文書番号	DB-P006567-003

EDB Postgres / PostgreSQL
運用ガイドライン（レプリケーション）

株式会社アシスト
ビジネスインフラ技術本部

ソフトバンク株式会社
IT 本部 IT インフラ統括部 IT 基盤開発部

改訂履歴

改訂日	版	改訂内容
2020/3/19	1.0	初版発行

はじめに

■本資料の概要と目的

本資料は EDB Postgres / PostgreSQL の運用の指針を示す『運用ガイドライン』です。
EDB Postgres や PostgreSQL の運用を設計する際に必要な項目示し、設定値を検討する際の参考情報となることを目的としています。

■本資料のEDB Postgres / PostgreSQLバージョン

バージョン 11 を対象として構成しています。

■本資料の対象者

データベース管理者、EDB Postgres / PostgreSQL を使用したアプリケーション開発を行う方を対象とします。

■資料内の記述について

指針

> 指針

構文

> **構文**

構文表記規則

[ ]	省略可能
{ A | B }	A または B のどちらかを選択
n	数値の指定
_	デフォルト値

プロンプト

#	root ユーザーで実行する
$	EDB Postgres/PostgreSQL インストールユーザーで実行する
=#	psql にて実行する （接続ユーザーはスーパーユーザーであることを表す）
=>	psql にて実行する （接続ユーザーは一般ユーザーであることを表す）
SQL>	SQL*Plus または EDB*Plus にて実行する

目　　次

1.　ストリーミングレプリケーションの運用 1
1.1. 起動/停止 1
1.2. 同期/非同期の切り替え 2
1.3. スレーブの昇格(フェイルオーバー) 3
1.4. スイッチオーバ / スイッチバック 4
1.4.1. スレーブ停止 （edb-serve2） 4
1.4.2. レプリケーションスロット削除 （edb-serve1） 4
1.4.3. マスター停止（edb-serve1） 5
1.4.4. スレーブ起動および昇格 (edb-serve2) 5
1.4.5. レプリケーションスロット作成 (edb-serve2) 5
1.4.6. 一時スレーブ起動およびスロット有効化 (edb-serve1) 7
1.4.7. 一時スレーブ停止 およびメンテナンス作業 (edb-serve1) 7
1.4.8. レプリケーションスロット削除 (edb-serve2) 8
1.4.9. 一時スレーブ起動（スロット無効化）および昇格 (edb-serve1) 8
1.4.10. レプリケーションスロット作成 (edb-serve1) 8
1.4.11. スレーブ起動（スロット有効化） (edb-serve2) 9
1.4.12. レプリケーションの状態を確認 9
2.　ストリーミングレプリケーション概要 10
2.1. ストリーミングレプリケーションの構成と目的 10
2.2. 同期レベル 11
2.3. レプリケーションスロット 12
2.4. 巻き戻し (pg_rewind) 13
2.5. 複数スレーブ構成 15
2.6. カスケード構成 16
3.　ストリーミングレプリケーションの構築 17
3.1. パスワードなしのssh実行環境の設定 17
3.2. レプリケーション用パラメータの設定 （edb-server1） 19
3.3. レプリケーション用ユーザの作成 （edb-server1） 20
3.4. レプリケーション用クライアント認証の設定 (edb-server1) 20

3.5. ディレクトリの確認（edb-server1）.....21
3.6. pg_basebackup実行前の準備作業（edb-server2）.....22
3.7. pg_basebackup実行（edb-server2）.....23
3.8. レプリケーションスロットの作成.....24
3.9. recovery.conf の作成　(edb-server2).....25
3.10. スレーブの起動.....26
4. ストリーミングレプリケーションの監視.....27
4.1. レプリケーションの確認.....27
4.2. レプリケーションスロットの確認.....29
4.3. スレーブからのレプリケーション確認.....30
4.3.1. pg_stat_wal_receiver ビュー.....30
4.3.2. リカバリ情報取得関数.....31
5. バックアップ・リカバリ.....32
5.1. システム構成.....32
5.2. 物理バックアップ.....33
5.2.1. バックアップの仕様.....33
5.2.2. バックアップ・リカバリ要件.....33
5.2.3. バックアップ方針.....33
6. HCP構成における有事対策.....34
6.1. 対障害設計.....34
6.1.1. プロセス障害.....34
6.1.2. 仮想マシン障害および物理ホスト障害.....34
6.1.3. ネットワーク障害.....34
6.1.4. ディスク障害.....35
6.2. インスタンス障害時の挙動.....35
6.2.1. インスタンス障害時のコンポーネント別まとめ.....35
6.2.2. マスター障害.....36
6.2.3. スレーブ障害（1台目）.....38
6.2.4. スレーブ障害（2台目）.....39
6.2.5. エージェント（マスター）障害.....40
6.2.6. エージェント（スレーブ）障害.....41

7. レプリケーション設定(postgresql.conf) 42
7.1. 送出ノードのパラメータ 43
7.1.1. max_wal_senders （送出ノード） 43
7.1.2. max_replication_slots （送出ノード） 43
7.1.3. wal_keep_segments （送出ノード） 44
7.1.4. wal_sender_timeout （送出ノード） 44
7.2. マスターのパラメータ 45
7.2.1. synchronous_commit （マスター） 45
7.2.2. synchronous_standby_names （マスター） 46
7.2.3. vacuum_defer_cleanup_age （マスター） 47
7.2.4. wal_log_hints （マスター） 47
7.3. スレーブ 48
7.3.1. hot_standby （スレーブ） 48
7.3.2. wal_receiver_status_interval （スレーブ） 48
7.3.3. hot_standby_feedback （スレーブ） 49
7.3.4. wal_receiver_timeout （スレーブ） 49
7.3.5. max_standby_archive_delay （スレーブ用） 50
7.3.6. max_standby_streaming_delay （スレーブ用） 50
8. スレーブ設定(recovery.conf) 51
8.1. PITR設定 51
8.1.1. restore_command 51
8.1.2. recovery_target_timeline (PITR) 51
8.1.3. recovery_target_time (PITR) 53
8.1.4. recovery_target_action (PITR) 53
8.2. ストリーミングレプリケーションのスレーブ設定 54
8.2.1. standby_mode (スレーブ) 54
8.2.2. primary_conninfo (スレーブ) 54
8.2.3. recovery_target_timeline (スレーブ) 55
8.2.4. restore_command (スレーブ) 55
8.2.5. primary_slot_name (スレーブ) 56
8.2.6. archive_cleanup_command (スレーブ) 57
8.2.7. trigger_file (スレーブ) 57

1. ストリーミングレプリケーションの運用

1.1.起動/停止

起動/停止にはpg_ctlコマンドを使用する。

ストリーミングレプリケーション専用のコマンドは用意されていないため、起動/停止の際は pg_ctl コマンドを使用する。マスター/スレーブの起動/停止順は任意であるが、正常停止を障害と誤認識されないよう注意する。

実行例）

```
/*  マスタ-の停止  */
$  pg_ctl stop

/*  スレーブの停止  */
$  pg_ctl stop

/*  スレーブの起動  */
$ pg_ctl start

/*  マスターの起動  */
$ pg_ctl start
```

1.2.同期/非同期の切り替え

同期/非同期の切り替えはALTER SYSTEM文を使用し、synchronous_standby_namesパラメータを変更する。

同期/非同期はスレーブ毎に設定できる。同期スレーブの障害発生時には非同期に切り替える必要がある。psql で完結するため、管理性に優れている。

同期/非同期の切り替え構文

```
ALTER SYSTEM SET synchronous_standby_names = { '' | * | 特定スレーブのapplication_name }
```

表 1-1 synchronous_standby_namesパラメータの設定値

設定値	概要
''	全スレーブを非同期
*	全スレーブを同期
特定スレーブの application_name	特定スレーブのみを同期

実行例）

```
$ psql

/*  postgresql.auto.conf に書き込み  */
=# ALTER SYSTEM SET synchronous_standby_names = '';      -- 全スレーブを非同期
ALTER SYSTEM

/*  リロードで全体に反映  */
=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 行)
```

1.3.スレーブの昇格(フェイルオーバー)

スレーブの昇格（フェイルオーバー）にはpg_ctl promoteコマンドを使用する。

スレーブを昇格（フェイルオーバー）する際は、スプリットブレイン状態を防ぐためマスターノードを先に停止する。

実行例）

```
/*  マスターノードで実行  */
/*  マスター停止          */
$ pg_ctl stop

/*  以後はスレーブノードで実行  */
/*  昇格前の状態          */
$ pg_controldata | head -4
pg_control バージョン番号:                1100
カタログバージョン番号:                   201809051
データベースシステム識別子:               6685453832160810991
データベースクラスタの状態:               アーカイブリカバリ中

/*  昇格                  */
$ pg_ctl promote
サーバの昇格を待っています....完了
サーバは昇格しました

/*  昇格後の状態          */
$ pg_controldata | head -4
pg_control バージョン番号:                1100
カタログバージョン番号:                   201809051
データベースシステム識別子:               6685453832160810991
データベースクラスタの状態:               運用中
```

レプリケーションスロットを使用している場合は、別途旧マスターをスレーブとして構成後にレプリケーションスロットの削除を行う。スレーブではデータの更新はできないが、レプリケーションスロットは$PGDATA/pg_replslot に保存しているため、削除可能である。

1.4.スイッチオーバ / スイッチバック

メンテナンス時など、一時的にマスター/スレーブの役割を入れ替えたい場合にはスイッチオーバー/スイッチバックを実施する。この際、メンテナンス中にWALが上書きされないようレプリケーションスロットを作成する。

以降、スイッチオーバー/スイッチバックの手順を記載する。本項に記載の手順では以下のように役割が変遷する。()内の数値はタイムライン ID であり、本手順ではタイムライン ID の変遷も確認する。

●開始時
edb-serve1　マスター（10）
edb-serve2　スレーブ（10）

●メンテナンス中
役割を便宜的に以下のように表記する。
edb-serve1　一時スレーブ（11）　停止（メンテナンス実行）
edb-serve2　一時マスター（11）　運用中

●終了時
edb-serve1　マスター（12）
edb-serve2　スレーブ（12）

1.4.1. スレーブ停止 （edb-serve2）

レプリケーションスロットを削除するため、スレーブを停止して非アクティブ化する。

実行例）

```
$ pg_ctl stop
サーバ停止処理の完了を待っています....完了
サーバは停止しました
```

1.4.2. レプリケーションスロット削除 （edb-serve1）

レプリケーションスロットが非アクティブ化されたことを確認後、スロットを削除する。

実行例）

```
/*  レプリケーションスロットの状況を確認  */
=# SELECT slot_name,active FROM pg_replication_slots ;
-[ RECORD 1 ]-------+------------
slot_name           | slot_slave1
active              | f                                    ★非アクティブ

/*  レプリケーションスロットを削除  */
=# SELECT pg_drop_replication_slot('slot_slave1');
-[ RECORD 1 ]------------+-
pg_drop_replication_slot |
```

1.4.3. マスター停止 （edb-serve1）

スプリットブレイン状態を避けるため、スレーブ昇格の前にマスターを停止する。

実行例）

```
$ pg_ctl stop
サーバ停止処理の完了を待っています....完了
サーバは停止しました
```

1.4.4. スレーブ起動および昇格 (edb-serve2)

スレーブを起動する前にrecovery.confからレプリケーションスロットの設定行をコメントアウトする。これにより「存在しないレプリケーションスロットを指定するとスレーブの起動に失敗する」状態を回避する。
スレーブ起動を起動し、昇格させる（一時マスター）。recovery.conf が recovery.done となる。

実行例）

```
/*  recovery.conf からレプリケーションスロットの指定行をコメントアウト */
$ vi $PGDATA/recovery.conf
-----------------------------------
#primary_slot_name = 'slot_slave1'
-----------------------------------

/* タイムライン ID の確認  */
$ pg_controldata | grep -i timeline
最終チェックポイントの PrevTimeLineID:          10

/*  スレーブ起動  */
$ pg_ctl start
サーバの起動完了を待っています....   ～中略～
サーバ起動完了

/*  昇格  （一時マスターに）  */
$ pg_ctl promote
サーバの昇格を待っています....完了
サーバは昇格しました

/*  状態を確認  */
$ pg_controldata | head -4
pg_control バージョン番号:                1100
カタログバージョン番号:                   201809051
データベースシステム識別子:               6685453832160810991
データベースクラスタの状態:               運用中                    ★マスター

/* タイムライン ID の確認 (10 → 11) */
$ pg_controldata | grep -i timeline
最終チェックポイントの PrevTimeLineID:          11
```

