PG-REX 14利用マニュアル

第 1.0.0版

2022年2月1日

NTT OSSセンタ

改版履歴

| 日付 | 版数 | 変更内容 |
| --- | --- | --- |
| 2022/02/01 | 1.0.0 | 初版制定 |

本マニュアルのいかなる記載も、マニュアルに登場するPacemaker等のオープンソースソフトウェアにおいて、第三者の特許権等の知的財産権を侵害しないことを保障するものではありません。

本マニュアルに記載されたオープンソースソフトウェアを利用するか否かの判断は、ユーザの判断と責任で行って下さい。

Pacemakerは、Pacemakerプロジェクト（<http://www.clusterlabs.org/>）によるオープンソースソフトウェアです。

Corosyncは、Corosyncプロジェクト（<http://www.corosync.org/>）によるオープンソースソフトウェアです。

Linuxは、Linus Torvaldsの登録商標です。Red Hat及びEnterprise Linuxは、米国およびその他の国におけるRed Hat,Inc.の商標または登録商標です。

その他記載された社名・銘柄・製品名等は該当する各社の商標または登録商標です。

Table of Contents

[1章. はじめに 8](#_Toc95807999)

[1.1. 本書の目的 8](#_Toc95808000)

[1.2. 対象読者 8](#_Toc95808001)

[1.3. 表記規則 9](#_Toc95808002)

[1.4. 対象とするソフトウェアとバージョン 10](#_Toc95808003)

[1.5. 前提条件 10](#_Toc95808004)

[2章. 概要 11](#_Toc95808005)

[2.1. PG-REXの概要 11](#_Toc95808006)

[2.2. 制約 12](#_Toc95808007)

[2.2.1. PG-REXの制約 12](#_Toc95808008)

[2.2.2. PG-REX運用補助ツールの制約 14](#_Toc95808009)

[2.3. 推奨設定 15](#_Toc95808010)

[3章. 環境構築 16](#_Toc95808011)

[3.1. ネットワーク 16](#_Toc95808012)

[3.2. ディレクトリ構成 18](#_Toc95808013)

[3.3. Pacemaker 19](#_Toc95808014)

[3.3.1. 事前作業 19](#_Toc95808015)

[3.3.2. インストール 20](#_Toc95808016)

[3.3.3. インストール完了後作業 22](#_Toc95808017)

[3.3.4. HAクラスタ構築 23](#_Toc95808018)

[3.3.5. ACPI Soft-Off機能の無効化 23](#_Toc95808019)

[3.4. PostgreSQL 25](#_Toc95808020)

[3.4.1. PostgreSQLのインストール 25](#_Toc95808021)

[3.4.2. 環境変数の設定 26](#_Toc95808022)

[3.4.3. DBクラスタ用ディレクトリの作成 27](#_Toc95808023)

[3.4.4. DBクラスタの初期化 27](#_Toc95808024)

[3.4.5. postgresql.confの編集 28](#_Toc95808025)

[3.4.6. レプリケーションユーザの作成 31](#_Toc95808026)

[3.4.7. pg\_hba.confの編集 32](#_Toc95808027)

[3.4.8. パスワードファイルの作成 32](#_Toc95808028)

[3.5. PG-REX運用補助ツール 33](#_Toc95808029)

[3.5.1. インストール 33](#_Toc95808030)

[3.5.2. pg-rex\_tools.confの編集 34](#_Toc95808031)

[3.5.3. ネットワーク接続登録 35](#_Toc95808032)

[3.6. リソースの設定 36](#_Toc95808033)

[3.6.1. pm\_pcsgenの概要 36](#_Toc95808034)

[3.6.2. 変更不要の設定 36](#_Toc95808035)

[3.6.3. リソース(Primary側仮想IP)の設定 37](#_Toc95808036)

[3.6.4. PG-REXにおけるPostgreSQL制御の設定 37](#_Toc95808037)

[3.6.5. リソース（ネットワーク監視）の設定 38](#_Toc95808038)

[3.6.6. リソース（STONITH）の設定 38](#_Toc95808039)

[3.6.7. リソース（Standby側仮想IP）の設定 38](#_Toc95808040)

[3.6.8. xmlファイルの作成 39](#_Toc95808041)

[3.7. アンインストール 40](#_Toc95808042)

[4章. 起動と停止 43](#_Toc95808043)

[4.1. PG-REXの起動 43](#_Toc95808044)

[4.2. Primaryの起動 43](#_Toc95808045)

[4.3. Standbyの起動 47](#_Toc95808046)

[4.4. PG-REXの停止 49](#_Toc95808047)

[4.5. Standbyの停止 49](#_Toc95808048)

[4.6. Primaryの停止 50](#_Toc95808049)

[4.7. PostgreSQL停止中のノードの停止 50](#_Toc95808050)

[5章. メンテナンス時の対応 51](#_Toc95808051)

[5.1. PostgreSQLのバックアップ 51](#_Toc95808052)

[5.1.1. PostgreSQLのバックアップ 51](#_Toc95808053)

[5.1.2. その他のバックアップツールを使用する場合の注意事項 52](#_Toc95808054)

[5.2. アーカイブログの削除 53](#_Toc95808055)

[5.2.1. PostgreSQLアーカイブログの削除 53](#_Toc95808056)

[5.3. 計画的なノード切り替え 55](#_Toc95808057)

[5.3.1. ノード切り替え 55](#_Toc95808058)

[5.4. 計画的なメンテナンスのためのノードの停止、起動 57](#_Toc95808059)

[5.4.1. メンテナンスのためのノードの停止 57](#_Toc95808060)

[5.4.2. メンテナンス後のノードの起動 58](#_Toc95808061)

[6章. 故障対応 59](#_Toc95808062)

[6.1. 前提条件 59](#_Toc95808063)

[6.2. 監視コマンド表示確認方法 60](#_Toc95808064)

[6.2.1. 表示部説明 63](#_Toc95808065)

[6.2.2. 正常状態確認方法 69](#_Toc95808066)

[6.3. 故障箇所特定手順 70](#_Toc95808067)

[6.3.1. ノード情報・リソース情報・故障回数表示部の確認 70](#_Toc95808068)

[6.3.2. 属性情報表示部の確認 72](#_Toc95808069)

[6.3.3. corosync-cfgtool -sによるIC-LAN状態の確認 73](#_Toc95808070)

[6.3.4. 制御エラー情報表示部の確認 74](#_Toc95808071)

[6.3.5. /var/log/messagesの確認 76](#_Toc95808072)

[6.4. ルータ故障 77](#_Toc95808073)

[6.4.1. 故障時のHAクラスタ状態 77](#_Toc95808074)

[6.4.2. 復旧 77](#_Toc95808075)

[6.5. S-LAN故障 80](#_Toc95808076)

[6.5.1. 故障時のHAクラスタ状態 80](#_Toc95808077)

[6.5.2. 復旧 80](#_Toc95808078)

[6.6. D-LAN故障 83](#_Toc95808079)

[6.6.1. 故障時のHAクラスタ状態 83](#_Toc95808080)

[6.6.2. 復旧 83](#_Toc95808081)

[6.7. リソース故障(monitor) 86](#_Toc95808082)

[6.7.1. 故障時のHAクラスタ状態 86](#_Toc95808083)

[6.7.2. 復旧 86](#_Toc95808084)

[6.8. リソース故障(demoteおよびstop) 89](#_Toc95808085)

[6.8.1. 故障時のHAクラスタ状態 89](#_Toc95808086)

[6.8.2. 復旧 89](#_Toc95808087)

[6.9. リソース故障(ipaddr-primary) 93](#_Toc95808088)

[6.9.1. 故障時のHAクラスタ状態 93](#_Toc95808089)

[6.9.2. 復旧 93](#_Toc95808090)

[6.10. リソース故障(ipaddr-replication) 96](#_Toc95808091)

[6.10.1. 故障時のHAクラスタ状態 96](#_Toc95808092)

[6.10.2. 復旧 97](#_Toc95808093)

[6.11. リソース故障(ipaddr-standby) 99](#_Toc95808094)

[6.11.1. 故障時のHAクラスタ状態 99](#_Toc95808095)

[6.11.2. 復旧 100](#_Toc95808096)

[6.12. ノード故障 102](#_Toc95808097)

[6.12.1. 故障時のHAクラスタ状態 102](#_Toc95808098)

[6.12.2. 復旧 102](#_Toc95808099)

[6.13. IC-LAN故障 106](#_Toc95808100)

[6.13.1. 故障時のHAクラスタ状態 106](#_Toc95808101)

[6.13.2. 復旧 106](#_Toc95808102)

[7章. コマンド直接実行の運用 109](#_Toc95808103)

[7.1. 起動と停止 109](#_Toc95808104)

[7.1.1. PG-REXの起動 109](#_Toc95808105)

[7.1.2. Primaryの起動 109](#_Toc95808106)

[7.1.3. Standbyの起動 115](#_Toc95808107)

[7.1.4. PG-REXの停止 118](#_Toc95808108)

[7.1.5. Standbyの停止 118](#_Toc95808109)

[7.1.6. Primaryの停止 119](#_Toc95808110)

[7.2. アーカイブログの削除 120](#_Toc95808111)

[7.2.1. PostgreSQLアーカイブログの削除 120](#_Toc95808112)

[7.3. 計画的なノード切り替え 123](#_Toc95808113)

[7.3.1. ノード切り替え 123](#_Toc95808114)

[8章. 用語集 126](#_Toc95808115)

# はじめに

## 本書の目的

本書は、PostgreSQL-REX14(以降、PG-REXと呼ぶ)の技術情報を説明します。本書の目的は、読者が、PG-REXの動作環境を構築し、PG-REXを操作できるようになることです。

## 対象読者

本書は、PG-REXの環境構築やシステム運用を行うエンジニアを対象読者としています。また、本書では本文中で以下のドキュメントやインターネット上の関連サイトを参照します。

Pacemaker関連サイト

* <http://clusterlabs.org/doc/en-US/Pacemaker/2.0/html/Pacemaker_Explained/index.html> (本家マニュアル)
* <http://clusterlabs.org> (Linux-HAのメインサイト(英語))
* <http://linux-ha.osdn.jp> (Linux-HA Japanプロジェクトのサイト)
* <http://ja.osdn.net/projects/linux-ha> (Linux-HA Japanの開発者向のサイト)
* ⇒本書では、上記をまとめて『Pacemaker関連サイト』と呼びます。

PostgreSQL 14付属ドキュメント

* <http://www.postgresql.jp/document/14>
* ⇒本書では、『PostgreSQLドキュメント』と呼びます

## 表記規則

本書では、以下の表記規則を使用します。なお、用語説明は、『用語集』に掲載されています。

* 『』 : 本書内で参照する書名（本書の章見出し等も含む）
  + 『Pacemaker関連サイト』
* *斜体* : 環境に応じて変更する項目
  + eno1
* # : rootユーザの操作記号
  + # pcs status
* $ : 一般ユーザの操作記号
  + $ psql
* 赤太字 : 端末操作においてユーザが意識すべき箇所
  + PGDATA=/dbfp/pgdata/data

## 対象とするソフトウェアとバージョン

対象とするソフトウェアとバージョンは以下の通りです。Errataの適用が必要となる場合、表の各ソフトウェアのバージョンの後ろに”+”をつけて示しました。

また、本書の実行例はRed Hat Enterprise Linux 8.4の環境でのものです。

対象とするOSとバージョン

| OS | バージョン |
| --- | --- |
| Red Hat Enterprise Linux[[1]](#footnote-1) | 8.4 |
| Red Hat Enterprise Linux High Availability Add-On[[2]](#footnote-2) | 8.4 + RHBA-2021:1996 |

対象とするミドルウェアとバージョン

| ミドルウェア | バージョン |
| --- | --- |
| PostgreSQL | 14[[3]](#footnote-3) |
| Pacemaker | 2.0.5-9[[4]](#footnote-4) |
| pm\_extra\_tools | 1.3 |
| 運用補助ツール | 14 |

## 前提条件

本書に記載されているコマンド実行例は、LANG=ja\_JP.UTF-8を前提にしています。LANGの設定によっては、出力メッセージなど実行例の内容が本書と異なる可能性があるため注意してください。例えば、PostgreSQL関連のツールは、LANG=ja\_JP.UTF-8では日本語でメッセージ等を出力します[[5]](#footnote-5)が、LANG=Cの場合は英語で出力します。

また、コンソールの表示はOSのバージョンによって異なる場合があります。

本書では、PG-REXを構成するサーバをノードと呼びます。また、PG-REXを構成する2つのノードの名前（uname -nの結果）をpgrex01、pgrex02とします。

# 概要

## PG-REXの概要

PG-REXは、PostgreSQL14の同期レプリケーションにPacemakerを組み合わせた高可用ソリューションです。PG-REXは以下の特徴を持ち、システムに対して高い信頼性と可用性を提供します。

* 同期レプリケーション
* PG-REXは、PostgreSQL14の同期レプリケーションを使用します。トランザクションは、PrimaryからStandbyに転送が完了するまで成功しません(クライアントに結果を返しません)。PG-REXは、トランザクションのコミットが確定した時点でPrimaryとStandbyの両ノードでトランザクションが完了したことを保証します。つまり、一方が故障してフェイルオーバ等が発生したとしても、コミット済のトランザクションが失われることはありません。PG-REXをシステムに適用することで、単一故障に対してデータが失われないことを保証することができます。
* 自動フェイルオーバ
* PG-REXでは、Pacemakerに同期レプリケーションのPrimaryとStandbyを管理させ、自動フェイルオーバを提供します。Primaryの故障を検知すると、Pacemakerは自動的にStandbyをPrimaryに昇格させ、クライアントがデータベースサービスを継続的に利用できることを保証します。また、Standbyの故障を検知した場合でも、Pacemakerは自動的にPrimaryを単独で稼働させ続け、故障によりデータベースサービスが停止するのを防ぎます。PG-REXをシステムに適用することで、単一故障に対してデータベースサービスのダウンタイムを極小化することができ、高い可用性を保証することができます。

## 制約

本節では、PG-REXおよびPG-REX運用補助ツール利用時の制約を示します。

### PG-REXの制約

PG-REXは、PacemakerとPostgreSQLの制約を引き継ぎます。それらの制約は、『Pacemaker関連サイト』、『PostgreSQLドキュメント』を参照してください。

PG-REXそのものの制約を以下に示します。ただし、パラメータに関する制約は『環境構築』の章の『postgresql.confの編集』を参照してください。なお、PG-REXでは二重故障時の継続動作を保証しません。

1. 動作環境の制約
   1. 両ノード共にSTONITHが設定されていること。
   2. PG-REXの運用には最低限、以下に示す3つの独立したNWセグメントが必要となる。
      * データベースサービス提供用LAN(以下S-LAN)
      * インターコネクト通信用LAN(以下IC-LAN)
      * データ転送用(以下D-LAN)とIC-LAN用の兼用LAN
   * そのため、PG-REXを構築するノードには少なくとも3つのLANポートが必要となる。また、上記の構成でSTONITH用のLAN(以下STONITH-LAN)と運用管理用のLANを他のLANと兼用する場合は、各自の判断で兼用するLANを選択すること。
   1. PG-REXで制御可能なノード数は2である。

1. 運用中の制約
   1. D-LANの切断やStandbyの停止・クラッシュによりレプリケーションが停止すると、一時的にトランザクションが停止するため、トランザクションの実行時間が長くなる。
   2. コミット発行時に、アプリケーション側に異常終了が返却された場合、トランザクションは以下のいずれかの状態となる。異常終了がPrimaryのクラッシュによるものである場合、フェイルオーバ後にそのトランザクションが見えるかどうかは不定である。
      * PrimaryとStandbyともにトランザクションは確定済
      * Primaryのみトランザクションは確定済。Standbyでは未確定
      * PrimaryとStandbyともにトランザクションは未確定
   3. pg\_cancel\_backend関数[[6]](#footnote-6) またはpg\_terminate\_backend関数[[7]](#footnote-7) をPrimaryで実行すると以下のログが出力されることがある。
   * 「DETAIL: The transaction has already committed locally, but might not have been replicated to the standby.」
   * この場合、それらの関数によってキャンセルされたSQLやトランザクションはPrimaryにおいて既にコミットされている状態となる。ただし、StandbyにWALの同期書込みがされている保証はない。
   1. 本マニュアル以外のPacemaker関連のドキュメントに記載されている各操作手順は、PG-REXでは動作保証しない。
   2. 以下の操作を行う場合は、事前に必要な準備を行った上で実行すること[[8]](#footnote-8)。
      * テーブルスペースの作成:両ノードにテーブルスペースのディレクトリを作成した上で、テーブルスペースを作成する必要がある。
      * 独自に開発したライブラリの登録:両ノードにライブラリを配置した上で、登録する必要がある。
   3. PostgreSQLのバックアップをStandbyで取得する際に、バックアップ中にフェイルオーバが発生した場合、バックアップが正常に取得できない。
   4. レプリケーション設定はPG-REXが自動的に設定するため、手動で設定を行った場合の動作は保証しない。 PG-REXが自動的に設定するレプリケーション設定の詳細は『postgresql.confの編集』を参照のこと。

1. フェイルオーバ動作時の制約
   1. フェイルオーバが発生した場合、一時的なデータベースサービスの停止が発生する。そのため、アプリケーション側で接続断に対する後処理を実施する必要がある。
   2. Standbyの起動後、StandbyがPrimaryに追いつく(同期が完了する)までの間にPrimaryが終了した場合、フェイルオーバは行われない。
   3. Standbyでクエリを実行する場合、クエリの読み込み処理とレプリケーションの書き込み処理の競合[[9]](#footnote-9)によりフェイルオーバ時間が長くなる可能性がある。これは、競合によりレプリケーションの書込み処理がその間停止するためである。

