname of the local node.

Setting up PCP

The script uses the PCP interface, so you need to set up the PCP and `.PCPPASS` file to allow PCP connections without a password prompt.

To set up PCP, see: Configuring pcp.conf .

To set up PCPPASS, see: PCP commands .

The load balancing is turned on to ensure read scalability by distributing read traffic across the standby nodes.

The health checking and error-triggered backend failover are turned off, as Failover Manager is responsible for performing health checks and triggering failover. We do not recommend using EDB Pgpool-II to perform health checking in this case, to avoid a conflict with Failover Manager or prematurely performing failover.

Finally, `search_primary_node_timeout` is set to a low value to ensure prompt recovery of EDB Pgpool-II services upon a Failover Manager-triggered failover.

Using virtual IP addresses

Both EDB Pgpool-II and Failover Manager provide the functionality to employ a virtual IP for seamless failover. While both provide this capability, the EDB Pgpool-II leader is the process that receives the application connections through the virtual IP. As in this design, such virtual IP management is performed by the EDB Pgpool-II watchdog system. Failover Manager VIP doesn't help in this design, so disable it.

In a failure situation of the active instance of EDB Pgpool-II (the primary EDB Pgpool-II server in our sample architecture), the next available standby EDB Pgpool-II instance (according to watchdog priority) is activated and takes charge as the leader EDB Pgpool-II instance.

Configuring the network load balancer

For Failover Manager / EDB Pgpool-II integration using the network load balancer in AWS or Azure, you need to perform some additional steps.

Add the following rules to the security groups to be used by the EDB Pgpool-II instances:

- Rules for the security group to be used by the EDB Pgpool-II instances (SG Pgpool).

Type	Protocol	Port range	Source	Description
Custom TCP	TCP	9000	Entire Subnet	Watchdog
Custom TCP	TCP	9694	Entire Subnet	Heartbeat
Custom TCP	TCP	9898	Entire Subnet	pcp
Custom TCP	TCP	9999	Entire Subnet	Pgpool

In addition to these rules, add the rules for SSH and Ping as per your requirement.

- Rules for the security group to be used by the database instances (SG DB):

Type	Protocol	Port range	Source	Description
Custom TCP	TCP	7800	Entire Subnet	Failover Manager
Custom TCP	TCP	5444	Entire Subnet	Postgres

Setting these rules ensures that the ports required to run the database, Failover Manager, and EDB Pgpool-II are open for communication between the nodes and the load balancer for traffic routing and health monitoring.

In addition to these rules, add the rules for SSH and Ping as per your requirement.

Configuring NLB in Azure

If using AWS, see *Configuring NLB in AWS* .

After configuring the security group rules described in *Configuring the network load balancer*, follow the Azure documentation to:

- Create a backend pool consisting of all the virtual machines running EDB Pgpool-II instances. Use the private IPs of the virtual machines to create the backend pool.
- Add a health probe to check if the EDB Pgpool-II instance is available on the virtual machines. Set the protocol to `TCP` and the port to `9999` .

- Add two load balancing rules, one each for port `9898` and port `9999` . These rules ensure that the network traffic coming toward that port is distributed evenly among all the virtual machines present in the backend pool. Set the type to `Public` load balancer or `Internal` load balancer.

After completing these configurations, you can connect to the database on the IP address of the network load balancer on port 9999. If a failure occurs on the primary database server, Failover Manager promotes a new primary and then reconfigures EDB Pgpool-II to redistribute traffic. If any of the EDB Pgpool-II processes is not available to accept traffic, the network load balancer redistributes all the traffic to the remaining two EDB Pgpool-II processes. Make sure that the `listen_backlog_multiplier` parameter is tuned to compensate for the higher number of connections in case of failover.

Configuring NLB in AWS

The following assumptions have been taken for the sample configuration:

- All the EC2 instances and the load balancer are deployed in the same subnet. If required, you can add the database nodes to another subnet, but that requires a more complex configuration and might have a performance impact.
- There is a security group for EDB Pgpool-II and a security group for the database instances.

After configuring the security group rules described in *Configuring the network load balancer*, follow the AWS documentation to:

- Create two target groups with the following details:

Name	Type	Protocol	Port	VPC
pcp	Instances	TCP	9898	Select the VPC to which the instances are connected.
pgpool	Instances	TCP	9999	Select the VPC to which the instances are connected.

Leave the rest of the settings (**HealthcheckTCP** and **Advancedhealthcheck** settings) as default.

Register the created target groups with the instances that are running PgBouncer.

- Create a load balancer with the following details:

Type	VPC	Listener
Public or *Internal*. EDB recommends using an internal load balancer.	Choose a VPC and map it to the desired zones.	Create a listener with *TCP* set to *9898*, and forward it to the target group pcp. Create another listener with *TCP* set to *9999*, and forward it to the target group pgpool.

After completing the configurations, you can connect to the database on the IP address of the network load balancer on port 9999. If a failure occurs on the primary database server, Failover Manager promotes a new primary and then reconfigures EDB Pgpool-II to redistribute traffic. If any of the EDB Pgpool-II processes is not available to accept traffic, the network load balancer redistributes all the traffic to the remaining two EDB Pgpool-II processes. Make sure that `listen_backlog_multiplier` is tuned to compensate for the higher number of connections in case of failover.

CHAPTER

FIVE

INSTALLING FAILOVER MANAGER ON LINUX

Select a link to access the applicable installation instructions:

5.1 Linux x86-64 (amd64)

5.1.1 Red Hat Enterprise Linux (RHEL) and derivatives

- *Installing Failover Manager on RHEL 9 or OL 9 x86_64* , *Installing Failover Manager on RHEL 8 or OL 8 x86_64* , *Installing Failover Manager on RHEL 7 or OL 7 x86_64*
- *Oracle Linux (OL) 9* , *Oracle Linux (OL) 8* , *Oracle Linux (OL) 7*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86_64* , *Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86_64*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86_64* , *RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x*
- *RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x*

5.1.2 SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86_64* , *Installing Failover Manager on SLES 12 x86_64*

5.1.3 Debian and derivatives

- *Installing Failover Manager on Ubuntu 22.04 x86_64* , *Installing Failover Manager on Ubuntu 20.04 x86_64* , *Installing Failover Manager on Ubuntu 18.04 x86_64*
- *Installing Failover Manager on Debian 11 x86_64* , *Installing Failover Manager on Debian 10 x86_64*

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

5.2.1 Red Hat Enterprise Linux (RHEL) and derivatives

- *Installing Failover Manager on RHEL 9 or OL 9 x86_64* , *Installing Failover Manager on RHEL 8 or OL 8 x86_64*

5.2.2 SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86_64* , *Installing Failover Manager on SLES 12 x86_64*

Prerequisites

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

Install Java 1.8 (or later)

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

Provide an SMTP server

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

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

If an event occurs, Failover Manager invokes the script (if provided) and can also send a notification email to any email addresses specified in the `user.email` parameter of the cluster properties file. For more information about using an SMTP server, see the Red Hat deployment guide .

Configure streaming replication

Failover Manager requires that PostgreSQL streaming replication be configured between the primary node and the standby nodes. Failover Manager doesn't support other types of replication.

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

On database version 12 or later, the `primary_conninfo` and `restore_command` properties are copied from a random standby node to the stopped primary during switchover unless otherwise specified with the `-sourcenode` option.

Modify pg_hba.conf

You must modify `pg_hba.conf` on the primary and standby nodes, adding entries that allow communication between all of the nodes in the cluster. The following example shows entries you might make to the `pg_hba.conf` file on the primary node:

```
#  access for itself

host fmdb efm 127.0.0.1/32 md5
#  access for standby

host fmdb efm 192.168.27.1/32 md5
#  access for witness

host fmdb efm 192.168.27.34/32 md5
```

Where:

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

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

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

systemctl reload edb-as-

Where `x` specifies the Postgres version.

Using Autostart for the Database Servers

If a primary node reboots, Failover Manager might detect the database is down on the primary node and promote a standby node to the role of primary. If this happens, the Failover Manager agent on the rebooted primary node doesn't get a chance to write the `recovery.conf` file, and the `recovery.conf` file prevents the database server from starting. If this happens, the rebooted primary node returns to the cluster as a second primary node.

To prevent this condition, ensure that the Failover Manager agent auto starts before the database server. The agent starts in idle mode and checks to see if there is already a primary in the cluster. If there is a primary node, the agent verifies that a `recovery.conf` or `standby.signal` file exists. If neither file exits, the agent creates the `recovery.conf` file.

Ensure communication through firewalls

If a Linux firewall (that is, iptables) is enabled on the host of a Failover Manager node, you might 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 -j ACCEPT

/sbin/service iptables save
```

This command opens the port 7800. Failover Manager connects by way of the port that corresponds to the port specified in the cluster properties file.

Ensure that the database user has sufficient privileges

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

`pg_current_wal_lsn()`

`pg_last_wal_replay_lsn()`

`pg_wal_replay_resume()`

`pg_wal_replay_pause()`

If the `reconfigure.num.sync` or `reconfigure.sync.primary` property is set to `true` , then:

- For database versions 10 and later, the db.user requires `pg_read_all_stats` privilege and permissions to run `pg_reload_conf()` .

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

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

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

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

Installing Failover Manager on Linux x86 (amd64)

For operating system-specific install instructions, including accessing the repo, see:

Red Hat Enterprise Linux (RHEL) and derivatives

- *Installing Failover Manager on RHEL 9 or OL 9 x86_64*
- *Installing Failover Manager on RHEL 8 or OL 8 x86_64*
- *Installing Failover Manager on RHEL 7 or OL 7 x86_64*
- *Oracle Linux (OL) 9*
- *Oracle Linux (OL) 8*
- *Oracle Linux (OL) 7*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86_64*
- *Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86_64*
- *Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86_64*
- *RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x*
- *RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x*

SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86_64*
- *Installing Failover Manager on SLES 12 x86_64*

Debian and derivatives

- *Installing Failover Manager on Ubuntu 22.04 x86_64*
- *Installing Failover Manager on Ubuntu 20.04 x86_64*
- *Installing Failover Manager on Ubuntu 18.04 x86_64*
- *Installing Failover Manager on Debian 11 x86_64*
- *Installing Failover Manager on Debian 10 x86_64*

Installing Failover Manager on RHEL 9 or OL 9 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Install the EPEL repository

```
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.
↪rpm
```

Install the package

```
sudo dnf -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on RHEL 8 or OL 8 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Install the EPEL repository

```
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.
↪rpm
```

Install the package

```
sudo dnf -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on AlmaLinux 9 or Rocky Linux 9 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.

3. Select the platform and software that you want to download.

- Install the EPEL repository

```
sudo dnf -y install epel-release
```

- Enable additional repositories to resolve dependencies

```
sudo dnf config-manager --set-enabled powertools
```

Install the package

```
sudo dnf -y install edb-efm<4x>
```

Where `<4x>` is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on AlmaLinux 8 or Rocky Linux 8 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Install the EPEL repository

```
sudo dnf -y install epel-release
```

- Enable additional repositories to resolve dependencies

```
sudo dnf config-manager --set-enabled powertools
```

Install the package

```
sudo dnf -y install edb-efm<4x>
```

Where `<4x>` is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on RHEL 7 or OL 7 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads

- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Install the EPEL repository

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

- Enable additional repositories to resolve dependencies

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

Install the package

```
sudo yum -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be edb-efm47 .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on CentOS 7 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
dnf repolist | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Install the EPEL repository

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

Install the package

```
sudo yum -y install edb-efm<4x>
```

Where `<4x>` is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.

4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on SLES 15 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
zypper lr -E | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Activate the required SUSE module

```
sudo SUSEConnect -p PackageHub/15.4/x86_64
```

- Refresh the metadata

```
sudo zypper refresh
```

Install the package

```
sudo zypper -n install edb-efm<4x>
```

Where `<4x>` is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on SLES 12 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
zypper lr -E | grep enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

- Activate the required SUSE module

```
sudo SUSEConnect -p PackageHub/12.5/x86_64
sudo SUSEConnect -p sle-sdk/12.5/x86_64
```

- Refresh the metadata

```
sudo zypper refresh
```

Install the package

```
sudo zypper -n install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Ubuntu 22.04 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
apt-cache search enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

Install the package

```
sudo apt-get -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Ubuntu 20.04 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
apt-cache search enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

Install the package

```
sudo apt-get -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Ubuntu 18.04 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
apt-cache search enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

Install the package

```
sudo apt-get -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Debian 11 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
apt-cache search enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

Install the package

```
sudo apt-get -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Debian 10 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host (not needed for witness nodes)
- See Installing EDB Postgres Advanced Server
- See PostgreSQL Downloads
- Set up the EDB repository

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To determine if your repository exists, enter this command:

```
apt-cache search enterprisedb
```

If no output is generated, the repository isn't installed.

To set up the EDB repository:

1. Go to EDB repositories .
2. Select the button that provides access to the EDB repo.
3. Select the platform and software that you want to download.

Install the package

```
sudo apt-get -y install edb-efm<4x>
```

Where <4x> is the version of Failover Manager that you are installing. For example, if you are installing version 4.7, the package name would be `edb-efm47` .

The installation process creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres.

Initial configuration

If you are using Failover Manager to monitor a cluster owned by a user other than enterprisedb or postgres, see *Extending Failover Manager permissions* .

After installing on each node of the cluster:

1. Modify the *The cluster properties file* on each node.
2. Modify the *The cluster members file* on each node.
3. If applicable, configure and test virtual IP address settings and any scripts that are identified in the cluster properties file.
4. Start the agent on each node of the cluster. For more information, see *Controlling the Failover Manager service* .

Installing Failover Manager on Linux IBM Power (ppc64le)

For operating system-specific install instructions, including accessing the repo, see:

Red Hat Enterprise Linux (RHEL)

- *Installing Failover Manager on RHEL 9 or OL 9 x86_64*
- *Installing Failover Manager on RHEL 8 or OL 8 x86_64*

SUSE Linux Enterprise (SLES)

- *Installing Failover Manager on SLES 15 x86_64*
- *Installing Failover Manager on SLES 12 x86_64*

Installation details

Components are installed in the following locations, where `4.x` indicates a minor release:

Component	Location
Executables	*/usr/edb/efm-4.x/bin*
Libraries	*/usr/edb/efm-4.x/lib*
Cluster configuration files	*/etc/edb/efm-4.x*

continues on next page

Table 1 – continued from previous page

Component	Location
Logs	*/var/log/efm-4.x*
Lock files	*/var/lock/efm-4.x*
Log rotation file	*/etc/logrotate.d/efm-4.x*
sudo configuration file	*/etc/sudoers.d/efm-4.x*
Binary to access VIP without sudo	*/usr/edb/efm-4.x/bin/secure*

CHAPTER

SIX

UPGRADING FAILOVER MANAGER

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

1. Install Failover Manager 4.7 on each node of the cluster. For detailed information about installing Failover Manager, see

   Installing Failover Manager on Linux .

2. After installing Failover Manager, invoke the `efm upgrade-conf` utility to create the `.properties` and `.nodes` files for Failover Manager 4.7. The Failover Manager installer installs the upgrade utility (*efm upgrade-conf*) to the `/usr/edb/efm-4.7/bin directory` . To invoke the utility, assume root privileges, and invoke the command:

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

The efm `upgrade-conf` utility locates the `.properties` and `.nodes` files of preexisting clusters and copies the parameter values to a new configuration file for use by Failover Manager. The utility saves the updated copy of the configuration files in the `/etc/edb/efm-4.7` directory.

3. Modify the `.properties` and `.nodes` files for Failover Manager 4.7, specifying any new preferences. Use your choice of editor to modify any additional properties in the properties file (located in the `/etc/edb/efm-4.7` directory) before starting the service for that node. For detailed information about property settings, see

   The cluster properties file .

Note: *db.bin* is a required property. When modifying the properties file, ensure that the *db.bin* property specifies the location of the Postgres *bin* directory.

4. If you're using Eager Failover, you must disable it before stopping the Failover Manager cluster. For more information, see *Disabling Eager Failover* .

5. Use a version-specific command to stop the old Failover Manager cluster. For example, you can use the following command to stop a version 4.4 cluster:

```
/usr/efm-4.4/bin/efm stop-cluster efm
```

6. Start the new *Controlling the Failover Manager service* (`edb-efm-4.7`) on each node of the cluster.

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

```
[root@hostname ~]# /usr/edb/efm-4.7/bin/efm upgrade-conf efm
Checking directory /etc/edb/efm-4.6
```

(continues on next page)

(continued from previous page)

```
Processing efm.properties file
Checking directory /etc/edb/efm-4.5
Processing efm.nodes file

Upgrade of files is finished. The owner and group for properties and nodes files have␣
↪been set as efm.
[root@hostname ~]#
```

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

Note: If you're using a unit file, manually update the file to reflect the new Failover Manager service name when you perform an upgrade.

6.1 Uninstalling Failover Manager

Note: If you are using custom scripts, check to see if they are calling any Failover Manager scripts. For example, a script that runs after promotion to perform various tasks and then calls Failover Manager's *efm_address* script to acquire a virtual IP address. If you have any custom scripts calling Failover Manager scripts, update the custom scripts to use the newly installed version of the Failover Manager script before uninstalling the older version of the Failover Manager script.

After upgrading to Failover Manager 4.7, you can use your native package manager to remove previous installations of Failover Manager. For example, use the following command to remove Failover Manager 4.4 and any unneeded dependencies:

- On RHEL or CentOS 7.x:

```
yum remove edb-efm45
```

- On RHEL or Rocky Linux or AlmaLinux 8.x:

```
dnf remove edb-efm45
```

- On Debian or Ubuntu:

```
apt-get remove edb-efm45
```

- On SLES:

```
zypper remove edb-efm45
```

6.2 Performing a maintenance task

You can perform maintenance activities such as an OS patch or a minor database version upgrade. You can upgrade from one minor version to another (for example, from 10.1.5 to version 10.2.7) or apply a patch release for a version.

First, update the database server on each standby node of the Failover Manager cluster. Then, perform a switchover, promoting a standby node to the role of primary in the Failover Manager cluster. Then, perform a database update on the old primary node.

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

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

For detailed information about controlling the EDB Postgres Advanced Server service or upgrading your version of EDB Postres Advanced Server, see the EDB Postgres Advanced Server Guide

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

CHAPTER

SEVEN

CONFIGURING STREAMING REPLICATION

Configuring a replication scenario can be complex. For detailed information about configuration options, see the PostgreSQL core documentation .

You might want to use a `.pgpass` file to enable md5 authentication for the replication user. This might not be the safest authentication method for your environment. For more information about the supported authentication options, see the PostgreSQL core documentation .

Note: From Version 3.10 onwards, Failover Manager uses *pg_ctl* utility for standby promotion. You don't need to set the *trigger_file* or *promote_trigger_file* parameter for promotion of a standby server.

7.1 Limited support for cascading replication

While Failover Manager doesn't 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 primary node.

For detailed information about configuring cascading replication, see the PostgreSQL documentation .

To use Failover Manager in a cascading replication scenario, 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 is promoted to the role of primary node. If failover occurs, standby node #2 continues to act as a read-only replica for the new primary node until you take actions to manually reconfigure the replication scenario to contain three nodes.

In the event of a failure of standby node #1, you won't have failover protection, but you'll receive an email notifying you of the failure of the node.

Note: Performing a switchover and switch back to the original primary might not preserve the cascading replication scenario.

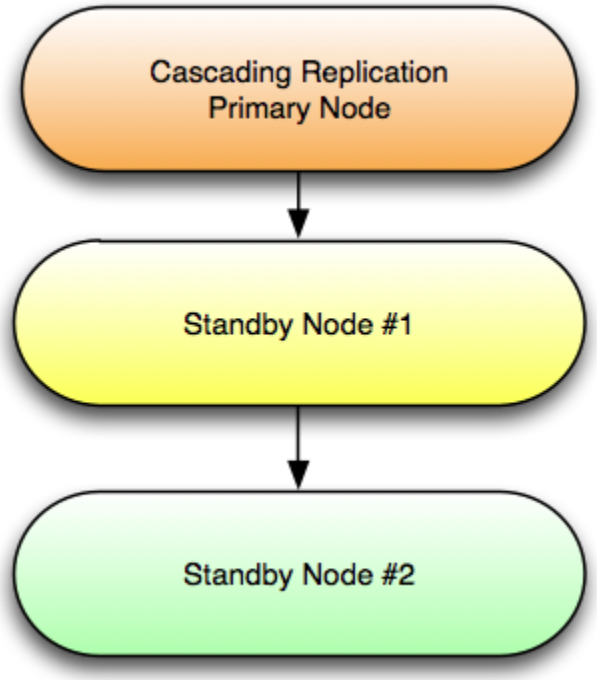

Fig. 1: Cascading replication.

CHAPTER

EIGHT

CONFIGURING FAILOVER MANAGER

See the following topics for information on how to configure Failover Manager:

- *The cluster properties file*
- *Encrypting your database password*
- *The cluster members file*
- *Extending Failover Manager permissions*
- *Using Failover Manager with virtual IP addresses*
- *Configuring for Eager Failover*

8.1 The cluster properties file

Each node in a Failover Manager cluster has a properties file (by default, named `efm.properties`) that contains the properties of the individual node on which it resides. The Failover Manager installer creates a file template for the properties file named `efm.properties.in` in the `/etc/edb/efm-4.<x>` directory.

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

```
#  cp /etc/edb/efm-4.7/efm.properties.in /etc/edb/efm-4.7/efm.properties
```

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

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

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* , modify the service script or unit file to instruct Failover Manager to use a different name.

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

The property files are owned by root. The Failover Manager service script expects to find the files in the `/etc/edb/efm-4.<x>` directory. If you move the property file to another location, you must create a symbolic link that specifies the new location.

Note: All user scripts referenced in the properties file are invoked as the Failover Manager user.

8.1.1 Specifying cluster properties

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

Property values are case sensitive. While Postgres uses quoted strings in parameter values, Failover Manager doesn't allow quoted strings in property values. For example, while you might specify an IP address in a Postgres configuration parameter as:

```
listen_addresses='192.168.2.47'
```

With Failover Manager, don't enclose the value in quotes:

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

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

Legends: In the following table:

- `A` : Required on primary or standby node
- `W` : Required on witness node
- `Y` : Yes

Property name	A	W	Default value	Comments
efm set-priority	Y	Y		Username for the database.
efm set-priority	Y	Y		Password encrypted using 'efm encrypt'.
efm set-priority	Y	Y		This value must be same for all the agents.
efm set-priority	Y	Y		Database name.
efm set-priority	Y			Owner of $PGDATA dir for db.database.
efm set-priority				Required if running the database as a service.
efm set-priority	Y			Directory containing the pg_controldata/pg_ctl commands such as '/usr/edb/as*nn*/bin'.
efm set-priority	Y			Same as the output of query 'show data_directory;'
efm set-priority				Same as the output of query 'show config_file;'. Should be specified if it is not same as *db.data.dir*.
efm set-priority	Y	Y	disable	See the *Important notes* .
efm set-priority				This value must be same for all the agents; can be left blank if using a notification script.
efm set-priority .			*efm set-priority*	Leave blank to use the default *efm set-priority* .
efm set-priority	Y	Y	INFO	See the *efm set-priority* .
efm set-priority				
efm set-priority				Required if user.email property is not used; both parameters can be used together.
efm set-priority	Y	Y		Example: <ip_address>:<port>
efm set-priority				Example: <ip_address/hostname>
efm set-priority	Y	Y	7809	Modify if the default port is already in use.
efm set-priority	Y	Y		See description.
efm set-priority	Y		10	
efm set-priority	Y		60	
efm set-priority	Y		10	
efm set-priority	Y	Y	10	

continues on next page

Table 1 – continued from previous page

Property name	A	W	Default value	Comments
efm set-priority	Y	Y	50	This value must be same for all the agents.
efm set-priority	Y	Y	false	This value must be same for all the agents
efm set-priority			true	This value must be same for all the agents. **Available in Failover Manager 4.2 and later.**
efm set-priority	Y		true	This value must be same for all the agents.
efm set-priority	Y		true	
efm set-priority	Y	Y	false	
efm set-priority	Y		0	
efm set-priority	Y	Y	8.8.8.8	
efm set-priority	Y	Y	/bin/ping -q -c3 -w5	
efm set-priority	Y	Y	false	
efm set-priority	Y	Y	false	
efm set-priority	Y		0	
efm set-priority	Y	Y	true	
efm set-priority	Y		true	This value must be same for all the agents.
efm set-priority	Y		true	
efm set-priority	Y	Y	true	This value must be same for all the agents.
efm set-priority			0	
efm set-priority				Set to replace the application_name portion of the primary_conninfo entry with this property value before starting the original primary database as a standby.
efm set-priority				Example: restore.command=scp <db_service_owner>@%h: <archive_path>/%f %p
efm set-priority	Y		false	If you are on Failover Manager 4.1, see reconfigure_num_sync_max to raise num_sync.
efm set-priority				**Available in Failover Manager 4.1 and later.**
efm set-priority	Y		false	
efm set-priority	Y	Y	0	This value must be same for all the nodes.
efm set-priority				**Available in Failover Manager 4.2 and later.**
efm set-priority	Y		1	
efm set-priority			60	
efm set-priority	Y		0	
efm set-priority			(see virtual.ip.single	Leave blank if you do not specify a VIP.
efm set-priority				Required if you specify a VIP.
efm set-priority				Required if you specify a VIP.
efm set-priority	Y	Y	Yes	This value must be same for all the nodes.
efm set-priority	Y	Y	Yes	
efm set-priority			false	**Available in Failover Manager 4.1 and later.**
efm set-priority				Required if pgpool.enable is set to true. **Available in Failover Manager 4.1 and later.**
efm set-priority				Required if pgpool.enable is set to true, this value must be same for all the agents. **Available in Failover Manager 4.1 and later.**
efm set-priority				Required if pgpool.enable is set to true, this value must be same for all the agents. **Available in Failover Manager 4.1 and later.**

continues on next page

Table 1 – continued from previous page

Property name	A	W	Default value	Comments
efm set-priority				Required if pgpool.enable is set to true. **Available in Failover Manager 4.1 and later.**
efm set-priority				Required if pgpool.enable is set to true. **Available in Failover Manager 4.1 and later. **
efm set-priority				Example: script.load.balancer.attach= /<path>/<attach_script> %h %t
efm set-priority				Example: script.load.balancer.detach= /<path>/<detach_script> %h %t
efm set-priority			true	Set to false if you want to keep a running primary database attached. **Available in Failover Manager 4.2 and later.**
efm set-priority				Example: script.fence= /<path>/<script_name> %p %f
efm set-priority				Example: script.post.promotion= /<path>/<script_name> %f %p
efm set-priority				Example: script.resumed= /<path>/<script_name>
efm set-priority				Example: script.db.failure= /<path>/<script_name>
efm set-priority				Example: script.primary.isolated= /<path>/<script_name>
efm set-priority				Example: script.remote.pre.promotion= /<path>/<script_name> %p
efm set-priority				Example: script.remote.post.promotion= /<path>/<script_name> %p
efm set-priority				Example: script.custom.monitor= /<path>/<script_name>
efm set-priority				Required if a custom monitoring script is specified.
efm set-priority				Required if a custom monitoring script is specified.
efm set-priority				Required if a custom monitoring script is specified.
efm set-priority	Y	Y	sudo	
efm set-priority	Y	Y	sudo -u %u	
efm set-priority				If not specified, defaults to '/var/lock/efm-<version>'
efm set-priority				If not specified, defaults to '/var/log/efm-<version>'
efm set-priority			localhost	
efm set-priority			514	
efm set-priority				
efm set-priority			UDP	
efm set-priority	Y	Y	true	
efm set-priority	Y	Y	false	
efm set-priority			info	
efm set-priority			info	
efm set-priority			-Xmx128m	

8.1.2 Cluster properties

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

```
#  The value for the password property should be the output from

#  efm encrypt -- do not include a cleartext password here. To

#  prevent accidental sharing of passwords among clusters, the

#  cluster name is incorporated into the encrypted password. If

#  you change the cluster name (the name of this file), you must

#  encrypt the password again with the new name.

#  The db.port property must be the same for all nodes.

db.user=
db.password.encrypted=
db.port=
db.database=
```

The `db.user` specified must have enough privileges to invoke selected PostgreSQL commands on behalf of Failover Manager. For more information, see *Prerequisites* .

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

Use the `db.service.owner` property to specify the name of the operating system user that owns the cluster that is being managed by Failover Manager. This property isn't 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.data.dir specified below.

db.service.owner=
```

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

```
#  Specify the proper service name in order to use service commands

#  rather than pg_ctl to start/stop/restart a database. For example, if

#  this property is set, then service <name> restart or systemctl

#  restart <name>

#  (depending on OS version) will be used to restart the database rather

#  than pg_ctl.

#  This property is required if running the database as a service.

db.service.name=
```

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

```
#  Specify the directory containing the pg_controldata/pg_ctl commands,

#  for example:

#  /usr/edb/as12/bin. Unless the db.service.name property is used, the

#  pg_ctl command is used to start/stop/restart databases as needed

#  after a failover or switchover. This property is required.

db.bin=
```

Use the `db.data.dir` property to specify the location to write a recovery file on the primary node of the cluster during promotion. This property is required on primary and standby nodes. It isn't required on a dedicated witness node.

```
#  For database version 12 and up, this is the directory where a

#  standby.signal file will exist for a standby node. For previous

#  versions, this is the location of the db recovery.conf file on

#  the node.

#  After a failover, the recovery.conf files on remaining standbys are

#  changed to point to the new primary db (a copy of the original is made

#  first). On a primary node, a recovery.conf file will be written during

#  failover and promotion to ensure that the primary node can not be

#  restarted as the primary database.
```

(continues on next page)

(continued from previous page)

```
#  This corresponds to database environment variable PGDATA and should

#  be same as the output of query show data_directory; on respective

#  database.

db.data.dir=
```

Use the `db.data.dir` property to specify the location to write a recovery file on the primary node of the cluster during promotion. This property is required on primary and standby nodes. It isn't required on a dedicated witness node.

```
#  For database version 12 and up, this is the directory where a

#  standby.signal file will exist for a standby node. For previous

#  versions, this is the location of the db recovery.conf file on

#  the node.

#  After a failover, the recovery.conf files on remaining standbys are

#  changed to point to the new primary db (a copy of the original is made

#  first). On a primary node, a recovery.conf file will be written during

#  failover and promotion to ensure that the primary node can not be

#  restarted as the primary database.

#  This corresponds to database environment variable PGDATA and should

#  be same as the output of query show data_directory; on respective

#  database.

db.data.dir=
```

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

```
#  Specify the location of database configuration files if they are

#  not contained in the same location as the recovery.conf or

#  standby.signal file. This is most likely the case for Debian

#  installations. The location specified will be used as the -D value

#  (the location of the data directory for the cluster) when calling
```

(continues on next page)

(continued from previous page)

```
#  pg_ctl to start or stop the database. If this property is blank,

#  the db.data.dir location specified by the db.data.dir property will

#  be used. This corresponds to the output of query show config_file;

#  on respective database.

db.config.dir=
```

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

Use the `jdbc.sslmode` property to instruct Failover Manager to use SSL connections. By default, SSL is disabled.

```
#  Use the jdbc.sslmode property to enable ssl for EFM

#  connections. Setting this property to anything but disable

#  will force the agents to use ssl=true for all JDBC database

#  connections (to both local and remote databases).

#  Valid values are:

#
#  disable - Do not use ssl for connections.

#  verify-ca - EFM will perform CA verification before allowing

#  the certificate.

#  require - Verification will not be performed on the server

#  certificate.

jdbc.sslmode=disable
```

Note: If you set the value of *jdbc.sslmode* to *verify-ca* and you want to use Java trust store for certificate validation, you need to set the following value:

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

For information about configuring and using SSL, see Secure TCP/IP Connections with SSL and
Using SSL in the PostgreSQL documentation.

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

```
#  Email address(es) for notifications. The value of this

#  property must be the same across all agents. Multiple email
```

(continues on next page)

(continued from previous page)

```
#  addresses must be separated by space. If using a notification

#  script instead, this property can be left blank.

user.email=
```

The `from.email` property specifies the value to use as the sender's address for email notifications from Failover Manager. You can:

- Leave `from.email` blank to use the default value (`efm@localhost`).
- Specify a custom value for the email address.
- Specify a custom email address, using the `%h` placeholder to represent the name of the node host (for example, example@%h). The placeholder is replaced with the name of the host as returned by the Linux hostname utility.

For more information about notifications, see *Notifications* .

```
#  Use the from.email property to specify the from email address that

#  will be used for email notifications. Use the %h placeholder to

#  represent the name of the node host (e.g. example@%h). The

#  placeholder will be replaced with the name of the host as returned

#  by the hostname command.

#  Leave blank to use the default, efm@localhost.

from.email=
```

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

Notifications .

```
#  Minimum severity level of notifications that will be sent by

#  the agent. The minimum level also applies to the notification

#  script (below). Valid values are INFO, WARNING, and SEVERE.

#  A list of notifications is grouped by severity in the users

#  guide.

notification.level=INFO
```

Use the `notification.ini.prefix` property to specify the ini to add to the beginning of every notification.

```
#  Text to add to the beginning of every notification. This could

#  be used to help identify what the cluster is used for, the role
```

(continues on next page)

(continued from previous page)

```
#  of this node, etc. To use multiple lines, add a backslash \ to

#  the end of a line of ini. To include a newline use \n.

#  Example:

#  notification.ini.prefix=Development cluster for Example dept.\n\

#  Used by Dev and QA \

#  See Example group for questions.

notification.ini.prefix=
```

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

```
#  Absolute path to script run for user notifications.

#
#  This is an optional user-supplied script that can be used for

#  notifications instead of email. This is required if not using

#  email notifications. Either/both can be used. The script will

#  be passed two parameters: the message subject and the message

#  body.

script.notification=
```

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

```
#  This property specifies the ip address and port that jgroups

#  will bind to on this node. The value is of the form

#  <ip>:<port>.

#  Note that the port specified here is used for communicating

#  with other nodes, and is not the same as the admin.port below,

#  used only to communicate with the local agent to send control

#  signals.

#  For example, <provide_your_ip_address_here>:7800
```

(continues on next page)

(continued from previous page)

```
bind.address=
```

Use the `external.address` property to specify the IP address or hostname to use for communication with all other Failover Manager agents in a NAT environment.

```
#  This is the ip address/hostname to be used for communication with all

#  other Failover Manager agents. All traffic towards this address

#  should be routed by the network to the bind.address of the node.

#  The value is in the ip/hostname format only. This address will be

#  used in scenarios where nodes are on different networks and broadcast

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

external.address=
```

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

```
#  This property controls the port binding of the administration

#  server which is used for some commands (ie cluster-status). The

#  default is 7809; you can modify this value if the port is

#  already in use.

admin.port=7809
```

Set the `is.witness` property to `true` to indicate that the current node is a witness node. If `is.witness` is `true` , the local agent doesn't 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 EDB Postgres Advanced Server `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 isn't 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 primary.
- If there's no local database, the agent assumes an idle state.

