PG-REX 11利用マニュアル

第 1.0.0版

2019年02月13日

NTT OSSセンタ

改版履歴

|  |  |  |
| --- | --- | --- |
| 日付 | 版数 | 変更内容 |
| 2019/02/13 | 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。

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

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 9](#_Toc950391)

[1.1. 本書の目的 9](#_Toc950392)

[1.2. 対象読者 9](#_Toc950393)

[1.3. 表記規則 10](#_Toc950394)

[1.4. 対象とするソフトウェアとバージョン 10](#_Toc950395)

[1.5. PG-REX 10からの主な変更点 10](#_Toc950396)

[1.6. 前提条件 11](#_Toc950397)

[2章. 概要 12](#_Toc950398)

[2.1. PG-REXの概要 12](#_Toc950399)

[2.2. システム構成 12](#_Toc950400)

[2.3. 制約 12](#_Toc950401)

[2.3.1. PG-REXの制約 13](#_Toc950402)

[2.3.2. PG-REX運用補助ツールの制約 14](#_Toc950403)

[2.4. 推奨設定 15](#_Toc950404)

[3章. 環境構築 16](#_Toc950405)

[3.1. ネットワーク 16](#_Toc950406)

[3.2. ディレクトリ構成 18](#_Toc950407)

[3.3. Pacemaker 18](#_Toc950408)

[3.3.1. 事前作業 18](#_Toc950409)

[3.3.2. インストール 20](#_Toc950410)

[3.3.3. Pacemaker関連プロセスの自動起動の無効化 21](#_Toc950411)

[3.3.4. クラスタの基本設定（corosync.conf） 22](#_Toc950412)

[3.3.5. クラスタ認証設定ファイルの生成（authkey） 23](#_Toc950413)

[3.3.6. Pacemaker設定ファイルの追加設定（pacemaker） 23](#_Toc950414)

[3.3.7. Corosync用Unitファイル(corosync.service)の修正 23](#_Toc950415)

[3.3.8. Pacemaker用Unitファイル(pacemaker.service)の修正 24](#_Toc950416)

[3.3.9. ログの出力設定（rsyslog.conf） 24](#_Toc950417)

[3.3.10. ログメッセージ制御機能設定（pm\_logconv.conf） 25](#_Toc950418)

[3.3.11. ログローテーションの追加設定（syslog） 26](#_Toc950419)

[3.3.12. pm\_logconvのプロセス起動 27](#_Toc950420)

[3.3.13. ifcheckdプロセスの起動 27](#_Toc950421)

[3.4. PostgreSQL 28](#_Toc950422)

[3.4.1. PostgreSQLのインストール 28](#_Toc950423)

[3.4.2. 環境変数の設定 29](#_Toc950424)

[3.4.3. DBクラスタ用ディレクトリの作成 30](#_Toc950425)

[3.4.4. DBクラスタの初期化 30](#_Toc950426)

[3.4.5. postgresql.confの編集 30](#_Toc950427)

[3.4.6. レプリケーションユーザの作成 32](#_Toc950428)

[3.4.7. pg\_hba.confの編集 33](#_Toc950429)

[3.4.8. .pgpassの作成 34](#_Toc950430)

[3.5. PG-REX運用補助ツール 35](#_Toc950431)

[3.5.1. インストール 35](#_Toc950432)

[3.5.2. pg-rex\_tools.confの編集 36](#_Toc950433)

[3.5.3. ネットワーク接続登録 36](#_Toc950434)

[3.6. リソースの設定 38](#_Toc950435)

[3.6.1. pm\_crmgenの概要 38](#_Toc950436)

[3.6.2. 変更不要の設定 38](#_Toc950437)

[3.6.3. リソース構成の設定 38](#_Toc950438)

[3.6.4. STONITH実行順序の設定 39](#_Toc950439)

[3.6.5. リソース(Master側仮想IP)の設定 39](#_Toc950440)

[3.6.6. リソース(PostgreSQL)の設定 39](#_Toc950441)

[3.6.7. リソース（ネットワーク監視）の設定 40](#_Toc950442)

[3.6.8. リソース（ディスク監視）の設定 40](#_Toc950443)

[3.6.9. リソース（STONITH）の設定 40](#_Toc950444)

[3.6.10. リソース配置制約の設定 40](#_Toc950445)

[3.6.11. リソース同居制約の設定 41](#_Toc950446)

[3.6.12. リソース起動順序制約の設定 41](#_Toc950447)

[3.6.13. リソース（Slave側仮想IP）の設定 41](#_Toc950448)

[3.6.14. crmファイルの作成 41](#_Toc950449)

[3.6.15. crmファイルの反映 42](#_Toc950450)

[3.7. アンインストール 42](#_Toc950451)

[4章. 起動と停止 45](#_Toc950452)

[4.1. 両系の起動 45](#_Toc950453)

[4.2. Masterの起動 45](#_Toc950454)

[4.3. Slaveの起動 48](#_Toc950455)

[4.4. 両系の停止 50](#_Toc950456)

[4.5. Slaveの停止 51](#_Toc950457)

[4.6. Masterの停止 51](#_Toc950458)

[4.7. PostgreSQL停止中の系の停止 52](#_Toc950459)

[5章. メンテナンス時の対応 53](#_Toc950460)

[5.1. PostgreSQLのバックアップ 53](#_Toc950461)

[5.1.1. PostgreSQLのバックアップ 53](#_Toc950462)