1. 故障対応時の制約
   1. 自動的なフェイルバックをサポートしない。つまり、フェイルオーバ後にpgrex02が単独でPrimaryとして稼働しているときに、旧Primaryのpgrex01をStandbyとして組み込んだ場合でも、自動的にpgrex01がPrimary、pgrex02がStandbyに切り替わることはない。

1. メンテナンス時の制約
   1. フェイルバックまたはスイッチオーバを実行したとき、フェイルオーバ時と同様に一時的なデータベースサービスの停止が発生する。

### PG-REX運用補助ツールの制約

PG-REX運用補助ツール利用時の制約を以下に示します。

1. pg-rex\_switchoverによるノード切り替えでは、Primaryの停止後からPrimaryの切り替え（新Primaryの起動）が完了するまでの間は一時的にデータベースサービスが停止した状態となる。
2. pg-rex\_switchoverによるノード切り替えの実施中に、pg-rex\_switchoverが異常終了した場合のクラスタ状態は不確定であり、データベースサービスが停止している可能性がある。この場合、元の状態への復旧は自動で実施されないため、クラスタ状態を確認し、手動復旧を試みること。
3. 起動確認はPostgreSQLやIPaddr2、Ping、STONITHなどの固有のリソースにしか確認を行わないため、Apacheなど新しくリソースを追加したとしてもその確認を行わない。
4. 両ノードの状態確認にネットワークの通信を用いるので、ツールが使用するLAN（D-LAN）切断時は、それ以外のLANが繋がっていても実行に失敗する。
5. PG-REXでインストールしたファイルのディレクトリ構成が2つのノードで同一であること。
6. アーカイブログを圧縮する場合、圧縮方式に対応した拡張子を付与しなければならない。 サポートする圧縮方式はgzip(拡張子.gz)のみである。

## 推奨設定

本節では、PG-REX運用時の推奨設定を示します。

1. データベースサービスが高負荷のときの影響を局所化するために、DBデータを配置するディスクとOSのインストールディスクは分けることを推奨する。
2. RAIDを用いる際に、高負荷時の誤ったフェイルオーバ防止や、Primary故障時の高速なフェイルオーバを実現したい場合は、大きめのRAIDキャッシュ（2GB以上）を搭載することを推奨する。

* また、スループット性能よりもフェイルオーバ時間短縮を重視したい場合は、kernelパラメータのvm.dirty\_background\_bytesを、postgresql.confのshared\_buffers以下のサイズ、もしくは搭載されているRAIDキャッシュ以下のサイズに設定することを推奨する。
* 以下はvm.dirty\_background\_bytesを1024MBに設定する際の例である。

|  |
| --- |
| # vim /etc/sysctl.conf vm.dirty\_background\_bytes=1073741824 # sysctl -p |

# 環境構築

本章では、ネットワークの構成、ディレクトリ構成、および、PG-REXのインストール手順を説明します。

## ネットワーク

PG-REXが推奨するネットワーク構成例を示します。この構成例では、各ノードにNWインターフェイスが7つある構成となっています。実際のノードのNWインターフェイスの数に応じてbonding等の設定を見直して環境を作成してください。本書では以下のIPアドレスを使用します。

※ ネットマスクはすべてのIPアドレスについて24 (255.255.255.0) とします。

* S-LAN(運用LAN)
  + pgrex01-bond0(eno0, eno5) : 192.168.0.11
  + pgrex02-bond0(eno0, eno5) : 192.168.0.12
  + ネットワーク監視先(ping) : 192.168.0.254
* D-LAN(DBレプリケーションLAN)
  + pgrex01-bond1(eno2, eno4) : 192.168.2.1
  + pgrex02-bond1(eno2, eno4) : 192.168.2.2
* IC-LAN(インターコネクトLAN)
  + pgrex01-eno1 : 192.168.1.1
  + pgrex01-eno3 : 192.168.3.1
  + pgrex02-eno1 : 192.168.1.2
  + pgrex02-eno3 : 192.168.3.2
* STONITH-LAN
  + pgrex01-eno6(管理LAN) : 172.20.144.41
  + pgrex02-eno6(管理LAN) : 172.20.144.42
  + pgrex01-HW制御ボード : 172.20.144.43
  + pgrex02-HW制御ボード : 172.20.144.44



* + - * 1. ネットワーク構成例

※ この図では、D-LANにおいてbonding設定を行ったNWインターフェイス同士を直接接続していますが、実際にはスイッチを介して接続することを推奨します。スイッチを使用せずに直接接続した場合、環境によってはOSによる故障検知に失敗する可能性があります。

PG-REXでは、Primary側接続用、Standby側接続用、およびレプリケーション受付用の仮想IPアドレスを使用します。本書では、各仮想IPアドレスを以下のように設定することを前提とします。

* S-LAN(運用LAN)
  + Primary側接続用 : 192.168.0.10
  + Standby側接続用 : 192.168.0.20
* D-LAN(DBレプリケーションLAN)
  + レプリケーション受付用 : 192.168.2.3

## ディレクトリ構成

PG-REXのディレクトリ構成例を以下に示します。

|  |
| --- |
| /var   +-lib      +-pgsql ......... postgresユーザのホームディレクトリ  /dbfp   +-pgdata ........... PostgreSQLのDBクラスタ格納先   +-pgwal   |  +-pg\_wal ........ PostgreSQLのWAL格納先   +-pgarch      +-arc1 .......... PostgreSQLのアーカイブログ格納先 |

この構成例では、PostgreSQLのDBクラスタ、WAL、アーカイブログの各格納先を別ディレクトリに配置しています。実際のディスク構成に応じて配置先を見直して環境を作成してください。ただし、両ノード（pgrex01とpgrex02）のディレクトリ構成は同一にしてください。

本書では、この構成例を前提に記述します。

## Pacemaker

本節では、Pacemakerのインストールおよび基本設定について説明します。

本節の作業は、rootユーザで行います。

### 事前作業

Pacemakerをインストールする前に、OSの確認、RHELのインストールDVD、RHEL HA Add-Onの ISOイメージ、及びpm\_extra\_toolsの準備をします。

【OSの確認】

インストールしたOSのバージョンについて、/etc/redhat-releaseの内容を確認します。以下の結果表示の斜体で記載しているバージョンを確認してください。(RHEL 8.4の場合の例)

|  |
| --- |
| # cat /etc/redhat-release Red Hat Enterprise Linux Server release 8.4 (Ootpa) |

【pm\_extra\_toolsの取得】

ダウンロードサイト[[10]](#footnote-10)より、RPMパッケージを入手します。

|  |
| --- |
| pm\_extra\_tools-1.3-1.el8.noarch.rpm  ※ ダウンロードサイトにて、使用するRHELのバージョンに適合するパッケージを選んでください。 |

### インストール

『Pacemaker関連サイト』を参考にpgrex01とpgrex02へPacemakerをインストールします。

1. mountコマンドでRHELのインストールDVDを/mediaにマウントします[[11]](#footnote-11)。

|  |
| --- |
| # mount -o ro /dev/sr0 /media |

1. /mnt/HighAvailabilityを作成し、mountコマンドでRHEL HA Add-OnのISOイメージをマウント します。

|  |
| --- |
| # mkdir /mnt/HighAvailability # mount -o ro /var/tmp/rhel-8.4-x86\_64-dvd.iso /mnt/HighAvailability  ※ バージョンは適宜読み替えてください。 |

1. /media配下、及び/mnt/HighAvailability配下をyumコマンドで参照されるリポジトリに追加する 設定を行います。

/etc/yum.repos.d配下にrheldvd.repoを新規に作成し、以下の内容を記述します。

|  |
| --- |
| [BaseOS] name=Red Hat Enterprise Linux $releasever - $basearch - BaseOS baseurl=file:///media/BaseOS enabled=1 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release  [AppStream] name=Red Hat Enterprise Linux $releasever - $basearch - AppStream baseurl=file:///media/AppStream enabled=1 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

/etc/yum.repos.d配下にrhel-ha.repoを新規に作成し、以下の内容を記述します。

|  |
| --- |
| [HighAvailability] name=Red Hat Enterprise Linux $releasever - $basearch - HighAvailability baseurl=file:///mnt/HighAvailability enabled=1 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

1. yumのキャッシュをクリアします。

|  |
| --- |
| # yum clean all |

1. Pacemakerをインストールします。

|  |
| --- |
| # yum install pcs pacemaker fence-agents-all -y |

1. pm\_extra\_toolsをインストールします。

|  |
| --- |
| # yum install pm\_extra\_tools-1.3-1.el8.noarch.rpm -y  ※ バージョンは適宜読み替えてください。 |

### インストール完了後作業

1. 追加したyumコマンドの/media配下、及び/mnt/HighAvailability配下への参照を無効化します。

/etc/yum.repos.d/rheldvd.repoを編集します。

|  |
| --- |
| [BaseOS] name=Red Hat Enterprise Linux $releasever - $basearch - BaseOS baseurl=file:///media/BaseOS enabled=0 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release  [AppStream] name=Red Hat Enterprise Linux $releasever - $basearch - AppStream baseurl=file:///media/AppStream enabled=0 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

/etc/yum.repos.d/rhel-ha.repoを編集します。

|  |
| --- |
| [HighAvailability] name=Red Hat Enterprise Linux $releasever - $basearch - HighAvailability baseurl=file:///mnt/HighAvailability enabled=0 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

1. umountコマンドでRHELのインストールDVD、RHEL HA Add-OnのISOイメージをアンマウン トします。

|  |
| --- |
| # umount /media # umount /mnt/HighAvailability |

### HAクラスタ構築

本章では、HAクラスタ構築の方法について記述しています。 HAクラスタ構築の際には、Pacemakerのインストールが完了している必要があります。

1. pcsdサービスを起動し、自動起動を有効にします。(2つのノードで実行)

|  |
| --- |
| # systemctl start pcsd.service # systemctl enable pcsd.service # systemctl is-enabled pcsd.service |

1. haclusterユーザのパスワードを設定します。(2つのノードで実行)

|  |
| --- |
| # passwd hacluster ユーザー hacluster のパスワードを変更。 新しいパスワード: 新しいパスワードを再入力してください: passwd: すべての認証トークンが正しく更新できました。 |

1. 各ノードのホスト名、管理用LANのIPアドレスを指定し、ノードの認証を行います。(いずれか１つのノードで実行)  
   認証の際はユーザ名にhacluster、パスワードに前項で設定したhaclusterのパスワードを入力してください。

|  |
| --- |
| # pcs host auth pgrex01 addr=172.20.144.41 pgrex02 addr=172.20.144.42 Username: hacluster Password: pgrex01: Authorized pgrex02: Authorized |

1. Pacemaker設定ファイルを変更します。(2つのノードで実行)

/etc/sysconfig/pacemakerのPCMK\_fail\_fastのコメントアウトを外し、以下の通り編集します。

|  |
| --- |
| ： (省略) PCMK\_fail\_fast=yes ： (省略) |

### ACPI Soft-Off機能の無効化

IPMI STONITH機能ではIPMIにより電源操作を実施しますが、その際にノードを即時に停止させる ためにACPI Soft-Off機能を無効にする必要があります。 Red Hat社のドキュメント[[12]](#footnote-12)を参照し、ACPI Soft-Off機能を無効にしてください。  
BIOSの設定でACPI Soft-Off機能を無効にできるか確認し、BIOSでACPI Soft-Off機能を無効にでき ない場合は、以下手順に従いlogind.confを編集し無効にします[[13]](#footnote-13)。

1. /etc/systemd/logind.confを以下の通り編集します。

|  |
| --- |
| HandlePowerKey=ignore |

1. systemd-logindサービスを再起動し、設定を反映させます。

|  |
| --- |
| # systemctl restart systemd-logind.service |

1. ACPI Soft-Off機能が無効化されていることを確認します。

|  |
| --- |
| # loginctl show-user ：(省略) HandlePowerKey=ignore ：(省略) |

## PostgreSQL

本節では、PostgreSQLのインストールおよび基本設定について説明します。

### PostgreSQLのインストール

『PostgreSQLドキュメント』を参考にpgrex01とpgrex02へPostgreSQLをインストールします。PG-REXで使用できるPostgreSQLのバージョンは14のみとなります。

本作業はrootユーザで行います。

PG-REXのインストールに必須のRPMパッケージを以下に示します。

|  |
| --- |
| postgresql14-libs-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-server-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-contrib-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-docs-14.0-1PGDG.rhel8.x86\_64.rpm  ※ バージョンは適宜読み替えてください。 ※ PL/PerlやPL/Tclなどの各種言語インターフェイスが必要な場合は、それぞれに対応するパッケージをインストールしてください。 |

PostgreSQLをRPMパッケージからインストールします。

|  |
| --- |
| # rpm -ivh postgresql14-libs-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-server-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-contrib-14.0-1PGDG.rhel8.x86\_64.rpm postgresql14-docs-14.0-1PGDG.rhel8.x86\_64.rpm  ※ バージョンは適宜読み替えてください。 |

PostgreSQLのRPMパッケージをインストールすると「/usr/pgsql-14」にインストールされ、「postgres」というOSのユーザと、「postgres」というグループが作成されます。ただし、作成されたユーザにはパスワードの設定はされていません。また、同名のユーザまたはグループが存在する場合は、新規作成されません。

/var/lib/pgsqlのパーミッションは700に変更され、/var/lib/pgsql配下の全ファイルのオーナ、グループがpostgres、postgresに変更されます。

PG-REXではpostgresユーザのuidを26、postgresグループのgidを26であることを前提としています。以下のコマンドを実行し、postgresユーザが規定のuid、gidで作成されていることを確認します。

|  |
| --- |
| # id postgres uid=26(postgres) gid=26(postgres) groups=26(postgres) |

規定のuid、gidになっていない場合は、下記のようにuid、gidを変更してください。

|  |
| --- |
| # groupmod -g 26 postgres # usermod -u 26 postgres # usermod -g 26 postgres |

RPMのインストールにより新規でユーザが作成された場合は、パスワードを設定してください。

|  |
| --- |
| # passwd postgres ユーザー postgres のパスワードを変更。 新しいパスワード: 新しいパスワードを再入力してください: passwd: すべての認証トークンが正しく更新できました。 |

### 環境変数の設定

pgrex01とpgrex02で/var/lib/pgsql/.bash\_profileに環境変数を設定します。

* PATH : PostgreSQLコマンドへのパスの追加
* PGDATA : DBクラスタのパスの設定

pgrex01とpgrex02で同じ設定値を使用してください。

本作業はpostgresユーザで行います。

設定例を以下に示します。

|  |
| --- |
| export PATH=/usr/pgsql-14/bin:$PATH export PGDATA=/dbfp/pgdata/data |

### DBクラスタ用ディレクトリの作成

pgrex01とpgrex02で、DBクラスタを格納するディレクトリとWALを格納するディレクトリ、アーカイブログを格納するディレクトリを以下の設定で作成します。

DBクラスタやWALを外付けストレージに配置する場合は、該当のディレクトリに外付けストレージの論理ディスクをマウントしてから実施します。

なお、ディレクトリが既に存在する場合は、ディレクトリを空にします。

本作業はrootユーザで行います。

DBクラスタ用ディレクトリの設定

| ディレクトリ | パス | 所有者 | グループ | 権限 |
| --- | --- | --- | --- | --- |
| DBクラスタ用 | /dbfp/pgdata/data | postgres | postgres | 700 |
| WAL用 | /dbfp/pgwal/pg\_wal | postgres | postgres | 700 |
| アーカイブログ用 | /dbfp/pgarch/arc1 | 任意 | 任意 | postgres ユーザが読み書き可能な権限 |

### DBクラスタの初期化

pgrex01で、postgresユーザにてDBクラスタを初期化します。

本作業はpostgresユーザで行います。

|  |
| --- |
| $ initdb -D /dbfp/pgdata/data -X /dbfp/pgwal/pg\_wal --encoding=UTF-8 --no-locale --data-checksums |

### postgresql.confの編集

pgrex01でpostgresql.confを編集します。本節は、PG-REXを構成するのに必要な設定、注意すべき設定のみ説明しています。PostgreSQL一般の設定については『PostgreSQLドキュメント』を参照してください。

PG-REXを構成するのに必要な設定、注意すべき設定を以下に示します。各パラメータの詳細な説明については、『PostgreSQLドキュメント』を参照してください。

本作業はpostgresユーザで行います。

|  |
| --- |
| listen\_addresses = '\*' password\_encryption = scram-sha-256 wal\_level = replica synchronous\_commit = on archive\_mode = always archive\_command = '/bin/cp %p /dbfp/pgarch/arc1/%f' max\_wal\_senders = 10 wal\_keep\_size = 512MB wal\_sender\_timeout = 20s max\_replication\_slots = 10 hot\_standby = on max\_standby\_archive\_delay = -1 max\_standby\_streaming\_delay = -1 hot\_standby\_feedback = on wal\_receiver\_timeout = 20s restart\_after\_crash = off |

