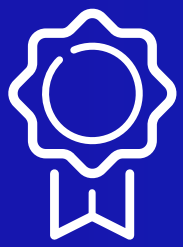
Best practices for Fujitsu Enterprise Postgres High Availability on Linux using Mirroring Controller and Arbitration Server



FUJITSU

Contents at a glance

Table of contents

1. Introduction

Fujitsu Enterprise Postgres (FEP) extends open-source PostgreSQL with advanced enterprise features, including integrated **High Availability (HA)** through the **Mirroring Controller** and **Arbitration Server**. This solution ensures continuous database service by automatically detecting failures, promoting standby servers, and maintaining data consistency through synchronous streaming replication.

This document outlines deployment and operational best practices for implementing HA in Linux environments using Fujitsu Enterprise Postgres.

2. Fujitsu Enterprise Postgres Mirroring Controller

2.1. Overview

The **Mirroring Controller** manages the replication cluster, performing:

- Continuous **health checks** on primary and standby database nodes.
- **Automatic Failover** during primary failure and **Switchover** with minimal downtime.
- Coordination with the **Arbitration Server** to prevent split-brain.

Mirroring Controller runs on both primary and standby nodes and uses TCP protocol for communication and heartbeats to verify peer status. Binary streaming replication is the backbone of the Fujitsu Enterprise Postgres high availability solution.

2.2. TCP ping monitor

Mirroring Controller periodically issues **TCP ping requests** to verify node reachability:

- The heartbeat monitoring of the database server uses the OS ping command
- Monitors the peer node and the arbitration server.
- Uses parameters such as heartbeat_interval and heartbeat_timeout.

2.3. Process check

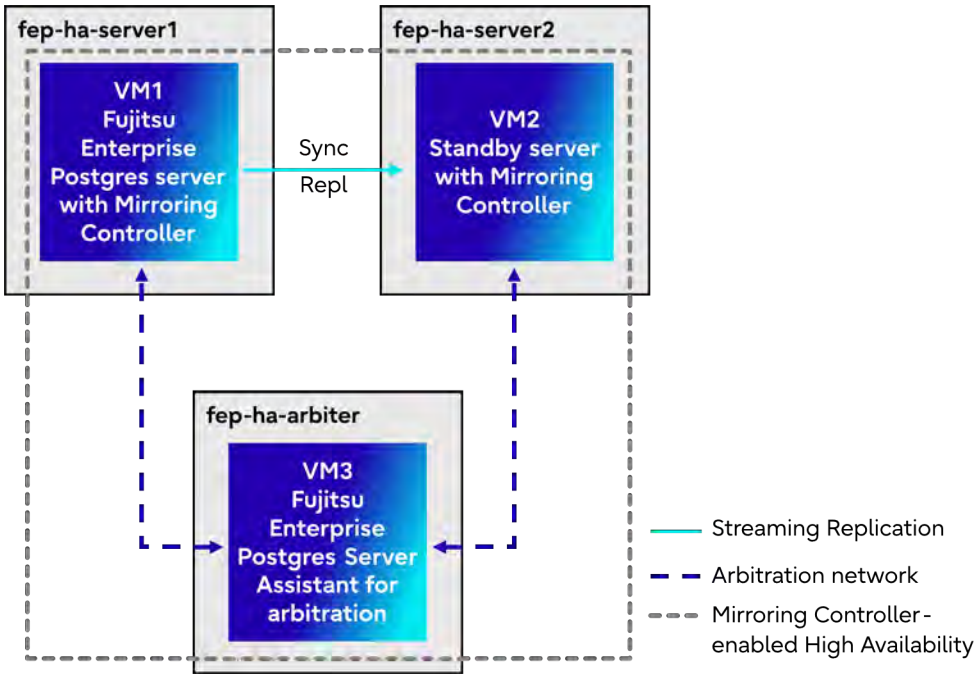
The Mirroring Controller verifies that Fujitsu Enterprise Postgres processes (primary and standby) are running correctly:

- **Process failure:** Mirroring Controller periodically accesses the database processes and checks the status. A process error is detected by monitoring whether an access timeout occurs.
- **Disk failure:** Mirroring Controller periodically creates files on the data storage destination disks (PGDATA, WAL destination & tablespace location). A disk error is detected when an I/O error occurs.
- **Replication issue:** Mirroring Controller detects streaming replication issues (log transfer network and WAL send/receive processes) by periodically accessing the PostgreSQL system views.
- **Mirroring Controller process failure:** The Mirroring Controller monitoring process detects Mirroring Controller process failures and no responses by periodically querying the Mirroring Controller process. If an issue is detected, Mirroring Controller is automatically restarted by the Mirroring Controller monitoring process.

3. Reference architecture

The Fujitsu Enterprise Postgres high availability architecture is built on primary and standby database instances plus a third server acting as the arbitration node. The primary and standby both run Fujitsu Enterprise Postgres in "database multiplexing mode" (i.e., streaming replication under Fujitsu Enterprise Postgres' control) such that transaction logs from the primary are continuously shipped to and applied at the standby. Each database node runs the Mirroring Controller process and a monitoring sub-process which periodically performs OS/server heartbeats, disk I/O checks, and replication-lag checks across the servers. The arbitration server resides on a separate server and serves as an objective third-party observer: when Mirroring Controller on one of the database nodes detects abnormality (for example network interface glitch or unresponsive nodes), the database node sends a query to the arbitration server which then assesses the health of the peer database node and decide to fence (isolate) the failing node or trigger a switchover to the standby.

From a systems-layout perspective, the architecture requires at least four distinct networks (job/business-application network, admin network for Mirroring Controller heartbeats, log-transfer network for the WAL/streaming replication, and arbitration network between the database nodes and the arbitration server) in order to ensure proper segregation of traffic and prevent split-brain conditions. In operation, the standby server can serve read-only/query workloads while the primary handles writes, and if a failure or fault is detected (for example a disk I/O fault, streaming replication lag or OS hang) the Mirroring Controller plus arbitration server flow enables quick failover or disconnection of the affected node. This architecture enables leveraging Fujitsu Enterprise Postgres a well-defined HA blueprint with clear roles for the database servers, Mirroring Controller, and arbitration services.



3.1. Architecture components

3.1.1. Primary

- Active Fujitsu Enterprise Postgres instance handling read-write operations.
- Streams WAL changes to the standby node.
- Continuously monitored by both Mirroring Controller agent and Arbitration Server.

3.1.2. Standby

- Receives WAL records through synchronous replication.
- Runs Mirroring Controller in standby mode, ready for promotion.
- Continuously monitored by both Mirroring Controller agent and Arbitration Server.

3.1.3. Arbitration Server

- Separate third host used for quorum decisions in two-node clusters.
- Prevents split-brain during network isolation by allowing only one node to be promoted as primary.

3.1.4. Mirroring Controller agents

- Runs on both primary and standby database nodes.
- Exchanges heartbeats, status messages, and control commands.
- Uses Arbitration Server to adjudicate primary when abnormality is detected.

4. Best practices – Deployment and installation

4.1. Package installation

- Install Fujitsu Enterprise Postgres packages **only from official Fujitsu repositories** or certified media.
- Ensure all nodes (primary, standby, arbitration) run the same OS version.
- Install all required OS Packages.
- Ensure all nodes (primary, standby, arbitration) run the same Fujitsu Enterprise Postgres version and patch level.
- Use install.sh (or silent.sh) file for consistent deployment.

4.2. File system layout

Path	Description	Best Practice
/opt/fsepv16server64/bin	Fujitsu Enterprise Postgres Server binaries	Mount on local SSD.
/database/inst1	Database cluster	Use dedicated volume (preferably EXT4).
/opt/fsepv16assistant/bin	Fujitsu Enterprise Postgres Server Assistant binaries	Mount on local SSD.
/mcdir/inst1	Mirroring Controller configuration files	Mount on local SSD.
/mcarb_dir/arbiter1	Arbiter configuration files	Mount on local SSD.