[5.2. アーカイブログの削除 53](#_Toc950463)

[5.2.1. PostgreSQLアーカイブログの削除 54](#_Toc950464)

[5.3. 計画的な系切り替え 55](#_Toc950465)

[5.3.1. 系切り替え 56](#_Toc950466)

[5.4. 計画的なメンテナンスのための系の停止、起動 58](#_Toc950467)

[5.4.1. メンテナンスのための系の停止 58](#_Toc950468)

[5.4.2. メンテナンス後の系の起動 59](#_Toc950469)

[6章. 故障対応 60](#_Toc950470)

[6.1. 前提条件 60](#_Toc950471)

[6.2. 監視コマンド表示確認方法 60](#_Toc950472)

[6.2.1. 表示部説明 62](#_Toc950473)

[6.2.2. 正常状態確認方法 66](#_Toc950474)

[6.3. 故障箇所特定手順 67](#_Toc950475)

[6.3.1. ノード情報・リソース情報・故障回数表示部の確認 67](#_Toc950476)

[6.3.2. 属性情報表示部の確認 69](#_Toc950477)

[6.3.3. 制御エラー情報表示部の確認 70](#_Toc950478)

[6.3.4. /var/log/pm\_logconv.outの確認 72](#_Toc950479)

[6.4. ルータ故障 73](#_Toc950480)

[6.4.1. 故障時のクラスタ状態 73](#_Toc950481)

[6.4.2. 復旧 73](#_Toc950482)

[6.5. S-LAN故障 76](#_Toc950483)

[6.5.1. 故障時のクラスタ状態 76](#_Toc950484)

[6.5.2. 復旧 76](#_Toc950485)

[6.6. D-LAN故障 79](#_Toc950486)

[6.6.1. 故障時のクラスタ状態 79](#_Toc950487)

[6.6.2. 復旧 79](#_Toc950488)

[6.7. リソース故障(monitor) 82](#_Toc950489)

[6.7.1. 故障時のクラスタ状態 82](#_Toc950490)

[6.7.2. 復旧 82](#_Toc950491)

[6.8. リソース故障(demote/stop) 85](#_Toc950492)

[6.8.1. 故障時のクラスタ状態 85](#_Toc950493)

[6.8.2. 復旧 85](#_Toc950494)

[6.9. リソース故障(vip-master) 88](#_Toc950495)

[6.9.1. 故障時のクラスタ状態 88](#_Toc950496)

[6.9.2. 復旧 88](#_Toc950497)

[6.10. リソース故障(vip-rep) 91](#_Toc950498)

[6.10.1. 故障時のクラスタ状態 91](#_Toc950499)

[6.10.2. 復旧 91](#_Toc950500)

[6.11. リソース故障(vip-slave) 93](#_Toc950501)

[6.11.1. 故障時のクラスタ状態 93](#_Toc950502)

[6.11.2. 復旧 93](#_Toc950503)

[6.12. 内蔵ディスク故障 95](#_Toc950504)

[6.12.1. 故障時のクラスタ状態 95](#_Toc950505)

[6.12.2. 復旧 95](#_Toc950506)

[6.13. ノード故障 98](#_Toc950507)

[6.13.1. 故障時のクラスタ状態 98](#_Toc950508)

[6.13.2. 復旧 98](#_Toc950509)

[6.14. IC-LAN故障 101](#_Toc950510)

[6.14.1. 故障時のクラスタ状態 101](#_Toc950511)

[6.14.2. 復旧 101](#_Toc950512)

[7章. コマンド直接実行の運用 104](#_Toc950513)

[7.1. 起動と停止 104](#_Toc950514)

[7.1.1. 両系の起動 104](#_Toc950515)

[7.1.2. Masterの起動 104](#_Toc950516)

[7.1.3. Slaveの起動 109](#_Toc950517)

[7.1.4. 両系の停止 112](#_Toc950518)

[7.1.5. Slaveの停止 112](#_Toc950519)

[7.1.6. Masterの停止 112](#_Toc950520)

[7.2. アーカイブログの削除 113](#_Toc950521)

[7.2.1. PostgreSQLアーカイブログの削除 113](#_Toc950522)

[7.3. 計画的な系切り替え 115](#_Toc950523)

[7.3.1. 系切り替え 115](#_Toc950524)

[8章. 用語集 118](#_Toc950525)

# はじめに

## 本書の目的

本書は、PostgreSQL-REX11(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/1.1/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 11付属ドキュメント

* <http://www.postgresql.jp/document/11>
* ⇒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
  + eth1
* # : rootユーザの操作記号
  + # crm\_mon
* $ : 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

本書が対象とする各ソフトウェアのバージョンは以下の通りです。

* RedHat Enterprise Linux 7
* PostgreSQL 11
* Pacemaker 1.1.19

## PG-REX 10からの主な変更点

PG-REX 10からの主な変更点は以下の通りです。

* 対応するPostgreSQLのメジャーバージョンを10から11に変更
* ホスト名に大文字が使用できないという制約の解除
* マニュアルの刷新

## 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[[1]](#footnote-1)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の動作環境を構築する2つのサーバをpgrex01、pgrex02と呼びます。それぞれのサーバのノード名（uname -nの結果）はpgrex01、pgrex02とします。実際の環境に応じて、ノード名は適宜読み替えてください。