設定値を決定するにあたって、注意すべき点を以下に示します。

* password\_encryption
* pg\_hba.confのMETHODと合わせる必要がある。
* wal\_level
* logicalを設定することも可能。
* synchronous\_commit
* 本マニュアルではレプリケーションの信頼性を重視するため、onを推奨する。remote\_writeを設定した場合、故障発生時にデータを損失する可能性がある。remote\_applyを設定した場合、on設定時と同様の信頼性となるが、トランザクションの応答時間が長くなる。
* archive\_mode
* 設定はalways必須。
* archive\_command
* WALファイルを圧縮して保存したい場合のコマンド設定例を以下に示す。
* '/bin/gzip -c %p > /dbfp/pgarch/arc1/%f.gz'
* ※ gzipを使用する場合は、後述するリストアコマンド[[14]](#footnote-14)にもgzipを使用すること。
* max\_wal\_senders
* PG-REXでは、レプリケーション機能を利用するため、1以上を設定する必要がある。ただし、pg\_basebackupによる運用中のバックアップ取得など、Standby以外からの接続に備え、余裕を持たせて設定することを推奨する。
* wal\_keep\_size
* 以下の点を考慮すること。
  + 設定値が小さい場合、レプリケーション接続が一時的に切断されたときに、Standbyに転送されていないWALファイルがPrimaryから削除される可能性が高まる。WALファイルが削除されると、レプリケーションの再接続が不可能となる。
* wal\_sender\_timeout
* Standbyの故障や両ノード間の通信断をPrimaryがすぐに検知できるように、タイムアウトを有効にすることを推奨する。postgresql.confで設定するTCP keepaliveに関する設定[[15]](#footnote-15) だけでは、検知に時間がかかることがあり、異常時のダウンタイムが長くなることがある。ただし、設定値が小さすぎると誤検知によりStandbyが切り離されることがあるため、設定値は事前検証等をして注意して決めること。
* max\_replication\_slots
* pg\_basebackupによるバックアップ取得などに備え、余裕を持たせて設定することを推奨する。

* hot\_standby
* 設定はon必須。
* max\_standby\_archive\_delay
* PG-REXでは、Standbyの監視クエリがキャンセルされるのを回避するために、-1(キャンセルを無効)を設定する。
* max\_standby\_streaming\_delay
* PG-REXでは、Standbyの監視クエリがキャンセルされるのを回避するために、-1(キャンセルを無効)を設定する。
* hot\_standby\_feedback
* 設定はon必須。
* wal\_receiver\_timeout
* wal\_sender\_timeoutと同程度に揃える必要がある。
* restart\_after\_crash
* PG-REX運用中にPostgreSQLが自動的に再起動すると、Pacemakerによる状態管理の整合性が崩れるため、offを設定しなければならない。

PG-REXでは自動的に必要な設定を行うため、ユーザは以下のパラメータを設定してはいけません。

* primary\_conninfo
* recovery\_target\_timeline
* restore\_command
* synchronous\_standby\_names

### レプリケーションユーザの作成

pgrex01で、PostgreSQLにレプリケーションのためのデータベースユーザを作成します。

本作業はpostgresユーザで行います。

データベースユーザ名とパスワードは適宜変更してください。

1. PostgreSQLを一度起動します。



|  |
| --- |
| $ pg\_ctl start |

1. CREATE ROLEコマンドでレプリケーションユーザを作成します。



|  |
| --- |
| $ psql -c "CREATE ROLE repuser REPLICATION LOGIN PASSWORD 'reppasswd'" |

1. PostgreSQLを停止します。



|  |
| --- |
| $ pg\_ctl stop |

### pg\_hba.confの編集

pg\_hba.confに、PG-REXを構成するのに必要な編集を行います。

本作業はpostgresユーザで行います。

pgrex01のpg\_hba.confを編集します。pgrex02でも同じファイルを使用するため、両ノードのD-LANアドレスからレプリケーション接続を許可する設定を記述します。

以下に、このマニュアルで想定している構成を前提とした記述例を示します。各フィールドの詳細な説明については『PostgreSQLドキュメント』を参照してください。

|  |
| --- |
| # TYPE DATABASE    USER    ADDRESS        METHOD host   replication repuser 192.168.2.1/32 scram-sha-256 host   replication repuser 192.168.2.2/32 scram-sha-256 |

### パスワードファイルの作成

PG-REX構成で使用するパスワードファイルを作成します。

本作業はpostgresユーザで行います。

pgrex01とpgrex02それぞれのpostgresユーザのホームディレクトリに、600の権限でパスワードファイル.pgpassを作成します。 PG-REXのレプリケーション接続および、運用補助ツールで利用するため、レプリケーション受付用の仮想IPアドレス、および相手のノードのD-LANのIPアドレスについて記述します。

以下に、このマニュアルで想定している構成を前提とした記述例を示します。各フィールドの詳細な説明については『PostgreSQLドキュメント』を参照してください。

|  |
| --- |
| # hostname:port:database:username:password 192.168.2.3:5432:replication:repuser:reppasswd [[16]](#footnote-16) 192.168.2.2:5432:replication:repuser:reppasswd [[17]](#footnote-17) |

## PG-REX運用補助ツール

本節では、PG-REX運用補助ツールのインストールおよび基本設定について説明します。

本節の作業はrootユーザで行います。

### インストール

pgrex01とpgrex02へPG-REX運用補助ツールをインストールします。

PG-REX運用補助ツールのインストール必須のRPMパッケージを以下に示します。

|  |
| --- |
| pg-rex\_operation\_tools\_script-14.0-1.el8.noarch.rpm IO\_Tty-1.11-1.el8.x86\_64.rpm Net\_OpenSSH-0.62-1.el8.x86\_64.rpm  ※ バージョンは適宜読み替えてください。 |

PG-REX運用補助ツールをRPMパッケージからインストールします。

|  |
| --- |
| # yum install pg-rex\_operation\_tools\_script-14.0-1.el8.noarch.rpm IO\_Tty-1.11-1.el8.x86\_64.rpm Net\_OpenSSH-0.62-1.el8.x86\_64.rpm  ※ バージョンは適宜読み替えてください。 |

### pg-rex\_tools.confの編集

pgrex01とpgrex02で/etc/pg-rex\_tools.confの設定を行います。運用補助ツールを使用するために必要な設定、および注意すべき設定は以下のとおりです。

* D\_LAN\_IPAddress
* 両ノードのD-LAN IPアドレスをカンマ区切りで設定する。
* IC\_LAN\_IPAddress
* IC-LAN系統ごとに括弧で括ったIPアドレスの組を2系統分記述する。
* Archive\_dir
* アーカイブディレクトリの絶対パスを設定する。
* IPADDR\_STANDBY
* Standby側接続用の仮想IPを使用する環境の場合はenable、それ以外の場合はdisableを設定する。
* HACLUSTER\_NAME
* Pacemakerで管理するHAクラスタ名を指定します。HAクラスタ名には英数字とアンダースコア、およびハイフンのみ使用可です。ただし先頭にはハイフンは使用できません。これ以外のPacemakerで許されている文字をHAクラスタ名に使用したい場合は、運用補助ツールを使用せず、「7章. コマンド直接実行の運用」の手順で構築してください。

設定例を以下に示します。

|  |
| --- |
| D\_LAN\_IPAddress = 192.168.2.1, 192.168.2.2 IC\_LAN\_IPAddress = (192.168.1.1, 192.168.1.2) , (192.168.3.1, 192.168.3.2) Archive\_dir = /dbfp/pgarch/arc1 IPADDR\_STANDBY = disable HACLUSTER\_NAME = pgrex\_cluster |

### ネットワーク接続登録

PG-REX運用補助ツールが相手ノードの操作や確認をD-LANを使用して行うため、事前に両ノードのrootユーザの.ssh/known\_hostsに相手先のD-LANのIPアドレスに対する接続登録をする必要があります。

1. pgrex01からD-LAN経由でpgrex02へ接続します。



|  |
| --- |
| # ssh 192.168.2.2 The authenticity of host '192.168.2.2 (192.168.2.2)' can't be established. ECDSA key fingerprint is \*\*\*\*\*\*\* Are you sure you want to continue connecting (yes/no/[fingerprint])? yes Warning: Permanently added '192.168.2.2' (ECDSA) to the list of known hosts. root@192.168.2.2's password: |

1. pgrex02からD-LAN経由でpgrex01へ接続します。



|  |
| --- |
| # ssh 192.168.2.1 The authenticity of host '192.168.2.1 (192.168.2.1)' can't be established. ECDSA key fingerprint is \*\*\*\*\*\*\* Are you sure you want to continue connecting (yes/no/[fingerprint])? yes Warning: Permanently added '192.168.2.1' (ECDSA) to the list of known hosts. root@192.168.2.1's password: |

## リソースの設定

本マニュアルとセットで提供されるPG-REX用の『pm\_pcsgen環境定義書』を用いて、クラスタ内のリソース構成、動作条件、パラメータ、配置制約などを設定します。

### pm\_pcsgenの概要

pm\_pcsgenは『pm\_pcsgen環境定義書』からxmlファイルを生成するツールです。生成されたxmlファイルを運用補助ツールあるいはpcsコマンドを使用してPacemakerに読みませることで、ファイルの内容が反映されます。

PG-REXでのxmlファイル生成までの手順は以下のとおりです。

1. PG-REX用の『pm\_pcsgen環境定義書』を編集する。
2. 編集した『pm\_pcsgen環境定義書』をcsvファイル形式に変換し、pgrex01に転送する。
3. pgrex01上でpm\_pcsgenコマンドを使用し、csvファイルをxmlファイルに変換する。

以降、『pm\_pcsgen環境定義書』の設定方法を説明します。

### 変更不要の設定

『pm\_pcsgen環境定義書』の下記の設定は変更不要です。

* 表1-1 クラスタ・ノード属性
* 表2-1 クラスタ・プロパティ
* 表3-1 リソース・デフォルト
* 表3-2 オペレーション・デフォルト
* 表5-1 リソース・パラメータ
* 表6-1 STONITHの実行順序
* 表10-1 リソース同居制約
* 表11-1 リソース起動順序制約
* 表12-1 ALERT設定
* 表13-1 追加設定

### リソース(Primary側仮想IP)の設定

『pm\_pcsgen環境定義書』の表7-1にPostgreSQLのPrimary側接続用の仮想IPを設定します。

『pm\_pcsgen環境定義書』の表7-1にレプリケーション受付用の仮想IPを設定します。

### PG-REXにおけるPostgreSQL制御の設定

『pm\_pcsgen環境定義書』の表7-1にPostgreSQLの制御に必要な設定をします。PG-REXを構成するのに必要な設定、注意すべき設定は以下のとおりです。

PG-REXにおけるPostgreSQL制御の設定

| パラメータ名 | 設定値 | 備考 |
| --- | --- | --- |
| rep\_mode | sync | PG-REXでは常にsyncを指定する。 |
| node\_list | pgrex01 pgrex02 | HAクラスタを構成するノードの名前を設定する。 (ノードの名前の間はスペース区切り) |
| master\_ip | 192.168.2.3 | 『pm\_pcsgen環境定義書』の表7-1に設定 したレプリケーション受付用の仮想IPアドレ スを設定する。 |
| restore\_command | /bin/cp /dbfp/pgarch/arc1/%f %p | アーカイブディレクトリからWALファイルを リストアするためのコマンドを設定する。 アーカイブコマンド[[18]](#footnote-18)に対応したコマンドを 設定すること。 (参考) アーカイブコマンドでgzipを使用した場合は リストアコマンドにもgzipを使用すること。 コマンド設定例を以下に示す。 '/bin/gzip -cd /dbfp/pgarch/arc1/%f.gz > %p' |
| repuser | repuser | 『レプリケーションユーザの作成』で作成した レプリケーションユーザを設定する。 |
| primary\_conninfo\_opt | keepalives\_idle=60 keepalives\_interval=5 keepalives\_count=5 | TCPキープアライブ用の制御パラメータを 設定する。PG-REXでは左記の設定値を 推奨する。 |
| stop\_escalate | 0 | このパラメータは、Primary停止時に PostgreSQLの高速シャットダウンを実行して から即時シャットダウンにエスカレーション するまでの待ち時間を設定する。 0を設定した場合、高速シャットダウンは実行 されずに、即座に即時シャットダウンが実行 される。 |
| xlog\_check\_count | 0 | このパラメータは、2つのノードを同時に 起動する際、どちらがPrimaryとして 起動するかをチェックする回数を設定する （デフォルトは3）。 このパラメータを設定すると、Primary起動の 際にmonitor間隔×設定値の時間だけ Standbyで待機することになり、その分 Primaryとして起動する時間が遅くなる。 本マニュアルでは、2つのノードを同時に 起動するケースがないため、0を設定する。 |

### リソース（ネットワーク監視）の設定

『pm\_pcsgen環境定義書』の表7-1にネットワーク監視を設定します。なお、PG-REXではS-LANを監視します。監視するネットワークのIPアドレスとしてS-LANのデフォルトゲートウェイ等を設定してください。

### リソース（STONITH）の設定

『pm\_pcsgen環境定義書』の表8-1、表9-1にSTONITHを設定します。

### リソース（Standby側仮想IP）の設定

Standbyへのデータベース接続を利用する場合は、『pm\_pcsgen環境定義書』の表7-1にStandby側接続用の仮想IPを設定します。Standbyへのデータベース接続を利用しない場合は、『pm\_pcsgen環境定義書』の表4-1、表7-1、表9-2からipaddr-standbyの設定値をコメントアウトします。

### xmlファイルの作成

以下の操作を行い、xmlファイルを作成します。なお、作成したxmlファイルはPrimary起動時にPacemakerに反映されます。

本作業はrootユーザで行います。

1. Microsoft® Excelを使って『pm\_pcsgen環境定義書』の修正したシートをCSV形式で保存します。

* 本書ではファイル名は「PG-REX14\_pm\_pcsgen\_env.csv」とします。

1. 保存したcsvファイル（PG-REX14\_pm\_pcsgen\_env.csv）を、pgrex01に転送します。

* ファイル転送の際に文字コード変換を行わないよう注意してください。

1. pgrex01上で、pm\_pcsgenコマンドを使用し、転送したcsvファイルをxmlファイルに変換します。

* 本書では、変換後のファイル名を「PG-REX14\_pm\_pcsgen\_env.xml」とします[[19]](#footnote-19)。

|  |
| --- |
| # pm\_pcsgen PG-REX14\_pm\_pcsgen\_env.csv |

## アンインストール

PG-REXをアンインストールするためには両ノードで作業をします。アンインストールの前にPG-REXが起動している場合は『PG-REXの停止』の手順に従って停止してください。必要なデータがある場合はバックアップを取得し、バックアップディレクトリから退避してください。

本作業はrootユーザで行います。

1. PostgreSQLをアンインストールします。



|  |
| --- |
| # rpm -e postgresql14-contrib-14.0-1PGDG.rhel8.x86\_64 # rpm -e postgresql14-server-14.0-1PGDG.rhel8.x86\_64 # rpm -e postgresql14-docs-14.0-1PGDG.rhel8.x86\_64 # rpm -e postgresql14-14.0-1PGDG.rhel8.x86\_64 # rpm -e postgresql14-libs-14.0-1PGDG.rhel8.x86\_64  ※ バージョンは適宜読み替えてください。 |

1. DBクラスタディレクトリ配下のファイル、作成した環境変数のファイルおよびMD5暗号化パスワード認証の自動化ファイル(.pgpass)を削除します。



|  |
| --- |
| # rm -r /dbfp/pgdata/data/\* # rm /var/lib/pgsql/.bash\_profile # rm /var/lib/pgsql/.pgpass |

1. その他、作成した環境に合わせて以下の作業を行います。
   * DBクラスタディレクトリのアンマウント
   * WALディレクトリ配下のファイルの削除・アンマウント
   * アーカイブディレクトリ配下のファイルの削除・アンマウント
   * 運用中に取得したDBクラスタのバックアップの削除



1. Pacemakerをアンインストールします。

既存のHAクラスタを破棄します。（いずれか1つのノードで実行）

|  |
| --- |
| # pcs cluster destroy --all |

ノードの認証を破棄します。（いずれか1つのノードで実行）

|  |
| --- |
| # pcs host deauth pgrex01 pgrex02 |

Pacemaker関連パッケージをアンインストールします。（2つのノードで実行）

|  |
| --- |
| # yum erase pcs pacemaker fence-agents-all -y |

Pacemaker関連のファイルを削除します。（2つのノードで実行）

|  |
| --- |
| # rm -fr /var/lib/pcsd/ /var/lib/pacemaker /var/lib/corosync /var/log/pacemaker /var/log/cluster /etc/corosync /mnt/HighAvailability /etc/systemd/system/corosync.service /etc/sysconfig/pacemaker.rpmsave /etc/sysconfig/sbd.rpmsave |

haclusterユーザおよび、haclientグループを削除します。（2つのノードで実行）

|  |
| --- |
| # userdel hacluster # groupdel haclient |

pm\_extra\_toolsをアンインストールします。（2つのノードで実行）

|  |
| --- |
| # yum erase pm\_extra\_tools -y |

1. BIOSの設定でACPI Soft-Off機能を無効にした場合はBIOSでACPI Soft-Off機能を有効に戻します。 /etc/systemd/logind.confを修正した場合は、以下手順に従いlogind.confを修正し有効に戻します。

ACPI Soft-off機能を有効化するため、/etc/systemd/logind.confを以下の通り修正します。（2つのノードで実行）

|  |
| --- |
| HandlePowerKey=poweroff |

systemd-logindサービスを再起動し、設定を反映させます。

|  |
| --- |
| # systemctl restart systemd-logind.service |

ACPI Soft-Off機能が有効化されていることを確認します。

|  |
| --- |
| # loginctl show-user ：(省略) HandlePowerKey=poweroff ：(省略) |

1. PG-REX運用補助ツールをアンインストールします。

* RPMパッケージのアンインストールを行います。

|  |
| --- |
| # yum remove pg-rex\_operation\_tools\_script-14.0-1.el8.noarch.rpm IO\_Tty-1.11-1.el8.x86\_64.rpm Net\_OpenSSH-0.62-1.el8.x86\_64.rpm  ※ バージョンは適宜読み替えてください。 |