Note: If *is.witness* is *true* , Failover Manager doesn't check the recovery state.

The following properties apply to the local server:

- The `local.period` property specifies the number of seconds between attempts to contact the database server.
- The `local.timeout` property specifies the number of seconds an agent waits for a positive response from the local database server.
- The `local.timeout.final` property specifies the number of seconds an agent waits after the previous checks have failed to contact the database server on the current node. If a response isn't received from the database within the number of seconds specified by the `local.timeout.final` property, the database is assumed to have failed.

For example, given the default values of these properties, a check of the local database happens once every 10 seconds. If an attempt to contact the local database doesn't come back positive within 60 seconds, Failover Manager makes a final attempt to contact the database. If a response isn't received within 10 seconds, Failover Manager declares database failure and notifies the administrator listed in the `user.email` property. These properties aren't 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, modify these values to suit your business model.

Use the `remote.timeout` property to limit how many seconds an agent waits for a response from a remote agent or database. Agents only send messages to each other during cluster events. Examples include:

- Attempting to connect to a remote database that may have failed and asking other agents if they can connect.
- A primary agent requesting recovery settings from a standby agent as part of a switchover.
- Telling nodes to prepare to shut down when stopping the Failover Manager cluster.

```
#  Timeout for a call to check if a remote database is responsive.

#  For example, this is how long a standby would wait for a

#  DB ping request from itself and the witness to the primary DB
```

(continues on next page)

(continued from previous page)

```
#  before performing failover.

remote.timeout=10
```

Use the `node.timeout` property to specify the number of seconds for an agent to wait for a heartbeat from another node when determining if a node has failed.

```
#  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 `encrypt.agent.messages` property to specify whether to encrypt the messages sent between agents.

```
#  Set to true to encrypt messages that are sent between agents.

#  This property must be the same on all agents or else the agents

#  will not be able to connect.

encrypt.agent.messages=false
```

Use the `enable.stop.cluster` property to enable or disable the `stop-cluster` command. The command is a convenience in some environments but can cause issues when unintentionally invoked. In Eager Failover mode, the command results in stopping EDB Postgres Advanced Server without failover.

```
#  Whether or not the efm stop-cluster <cluster name> command is enabled.

#  Set to false to disable the command, in which case all Failover

#  Manager agents must be stopped individually. Note that stopping each

#  agent separately will change the .nodes files on remaining agents

#  unless stable.nodes.file is also true. This property value must

#  be the same on all agents if set. The default is true if not set.

enable.stop.cluster=true
```

Use the `stop.isolated.primary` property to instruct Failover Manager to shut down the database if a primary agent detects that it's isolated. When `true` (the default), Failover Manager stops the database before invoking the script specified in the `script.primary.isolated` property.

```
#  Shut down the database after a primary agent detects that it has

#  been isolated from the majority of the efm cluster. If set to
```

(continues on next page)

(continued from previous page)

```
#  true, efm will stop the database before running the

#  script.primary.isolated script, if a script is specified.

stop.isolated.primary=true
```

Use the `stop.failed.primary` property to instruct Failover Manager to attempt to shut down a primary database if it can't reach the database. If `true` , Failover Manager runs the script specified in the `script.db.failure` property after attempting to shut down the database.

```
#  Attempt to shut down a failed primary database after EFM can no

#  longer connect to it. This can be used for added safety in the

#  case a failover is caused by a failure of the network on the

#  primary node.

#  If specified, a script.db.failure script is run after this attempt.

stop.failed.primary=true
```

Use the `primary.shutdown.as.failure` property to treat any shutdown of the Failover Manager agent on the primary node as a failure. If this property is set to `true` and the primary agent is shut down, the rest of the cluster treats the shutdown as a failure. This includes any proper shutdown of the agent such as a shutdown of the whole node. None of the timeout properties apply in this case: when the agent exits, the rest of the cluster is notified immediately. After the agent exits, the rest of the cluster performs *checks* that happen in the case of a primary agent failure. The *checks* include attempting to connect to the primary database, seeing if the VIP is reachable if used, and so on).

- If the database is reached, a notification is sent informing you of the agent status.
- If the database isn't reached, a failover occurs.

```
#  Treat a primary agent shutdown as an agent failure. This can be set

#  to true to treat a primary agent shutdown as a failure situation.

#  Caution should be used when using this feature, as it could

#  cause an unwanted promotion in the case of performing primary

#  database maintenance.

#  Please see the users guide for more information.

primary.shutdown.as.failure=false
```

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

To perform maintenance on the primary database when `primary.shutdown.as.failure` is `true` , stop the primary agent and wait to receive a notification that the primary agent has failed but the database is still running. Then, it is safe

to stop the primary database. Alternatively, you can use the `stop-cluster` command to stop all of the agents without performing failure checks.

Use the `update.physical.slots.period` property to define the slot advance frequency for database version 12 and later. When `update.physical.slots.period` is set to a positive integer value, the primary agent reads the current `restart_lsn` of the physical replication slots after every `update.physical.slots.period` seconds and sends this information with its `pg_current_wal_lsn` and `primary_slot_name` (if it is set in the postgresql.conf file) to the standbys. The physical slots must already exist on the primary for the agent to find them. If physical slots do not already exist on the standbys, standby agents create the slots and then update `restart_lsn` parameter for these slots. A non-promotable standby doesn't create new slots but updates them if they exist.

Before updating the `restart_lsn` value of a slot, the agent checks to see if an `xmin` value has been set, which may happen if this was previously a primary node. If an `xmin` value has been set for the slot, the agent drops and recreates the slot before updating the `restart_lsn` value.

Note: all slot names, including one set on the current primary if desired, must be unique.

```
#  Period in seconds between having the primary agent update promotable

#  standbys with physical replication slot information so that

#  the cluster will continue to use replication slots after a failover.

#  Set to zero to turn off.

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

Use the `ping.server.ip` property to specify the IP address of a server that Failover Manager can use to confirm that network connectivity isn't 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

#  (primary, standby, or witness) because this ping server is meant

#  to provide an additional layer of information should the EFM

#  nodes lose sight of each other.
```

(continues on next page)

(continued from previous page)

```
#
#  The installation default is Googles DNS server.

ping.server.ip=8.8.8.8
```

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

```
#  This command will be used to test the reachability of certain

#  nodes.

#
#  Do not include an IP address or hostname on the end of

#  this command - it will be added dynamically at runtime with the

#  values contained in virtual.ip and ping.server.ip.

#
#  Make sure this command returns reasonably quickly - test it

#  from a shell command line first to make sure it works properly.

ping.server.command=/bin/ping -q -c3 -w5
```

Use the `auto.allow.hosts` property to instruct the server to use the addresses specified in the `.nodes` file of the first nostarts to update the allowed host list. Enabling this property by setting `auto.allow.hosts` to `true` can simplify cluster startup.

```
#  Have the first nostarts automatically add the addresses

#  from its .nodes file to the allowed host list. This will make

#  it faster to start the cluster when the initial set of hosts

#  is already known.

auto.allow.hosts=false
```

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

```
#  When set to true, EFM will not rewrite the .nodes file whenever

#  new nodes join or leave the cluster. This can help starting a

#  cluster in the cases where it is expected for member addresses

#  to be mostly static, and combined with auto.allow.hosts makes

#  startup easier when learning failover manager.
```

(continues on next page)

(continued from previous page)

```
stable.nodes.file=false
```

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

```
#  This property controls how many times a database connection is

#  reused before creating a new one. If set to zero, a new

#  connection will be created every time an agent pings its local

#  database.

db.reuse.connection.count=0
```

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

```
#  Whether or not failover will happen automatically when the primary

#  fails. Set to false if you want to receive the failover notifications

#  but not have EFM actually perform the failover steps.

#  The value of this property must be the same across all agents.

auto.failover=true
```

Use the `auto.reconfigure` property to instruct Failover Manager to enable or disable automatic reconfiguration of remaining standby servers after the primary standby is promoted to primary. Set the property to `true` (the default) to enable automatic reconfiguration or `false` to disable automatic reconfiguration. This property isn't required on a dedicated witness node. If you're using EDB Postgres Advanced Server or PostgreSQL version 11 or earlier, the `recovery.conf` file is backed up during the reconfiguration process.

```
#  After a standby is promoted, Failover Manager will attempt to

#  update the remaining standbys to use the new primary. For database

#  versions before 12, Failover Manager will back up recovery.conf.

#  Then it will change the host parameter of the primary_conninfo entry

#  in recovery.conf or postgresql.auto.conf, and restart the database.

#  The restart command is contained in either the efm_db_functions or

#  efm_root_functions file; default when not running db as an os

#  service is: "pg_ctl restart -m fast -w -t <timeout> -D <directory>"

#  where the timeout is the local.timeout property value and the
```

(continues on next page)

(continued from previous page)

```
#  directory is specified by db.data.dir. To turn off

#  automatic reconfiguration, set this property to false.

auto.reconfigure=true
```

Note: *primary_conninfo* is a space-delimited list of keyword=value pairs.

Use the `promotable` property to indicate not to promote a node. The `promotable` property is ignored when a primary agent starts. This simplifies switching back to the original primary after a switchover or failover. To override the setting, use the `efm set-priority` command at runtime. For more information about the `efm set-priority` command, see *Using the efm utility* .

```
#  A standby with this set to false will not be added to the

#  failover priority list, and so will not be available for

#  promotion. The property will be used whenever an agent starts

#  as a standby or resumes as a standby after being idle. After

#  startup/resume, the node can still be added or removed from the

#  priority list with the efm set-priority command. This

#  property is required for all non-witness nodes.

promotable=true
```

If the same amount of data was written to more than one standby node and a failover occurs, the `use.replay.tiebreaker` value determines how Failover Manager selects a replacement primary. Set the `use.replay.tiebreaker` property to `true` to instruct Failover Manager to failover to the node that will come out of recovery faster, as determined by the log sequence number. To ignore the log sequence number and promote a node based on user preference, set `use.replay.tiebreaker` to `false` .

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

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

#  consider replay location as more important than failover priority

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

#  standby to promote.

use.replay.tiebreaker=true
```

Use the `standby.restart.delay` property to specify the time in seconds for the standby to wait before it gets reconfigured (stoppstarts) to follow the new primary after a promotion.

```
#  Time in seconds for this standby to delay restarting to follow the

#  primary after a promotion. This can be used to have standbys restart

#  at different times to increase availability. Caution should be used

#  when using this feature, as a delayed standby will not be following

#  the new primary and care must be taken that the new primary retains

#  enough WAL for the standby to follow it.

#  Please see the users guide for more information.

standby.restart.delay=0
```

You can use the `application.name` property to provide the name of an application to copy to the `primary_conninfo` parameter before restarting an old primary node as a standby.

```
#  During a switchover, recovery settings are copied from a standby

#  to the original primary. If the application.name property is set,

#  Failover Manager will replace the application_name portion of the

#  primary_conninfo entry with this property value before starting

#  the original primary database as a standby. If this property is

#  not set, Failover Manager will remove the parameter value

#  from primary_conninfo.

application.name=
```

Note: Set the *application.name* property on the primary and any promotable standby. In the event of a failover/switchover, the primary node can potentially become a standby node again.

Use the `restore.command` property to instruct Failover Manager to update the `restore_command` value when a new primary is promoted. `%h` represents the address of the new primary. Failover Manager replaces `%h` with the address of the new primary. `%f` and `%p` are placeholders used by the server. If the property is left blank, Failover Manager doesn't update the `restore_command` values on the standbys after a promotion.

See the PostgreSQL documentation for more information about using a
restore_command .

```
#  If the restore_command on a standby restores directly from the

#  primary node, use this property to have Failover Manager change

#  the command when a new primary is promoted.
```

(continues on next page)

(continued from previous page)

```
#
#  Use the %h placeholder to represent the address of the new primary.

#  During promotion it will be replaced with the address of the new

#  primary.

#
#  If not specified, failover manager will not change the

#  restore_command value, if any, on standby nodes.

#
#  Example:

#  restore.command=scp <db service owner>@%h:/var/lib/edb/as12/data/archive/%f %p

restore.command=
```

The database parameter `synchronous_standby_names` on the primary node specifies the names and count of the synchronous standby servers that confirm receipt of data to ensure that the primary nodes can accept write transactions. When the `reconfigure.num.sync` property is set to `true` , Failover Manager reduces the number of synchronous standby servers and reloads the configuration of the primary node to reflect the current value.

```
#  Reduce num_sync when the number of synchronous standbys drops below

#  the value required by the primary database. If set to true, Failover

#  Manager will reduce the number of standbys needed in the primarys

#  synchronous_standby_names property and reload the primary

#  configuration. Failover Manager will not reduce the number below 1,

#  taking the primary out of synchronous replication, unless the

#  reconfigure.sync.primary property is also set to true.

#  To raise num_sync, see the reconfigure.num.sync.max property below.

reconfigure.num.sync=false
```

Note: If you're using the *reconfigure.num.sync* property, make sure that the *wal_sender_timeout* value in the primary database is set to at least 10 seconds less than the *efm.node.timeout* value.

Use the `reconfigure.num.sync.max` property to specify the maximum number to which num-sync can be raised when a standby is added to the cluster.

```
#  If reconfigure.num.sync is set to true and this property is set,

#  Failover Manager will check if num_sync can be raised when a standby
```

(continues on next page)

(continued from previous page)

```
#  is added to the cluster.

#  Failover Manager will not raise the value above the maximum set here.

#  If the primary database has been taken out of synchronous mode

#  completely (see the reconfigure.sync.primary property), then Failover

#  Manager will not reconfigure the primary database if standbys are

#  added to the cluster.

reconfigure.num.sync.max=
```

Set the `reconfigure.sync.primary` property to `true` to take the primary database out of synchronous replication mode if the number of standby nodes drops below the level required. Set `reconfigure.sync.primary` to `false` to send a notification if the standby count drops without interrupting synchronous replication.

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

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

#  synchronous_standby_names configuration parameter on the primary

#  if the number of synchronous standbys drops below the required

#  level for the primary to accept writes.

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

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

#  required level.

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

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

#  YOU ARE SURE THIS IS OK.

reconfigure.sync.primary=false
```

Note: If you're using the *reconfigure.sync.primary* property, ensure that the *wal_sender_timeout* value in the primary database is set to at least 10 seconds less than the *efm.node.timeout* value.

Use the `minimum.standbys` property to specify the minimum number of standby nodes to retain on a cluster. If the standby count drops to the specified minimum, a replica node isn't promoted if a failure of the primary node occurs.

```
#  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 `priority.standbys` property to specify the priority of standbys after this node is promoted.

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

#  promotion when this node is the primary. If set, when this node is

#  promoted, addresses in this list will be added to the front of the

#  standby priority list. If this list contains addresses that are not

#  standbys at the time of promotion, they will not be added.

priority.standbys=
```

Use the `recovery.check.period` property to specify the number of seconds for Failover Manager to wait before it 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=1
```

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

```
#  Time in seconds to keep trying to connect to a database after a

#  start or restart command returns successfully but the database

#  is not ready to accept connections yet (a rare occurance). This

#  applies to standby databases that are restarted when being

#  reconfigured for a new primary, and to primary databases that

#  are stopped astarts as standbys during a switchover.

#  This retry mechanism is unrelated to the auto.resume.period
```

(continues on next page)

(continued from previous page)

```
#  parameter.

restart.connection.timeout=60
```

Use the `auto.resume.period` property to specify the number of seconds for an agent to attempt to resume monitoring that database. This property applies after a monitored database fails and an agent has assumed an idle state or when starting in IDLE mode.

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

#  after a database failure or when starting in IDLE mode. Set to

#  0 for agents to not try to resume (in which case the

#  efm resume <cluster> command is used after bringing a

#  database back up).

auto.resume.period=0
```

Failover Manager provides support for clusters that use a virtual IP. If your cluster uses a virtual IP, provide the host name or IP address in the `virtual.ip` property. Specify the corresponding prefix in the `virtual.ip.prefix` property. Leave `virtual.ip` to disable virtual IP support.

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

The specified virtual IP address is assigned only to the primary node of the cluster. If you specify `virtual.ip.single=true` , the same VIP address is used on the new primary if a failover occurs. Specify a value of `false` to provide a unique IP address for each node of the cluster.

For information about using a virtual IP address, see *Using Failover Manager with virtual IP addresses* .