# 概要

## PG-REXの概要

PG-REXは、PostgreSQL11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、PostgreSQL11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、MasterからSlave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MasterとSlaveの両系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MasterとSlaveを管理させ、自動フェイルオーバを提供します。Masterの故障を検知すると、Pacemakerは自動的にSlaveをMaster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、Slaveの故障を検知した場合でも、Pacemakerは自動的にMaster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。

## システム構成

本書の内容は以下の環境で確認しました。実行例も以下の環境でのものです。

* ハードウェア
  + HP　DL360 G7 (NIC ： 8ポート)
* OS
  + RedHat Enterprise Linux 7.5
* DBMS
  + PostgreSQL 11.1
* クラスタリング
  + Pacemaker 1.1.19-1.1

## 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. 両系共に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。
      * 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を構築するサーバには少なくとも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で制御可能なサーバ数は2台である。

1. 運用中の制約
   1. D-LANの切断やSlave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Master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。
      * MasterとSlaveともにトランザクションは確定済
      * Masterのみトランザクションは確定済。Slaveでは未確定
      * MasterとSlaveともにトランザクションは未確定
   3. pg\_cancel\_backend関数[[2]](#footnote-2) またはpg\_terminate\_backend関数[[3]](#footnote-3) をMaster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Masterにおいて既にコミットされている状態となる。ただし、Slave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[[4]](#footnote-4)。
      * テーブルスペースの作成 両系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 両系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Slave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。

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. Slaveの起動後、SlaveがMasterに追いつく(同期が完了する)までの間にMasterが終了した場合、フェイルオーバは行われない。
   3. Slave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[[5]](#footnote-5)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Masterとして稼働しているときに、旧Masterのpgrex01をSlaveとして組み込みしても、自動的にpgrex01がMaster、pgrex02がSlave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による系切り替えでは、Masterの停止後からMasterの切り替え（新Masterの起動）が完了するまでの間は一時的にサービスが停止した状態となる。
2. pg-rex\_switchoverによる系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、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、Diskd、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. 両系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（デフォルトは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両系で同一であること。
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. DB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、Master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、各サーバにNWインターフェイスが7つある構成となっています。実際のサーバの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(eth0, eth5) : 192.168.0.11
  + pgrex02-bond0(eth0, eth5) : 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(eth2, eth4) : 192.168.2.1
  + pgrex02-bond1(eth2, eth4) : 192.168.2.2
* IC-LAN(インターコネクトLAN)
  + pgrex01-eth1 : 192.168.1.1
  + pgrex01-eth3 : 192.168.3.1
  + pgrex02-eth1 : 192.168.1.2
  + pgrex02-eth3 : 192.168.3.2
* STONITH-LAN
  + pgrex01-eth6 : 172.20.144.41
  + pgrex02-eth6 : 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、Master側接続用、Slave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)
  + Master側接続用 : 192.168.0.10
  + Slave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、両系（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

【watchdogデバイスを使用しているhpwdtモジュール停止の設定】

ハードウェアwatchdogのモジュールを停止します。

HP社のハードウェアを使用している環境においては、OS起動時にhpwdtのモジュールがロードされている場合、Pacemakerがwatchdogデバイスを使用できないため、hpwdtモジュールを停止します。

1. hpwdtモジュールがロードされているかを確認します。

* hpwdtモジュールがロードされている場合は、以下の例のように表示されます。

|  |
| --- |
| # dmesg | grep "hpwdt" ：(省略) hpwdt 0000:02:00.0: HP Watchdog Timer Driver: NMI decoding initialized,... ：(省略) |

1. hpwdtモジュールがロードされている場合は、/etc/modprobe.d/blacklist.conf に「blacklist hpwdt」を追記します。

* さらに、hp-asrdサービスの停止と自動起動を停止する設定を行い、モジュールをアンロードします。

|  |
| --- |
| # systemctl stop hp-asrd.service ← hp-asrdサービスを停止 # chkconfig hp-asrd off ← hp-asrdサービスの自動起動を停止 # rmmod hpwdt ← モジュールをアンロード |

【watchdogデバイスを使用しているiTCO\_wdtモジュール停止の設定】

OS起動時にiTCO\_wdtモジュールがロードされている場合、Pacemakerがwatchdogデバイスを使用できないため、iTCO\_wdtモジュールを停止します。

1. iTCO\_wdtモジュールがロードされているかを確認します。以下の両方を満たす場合はiTCO\_wdtモジュールがロードされています。

・ コマンド実行結果に以下のメッセージが表示されている。

|  |
| --- |
| # dmesg | grep iTCO\_wdt [ 16.638148] iTCO\_wdt: Intel TCO WatchDog Timer Driver v1.11 ：(省略) |

・ 上記コマンド実行結果に以下のメッセージが含まれていない。(ハードウェア/BIOSにより利用不可の場合)

|  |
| --- |
| [ 16.638189] iTCO\_wdt: unable to reset NO\_REBOOT flag, device disabled by hardware/BIOS |

1. iTCO\_wdtモジュールがロードされている場合は、/etc/modprobe.d/blacklist.conf に「blacklist iTCO\_wdt」を追記します。

### 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[[6]](#footnote-6)。

1. RHELのインストールDVDイメージを/mediaにマウントします。



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

1. /media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。



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

1. Pacemakerリポジトリパッケージ[[7]](#footnote-7)を/opt/linux-ha配下に展開します。



|  |
| --- |
| # yum install -y pacemaker-repo-1.1.19-1.1.el7.x86\_64.rpm  ※ バージョン部分は適宜読み替えてください。 |