# 起動と停止

本章では、PG-REX運用補助ツールを用いてPrimaryとStandbyの起動・停止方法について説明します。

本章の作業はrootユーザで行います。

コマンドを直接実行してPrimaryとStandbyの起動・停止をする場合は、『コマンド直接実行の運用』を参照してください。

## PG-REXの起動

PG-REXの起動は、一方のノードでPrimaryを起動させ、起動完了後、もう一方のノードでStandbyを起動させます。どちらのノードをPrimary、Standbyとして稼働させるかは、ユーザが決定します。

* Primaryの起動手順については、『Primaryの起動』を参照してください。
* Standbyの起動手順については、『Standbyの起動』を参照してください。

## Primaryの起動

本節では、Primaryの起動手順を説明します。

1. どのノードをPrimaryとして起動するか決定します。 PG-REXでは、最新のDBデータを持つノードをPrimaryとして起動しなければなりません。古いDBデータを持つノードをPrimaryとして起動すると、その古い分だけDBデータは失われてしまいます。

* 以下は、Primaryとして起動するノードを決めるときの考え方の例です。
* + DBクラスタが片方のノードのみに存在し、そのDBクラスタを使ってPG-REXを起動する場合(初めてPrimaryを起動する場合を含む)は、DBクラスタが存在するノードをPrimaryとして起動する。
  + DBクラスタが両ノードに存在する場合は、直前までPrimaryとして稼働していたノードをPrimaryとして起動する。
  + 既存のDBクラスタを使わず(もしくは既存のDBクラスタが壊れている)、以前に取得したベースバックアップからPG-REXを起動する場合は、そのベースバックアップを展開したノードをPrimaryとして起動する。
* 以降の手順では、pgrex01をPrimaryとして起動します。

1. ベースバックアップからPrimaryを起動する場合に限り、pgrex01でPostgreSQL単体のアーカイブリカバリを行います。アーカイブリカバリの手順については、『PostgreSQLドキュメント』を参照してください。アーカイブリカバリが完了したら、PostgreSQLを停止します。

* 本作業のみpostgresユーザで行う必要があります。

|  |
| --- |
| $ pg\_ctl start サーバの起動完了を待っています.... ：(略) 完了 サーバ起動完了 $ 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 |

【注意】

PG-REXでは、アーカイブリカバリをさせながらPrimaryを起動することを推奨しません。これは、アーカイブリカバリにより起動に時間がかかり、Pacemakerによって起動失敗とみなされてしまう可能性があるからです。そのため、Primaryでアーカイブリカバリを行う場合は、Pacemaker経由ではなく、まずはPostgreSQL単体で起動させるようにしてください。アーカイブリカバリの完了後、PostgreSQLを停止させた上で、Primaryの起動の手順を行います。これにより、Primary起動時のアーカイブリカバリは必要なくなるため、Primaryの起動に時間がかかることはありません。

また、PG-REXでリカバリを実施する場合は必ず最新の状態までリカバリされます。 アーカイブリカバリを行う際にユーザが追加した設定は、アーカイブリカバリ終了後にすべて削除してください。

1. pgrex01で起動禁止フラグのファイルが存在する場合削除します。



|  |
| --- |
| # rm /var/lib/pgsql/tmp/PGSQL.lock |

1. Primaryを起動します。

【初回もしくは新しいxmlファイルを反映させる場合】

Primary初回起動時、もしくは既存のPacemakerの設定をクリアして新しいxmlファイルを反映させる場合は、xmlファイルを指定して、Primaryを起動します。

|  |
| --- |
| # pg-rex\_primary\_start PG-REX14\_pm\_pcsgen\_env.xml root@192.168.2.2's password: パスワードが入力されました 1. Pacemaker および Corosync が停止していることを確認 ...[OK] 2. 稼働中の Primary が存在していないことを確認 ...[OK] 3. 起動禁止フラグの存在を確認 ...[OK] 既存のHAクラスタは再作成されます。よろしいですか？ (y/N) y 4. HAクラスタ の破棄 ...[OK] 5. HAクラスタ の作成 ...[OK] 6. Pacemaker 起動 ...[OK] 7. xml ファイルの反映 ...[OK] 8. Primary の起動確認 ...[OK] ノード(pgrex01)が Primary として起動しました |

【2回目以降の場合】

Primaryを起動します。

|  |
| --- |
| # pg-rex\_primary\_start root@192.168.2.2's password: パスワードが入力されました 1. Pacemaker および Corosync が停止していることを確認 ...[OK] 2. 稼働中の Primary が存在していないことを確認 ...[OK] 3. Primary として稼働することが出来るかを確認 ...[OK] 4. 起動禁止フラグの存在を確認 ...[OK] 5. Pacemaker 起動 ...[OK] 6. Primary の起動確認 ...[OK] ノード(pgrex01)が Primary として起動しました |

【注意】

以下のいずれかのGUCを変更した後にPrimary起動を試みるとPostgreSQLが起動しないことがあります。新しい設定値は、変更前の設定値以上にする必要があるためです。

* max\_connections
* max\_worker\_processes
* max\_prepared\_transactions
* max\_locks\_per\_transaction

より詳しい説明については、PostgreSQLドキュメント[[20]](#footnote-20)を参照してください。

新しい設定値が変更前の設定値未満の場合、PostgreSQLのみを一度起動・終了する必要があります。postgresユーザでpg\_ctl startとpg\_ctl stopを実行してください。

1. pgrex02のSTONITH履歴をクリアします。

|  |
| --- |
| # stonith\_admin -c -H pgrex02 |

## Standbyの起動

本節では、Standbyの起動手順を説明します。以降の手順では、pgrex02をStandbyとして起動します。

1. pgrex02で起動禁止フラグのファイルが存在する場合削除します。



|  |
| --- |
| # rm /var/lib/pgsql/tmp/PGSQL.lock |

1. pgrex02でStandbyを起動します。同期する必要の無い不要なアーカイブログが多い場合は、Standbyの起動の前にアーカイブログの削除を行なってください。アーカイブログの削除は『PostgreSQLアーカイブログの削除』を参照してください。



|  |
| --- |
| # pg-rex\_standby\_start root@192.168.2.1's password: パスワードが入力されました 1. Pacemaker および Corosync が停止していることを確認 ...[OK] 2. 稼働中の Primary が存在していることを確認 ...[OK] 3. 起動禁止フラグが存在しないことを確認 ...[OK] |

1. 起動方法を選択します。

* 以下の手順では、ベースバックアップを取得する場合を示します。

|  |
| --- |
| 4. DB クラスタの状態を確認 4.1 現在のDBクラスタのまま起動が可能か確認 DB クラスタが存在していません ...[NG] 4.2 巻き戻しを実行することで起動が可能か確認 DB クラスタが存在していません ...[NG] 4.3 ベースバックアップを取得することが可能か確認 ...[OK] 以下の方法で起動が可能です b) ベースバックアップを取得してStandbyを起動 q) Standbyの起動を中止する 起動方法を選択してください(b/q) b 5. IC-LAN が接続されていることを確認 ...[OK] 6. Primary からベースバックアップ取得 22631/22631 kB (100%), 1/1 テーブル空間 NOTICE: all required WAL segments have been archived ...[OK] 7. Primary のアーカイブディレクトリと同期 000000010000000000000002.partial 00000002.history 000000020000000000000003.00000028.backup 000000010000000000000001 000000020000000000000002 000000020000000000000003 ...[OK] 8. Standby の起動 (アーカイブリカバリ対象 WAL セグメント数: 1) ...[OK] 9. Standby の起動確認 ...[OK] ノード(pgrex02)が Standby として起動しました |

## PG-REXの停止

PG-REXを停止するには、Standbyを停止させ、Standbyの停止完了後にPrimaryを停止させます。

Primaryから停止した場合、フェイルオーバが発生しますので、ご注意ください。

* Primaryの停止手順については、『Primaryの停止』を参照してください。
* Standbyの停止手順については、『Standbyの停止』を参照してください。

この手順でPG-REXを停止させた場合、次にPG-REXを起動するときには、Primaryからのベースバックアップの取得は必要ありません。

## Standbyの停止

本節では、Standbyの停止手順を説明します。

本作業は停止対象のノードで行います。

1. Standbyを停止します。



|  |
| --- |
| # pg-rex\_stop Standby を停止します 1. Pacemaker 停止 ...[OK] 2. Pacemaker 停止確認 ...[OK] PG-REX の Standby (pgrex02)を停止しました |

## Primaryの停止

本節では、Primaryの停止手順を説明します。Standby稼働中にPrimaryを停止した場合、フェイルオーバが発生することに注意してください。

本作業は停止対象のノードで行います。

1. Primaryを停止します。



|  |
| --- |
| # pg-rex\_stop Primary を停止します 1. Pacemaker 停止 ...[OK] 2. Pacemaker 停止確認 ...[OK] PG-REX の Primary (pgrex01)を停止しました |

【注意】

Standby稼働中の場合、以下の問い合わせが出力されます。フェイルオーバしても問題がなければ「y」を入力してください。

|  |
| --- |
| Standby がまだ起動しています ノード切り替えが目的の場合は pg-rex\_switchover コマンドの使用を推奨します 今停止すると F/O しますが本当に停止しても宜しいですか？ (y/N) |

## PostgreSQL停止中のノードの停止

本節では、PostgreSQL停止中(Stopped)のノードのPacemakerの停止手順を説明します。主に、運用中に故障が発生した後、復旧するための手順の一つとして行われます。

本作業は停止対象のノードで行います。

1. PostgreSQL停止中(Stopped)のノードのPacemakerを停止します。



|  |
| --- |
| # pg-rex\_stop PostgreSQL の状態を確認できませんでした Pacemaker を停止します 1. Pacemaker 停止 ...[OK] 2. Pacemaker 停止確認 ...[OK] ノード(pgrex01)で Pacemaker を停止しました |

# メンテナンス時の対応

本章は、PostgreSQLのバックアップ取得の操作手順、アーカイブログ削除の操作手順、およびPG-REX(PrimaryまたはStandby)の停止を伴うメンテナンス(PostgreSQLのマイナーバージョンアップやハードウェアの増設等の作業)時のノードの停止、起動手順について記述します。

作業を行う際には、記述されている手順以外は行なわない様にしてください。

## PostgreSQLのバックアップ

PG-REXでは、PostgreSQL単独の場合と同じ方法で、Primaryからオンライン物理バックアップを取得することができます。しかし、バックアップ取得は、大量のI/Oを発生させる負荷の高い処理であるため、それがPrimaryで実行されるオンライン負荷に影響を与えないようにStandbyで取得することも可能です。

本節では、PostgreSQLのバックアップを取得する手順について記述します。

本節の作業は、postgresユーザで行います。

### PostgreSQLのバックアップ

PostgreSQLのバックアップを取得する手順は、『PostgreSQLドキュメント』[[21]](#footnote-21) の手順に従い実施してください。また、フェイルオーバが発生した場合は、新Primaryからバックアップを取得し直してください。旧Primaryから取得したバックアップは使用できない可能性があります。

【参考】

以下のようなコマンドを実行することになります。ここでは、ipaddr-replicationを使用してPrimaryからバックアップを取得しています。

|  |
| --- |
| $ pg\_basebackup -h 192.168.2.3 -U repuser -D /backupdirectory -X stream -P |

### その他のバックアップツールを使用する場合の注意事項

ここではpg\_basebackup 以外のツールを用いてデータベースのバックアップを取得して使用する際の注意事項を説明します。

【バックアップツールの選択】

アーカイブログが利用できるタイプのバックアップツールを選択してください。PG-REXではPostgreSQLの継続的アーカイブ機能を使用してアーカイブログを生成します。運用途中でprimary, standby両方のデータベースファイルが失われた場合であっても、継続的アーカイブが利用できるバックアップファイルがあれば、データベースファイルが失われる直前の状態のデータベースを再現することが出来ます。

【アーカイブログの削除】

バックアップの取得にpg-basebackup を用いformat として plain (デフォルト設定)を指定した場合[[22]](#footnote-22)に限り 、アーカイブログの削除にpg-rex\_archivefile\_deleteが使用できます。それ以外の場合は「コマンド直接実行の運用」の章の「アーカイブログの削除」の節にある説明を参照して、不要なアーカイブログを削除します。

【バックアップファイルからのリストアとリカバリ】

使用しているバックアップツールの定める手順にそって、バックアップファイルをリストアします。次いでPostgreSQL本体でリカバリすることでprimaryデータベースとして使用できるようになります。その際の注意事項は「起動と停止」の章の「Primaryの起動」の節にありますので参照してください。

## アーカイブログの削除

本節では、PG-REX運用補助ツールを用いてデータベースの復旧に必要のないファイルをアーカイブディレクトリから削除する手順について記述します。

本節の作業はroot、もしくはPG-REXで管理しているDBクラスタを作成したユーザのみ実行可能です。

コマンドを直接実行してデータベースの復旧に必要のないファイルをアーカイブディレクトリから削除する手順については、『コマンド直接実行の運用』を参照してください。

### PostgreSQLアーカイブログの削除

運用を続けていくにしたがってアーカイブファイルの容量は増えていくため、継続的な運用のためには適宜削除を行う必要があります。

PostgreSQLのアーカイブログ(および付随するバックアップ履歴ファイル[[23]](#footnote-23)、タイムライン履歴ファイル[[24]](#footnote-24))は、PrimaryおよびStandbyのリカバリに加え、『PostgreSQLのバックアップ』の方法で取得したバックアップからのリカバリのいずれでも必要とならない場合に削除できます。

アーカイブログを削除したいノードで、pg-rex\_archivefile\_deleteコマンドを実行することで、これらの不要なファイルを自動的に判断し、削除または移動します。

以下に、アーカイブログ削除の実行例を示します。

ここでは、保存すべき最古のベースバックアップを格納しているディレクトリを /somewhere/basebackups/oldestとします。

|  |
| --- |
| $ pg-rex\_archivefile\_delete -m /somewhere/basebackups/oldest  \*\*\*\* 1. 実行準備 \*\*\*\* 移動モードで実行します 環境設定ファイル (pg-rex\_tool.conf) を読み込みます postgres@192.168.2.2's password: パスワードが入力されました 両ノードの名前を取得します cib.xml ファイルを読み込みます  \*\*\*\* 2. WAL ファイル名の取得 \*\*\*\* 指定されたバックアップからリカバリを行うために必要な最初の WAL ファイル名を取得します "00000051000000010000003A" 自身のノード "pgrex01" の現時点の PGDATA "/dbfp/pgdata/data "からリカバリに必要な最初の WAL ファイル名を取得します "00000052000000010000003E" 相手のノード "pgrex02" の現時点の PGDATA "/dbfp/pgdata/data "からリカバリに必要な最初の WAL ファイル名を取得します "00000052000000010000003D"  \*\*\*\* 3. 削除基準の算出 \*\*\*\* 削除基準を "00000051000000010000003A" としました  \*\*\*\* 4. アーカイブログの削除 \*\*\*\* 削除対象のリストに "0000004F0000000100000035.00000020.backup"を追加します 削除対象のリストに "0000004D0000000100000034" を追加します :(略) 移動先ディレクトリ "/dbfp/pgarch/arc1/20211119\_150226"を作成しました -- 移動 -- 0000004F0000000100000035.00000020.backup :(略) アーカイブログの移動に成功しました 移動モード実行のため、移動したファイルは"/dbfp/pgarch/arc1/20211119\_150226" に格納されています |

この実行例ではコマンドに -mオプションを指定しているため、作業の終了後には /dbfp/pgarch/arc1/20211119\_150226 ディレクトリに不要となったファイルが格納されています。これを削除するか別のディスク等に移動させるなどすることで作業は完了します。

-m オプションの代わりに -r オプションを指定するとツール自身がファイルを削除します。

## 計画的なノード切り替え

PrimaryとStandbyという2つのノードの役割を入れ替えることを、PG-REXではノード切り替えまたはスイッチオーバと呼びます。ここでは、PG-REX運用補助ツールのスイッチオーバ機能(pg-rex\_switchover)を使用したノード切り替え手順を説明します。

本節の作業は、rootユーザで行います。

運用補助ツールを使用せずに直接コマンドを実行してノード切り替えを実施したい場合は、『コマンド直接実行の運用』を参照してください。

### ノード切り替え

運用補助ツールを使用したノード切り替えは、PG-REXのPrimaryとStandbyのどちらのノードでも実施することができます。以下に、pgrex01でノード切り替えを実行する場合の手順を示します。

1. pg-rex\_switchoverコマンドを実行します。パスワードの入力要求に対して、相手ノードのrootユーザのパスワードを入力してください。



|  |
| --- |
| # pg-rex\_switchover root@192.168.2.2's password: パスワードが入力されました |

1. 表示された現在およびノード切り替え後のHAクラスタ状態を確認します。ノード切り替えを実行しても良い場合は[y]を入力します。

* なお、ノード切り替え中は可用性が保証されないことに注意してください。

|  |
| --- |
| \*\*\*\* 実行準備 \*\*\*\* 1. 環境設定ファイル (pg-rex\_tools.conf)の読み込みと両ノードの名前を取得 ...[OK] 2. 現在およびノード切り替え後のHAクラスタ状態を確認  [ 現在 / ノード切り替え後のHAクラスタ状態 ] Primary : pgrex01 -> pgrex02 Standby : pgrex02 -> pgrex01  ノード切り替え中は可用性が保証されません ノード切り替えを実行してもよろしいでしょうか？ (y/N) y |