4.3. User and privileges

- Fujitsu Enterprise Postgres server and Server Assistant installation requires root or sudo access.
- Run Fujitsu Enterprise Postgres database instance under a dedicated user account (fepuser).
- Mirroring Controller and arbitration server configuration files should be owned by same OS user account (fepuser), as database.
- Root or sudo privileges must not be used for daily operations.

5. Network requirements

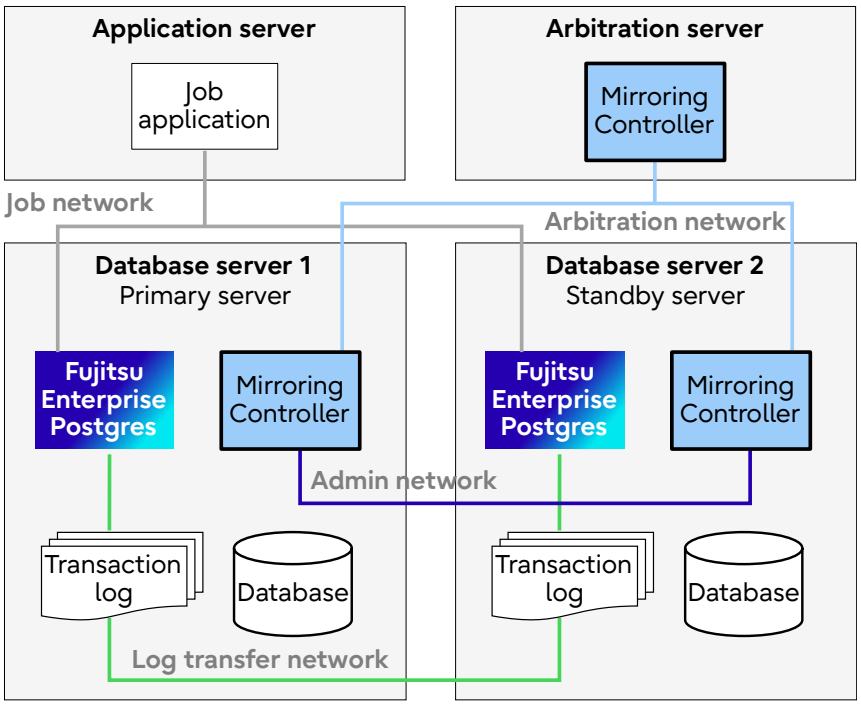
The **Mirroring Controller (MC)** requires a **dedicated, reliable network** between the primary and standby servers to ensure synchronous data replication and monitoring. The networking design is a critical component to ensure correct fail-over behavior, split-brain prevention, and optimal performance. The Arbitration network must be designed so that it does not share the same network fabric as the admin or log-transfer networks – this prevents faults in replication or monitoring traffic of Mirroring Controller affecting the failover behavior.

In terms of security and isolation, the admin network and log transfer network should **not** be reachable from external networks (i.e., they must be isolated from client or public access). Similarly, the line between the database nodes and the arbitration server must also be secured and normally not accessible from outside the cluster environment.

5.1. Network types

In Mirroring Controller, there are various types of networks used:

Network Type	Description
Job network	Network between the application that accesses the database, and the database server.
Arbitration network	Network used by the arbitration server to check the status of the primary server and standby server, and communicate with Mirroring Controller of the database servers.
Admin network	Network used by the primary server and the standby server to monitor each other using Mirroring Controller, and to control Mirroring Controller of other servers.
Log transfer network	Network used to transfer the WAL transaction logs of the database, which is part of database multiplexing.



5.1.1. Two network interfaces are recommended

- For automatic failover due to network abnormalities on the admin network, it is recommended to have two network interfaces, as below,
  - **Private network** – for WAL streaming, admin and application connections.
  - **Arbitration network** – dedicated network interface not affected by line failures or load.
- Low latency network is highly recommended.