```
#  These properties specify the IP and prefix length that will be

#  remapped during failover. If you do not use a VIP as part of

#  your failover solution, leave the virtual.ip property blank to

#  disable Failover Manager support for VIP processing (assigning,

#  releasing, testing reachability, etc).

#
#  If you specify a VIP, the interface and prefix are required.

#
#  If you specify a host name, it will be resolved to an IP address

#  when acquiring or releasing the VIP. If the host name resolves

#  to more than one IP address, there is no way to predict which

#  address Failover Manager will use.
```

(continues on next page)

(continued from previous page)

```
#
#  By default, the virtual.ip and virtual.ip.prefix values must be

#  the same across all agents. If you set virtual.ip.single to

#  false, you can specify unique values for virtual.ip and

#  virtual.ip.prefix on each node.

#
#  If you are using an IPv4 address, the virtual.ip.interface value

#  should not contain a secondary virtual ip id (do not include

#  ":1", etc).

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

Note: If a primary agent starts and the node doesn't currently have the VIP, the Failover Manager agent acquires it. Stopping a primary agent doesn't drop the VIP from the node.

Set the `check.vip.before.promotion` property to `false` to prevent Failover Manager from checking to see if a VIP is in use before assigning it to a new primary in case of a failure. This might result in multiple nodes broadcasting on the same VIP address. Unless the primary node is isolated or can be shut down via another process, set this property `to true` .

```
#  Whether to check if the VIP (when used) is still in use before

#  promoting after a primary failure. Turning this off may allow

#  the new primary to have the VIP even though another node is also

#  broadcasting it. This should only be used in environments where

#  it is known that the failed primary node will be isolated or

#  shut down through other means.

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

Use the `pgpool.enable` property to specify if you want to enable the Failover Manager and Pgpool integration for high availability. If you want to enable Pgpool integration in a non-sudo mode (running as the DB owner), the PCPPASS file must be owned by the DB owner operating system user and you must set the file permissions to 600.

```
#  A boolean property to enable Failover Manager managed Pgpool HA.

#  If enabled, Failover Manager would natively update the joining
```

(continues on next page)

(continued from previous page)

```
#  and leaving status of database nodes to active pgpool instance.

#  Failover manager expects properly configured and running pgpool

#  instances on required nodes. It does not manage setup and

#  configuration of pgpool on any node.

#
#  By default the property is disabled.

pgpool.enable=false
```

Use the following parameters to specify the values to use for Pgpool integration.

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

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

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

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

You don't need to set the following properties if you are using Pgpool as the load balancer solution and you have set the Pgpool integration properties.

Provide a script named after the `script.load.balancer.attach` property to identify a script to invoke when you want to attach a node to the load balancer. Use the `script.load.balancer.detach` property to specify the name of a script to invoke when you want to detach node from the load balancer. Include the `%h` placeholder to represent the IP address of the node that's being attached or removed from the cluster. Include the `%t` placeholder to instruct Failover Manager to include a p (for a primary node) or an s (for a standby node) in the string.

```
#  Absolute path to load balancer scripts

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

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

# and the letter s for standby nodes.
#
#  Example:

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

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

Use the `detach.on.agent.failure` property to indicate that you don't want to detach a node from the load balancer in a scenario where the primary agent fails but the database is still reachable. The default value is `true.`

```
#  If set to true, Failover Manager will detach the node from load

#  balancer if the primary agent fails but the database is still

#  reachable. In most scenarios this is NOT the desired situation. In

#  scenarios where the detach script should run with a failed primary

#  agent, even when the primary database is still healthy this parameter

#  should be set to true. If no value specified it defaults to true (for

#  backwards compatibility).

#  This is not applicable for standbys.

detach.on.agent.failure=
```

The `script.fence` property specifies the path to an optional user-supplied script to invoke during the promotion of a standby node to primary 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.
```

(continues on next page)

(continued from previous page)

```
#
#  Parameters can be passed into this script for the failed primary

#  and new primary node addresses. Use %p for new primary and %f

#  for failed primary. On a node that has just been promoted, %p

#  should be the same as the nodes efm binding address.

#
#  Example:

#  script.fence=/somepath/myscript %p %f

#
#  NOTE: FAILOVER WILL NOT OCCUR IF THIS SCRIPT RETURNS A NON-ZERO EXIT

#  CODE.

script.fence=
```

Use the `script.post.promotion` property to specify the path to an optional user-supplied script to invoke after a standby node is promoted to primary.

```
#  Absolute path to fencing script run after promotion

#
#  This is an optional user-supplied script that will be run after

#  failover on the standby node after it has been promoted and

#  is no longer in recovery. The exit code from this script has

#  no effect on failover manager, but will be included in a

#  notification sent after the script executes.

#
#  Parameters can be passed into this script for the failed primary

#  and new primary node addresses. Use %p for new primary and %f

#  for failed primary. On a node that has just been promoted, %p

#  should be the same as the nodes efm binding address.

#
#  Example:

#  script.post.promotion=/somepath/myscript %f %p
```

(continues on next page)

(continued from previous page)

```
script.post.promotion=
```

Use the `script.resumed` property to specify an optional path to a user-supplied script to invoke when an agent resumes monitoring a database.

```
#  Absolute path to resume script

#
#  This script is run before an IDLE agent resumes

#  monitoring its local database.

script.resumed=
```

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

```
#  Absolute path to script run after database failure

#  This is an optional user-supplied script that will be run after

#  an agent detects that its local database has failed.

script.db.failure=
```

Use the `script.primary.isolated` property to specify the complete path to an optional user-supplied script that Failover Manager invokes if the agent monitoring the primary database detects that the primary 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 primary

#  This is an optional user-supplied script that will be run after

#  a primary agent detects that it has been isolated from the

#  majority of the efm cluster.

script.primary.isolated=
```

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

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

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

#  before a promotion.

#
#  This optional user-supplied script will be invoked on other

#  agents when a node is about to promote its database. The exit
```

(continues on next page)

(continued from previous page)

```
#  code from this script has no effect on Failover Manager, but

#  will be included in a notification sent after the script

#  executes.

#
#  Pass a parameter (%p) with the script to identify the new

#  primary node address.

#
#  Example:

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

script.remote.pre.promotion=
```

Use the `script.remote.post.promotion` property to specify the path and name of a script to invoke on any non-primary nodes after a promotion occurs.

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

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

#  after a promotion.

#
#  This optional user-supplied script will be invoked on nodes

#  (except the new primary) after a promotion occurs. The exit code

#  from this script has no effect on Failover Manager, but will be

#  included in a notification sent after the script executes.

#
#  Pass a parameter (%p) with the script to identify the new

#  primary node address.

#
#  Example:

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

script.remote.post.promotion=
```

Use the `script.custom.monitor` property to provide the name and location of an optional script to invoke on regular intervals, specified in seconds by the `custom.monitor.interval` property.

Use `custom.monitor.timeout` to specify the maximum time for the script to run. If script execution doesn't finish in the time specified, Failover Manager sends a notification.

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

```
#  Absolute path to a custom monitoring script.

#
#  Use script.custom.monitor to specify the location and name of

#  an optional user-supplied script that will be invoked

#  periodically to perform custom monitoring tasks. A non-zero

#  exit value means that a check has failed; this will be treated

#  as a database failure. On a primary node, script failure will

#  cause a promotion. On a standby node script failure will

#  generate a notification and the agent will become IDLE.

#
#  The custom.monitor.\* properties are required if a custom

#  monitoring script is specified:

#
#  custom.monitor.interval is the time in seconds between executions

#  of the script.

#
#  custom.monitor.timeout is a timeout value in seconds for how

#  long the script will be allowed to run. If script execution

#  exceeds the specified time, the task will be stopped and a

#  notification sent. Subsequent runs will continue.

#
#  If custom.monitor.safe.mode is set to true, non-zero exit codes

#  from the script will be reported but will not cause a promotion

#  or be treated as a database failure. This allows testing of the

#  script without affecting EFM.

#
script.custom.monitor=

custom.monitor.interval=
custom.monitor.timeout=
custom.monitor.safe.mode=
```

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

Use the `sudo.user.command` property to specify a command for Failover Manager to invoke when executing commands performed by the database owner.

```
#  Command to use in place of sudo if desired when efm runs

#  the efm_db_functions or efm_root_functions, or efm_address

#  scripts.

#  Sudo is used in the following ways by efm:

#
#  sudo /usr/edb/efm-<version>/bin/efm_address <arguments>

#  sudo /usr/edb/efm-<version>/bin/efm_root_functions <arguments>

#  sudo -u <db service owner> /usr/edb/efm-<version>/bin/efm_db_functions <arguments>

#
#  sudo in the first two examples will be replaced by the value

#  of the sudo.command property. sudo -u <db service owner> will

#  be replaced by the value of the sudo.user.command property.

#  The %u field will be replaced with the db owner.

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

Use the `lock.dir` property to specify an alternative location for the Failover Manager lock file. The file prevents Failover Manager from starting multiple, potentially orphaned, agents for a single cluster on the node.

```
#  Specify the directory of lock file on the node. Failover

#  Manager creates a file named <cluster>.lock at this location to

#  avoid starting multiple agents for same cluster. If the path

#  does not exist, Failover Manager will attempt to create it. If

#  not specified defaults to /var/lock/efm-<version>

lock.dir=
```

Use the `log.dir` property to specify the location to write agent log files. Failover Manager attempts to create the directory if the directory doesn't exist.

```
#  Specify the directory of agent logs on the node. If the path
```

(continues on next page)

(continued from previous page)

```
#  does not exist, Failover Manager will attempt to create it. If

#  not specified defaults to /var/log/efm-<version>. (To store

#  Failover Manager startup logs in a custom location, modify the

#  path in the service script to point to an existing, writable

#  directory.)

#  If using a custom log directory, you must configure

#  logrotate separately. Use man logrotate for more information.

log.dir=
```

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

```
#  Syslog information. The syslog service must be listening on

#  the port for the given protocol, which can be UDP or TCP.

#  The facilities supported are LOCAL0 through LOCAL7.

syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

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

```
#  Which logging is enabled.

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

For more information about configuring syslog logging, see
Enabling syslog log file entries .

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

```
#  Logging levels for JGroups and EFM.

#  Valid values are: TRACE, DEBUG, INFO, WARN, ERROR

#  Default value: INFO
```

(continues on next page)

(continued from previous page)

```
#  It is not necessary to increase these values unless debugging a

#  specific issue. If nodes are not discovering each other at

#  startup, increasing the jgroups level to DEBUG will show

#  information about the TCP connection attempts that may help

#  diagnose the connection failures.

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

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

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

#  the agent.

jvm.options=-Xmx128m
```

encrypting_database_password

8.2 Encrypting your database password

Failover Manager requires you to encrypt your database password before including it in the cluster properties file. Use the *Using the efm utility* located in the `/usr/edb/efm-4.<x>/bin` directory to encrypt the password. When encrypting a password, you can either pass the password on the command line when you invoke the utility or use the `EFMPASS` environment variable.

To encrypt a password, use the command:

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

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

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

```
export EFMPASS=password
```

If you don't include the `--from-env` option, Failover Manager prompts 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.

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, download and enable a Java cryptography extension (JCE) that provides an unlimited policy for your platform.

This example shows 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 Failover Manager cluster property file:
/etc/edb/efm-4.7/acctg.properties
Please enter the password and hit enter:
Please enter the password again to confirm:
The encrypted password is: 516b36fb8031da17cfbc010f7d09359c
Please paste this into your acctg.properties file
db.password.encrypted=516b36fb8031da17cfbc010f7d09359c
```

Note: The utility notifies you if a properties file doesn't exist.

After receiving your encrypted password, paste the password into the properties file and start the Failover Manager service. If there's a problem with the encrypted password, the Failover Manager service doesn't start:

```
[witness@localhost ~]# systemctl start edb-efm-4.7
Job for edb-efm-4.7.service failed because the control process exited with error code.␣
↪See "systemctl status edb-efm-4.7.service" and "journalctl -xe" for details.
```

If you receive this message when starting the Failover Manager service, see the startup log `/var/log/efm-4.7/startup-efm.log` for more information.

If you are using RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, startup information is also available with the following command:

```
systemctl status edb-efm-4.7
```

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

8.2.1 Using the EFMPASS environment variable

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

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

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

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

#  7ceecd8965fa7a5c330eaa9e43696f83
```

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

8.3 The cluster members file

Each node in a Failover Manager cluster has a cluster members file (by default named `efm.nodes`) that contains a list of the current Failover Manager cluster members. When an agent starts, it uses the file to locate other cluster members. The Failover Manager installer creates a file template for the cluster members file named `efm.nodes.in` in the `/etc/edb/efm-4.<x>` directory.

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

```
cp /etc/edb/efm-4.7/efm.nodes.in /etc/edb/efm-4.7/efm.nodes
```

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

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

By default, Failover Manager expects the cluster members file to be named `efm.nodes` . If you name the cluster members file something other than `efm.nodes` , 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 becomes the membership coordinator. On each subsequent node, the cluster member file must contain the address and port number of at least the membership coordinator. Each entry in the cluster members file must be listed in an address:port format. Separate multiple entries with a space.

The agents 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 doesn't modify the file. Note: an agent doesn't write its own information to the file as it does not need its own location to discover other members at startup.

If the membership coordinator leaves the cluster, another node assumes the role. You can use the *efm cluster-status* command to find the address of the membership coordinator. If a node joins or leaves a cluster while an agent is down, before starting that agent, manually ensure that the file includes at least the current membership coordinator address and port.

If you know the addresses and ports of the nodes that are joining the cluster, you can include the addresses in the cluster members file at any time. At startup, any addresses that don't identify cluster members are ignored unless the `auto.allow.hosts` property in the

> *The cluster properties file* is set to `true` .

If the `stable.nodes.file` property in the *The cluster properties file*) is set to `true` , the agent doesn't update the `.nodes` file when cluster members join or leave the cluster. This behavior is most useful when the IP addresses of cluster members don't change often.

8.4 Extending Failover Manager permissions

During the Failover Manager installation, the installer creates a user named efm. efm doesn't have enough 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.

- When enabling Pgpool integration, efm invokes the `efm_pgpool_functions` 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 efm to manage Postgres clusters owned by other users.

The `efm-42` file is located in `/etc/sudoers.d` and contains the following entries:

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

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

#  If you run your db service under a non-default account, you will need to copy

#  this file to grant the proper permissions and specify the account in your efm

#  cluster properties file by changing the db.service.owner property.

efm    ALL=(postgres)      NOPASSWD:   /usr/edb/efm-4.7/bin/efm_db_functions
efm    ALL=(enterprisedb)  NOPASSWD:   /usr/edb/efm-4.7/bin/efm_db_functions

#  Allow user efm to sudo efm_root_functions as root to write/delete the PID file,

#  validate the db.service.owner property, etc.

efm    ALL=(ALL)           NOPASSWD:   /usr/edb/efm-4.7/bin/efm_root_functions

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

efm    ALL=(ALL)           NOPASSWD:   /usr/edb/efm-4.7/bin/efm_address

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

efm    ALL=(ALL)           NOPASSWD:   /usr/edb/efm-4.7/bin/efm_pgpool_functions

#  relax tty requirement for user efm

Defaults:efm !requiretty
```

If you're using Failover Manager to monitor clusters that are owned by users other than postgres or enterprisedb, make a copy of the `efm-42` file and modify the content to allow the user to access the `efm_functions` script to manage their clusters.