1.4.5. レプリケーションスロット作成 (edb-serve2)

新マスターにてレプリケーションスロットを作成する。
メンテナンス作業中はこのレプリケーションスロットにて WAL を保全する。

実行例）

```
=# SELECT pg_drop_replication_slot('slot_slave0');
-[ RECORD 1 ]-------------+-
pg_drop_replication_slot |
```

1.4.6. 一時スレーブ起動およびスロット有効化 (edb-serve1)

スレーブとして起動するため、recovery.conf を作成する。
edb-serve2 の recovery.done を参考にホスト名とレプリケーションスロット名を変更する。

実行例）

```
/*  recovery.conf の作成   */
$  vi $PGDATA/recovery.conf
-----------------------------------------
standby_mode = on
primary_conninfo = 'host=edb-serve2 port=5444 user=repuser password=repuser'
recovery_target_timeline = latest
primary_slot_name = 'slot_slave0'
-------------------------------

/* タイムライン ID の確認  (10) */
$ pg_controldata | grep -i timeline         10

/*  recovery.conf があるため、一時スレーブとして起動   */
$ pg_ctl start
サーバの起動完了を待っています. .  ～中略～
サーバ起動完了

/*  状態を確認   */
$ pg_controldata | head -4
pg_control バージョン番号:                    1100
カタログバージョン番号:                       201809051
データベースシステム識別子:                   6685453832160810991
データベースクラスタの状態:                   アーカイブリカバリ中      ★スレーブ

/*  再起動してタイムライン ID の追いつきを実施   */
$ pg_ctl restart
サーバ停止処理の完了を待っています....完了
サーバは停止しました
サーバの起動完了を待っています....  ～中略～
サーバ起動完了

/* タイムライン ID の確認  (10 → 11) */
$ pg_controldata | grep -i timeline
最終チェックポイントの PrevTimeLineID:          11
```

1.4.7. 一時スレーブ停止 およびメンテナンス作業 (edb-serve1)

メンテナンス作業をするため、一時スレーブを停止する。

実行例）

```
/*  一時スレーブ停止   */
$ pg_ctl stop
サーバ停止処理の完了を待っています....完了
サーバは停止しました
```

この状態でメンテナス作業を実施する。

1.4.8. レプリケーションスロット削除 (edb-serve2)

メンテナンス作業終了後、レプリケーションスロットを削除する。

実行例）

```
=# SELECT pg_drop_replication_slot('slot_slave0');
-[ RECORD 1 ]------------+-
pg_drop_replication_slot |
```

1.4.9. 一時スレーブ起動（スロット無効化）および昇格 (edb-serve1)

一時スレーブを起動し、昇格させる。

実行例）

```
/*  recovery.conf からレプリケーションスロットの指定行をコメントアウト */
$ vi $PGDATA/recovery.conf
-----------------------------------
#primary_slot_name = 'slot_slave0'
-----------------------------------

/*  スレーブ起動  */
$ pg_ctl start
サーバの起動完了を待っています....   ～中略～
完了
サーバ起動完了

/*  昇格  （マスターに）  */
$ pg_ctl promote
サーバの昇格を待っています....完了
サーバは昇格しました

/*  状態を確認  */
$ pg_controldata | head -4
pg_control バージョン番号:                1100
カタログバージョン番号:                   201809051
データベースシステム識別子:               6685453832160810991
データベースクラスタの状態:               運用中                    ★マスター

/* タイムライン ID の確認 (11 → 12) */
$ pg_controldata | grep -i timeline
最終チェックポイントの PrevTimeLineID:    12
```

1.4.10. レプリケーションスロット作成 (edb-serve1)

マスターにてレプリケーションスロットを作成する。

実行例）

```
/*  レプリケーションスロット作成  */
=#  SELECT pg_create_physical_replication_slot('slot_slave1');
 pg_create_physical_replication_slot
-------------------------------------
 (slot_slave1,)
(1 行)
```

1.4.11. スレーブ起動（スロット有効化） (edb-serve2)

スレーブを起動し、レプリケーションスロットを有効化する。

実行例）

```
/*  recovery.conf からレプリケーションスロットの指定行を有効化  */
$ cd $PGDATA
$ mv recovery.done recovery.conf
$ vi $PGDATA/recovery.conf
-----------------------------------
primary_slot_name = 'slot_slave1'
-----------------------------------

/*  スレーブ起動  */
$ pg_ctl start
サーバの起動完了を待っています....  ～中略～
完了
サーバ起動完了

/*  再起動してタイムライン ID の追いつきを実施  */
$ pg_ctl restart
サーバ停止処理の完了を待っています....完了
サーバは停止しました
サーバの起動完了を待っています....  ～中略～
サーバ起動完了

/* タイムライン ID の確認  (11 → 12) */
$ pg_controldata | grep -i timeline
最終チェックポイントの PrevTimeLineID:         12
```

1.4.12. レプリケーションの状態を確認

レプリケーションに遅延がないことを確認する。

- state が streaming であること
- sent_lsn,write_lsn,flush_lsn,replay_lsn が一致していること
- wirte_lag,flush_lag,replay_lag が NULL であること

実行例）

```
=# SELECT * FROM pg_stat_replication ;
-[ RECORD 1 ]----+---------------------------------
pid              | 23260
usesysid         | 240687
usename          | repuser
application_name | walreceiver
state            | streaming
～中略～
sent_lsn         | B/B30003D8
write_lsn        | B/B30003D8
flush_lsn        | B/B30003D8
replay_lsn       | B/B30003D8
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async

=# SELECT slot_name,active FROM pg_replication_slots ;
-[ RECORD 1 ]----------
slot_name | slot_slave1
active    | t
```

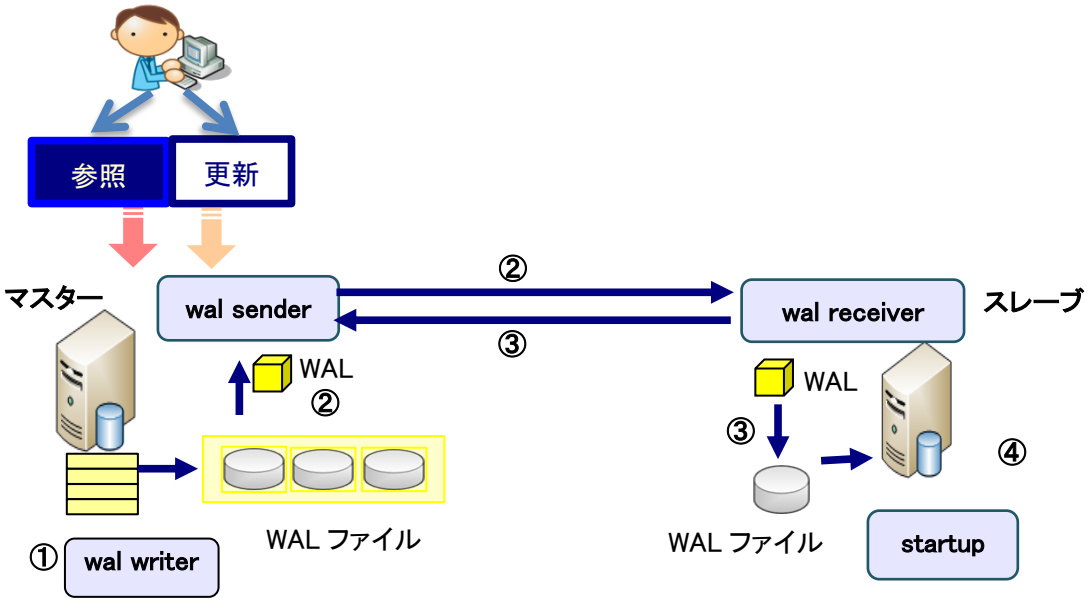
2. ストリーミングレプリケーション概要

2.1.ストリーミングレプリケーションの構成と目的

ストリーミングレプリケーションは可用性の向上を目的とした EDB Postgres / PostgreSQL の機能である。データベースの変更履歴が格納された WAL を操作単位でマスターからスレーブへ転送・適用することで物理的に同一のデータベースクラスタを構成する。マスターに障害が発生した際は、スレーブに切り替えることで運用への影響ををを最小限に抑えることができる。

また、物理的にデータをコピーしているためディスク障害へも対処でき、物理バックアップの代替としても有用である。更に、ホットスタンバイ機能を有効にすることでスレーブを参照用に活用できるため、マスターの負荷を軽減し、性能向上が見込める。

図 2-1 ストリーミングレプリケーション概要図



① **マスタの wal writer** プロセスが WAL レコードを WAL バッファから WAL ファイルへ書き出す（COMMIT）
② **マスタの wal sender** プロセスが WAL レコードを WAL ファイルから取り出しスレーブへ送信
非同期モードの場合は送信時点でマスターの COMMIT が完了
③ **スレーブの wal receiver** プロセスが WAL レコードを受け取り、WAL ファイルへ書き込む
同期モードの場合は書き込み完了時点でマスターのコミットが完了
④ **スレーブの startup** プロセスが WAL ファイルから WAL レコードを取り出してデータベースに適用スレーブの wal receiver プロセスと startup プロセスは個別に動作するため、WAL レコード適用の完了を待たずに次の WAL レコードの受信が可能である

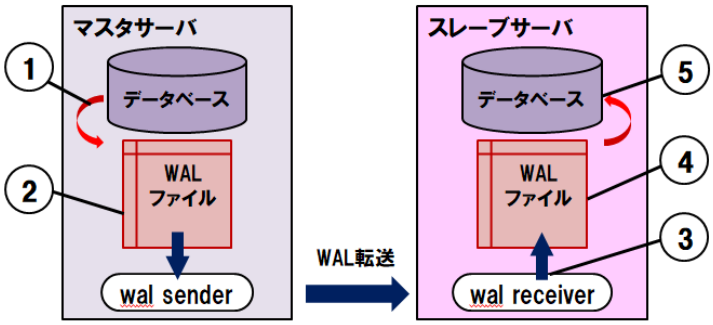
2.2.同期レベル

同期レベルは性能とデータ保護のバランスに優れた「同期」をベースとして検討する。

ストリーミングレプリケーションには以下の同期レベルがある。性能、データ保護、スレーブの参照利用（参照ロードバランス）の 3 要素を基に同期レベルを検討する。

表 2-1　ストリーミングレプリケーションの同期レベル

設定値	同期レベル	説明
remote_apply	完全同期	WAL 適用まで保証。データ保護優先と参照負荷分散。
on	同期	WAL 転送（ディスク書き込み）まで保証。データ保護優先。
remote_write	準同期	WAL 転送（メモリ書き込み）まで保証。データ保護優先。
local	非同期	ローカルの WAL 書き込みまで保証。性能優先。
off	完全非同期	ローカルの WAL 書き込みを保証しない。



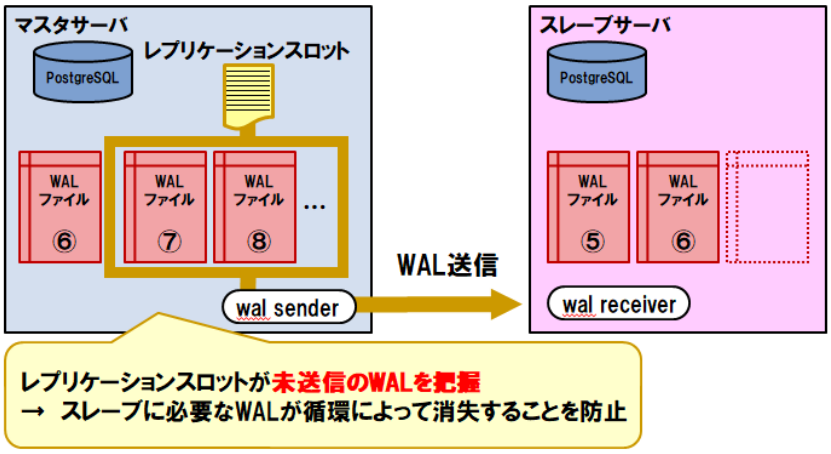
① off　：ローカルへの書き込みも保証しない
② local　：ローカルへの書き込みまで保証
③ remote_write　：WAL転送後メモリ書き込みまで保証
④ on　：WAL転送後ディスクへの書き込みまで保証
⑤ remote_apply　：WAL転送から適用まで保証