1. ノード切り替えが完了するまで待機し、コマンドが正常に完了したことを確認します。異常終了した場合は、HAクラスタ状態が不確定であり、データベースサービスが停止している可能性があります。この場合、元の状態への復旧は自動で実施されません。『故障対応』に従い、手動復旧を試みてください。



|  |
| --- |
| \*\*\*\* ノード切り替えを実行 \*\*\*\* 3. Pacemaker の監視を停止 ...[OK] 4. Primary (pgrex01) の PostgreSQL を停止 ...[OK] 5. Pacemaker の監視を再開しノード切り替えを実行 ...[OK] 6. pgrex02 が新 Primary になったことを確認  \*\*\*\* pgrex02 が Primary として起動しました \*\*\*\*  7. pgrex01 の Pacemaker を停止 ...[OK] 8. pgrex01 で Standby を起動 00000011000000000000000C 00000012000000000000000E 00000013.history  \*\*\*\* pgrex01 が Standby として起動しました \*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\* ノード切り替えが正常に完了しました \*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  [ 現在のHAクラスタ状態 ]  Primary : pgrex02  Standby : pgrex01 |

【注意】

Pacemakerの監視を停止中(上記の4～6の間)にコマンドが異常終了した場合は、Pacemakerの監視が停止している可能性があるため、pcs statusコマンドを実行しHAクラスタ状態を確認してください。pcs statusコマンドの実行結果のリソース情報表示部に"unmanaged"が表示されている場合はPacemakerの監視が停止しています。Pacemakerの監視が停止している場合は、Pacemakerの監視を再開してください。

Pacemakerの監視を再開するには以下のコマンドを実行します。

|  |
| --- |
| # pcs property set maintenance-mode= |

## 計画的なメンテナンスのためのノードの停止、起動

本節では、PrimaryおよびStandbyの計画されたメンテナンス(PostgreSQLのマイナーバージョンアップやハードウェアの増設等の作業)を実施する時のノードの停止、メンテナンス後の起動の操作手順を記述します。

本節の作業は、rootユーザで実行します。

### メンテナンスのためのノードの停止

メンテナンス対象のノードを停止する手順を以下に示します。

1. メンテナンス対象のノードの停止

* メンテナンス対象のノードの運用を停止します。停止手順については、『起動と停止』の章を参照してください。

1. データベースサービス継続の確認

* 起動中のノードでデータベースサービスが継続していることを、pcs statusコマンドを実行して確認します。
  + ノード情報表示部で、起動中のノードの状態が"Online"になっていること。
  + リソース情報表示で、PG-REXリソース(pgsql-clone)に、"Master pgrex02"のように起動中のノードが表示されていること。
  + リソース情報表示で、全てのIPaddr2リソース(ipaddr-primary、ipaddr-standby、ipaddr-replication)が、起動中のノードで稼働していること。
* 以下に、pgrex01停止後、pgrex02で確認した場合の例を示します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略）   \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） |

### メンテナンス後のノードの起動

メンテナンス作業終了後は、メンテナンスのために停止したノードをStandbyとして起動します。起動手順については、『起動と停止』の章を参照してください。

起動後、PrimaryとStandbyを切り替えたい場合は、『計画的なノード切り替え』に従い切り替えます。

# 故障対応

本章では、HAクラスタに故障が発生した場合の監視者が行う作業について説明します。

監視者による作業にはHAクラスタ状態確認と復旧があります。

* HAクラスタ状態確認では、HAクラスタに発生した故障箇所を特定し、保守者への報告と修復作業の依頼を実施します。
* 復旧では、データベースサービス継続に必要となる処置及び、保守者の修復作業終了後にHAクラスタを故障発生前の状態に戻す作業を実施します。

保守者は、監視者からの修復作業依頼により、故障したアプリケーションの詳細解析、原因調査、修復作業を実施します。故障の種類によっては、復旧の際に保守者に作業を委ねている箇所があります(保守者介在処理と呼びます)。

なお、本作業にはスーパーユーザ権限が必要となります。

STONITHが発生した場合は、復旧手順の最後に以下のコマンドで両ノードに対してSTONITH履歴のクリアを実施してください。

|  |
| --- |
| # stonith\_admin -c -H <ノード名> |

## 前提条件

この章の前提条件を以下に示します。

* 両ノードが稼働している状態でpgrex01の故障が起きたと仮定しています。一般的に多重故障はPG-REXの適用範囲外となりますが、STONITHの失敗については対応しており本章の説明にも含まれています。

## 監視コマンド表示確認方法

HAクラスタに発生した故障を特定するために、pcs status --fullコマンド実行時に表示されるノード情報、リソース情報、属性情報、故障回数、制御エラー情報、及びcorosync-cfgtool -sコマンド実行時に表示されるIC-LANの通信状態を確認する必要があります。

以下にpcs statusコマンドの実行時の表示例を示します。

なお、pcs status --fullコマンドを実行し、"Error: error running crm\_mon, is pacemaker running? Error: cluster is not available on this node"と表示された場合、pcs status --fullコマンドを実行したノードのPacemakerは停止しています。

|  |
| --- |
| # pcs status --full Cluster name: HAクラスタ名 Cluster Summary:  \* Stack: corosync  \* Current DC: pgrex01 (1) バージョン - partition with quorum  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ]  Full list of resources:   \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pgrex02  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01  Node Attributes:  \* Node pgrex01 (1):   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 0000000005000060   \* pgsql-status : PRI   \* ping-status : 1  \* Node pgrex02 (2):   \* master-pgsql : 100   \* pgsql-data-status : STREAMING|SYNC   \* pgsql-status : HS:sync   \* ping-status : 1  Migration Summary:  PCSD Status:   pgrex01: Online   pgrex02: Online  Daemon Status:   corosync: active/disabled   pacemaker: active/disabled   pcsd: active/enabled |

以下にcorosync-cfgtoolコマンドの実行時の表示例を示します。

|  |
| --- |
| # corosync-cfgtool -s Printing link status. Local node ID 1 LINK ID 0 addr = 192.168.1.1 status:  nodeid 1: localhost  nodeid 2: connected LINK ID 1 addr = 192.168.3.1 status:  nodeid 1: localhost  nodeid 2: connected |

### 表示部説明

pcs statusコマンド実行時の各表示部について説明します。

【ノード情報表示部】

|  |
| --- |
| \* Online: [ pgrex01 (1) pgrex02 (2) ] |

各ノードの名前やPacemakerの稼働状態が表示されます。稼働状態にはOnline、OFFLINE、UNCLEAN(online)、UNCLEAN(offline)が存在します。

また、Pacemaker稼働時は、HAクラスタを構成しているノードが"[]"内に表示されます。

ノード名の横に括弧で表示されている数字はノードIDを表します。

【リソース情報表示部】

|  |
| --- |
| Full list of resources:   \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):    Started pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pgrex02  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 |

リソースID、リソースの稼働状態や稼働ノードの名前が表示されます。

各リソースが示す稼働状況を、以下に示します。

■ pgsql-clone(PG-REXリソース)

PG-REXリソースの稼働状況表示一覧

| pcs status実行結果の表示 | 稼働状況の説明 |
| --- | --- |
| Master pgrex01 Slave pgrex02 | 両ノードでリソースが稼働中 |
| Master pgrex01 Stopped | 片方のノード(pgrex01)でリソースが稼働中 |
| Stopped Stopped | 両ノードでリソースが停止中 |

■ ipaddr-primary, ipaddr-replication, ipaddr-standby(IPaddr2リソース), fence1-ipmilan, fence2-ipmilan(STONITHリソース)

IPaddr2リソースおよびSTONITHリソースの稼働状況表示一覧

| pcs status実行結果の表示 | 稼働状況の説明 |
| --- | --- |
| Started pgrex01 | pgrex01でリソースが稼働中 |
| Stopped | リソースが停止中 |

■ ping-clone(PINGリソース)

PINGリソースの稼働状況表示一覧

| pcs status実行結果の表示 | 稼働状況の説明 |
| --- | --- |
| Started pgrex01 Started pgrex02 | 両ノードでリソースが稼働中 |
| Started pgrex01 Stopped | 片方のノード(pgrex01)でリソースが稼働中 |
| Stopped Stopped | 両ノードでリソースが停止中 |

【属性情報表示部】

|  |
| --- |
| Node Attributes:  \* Node pgrex01 (1):   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 0000000005000060   \* pgsql-status : PRI   \* ping-status : 1  \* Node pgrex02 (2):   \* master-pgsql : 100   \* pgsql-data-status : STREAMING|SYNC   \* pgsql-status : HS|SYNC   \* ping-status : 1 |

各ノードにおけるネットワーク経路の監視状態、およびPG-REXリソース状態を表示します。

pgrex01の各監視先の属性の正常値の例を以下に示します。

監視先の属性の正常値

| 属性 | 正常値 | 説明 |
| --- | --- | --- |
| master-pgsql | 1000(pgrex01), 100(pgrex02) | PG-REXリソースの属性で、 PacemakerがPrimaryおよびStandby のリソースの状態を管理する |
| pgsql-data-status | LATEST(pgrex01), STREAMING|SYNC(pgrex02) | PG-REXリソースの属性で、 PostgreSQLのデータの状態を示す |
| pgsql-master-baseline | LSN(pgrex01), 表示なし(pgrex02) | PG-REXリソースの属性で、 PostgreSQLがpromote(Primaryに なる)直前のLSNの値を示す |
| pgsql-status | PRI(pgrex01), HS:sync(pgrex02) | PG-REXリソースの属性で、 PostgreSQLの現在の遷移状態を 示す |
| pgsql-xlog-loc | 正常時は属性そのもの が表示されない | PG-REXリソースの属性で、 起動時にPrimaryが存在しない場合 に、Primaryになれるかどうかを 決定するために設定される |
| ping-status | 1 | ネットワーク経路の状況を示す |

【故障回数表示部】

|  |
| --- |
| Migration Summary: |

ノード毎に故障したリソースIDと故障許容回数(migration-threshold)、故障した回数が表示されます。

以下に故障回数表示部の出力フォーマットを示します。

|  |
| --- |
| Migration Summary:  \* Node pgrex01:   \* pgsql: migration-threshold=1 fail-count=1 last-failure='日時表示'     (1)           (2)               (3)  (1) 故障リソースID (2) 故障許容回数 (3) 故障回数 |

【制御エラー情報表示部(※ リソース故障時のみ表示)】

|  |
| --- |
| Failed Resource Actions: |

制御エラー情報表示部は制御エラーが発生した場合のみ表示されます。

制御エラーが発生したリソースIDと検知オペレーション(start/stop/monitor/promote/demote)、故障発生ノード、リターンコード、エラー内容("error"、"TimedOut"等)が表示されます。

以下に制御エラー情報表示部の出力フォーマットを示します。

|  |
| --- |
| Failed Resource Actions:  \* pgsql\_monitor\_9000 on pgrex01 'not running' (7): call=79, status=complete,    (1)    (2)             (3)        (4)     (5) exitreason='', last-rc-change='日時表示', queued=0ms, exec=0ms  (1) 故障リソースID (2) 検知オペレーション (3) 故障発生ノード名 (4) エラー内容 (5) リターンコード |

【Fencing History情報表示部(※履歴がある場合のみ表示)】

|  |
| --- |
| Fencing History:  \* reboot of pgrex02 successful: delegate=pgrex01,               (1)                          (2) client=pacemaker-controld.21654, origin=pgrex01, completed=’日時表示'  (1) STONITH対象ノード (2) STONITH実行ノード |

corosync-cfgtool -sコマンド実行時の表示部について説明します。

このコマンドでは、IC-LANのIPアドレス毎にPrimaryノードおよびStandbyノードの通信状態が表示されます。

以下に出力フォーマットを示します。

|  |
| --- |
| # corosync-cfgtool -s Printing link status. Local node ID 1 LINK ID 0 addr = 192.168.1.1 (1) status:  nodeid 1: localhost  (2)  nodeid 2: connected  (3) LINK ID 1 addr = 192.168.3.1 (4) status:  nodeid 1: localhost  (2)  nodeid 2: connected  (3) (1) IC-LAN(eno1)のIPアドレス (2) ノードID[[25]](#footnote-25)が1のノードとのIC-LANの通信状態 (3) ノードID[[26]](#footnote-26)が2のノードとのIC-LANの通信状態 (4) IC-LAN(eno3)のIPアドレス |

IC-LANの通信状態を、以下に示します。

IC-LANの通信状態

| corosync-cfgtool実行結果の表示 通信状態の説明 |
| --- |
| connected 接続状態 |
| disconnected 未接続状態 |

### 正常状態確認方法

pcs status --fullコマンド実行及びcorosync-cfgtool -sコマンド実行結果の全ての表示部において正常状態の確認ができた場合、HAクラスタは正常状態です。

● pcs status --fullコマンド確認項目

【ノード情報表示部】

pgrex01とpgrex02のノード状態が"Online"状態になっていることを確認する

【『リソース情報表示部』の『PG-REXリソース、IPaddr2リソース』】

それぞれのノードで、PG-REXリソース(pgsql-clone)、IPaddr2リソース(ipaddr-primary、ipaddr-standby、ipaddr-replication)が、以下のとおりに稼働していることを確認する。

* pgrex01側 ： pgsql-clone(Primary), ipaddr-primary, ipaddr-replication
* pgrex02側 ： pgsql-clone(Standby), ipaddr-standby

【『リソース情報表示部』の『STONITHリソース』】

STONITHリソース(fence1-ipmilan、fence2-ipmilan)が、それぞれのノードで稼働していることを確認する。

【『リソース情報表示部』の『PINGリソース』】

ネットワーク経路監視用PINGリソース(ping-clone)が、それぞれのノードで稼働していることを確認する。

【属性情報表示部】

PG-REXリソース、ネットワーク経路の監視状態の各属性が正常値であることを確認する。

【制御エラー情報表示部】

制御エラー情報が表示されていないことを確認する。

【Fencing History情報表示部】

Fencing History情報が表示されていないことを確認する。

● corosync-cfgtool -sコマンド確認項目

corosync-cfgtool -sコマンドでは、出力が「localhost」、「connected」であることを確認します。

## 故障箇所特定手順

本節では、HAクラスタに発生した故障を特定する手順について説明します。

故障箇所特定を以下の手順で行います。

1. ノード情報・リソース情報・故障回数表示部の確認

* pcs statusコマンドの実行結果から、ノードの起動状態の確認、D-LAN故障の特定、ハードウェア故障・リソース故障の切り分けを行います。

1. 属性情報表示部の確認

* ネットワーク経路故障のハードウェア故障を特定します。

1. corosync-cfgtool -sによるIC-LAN状態の確認

* IC-LAN故障を特定します。

1. 制御エラー情報表示部の確認

* 故障リソース、故障ノード、故障オペレーションを特定します。

1. /var/log/messagesの確認

* Pacemaker停止ノードに発生した故障を特定します。

### ノード情報・リソース情報・故障回数表示部の確認

ノード情報・リソース情報・故障回数表示部の確認では、ノードの起動状態を確認し、その状態に応じた確認を行います。pgrex01、pgrex02それぞれでpcs status --fullを実行し、以下の順に確認します。

1. ノードの起動状態の確認
2. D-LAN故障の特定
3. ハードウェア故障・リソース故障の特定

【ノードの起動状態の確認】

pgrex02のpcs status --fullの実行結果のノード表示部が以下に該当する場合、『6.3.5. /var/log/messagesの確認』へ進みます。

|  |
| --- |
| \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)]  ※ pgrex01がOFFLINEになっている |

【D-LAN故障の特定】

pgrex01でpcs status --fullコマンドを実行し、リソース情報、属性情報を確認します。

|  |
| --- |
| ：（略）  \* Node pgrex02:   \* master-pgsql : -INFINITY   \* pgsql-data-status : DISCONNECT   \* pgsql-status : HS:alone   \* ping-status : 1 |

属性情報表示部で、pgrex02のpgsql-data-statusがDISCONNECT、pgsql-statusがHS:aloneにそれぞれ変更されていることを確認します。

次に、pgrex01のPostgreSQLのサーバログファイル[[27]](#footnote-27) からwalsenderプロセスの停止を示すログ(terminating walsender)を検索します。

該当のログが出力されていた場合、D-LAN故障と特定できます。

D-LAN故障と特定された場合は、『D-LAN故障』の節へ進みます。

|  |
| --- |
| # grep 'terminating walsender' /var/log/pg\_log/postgresql-2020-04-17.log 2020-04-17 13:13:41 JST 13673 5ea65b91.3569-2 0 (repuser, [unknown], 192.168.2.2(50338), pgrex01) LOG:  terminating walsender process due to replication timeout  ※ PostgreSQLのサーバログファイルのファイル名は適宜読み替えてください。 |

【ハードウェア故障・リソース故障の特定】

pgrex02のpcs statusの実行結果の故障回数表示部から、故障回数(fail-count)の表示の有無によりハードウェア故障(ネットワーク等)またはリソース故障を判断します。

故障回数表示部に故障回数が表示されている場合、リソース故障と判断して『制御エラー情報表示部の確認』の項へ進みます。

故障回数が表示されていない場合、ハードウェア故障と判断して『属性情報表示部の確認』の項へ進みます。

### 属性情報表示部の確認

属性情報表示部の確認では、pgrex02でpcs statusコマンドを実行し、異常値を示している属性名を調べ、故障を特定します。

属性情報表示部の出力と対応する故障内容を以下の表に示します。故障が特定できた場合は、特定された故障内容の節へ進みます。

属性情報別故障箇所対応表

| 属性名 | 属性値(pgrex01) | 属性値(pgrex02) | 故障発生ノードまたは故障箇所 |
| --- | --- | --- | --- |
| ping-status | 0 | 1 | S-LAN故障(pgrex01) |
| 同上 | 1 | 0 | S-LAN故障(pgrex02) |
| 同上 | 0 | 0 | ルータ故障(pgrex01, pgrex02) |
| pgsql-data-status | LATEST | DISCONNECT | D-LAN故障 |
| pgsql-status | PRI | HS:alone | D-LAN故障 |