5.1.2. Recommendation for cloud deployment

When deploying the Fujitsu Enterprise Postgres HA solution in the cloud environment, consider:

- Install each database server and arbitration server in different Availability Zones (AZ).
- Prepare each network (business network, arbitration network, admin network, and log transfer network) using the subnet function of the cloud service. In this document, assume each network as separate subnet created by the cloud service.
- If a virtual network failure occurs, you cannot use the database multiplexing mode feature. Wait for the cloud service to recover, and after recovery, check the redundancy of the database.
- Ensure reliable fencing script is in place, to prevent split-brain. (Refer section:8)
- If you cannot prepare four networks in environments such as public cloud, or cannot add network interface cards to the server, the following configurations that share some networks are possible:
  - Share all networks: Share job network, admin network and arbitration network.

When sharing networks, there are no impacts other than increased load due to network sharing during regular operations. However, there are impacts such as restrictions on features for multiplexed modes and the need for special designs for operations in response to network abnormality.

5.2. Fujitsu Enterprise Postgres and Mirroring Controller ports

Mirroring Controller uses two separate ports for Arbitration and admin networks.

Purpose	Default port
Fujitsu Enterprise Postgres instance port	27500
Mirroring Controller admin network port	27540
Arbitration network port	27541

It is best practice to NOT use the default ports and recommend using different port numbers for the above functionalities.

Also ensure the assigned ports are open and reserved for Fujitsu Enterprise Postgres and make sure these ports are allowed by the firewalls.

6. Mirroring Controller configuration parameters

Configuration parameters for "server<identifier>.conf" on database nodes. For example, server1.conf:

Parameter	Description	Remarks
db_instance	Specify PGDATA directory	
target_db	Specify 'template1' or 'postgres'	Use 'template1', in case of enabling session audit, as auditing will capture Mirroring Controller connection
heartbeat_error_action	Operation when heartbeat abnormality is detected.	arbitration: Perform automatic degradation using the arbitration server. It also accepts other values, please refer Fujitsu Enterprise Postgres manuals.
heartbeat_interval	Interval time for abnormality monitoring (in milliseconds)	800ms
fencing_command	Fence the database server, where an error is determined to have occurred.	Refer the sample fencing script available and test it thoroughly. (Refer section:8)
pre_detach_command	Specify the full path of the command to be called by Mirroring Controller before the standby server is disconnected from the cluster system.	This parameter can be leveraged for email notification, before the standby is detached.
post_attach_command	Specify the full path of the command to be called by Mirroring Controller after the standby server is attached to the cluster system.	This parameter can be leveraged for email notification, after the standby is attached.
post_switch_command	Specify the full path of the command to be called by Mirroring Controller after a new primary server is promoted during a failover of the primary server.	This parameter can be leveraged for email notification, after the failover or switchover on the instance.

Configuration parameters for "arbitration.conf" on database nodes. For example, arbitration.conf:

Parameter	Description	Remarks
port	Port number of the Mirroring Controller arbitration process	
my_address	The IP address or host name of the arbitration server to be specified in network.conf on the database server.	
fencing_command	Specify the full path of the fencing command that fences a database server where an error is determined to have occurred.	Fencing script should be well tested in the environment. Sample scripts are available for reference.

Parameter	Description	Remarks
syslog_ident	Specify an identifier for arbitration process message in the system log.	

7. Sample script – Fencing

The fencing command can be implemented by simply stopping the operating system or server. For example, if stopping the power for the database server, it is possible to use a utility to control the hardware control board in environments equipped with boards compatible with IPMI hardware standard.

7.1. Sample script for IPMI tool

Below is a sample script of a fencing command that powers off the database server using the IPMI tool.