2.3.レプリケーションスロット

レプリケーションスロットを使用し、未転送WALファイルの上書き消失を防ぐ。

スレーブが一定期間以上停止した場合、wal_keep_segments パラメータを用いた管理では不足が生じる可能性がある。レプリケーションスロットはスレーブへの WAL 転送状況を把握し、WAL の要不要を判断する。そのため、wal_keep_segments パラメータでの固定数による管理と比較して過不足なく柔軟な WAL ファイルの管理ができる。特に複数スレーブ構成ではすべてのスレーブに転送されるまで WAL を保持するため、安心感がある。

図 2-2 レプリケーションスロットのイメージ



●レプリケーションスロット使用時の注意点

レプリケーションスロットは、一旦アクティブになると保存すべき最小の WAL 情報を記録する。その後に非アクティブになっても以前の古い情報のまま WAL を保持する機能は有効である。そのため、不要になったレプリケーションスロットは削除する必要がある。

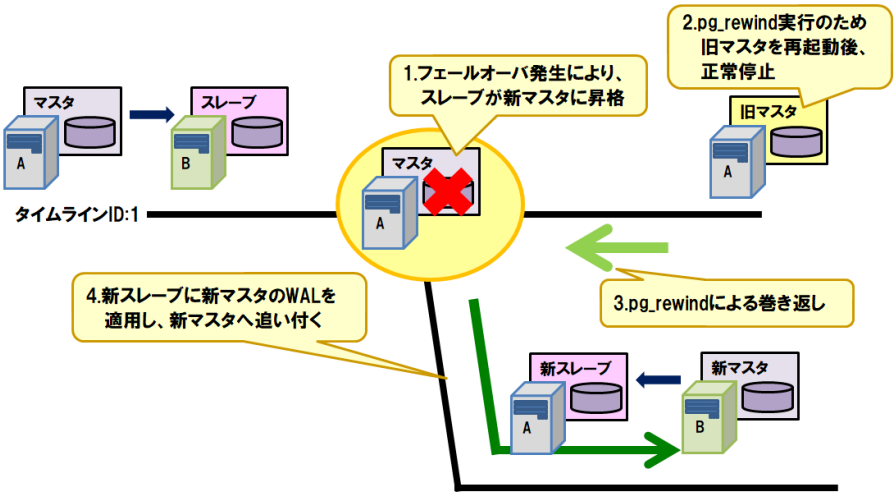
具体的には以下のケースで注意する。

- スレーブの除外
- スイッチオーバー

2.4.巻き戻し (pg_rewind)

タイムラインが枝分かれした旧マスターを再構成する際は、pg_rewindコマンドを利用する。

スレーブ昇格の直後はシングル構成に縮退する。高可用性構成に復旧するために旧マスターを新スレーブとし、新マスタとのレプリケーション構成とする必要がある。ただし、タイムラインが枝分かれした場合、従来は pg_basebackup コマンドでデータベースクラスタ全体のバックアップを再取得する必要があり、データ量によっては再構成に多くの時間を要していた。
しかし、pg_rewind コマンドが実装されたことで、旧マスターを枝分かれ前に巻き戻すことが可能となった。



pg_rewind 実行例）

```
$ pg_rewind -D $PGDATA --source-server="host=edb-server2 port=5444"
タイムライン 17 の WAL 位置 B/A9000140 でサーバが分岐しています
タイムライン 17 の B/A9000098 で最新の共通チェックポイントから巻き戻しています
完了！
```

この後は recovery.conf を作成しスレーブとして起動できる（必要に応じてレプリケーションスロットを作成する）。

■pg_rewind 使用時の注意点

巻き戻しには枝分かれ直後からの WAL が必要であるが、新マスターにて一定量のトランザクションを実行すると上書きされ消失する可能性がある。昇格後、あまり間を置かずに実行するのが重要である。

新マスター側の WAL 消失によるエラー）

```
ERROR: requested WAL segment 0000000D00000000000000F3 has already been removed
```

また、自分自身（旧マスター）の WAL も必要であるため、スプリットブレイン状態で一定量の更新処理を行った場合に WAL が消失し、以下のエラーが発生する。

旧マスター側の WAL 消失によるエラー）

```
could not open file "/data/edb11/data/pg_wal/0000000D000000020000000CF": No such file or directory
could not find previous WAL record at 2/CF000140 Failure, exiting
```

pg_rewind は、ターゲットとソースクラスタのタイムライン ID が枝分かれした場合に必要となる。
枝分かれしていない場合は実行できない。
必要性や実行可否の事前チェックができる（--dry-run オプション）。

pg_rewind を実行するための条件（実行する必要のある場合）を以下にまとめる。

- wal_log_hints = on または データベースチェックサムが有効化されていること
- スレーブのマスター昇格に伴い、タイムラインの枝分かれが発生していること
- 旧マスターが正常に起動/停止ができること
- 旧マスターおよび新マスターにタイムラインが枝分かれして以後の WAL が存在すること

recovery.conf が存在する場合、pg_rewind 終了時点で recovery.done に変更される。

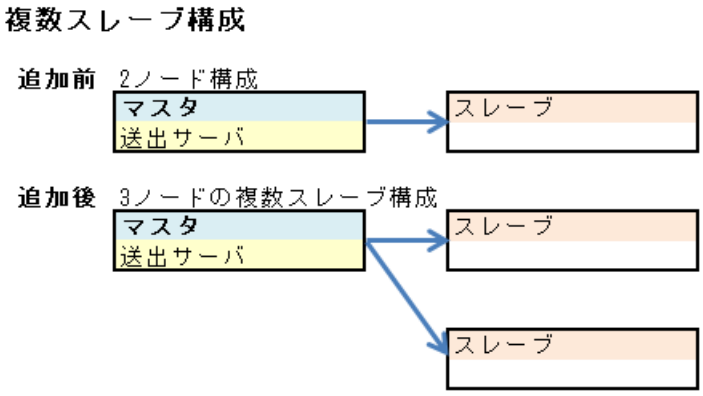
2.5.複数スレーブ構成

スレーブを用途によって使い分けたい場合、複数スレーブ構成を利用を検討する。

単一マスターに対して複数のスレーブを構成し、一元管理することが可能である。同期モードはスレーブ毎に同期/非同期を選択できる。また、クォーラムコミットにより任意の複数台を同期とする設定も可能である。

スレーブ 1 は「完全同期モードで参照用」、スレーブ 2 は「非同期モードでバックアップ用」のように各スレーブに違った役割を持たせたい場合に利用を検討する。

図 2-3 複数スレーブ構成イメージ図



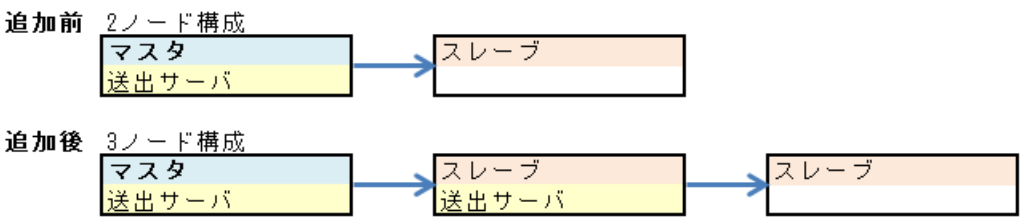
2.6.カスケード構成

ディザスタリカバリ環境としてスレーブを遠隔地に配置したい場合、利用を検討する。

ストリーミングレプリケーションでは、カスケード構成（親→子→孫）をとることも可能である。
「子」にあたるスレーブは送出ノードとなるため、wal sender プロセスが起動する。マスターは「孫」スレーブを管理する必要はない（管理できない）点に注意する。

「孫」スレーブをディザスタリカバリ環境として遠隔地に配置したい場合等に利用を検討する。

図 2-4 カスケード構成イメージ図



3. ストリーミングレプリケーションの構築

本章では、基本的な２ノード（マスター1、スレーブ１）構成のストリーミングレプリケーション環境構築手順を記載する。具体的な構成は下表のとおり。

役割	ホスト名	IP アドレス	備考
マスター	edb-server1	192.168.150.21	データベースクラスタ作成済
スレーブ	edb-server2	192.168.150.22	データベースクラスタは作成されていない

なお、複数スレーブ構成やカスケード構成（マスター1、スレーブ 2）も基本的な構築手順は同様である。ただし、カスケード構成の場合、マスターと接続されたスレーブ１は送出サーバでもある。

3.1.パスワードなしの ssh 実行環境の設定

以下の機能で必要なため、パスワードなしで実行可能なssh環境を設定する。
- recovery.confのrestore_commandでscpを実行する。
- 障害発生時にsshで相手ノードに対してRPCを実行する。

実行例）

```
# edb-server1  generate .ssh directory and pub key
su - enterprisedb
ssh-keygen -t rsa
cd .ssh

# edb-server2  generate .ssh directory and pub key
su - enterprisedb
ssh-keygen -t rsa
cd .ssh

# edb-server1  create authorized_keys  and non-pass ssh
touch authorized_keys
chmod 600 authorized_keys

cat id_rsa.pub >> authorized_keys
ssh edb-server2 cat /home/enterprisedb/.ssh/id_rsa.pub >> authorized_keys
scp /home/enterprisedb/.ssh/authorized_keys edb-server2:/home/enterprisedb/.ssh/

ssh edb-server1 date
ssh edb-server1 date

ssh edb-server2 date
ssh edb-server2 date

# edb-server2   non-pass ssh
ssh edb-server1 date
ssh edb-server1 date

ssh edb-server2 date
ssh edb-server2 date
```

3.2.レプリケーション用パラメータの設定 （edb-server1）

役割切り替え後もそのまま使用できるようpostgresql.confはマスター/スレーブ共に同一のものを使用する。

マスター/スレーブの役割切り替え時にもそのまま利用できるよう、スレーブの postgresql.conf は pg_basebackup 等でマスターのコピーを作成する。パラメータは下表のとおりマスター/スレーブ両役割のものを予め設定しておく。なお、自ノードの役割とは異なるパラメータは無視される。

表 3-1 ストリーミングレプリケーション用パラメータ設定値

パラメータ	設定値	説明
listen_address	0.0.0.0	通常の設定で良い。変更反映には再起動が必要。
wal_level	replica	WAL に書かれる情報量を指定。変更反映には再起動が必要。 SR 構成ではデフォルトの replica 。
wal_keep_segments	32	暫定値として 32 を設定。変更反映はリロードで可能。 レプリケーションスロットを使用する場合は設定不要。
synchronous_commit	on	同期レベルの要件に応じて設定する。変更反映はリロードで可能。 remote_apply　完全同期。WAL 適用までを保証。 on　同期。WAL のディスク書込みを保証。 remote_write　準同期。WAL のメモリ書込みを保証。 local　非同期。WAL のローカルへの書込みを保証。 off　完全非同期。WAL のローカルへの書込みも保証しない。
synchronous_standby_names	*	同期/非同期の切り替えに使用。変更反映はリロードで可能。 同期の場合　'' 非同期の場合　* または同期スレーブの application_name
max_wal_senders	10	wal sender プロセスの最大数を指定。変更反映には再起動が必要。 デフォルトの 10 で良い。
max_replication_slots	10	レプリケーションスロットの最大数を指定。変更反映には再起動が必要。 デフォルトの 10 で良い。
restart_after_crash	off	インスタンス障害時の再起動有無を指定。変更反映はリロードで可能。 スプリットブレイン状態を回避するため、自動再起動を行わない。
wal_log_hints	on	pg_rewind を有効化。
hot_standby	on	死活監視にも必要なホットスタンバイ機能を有効化（デフォルト）。
hot_standby_feedback	on	以下のいずれかを使用する場合は有効化。 ・レプリケーションスロット ・VACUUM 待機（vacuum_defer_cleanup_age）

3.3.レプリケーション用ユーザの作成 （edb-server1）

レプリケーション属性を持つ専用ユーザを作成する。認証方式はscram-sha-256とする。

レプリケーション用のユーザは enterprisedb ユーザや postgres ユーザ等のスーパーユーザ属性を持つユーザも使用可能だが、セキュリティ面や管理面を考慮し、レプリケーション属性を持つ専用のユーザを作成する。

実行例）

```
/*  パスワード暗号化を scram-sha-256 に指定  */
=# SET password_encryption TO 'scram-sha-256';
SET

/*  レプリケーション属性を持つユーザを作成   */
=# CREATE USER repuser WITH REPLICATION PASSWORD 'repuser';
CREATE ROLE
```

3.4.レプリケーション用クライアント認証の設定（edb-server1)