If an agent can't 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)
```

(continues on next page)

(continued from previous page)

```
# includedir /etc/sudoers.d
```

8.4.1 Running Failover Manager without sudo

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

- Install the Failover Manager RPM.
- Perform Failover Manager setup tasks.

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

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

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

This command allows the user to write to `/var/run/efm-4.<x>` and `/var/lock/efm-4.<x>` .

2. If you're reusing a cluster name, remove any previously created log files. The new user can't write to log files created by the default or other owner.
3. Copy the cluster properties template file and the nodes template file:

```
su - enterprisedb

cp /etc/edb/efm-4.7/efm.properties.in <directory/cluster_name>.properties

cp /etc/edb/efm-4.7/efm.nodes.in <directory>/<cluster_name>.nodes
```

Then, modify the cluster properties file, providing the name of the user in the `db.service.owner` property. Also make sure that the `db.service.name` property is blank. Without sudo, you can't run services without root access.

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

```
/usr/edb/efm-4.7/bin/runefm.sh start|stop <directory/cluster_name>.properties
```

Where `<directory/cluster_name.properties>` specifies the full path of the cluster properties file. The user provides the full path to the properties file whenever the nondefault user is controlling agents or using the `efm` script.

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

Failover Manager uses a binary named `manage-vip` that resides in `/usr/edb/efm-4.<x>/bin/secure/` to perform VIP management operations without sudo privileges. This script uses setuid to acquire with the privileges needed to manage virtual IP addresses.

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

For security reasons, we recommend against modifying the access privileges of the `/usr/edb/efm-4.<x>/bin/secure/` directory or the `manage-vip` script.

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

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

8.5 Using Failover Manager with virtual IP addresses

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

Note: Virtual IP addresses aren't supported by many cloud providers. In those environments, use another mechanism, such as an elastic IP address on AWS, that can be changed when needed by a fencing or post-promotion script.

By default, the script resides in:

`/usr/edb/efm-4.<x>/bin/efm_address`

Failover Manager uses the following command variations to assign or release an IPv4 or IPv6 IP address.

To assign a virtual IPv4 IP address:

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

To assign a virtual IPv6 IP address:

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

To release a virtual address:

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

Where:

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

`<IPv4_addr>` or `<IPv6_addr>` matches the value specified in the `virtual.ip` property in the cluster properties file.

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

For more information about properties that describe a virtual IP address, see *The cluster properties file* .

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

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

8.5.1 Testing the VIP

When using a virtual IP (VIP) address with Failover Manager, it's important to test the VIP functionality manually before starting Failover Manager. This catches any network-related issues before they cause a problem during an actual failover. While testing the VIP, make sure that Failover Manager isn't running.

The following steps test the actions that Failover Manager takes. The example uses the following property values:

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

Note: The *virtual.ip.prefix* specifies the number of significant bits in the virtual IP address.

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

1. Ping the VIP from all nodes to confirm that the address isn't 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 see 100% packet loss. !!!important Failover Manager uses the exit code of the ping command to determine whether the address was reachable. In this case, the exit code isn't zero. If you're using a command other than ping, it must return a non-zero exit code if the address isn't reachable.

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

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

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

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

No packet loss occurs. !!!Important Failover Manager uses the exit code of the ping command to determine whether the address was reachable. In this case, the exit code is zero. If you're using a command other than ping, it must return a zero exit code if the address is reachable.

4. Use the `efm_address del` command to release the address on the primary node and confirm the node was released with ip address:

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

The output from this step doesn't show an eth0 interface.

5. Repeat step 3, this time verifying that the standby and witness don't 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
```

100% packet loss occurs. 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's in use.

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

After these test steps, release the VIP from any nonprimary node before attempting to start Failover Manager.

Note: The network interface used for the VIP doesn't have to be the same interface used for the Failover Manager agent's *bind.address* value. The primary agent drops the VIP as needed during a failover, and Failover Manager verifies that the VIP is no longer available before promoting a standby. A failure of the bind address network leads to primary isolation and failover.

If the VIP uses a different interface, you might encounter a timing condition in which the rest of the cluster checks for a reachable VIP before the primary agent drops it. In this case, Failover Manager retries the VIP check for the number of seconds specified in the `node.timeout` property to help ensure that a failover happens as expected.

8.6 Configuring for Eager Failover

In default run mode, if a primary Failover Manager process fails, there's no failover protection until the agent restarts. To avoid this case, you can set up the primary node through `systemd` to cause a failover when the primary agent exits, which is called Eager Failover.

You can set up Eager Failover by performing the following steps. The example uses EDB Postgres Advanced Server version 12 and Failover Manager version 4.5.

8.6.1 Enabling Eager Failover

- Since the database server stops as soon as the Failover Manager agent stops or fails, you must set the following property for all the agents before starting Failover Manager:

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

If you don't set this property before starting Failover Manager, shutting down a Failover Manager agent shuts down the database without failover.

- With Eager Failover enabled, using the `efm stop-cluster` command stops all of the Failover Manager agents and shuts down the primary database. Since the agents aren't running, there's no failover. To avoid thihs scenario, you can disable the command using the `enable.stop.cluster` property.

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

- Ensure that the database server and the local Failover Manager agent are running.
- As root, create `/etc/systemd/system/edb-as-12.service` file and include:

```
.include /lib/systemd/system/edb-as-12.service
[Unit]
BindsTo=edb-efm-4.5.service
```

- Run the following command to reload the configuration files:

```
systemctl daemon-reload
```

With these changes, when the Failover Manager agent is stopped or ended, the rest of the cluster treats this situation as a failure and attempts a failover.

8.6.2 Disabling Eager Failover

- If you want to stop Failover Manager without stopping the database, comment out the following line in `/etc/systemd/system/edb-as-12.service` :

```
BindsTo=edb-efm-4.5.service
```

- Run the following command to reload the configuration files:

```
systemctl daemon-reload
```

8.6.3 Upgrading Failover Manager in Eager Failover mode

To upgrade Failover Manager without stopping EDB Postgres Advanced Server, temporarily disable the Eager Failover mode.

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

8.6.4 Important notes

- Since the `systemd` command isn't used to manage the database while running Failover Manager with a non-sudo setup, Eager Failover is supported only in sudo mode. It isn't supported in a non-sudo mode.
- Eager Failover isn't suitable for situations in which a VIP wouldn't be released by the old primary.
- Eager Failover is suitable in the following situations:
- With the EDB Postgres Advanced Server high-availability setup. - In a setup using client connection failover with jdbc or libpq

  (target-session-attrs) . - When custom scripting triggered by `script.fence`

would fence the old primary server (STONITH). Some examples are to shut down the VM with VMWare vCenter integration, openstack integration, or lights-out management. - When custom scripting triggered by `script.fence` would use ssh to deactivate the VIP.

Note: Setting *check.vip.before.promotion=false* is required to allow the new primary to attach the VIP before the old primary releases it.

- Use care when using `primary.shutdown.as.failure=true` . See the description of the *primary.shutdown.as.failure* property for information on how to safely bring down the database if needed.
- With every failover, a primary ends up being a failed primary, which doesn't automatically recover as an operational standby. Therefore, make sure the cluster contains multiple promotable standbys, and the total number of standbys is at least two more than the value specified for the `minimum.standbys` property. This is a general recommendation, but it becomes more pressing when using Eager Failover.
- If the database server is stopped, restarting the database also starts Failover Manager.

Note:

- If there's a problem starting Failover Manager, such as a bad property value, the database server starts and shuts down again without displaying any warning that it isn't running. - If the Failover Manager process was previously ended, the lock file still exists, and the agent can't restart automatically. - If problems occur when starting the database server or the Failover Manager agent, check the Failover Manager startup log for information.

- As a result of running the `stop-cluster` command, Failover Manager stops on all the nodes. In Eager Failover mode, the `stop-cluster` command also stops EDB Postgres Advanced Server without a failover. Set `enable.stop.cluster=false` to make sure the `stop-cluster` command can't be invoked unintentionally.

CHAPTER

NINE

CONFIGURING SSL AUTHENTICATION ON A FAILOVER MANAGER CLUSTER

You can enable SSL authentication for Failover Manager. All connecting clients are required to use SSL authentication when connecting to any database server in the cluster. You must modify the connection methods currently used by existing clients.

To enable SSL on a Failover Manager cluster:

1. Place a `server.crt` and `server.key` file in the `data` directory under your EDB Postgres 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 .
2. Modify the `postgresql.conf` file on each database in the Failover Manager cluster, enabling SSL:

```
ssl=on
```

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

1. 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 aren't 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 .

1. After placing the `server.crt` and `server.key` files 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 the Postgres JDBC documentation .

1. 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:

edb_notranlate_4 The certificate from each database server must be imported into the trusted certificates file of each agent. The location of the `cacerts` file can vary on each system. For more information, see the Postgres JDBC documentation .

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

CHAPTER

TEN

USING FAILOVER MANAGER

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

If a primary node reboots, Failover Manager might detect the database is down on the primary node and promote a standby node to the role of primary. If this happens, the Failover Manager agent on the rebooted primary node attempts to write a `recovery.conf` file to make sure Postgres doesn't start as a secondary primary. Therefore, you must start the Failover Manager agent before starting the database server. The agent starts in idle mode and checks to see if there is already a primary in the cluster. If there is a primary node, the agent verifies that a `recovery.conf` or `standby.signal` file exists or creates `recovery.conf` , if needed, to prevent the database from starting as a second primary.

10.1 Managing a Failover Manager cluster

Once configured, a Failover Manager cluster requires no regular maintenance. However, you can perform management tasks that a Failover Manager cluster might occasionally require.

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

- `:ref:`efm allow-node<efm allow-node>`
- `:ref:`efm disallow-node<efm disallow-node>`
- `:ref:`efm promote<efm promote>`
- `:ref:`efm resume<efm resume>`
- `:ref:`efm set-priority<efm set-priority>`
- `:ref:`efm stop-cluster<efm stop-cluster>`
- `:ref:`efm upgrade-conf<efm upgrade-conf>`

10.1.1 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/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, assume superuser privileges, and invoke the command:

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

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

If the node is not a dedicated witness node, Failover Manager connects to the local database and invokes the `pg_is_in_recovery()` function. If the server responds `false` , the agent assumes the node is a primary 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 doesn't respond, the agent starts 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 can't connect, the agent shuts down.

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

Note: If you are running */var/lock* or */var/run* on *tmpfs* (Temporary File System), make sure that the systemd service file for Failover Manager has a dependency on *systemd-tmpfiles-setup.service* .

10.1.2 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.

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

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

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

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

2. Configure the cluster members file on the new node, adding an entry for the membership coordinator. For more information about modifying the cluster members file, see *The cluster members file* .

3. Assume superuser privileges on the new node, and start the Failover Manager agent. To start the Failover Manager cluster on RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, invoke the command:

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

When the new node joins the cluster, Failover Manager sends a notification to the administrator email provided in the `user.email` property, and invokes the specified notification script.

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

10.1.3 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` is the primary standby `(1)` :

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

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

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

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

In the event of a failover, Failover Manager first retrieves 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 is promoted to primary unless *use.replay.tiebreaker* is set to `true` . To check the priority value of your standby nodes, use the command:

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

Note: The promotion priority for nodes changes when a new primary is promoted.

If the `efm set-priority` command was used to change whether a standby is promotable, it may be reset to the value in the standby's properties file through promotion or cluster splits and rejoins. If the agent is restarted, the promotable status reverts to the value in the properties file.

10.1.4 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 primary database.

Perform manual promotion only during a maintenance window for your database cluster. If you don't have an up-to-date standby database available, you are prompted before continuing. To start a manual promotion, assume the identity of efm or the OS superuser, and invoke the command:

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

Where:

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

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

Include the `-sourcenode` keyword to specify the node from which to copy the recovery settings to the primary.

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

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

During switchover:

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

During a promotion, the primary agent releases the virtual IP address. If it isn't a switchover, a `recovery.conf` file is created in the directory specified by the `db.data.dir` property. The `recovery.conf` file is used to prevent the old primary database from starting until the file is removed, preventing the node from starting as a second primary in the cluster. If the promotion is part of a switchover, recovery settings are handled as described above.

The primary 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 isn't isolated from the network. The standby agent runs the fencing script and promotes the standby database to primary. The standby agent then assigns the virtual IP address to the standby node and runs the post-promotion script (if applicable).

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 primary, place the node first in the promotion list:

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

Then, perform a manual promotion:

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

For more information about the efm utility, see *Using the efm utility* .

10.1.5 Stopping a Failover Manager agent

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

To stop the Failover Manager agent on RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, assume superuser privileges and invoke the command:

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

Until you invoke the `efm disallow-node` command (removing the node's address from the Allowed Node host list), you can use the `service edb-efm-4.<x> start` command to restart the node later without first running the `efm allow-node` command again.

Stopping an agent doesn't signal the cluster that the agent has failed unless the *primary.shutdown.as.failure* property is set to `true` .

10.1.6 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 causes all Failover Manager agents to exit. Terminating the Failover Manager agents completely disables all failover functionality.

Note: When you invoke the *efm stop-cluster* command, all authorized node information is lost from the Allowed Node host list.

10.1.7 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. Then invoke the `efm disallow-node` command, specifying the cluster name and the IP address of the node:

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

The `efm disallow-node` command doesn't stop a running agent. The service continues to run on the node until you stop the agent . If the agent or cluster is later stopped, the node isn't allowed to rejoin the cluster and is removed from the failover priority list. It becomes 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.

10.2 Running multiple agents on a single node

You can monitor multiple database clusters that reside on the same host by running multiple primary or standby agents on that Failover Manager node. You can 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 don't interfere with each other:

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 unit file (on a RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x system) for each cluster to specify the names of the cluster properties and the cluster members files.
4. Start the services for each cluster.

These examples use 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. This example creates `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.data.dir`

`virtual.ip` (if used)

`db.service.name` (if used)

In each cluster properties file, the `db.port` parameter specifies a unique value for each cluster. The `db.user` and `db.database` parameter can have the same value or a unique value. For example, the `acctg.properties` file can specify:

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

While the `sales.properties` file can 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 can work, but it's easier to keep the information clear if using ports that aren't too close to each other.

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

Use the following parameters when assigning the virtual IP address to a node. If your Failover Manager cluster doesn't use a virtual IP address, leave these parameters blank.

`virtual.ip`

`virtual.ip.interface`

`virtual.ip.prefix`

This parameter value is determined by the virtual IP addresses being used and can be the same for both `acctg.properties` and `sales.properties` .

After creating the `acctg.properties` and `sales.properties` files, create a service script or unit file for each cluster that points to the respective property files. This step is platform specific. If you're using RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, see

RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x .

Note: If you're using a unit file, manually update the file to reflect the new service name when you upgrade Failover Manager.

10.2.1 RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x

If you're using RHEL/CentOS 7.x or RHEL/Rocky Linux/AlmaLinux 8.x, copy the `edb-efm-4.<x>` unit file to a new file with a name that is unique for each cluster. For example, if you have two clusters named acctg and sales, the unit file names might be:

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

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

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

```
Environment=CLUSTER=acctg
```

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

```
PIDFile=/var/run/efm-4.7/acctg.pid
```

After copying the service scripts, 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, to start the `acctg` agent:

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

For information about customizing a unit file, see *Understanding and administering systemd* .

CHAPTER

ELEVEN

USING THE EFM UTILITY

Failover Manager provides the `efm` utility to assist with cluster management. The RPM installer adds the utility to the `/usr/edb/efm-4.<x>/bin` directory when you install Failover Manager.

11.1 efm allow-node

```
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.

11.2 efm disallow-node

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

11.3 efm cluster-status

```
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 status report, see *Monitoring a Failover Manager cluster* .

11.4 efm cluster-status-json

```
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 from 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 three nodes:

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

11.5 efm encrypt

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

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

11.6 efm promote

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

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

Attempt a manual promotion only during a maintenance window for your database cluster and if the status command reports that all standbys in the cluster are up to date with the primary.

Include the `-switchover` clause to promote a standby node and reconfigure a primary node as a standby node. Include the `-sourcenode` keyword, and specify a node address to indicate the node whose recovery settings to copy to the old primary node, which makes it a standby. Include the `-quiet` keyword to suppress notifications during the switchover process. Include the `-noscripts` keyword to instruct Failover Manager not to invoke fencing or post-promotion scripts.

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

Note: This command instructs the service to ignore the value specified in the *auto.failover* parameter in the cluster properties file.

11.7 efm resume

`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.

11.8 efm set-priority

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

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

Use the priority option to specify the place for the node in the priority list. For example, specify a value of `1` to indicate that the node is the primary standby and will be the first node promoted in the event of a failover. A priority value of `0` instructs Failover Manager not to promote the standby.

11.9 efm stop-cluster

`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's no failover protection in place.

Note: When you invoke the *efm stop-cluster* command, all authorized node information is removed from the Allowed Node host list.

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

11.10 efm upgrade-conf

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

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

If you're upgrading from a Failover Manager configuration that doesn't use sudo, include the `-source` flag and specify the name of the directory in which the configuration files reside when invoking `upgrade-conf` .

11.11 efm node-status-json

`efm node-status-json <cluster_name>`

Invoke the `efm node-status-json` command to display the status of a local node in JSON format. A successful execution of this command returns `0` as its exit code. In case of a database failure or an agent status becoming IDLE, the command returns `1` as exit code.

The following is an example output of the `efm node-status-json` command:

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

11.12 efm –help

`efm --help`

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

CHAPTER

TWELVE

MONITORING A FAILOVER MANAGER CLUSTER

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

12.1 Reviewing the cluster status report

The `efm cluster-status` *The cluster properties file* command returns a report that contains information about the status of the Failover Manager cluster:

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

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

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

Allowed node host list:
172.19.14.9 172.19.12.163 172.19.10.2

Membership coordinator: 172.19.14.9

Standby priority host list:
172.19.12.163 172.19.10.2

Promote Status:

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

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

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

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

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

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

The `Allowed node host list` and `Standby priority host list` provide an easy way to see the nodes that can join the cluster and the promotion order of the nodes. The IP address of the membership coordinator is also displayed in the report:

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

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. Because the queries to each database return at different times, the LSNs might not match even if streaming replication is working normally for the cluster. To get the latest view of replication, connect to the primary database, and execute SQL command `SELECT * FROM pg_stat_replication;` .

```
Promote Status:

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

If a database is down or if the database was restarted, but the resume command was not yet invoked, the state of the agent that resides on that host is idle. If an agent is idle, the cluster status report includes a summary of the condition of the idle node. For example:

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

12.1.1 Exit codes

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

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

A database is down, unknown, or has an idle agent.

Failover Manager can't decrypt the provided database password.

There's a problem contacting the databases to get WAL locations.

There's no primary agent.

There are no standby agents.

One or more standby nodes aren't in sync with the primary.

12.2 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 interface) to display the state of a primary or standby node that is part of a Streaming Replication scenario.