1. yumのキャッシュをクリアします。



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

1. Pacemakerおよびipmitool[[8]](#footnote-8)をインストールします。



|  |
| --- |
| # yum install -y pacemaker-all |

1. 全てのパッケージがインストールされたこと(「パッケージxxxはインストールされていません。」が表示されないこと)を確認します。



|  |
| --- |
| # rpm -q pacemaker-libs pacemaker-cli pacemaker-cluster-libs pacemaker corosynclib corosync cluster-glue cluster-glue-libs resource-agents crmsh libqb pm\_diskd pm\_extras pm\_crmgen pm\_logconv-cs ipmitool |

1. RHELのインストールDVDイメージを/mediaからアンマウントします。



|  |
| --- |
| # umount /media |

### Pacemaker関連プロセスの自動起動の無効化

pgrex01とpgrex02でPacemaker関連プロセスの自動起動を無効に設定します。

|  |
| --- |
| # systemctl disable corosync # systemctl disable pacemaker |

### クラスタの基本設定（corosync.conf）

pgrex01とpgrex02でクラスタの基本設定ファイル（corosync.conf）を、『Pacemaker関連サイト』に従って作成します。

|  |
| --- |
| ファイル名 : /etc/corosync/corosync.conf 所有者 : root グループ : root 権限 : 644 ファイルの内容  totem {  version: 2  rrp\_mode: active  token: 1000  rrp\_problem\_count\_timeout: 2000  interface {  ringnumber: 0  bindnetaddr: 192.168.1.0  mcastaddr: 239.255.1.1  mcastport: 5405  }  interface {  ringnumber: 1  bindnetaddr: 192.168.3.0  mcastaddr: 239.255.1.2  mcastport: 5405  } } logging {  syslog\_facility: local1  debug: off } quorum {  provider: corosync\_votequorum  expected\_votes: 2 }  ※ 「bindnetaddr」には、IC-LANのネットワークアドレスを設定します。「syslog\_facility」は、『ログの出力設定（rsyslog.conf）』で/etc/rsyslog.confに追記する設定と同一とします。 |

### クラスタ認証設定ファイルの生成（authkey）

pgrex01とpgrex02にクラスタ認証設定ファイルを生成します。

1. pgrex01でCorosync認証設定ファイルを生成します。



|  |
| --- |
| # corosync-keygen -l |

1. ファイル/etc/corosync/authkeyが生成されたことを確認し、pgrex02の同じ場所にコピーします。



|  |
| --- |
| # scp /etc/corosync/authkey 192.168.2.2:/etc/corosync/ |

### Pacemaker設定ファイルの追加設定（pacemaker）

pgrex01とpgrex02で、Pacemaker設定ファイル(/etc/sysconfig/pacemaker)に以下の内容を追加します。

|  |
| --- |
| PCMK\_logfile=none PCMK\_logfacility=local1 PCMK\_logpriority=info PCMK\_fail\_fast=yes |

### Corosync用Unitファイル(corosync.service)の修正

pgrex01とpgrex02で、Corosync用Unitファイル(corosync.service)を修正します。

1. corosyncプロセスが故障した場合にcorosyncのwatchdogを効かせたい場合は、Corosync用Unitファイル(corosync.service)を/usr/lib/systemd/system/corosync.serviceから/etc/systemd/system配下にコピーし、下記の行のコメントを外します。



|  |
| --- |
| Restart=on-failure RestartSec=70 |

1. softdogを使用する場合は、"/etc/systemd/system/corosync.service"の下記の行のコメントを外します。



|  |
| --- |
| ExecStartPre=/sbin/modprobe softdog  ※ softdog使用時のwatchdog発動待ち時間の設定はCorosyncのデフォルト値(6秒)が有効となります |

### Pacemaker用Unitファイル(pacemaker.service)の修正

pgrex01とpgrex02で、Pacemaker用Unitファイル(pacemaker.service)を修正します。

pacemaker.serviceは、/usr/lib/systemd/system/pacemaker.serviceを/etc/systemd/system配下にコピーし、下記の行のコメントを外します。

|  |
| --- |
| ExecStopPost=/bin/sh -c 'pidof crmd || killall -TERM corosync' |

### ログの出力設定（rsyslog.conf）

『Pacemaker関連サイト』を参考にpgrex01とpgrex02でPacemakerのログ出力先を設定します。

1. /etc/rsyslog.confにログの出力先を設定(赤字の部分を追記)します。



|  |
| --- |
| (略) \*.info;mail.none;authpriv.none;cron.none;local1.none /var/log/messages (略) # Save boot messages also to boot.log local7.\* /var/log/boot.log (略) local1.info /var/log/ha-log;RSYSLOG\_TraditionalFileFormat |