レプリケーション用ユーザのクライアント認証設定を追加する。
接続先データベースには reprication（仮想データベース）を指定する。

設定例）

```
host    replication     repuser         192.168.150.21/32       scram-sha-256
host    replication     repuser         192.168.150.22/32       scram-sha-256
```

3.5.ディレクトリの確認 (edb-server1)

pg_basebackup の実行によりスレーブ（edb-server2）側に作成されるディレクトリの構成やテーブルスペースの有無を確認する。

■データベースクラスタ作成領域（$PGDATA）

確認例）

```
$ env | grep PGDATA=
PGDATA=/opt/as11/data/

$ ls -l /opt/as11 | grep data
drwx------ 23 enterprisedb enterprisedb 4096  3 月  9 10:31 data
```

■WAL 出力先

WAL 出力先はパラメータで設定されておらず、内部的には「$PGDATA/pg_wal」固定されている。
initdb 実行時のオプションで WAL 出力先を変更した場合は、シンボリックリンクで設定される。

確認例）

```
$ ls -l $PGDATA | grep pg_wal
drwxr-xr-x  3 enterprisedb enterprisedb   4096  3 月  2 23:08 pg_wal
```

■テーブルスペース

確認例）

```
=# ¥db
                        テーブル空間一覧
    名前    |    所有者    |                場所
------------+--------------+---------------------------------
 pg_default | enterprisedb |
 pg_global  | enterprisedb |
 user_space | enterprisedb | /data/as11/tablespace/user_space
(3 行)
```

3.6.pg_basebackup 実行前の準備作業 (edb-server2)

pg_basebackup の実行に必要なディレクトリの作成および適切なパーミッション設定を行う。

■データベースクラスタ作成領域/WAL 出力先
enterprisedb ユーザが「3.5.ディレクトリの確認」で確認したディレクトリを作成できるよう、パーミッションを設定する。

実行例）

```
# mkdir  -p /opt/as11/data
# cd /opt
# chowner -R enterprisedb:enterprisedb as11
```

■テーブルスペース
enterprisedb ユーザが「3.5.ディレクトリの確認」で確認したディレクトリを作成できるよう、上位ディレクトリの作成とパーミッションの設定を行う。

実行例）

```
# mkdir  -p /data/as11/tablespace
# cd /data
# chowner -R enterprisedb:enterprisedb as11
```

3.7.pg_basebackup 実行（edb-server2）

スレーブ（edb-server2）で pg_basebackup を実行し、データベースクラスタをコピーする。
ただし、パスワードファイルは対象外であるため、別途コピーする。

表 3-2　pg_basebackupコマンドのパラメータ

パラメータ	説明
-D	ファイル出力先を指定。マスターと異なるディレクトリを設定することも可能。
-h	マスターのホスト名を指定。
-p	マスターのデータベースクラスタのポート番号を指定。
-U	マスターのデータベースへの接続ユーザを指定。
--format , -F	バックアップファイルのフォーマットを指定（plain or tar）。
--checkpoint, -c	バックアップ中のチェックポイント処理を I/O 分散実行するか、即実行するかを指定 （fast or spread）。
--wal-method , -X	WAL ファイルをバックアップする場合のタイミングを指定。 f（fetch）　WAL ファイルはバックアップの最後に取得 s（stream）バックアップ取得中に WAL をストリーム（並列処理）
--waldir	WAL ファイルの出力先を指定。マスターと異なるディレクトリを設定することも可能。
--label , -l	バックアップラベル名を指定。
--gzip , -z	tar ファイルのフォーマットを圧縮する場合に指定。
--progress, -P	バックアップの進捗状況を表示する場合に指定。
--verbose, -v	バックアップ開始・終了時点の WAL の位置を表示する場合に指定。

実行例）

```
/*  pg_basebackup の実行  */
$ pg_basebackup -h edb-server1 -p 5444 -U repuser -D $PGDATA -X s -P
1861207/1861207 kB (100%), 2/2 テーブル空間

/*  パスワードファイルのコピー  */
$ scp edb-server1:~/.pgpass~/
```

3.8.レプリケーションスロットの作成

メンテナンス時など、一時的にWALの転送を停止している際にWALが上書きされないようレプリケーションスロットを作成する。

レプリケーションスロットの詳細については、「2.4.レプリケーションスロット」を参照。

実行例）

```
/*  レプリケーションスロットの作成        */
/*  スロット作成時にはスレーブ情報は不要  */
=# SELECT pg_create_physical_replication_slot('slot_slave1');
-[ RECORD 1 ]----------------------+---------------
pg_create_physical_replication_slot | (slot_slave1,)

/*  レプリケーションスロットの確認        */
/*  作成直後は非アクティブ                */
=# SELECT * FROM pg_replication_slots ;
-[ RECORD 1 ]-------+------------
slot_name           | slot_slave1
plugin              |
slot_type           | physical
datoid              |
database            |
temporary           | f
active              | f
active_pid          |
xmin                |
catalog_xmin        |
restart_lsn         |
confirmed_flush_lsn |
```

3.9.recovery.conf の作成　(edb-server2)

下表を参照し、スレーブの設定ファイル(recovery.conf)にマスターへの接続と WAL 確保方法を設定する。なお、接続先のデータベースは replication 疑似データベースであるため、指定は不要である。

表 3-3 recovery.confの設定項目

パラメータ	設定例	説明
standby_mode	on	スタンバイモード(固定)
primary_conninfo	host=edb-server2	マスターのホスト名
	port=5444	受付ポート
	user=repuser	レプリケーション用ユーザ
	password=repuser	レプリケーション用ユーザの パスワード
	application_name='slave1'	任意のアプリケーション名を指定 複数スレーブ時に判別に使用
recovery_target_timeline	latest	最新のタイムラインまで
primary_slot_name (*1)	slot_slave1	レプリケーションスロット名
restore_command (*1)	'scp edb-server1:/archive/%f %p'	マスター側のアーカイブをコピー

(*1) primary_slot_name と restore_command は両方指定する必要はない。

設定例)

```
standby_mode = on
primary_conninfo = 'host=edb-server1 port=5444 user=repuser password=repuser'
recovery_target_timeline = latest
primary_slot_name = 'slot_slave1'
```

3.10. スレーブの起動

recovery.conf が存在する状態でデータベースクラスタを起動すると、スレーブ特有のプロセス（wal receiver / startup）が起動する。マスターの wal sender プロセスはスレーブ起動時点で起動する。

実行例）

```
/*  スレーブ側の起動  */
$ pg_ctl start
サーバの起動完了を待っています.
    ～中略～
サーバ起動完了

/*  データベースクラスタの状態を確認  */
$ pg_controldata | head -4
pg_control バージョン番号:                1100
カタログバージョン番号:                   201809051
データベースシステム識別子:               6685453832160810991
データベースクラスタの状態:               アーカイブリカバリ中        ★

/*  スレーブ用プロセス確認      */
ps -ef | grep postgres | grep enterpr | grep -v grep
enterpr+  85225      1  0 12:55 pts/0    00:00:00 /usr/edb/as11/bin/edb-postgres
enterpr+  85226  85225  0 12:55 ?        00:00:00 postgres: logger
enterpr+  85227  85225  0 12:55 ?        00:00:00 postgres: startup
enterpr+  85231  85225  0 12:55 ?        00:00:00 postgres: checkpointer
enterpr+  85232  85225  0 12:55 ?        00:00:00 postgres: background writer
enterpr+  85233  85225  0 12:55 ?        00:00:00 postgres: stats collector
enterpr+  85234  85225  0 12:55 ?        00:00:00 postgres: walreceiver
```

4. ストリーミングレプリケーションの監視

4.1.レプリケーションの確認

レプリケーションの状態は、マスターのpg_stat_replicationビューで確認する。

スレーブ毎に 1 行表示され、行が返ることでレプリケーション状態であることが確認できる。
sent_lsn および write_lsn などからレプリケーションの遅延有無を確認できる。

実行例）

```
■スレーブが 1 台起動している場合
=# SELECT * FROM pg_stat_replication ;
-[ RECORD 1 ]----+---------------------------------
pid              | 3090
usesysid         | 240687
usename          | repuser
application_name | walreceiver
client_addr      | 192.168.150.22
client_hostname  |
client_port      | 47180
backend_start    | 09-FEB-20 10:34:40.748307 +09:00
backend_xmin     |
state            | streaming
sent_lsn         | B/A9000060
write_lsn        | B/A9000060
flush_lsn        | B/A9000060
replay_lsn       | B/A9000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
（1 行）

■スレーブが一台も起動していない場合
=# SELECT * FROM pg_stat_replication ;
（0 行）
```

表 4-1　pg_stat_replicationビュー

列名	説明
pid	バックエンドプロセスのプロセス ID。
usesysid	バックエンドプロセスのユーザ ID。
usename	レプリケーションのユーザ名。本例では‘repuser’。
application_name	スレーブの recovery.conf の primary_conninfo パラメータ内 の application_name の値が表示される。 本例では省略しているためデフォルトの'walreceiver’。
client_addr	スレーブの IP アドレス。SR 構成ではスレーブがクライアントとなるため、 edb-server1 がマスタの場合は edb-server2 がスレーブとなる。
client_hostname	スレーブのホスト名。 本例では log_hostname が off(デフォルト)であるため NULL。 log_hostname を on に設定すると名前解決のため一定の負荷がかかる。
client_port	スレーブの接続ポート番号。 例： 22319 (不定)
backend_start	バックエンドプロセスの開始時刻(スレーブインスタンスの起動時刻)。
backend_xmin	hot_standby_feedback パラメータ(デフォルト：off)が有効の場合に、 マスターに報告されたスレーブ上で実行中の最小トランザクション ID。
state	WAL 送信プロセス（wal sender process）の状態を示す。 startup　：起動中 catchup　：過去の更新履歴を取得しつつ、最新状態へ追いつき中 streaming ：ストリーミング状態で最新の変更を適用中（通常時） backup　：pg_basebackup によるバックアップ中 stopping　：停止中 例： streaming
sent_lsn	送信された最後の WAL の先行書き込みログ位置。
write_lsn	スレーブのディスクに書き込みされた最後の WAL 先行書き込みログ位置。
flush_lsn	スレーブのディスクに吐き出された（フラッシュ）最後の WAL 先行書き込みログ位置。
replay_lsn	スレーブ DB に適用された最後の WAL の先行書き込みログ位置。
write_lag	ローカルへの WAL 書き込みからスレーブへの書き込みまでの経過時間。 synchronous_commit=remote_write の遅延測定に有効。
flush_lag	ローカルへの WAL 書き込みからスレーブへのフラッシュまでの経過時間。 synchronous_commit=remote_write の遅延測定に有効。
replay_lag	ローカルへの WAL 書き込みからスレーブ DB に適用されるまでの経過時間。 synchronous_commit=remote_apply の遅延測定に有効。
sync_priority	複数スレーブの場合の同期レプリケーションモードのプライオリティ。 1 以上　：同期スレーブのプライオリティ。 synchronous_standby_names パラメータに名前がある ノードに左から順に 1 から昇順で割り当てられる。 0　：非同期ノードのプライオリティ
sync_state	現在のレプリケーション方式 async：非同期転送モード sync　：同期転送モード potential　：非同期で転送中でより上位のプライオリティを持つ ノードとの接続がなくなった場合には同期転送に昇格 quorum　：クオーラムの同期スレーブ候補

4.2.レプリケーションスロットの確認

レプリケーションスロットの状態は、マスターのpg_replication_slotsビューで確認する。

active 列が t であれば、スロットが使用されている（スレーブが起動している）ことを示す。
スレーブが停止した際は active 列は f となり、未転送の WAL が消失することを防ぐ。

実行例）

```
/*  レプリケーションスロットの確認    */
=# SELECT * FROM pg_replication_slots ;
-[ RECORD 1 ]-------+------------
slot_name           | slot_slave1
plugin              |
slot_type           | physical
datoid              |
database            |
temporary           | f
active              | t                ← アクティブ
active_pid          | 5177
xmin                |
catalog_xmin        |
restart_lsn         | B/A9000060
confirmed_flush_lsn |
```