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

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

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

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

- Streaming Replication
- WAL Archive Status

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

- Streaming Replication Lag Time

For more information about PEM, see the Postgres Enterprise Manager documentation .

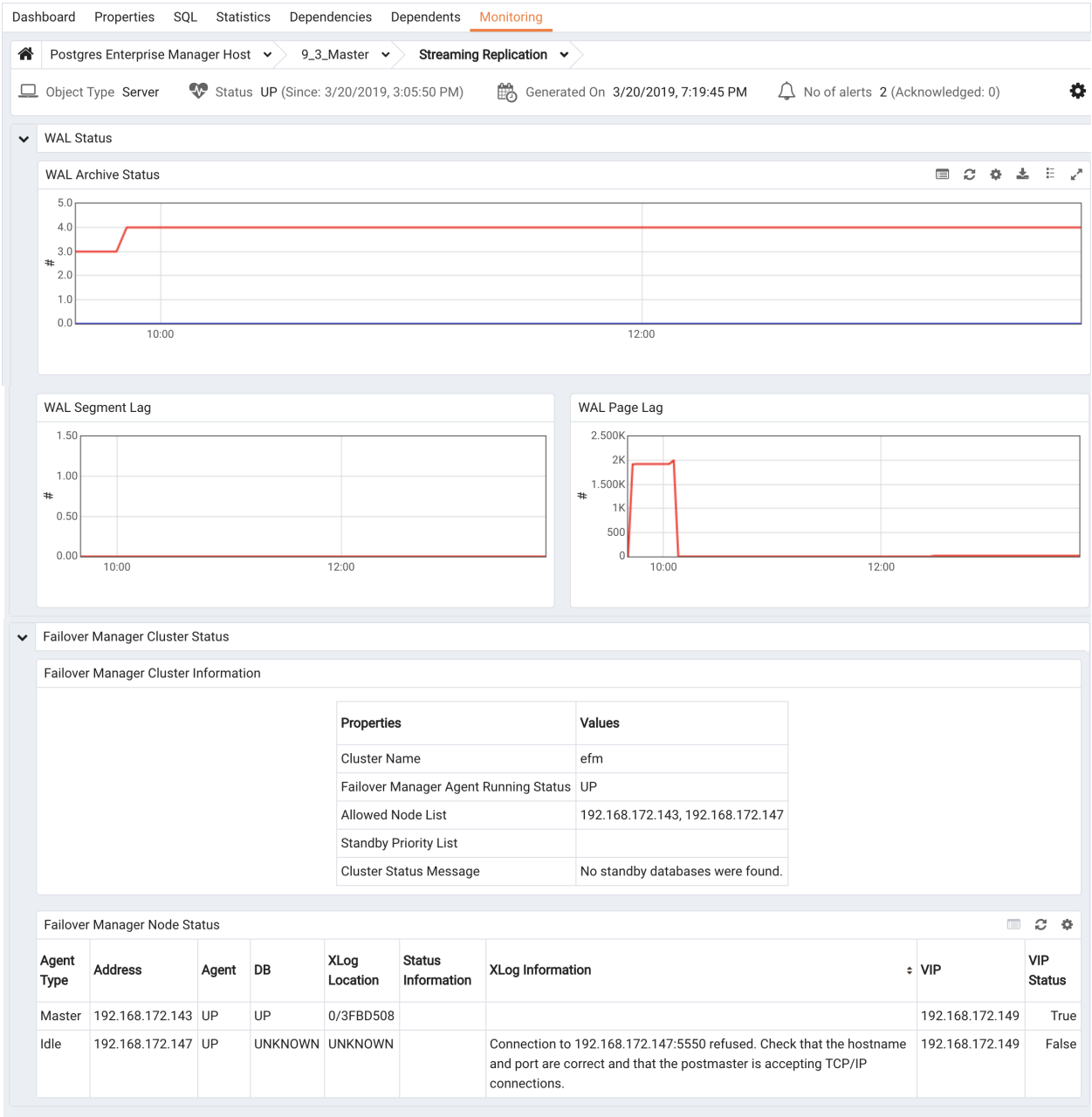

Fig. 1: The Streaming Replication dashboard (Primary node)

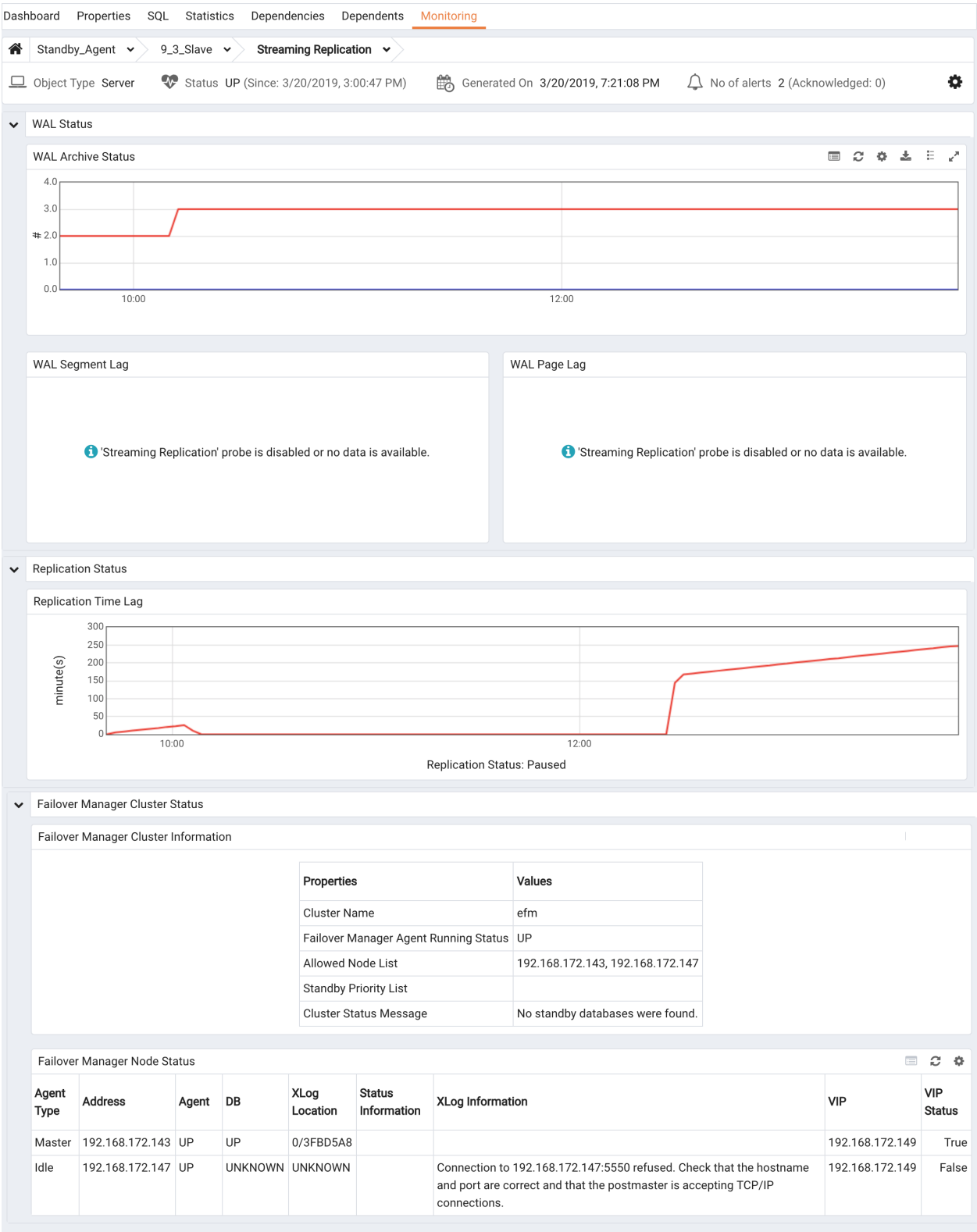

Fig. 2: The Streaming Replication dashboard (Standby node)

CHAPTER

THIRTEEN

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.

If you're running multiple clusters on a single node, you 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.

13.1 Using the systemctl utility on RHEL/CentOS 7.x and RHEL/Rocky Linux/AlmaLinux 8.x

On RHEL/CentOS 7.x and RHEL/Rocky Linux/AlmaLinux 8.x, Failover Manager runs as a Linux service named (by default) `edb-efm-4.<x>.service` that is located in `/usr/lib/systemd/system` . Each database cluster monitored by Failover Manager runs 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/CentOS 7.x and RHEL/Rocky Linux/AlmaLinux 8.x host:

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

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

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

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

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 edb-efm-4.7
  edb-efm-4.7.service - EnterpriseDB Failover Manager 4.7
   Loaded: loaded (/usr/lib/systemd/system/edb-efm-4.7.service; disabled; vendor preset:␣
↪disabled)
   Active: active (running) since Wed 2013-02-14 14:02:16 EST; 4s ago
```

(continues on next page)

(continued from previous page)

```
 Process: 58125 ExecStart=/bin/bash -c /usr/edb/edb-efm-4.7/bin/runefm.sh start $
↪{CLUSTER} (code=exited, status=0/SUCCESS)
Main PID: 58180 (java)
  CGroup: /system.slice/edb-efm-4.7.service
          └─58180 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/jre/bin/
↪java -cp /usr/edb/edb-efm-4.7/lib/EFM-4.7.jar -Xmx128m...
```

CHAPTER

FOURTEEN

CONTROLLING LOGGING

Failover Manager writes and stores one log file per agent and one startup log per agent in `/var/log/<cluster_name>-4` .

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

```
#  Logging levels for JGroups and EFM.

#  Valid values are: TRACE, DEBUG, INFO, WARN, ERROR

#  Default value: INFO

#  It is not necessary to increase these values unless debugging a

#  specific issue. If nodes are not discovering each other at

#  startup, increasing the jgroups level to DEBUG will show

#  information about the TCP connection attempts that may help

#  diagnose the connection failures.

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

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

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

For example, if you set the `efm.loglevel` parameter to `WARN` , Failover Manager only logs messages at the `WARN` level and above, that is, `WARN` and `ERROR` .

By default, Failover Manager log files are rotated daily, compressed, and stored for a week. You can modify the file rotation schedule by changing settings in the log rotation file (`/etc/logrotate.d/efm-4.<x>`). For more information about modifying the log rotation schedule, consult the logrotate man page:

```
$ man logrotate
```

14.1 Enabling syslog log file entries

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

To allow a connection to syslog, edit the `/etc/rsyslog.conf` file and uncomment the protocol you want to use. Also make sure that the `UDPServerRun` or `TCPServerRun` entry associated with the protocol includes the port number to which log entries are sent. For example, the following configuration file entries enable UDP connections to port 514:

```
#  Provides UDP syslog reception

$ModLoad imudp
$UDPServerRun 514
```

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

```
#  Provides TCP syslog reception

$ModLoad imtcp
$InputTCPServerRun 514
```

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

```
systemctl restart rsyslog.service
```

After modifying the `rsyslog.conf` file on the Failover Manager host, modify the Failover Manager properties to enable logging. Use your choice of editor to *modify the properties file* `/etc/edb/efm-4.<x>/efm.properties.in` , specifying the type of logging that you want to implement:

```
#  Which logging is enabled.

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

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

```
#  Syslog information. The syslog service must be listening # on the

 port for the given protocol, which can be UDP or
#  TCP. The facilities supported are LOCAL0 through LOCAL7.

#  syslog.host=localhost

syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
```

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