1. 大量ログ出力時の省略を抑止するため、/etc/rsyslog.conf、/etc/systemd/journald.confに設定を追加します。

|  |
| --- |
| /etc/rsyslog.confの追加内容 |
| $SystemLogRateLimitInterval 0 $imjournalRatelimitInterval 0 |

|  |
| --- |
| /etc/systemd/journald.confの追加内容 |
| RateLimitInterval=0s |

1. 設定を反映するため、rsyslogを再起動します。



|  |
| --- |
| # systemctl restart rsyslog |

1. 設定を反映するため、journaldを再起動します。



|  |
| --- |
| # systemctl restart systemd-journald |

### ログメッセージ制御機能設定（pm\_logconv.conf）

『Pacemaker関連サイト』を参考にpgrex01とpgrex02のログメッセージ制御機能の設定を行います。pm\_logconv.confは、/etc/pm\_logconv.conf.sampleをコピーして/etc/pm\_logconv.confを作成し、基本設定部([Setting]セクション)を以下を参考に設定します。

|  |
| --- |
| [Settings] #ha\_log\_path = /var/log/ha-log #output\_path = /var/log/pm\_logconv.out #output\_logfacility = local2 #syslogformat = True attribute\_ping = not\_defined default\_ping\_set or default\_ping\_set lt 100 # ネットワーク監視を行う場合の設定 attribute\_diskd = not\_defined diskcheck\_status or diskcheck\_status eq ERROR # ディスク監視を行う場合の設定 attribute\_diskd\_inner = not\_defined diskcheck\_status\_internal or diskcheck\_status\_internal eq ERROR # ディスク監視を行う場合の設定 #logconv\_logfacility = daemon #logconv\_logpriority = info act\_rsc = vip-master, vip-rep # フェイルオーバの発生と成否の判断基準となるリソースのIDを設定 （略） |

### ログローテーションの追加設定（syslog）

pgrex01とpgrex02のログファイルのローテーションの設定を行ないます。

Pacemaker用ログファイル(/var/log/ha-log)のローテーション設定をするため、/etc/logrotate.d/syslogに赤字部分を追記します。

|  |
| --- |
| /var/log/cron /var/log/maillog /var/log/messages /var/log/secure /var/log/spooler /var/log/ha-log {  missingok  sharedscripts  postrotate  /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true  endscript } |

ログメッセージ制御機能用のログファイル(/var/log/pm\_logconv.out)のローテーション設定ファイルは、インストール時に作成されるため追加設定は不要です。

|  |
| --- |
| /var/log/pm\_logconv.out {  missingok } |

インストール時に作成されるログローテーション設定ファイル/etc/logrotate.d/pacemakerは使用しないため、削除します。

|  |
| --- |
| # rm -f /etc/logrotate.d/pacemaker |

### pm\_logconvのプロセス起動

pgrex01とpgrex02でpm\_logconvプロセスを起動します。

1. pm\_logconvプロセスの自動起動を有効に設定します。



|  |
| --- |
| # systemctl is-enabled pm\_logconv disabled # systemctl enable pm\_logconv Created symlink from /etc/systemd/system/multi-user.target.wants/pm\_logconv.service to /usr/lib/systemd/system/pm\_logconv.service. # systemctl is-enabled pm\_logconv enabled |

1. pm\_logconvプロセスを即時、起動します。



|  |
| --- |
| # systemctl start pm\_logconv |

1. pm\_logconvプロセスが起動していることを確認します。



|  |
| --- |
| # systemctl is-active pm\_logconv active |

### ifcheckdプロセスの起動

pgrex01とpgrex02でifcheckdプロセスを起動します。

1. ifcheckdプロセスの自動起動を有効に設定します。



|  |
| --- |
| # systemctl is-enabled ifcheckd disabled # systemctl enable ifcheckd Created symlink from /etc/systemd/system/multi-user.target.wants/ifcheckd.service to /usr/lib/systemd/system/ifcheckd.service. # systemctl is-enabled ifcheckd enabled |

1. ifcheckdプロセスを即時、起動します。



|  |
| --- |
| # systemctl start ifcheckd |

1. ifcheckdプロセスが起動していることを確認します。



|  |
| --- |
| # systemctl is-active ifcheckd active |

## 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11のみとなります。

本作業はrootユーザで行います。

PG-REXのインストール必須のRPMパッケージを以下に示します。

|  |
| --- |
| postgresql11-libs-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-server-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-contrib-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-docs-11.1-1PGDG.rhel7.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 postgresql11-libs-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-server-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-contrib-11.1-1PGDG.rhel7.x86\_64.rpm postgresql11-docs-11.1-1PGDG.rhel7.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-11」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。

以下のコマンドを実行し、PostgreSQLのRPMパッケージが正常にインストールされていることを確認します。

|  |
| --- |
| # rpm -qa | grep postgres postgresql11-server-11.1-1PGDG.rhel7.x86\_64 postgresql11-11.1-1PGDG.rhel7.x86\_64 postgresql11-contrib-11.1-1PGDG.rhel7.x86\_64 postgresql11-docs-11.1-1PGDG.rhel7.x86\_64 postgresql11-libs-11.1-1PGDG.rhel7.x86\_64  ※ バージョンは適宜読み替えてください。 |

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 # id postgres uid=26(postgres) gid=26(postgres) groups=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-11/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 | 任意 | 任意 | postgres ユーザが読み書き可能な権限 |
| 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 (略) Success. You can now start the database server using:   pg\_ctl -D /dbfp/pgdata/data -l logfile start |

### 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 = '\*' wal\_level = replica max\_wal\_senders = 10 wal\_keep\_segments = 32 hot\_standby = on max\_standby\_streaming\_delay = -1 max\_standby\_archive\_delay = -1 archive\_mode = always archive\_command = '/bin/cp %p /dbfp/pgarch/arc1/%f' #synchronous\_standby\_names = '' synchronous\_commit = on restart\_after\_crash = off wal\_sender\_timeout = 20s wal\_receiver\_timeout = 20s hot\_standby\_feedback = on max\_replication\_slots = 10 |