表 4-2　pg_replication_slotsビュー

列名	説明
slot_name	クラスタ間で一意なレプリケーションスロット名（オブジェクト名であるため小文字）
plugin	出力プラグインに使用されている論理スロット。 物理スロットの場合は NULL を含む共有オブジェクトの基底名
slot_type	スロットのタイプ - physical（物理）または logical（論理）
datoid	このスロットと関連しているデータベースの OID、または null。 論理スロットだけがデータベースと関連を持つ。
database	このスロットと関連しているデータベース名、または null。 論理スロットだけがデータベースと関連を持つことができます。
temporary	一時レプリケーションスロットの場合に、真。
active	このスロットが現在アクティブで使用されている場合に、真。
active_pid	現在アクティブで使用されている場合は、スロットを使用しているセッションのプロセス ID。アクティブでなければ NULL。
xmin	このスロットがデータベースとの接続を必要としている最も古いトランザクション。
catalog_xmin	このスロットがデータベースとの接続を必要としている、システムカタログに影響する最も古いトランザクション。
restart_lsn	このスロットの利用者に必要かもしれないため、チェックポイント中に自動除去されない、もっとも古い WAL の（LSN）アドレス。
confirmed_flush_lsn	データの受信を確認できている論理スロットのアドレス。 本システムは物理スロットであるため確認不要。

4.3.スレーブからのレプリケーション確認

ホットスタンバイが有効な場合、スレーブからもレプリケーションの状態が確認できる。
確認の方法には pg_stat_wal_receiver ビューでの確認とリカバリ情報を取得する関数を利用する方法の 2 種類がある。

4.3.1. pg_stat_wal_receiver ビュー

pg_stat_wal_receiver ビューでは、1 行に wal receiver プロセスの統計情報が表示される。

実行例）

```
/*  wal receiver の状態確認  */
=# SELECT * FROM pg_stat_wal_receiver ;
pid                   | 87610
status                | streaming
receive_start_lsn     | B/BE000000
receive_start_tli     | 17
received_lsn          | B/BEB4C710
received_tli          | 17
last_msg_send_time    | 10-MAR-20 09:13:18.111862 +09:00
last_msg_receipt_time | 10-MAR-20 14:56:08.420374 +09:00
latest_end_lsn        | B/A9000060
latest_end_time       | 10-MAR-20 08:54:47.113921 +09:00
slot_name             | slot_slave1
sender_host           | edb-server1
sender_port           | 5444
connin                | user=repuser password=******** dbname=replication host=edb-server1 port=5444
fallback_application_name=walreceiver      sslmode=prefer      sslcompression=0      krbsrvname=postgres
target_session_attrs=any
```

表 4-3　pg_stat_wal_receiverビュー

列名	説明
pid	WAL レシーバプロセスのプロセス ID
status	WAL レシーバプロセスの活動状態
receive_start_lsn	WAL レシーバ開始時に使われるトランザクションログの最初の位置
receive_start_tli	WAL レシーバ開始時に使われる初期タイムライン番号
received_lsn	ディスクにフラッシュされたトランザクションログの最新位置
received_tli	ディスクにフラッシュされたトランザクションログのタイムライン ID
last_msg_send_time	オリジン WAL センダから受信した最新メッセージの送信時間
last_msg_receipt_time	オリジン WAL センダから受信した最新メッセージの受信時間
latest_end_lsn	オリジン WAL センダへ報告された最新のトランザクションログ位置
latest_end_time	オリジン WAL センダへ最新のトランザクションログ位置が報告された時間
slot_name	使用されているレプリケーションスロット名
sender_host	WAL レシーバーが接続しているインスタンスのホスト
sender_port	WAL レシーバーが接続しているインスタンスのポート番号
connin	WAL レシーバによって使用された接続文字列

4.3.2. リカバリ情報取得関数

スレーブからリカバリ情報を取得する関数が実装されており、これらを利用することでレプリケーションの進行状況が確認できる。

実行例） pg_last_wal_receive_lsn 関数での確認

```
# SELECT pg_last_wal_receive_lsn(),
       pg_last_wal_replay_lsn(),
         pg_last_xact_replay_timestamp(),         -- マスタのトランザクション時刻
         sysdate;                                 -- スレーブの現在時刻
-[ RECORD 1 ]----------------+--------------------------------
pg_last_wal_receive_lsn       | B/BEB4C710
pg_last_wal_replay_lsn        | B/BEB4C710
pg_last_xact_replay_timestamp | 10-MAR-20 20:28:29.389006 +09:00
sysdate                       | 10-MAR-20 20:34:19
```

表 4-4 リカバリ情報取得関数

関数	説明
pg_last_wal_receive_lsn	受信されディスクに書込みされた最後の WAL 位置を取得。 pg_stat_replication ビューの write_lsn 列に相当。
pg_last_wal_replay_lsn	リカバリ中に適用された最後の WAL の位置を取得。 pg_stat_replication ビューの replay_lsn 列に相当。
pg_last_xact_replay_timestamp	リカバリ中に再生された最後のトランザクション時刻を取得。 マスター側で COMMIT/ROLLBACK された時刻。 pg_stat_replication ビューには相当する列はない。

5. バックアップ・リカバリ

本章では EDB Postgres 高可用性パック（Hot Cluster Pack）（以下、HCP）におけるバックアップ・リカバリの仕様を説明する。

5.1.システム構成

HCP の特徴を以下に記す。

- 3 ノードのレプリケーション構成である。クォーラムコミットを採用しており、スレーブ 2 台のうち任意の 1 台が残存すれば問題ない構成である。
- Enterprise Failover Manager（以下、EFM）による障害検知および対処を実装している。
- ネットワークはパブリック・ネットワークとプライベート・ネットワークの 2 系統を備え、バックアップは管理サーバからプライベートネットワークを使用して取得する。

図 5-1 EDB Postgres高可用性パック（HCP）構成イメージ

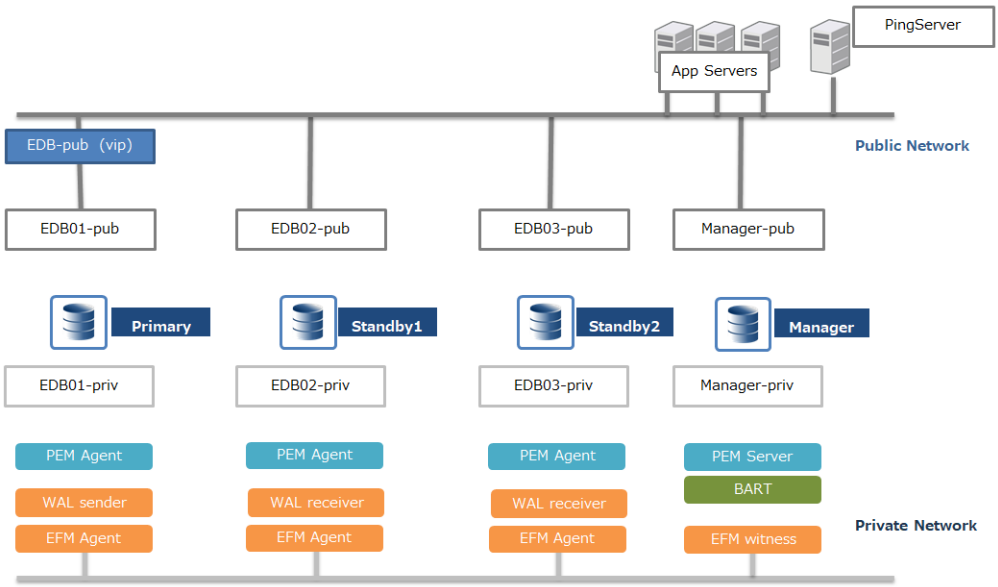


表 5-2 EDB Postgres高可用性パック（HCP）のネットワーク構成

項目	説明
パブリック・ネットワーク	アプリケーションからのアクセスに使用
プライベート・ネットワーク	バックアップ取得、アーカイブ転送、および正常性監視に使用

5.2.物理バックアップ

物理バックアップはBackup and Recovery Tool(BART)を使用して取得する。

BART は EDB Postgres に付属のバックアップツールであり、以下の機能を備えている。
- バックアップの一括管理
- バックアップの世代管理
- 増分バックアップ

5.2.1. バックアップの仕様

表 5-3 バックアップの仕様

項目	内容
バックアップツール	Backup and Recovery Tool (BART)
バックアップ方法	オンラインバックアップ
バックアップ頻度	1 日 1 回
バックアップ単位	フルバックアップ
バックアップ対象ノード	マスターノードで取得する。 3 ノードの中からマスタを選別し、バックアップのターゲットとする。
ネットワーク	プライベート・ネットワークを使用する。 これによりアプリケーションへの影響を最小限とする。

5.2.2. バックアップ・リカバリ要件

表 5-4 バックアップ・リカバリ要件

リカバリが必要となる障害ケース	復旧時点(RPO)	復旧時間(RTO)
WAL ファイルを除く データファイルの破損	障害発生直前、 または 1 日以内の任意の時刻	データサイズおよび 日次の更新量に依存
データファイルの全損	アーカイブ取得時点	

5.2.3. バックアップ方針

全てのバックアップ・リカバリ要件を満たす、以下の方針でバックアップを取得する。
- 全ての想定障害ケースに対応可能なバックアップを日次で取得
- バックアップは一次バックアップを管理サーバに保管し、2 世代保管

6. HCP 構成における有事対策

6.1.対障害設計

HCP では、EFM による障害検知と対処を実装している。

6.1.1. プロセス障害

DBプロセスの障害はEFMが検知し、フェイルオーバーを実施する。

表 6-1 プロセス障害

対象プロセス	説明
DB プロセス	EFM で検知し、AP アクセスを再開するよう自動で状態遷移（フェイルオーバーまたは縮退）
EFM Agent	EFM で相互検知するが、DB プロセスが稼働していれば AP アクセスに影響ない

6.1.2. 仮想マシン障害および物理ホスト障害

仮想マシン障害および物理ホスト障害は最終的にDBプロセス障害とみなされ、クラスタ状態を遷移する。

6.1.3. ネットワーク障害

障害発生経路を使用しているプロセス監視に失敗することで検知し、管理者に通知する。

表 6-2 ネットワーク障害

対象ネットワーク	説明
パブリック・ネットワーク	PEM Agent が応答しないことで、PEM サーバによる死活監視が失敗
プライベート・ネットワーク	EDB Postgres の機能以外で監視する

各サーバの NIC 故障は PEM で検知できないケースがある。
- 物理環境では NIC を冗長化し、NIC の故障を検知するよう OS レイヤの監視に組込む。
- 仮想環境で仮想 NIC の故障への対応が必要な場合は、他のツールで監視する。

6.1.4. ディスク障害

故障ブロックへアクセス時のDBプロセス停止をプロセス障害として検知する。

EDB Postgres にはディスク障害に対して即座に検知・通知する機能が実装されていない。
故障ブロックへアクセスした際に DB プロセスが停止するため、これをプロセス障害として検知する。