以下にpcs statusコマンド実行時の表示例を示します。この例では、pgrex01のNode Attributesに表示されているping-statusの属性値が異常を示していることから、pgrex01にてS-LANの故障を検知したことを示しています。

|  |
| --- |
| ： (略) Node Attributes:  \* Node pgrex01:   \* master-pgsql : -INFINITY   \* pgsql-data-status : DISCONNECT   \* pgsql-status : STOP   \* ping-status : 0 : Connectivity is lost  \* Node pgrex02:   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 00000139210029D8   \* pgsql-status : PRI   \* ping-status : 1 ：（略） |

### corosync-cfgtool -sによるIC-LAN状態の確認

corosync-cfgtool -sによるIC-LAN状態の確認では、Pacemakerが起動しているノードでcorosync-cfgtoolコマンドを実行してIC-LAN通信状態を確認し、「/var/log/messagesの確認」の節へ進みます。

以下にIC-LAN故障発生時のcorosync-cfgtoolコマンド実行時の表示例を示します。この例では、nodeid 1がdisconnectedを示していることから、pgrex01にてIC-LANの故障を検知したことを示しています。

|  |
| --- |
| ： (略) LINK ID 0 addr = 192.168.1.1 status:  nodeid 1: disconnected  nodeid 2: localhost LINK ID 1 addr = 192.168.3.1 status:  nodeid 1: disconnected  nodeid 2: localhost |

### 制御エラー情報表示部の確認

制御エラー情報表示部の確認では、pgrex02でpcs statusコマンドを実行し、リソースに発生した故障を特定します。故障を特定できた場合は、特定された故障の節へ進みます。

【monitor故障の特定】

以下にmonitor故障発生時のpcs statusコマンド実行時の表示例を示します。

|  |
| --- |
| # pcs status --full :（略） Failed Resource Actions:  \* pgsql\_monitor\_9000 on pgrex01 'not running' (7): call=79, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【demote故障の特定】

以下にdemote故障発生時のpcs statusコマンド実行時の表示例を示します。

|  |
| --- |
| # pcs status --full :（略） Failed Resource Actions:  \* pgsql\_demote\_0 on pgrex01 'unknown error' (1): call=88, status=Timed Out, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【stop故障の特定】

以下にstop故障発生時のpcs statusコマンド実行時の表示例を示します。

|  |
| --- |
| # pcs status --full :（略） Failed Resource Actions:  \* pgsql\_stop\_0 on pgrex01 'unknown error' (1): call=87, status=Timed Out, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【ipaddr-primary故障の特定】

以下にipaddr-primary故障発生時のpcs statusコマンド実行時の表示例を示します。

|  |
| --- |
| # pcs status --full :（略） Failed Resource Actions:  \* ipaddr-primary\_monitor\_10000 on pgrex01 'unknown error' (1): call=22, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【ipaddr-standby故障の特定】

以下にipaddr-standby故障発生時のpcs statusコマンド実行時の表示例を示します。

|  |
| --- |
| # pcs status --full :（略） Failed Resource Actions:  \* ipaddr-standby\_monitor\_10000 on pgrex02 'unknown error' (1): call=16, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【ipaddr-replication故障の特定】

以下にipaddr-replication故障発生時のpcs statusコマンド実行時の表示例を示します。

ipaddr-replicationの故障(monitor)の場合、リソースを再起動することで故障からの自動回復をする設定になっています。この時、故障回数表示部に故障回数だけが表示されます。

以下の例では、"ipaddr-replication"が1回故障したことを示しています。

|  |
| --- |
| # pcs status --full :（略） Migration Summary:  \* Node pgrex01:   \* ipaddr-replication: migration-threshold=0 fail-count=1 last-failure='日時表示'  \* Node pgrex02:  Failed Resource Actions:  \* ipaddr-replication\_monitor\_10000 on pgrex01 'unknown error' (1): call=77, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

以下の表示例のようにリソース故障(monitor)とリソース故障(stop)が発生している場合、特定される故障内容はリソース故障(stop)になります。

|  |
| --- |
| ：（略） Failed Resource Actions:  \* ipaddr-primary\_monitor\_10000 on pgrex01 'unknown error' (1): call=22, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms  \* pgsql\_stop\_0 on pgrex01 'unknown error' (1): call=87, status=Timed Out, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

### /var/log/messagesの確認

両方のノードでPacemakerが稼働している場合は両ノードのログファイル(/var/log/messages)から、Pacemakerが停止しているノードがある場合はPacemakerが停止しているノードのログファイルから、以下に示す2つのエラーメッセージを検索し、3種の故障を特定します。故障特定後は、特定された故障の節へ進みます。

【リソース故障(stop)の確認】

以下のコマンドを実行します。コマンドを実行した結果、エラーメッセージが出力された場合、PostgreSQLリソース(pgsql)のstop故障が発生したことを示しています。

|  |
| --- |
| # grep -e '.\*pacemaker-controld.\*notice:.\*Result of stop operation for.\*: error.\*' -e '.\*pacemaker-controld.\*error:.\*Result of stop operation for.\*: Timed Out.\*' /var/log/messages Jan 20 15:22:47 pgrex01 <daemon.err> pacemaker-controld[11207]: error: Result of stop operation for pgsql on pgrex01: Timed Out |

【IC-LAN故障の確認】

以下のコマンドを実行します。コマンドを実行した結果、エラーメッセージが出力された場合、IC-LAN故障が発生したことを示しています。

|  |
| --- |
| # grep '.\*pacemaker-controld.\*notice:.\*is now lost.\*' /var/log/messages 　Jan  9 14:24:58 pgrex01 <daemon.notice> pacemaker-controld[20520]: notice: Node pgrex02 state is now lost |

【ノード故障の確認】

リソース故障(stop)およびIC-LAN故障の確認でエラーメッセージが検索されなかった場合、ノード故障 (Pacemaker故障またはOS故障) が発生したと判断します。

## ルータ故障

この節では、ルータ故障時の対処について説明します。

### 故障時のHAクラスタ状態

S-LANのルータ故障を検知し、両ノードのPostgreSQLリソース(pgsql)、IPaddr2リソース(ipaddr-primary、ipaddr-standby、ipaddr-replication)が停止した状態となっています。

データベースサービスは停止しています。

### 復旧

pgrex01とpgrex02でPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01とpgrex02のPacemakerを再起動し、復旧します。

復旧後のHAクラスタ状態は、pgrex02が先に停止した場合はpgrex01(Primary)-pgrex02(Standby)、pgrex01が先に停止した場合はpgrex01(Standby)-pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex01]】

pcs statusの結果を確認し、データが進んでいるノードを特定します。

pgsql-data-statusがLATESTとなっている場合、そのノードのデータが進んでいることを示しています。

以降、pgrex01が「データが進んでいるノード」の場合の復旧手順を示します。pgrex02が「データが進んでいるノード」の場合はpgrex01とpgrex02を読み替えて下さい。

【STEP2：Pacemaker停止 [pgrex02]】

保守者の作業中に、PostgreSQLリソースが再起動しないようpgrex02のPacemakerを停止します。停止する手順は『PostgreSQL停止中のノードの停止』を参照してください。

【STEP3：Pacemaker停止 [pgrex01]】

保守者の作業中に、PostgreSQLリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『PostgreSQL停止中のノードの停止』を参照してください。

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(データベースサービス停止中)
* 報告時点でのHAクラスタ状態(2つのノードでPacemaker停止中)
* 故障箇所(ルータ故障が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。起動する手順は『Primaryの起動』を参照してください。

【STEP7：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1)]  \* OFFLINE: [ pgrex02 (2)] ：（略）   \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Stopped  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

【STEP8：Pacemaker起動 [pgrex02]】

pgrex02のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

【STEP9：ノード状態・リソース状態確認 [pgrex02]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pgrex02 ：（略） |

## S-LAN故障

この節では、S-LAN故障時の対処について説明します。

### 故障時のHAクラスタ状態

S-LAN故障を検知し、pgrex01でPostgreSQLリソース(pgsql)が停止した状態となっています。

その結果フェイルオーバが発生し、pgrex02のPostgreSQLリソースがPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。

データベースサービスは片方のノード(pgrex02)で継続しています。

### 復旧

データベースサービス状況の確認とpgrex01のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01のPacemakerを再起動します。

復旧後のHAクラスタ状態は、pgrex01(Standby) - pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex02]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略）   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02 ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

保守者の作業中に、PostgreSQLリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『PostgreSQL停止中のノードの停止』を参照してください。

【STEP3：ノード状態確認 [pgrex02]】

ノード状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex02でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(S-LAN故障が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。

pgrex02でPostgreSQLリソースがPrimaryとして稼働中のため、pgrex01をStandbyとして起動します。

起動する手順は『Standbyの起動』を参照してください。

【STEP7：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

## D-LAN故障

この節では、D-LAN故障時の対処について説明します。

### 故障時のHAクラスタ状態

D-LAN故障を検知し、pgrex02を切り離した状態となっています。

その結果、pgrex02のIPaddr2リソース(ipaddr-standby)がpgrex01へ移動します。

データベースサービスは片方のノード(pgrex01)で継続しています。

### 復旧

データベースサービス状況の確認とpgrex02のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex02のPacemakerを再起動し、復旧します。

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex01]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01 ：（略） |

【STEP2：Pacemaker停止 [pgrex02]】

pgrex02のPacemakerを停止します。停止する手順は『Standbyの停止』を参照してください。

【STEP3：ノード状態確認 [pgrex01]】

ノード状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1)]  \* OFFLINE: [ pgrex02 (2)] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex01でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex02でPacemaker停止中)
* 故障箇所(D-LAN故障が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：Pacemaker起動 [pgrex02]】

pgrex02のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

【STEP7：ノード状態・リソース状態確認 [pgrex02]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pgrex02 ：（略） |

## リソース故障(monitor)

この節では、リソース故障(monitor)時の対処について説明します。

### 故障時のHAクラスタ状態

リソース故障(monitor)を検知し、pgrex01でPostgreSQLリソース(pgsql)が停止した状態となっています。

その結果フェイルオーバが発生し、pgrex02のPostgreSQLリソースがPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。

データベースサービスは片方のノード(pgrex02)で継続しています。

### 復旧

データベースサービス状況の確認とpgrex01のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01のPacemakerを再起動し、復旧します。

復旧後のHAクラスタ状態は、pgrex01(Standby) - pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex02]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

pgrex01のPacemakerを停止します。停止する手順は『PostgreSQL停止中のノードの停止』を参照してください。

【STEP3：ノード状態確認 [pgrex02]】

ノード状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex02でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(pgrex01のリソース故障(monitor)が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。

pgrex02でPostgreSQLリソースがPrimaryとして稼働中のため、pgrex01をStandbyとして起動します。

起動する手順は『Standbyの起動』を参照してください。

【STEP7：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

## リソース故障(demoteおよびstop)

この節では、リソース故障(demoteおよびstop)時の対処について説明します。

### 故障時のHAクラスタ状態

リソース故障(demoteおよびstop)を検知し、pgrex02からpgrex01へのreset処理を実施しています。 reset処理の実行状況により、HAクラスタおよびデータベースサービスの状態は以下のようになります。

1. reset処理が成功した場合

* フェイルオーバが発生し、pgrex02のPostgreSQLリソース(pgsql)がPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。
* データベースサービスは片方のノード(pgrex02)で継続しています。

1. reset処理が失敗した場合

* pgrex01の状態は不定となり、pgrex02はStandbyのままで、pgrex01に対してreset処理を繰り返しています。
* データベースサービスの稼働状況は不定です。

### 復旧

データベースサービス状況の確認とpgrex01のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01のPacemakerを再起動し、復旧します。

復旧後のHAクラスタ状態は、pgrex01(Standby) - pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：強制電源断 [pgrex01]】

この作業は、pgrex01のノードの状態がOFFLINEとなって**いない**場合に実施します。

pgrex01の電源ボタンを押下し、電源を停止します。

【STEP2：ノード状態・リソース状態確認 [pgrex02]】

ノード状態とリソース状態が以下のとおりとなっていることを確認し、【STEP4】へ進みます。 pgrex01のノード状態がOFFLINEとなっていない場合は、【STEP3】へ進みます。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） |

【STEP3：保守者介在処理 [pgrex02]】

手動でpgrex01を停止させたことをHAクラスタに通知するために、stonith\_adminコマンドによる保守者介在処理を行います。

pgrex02で"pacemaker-controld.\*notice:.\*was not terminated.\*"をキーワードとし、/var/log/messagesに以下のログが出力されていることを確認します。

|  |
| --- |
| Jan 23 15:49:27 pgrex02 pacemaker-controld[1531920]: notice: Peer pgrex01 was not terminated (reboot) by pgrex02 on behalf of pacemaker-controld.1531920: エラー内容 |

pgrex02でstonith\_adminコマンドを以下のとおり実施します。

実施後、再度ノード状態を確認するため、【STEP2】へ戻ります。

|  |
| --- |
| # stonith\_admin -C pgrex01 |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex02でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(pgrex01のリソース故障(demoteおよびstop)が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：ノード起動 [pgrex01]】

この作業は、【STEP1】で、電源を停止した場合のみ実施します。

pgrex01の電源ボタンを押下し、ノードを起動します。

【STEP7：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

※ pgrex02でPostgreSQLリソースがPrimaryとして稼働中のため、pgrex01をStandbyとして起動します。

【STEP8：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

## リソース故障(ipaddr-primary)

この節では、リソース故障(ipaddr-primary)時の対処について説明します。

### 故障時のHAクラスタ状態

リソース故障(ipaddr-primary)を検知し、pgrex01でPostgreSQLリソース(pgsql)が停止された状態となっています。 その結果フェイルオーバが発生し、pgrex02のPostgreSQLリソースがPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。

データベースサービスは、片方のノード(pgrex02)で継続しています。

### 復旧

データベースサービス状況の確認とpgrex01のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01のPacemakerを再起動し、復旧します。

復旧後のHAクラスタ状態は、pgrex01(Standby) - pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex02]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略）   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02 ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

保守者の作業中に、PostgreSQLリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『PostgreSQL停止中のノードの停止』を参照してください。

【STEP3：ノード状態確認 [pgrex02]】

ノード状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex02でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(IPaddr2(ipaddr-primary)故障が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

※ pgrex02でPostgreSQLリソースがPrimaryとして稼働中のため、pgrex01のPostgreSQLリソースをStandbyとして起動します。

【STEP7：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

## リソース故障(ipaddr-replication)

この節では、リソース故障(ipaddr-replication)時の対処について説明します。

### 故障時のHAクラスタ状態

リソース故障(ipaddr-replication)を検知し、pgrex01のIPaddr2リソース(ipaddr-replication)が再起動した状態となっています。

データベースサービスは両ノードで継続しています。

### 復旧

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex01]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） Migration Summary:  \* Node pgrex01 (1):   \* (故障回数表示)  Failed Resource Actions:  \* (ipaddr-replication制御エラー情報表示) ：（略） |

【STEP2：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex01でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(両ノードでPacemaker起動中)
* 故障箇所(IPaddr2(ipaddr-replication)故障が発生)

【STEP3：保守者による故障復旧】

復旧の必要がある場合、保守者が故障復旧を実施します。

【STEP4：リソース(ipaddr-replication)のフェイルカウントクリア [pgrex01]】

pgrex01のipaddr-replicationリソースのフェイルカウントをクリアします。

|  |
| --- |
| # pcs resource cleanup ipaddr-replication node=pgrex01 Cleaned up ipaddr-primary on pgrex01 Cleaned up ipaddr-replication on pgrex01 Waiting for 1 reply from the controller. OK |

【STEP5：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） Migration Summary:   (表示無し) ：（略） |

## リソース故障(ipaddr-standby)

この節では、リソース故障(ipaddr-standby)時の対処について説明します。

### 故障時のHAクラスタ状態

リソース故障(ipaddr-standby)を検出し、pgrex02のIPaddr2リソース(ipaddr-standby)がpgrex01へ移動した状態となっています。

データベースサービスは両ノードで継続しています。

### 復旧

復旧手順を以下に説明します。

【STEP1：リソース状態確認 [pgrex01]】

リソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01 ：（略） Migration Summary:  \* Node pgrex02 (2):   \* (故障回数表示)  Failed Resource Actions:  \* (ipaddr-standby制御エラー情報表示) ：（略） |

【STEP2：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex01でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(両ノードでPacemaker起動中)
* 故障箇所(IPaddr2(ipaddr-standby)故障が発生)

【STEP3：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP4：リソース(ipaddr-standby)の切り替え [pgrex02]】

pgrex02のipaddr-standbyリソースのフェイルカウントをクリアします。クリアすると自動的に切り替えが行なわれます。

|  |
| --- |
| # pcs resource cleanup ipaddr-standby node=pgrex02 Cleaned up ipaddr-standby on pgrex02 Waiting for 1 reply from the controller. OK |

【STEP5：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） Migration Summary:   (表示無し) ：（略） |

## ノード故障

この節では、ノード故障時の対処について説明します。

### 故障時のHAクラスタ状態

OS故障、電源断、またはPacemakerのプロセス故障により、pgrex02がpgrex01の異常を検知し、pgrex01へreset処理を実施しています。 reset処理の実行状況により、HAクラスタおよびデータベースサービスの状態は以下のようになります。