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。

* wal\_level
* logicalを指定することも可能。
* 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以上を指定する必要がある。ただし、pg\_basebackupによる運用中のバックアップ取得など、Slave以外からの接続に備え、余裕を持たせて指定することを推奨する。
* wal\_keep\_segments
* 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、Slaveに転送されていないWALファイルがMaster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。
  + 設定値が大きい場合、WALの不要なファイルキャッシュの削除にかかる時間が長くなり、性能が劣化する。
* hot\_standby
* 設定はon必須。
* max\_standby\_streaming\_delay
* PG-REXでは、Slave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)を指定する。
* max\_standby\_archive\_delay
* PG-REXでは、Slave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。
* synchronous\_standby\_names
* PG-REXでは自動的に必要な設定を行うため、ユーザは設定してはならない。
* 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を指定した場合、故障発生時にデータを損失する可能性がある。remote\_applyを指定した場合、on指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指定例を以下に示す。
* '/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[[9]](#footnote-9)にもgzipを使用すること。
* 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を指定しなければならない。
* wal\_sender\_timeout
* Slaveの故障や両系間の通信断をMaster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[[10]](#footnote-10) 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Slave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。
* wal\_receiver\_timeout
* wal\_sender\_timeoutと同程度に揃える必要がある。
* hot\_standby\_feedback
* 設定はon必須。
* max\_replication\_slots
* pg\_basebackupによるバックアップ取得などに備え、余裕を持たせて指定することを推奨する。

### 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 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 |

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'" CREATE ROLE |

1. 作成したデータベースユーザにレプリケーション権限が付与されていることを確認します。



|  |
| --- |
| $ psql -c "\du repuser"                   ロール一覧  ロール名 |        属性        | 所属グループ  ----------+--------------------+--------------  repuser  | レプリケーション可 | {} |

1. PostgreSQLを停止します。



|  |
| --- |
| $ 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 |

### pg\_hba.confの編集

pgrex01でpg\_hba.confを作成します。本節は、PG-REXを構成するのに必要な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は以下のとおりで、レプリケーションの接続を許可しなければなりません。必ず全てのパラメータをpg\_hba.confに設定してください。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。

|  |
| --- |
| # TYPE DATABASE    USER    ADDRESS        METHOD host replication repuser 192.168.2.1/32 md5 host replication repuser 192.168.2.2/32 md5 |

レプリケーションの接続を許可するにあたって、注意すべき点を以下に示します。

* TYPE
* hostを設定する。
* DATABASE
* replicationを設定する。
* USER
* 『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。
* ADDRESS
* 各系のD-LANのIPアドレスを設定する。
* METHOD
* md5を設定する。

### .pgpassの作成

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.pgpassを作成します。

本節は、PG-REXを構成するのに必要な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について、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。

|  |
| --- |
| # hostname:port:database:username:password 192.168.2.3:5432:replication:repuser:reppasswd # レプリケーション受付用の仮想IPアドレスの設定 192.168.2.2:5432:replication:repuser:reppasswd # 相手ノードのD-LANのIPアドレスの設定 |

.pgpassファイルを作成するにあたって、注意すべき点を以下に示します。

* hostname
* レプリケーション受付用の仮想IPアドレスおよび相手ノードのD-LANのIPアドレスを指定する。
* port
* 相手の系が接続を待ち受けるポート番号を指定する。
* database
* replicationを指定する。
* username
* 『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指定する。
* password
* 『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のパスワードを指定する。

## 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-11.0-1.el7.noarch.rpm IO\_Tty-1.11-1.el7.x86\_64.rpm Net\_OpenSSH-0.62-1.el7.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。

|  |
| --- |
| # rpm -ivh Net\_OpenSSH-0.62-1.el7.x86\_64.rpm # rpm -ivh IO\_Tty-1.11-1.el7.x86\_64.rpm # rpm -ivh pg-rex\_operation\_tools\_script-11.0-1.el7.noarch.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 : 192.168.2.1,192.168.2.2
* 両系のD-LAN IPアドレスを指定する。
* Archive\_dir : /dbfp/pgarch/arc1
* アーカイブディレクトリの絶対パスを指定する。
* VIP\_SLAVE : disable
* vip\_slaveを使用する環境の場合は enable、それ以外の場合はdisableを指定する。
* DISKD\_ResourceID : clnDiskd1, clnDiskd2
* diskdのリソースID を複数設定する。

設定例を以下に示します。

|  |
| --- |
| D-LAN\_IPAddress = 192.168.2.1, 192.168.2.2 Archive\_dir = /dbfp/pgarch/arc1 VIP\_SLAVE = disable DISKD\_ResourceID = clnDiskd1, clnDiskd2 |

### ネットワーク接続登録

PG-REX運用補助ツールでは両系の状態確認にネットワークの通信を用いるので、事前に両系の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)? 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)? 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

PG-REX用の『pm\_crmgen環境定義書』(ファイル名:PG-REX11\_pm\_crmgen\_env.xlsx)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\_crmgenの概要

pm\_crmgenは『pm\_crmgen環境定義書』からcrmファイルを生成するツールです。生成されたcrmファイルは、crmコマンドを使用することでPacemakerに読み込まれ、設定ファイルが生成されます。

PG-REXでのcrm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\_crmgen環境定義書』を編集する。
2. 編集した『pm\_crmgen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\_crmgenコマンドを使用し、csvファイルをcrmファイルに変換する。

以降、『pm\_crmgen環境定義書』を編集します。

### 変更不要の設定

『pm\_crmgen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11-1 リソースチケット制約
* 表12-1 追加設定
* 表13-1 ALERT設定