6.2.インスタンス障害時の挙動

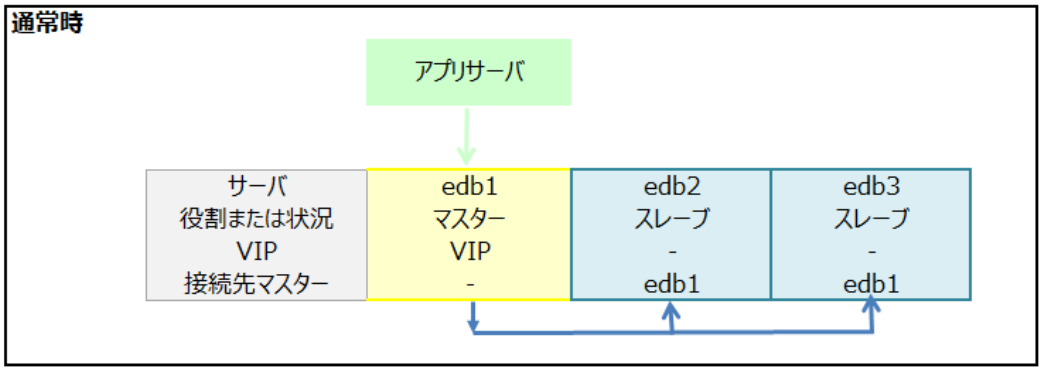
6.2.1. インスタンス障害時のコンポーネント別まとめ

各コンポーネント（ノード）毎にインスタンス障害が発生した際の挙動と対処を下表に記す。

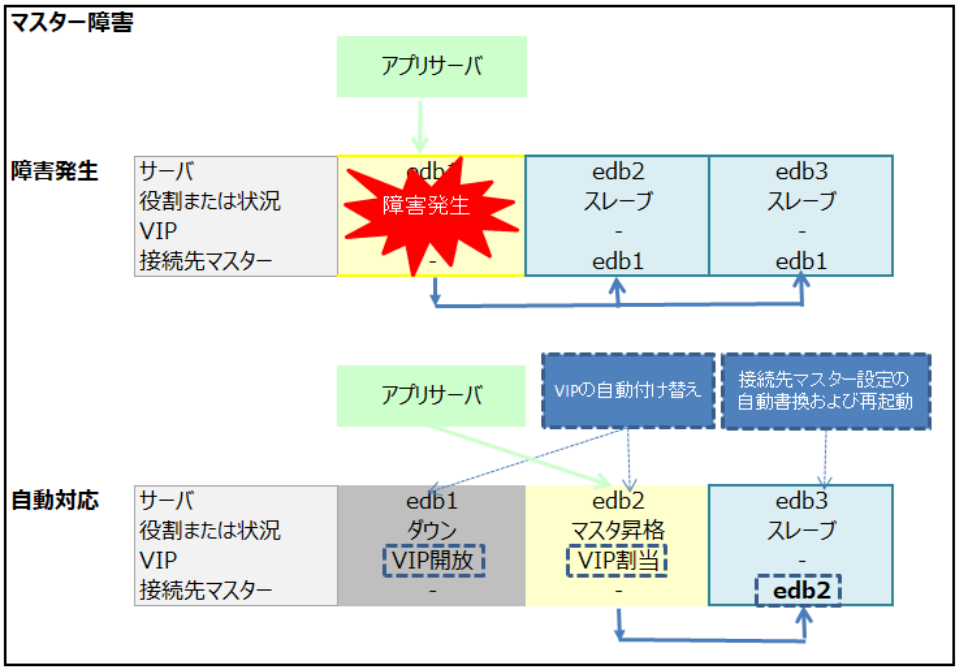
表 6-3 障害ノード別の挙動

障害ノード	挙動	対処
マスター障害	2 台のスレーブのうち、任意のスレーブにフェイルオーバーする。	新マスターに仮想 IP が付与され、運用が再開される。
スレーブ障害（1 台目）	クオーラムコミットのため、スレーブが残存する場合は問題ない。	運用への影響がないため、対処は不要。
スレーブ障害（2 台目）	スレーブが 1 台も存在しない場合、マスターの更新処理がハングする。	手動で非同期モードに変更する必要がある。
管理サーバ	監視が停止するのみであり、AP へ影響はない。	対処は不要。

通常時のイメージを以下の図で示す。



6.2.2. マスター障害



■昇格スレーブの選定
マスター障害時に昇格可能なスレーブが複数存在する場合は、昇格スレーブの選定を以下の 2 段階で実施する。

1. 最新スレーブ
   ストリーミングレプリケーション情報を取得し、どのスレーブに最新データがあるかを確認する。データ損失の可能性が最も低いスレーブを選定する。
2. 優先スレーブ
   最新データを有するスレーブが複数存在する場合、ユーザーが指定した優先順位の値が高いスレーブが選定される。

■残存スレーブの新マスターへの接続先変更
残存するスレーブの接続先のマスター情報の変更を行う。
上図例では、接続先マスターは edb1 からマスターに昇格した edb2 に変更される。

表 6-4　マスター障害におけるマスター接続先情報の変更

タイミング	edb1	edb2	edb3	同期ノード数
マスター障害発生前	マスター	edb1	edb1	2　ノード
マスター障害発生後	停止	マスター	edb2	1　ノード

表 6-5 残スレーブの自動再構成の有効/無効

パラメータ	デフォルト値	パラメータの意味
auto.reconfigure	true	マスター障害時に昇格スレーブ以外のスレーブを自動的に新マスターのスレーブにする

■マスターへの接続先設定変更および再起動
recoveryconf の priamry_conninfo パラメータが変更される。
設定を反映するために、残存スレーブ(edb3)の再起動が行われる。

■VIP の付け替え
マスター停止に伴い、VIP の付替えを行う。
これにより昇格後はアプリケーションが新マスターに接続される。

1. edb1 に割り当てられた VIP を開放
2. 昇格スレーブに選定された edb2 に割当

VIP の付け替えは新マスタの EFM Agent（マスターエージェント）が行う。
マスターエージェントを停止しても VIP は削除されないため、エージェントが停止しても運用には支障はない。

■監視間隔
EFM の監視設定は efm.properties に記載される以下のパラメータで設定できます。
インスタンス停止であれば即座にエラーが戻るが、ネットワーク停止など反応がない場合には以下のプロパティによるタイムタウトまで待機することで障害と見なす。

表 6-6 エージェント間のタイムアウト

パラメータ	デフォルト値	パラメータの意味
node.timeout	50	エージェントからの応答がない場合に、障害と判断するまでの時間

6.2.3. スレーブ障害（1台目）

1 台目のスレーブ障害が発生しても同期スレーブ数は 1 となり、充足数を満たしている。
更新も正常に処理できるため、緊急の対応は不要である。

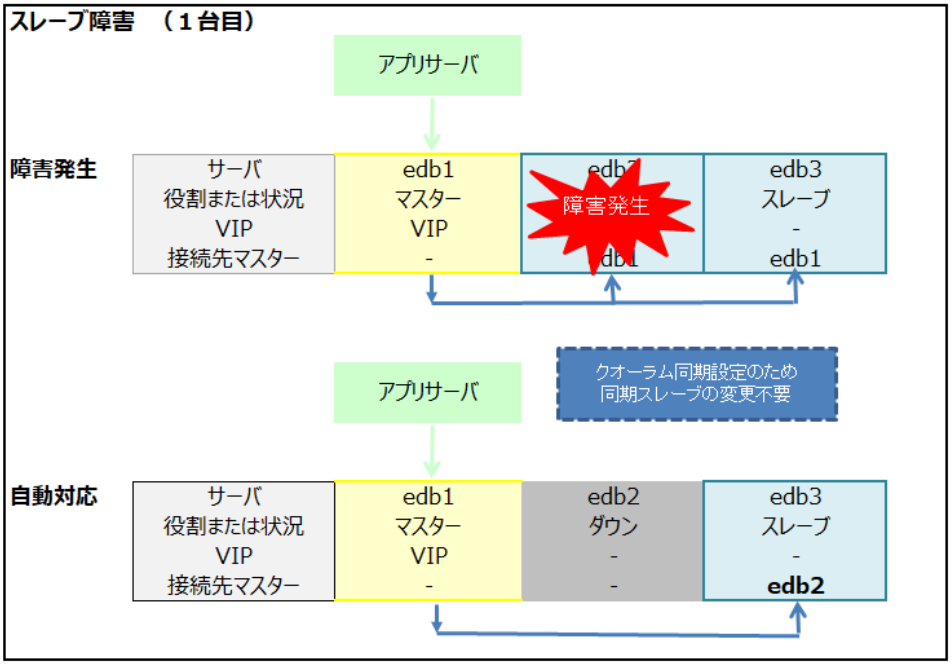


表 6-7 スレーブ障害（1台目）におけるマスター接続先情報の変更

タイミング	edb1	edb2	edb3	同期ノード数
マスター障害発生前	マスター	edb1	edb1	2 ノード
マスター障害発生後	マスター	停止	edb1	1 ノード

6.2.4. スレーブ障害（2台目）

2 台目のスレーブ障害が発生すると同期スレーブ数が 0 となり、充足数に足りなくなる。
マスターで行われた更新処理が待ち状態でハングするため、いずれかの対応が必要である。

- 任意のスレーブを起動
- マスターの同期設定を非同期に切り替え
  postgresql.conf の synchronous_standby_names=''

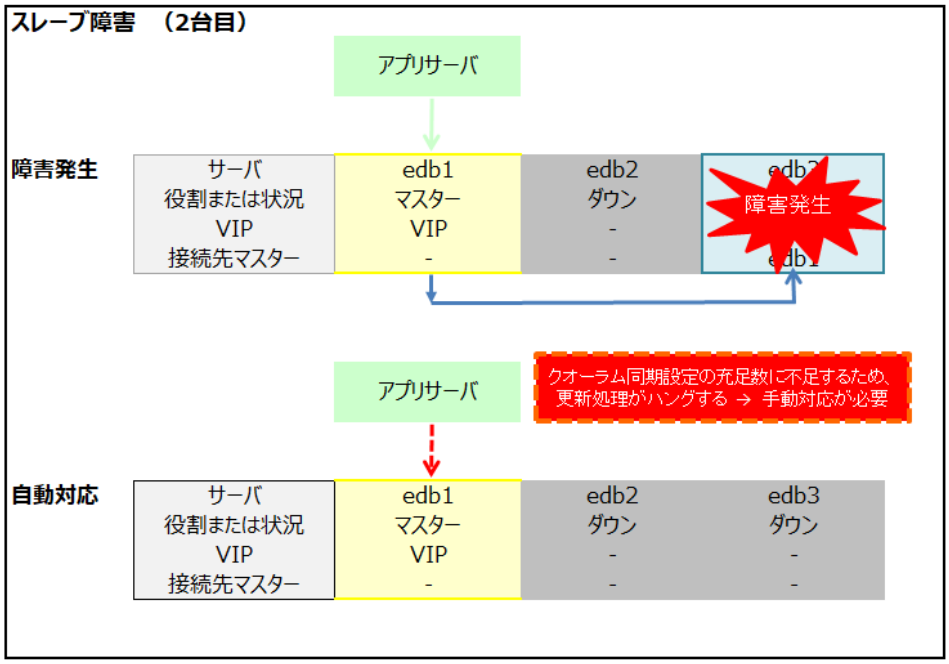


表 6-8　スレーブ障害（2台目）におけるマスター接続先情報

タイミング	edb1	edb2	edb3	同期ノード数
マスター障害発生前	マスター	ダウン	edb1	1　ノード
マスター障害発生前	マスター	ダウン	ダウン	0　ノード

6.2.5. エージェント（マスター）障害

エージェント（マスター）が停止すると、スレーブのエージェントがそれを検知する。
エージェントのみの停止かどうかをチェックする。

マスターデータベースに接続可能か
VIP にアクセス可能か

マスターデータベースに異常がない場合は、エージェントのみの障害とみなし、データベースのフェイルオーバは行わない。

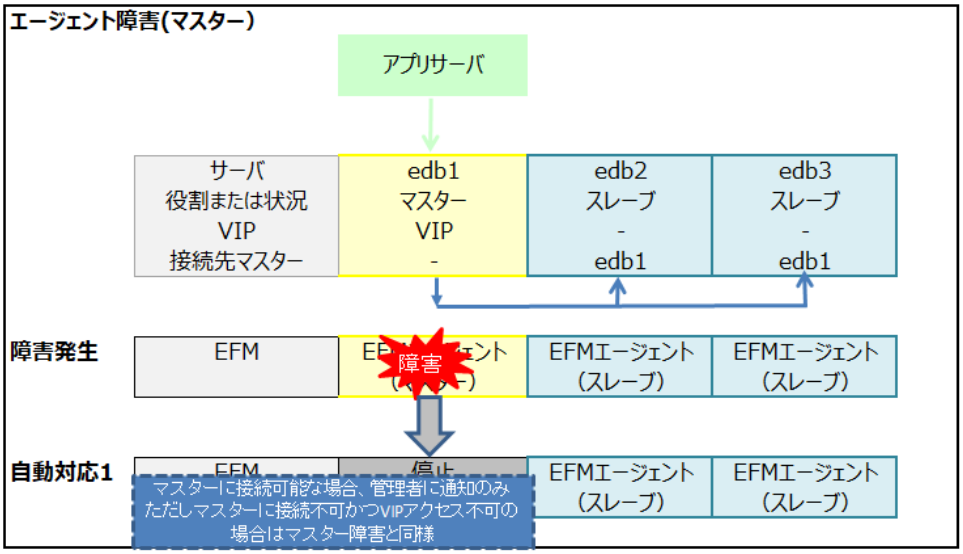


表 6-9 ローカルデータベースに対する監視プロパティ

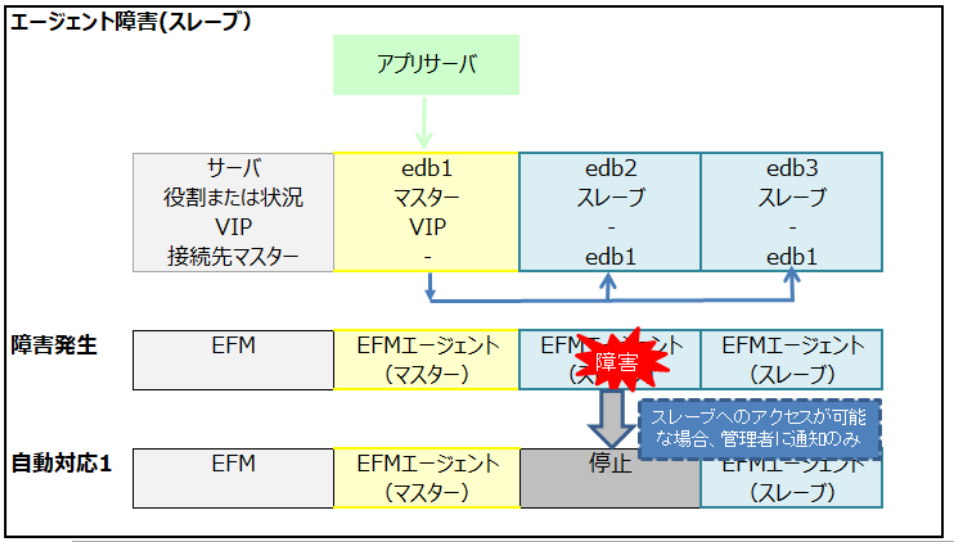
パラメータ	デフォルト値	パラメータの意味
local.period	10	ローカルデータベースの監視間隔
local.timeout	60	ローカルデータベースから応答がない場合に、障害と判断するまでの時間
local.timeout.final	10	local.timeout 後、最終確認の時間 この時間の経過後にフェイルオーバが発生する