Sample shell script

```
/installDir/fsepv17assistant/share/mcarb_execute_fencing.sh.sample
```

Modify the parameter values that are highlighted in blue.

```
# Need to be modified
srv1ident="server1"          # Server identify of Mirroring Controller
srv2ident="server2"          # Server identify of Mirroring Controller
ipmi_admin="fsepuser"        # Remote server username for IPMItool
ipmi_password="fsepuser"     # Remote server password for IPMItool
check_interval=500000        # Interval for checking the power status power-off
srv1addr="192.0.4.100"       # Remote server address of srv1ident for IPMItool
srv2addr="192.0.4.110"       # Remote server address of srv2ident for IPMItool

logdir="/var/tmp/work"
logfile="${logdir}/fencing.$(date '+%Y%m%d%H%M%S').log"
```

7.2. Sample shell script for Amazon Web Services

Below is a sample script of a fencing command to stop the power of the database server using the CLI command of the AWS cloud service.

```
/installDir/fsepv17assistant/share/mcarb_execute_fencing.sh.aws.sample
```

Modify the parameter values that are highlighted in blue.

```
# Need to be modified
################################################################################
srv1ident="server1"            # Server identify of Mirroring Controller
srv1id="server1-id"            # Remote server id of srv1ident for AWS CLI

srv2ident="server2"            # Server identify of Mirroring Controller
srv2id="server2-id"            # Remote server id of srv1ident for AWS CLI

check_interval=0.5             # Interval for checking the power status power-off

logdir="/var/tmp/work"
logfile="${logdir}/fencing.$(date '+%Y%m%d%H%M%S').log"
################################################################################
```

7.3. Sample shell script for Microsoft Azure

Below is a sample script of a fencing command to stop the power of the database server using the CLI command of the Azure cloud service.

```
/installDir/fsepv17assistant/share/mcarb_execute_fencing.sh.az.sample
```

Modify the parameter values that are highlighted in blue.

```
# Need to be modified
#################################################################################
srv1ident="server1"                   # Server identify of Mirroring Controller
srv1name="server1-name"               # Remote server name of srv1ident for Azure CLI

srv2ident="server2"                   # Server identify of Mirroring Controller
srv2name="server2-name"               # Remote server name of srv1ident for Azure CLI

resource_group="resource-group"       # resource-group for Azure CLI

check_interval=0.5                    # Interval for checking the power status power-off

logdir="/var/tmp/work"
logfile="${logdir}/fencing.$(date '+%Y%m%d%H%M%S').log"

# Azure login
az login –identity
#################################################################################
```

8. Other recommendations

8.1. synchronous_standby_ names

Synchronous mode of streaming replication is mandatory requirement to implement high availability using Mirroring Controller and Server Assistant.

Set synchronous replication on the database servers, as below:

```
synchronous_commit = on
synchronous_standby_names = 'standby1'
```

Check replication status via:

```
SELECT application_name, sync_state FROM pg_stat_replication;
```

Note:

- Never set the parameter 'synchronous_standby_names' in the postgresql.auto.conf file. As this alters the behavior of failover or switchover.
- Remove multiple entries of the parameter 'synchronous_standby_names' and retain only one entry.

8.2. Low latency network

It is important to have low network latency for smooth operations of Mirroring Controller.

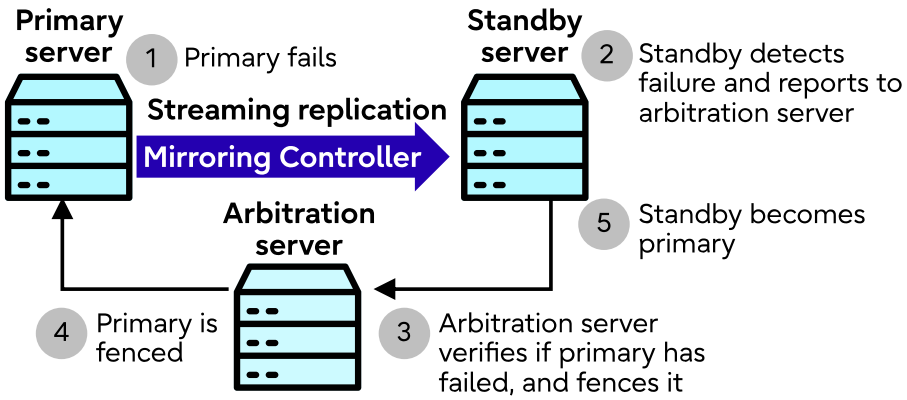
Tune the heartbeat_interval and heartbeat_timeout, based on the round-trip time of ping command response.