```
syslog man
```

CHAPTER

FIFTEEN

NOTIFICATIONS

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

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

For more information about editing the configuration properties, see
Specifying cluster properties .

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

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

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

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

- `INFO` indicates an informational message about the agent and doesn't require any manual intervention (for example, Failover Manager has started or stopped). See List of INFO level notifications
- `WARNING` indicates that an event has happened that requires the administrator to check on the system (for example, failover has occurred). See List of WARNING level notifications
- `SEVERE` indicates that a serious event has happened and requires the immediate attention of the administrator (for example, failover was attempted but can't complete). See List of SEVERE level notifications

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` calls your attention to operational information about your cluster that doesn't require user action. Notification severity levels aren't related to logging levels. All notifications are sent regardless of the log level detail specified in the configuration file.

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

Note: In addition to sending notices to the administrative email address, all notifications are recorded in the agent log

file (*/var/log/efm-4.<x>/<*cluster name*>.log*).

The conditions listed in this table trigger an INFO level notification:

Subject	Description	Note
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 remote pre-promotion script	Executed remote pre-promotion script *script_name* Results: *script_results*	
Executed remote post-promotion script	Executed remote post-promotion script *script_name* Results: *script_results*	
Executed post-database failure script	Executed post-database failure script *script_name* Results: *script_results*	
Executed primary isolation script	Executed primary isolation script *script_name* Results: *script_results*	
Witness agent running on *node_address* for cluster *cluster_name*	Witness agent is running.	
Primary agent running on *node_address* for cluster *cluster_name*	Primary 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*	
Starting auto resume check for cluster *cluster_name*	The agent on this node checks every *auto.resume.period* seconds to see if it can resume monitoring the failed database. Check the cluster during this time and stop the agent if the database won't be started again. See the agent log for more details.	
Executed agent resumed script	Executed agent resumed script script_name Results: script_results	
WAL logs backed up during promotion	When reconfiguring this standby to follow the new primary, the pg_wal contents were backed up in the pgdata directory. Remove this backup when convenient to free up disk space.	

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

Subject	Description	Notes
Witness agent exited on *node_address* for cluster *cluster_name*	Witness agent has exited.	
Primary agent exited on *node_address* for cluster *cluster_name*	Database health is not being monitored.	
Cluster *cluster_name* notified that primary agent has left	Failover is disabled for the cluster until the primary 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.	
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.	
Virtual IP address assigned to non-primary node	The virtual IP address appears to be assigned to a non-primary node. To avoid any conflicts, Failover Manager will release the VIP. Confirm that the VIP is assigned to your primary node and manually reassign the address if it isn't.	
Virtual IP address not assigned to primary node.	The virtual IP address appears not to be assigned to a primary node. Failover Manager will attempt to reacquire the VIP.	
Standby failed in 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 can't 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. Add witness or agent node to the cluster.	
Subset of cluster *cluster name* disconnected from primary	This node is no longer connected to the majority of the cluster *clustername* . 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* .	

continues on next page

Table 2 – continued from previous page

Subject	Description	Notes
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.	
Failover Manager tried to promote, but primary DB is still running	Failover Manager has started promotion steps, but detected that the primary DB is still running on *address* . This usually indicates that the primary Failover Manager agent has exited. Failover has not occurred. There is no failover protection until the primary agent is restarted.	
Primary agent missing for cluster *cluster_name*	The primary agent has previously left the cluster. Until a primary agent joins the cluster, there is no failover protection.	Available in Failover Manager 4.2 and later.
Standby agent started to promote, but primary has rejoined.	The standby Failover Manager agent started to promote itself but found that a primary agent has rejoined the cluster. Failover has not occurred.	
Standby agent tried to promote, but could not verify primary DB	The standby Failover Manager agent tried to promote itself but could not detect whether the primary DB is still running on *node_address* . Failover has not occurred.	Description applicable for Failover Manager 4.2 and later.
Standby agent tried to promote, but could not verify primary DB	The standby Failover Manager agent tried to promote itself but detected that the primary DB is still running on *node_address* . This usually indicates that the primary Failover Manager agent has exited. Failover has not occurred.	Description applicable for Failover Manager 4.1 and earlier.
Standby agent tried to promote, but VIP appears to still be assigned	The standby Failover Manager agent tried to promote itself but couldn't because the virtual IP address (VIP_address) appears to still be assigned to another node. Promoting under these circumstances can cause data corruption. Failover has not occurred.	
Standby agent tried to promote, but appears to be orphaned	The standby Failover Manager agent tried to promote itself but couldn't because the well-known server (server_address) couldn't be reached. This usually indicates a network issue that has separated the standby agent from the other agents. Failover has not occurred.	
Potential manual failover required on cluster *cluster_name*.	A potential failover situation was detected for cluster *cluster_name* . Automatic failover was disabled for this cluster, so manual intervention is required.	
Failover has completed on cluster *cluster_name*	Failover has completed on cluster *cluster_name* .	

continues on next page

Table 2 – continued from previous page

Subject	Description	Notes
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. Restore this file to prevent accidentally starting another agent for cluster.	
A recovery file for cluster *cluster_name* has been found on primary node	A recovery file for cluster *cluster_name* hwas found at: path_name on primary node *node_address* . This can cause a problem if you attempt to restart the DB on this node.	
recovery_target_timeline is not set to latest in recovery settings	The recovery_target_timeline parameter isn't set to latest in the recovery settings. The standby server can't follow a timeline change that occurs when a new primary is promoted.	
Promotion has not occurred for cluster *cluster_name*	A promotion was attempted but there is already a node being promoted: ip_address.	
Standby will not be reconfigured after failover in cluster **cluster_name**	The 'auto.reconfigure' property has been set to false for this node, so it won't be reconfigured to follow the new primary node after a promotion.	Subject and Description applicable to Failover Manager 4.2 and later.
Standby not reconfigured after failover in cluster **cluster_name**	The *auto.reconfigure* property was set to false for this node. The node was not reconfigured to follow the new primary node after a failover.	Subject and Description applicable to Failover Manager 4.1 and earlier.
Could not resume replay for standby standby_id.	Couldn't resume replay for standby. Manual intervention might 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 might assume that the database has failed though other agents can connect. While this doesn't cause a failover, it might force the local agent to stop monitoring, leaving you without failover protection.	
No standbys available for promotion in cluster *cluster_name*	The current number of standby nodes in the cluster has dropped to the minimum number: number. There can't be a failover unless another standby node is added or made promotable.	
No promotable standby for cluster *cluster_name*	The current failover priority list in the cluster is empty. You have removed the only promotable standby for the cluster *cluster_name* . There cannot be a failover unless another promotable standby node(s) is added or made promotable by adding to failover priority list.	
Synchronous replication has been reconfigured for cluster *cluster_name*	The number of synchronous standby nodes in the cluster has dropped below number. The synchronous standby names on primary was reconfigured to: new synchronous_standby_names value.	
Synchronous replication has been disabled for cluster *cluster_name*.	The number of synchronous standby nodes in the cluster has dropped below count. The primary was taken out of synchronous replication mode.	
Could not reload database configuration.	Couldn't reload database configuration. Manual intervention is required. Error: error_message.	

continues on next page

Table 2 – continued from previous page

Subject	Description	Notes
Custom monitor timeout for cluster *cluster_name*	The following custom monitoring script has timed out: script_name	
Custom monitor 'safe mode' failure for cluster *cluster_name*	The following custom monitor script has failed, but is being run in "safe mode": script_name. Output: script_results	
primary.shutdown.as.failure set to true for primary node	The primary.shutdown.as.failure property has been set to true for this cluster. Stopping the primary agent without stopping the entire cluster is treated by the rest of the cluster as an immediate primary agent failure. If maintenance is required on the primary database, shut down the primary agent and wait for a notification from the remaining nodes that failover will not happen.	
Primary_or_Standby cannot ping local database for cluster *cluster_name*	The Primary_or_Standby agent can no longer reach the local database running at *node_address* . Other nodes are able to access the database remotely, so the agent becomes IDLE and attempts to resume monitoring the database.	Subject and Description applicable to Failover Manager 4.1 and later.
PPrimary cannot ping local database for cluster cluster_name	The Primary agent can no longer reach the local database running at node_address. Other nodes can access the database remotely, so the primary becomes IDLE and attempts to resume monitoring the database.	Subject and Description applicable to Failover Manager 4.0.
Standby cannot resume monitoring local database for cluster *cluster_name*	The standby agent can no longer reach the local database running at *node_address* . Other nodes can access the database remotely. The standby agent remains IDLE until the *resume* command is run to resume monitoring the database.	Available in Failover Manager 4.1 and later.
Primary agent left the cluster and node detached from load balancer	The standby agent can no longer reach the local database running at *node_address* . Other nodes can to access the database remotely. The standby agent remains IDLE until the *resume* command is run to resume monitoring the database.	Available in Failover Manager 4.2 and later.
Standby database in cluster *cluster_name* not stopped before primary is promoted	The *standby.restart.delay* property is set for this agent, so it is not reconfigured to follow the new primary until num_seconds seconds after promotion has completed. In some cases it might not be able to follow the new primary without manual intervention.	Available in Failover Manager 4.2 and later.

The conditions listed in this table trigger a SEVERE notification:

Subject	Description	Notes
Standby database restarted but Failover Manager cannot connect	The start or restart command for the database ran successfully but the database is not accepting connections. Failover Manager will keep trying to connect for up to restart.connection.timeout seconds.	
Unable to connect to DB on *node_address*	The maximum connections limit was reached.	

continues on next page

Table 3 – continued from previous page

Subject	Description	Notes
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.	
Primary cannot resume monitoring local database for cluster *cluster_name*	The primary agent can no longer reach the local database running at *node_address* . Other nodes are able to access the database remotely, so the primary does not release the VIP or create a *recovery.conf* file. The primary agent remains IDLE until the *resume* command runs to resume monitoring the database.	
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	
Remote post-promotion script failed	Remote post-promotion script script_name failed to execute successfully Exit Value: exit_code Results: script_resultsNode: *node_address*	
Remote pre-promotion script failed	Remote pre-promotion script script_name failed to execute successfully Exit Value: exit_code Results: script_resultsNode: *node_address*	
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	
Primary isolation script failed	Primary isolation script script_name failed to execute successfully. Exit Value: exit_code Results: script_results	
Could not promote standby	The promote command failed on node. Couldn't promote standby. Error details: error_details	
Error creating recovery.conf file on *node_address* for cluster *cluster_name*	There was an error creating the recovery.conf file on primary node *node_address* during promotion. Promotion has continued but requires manual intervention to ensure that the old primary node can't be restarted. Error details: message_details	
An unexpected error has occurred for cluster *cluster_name*	An unexpected error has occurred on this node. Check the agent log for more information. Error: error_details	
Primary database being fenced off for cluster *cluster_name*	The primary database has been isolated from the majority of the cluster. The cluster is telling the primary agent at ip_address to fence off the primary database to prevent two primaries when the rest of the failover manager cluster promotes a standby.	
Isolated primary database shutdown.	The isolated primary database has been shutdown by failover manager.	

continues on next page

Table 3 – continued from previous page

Subject	Description	Notes
Primary database being fenced off for cluster *cluster_name*	The primary database was isolated from the majority of the cluster. Before the primary 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 primary database.	
Could not assign VIP to node *node_address*	Failover manager couldn't assign the VIP address for some reason.	
primary_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 successfully pinged the database and resumed monitoring. However, check the node to make sure it is performing normally to prevent a possible database or agent failure.	
Resume timed out for cluster *cluster_name*	This agent couldn't 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 didn't match the actual state of the cluster members. This is rare and can be caused by a timing issue of nodes joining the cluster or changing their state. The problem should be resolved, but you 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 primary database is no longer available in cluster *cluster_name* , but there are no standby nodes available for failover.	
Failover has not occurred	An agent has detected that the primary 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 now becomes IDLE. Manual intervention is required.	
Database in wrong state on *node_address*	The primary agent has detected that the local database is in recovery. The agent now becomes 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 might not work properly because this node can't check whether the database is running.	
Standby custom monitor failure for cluster *cluster_name*	The following custom monitor script has failed on a standby node. The agent will stop monitoring the local database. Script location: script_name Script output: script_results	
Primary custom monitor failure for cluster *cluster_name*	The following custom monitor script has failed on a primary node. Failover Manager will attempt to promote a standby. Script location: script_name Script output: script_results	

continues on next page

Table 3 – continued from previous page

Subject	Description	Notes
Loopback address set for ping.server.ip	Loopback address is set for ping.server.ip property. This setting can interfere with the network isolation detection and hence you should change it.	
Load balancer attach script error	Load balancer attach script script_name failed to execute successfully. Exit Value: exit_code Results: script_results	
Load balancer detach script error	Load balancer detach script script_name failed to execute successfully. Exit Value: exit_code Results: script_results	
Pgpool attach node error	Failover Manager failed to attach pgpool node. Exit Value: exit_code. Results: script_results	Available in Failover Manager 4.1 and later.
Pgpool detach node error	Failover Manager failed to detach pgpool node. Exit Value: exit_code. Results: script_results	
Not enough synchronous standbys available in cluster *cluster_name*	The number of synchronous standby nodes in the cluster has dropped to count. All write queries on the primary are blocked until enough synchronous standby nodes are added.	

CHAPTER

SIXTEEN

SUPPORTED FAILOVER AND FAILURE SCENARIOS

Failover Manager monitors a cluster for failures that might result in failover.

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

- If the primary database crashes or is shut down.
- If the node hosting the primary database crashes or becomes unreachable.

Failover Manager makes every attempt to verify the accuracy of these conditions. If agents can't confirm that the primary database or node has failed, Failover Manager doesn't perform any failover actions on the cluster.

Failover Manager also supports a *noauto-failover* mode for situations in which you want Failover Manager to monitor and detect failover conditions but not perform an automatic failover to a standby. In this mode, a notification is sent to the administrator when failover conditions are met. To disable automatic failover, modify the cluster properties file, setting the *auto.failover* parameter to `false` .

Failover Manager alerts an administrator to situations that require administrator intervention but that don't merit promoting a standby database to primary.

16.1 Primary database is down

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

If the agent on the primary node detects that the primary database has failed, all agents attempt to connect directly to the primary database. If an agent can connect to the database, Failover Manager sends a notification about the state of the primary node. If no agent can connect, the primary 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 primary database, and assigns the virtual IP address to the standby node. Any additional standby nodes are configured to replicate from the new primary unless `auto.reconfigure` is set to `false` . If applicable, the agent runs a post-promotion script.

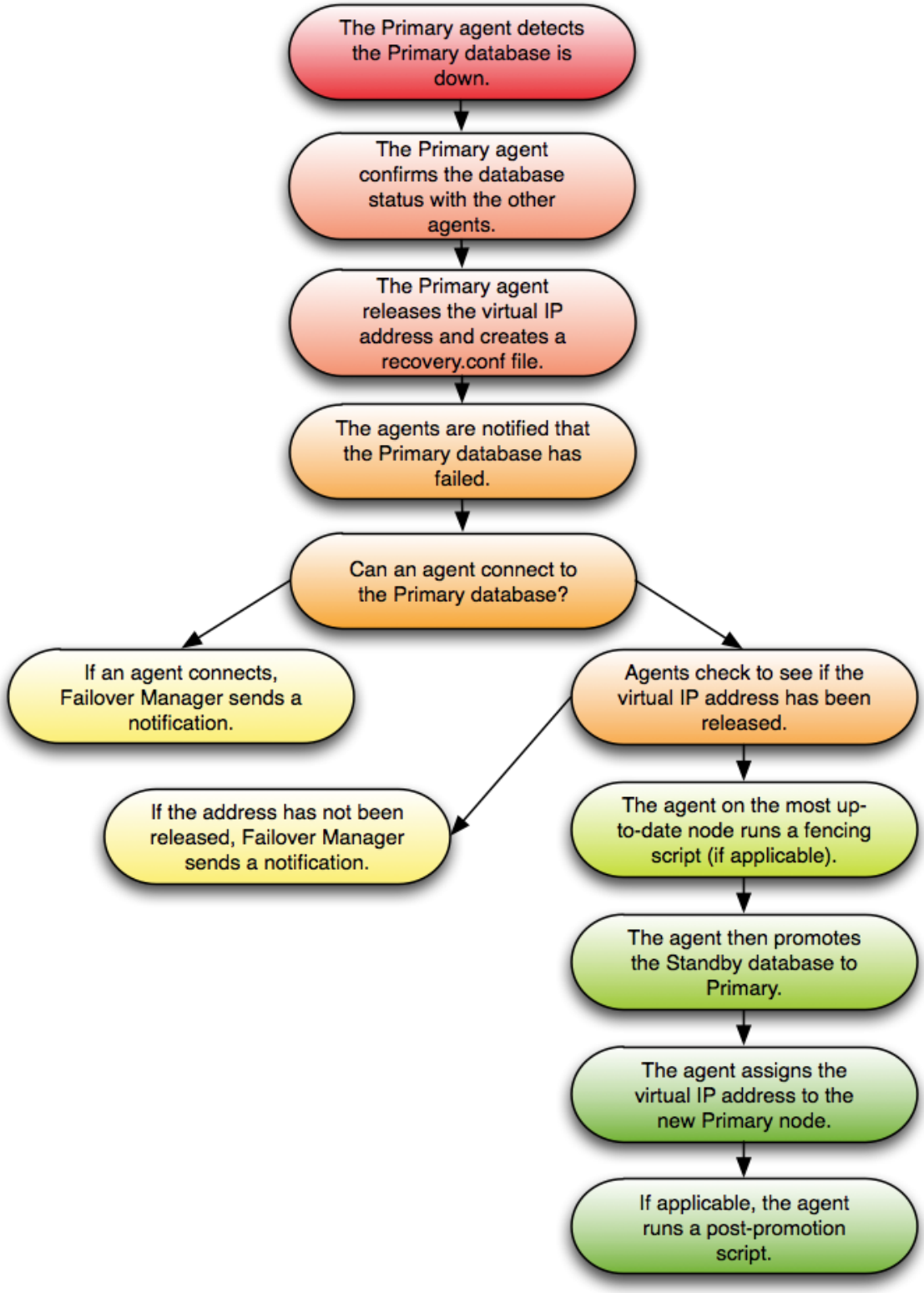

Fig. 1: Confirming the failure of the primary database.

16.2 Return the node to the cluster

To recover from this scenario without restarting the entire cluster:

1. Restart the database on the original primary node as a standby database.
2. Invoke the `efm resume` command on the original primary node.

16.3 Return the node to the role of primary

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

1. If the cluster has more than one standby node, use the `efm set-priority` 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 primary node.

Note: Failover Manager doesn't rebuild a failed primary database to become a standby. Before rebuilding, it's important to determine why the primary failed, and ensure that all the data is available on the new primary. Once the server is ready to be reinstated as a standby, the old data directory can be removed and the server can be reinstated. For more information, refer to the PostgreSQL documentation on Setting up a standby server . In some cases, you can also reinstate the server using pg_rewind .

16.4 Standby database is down

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

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

16.5 Primary agent exits or node fails

If the Failover Manager primary agent crashes or the node fails, a standby agent detects the failure and, if appropriate, initiates a failover.

If an agent detects that the primary agent has left, all agents attempt to connect directly to the primary database. If any agent can connect to the database, an agent sends a notification about the failure of the primary agent. If no agent can connect, the agents attempt to ping the virtual IP address (if applicable) to determine if it was 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 primary 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 primary unless `auto.reconfigure` is set to `false` .

If this scenario occurred because the primary was isolated from network, the primary agent detects the isolation, releases the virtual IP address, and creates the `recovery.conf` file. Failover Manager performs these same steps on the remaining nodes of the cluster.

To recover from this scenario without restarting the entire cluster:

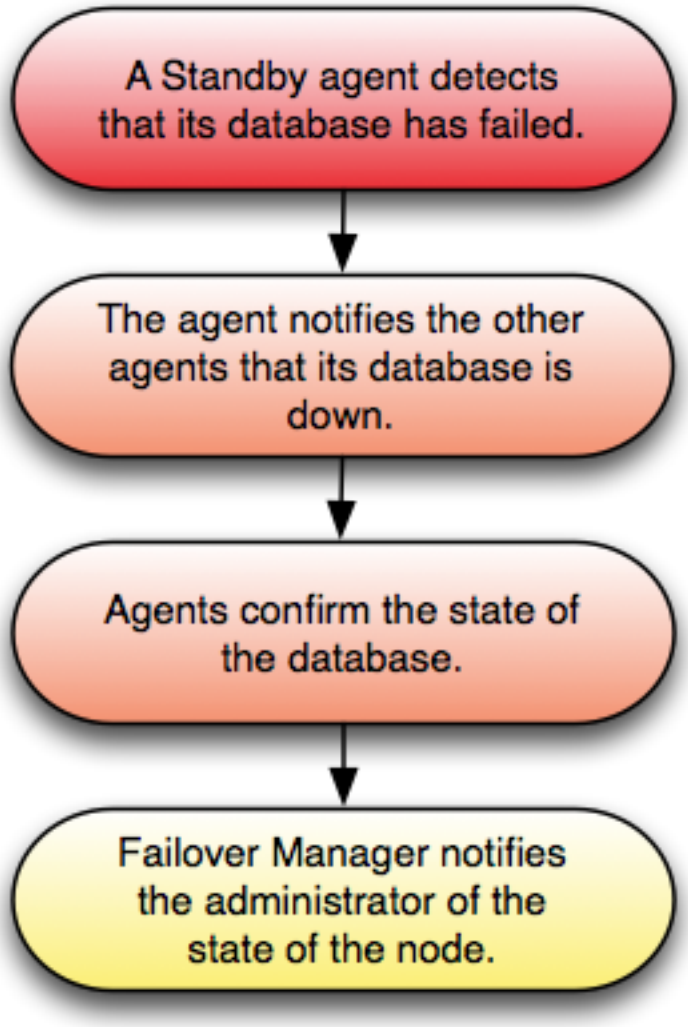

Fig. 2: Confirming the failure of a standby database.

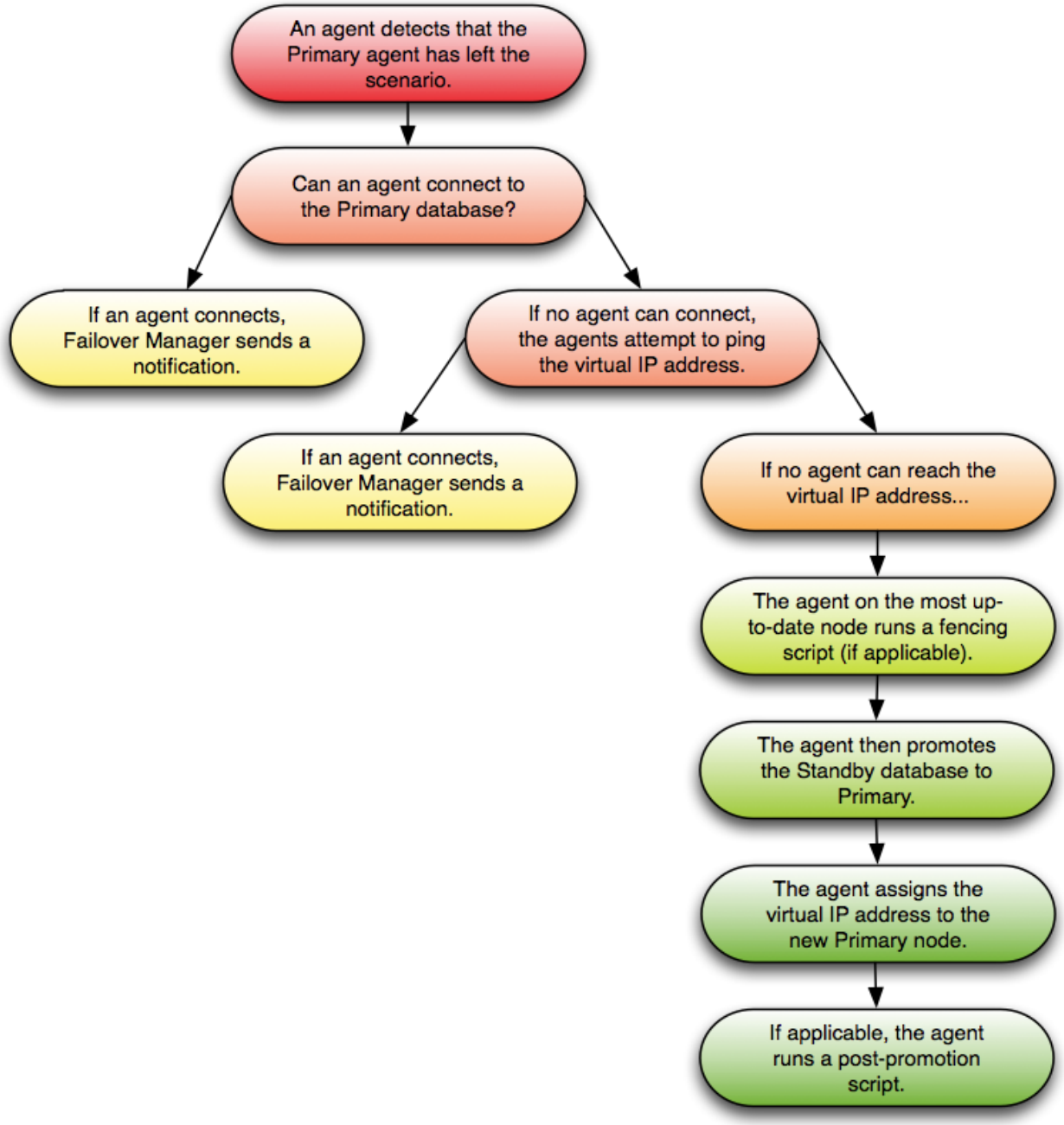

Fig. 3: Confirming the failure of the primary agent.

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

If a primary Failover Manager process fails, there's no failover protection until the agent is restarted. To avoid this case, you can set up the primary node through systemd to cause a failover when the primary agent exits. For more information, see *Configuring for Eager Failover* .

16.6 Standby agent exits or node fails

If a standby agent exits or a standby node fails, the other agents detect that it's no longer connected to the cluster.

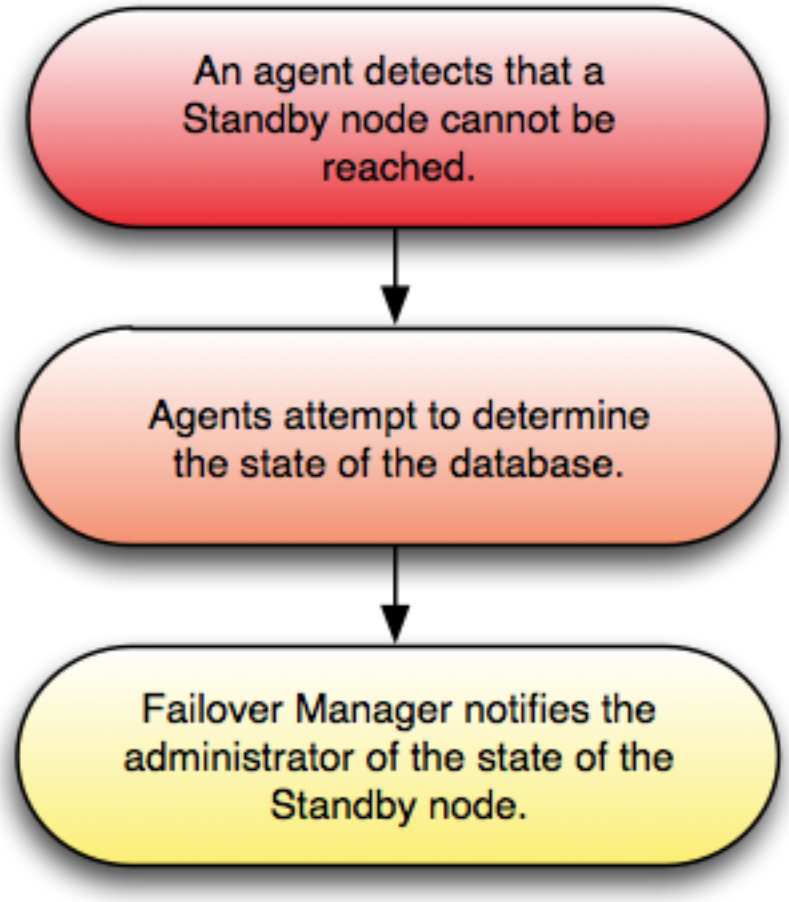

Fig. 4: Failure of standby agent.

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

If there is only one primary and one standby remaining, there's no failover protection in the case of a primary node failure. In the case of a primary database failure, the primary and standby agents can agree that the database failed and

proceed with failover.

16.7 Dedicated witness agent exits/node fails

This scenario details the actions taken if a dedicated witness (a node that is not hosting a database) fails.

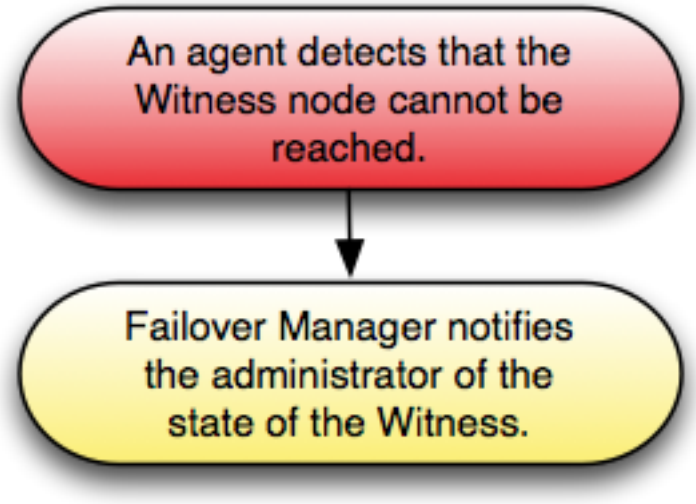

Fig. 5: Confirming the failure of a dedicated witness.

When an agent detects that the witness node can't be reached, Failover Manager notifies the administrator of the state of the witness.

Note: If the witness fails and the cluster has only two nodes, then there is no failover protection. The standby node can't know if the primary failed or was disconnected. In a two-node cluster, if the primary database fails but the nodes are still connected, failover still occurs. The standby can confirm the condition of the primary database.

16.8 Nodes become isolated from the cluster

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

If one or more nodes (but less than half of the cluster) become isolated from the rest of the cluster, the remaining cluster behaves as if the nodes have failed. The agents attempt to discern if the primary node is among the isolated nodes. If it is, the primary 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 primary unless `auto.reconfigure` is set to `false` .

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

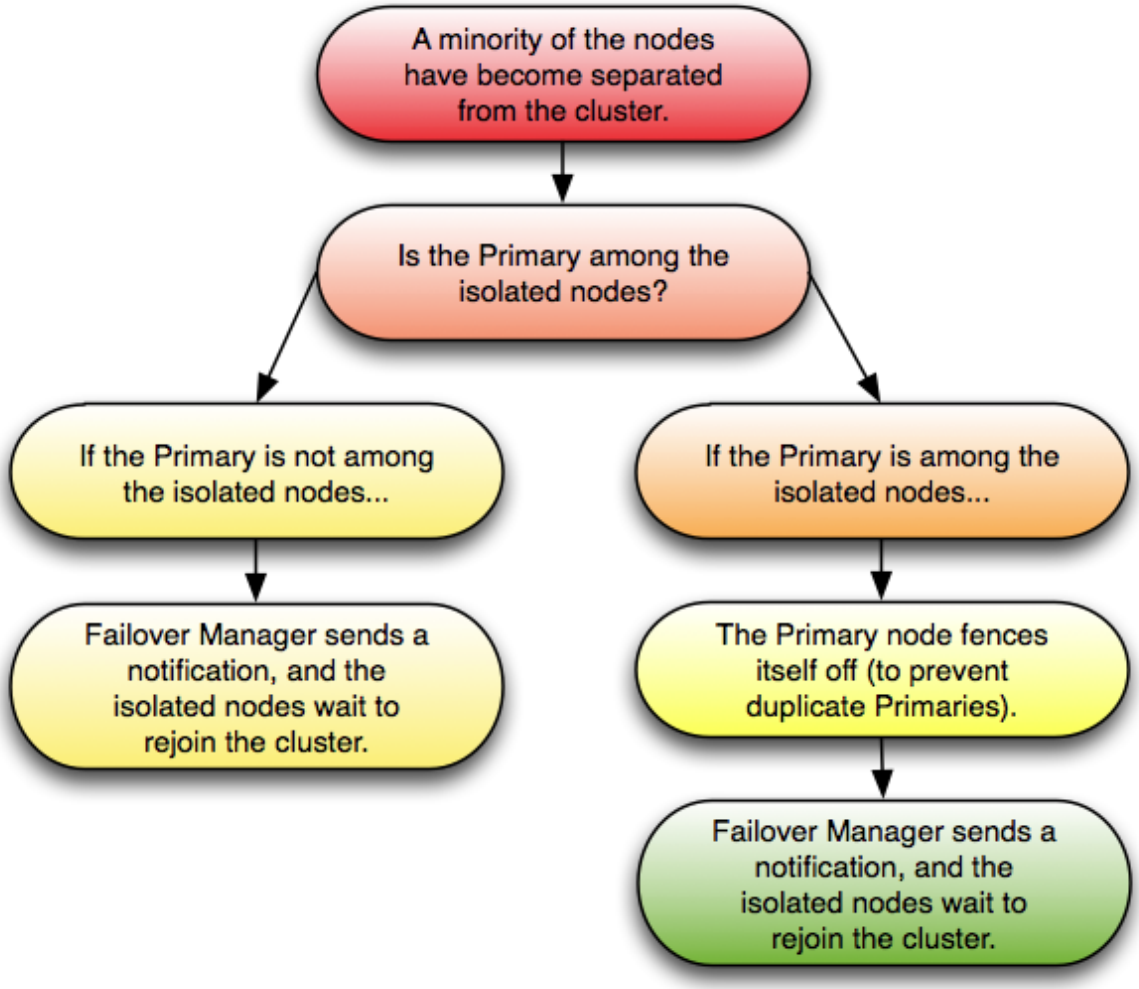

Fig. 6: If members of the cluster become isolated.

CHAPTER

SEVENTEEN

TROUBLESHOOTING

17.1 Authorization file not found. Is the local agent running?

If you invoke an Failover Manager cluster management command and Failover Manager isn't running on the node, the `efm` command displays an error:

```
Authorization file not found. Is the local agent running?
```

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

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

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

17.3 Notification; Unexpected error message

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

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

jvm.options=-Xmx128m
```

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

17.4 Confirming the OpenJDK version

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

```
#  java -version

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

CHAPTER

EIGHTEEN

CREATING A FAILOVER MANAGER CLUSTER

Failover Manager is a high-availability tool that allows a Postgres primary node to automatically failover to a standby node in the case of a software or hardware failure on the primary node.

This tutorial describes configuring a Failover Manager cluster in a test environment. Before configuring Failover Manager for a production deployment, read and understand the rest of the Failover Manager documentation.

Using EDB Postgres Advanced Server as an example (Failover Manager also works with PostgreSQL), follow these steps for basic installation and configuration before beginning the tutorial:

- Install and initialize a database server on one primary and one or two standby nodes. For information about installing, refer to the

  EDB Postgres Advanced Server documentation.

- Postgres streaming replication must be configured and running between the primary and standby nodes. For detailed information about configuring streaming replication, refer to *Configuring streaming replication* .

- Install Failover Manager on each primary and standby node. During EDB Postgres Advanced Server installation, you configured an EDB repository on each database host. You can use the EDB repository and the `yum install` command to install Failover Manager on each node of the cluster:

```
yum install edb-efm47
```

During the installation process, the installer creates a user named efm that has privileges to invoke scripts that control the Failover Manager service for clusters owned by enterprisedb or postgres. The example that follows creates a cluster named `efm` .

Start the configuration process on a primary or standby node. Then, copy the configuration files to other nodes to save time.

1. Create working configuration files. Copy the provided sample files to create Failover Manager configuration files, and correct the ownership and version number if you are installing a different version:

```
cd /etc/edb/efm-4.7

cp efm.properties.in efm.properties

cp efm.nodes.in efm.nodes

chown efm:efm efm.properties

chown efm:efm efm.nodes
```

1. Create an *encrypted password* needed for the properties file:

```
/usr/edb/efm-4.7/bin/efm encrypt efm
```

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

1. Update `efm.properties` . The `<cluster_name>.properties` file (`efm.properties` in this example) contains parameters that specify connection properties and behaviors for your Failover Manager cluster. Modifications to property settings are applied when Failover Manager starts.

The properties mentioned in this tutorial are the minimal properties required to configure a Failover Manager cluster. If you're configuring a production system, review *Configuring Failover Manager* for detailed information about Failover Manager options.

Provide values for the following properties on all cluster nodes:

Property	Description
db.user	The name of the database user.
db.password.encrypted	The encrypted password of the database user.
db.port	The port monitored by the database.
db.database	The name of the database.
db.service.owner	The owner of the *data* directory (usually *postgres* or *enterprisedb*). Required only if the database is running as a service.
db.service.name	The name of the database service (used to restart the server). Required only if the database is running as a service.
db.bin	The path to the *bin* directory (used for calls to *pg_ctl*).
db.data.dir	The *data* directory in which EFM will find or create the *recovery.conf* file or the *standby.signal* file.
user.email	An email address at which to receive email notifications (notification text is also in the agent log file).
bind.address	The local address of the node and the port to use for Failover Manager. The format is: *bind.address=1.2.3.4:7800*
is.witness	*true* on a witness node and *false* if it is a primary or standby.
ping.server.ip	If you are running on a network without Internet access, set *ping.server.ip* to an address that is available on your network.
auto.allow.hosts	On a test cluster, set to *true* to simplify startup; for production usage, consult the Failover Manager User Guide.
stable.nodes.file	On a test cluster, set to *true* to simplify startup; for production usage, consult the Failover Manager User Guide.

1. Update `efm.nodes` . The `<cluster_name>.nodes` file (`efm.nodes` in this example) is read at startup to tell an agent how to find the rest of the cluster or, in the case of the first node started, can be used to simplify authorization of subsequent nodes. Add the addresses and ports of each node in the cluster to this file. One node acts as the membership coordinator. Include in the list at least the membership coordinator's address. For example:

```
1.2.3.4:7800

1.2.3.5:7800

1.2.3.6:7800
```

The Failover Manager agent doesn't validate the addresses in the `efm.nodes` file. The agent expects that some of the addresses in the file can't be reached (for example, that another agent hasn't been started yet).

1. Configure the other nodes. Copy the `efm.properties` and `efm.nodes` files to `/etc/edb/efm-4.7` on the other nodes in your sample cluster. After copying the files, change the file ownership so the files are owned by efm:efm. The `efm.properties` file can be the same on every node, except for the following properties:

   - Modify the `bind.address` property to use the node's local address.

- Set `is.witness` to `true` if the node is a witness node. If the node is a witness node, the properties relating to a local database installation are ignored.

2. Start the Failover Manager cluster. On any node, start the Failover Manager agent. The agent is named `edb-efm-4.7` ; you can use your platform-specific service command to control the service. For example, on a RHEL 7.x or Rocky Linux/AlmaLinux/RHEL 8.x host, use the command:

```
systemctl start edb-efm-4.7
```

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

```
/usr/edb/efm-4.7/bin/efm cluster-status efm
```

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

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

```
cat /var/log/efm-4.7/startup-efm.log
```

18.1 Perform a switchover

If the cluster status output shows that the primary and standby nodes are in sync, you can perform a switchover:

```
/usr/edb/efm-4.7/bin/efm promote efm -switchover
```

The command promotes a standby and reconfigures the primary database as a new standby in the cluster. To switch back, run the command again.

18.2 Access online help

For quick access to online help, use:

```
/usr/edb/efm-4.7/bin/efm --help
```