1. reset処理が成功した場合

* フェイルオーバが発生し、pgrex02のPostgreSQLリソース(pgsql)がPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。
* データベースサービスは片方のノード(pgrex02)で継続しています。

1. reset処理が失敗した場合

* pgrex02はStandby状態のままで、pgrex01に対してreset処理を繰り返しています。
* データベースサービス稼動状況は不定です。

### 復旧

データベースサービス状況の確認とpgrex01のPacemakerを停止し、保守者による復旧を依頼します。

保守者による故障復旧後、pgrex01のPacemakerを再起動し、復旧します。

復旧後のHAクラスタ状態は、pgrex01(Standby) - pgrex02(Primary)となります。

復旧手順を以下に説明します。

【STEP1：強制電源断 [pgrex01]】

この作業は、pgrex01のノードの状態がOFFLINEとなって**いない**場合に実施します。

pgrex01の電源ボタンを押下し、電源を停止します。

【STEP2：ノード状態・リソース状態確認 [pgrex02]】

ノード状態とリソース状態が以下のとおりとなっていることを確認し、【STEP4】へ進みます。

pgrex01のノード状態がOFFLINEとなっていない場合は、【STEP3】へ進みます。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02 ：（略） |

【STEP3：保守者介在処理 [pgrex02]】

手動でpgrex01を停止させたことをHAクラスタに通知するために、stonith\_adminコマンド による保守者介在処理を行います。

pgrex02で"pacemaker-controld.\*notice:.\*was not terminated.\*"をキーワードとし、/var/log/messagesに以下のログが出力されていることを確認します。

|  |
| --- |
| Jan 23 15:49:27 pgrex02 pacemaker-controld[1531920]: notice: Peer pgrex01 was not terminated (reboot) by pgrex02 on behalf of pacemaker-controld.1531920: エラー内容 |

pgrex02でstonith\_adminコマンドを以下のとおり実施します。

実施後、再度ノード状態を確認するため、【STEP2】へ戻ります。

|  |
| --- |
| # stonith\_admin -C pgrex01 |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex02でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(pgrex01のノードまたはPacemaker故障が発生)

【STEP5：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP6：ノード起動 [pgrex01]】

この作業は、【STEP1】で、電源を停止した場合のみ実施します。

pgrex01の電源ボタンを押下し、ノードを起動します。

【STEP7：Pacemaker起動 [pgrex01]】

pgrex01のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

※ pgrex02でPostgreSQLリソースがPrimaryとして稼働中のため、pgrex01をStandbyとして起動します。

【STEP8：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02 ：（略）   \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pgrex01 ：（略） |

## IC-LAN故障

この節では、IC-LAN故障時の対処について説明します。

### 故障時のHAクラスタ状態

IC-LAN故障を検知し、reset処理を実行しています。 reset処理の実行状況により、HAクラスタおよびデータベースサービスの状態は以下のようになります。

1. pgrex01からpgrex02へのreset処理が成功した場合

* pgrex02がreset処理によりPacemakerを停止し、IPaddr2リソース(ipaddr-standby)がpgrex01へ移動します。
* データベースサービスは片方のノード(pgrex01)で継続しています。

1. pgrex02からpgrex01へのreset処理が成功した場合

* pgrex01がreset処理によりPacemakerを停止し、pgrex02のPostgreSQLリソース(pgsql)がPrimaryへ昇格し、pgrex01のIPaddr2リソース(ipaddr-primary、ipaddr-replication)がpgrex02へ移動します。
* データベースサービスは片方のノード(pgrex02)で継続しています。

1. 両方のノードからのreset処理が失敗した場合

* pgrex01はPrimaryのままで、pgrex02はStandbyのままでそれぞれ相手ノードに対してreset処理を繰り返しています。
* データベースサービスは両ノードで継続しています。

### 復旧

pgrex01がPrimaryの場合の復旧手順を以下に説明します。

pgrex02がPrimaryの場合は、pgrex01とpgrex02を読み替えて下さい。

【STEP1：強制電源断 [pgrex02]】

この作業は、pgrex02のノードの状態がOFFLINEとなって**いない**場合に実施します。

pgrex02の電源ボタンを押下し、電源を停止します。

【STEP2：ノード状態・リソース状態確認 [pgrex01]】

ノード状態とリソース状態が以下のとおりとなっていることを確認し、【STEP4】へ進みます。

pgrex02のノード状態がOFFLINEとなっていない場合は、【STEP3】へ進みます。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1)]  \* OFFLINE: [ pgrex02 (2)] ：（略）   \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01 ：（略） |

【STEP3：保守者介在処理 [pgrex01]】

手動でpgrex02を停止させたことをHAクラスタに通知するために、stonith\_adminコマンドによる保守者介在処理を行います。

pgrex01で"pacemaker-controld.\*notice:.\*was not terminated.\*"をキーワードとし、/var/log/messagesに以下のログが出力されていることを確認します。

|  |
| --- |
| Jan 23 15:49:27 pgrex01 pacemaker-controld[1531920]: notice: Peer pgrex02 was not terminated (reboot) by pgrex01 on behalf of pacemaker-controld.1531920: エラー内容 |

pgrex01でstonith\_adminコマンドを以下のとおり実施します。

実施後、再度ノード状態を確認するため、【STEP2】へ戻ります。

|  |
| --- |
| # stonith\_admin -C pgrex02 |

【STEP4：ノード起動 [pgrex02]】

この作業は、【STEP1】で、電源を停止した場合のみ実施します。

pgrex02の電源ボタンを押下し、ノードを起動します。

【STEP5：保守者へ報告】

以下の内容を報告します。

* 報告時点でのデータベースサービス稼働状況(pgrex01でデータベースサービス継続中)
* 報告時点でのHAクラスタ状態(pgrex02でPacemaker停止中)
* 故障箇所(IC-LAN故障が発生)

【STEP6：保守者による故障復旧】

保守者が故障復旧を実施します。

【STEP7：Pacemaker起動 [pgrex02]】

pgrex02のPacemakerを起動します。起動する手順は『Standbyの起動』を参照してください。

【STEP8：ノード状態・リソース状態確認 [pgrex02]】

ノードとリソース状態が以下のとおりとなっていることを確認します。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日付表示 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pgrex02 ：（略） |

# コマンド直接実行の運用

本章では、コマンドを直接実行して運用する手順を説明します。PG-REX運用補助ツールを用いて運用する場合は、『起動と停止』の章および『メンテナンス時の対応』の章を参照してください。

## 起動と停止

本節では、コマンドを直接実行して起動と停止を行う手順を説明します。

### PG-REXの起動

PG-REXの起動は、一方のノードでPrimaryを起動させ、起動完了後、もう一方のノードでStandbyを起動させます。どちらのノードをPrimary、Standbyとして稼働させるかは、ユーザが決定します。

PrimaryおよびStandbyの起動手順については、以降の各項を参照してください。

### Primaryの起動

本節では、Primaryの起動手順を説明します。

1. どのノードをPrimaryとして起動するか決定します。

* PG-REXでは、最新のDBデータを持つノードをPrimaryとして起動しなければなりません。古いDBデータを持つノードをPrimaryとして起動すると、その古い分だけDBデータは失われてしまいます。
* 以下は、Primaryとして起動するノードを決めるときの考え方の例です。
  + DBクラスタが片方のノードのみに存在し、そのDBクラスタを使ってPG-REXを起動する場合(初めてPrimaryを起動する場合を含む)は、DBクラスタが存在するノードをPrimaryとして起動する。
  + DBクラスタが両方のノードに存在する場合は、直前までPrimaryとして稼働していたノードをPrimaryとして起動する。
  + 既存のDBクラスタを使わず(もしくは既存のDBクラスタが壊れている)、以前に取得したベースバックアップからPG-REXを起動する場合は、そのベースバックアップを展開したノードをPrimaryとして起動する。

以降の手順では、pgrex01をPrimaryとして起動します。

1. pgrex01およびpgrex02で、Pacemakerが停止していることを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs status --full Error: error running crm\_mon, is pacemaker running?  Error: cluster is not available on this node |

1. ベースバックアップからPrimaryを起動する場合に限り、pgrex01でPostgreSQL単体のアーカイブリカバリを行います。

* アーカイブリカバリの手順については、『PostgreSQLドキュメント』を参照してください。アーカイブリカバリが完了したら、PostgreSQLを停止します。
* 本作業はpostgresユーザで行います。

|  |
| --- |
| $ pg\_ctl start サーバの起動完了を待っています.... ：(略) 完了 サーバ起動完了 $ 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 |

【注意】

PG-REXでは、アーカイブリカバリをさせながらPrimaryを起動することを推奨しません。これは、アーカイブリカバリにより起動に時間がかかり、Pacemakerによって起動失敗とみなされてしまう可能性があるからです。そのため、Primaryでアーカイブリカバリを行う場合は、Pacemaker経由ではなく、まずはPostgreSQL単体で起動させるようにしてください。アーカイブリカバリの完了後、PostgreSQLを停止させた上で、Primaryの起動の手順を行います。これにより、Primary起動時のアーカイブリカバリは必要なくなるため、Primaryの起動に時間がかかることはありません。

また、PG-REXでリカバリを行う場合は、必ず最新の状態となります。

アーカイブリカバリを行う際にユーザが追加した設定は、アーカイブリカバリ終了後にすべて削除してください。

1. pgrex01で起動禁止フラグのファイルが存在する場合は削除します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # rm /var/lib/pgsql/tmp/PGSQL.lock |

1. 初回起動時、もしくは新しいリソース定義xmlファイルを反映させる場合、HAクラスタを作成します。すでにHAクラスタが存在する場合は、既存のHAクラスタを削除します。

* HAクラスタの作成には、HAクラスタ名、それぞれのノードのホスト名とインターコネクトLANのIPアドレス2つのセットを2組指定します。
* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs cluster destroy --all Warning: Unable to load CIB to get guest and remote nodes from it, those nodes will not be deconfigured. pgrex01: Stopping Cluster (pacemaker)… pgrex02: Stopping Cluster (pacemaker)… pgrex01: Successfully destroyed cluster pgrex02: Successfully destroyed cluster  # pcs cluster setup pgrex\_cluster pgrex01 addr=192.168.1.1 addr=192.168.3.1 pgrex02 addr=192.168.1.2 addr=192.168.3.2 Destroying cluster on hosts: ‘pgrex01’, ‘pgrex02’… pgrex02: Successfully destroyed cluster pgrex01: Successfully destroyed cluster Requesting remove ‘pcsd settings’ from ‘pgrex01’, ‘pgrex02’ pgrex01: successful removal of the file ‘pcsd settings’ pgrex02: successful removal of the file ‘pcsd settings’ Sending ‘corosync authkey’, ‘pacemaker authkey’ to ‘pgrex01’, ‘pgrex02’ pgrex02: successful distribution of the file ‘corosync authkey’ pgrex02: successful distribution of the file ‘pacemaker authkey’ pgrex01: successful distribution of the file ‘corosync authkey’ pgrex01: successful distribution of the file ‘pacemaker authkey’ Sending ‘corosync.conf’ to ‘pgrex01’, ‘pgrex02’ pgrex01: successful distribution of the file ‘corosync.conf’ pgrex02: successful distribution of the file ‘corosync.conf’ Cluster has been successfully set up. |

1. pgrex01でPrimaryを起動します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs cluster start Starting Cluster… |

1. pgrex01のPacemakerが起動したことを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示  \* Last change:  日時表示 by hacluster via crmd on pgrex01 ：（略） Node List:  \* Online: [ pgrex01 (1)] ：（略） |

Onlineになるには数十秒の時間を要する場合があります。

【注意】

以下のWARNINGS表示はSTONITH設定が行われていないことによるものですので、現段階では問題ありません。

|  |
| --- |
| WARNINGS: No stonith devices and stonith-enabled is not false |

1. 初回起動時、もしくは新しいリソース定義xmlファイルを反映させる場合、リソース定義xmlファイルを反映させます。

* pgrex01でリソース定義xmlファイルを反映させるコマンドを実行します。
* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs cluster cib-push PG-REX14\_pm\_pcsgen\_env.xml CIB updated |

1. quorumの解除をします。

* quorumを解除するコマンドを実行します。 本作業はrootユーザで行います。

|  |
| --- |
| # pcs quorum unblock --force Node: pgrex02 confirmed fenced Quorum unblocked Waiting for nodes canceled |

1. pgrex02のSTONITH履歴をクリアします。

|  |
| --- |
| # stonith\_admin -c -H pgrex02 |

1. pgrex01でpcs status --full コマンドを実行し、PrimaryのPacemakerが正常に起動したことを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示  \* Last change:  日時表示 by hacluster via crmd on pgrex01 ：（略） Node List:  \* Online: [ pgrex01 (1)] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01 ：（略）  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01 ：（略）  \* fence2-ipmilan   (stonith:fence\_ipmilan): Started pg-rex01 ：（略） Node Attributes:  \* Node pgrex01 (1):   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 0000000005000060   \* pgsql-status : PRI   \* ping-status : 1  ：（略） |

### Standbyの起動

本節では、Standbyの起動手順を説明します。以降の手順では、pgrex02をStandbyとして起動します。

1. pgrex01でPrimaryが稼働中であること、およびpgrex02でPacemakerが停止していることを確認します。

* pgrex02でPacemakerが稼働中の場合はPacemakerを停止します。
* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs status --full Error: error running crm\_mon, is pacemaker running?  Error: cluster is not available on this node |

1. 以下のいずれかの場合に限り、Primaryから新たにベースバックアップを取得します。
   * pgrex02が初回起動時の場合
   * pgrex02にDBクラスタ(または展開されたベースバックアップ)が存在しない場合
   * pgrex02に起動禁止フラグのファイル(/var/lib/pgsql/tmp/PGSQL.lock)が存在する場合
   * 起動禁止フラグのファイルが存在するということは、前回停止時(異常終了も含む)にpgrex02がPrimaryとして稼働していたことを意味します。このようなノードをStandbyとして起動する場合(例えば、異常終了した旧Primaryのノードを、フェイルオーバ後に再組み込みする場合)は、Primaryからベースバックアップを取得する必要があります。一方、例えば、Standbyとして稼働していたノードを、停止後に再び起動する場合は、起動禁止フラグのファイルが存在せず、ベースバックアップの取得は不要となります。
   * pgrex02に存在するDBクラスタが非常に古い場合
   * Standbyを長期間停止していたなどで、pgrex02のDBクラスタがpgrex01のDBクラスタと比べて非常に古い場合。この場合、pgrex02のDBクラスタをそのまま使用するとリカバリが大量に必要になります。
   * pgrex01で稼働中のPrimaryにレプリケーション接続ができない場合
   * たとえば、システムIDが異なる、タイムライン履歴が一致しないなどの場合。

* Primaryからバックアップを取得しなければならない場合は、『PostgreSQLドキュメント』を参考に、pgrex01からバックアップを取得し、pgrex02の$PGDATAに展開します。バックアップの方法は、システムの要件に応じて決める必要があります。例えば、pgrex02において、postgresユーザで以下の操作を行い、バックアップを取得します。pg\_basebackupコマンドの詳細は、『PostgreSQLドキュメント』を参照してください。
* 本作業はpostgresユーザで行います。

|  |
| --- |
| $ rm -rf $PGDATA $ rm -rf /dbfp/pgwal/pg\_wal $ pg\_basebackup -h 192.168.2.3 -U repuser -D $PGDATA -X none -P 70762/70762 kB (100%), 1/1 テーブル空間 NOTICE: all required WAL segments have been archived  ※ -hにはレプリケーション受付用の仮想IPアドレス   -Uにはレプリケーションユーザを指定 |

1. pgrex01からpgrex02へアーカイブディレクトリを同期します。

* 本作業はpostgresユーザで行います。
* アーカイブディレクトリの同期は必ずPrimaryからStandbyに行ってください。誤ってStandbyからPrimaryに同期すると、DBデータの一部が失われる可能性があります。

|  |
| --- |
| $ rsync -av --size-only 192.168.2.3:/dbfp/pgarch/arc1/ /dbfp/pgarch/arc1/ postgres@192.168.2.3’s password: receiving incremental file list ：（略） sent 1,434 bytes received 889,422,925 bytes 61,339,610.97 bytes/sec total size is 889,200,754 speedup is 1.00  ※ 接続先はレプリケーション受付用の仮想IPアドレス |

1. pgrex02で起動禁止フラグのファイルが存在する場合は削除します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # rm /var/lib/pgsql/tmp/PGSQL.lock |

1. pgrex02でStandbyを起動します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs cluster start Starting Cluster… |

1. pgrex02でpcs status --full コマンドを実行し、StandbyのPacemakerが正常に起動したことを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # pcs status --full ：（略）  \* Last updated: 日時表示  \* Last change:  日時表示 by hacluster via crmd on pgrex01 ：（略） Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex01   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex01  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02  \* Clone Set: ping-clone [ping]:   \* ping   (ocf::pacemaker:ping):  Started pgrex01   \* ping   (ocf::pacemaker:ping):  Started pgrex02  \* fence1-ipmilan   (stonith:fence\_ipmilan): Started pg-rex02 ：（略）  Node Attributes:  \* Node pgrex01 (1):   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 0000000005000060   \* pgsql-status : PRI   \* ping-status : 1  \* Node pgrex02 (2):   \* master-pgsql : 100   \* pgsql-data-status : STREAMING|SYNC   \* pgsql-status : HS:sync   \* ping-status : 1  ：（略） |