表 6-10 リモートデータベースに対する監視プロパティ

パラメータ	デフォルト値	パラメータの意味
remote.timeout	10	リモートデータベースからの応答がない場合に障害と判断するまでの時間

6.2.6. エージェント（スレーブ）障害

エージェント（スレーブ）が停止すると、他ノードのエージェントが検知する。
エージェントのみの停止かどうかをチェックする。



7. レプリケーション設定(postgresql.conf)

役割切り替え後もそのまま使用できるようpostgresql.confはマスター/スレーブ共に同一のものを使用する。

マスター/スレーブの役割切り替え時にもそのまま利用できるよう、スレーブの postgresql.conf は pg_basebackup 等でマスターのコピーを作成する。パラメータはマスター/スレーブ両役割のものを予め設定しておく。複数スレーブ構成の場合も管理上はできるだけ共通化するのが望ましい。なお、自ノードの役割とは異なるパラメータは無視される。

ストリーミングレプリケーション関連のパラメータは以下のノード分類毎に必要なパラメータが異なる。
以降、ノード分類毎に設定すべきパラメータを説明する。

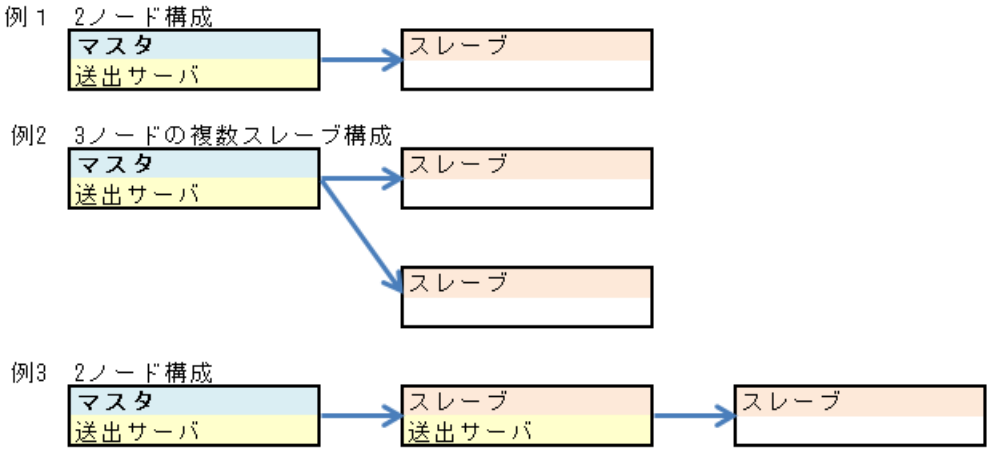


表 7-1 ストリーミングレプリケーションのノード分類

ノード分類	説明
送出ノード	マスターまたはカスケード構成時に配下にスレーブを持つノード
マスター	配下にスレーブを持つ更新可能なノード（常に送出ノード）
スレーブ	上位に送出ノードがあり、配下にスレーブを持たないノード

通常、送出ノードはマスターのみであるが、カスケード構成（親→子→孫）の「子」に当たるスレーブでは送出ノードの役目も担う。

7.1.送出ノードのパラメータ

7.1.1. max_wal_senders （送出ノード）

標準値　　：10
デフォルト：10

wal sender プロセスの最大数を設定する。wal sender プロセスは以下で使用される。
- ストリーミングレプリケーションのマスタープロセスとして起動
- pg_basebackup コマンド実行時に起動

デフォルトは 10 であり、自ノードがストリーミングレプリケーションのマスターとして構成された時、最大 10 のスレーブにを持つことができる。スレーブ毎に専用の wal sender プロセスが必要となるため、スレーブ数分、wal sender プロセスが起動する。
起動しなければサーバへの負荷は無いためストリーミングレプリケーション構成の採用可否に関わらずデフォルトの 10 とする。

7.1.2. max_replication_slots （送出ノード）

標準値　　：10
デフォルト：10

レプリケーションスロットの最大数を設定する。
スレーブ毎にレプリケーションスロットの使用/未使用を設定できる。
使用する場合は、スレーブ毎にレプリケーションスロットを作成する。
レプリケーションスロットの採用可否に関わらず max_wal_senders と合わせてデフォルトの 10 とする。

7.1.3. wal_keep_segments （送出ノード）

標準値　　：レプリケーションスロットの仕様有無で設定を変更する。
- レプリケーションスロット使用時は、0とする。
- レプリケーションスロット未使用時は、以下を試算し乗算した値とする。
  - 1日分のWAL生成数（アプリケーション依存）
  - スレーブ停止許容日数（運用依存）
  - 試算が困難な場合は暫定値として32とする。

デフォルト　：　0

WAL ファイルを維持する下限数を指定する。
スレーブ停止時に備えて常に一定の WAL を維持する機能が2種類ある。
- wal_keep_segments パラメータ
- レプリケーションスロット

本パラメータで指定した数の WAL を残すことで、スレーブの一時的な停止後もレプリケーションを再開できるようにする。ただし、適切な設定が困難なことから代替機能としてレプリケーションスロットがオプション機能として実装された。レプリケーションスロットはレプリケーションの状況を確認し、未転送の WAL を維持するため、見積もりの必要がない。
ただし、レプリケーションスロットも取り扱いの注意点があることから、本パラメータによる設定も有用である。また併用も可能である。

7.1.4. wal_sender_timeout （送出ノード）

標準値　　：　60　秒
デフォルト　：　60　秒

指定された値（ミリ秒単位）より長い時間非活動のレプリケーション接続を停止する。
スレーブのプロセス異常停止時にはエラーが返るため即時にダウンを検知するが、スレーブのネットワーク停止時はスレーブからのレスポンスがないためタイムアウトになることでダウンと見做す。

7.2.マスターのパラメータ

7.2.1. synchronous_commit （マスター）

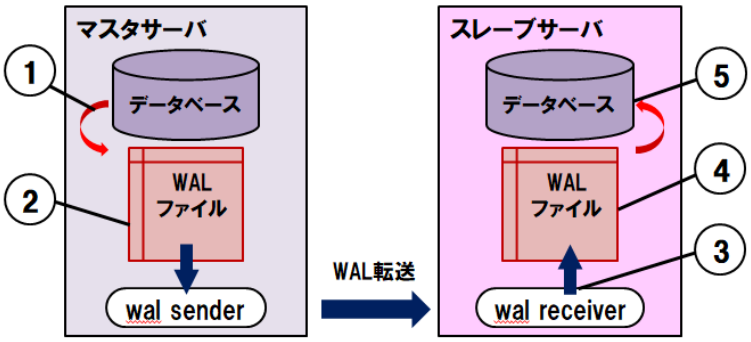
標準値　　：　性能とデータ保護バランスの良いonをベースに検討する。
デフォルト　：　on

ストリーミングレプリケーションには以下の同期レベルがある。性能、データ保護、スレーブの参照利用（参照ロードバランス）の 3 要素を基に同期レベルを検討する。

表 7-1 ストリーミングレプリケーションの同期レベル

設定値	同期レベル	説明
remote_apply	完全同期	WAL 適用まで保証。データ保護優先と参照負荷分散。
On	同期	WAL 転送（ディスク書き込み）まで保証。データ保護優先。
remote_write	準同期	WAL 転送（メモリ書き込み）まで保証。データ保護優先。
Local	非同期	ローカルの WAL 書き込みまで保証。性能優先。
Off	完全非同期	ローカルの WAL 書き込みを保証しない。

同期設定（remote_write 以上）の場合は synchronous_standby_names パラメータで同期スレーブを指定する。詳細は次項を参照。



① off　　：ローカルへの書き込みも保証しない
② local　　：ローカルへの書き込みまで保証
③ remote_write　：WAL転送後メモリ書き込みまで保証
④ on　　：WAL転送後ディスクへの書き込みまで保証
⑤ remote_apply　：WAL転送から適用まで保証

7.2.2. synchronous_standby_names （マスター）

標準値　　：同期するスレーブのapplication_nameを指定する。
デフォルト：''

同期レプリケーションをサポート可能なスタンバイサーバのリストを指定する。

活動中の同期スタンバイサーバが 1 つまたはそれ以上存在することが前提であり、設定する値は、スレーブ側の設定ファイル(recovery.conf)に設定されている primary_info の application_name である。

設定例)

```
/*  recovery.conf の設定値  */
primary_conninfo = 'host=edb-server1 port=5444 user=repuser application_name=edb-server2'

/*  postgresql.conf の設定値  */
synchronous_standby_names ='edb-server2'
```

複数スレーブを同期する場合、設定方法には固定方式/可変方式(クォーラムコミット)の 2 方式がある。すべてのスレーブを同期する場合、'*'を指定することも可能である(下表参照)。

■固定方式

設定例)

```
synchronous_standby_names = '"edb-server2","edb-server3"'
synchronous_standby_names = 'FIRST 2 ("edb-server2", "edb-server3","edb-server4")'
```

■可変方式(クオーラムコミット(充足数の指定))
複数スレーブ構成においては、任意のスレーブを同期することができる。
どちらかに WAL の書き込みが完了した時点で COMMIT が返ることで、性能が向上する可能性がある。

設定例) スレーブ 2 台のうち、任意の 1 台で同期が取れた時点で COMMIT が返る

```
synchronous_standby_names = 'ANY 1 ("edb-server2", "edb-server3")'
```

表 7-2 synchronous_standby_namesパラメータの設定値

設定値	概要
''	全スレーブを非同期
*	全スレーブを同期
特定スレーブの application_name	特定スレーブのみを同期

7.2.3. vacuum_defer_cleanup_age （マスター）

標準値　　：0
デフォルト：0

VACUUM および HOT 更新が不要行バージョンの回収を延期するトランザクションの数を指定。マスターでの VACUUM 処理とスレーブでの参照処理がコンフリクトする場合がある。スレーブでの参照処理のためにマスターでの VACUUM 処理を待機する場合に設定を検討する。

スレーブの参照を行わない場合は設定不要。
スレーブ参照時のコンフリクトが許容できない場合は動作検証で設定値を決定する。

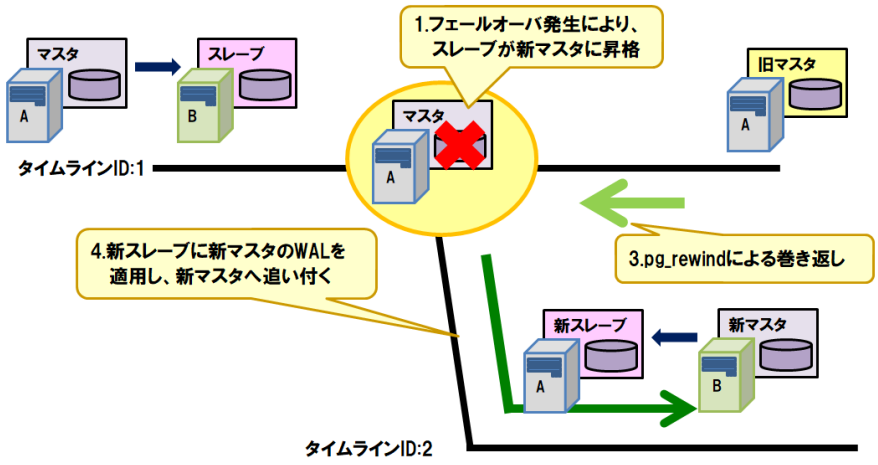
7.2.4. wal_log_hints (マスター)