### リソース構成の設定

DBクラスタとOSとを異なるディスクに配置している場合、『pm\_crmgen環境定義書』の表4-1の設定内容は変更不要です。反対に、同じディスクに配置している場合、表4-1から「リソースid」がclnDiskd2とprmDiskd2の行(計2行)を削除します。

### STONITH実行順序の設定

『pm\_crmgen環境定義書』の表6-1にSTONITHの実行順序を設定します。「STONITHの対象ノード」を環境にあわせて設定してください。

### リソース(Master側仮想IP)の設定

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

『pm\_crmgen環境定義書』の表7-1-2にレプリケーション受付用の仮想IPを設定します。

### リソース(PostgreSQL)の設定

『pm\_crmgen環境定義書』の表7-1-4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 | 同期モードを指定する。 |
| node\_list | pgrex01 pgrex02 | pgrex01とpgrex02のノード名を指定する。 (ノード名の間はスペース区切り) |
| master\_ip | 192.168.2.3 | 『pm\_crmgen環境定義書』の表7-1-2に設定 したレプリケーション受付用の仮想IPアドレ スを指定する。 |
| 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指定する。 アーカイブコマンド[[11]](#footnote-11)に対応したコマンドを 指定すること。 (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。 コマンド指定例を以下に示す。 '/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指定する。 |
| 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、Master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指定する。 0を指定した場合、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台同時に起動する際、 どちらがMasterとして起動するかをチェック する回数を指定する（デフォルトは3）。 このパラメータを設定すると、Master起動の 際にmonitor間隔×設定値の時間だけSlave で待機することになり、その分Masterとして 起動する時間が遅くなる。 本マニュアルでは、2台同時に起動するケー スがないため、0を指定する。 |

### リソース（ネットワーク監視）の設定

『pm\_crmgen環境定義書』の表7-2-1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（ディスク監視）の設定

『pm\_crmgen環境定義書』にディスク監視の設定をします。DBクラスタとOSを異なるディスクに配置している場合は、表7-3-1と7-3-2を設定します。反対に、同じディスクの構成にしている場合は、表7-3-1だけを設定し、表7-3-2はすべての行(計16行)を削除します。

### リソース（STONITH）の設定

『pm\_crmgen環境定義書』の表7-4-1～7-4-2にpgrex01のSTONITHを設定します。表7-5-1～7-5-2にはpgrex02のSTONITHを設定します。

### リソース配置制約の設定

DBクラスタとOSを異なるディスクに配置している場合、『pm\_crmgen環境定義書』の表8-1の設定内容は変更不要です。反対に、同じディスクの構成にしている場合、表8-1から「条件属性名」がdiskcheck\_statusの行(計2行)を削除します。

### リソース同居制約の設定

DBクラスタとOSを異なるディスクに配置している場合、『pm\_crmgen環境定義書』の表9-1の設定内容は変更不要です。反対に、同じディスクの構成にしている場合、表9-1から「制約対象リソースID」がclnDiskd2の行(計1行)を削除します。

### リソース起動順序制約の設定

DBクラスタとOSを異なるディスクに配置している場合、『pm\_crmgen環境定義書』の表10-1の設定内容は変更不要です。反対に、同じディスクの構成にしている場合、表10-1から「先に起動するリソース」がclnDiskd2の行(計1行)を削除します。

### リソース（Slave側仮想IP）の設定

Slaveへのデータベース接続を利用する場合は、『pm\_crmgen環境定義書』の表7-1-3にSlave側接続用の仮想IPを設定します。Slaveへのデータベース接続を利用しない場合は、『pm\_crmgen環境定義書』の表4-1、表7-1-3、表8-1からvip-slaveの設定値をコメントアウトします。

### crmファイルの作成

以下の操作を行い、crmファイルを作成します。

本作業はrootユーザで行います。

1. Microsoft® Excelを使って『pm\_crmgen環境定義書』の「pm\_crmgen\_環境定義書(PG-REX11)」シートを修正し、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-REX11\_pm\_crmgen\_env.csv」とします。

1. 保存したcsvファイル（PG-REX11\_pm\_crmgen\_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\_crmgenコマンドを使用し、転送したcsvファイルをcrm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-REX11\_pm\_crmgen\_env.crm」とします。

|  |
| --- |
| # pm\_crmgen -o PG-REX11\_pm\_crmgen\_env.crm PG-REX11\_pm\_crmgen\_env.csv |

1. 生成後のcrmファイルを確認します。



|  |
| --- |
| # ls -l PG-REX11\_pm\_crmgen\_env.crm -rw-r--r-- 1 root root  6722  11月 19 10:05 2018 PG-REX11\_pm\_crmgen\_env.crm  ※ ファイルの権限、所有者が上記と異なる場合は修正します。 # chown root:root PG-REX11\_pm\_crmgen\_env.crm # chmod 644 PG-REX11\_pm\_crmgen\_env.crm |

### crmファイルの反映

作成したcrmファイルをPacemakerへ反映します。

crmファイルの反映はMaster起動時に行います。

crmファイルを反映させてMasterを起動する手順については、『Masterの起動』を参照してください。

## アンインストール

PG-REXをpgrex01とpgrex02の両方からアンインストールします。PG-REXが起動している場合は『両系の停止』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 postgresql11-contrib-11.1-1PGDG.rhel7.x86\_64 # rpm -e postgresql11-server-11.1-1PGDG.rhel7.x86\_64 # rpm -e postgresql11-docs-11.1-1PGDG.rhel7.x86\_64 # rpm -e postgresql11-11.1-1PGDG.rhel7.x86\_64 # rpm -e postgresql11-libs-11.1-1PGDG.rhel7.x86\_64 # rpm -qa | grep postgresql (出力なし)  ※ 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。