1. corosync-cfgtoolコマンドで、各インターコネクトLANの状態が「localhost」、「connected」であることを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # corosync-cfgtool -s Printing link status. Local node ID 1 LINK ID 0  addr = 192.168.1.10  status:  node 0:localhost  node 1:connected LINK ID 1  addr = 192.168.3.10  status:  node 0:localhost  node 1:connected |

### PG-REXの停止

PG-REXを停止するには、Standbyを停止させ、Standbyの停止完了後にPrimaryを停止させます。

Primaryから停止した場合、フェイルオーバが発生しますので、ご注意ください。

PrimaryおよびStandbyの停止手順については、以降の項を参照してください。

この手順でPG-REXを停止させた場合、次にPG-REXを起動するときには、Primaryからのベースバックアップの取得は必要ありません。

### Standbyの停止

本節では、Standbyの停止手順を説明します。

本作業は停止対象のノードにて、rootユーザで行います。

1. Standbyを停止します。



|  |
| --- |
| # pcs cluster stop Stopping Cluster (pacemaker)… Stopping Cluster (corosync)… |

1. StandbyのPacemakerが正常に停止されたことを確認します。



|  |
| --- |
| # pcs status --full Error: error running crm\_mon, is pacemaker running?  Error: cluster is not available on this node |

1. StandbyのPostgreSQLが正常に停止されたことを確認します。



|  |
| --- |
| # ps -ef | grep postgres root      3252 17408  0 17:18 pts/0    00:00:00 grep postgres  ※ PostgreSQLのプロセスが存在しないことを確認します。 |

### Primaryの停止

本節では、Primaryの停止手順を説明します。

Standby稼働中にPrimaryを停止した場合、フェイルオーバが発生することに注意してください。

本作業は停止対象のノードにて、rootユーザで行います。

1. Primaryを停止します。



|  |
| --- |
| # pcs cluster stop --force Stopping Cluster (pacemaker)… Stopping Cluster (corosync)… |

1. PrimaryのPacemakerが正常に停止されたことを確認します。



|  |
| --- |
| # pcs status --full Error: error running crm\_mon, is pacemaker running?  Error: cluster is not available on this node |

1. PrimaryのPostgreSQLが正常に停止されたことを確認します。



|  |
| --- |
| # ps -ef | grep postgres root     20518 24450  0 17:28 pts/0    00:00:00 grep postgres  ※ PostgreSQLのプロセスが存在しないことを確認します。 |

## アーカイブログの削除

本節では、コマンドを直接実行してデータベースの復旧に必要のないファイルをアーカイブディレクトリから削除する手順について記述します。

本節の作業はpostgresユーザで行います。

### PostgreSQLアーカイブログの削除

運用を続けていくにしたがってアーカイブファイルの容量は増えていくため、継続的な運用のためには適宜削除を行う必要があります。

PostgreSQLのアーカイブログ、バックアップ履歴ファイル[[28]](#footnote-28)、タイムライン履歴ファイル[[29]](#footnote-29)は、以下の条件を全て満足する場合、アーカイブディレクトリから削除することができます。

* 管理している最古のバックアップ取得時点以前のファイル
* Standbyが既に反映・チェックポイント済みのファイル

定期的に以下の手順に従って不要なファイルを削除します。

1. 最古のバックアップに必要な最も古いWALファイル名を取得

* 最古のバックアップが存在するサーバで、最古のバックアップのbackup\_labelファイルに記述されたSTART WAL LOCATIONの括弧内のWALファイル名を取得します。
* ここでは管理する最古のバックアップの格納先を、$OLDEST\_BACKUPとします。

|  |
| --- |
| [~] $ cat $OLDEST\_BACKUP/pgdata/backup\_label START WAL LOCATION: 6A/4E0013D8 (file 000000730000006A0000004E) CHECKPOINT LOCATION: 6A/4E002740 BACKUP METHOD: streamed BACKUP FROM: master START TIME: 2018-12-10 14:26:19 JST LABEL: pg\_basebackup base backup START TIMELINE: 73 |

1. Standbyに必要な最も古いWALファイル名を取得

* pgrex02で、pg\_controldataを実行して、Standbyの"最終チェックポイントのREDO WALファイル"の値を取得します。

|  |
| --- |
| [pgrex02] $ pg\_controldata $PGDATA ：(略) 最終チェックポイントのREDO WALファイル:    000000750000006A00000051 ：(略) |

1. 削除対象の計算

* (1)のWALファイル名と(2)のWALファイル名を辞書順で比較します。その結果、小さいほうのWALファイル名を以降の手順で使用します。

1. データベースの復旧に必要のないファイルの削除

* 両ノードで、(3)で決定したWALファイル名よりも古いアーカイブファイルを削除します。
* 以下のコマンドはpgrex01を想定した例になります。以下の手順をpgrex01で実行後、必ずpgrex02でも同じ手順を実行してください。
* ls -lコマンドを用いてアーカイブディレクトリを辞書順にソートして出力します。このとき、(3)で決定したWALファイル名と同じ名前のファイルをチェックします。そのファイルより前に出力されたファイルを削除することができます。ただし、そのWALファイル名と先頭8桁が同じタイムライン履歴ファイル(\*\*\*\*\*\*\*\*.history)は削除しません。
* (例)
* (3)で「000000730000006A0000004E」と決定した場合、赤字が削除可能なファイルとなります。これらのファイルをrmコマンド等で削除してください。

|  |
| --- |
| [pgrex01] $ ls -l /dbfp/pgarch/arc1 ~ 000000710000006A0000004B ~ 000000710000006A0000004B.00000020.backup ~ 00000072.history ~ 000000720000006A0000004C ~ 000000720000006A0000004D ~ 000000720000006A0000004E ~ 000000720000006A0000004F ~ 00000073.history ~ 000000730000006A0000004C ~ 000000730000006A0000004D ~ 000000730000006A0000004E ~ 000000730000006A0000004E.000013D8.backup ~ 000000730000006A0000004F ~ 000000730000006A00000050 ~ 00000074.history ~ 000000740000006A00000050 ~ 000000740000006A00000051 ~ 00000075.history ~ 000000750000006A00000051 |

## 計画的なノード切り替え

本節では、手動による計画的なノード切り替えを実施する手順を記述します。

本節の作業は、rootユーザで行います。

### ノード切り替え

手動によるノード切り替えの手順を以下に示します。ノード切り替えの手順はフェイルオーバとリソースの再組み込みの手順で構成されています。

1. フェイルオーバ実行

* pgrex01のPacemakerを停止し、Primaryをpgrex02に切り替えます。

1. フェイルオーバ確認

* pgrex02でpcs statusコマンドを実行し、以下を確認します。
  + ノード情報表示で、pgrex01がOFFLINEとなっていること。
  + pgsql-cloneリソースが、pgrex02上でPrimaryとして動作していること。
  + ipaddr-primaryリソース、ipaddr-standbyリソースが、pgrex02上で起動していること。

|  |
| --- |
| [pgrex02] # pcs status --full ：（略）  \* Last updated: 日時表示  \* Last change:  日時表示 by hacluster via crmd on pgrex01 ：（略）  Node List:  \* Online: [ pgrex02 (2)]  \* OFFLINE: [ pgrex01 (1)] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex02 ：（略） |

1. pgrex01の再組み込み準備

* pgrex01の再組み込みの準備をします。再組み込みの準備には、pgrex02からベースバックアップの取得、pg\_walの構成の修正、pgrex02のアーカイブログとの同期、起動禁止フラグの削除があります。

|  |
| --- |
| [pgrex01] # su - postgres 最終ログイン: 2020/06/12 (金) 11:38:05 JST日時 pts/0 [pgrex01] $ rm -rf $PGDATA [pgrex01] $ rm -rf /dbfp/pgwal/pg\_wal [pgrex01] $ pg\_basebackup -h 192.168.2.3 -U repuser -D $PGDATA -X none -P 70762/70762 kB (100%), 1/1 テーブル空間 NOTICE: all required WAL segments have been archived  ※ -hにはレプリケーション受付用の仮想IPアドレス、   -Uにはレプリケーションユーザを指定  [pgrex01] $ rsync -av --size-only 192.168.2.3:/dbfp/pgarch/arc1/ /dbfp/pgarch/arc1/ postgres@192.168.2.3’s password: receiving incremental file list ：（略） sent 1,434 bytes received 889,422,925 bytes 61,339,610.97 bytes/sec total size is 889,200,754 speedup is 1.00  ※ 接続先はレプリケーション受付用の仮想IPアドレス  [pgrex01] $ exit ログアウト  [pgrex01] # rm /var/lib/pgsql/tmp/PGSQL.lock |

1. pgrex01の再組み込み

* pgrex01をStandbyとして起動します。

1. ノード切り替え確認

* pgrex01でpcs statusコマンドを実行し、以下を確認します。
  + ノード情報表示で、pgrex01がOnlineとなっていること。
  + pgsql-cloneリソースが、pgrex01上でStandbyとして動作していること。
  + ipaddr-replicationリソースが、pgrex01上で起動していること。

|  |
| --- |
| [pgrex01] # pcs status --full ：（略）  \* Last updated: 日時表示  \* Last change:  日時表示 by hacluster via crmd on pgrex01 ：（略）  Node List:  \* Online: [ pgrex01 (1) pgrex02 (2) ] ：（略）  \* Clone Set: pgsql-clone [pgsql] (promotable):   \* pgsql  (ocf::linuxhajp:pgsql): Slave pgrex01   \* pgsql  (ocf::linuxhajp:pgsql): Master pgrex02  \* Resource Group: primary-group:   \* ipaddr-primary     (ocf::heartbeat:IPaddr2):   Started pgrex02   \* ipaddr-replication (ocf::heartbeat:IPaddr2):   Started pgrex02  \* ipaddr-standby (ocf::heartbeat:IPaddr2):   Started pgrex01  \* Clone Set: ping-clone [ping]: ：（略） Node Attributes:  \* Node pgrex01 (1):   \* master-pgsql : 1000   \* pgsql-data-status : LATEST   \* pgsql-master-baseline : 0000000005000060   \* pgsql-status : PRI   \* ping-status : 1  \* Node pgrex02 (2):   \* master-pgsql : 100   \* pgsql-data-status : STREAMING|SYNC   \* pgsql-status : HS:sync   \* ping-status : 1 ：（略） |

# 用語集

本章では、資料に使用されている用語について以下に示します。

用語集

| 用語・略語 | 説明 |
| --- | --- |
| 管理用LAN | 監視者や保守者が運用管理やメンテナンスのために PG-REXの環境へアクセスして使用するLANのこと。 |
| 起動禁止フラグ | ノードの起動を防止するためのファイル。 起動前に状態の確認を要する場合に配置される。 (作成されるファイル : /var/lib/pgsql/tmp/PGSQL.lock) |
| 昇格 | ノードの状態をStandbyからPrimaryに遷移させること。 |
| スイッチオーバ | HAクラスタのノードの状態を切り替える(PrimaryをStandbyに、 StandbyをPrimaryに変更する)こと。 |
| ノード | PG-REXにおいてPacemakerが管理するサーバ。 この上でデータベースサービスが稼働する。 |
| フェイルオーバ | Primaryに故障が発生した際、Standbyを昇格させてデータベース サービスを切り替えること。 |
| フェイルバック | フェイルオーバ後、故障したノードを復旧して、フェイルオーバ前 の状態に戻すこと。 |
| リソース | Pacemakerが制御するノード上のソフトウェア・ハードウェア。 PostgreSQLプロセスなどがある。 |
| D-LAN (DBレプリケーションLAN) | PostgreSQLのレプリケーション機能で、データ転送に使用する ネットワーク。 |
| DBクラスタ | PostgreSQLのデータベース情報を格納するファイルの集まり。 |
| HAクラスタ | (1) PG-REXにおいてPacemakerが制御するノードの集まり、 およびそれらが構成するシステム。 (2) Pacemakerが管理する(1)のシステムの構成情報。 |
| IC-LAN (インターコネクトLAN) | HAクラスタの各ノードのPacemakerが互いに死活監視するために 使用されるネットワーク。 |
| Linux-HA | Linuxで高可用性を実現するためのソフトウェアを開発する プロジェクトの名称。 |
| PG-REX | PostgreSQLの同期レプリケーションにPacemakerを組み合わせた 高可用ソリューション、およびそれが稼働するシステム。 |
| Primary | PostgreSQLデータベースへの更新・参照ができる状態、または その状態のノード。 |
| S-LAN(運用LAN) | ユーザやアプリケーションがPostgreSQLにアクセスするために 使用するネットワーク。 |
| Standby | PostgreSQLデータベースにてPrimaryでの更新情報を複製して 格納できる状態、またはその状態のノード。 |
| STONITH-LAN | STONITH機能で使用するネットワーク。 |
| xmlファイル | Pacemakerのリソース定義をしたxmlファイルのこと。 |

1. 以降はRHELと表記します。 [↑](#footnote-ref-1)
2. 以降はRHEL HA Add-Onと表記します。 [↑](#footnote-ref-2)
3. 本書の内容はPostgreSQL14.0で確認しました。 [↑](#footnote-ref-3)
4. RHEL HA Add-On 8.4に同梱されたものです。 [↑](#footnote-ref-4)
5. PostgreSQL関連のツールの対応状況によって日本語が表示されないこともあります。 [↑](#footnote-ref-5)
6. 参照先：[<http://www.postgresql.jp/document/14/html/functions-admin.html>](http://www.postgresql.jp/document/14/html/functions-admin.html) [↑](#footnote-ref-6)
7. 参照先：[<http://www.postgresql.jp/document/14/html/functions-admin.html>](http://www.postgresql.jp/document/14/html/functions-admin.html) [↑](#footnote-ref-7)
8. 参照先：[<http://www.postgresql.jp/document/14/html/warm-standby.html#STANDBY-PLANNING>](http://www.postgresql.jp/document/14/html/warm-standby.html#STANDBY-PLANNING) [↑](#footnote-ref-8)
9. 参照先：[<http://www.postgresql.jp/document/14/html/hot-standby.html#HOT-STANDBY-CONFLICT>](http://www.postgresql.jp/document/14/html/hot-standby.html#HOT-STANDBY-CONFLICT) [↑](#footnote-ref-9)
10. ダウンロードサイト：[<http://linux-ha.osdn.jp/wp/dl>](http://linux-ha.osdn.jp/wp/dl) [↑](#footnote-ref-10)
11. インストールDVDがメディア挿入時に自動マウントされた場合は、一旦手動でアンマウントし、再度マウントしてください。 [↑](#footnote-ref-11)
12. <https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/8/html/configuring_and_managing_high_availability_clusters/assembly_configuring-fencing-configuring-and-managing-high-availability-clusters#proc_configuring-acpi-for-fence-devices-configuring-fencing> [↑](#footnote-ref-12)
13. logind.confを修正する手順は、デフォルトターゲットがmultiuser.targetの場合にのみ有効です。graphical.target等の他のデフォルトターゲットの場合は、Red Hat社の次のドキュメントを参照し てください。 <https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/8/html/using_the_desktop_environment_in_rhel_8/customizing-gnome-desktop-features_using-the-desktop-environment-in-rhel-8#changing-behavior-when-pressing-the-powerbutton_customizing-gnome-desktop-features> [↑](#footnote-ref-13)
14. リストアコマンドはpm\_pcsgen環境定義書に設定します。設定方法は『リソース（PostgreSQL）の設定』を参照してください。 [↑](#footnote-ref-14)
15. tcp\_keepalives\_idle、tcp\_keepalives\_interval、tcp\_keepalives\_countの設定を指します。 [↑](#footnote-ref-15)
16. レプリケーション受付用の仮想IPアドレスの設定 [↑](#footnote-ref-16)
17. 相手のノードのD-LANのIPアドレスの設定 [↑](#footnote-ref-17)
18. アーカイブコマンドの設定は『postgresql.confの編集』を参照してください。 [↑](#footnote-ref-18)
19. リソース定義xmlファイル生成時にリソース定義用スクリプトファイル(PG-REX14\_pm\_pcsgen\_env.sh)も生成されますが、本マニュアルでは利用しません。 [↑](#footnote-ref-19)
20. 参照先: [<http://www.postgresql.jp/document/14/html/hot-standby.html#HOT-STANDBY-ADMIN>](http://www.postgresql.jp/document/14/html/hot-standby.html#HOT-STANDBY-ADMIN) [↑](#footnote-ref-20)
21. 参照先：[<http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP) [↑](#footnote-ref-21)
22. 本書の「PostgreSQLのバックアップ」の節にある操作例を参照 [↑](#footnote-ref-22)
23. バックアップ履歴ファイルは、拡張子が.backupのファイルを指します。  
    [<http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP) [↑](#footnote-ref-23)
24. タイムライン履歴ファイルは、拡張子が.historyのファイルを指します。  
    [<http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-TIMELINES>](http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-TIMELINES) [↑](#footnote-ref-24)
25. pcs statusでノード名との対応が表示されます。詳細は【ノード情報表示部】を参照してください。 [↑](#footnote-ref-25)
26. pcs statusでノード名との対応が表示されます。詳細は【ノード情報表示部】を参照してください。 [↑](#footnote-ref-26)
27. PostgreSQLのサーバログファイルの場所は設定により異なります。詳細は『PostgreSQLドキュメント』を参照してください。 [↑](#footnote-ref-27)
28. バックアップ履歴ファイルは、拡張子が.backupのファイルを指します。  
    [<http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-BASE-BACKUP) [↑](#footnote-ref-28)
29. タイムライン履歴ファイルは、拡張子が.historyのファイルを指します。  
    [<http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-TIMELINES>](http://www.postgresql.jp/document/14/html/continuous-archiving.html#BACKUP-TIMELINES) [↑](#footnote-ref-29)