標準値　　：off （pg_rewindを使用する要件がある場合は、on）
デフォルト：off

on の場合はチェックポイント後にはじめてページを変更する際にページの全内容を WAL に書き出す（full page writes の発生）。メリットと注意点のトレードオフとなる。

メリット：pg_rewind による管理性の向上
ストリーミングレプリケーションにおいては、pg_rewind が使用可能になる。フェイルオーバー後の旧マスターを新スレーブに組み入れる際に pg_rewind が使用できることで管理性が向上します。

注意点：full page writes 発生による性能への懸念
ヒントビットに対する変更にも full page writes が発生するため、チェックポイント発生後の処理に一定の負荷がかかる。



7.3. スレーブ

7.3.1. hot_standby （スレーブ）

標準値	：	on
デフォルト	：	on

ホットスタンバイ機能の有効可否を指定する。
ホットスタンバイ機能は常に有効化してスレーブにて参照可能とする。
ホットスタンバイが無効の場合、スレーブに問い合わせができないため、死活監視が制限されるためである。

7.3.2. wal_receiver_status_interval （スレーブ）

標準値	：	10秒
デフォルト	：	10秒

スレーブの wal receiver プロセスがマスターに対してレプリケーションの進捗情報を送信する最大間隔を指定する。トランザクションが発生していない時間帯でも設定された間隔で pg_stat_replication ビューが更新される。特に以下の項目は更新頻度が高い。

- 書き込みされた直近ログ位置
- ディスクにフラッシュされた直近ログ位置
- 適用された直近ログ位置

7.3.3. hot_standby_feedback （スレーブ）

標準値　　：レプリケーションスロット使用スレーブの有無により以下の設定とする。
- 使用するスレーブがある場合　　：on
- 使用しないスレーブがない場合　　：off

デフォルト　：off

スレーブの状況をマスターにフィードバックする機能の有効可否を指定する。
以下の機能を使用する場合には有効化する。

- レプリケーションスロット
- VACUUM 待機（vacuum_defer_cleanup_age パラメータ）

パラメータ設定はできるだけ共通化するのが望ましいため、レプリケーションスロットを使用するスレーブが一つでもあれば、全体で on を設定する。

7.3.4. wal_receiver_timeout （スレーブ）

標準値　　：60秒
デフォルト　：60秒

指定された値（ミリ秒単位）より長い時間非活動のレプリケーション接続を停止する。
マスターのプロセス異常停止時にはエラーが返るため即時にダウンを検知するが、マスターのネットワーク停止時はマスターからのレスポンスがないためタイムアウトになることでダウンと見做す。

7.3.5. max_standby_archive_delay （スレーブ用）

標準値　　：同期モードに応じて以下の設定とする。
- 完全同期モード以外　　：30秒
- 完全同期モード　　：0秒

デフォルト　：30秒

スレーブで適用されようとしているアーカイブ WAL エントリとスレーブの参照処理がコンフリクトする場合に参照処理がキャンセルするまでの WAL の適用処理の待機時間を設定する。
スレーブの適用が遅延する可能性があるため、同期モードを考慮する。

- 完全同期以外の同期モード　　WAL の書き込みまでの保証であるため、影響はない
- 完全同期モード　　書き込み遅延が発生するため、0 に設定する

7.3.6. max_standby_streaming_delay （スレーブ用）

標準値　　：同期モードに応じて以下の設定とする。
- 完全同期モード以外　　：30秒
- 完全同期モード　　：0秒

デフォルト　：30秒

スレーブで適用されようとしているWALエントリとスレーブの参照処理がコンフリクトする場合に参照処理がキャンセルするまでの WAL の適用処理の待機時間を設定する。
スレーブの適用が遅延する可能性があるため、同期モードを考慮する。

- 完全同期以外の同期モード　　WAL の書き込みまでの保証であるため、影響はない
- 完全同期モード　　書き込み遅延が発生するため、0 に設定する

8. スレーブ設定(recovery.conf)

スレーブの設定ファイル（recovery.conf）は以下の用途で利用される。
- PITR 設定
- ストリーミングレプリケーションのスレーブ設定

テンプレートが用意されていないため、事前に管理者が作成しておく必要がある。

8.1.PITR 設定

8.1.1. restore_command

標準値　：'cp /<アーカイブディレクトリ>/%f ″ %p″'
デフォルト：''

アーカイブ WAL ファイルを取得するために実行するコマンドを指定する。
PITR においては必須であり、必ず設定する。

（設定例）

```
'cp /archivedir/%f ″ %p″'
```

変数%p および%f は自動的に以下の値に置換される。
%f　アーカイブ WAL ファイル名
%p　コピー先のディレクトリ名

8.1.2. recovery_target_timeline (PITR)

標準値　：latest
デフォルト：ベースバックアップ取得時点と同一タイムラインID

通常は latest を指定する。
過去のタイムラインにリカバリする場合は、過去のタイムラインで取得したバックアップをリストアし、ターゲットとするライムライン ID を指定する。

リストア時点でのタイムライン ID は制御ファイルから確認できる。
（確認例）

```
$ pg_controldata | grep -i timelineid
最終チェックポイントの PrevTimeLineID:         3
```

8.1.3. recovery_target_time (PITR)

標準値　　：　指定した過去にリカバリする要件がある場合に指定する。
デフォルト　：　なし

リカバリしたい過去の時間を指定する。
障害発生直前の時点あるいはリカバリ可能な時点までリカバリする場合は、指定不要。

8.1.4. recovery_target_action (PITR)

標準値　　：　promote
デフォルト　：　pause

通常は promote を指定して障害復旧直前までリカバリし、リカバリ完了と同時に読み書き可能（運用可能）な状態とする。ただし、指定した過去にリカバリする、もしくはデータの状態を確認しながらリカバリする必要がある場合は、pause を指定する。これによりリカバリ目標時間が不明瞭な場合に各時点でのデータの状態を確認しながら少しずつリカバリを進めることが可能である。

表 8-1 ストリーミングレプリケーションのノード分類

設定値	説明
pause	リカバリを休止（参照専用モードで接続可能）
promote	リカバリの過程が終われば受付可能（読み書き可能モードで接続可能）
shutdown	リカバリ対象に到達した後にインスタンス停止

例）あるテーブルの件数が 100 件の時点にリカバリしたいが、明確な時間が不明である

recovery_target_time	recovery_target_action	データ件数	アクション
pause	'2020-03-01 12:00:00 JST'	90 件	未達、再リカバリ
pause	'2020-03-01 12:01:00 JST'	95 件	未達、再リカバリ
pause	'2020-03-01 12:02:00 JST'	100 件	到達、リカバリ完了

リカバリモードから運用モードに移行する場合は、以下のコマンドで昇格する。

```
$ pg_ctl promote
サーバの昇格を待っています....完了
サーバは昇格しました
```

8.2.ストリーミングレプリケーションのスレーブ設定

8.2.1. standby_mode (スレーブ)

標準値　　：　on
デフォルト　：　off

ストリーミングレプリケーションのスレーブとして起動するかどうかを指定する。
ストリーミングレプリケーション構成においては必須設定であるため、on を指定する。

8.2.2. primary_conninfo (スレーブ)

host、port、userを必須項目とし、下表を参考に設定する。
複数スレーブ構成の場合は、application_nameも設定する。

スレーブがマスターに接続するための接続文字列を指定する。環境変数で代替も可能だが、管理性向上のため本パラメータで明示的に指定する。

表 8-2 primary_conninfoの設定内容

項目	内容
host	マスターのホスト名または IP アドレスを指定。 複数のネットワークがある場合、WAL 転送に使用するネットワークのアドレスを指定する。
port	データベースがリスニングしているポートを指定。
user	接続するデータベースユーザを指定。REPLICATION 属性が有効である必要がある。
password	上記ユーザのパスワードを指定。
application_name	複数スレーブ構成では、区別するために指定が必要。 デフォルトは walreceiver。 以下で指定または表示される。 マスターの synchronous_standby_names パラメータ （同期の場合） マスターの pg_stat_replication ビューの application_name 列 マスターの pg_stat_activity ビューの application_name 列 どのサーバからの接続か分かる名前（サーバ名を含める）とする。

8.2.3. recovery_target_timeline (スレーブ)

標準値　　：　latest
デフォルト　：　latest

昇格時の作業やレプリケーションの再構築が簡単になるようストリーミングレプリケーションのスレーブでは latest を指定する。

8.2.4. restore_command (スレーブ)

標準値　　：　'scp /<アーカイブディレクトリ>/%f ″ %p″'
デフォルト　：　''

基本的には PITR の restore_command と同様であるが、以下の差異がある。

- アーカイブ WAL ファイルの配置先がマスターのローカルである場合はリモートからコピーする必要があるため、scp コマンドを使用する。
- ストリーミングレプリケーションにおいて必須の設定ではない。レプリケーションスロットを使用する場合は設定不要である。

ストリーミングレプリケーションにおいては基本的には primary_conninfo の接続先の wal_sender から WAL が送信されるため、restore_command によるアーカイブ WAL の指定は保険的な位置付けである。特にレプリケーションスロットを使用する場合はレプリケーションに必要な WAL ファイルが確保されるため、アーカイブの転送設定は不要である。

8.2.5. primary_slot_name (スレーブ)

標準値　　：各スレーブ用のレプリケーションスロット名を指定する
デフォルト：なし

レプリケーションスロットの使用は必須ではないが、本書ではレプリケーションスロットの使用を標準としているため、本パラメータも設定する。

設定値の反映には再起動が必要である。また、指定したレプリケーションスロットがマスターに存在しない場合はスレーブが起動しないため、注意が必要である。

設定例）

1. レプリケーションスロットを作成（マスター）
   スレーブが複数ある場合は、スレーブ毎に作成する。

```
=# SELECT pg_create_physical_replication_slot('slot_slave1');
=# SELECT pg_create_physical_replication_slot('slot_slave2');
```

2. recovery.conf でレプリケーションスロット名を指定し、再起動（スレーブ）

スレーブ 1 の recovery.conf）

```
standby_mode = on
primary_conninfo = 'host=master port=5444 user=repuser'
recovery_target_timeline = latest
primary_slot_name = 'slot_slave1'
```

スレーブ 2 の recovery.conf）

```
standby_mode = on
primary_conninfo = 'host=master port=5444 user=repuser'
recovery_target_timeline = latest
primary_slot_name = 'slot_slave2'
```

8.2.6. archive_cleanup_command (スレーブ)

標準値　　：　'pg_archivecleanup /<アーカイブWALファイルの出力先> %r'
デフォルト　：　''

アーカイブモードではアーカイブ WAL ファイルの削除を定期的に行う必要がある。
pg_archivecleanup コマンドを使用すると便利である。ただし、複数スレーブ構成ではどのスレーブでも必要ないことが保証されないため、単一スレーブ構成での利用のみとする。また、バックアップを構築している場合、アーカイブ WAL ファイルは PITR の要件で必要となるため、本コマンドでのアーカイブ WAL ファイルの削除は要検討である

設定例）

```
archive_cleanup_command   =   'pg_archivecleanup /archive_dir %r'
```

8.2.7. trigger_file (スレーブ)

標準値　　：　/tmp/<クラスタ名>.trigger
デフォルト　：　なし

リカバリ完了を示すトリガファイルを指定する。この値を指定すると、スレーブはこのファイルの存在をチェックする。リカバリ完了時点でこのファイルを作成することで、スレーブを昇格させることができる。

本パラメータの設定有無に関わらず pg_ctl promote による昇格は有効である。通常は pg_ctl promote コマンドで昇格させるため、本パラメータは指定しない。但し、EFM 使用時には本パラメータの設定が必須であるため、EFM 使用時のみ設定する。

参考文献

- PostgreSQL 日本語マニュアル
  https://www.postgresql.jp/document/11/html/
- PGECons 2017 年度 WG3 活動報告書 レプリケーション調査編
  https://pgecons-sec-tech.github.io/tech-report/html_wg3_replication/wg3_replica.html
- PGECons 2016 年度 WG3 活動報告書 レプリケーション調査編
  https://pgecons-sec-tech.github.io/tech-report/html_wg3_2016_replica/wg3_2016_replica.html