Use tools like ping or mtr for accurate latency measurements.

9. Failover and Recovery strategy

9.1. Failover sequence

1. Primary DB server crashed
2. Mirroring Controller on standby detects loss of primary heartbeats and report to arbitration server
3. Arbitration Server verifies the primary status, and it is unreachable.
4. Primary server is fenced, when Mirroring Controller on database nodes are also unreachable.
5. Mirroring Controller promotes standby as new primary instance.
6. Fetch the internal state of database servers.
7. Connection Manager redirects the client requests automatically to newly promoted server.



9.2. Rebuild strategy

9.2.1. Standby failure

1. Stop the Mirroring Controller on standby server.
2. Using pgbackrest, restore the standby database from backup.
3. If pgbackrest is not configured, then remove the files under $PGDATA and use pg_basebackup and copy files from primary server.

```
pg_basebackup -h primary -D /database/inst1 -X stream -R -P
```

4. Update the application_name field for primary_conninfo parameter in the $PGDATA/postgresql.auto.conf file.
5. Uncomment 'synchronous_standby_names' in the postgresql.conf on the primary server and reload the configuration file.
6. Start the Mirroring Controller on standby.
7. Check the Mirroring Controller status on the primary and standby server.

9.2.2. Primary failure

When primary server fails for whatever reason, the Mirroring Controller automatically fails over to the standby server.

1. Stop the Mirroring Controller on the old primary server.
2. Use pg_rewind, if possible (Either wal_log_hints or data_checksums must be enabled.)
3. If pg_rewind is not possible, use pgbackrest to restore from the backup.
4. If pgbackrest is not configured, then remove the files under $PGDATA and use pg_basebackup and copy files from primary server.

```
pg_basebackup -h primary -D /database/inst1 -X stream -R -P
```

5. Update the application_name field for primary_conninfo parameter in the $PGDATA/postgresql.auto.conf file.
6. Uncomment 'synchronous_standby_names' in the postgresql.conf on the primary server and reload the configuration file.
7. Start the Mirroring Controller on the new standby (old primary).
8. Check the Mirroring Controller status on the primary and standby server.

9.2.3. Switchover _planned maintenance–

Use Server Assistant or Mirroring Controller command:

```
mc_ctl status -M /mcdir/inst1
mc_ctl switch -M /mcdir/inst1
mc_ctl status -M /mcdir/inst1
```

Performs controlled failover to the standby server and follow the primary failure rebuilding steps.

10. Logging

Component	Location	Notes
PostgreSQL	$PGDATA/log/postgresql.log	Database and instance logs
Mirroring Controller log	/var/log/messages	Specify the program name in the syslog_ident parameter of the serverIdentifier.conf file of the database server.
Arbitration Server log	/var/log/messages	Specify the program name in the syslog_ident parameter of the arbitration.conf file of the arbitration server.

11. Conclusion

Fujitsu Enterprise Postgres combines the robustness of PostgreSQL with enterprise-class HA through the **Mirroring Controller** and **Arbitration Server**.

Proper installation, tuned configuration, redundant networking, and disciplined operational practices ensure zero data loss, automatic failover, and resilient service continuity.

By following these best practices, on Linux deployments can achieve:

- Predictable failover and recovery behavior.
- Data integrity even in network isolation scenarios.
- Simplified administration and monitoring under real-world workloads.

FUJITSU

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications

Contact
Fujitsu Limited
Email: enterprisepostgresql@fujitsu.com
Website: fast.fujitsu.com