* Pacemaker本体およびPacemakerリポジトリパッケージのアンインストールを行います。

|  |
| --- |
| # yum erase -y pacemaker corosync libqb cluster-glue cluster-glue-libs resource-agents pm\_crmgen pm\_logconv-cs pm\_diskd pm\_extras crmsh pssh # yum erase -y pacemaker-repo # yum list installed | grep pacemaker-repo (出力なし) |

1. 以下の設定ファイル、ログファイル、ディレクトリを削除します。

* 設定ファイル
  + /etc/pm\_logconv.conf
  + /etc/sysconfig/pacemaker.rpmsave
  + /etc/systemd/system/corosync.service
  + /etc/systemd/system/pacemaker.service
* ログファイル
  + /var/log/ha-log\*
  + /var/log/pm\_logconv.out\*
* ディレクトリ
  + /etc/corosync
  + /var/lib/pacemaker
  + /var/run/heartbeat
  + /var/lib/corosync
  + /usr/libexec/pacemaker
  + /usr/lib/ocf/resource.d/heartbeat[[12]](#footnote-12)
  + /usr/lib/ocf/resource.d/pacemaker[[13]](#footnote-13)
* 以下の設定ファイル内に追記した設定を削除します。
  + /etc/rsyslog.conf
  + /etc/logrotate.d/syslog
  + /etc/systemd/journald.conf

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。

|  |
| --- |
| # rpm -e pg-rex\_operation\_tools\_script-11.0-1.el7.noarch # rpm -e Net\_OpenSSH-0.62-1.el7.x86\_64 # rpm -e IO\_Tty-1.11-1.el7.x86\_64 # rpm -qa | grep IO\_Tty (出力なし) # rpm -qa | grep Net\_OpenSSH (出力なし) # rpm -qa | grep pg-rex (出力なし)  ※ 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MasterとSlave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MasterとSlave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。

## 両系の起動

両系を起動するには、一方の系でMasterを起動させ、起動完了後、もう一方の系でSlaveを起動させます。どちらの系をMaster、Slaveとして稼働させるかは、ユーザが決定します。

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

## Masterの起動

本節では、Masterの起動手順を説明します。

1. どの系をMasterとして起動するか決定します。 PG-REXでは、最新のDBデータを持つ系をMasterとして起動しなければなりません。古いDBデータを持つ系をMaster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、Masterとして起動する系を決めるときの考え方の例です。
* + DBクラスタが片系のみに存在し、そのDBクラスタを使ってPG-REXを起動する場合(初めてMasterを起動する場合を含む)は、DBクラスタが存在する系をMasterとして起動する。
  + DBクラスタが両系に存在する場合は、直前までMasterとして稼働していた系をMaster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系をMasterとして起動する。
* 以降の手順では、pgrex01をMasterとして起動します。

1. ベースバックアップからMaster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Master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、Master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、Masterの起動の手順を行います。これにより、Master起動時のアーカイブリカバリは必要なくなるため、Master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。

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. Masterを起動します。



【初回もしくは新しいcrmファイルを反映させる場合】

Master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crmファイルを反映させる場合は、crmファイルを指定して、Masterを起動します。

|  |
| --- |
| # pg-rex\_master\_start PG-REX11\_pm\_crmgen\_env.crm 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 Master 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] /var/lib/pacemaker/cib 配下に既に cib ファイルがあります 削除して反映しても宜しいでしょうか？ (y/N) y 4. /var/lib/pacemaker/cib 配下の cib ファイルを削除 ...[OK] 5. Pacemaker 起動 ...[OK] 6. crm ファイルの反映 ...[OK] 7. Master の起動確認 ...[OK] ノード(pgrex01)が Master として起動しました |

【2回目以降の場合】

Masterを起動します。

|  |
| --- |
| # pg-rex\_master\_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 Master が存在していないことを確認 ...[OK] 3. Master 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. Master の起動確認 ...[OK] ノード(pgrex01)が Master 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Master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\_transactions

より詳しい説明については、PostgreSQLドキュメント[[14]](#footnote-14)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。

## Slaveの起動

本節では、Slaveの起動手順を説明します。以降の手順では、pgrex02をSlave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Slave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、Slave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\_slave\_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 Master 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Slaveを起動 q) Slave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. Master からベースバックアップ取得 22631/22631 kB (100%), 1/1 tablespace NOTICE: pg\_stop\_backup complete, all required WAL segments have been archived ...[OK] 7. Master 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. /var/lib/pacemaker/cib 配下のファイルを削除 ...[OK] 9. Slave の起動 (アーカイブリカバリ対象 WAL セグメント数: 1) ...[OK] 10. Slave の起動確認 ...[OK] ノード(pgrex02)が Slave として起動しました |

## 両系の停止

両系を停止するには、Slaveを停止させ、Slaveの停止が完了後にMasterを停止させます。

Master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。

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

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

## Slaveの停止

本節では、Slave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. Slaveを停止します。



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

## Masterの停止

本節では、Masterの停止手順を説明します。Slave稼働中にMaster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. Masterを停止します。



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

【注意】

Slave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。

|  |
| --- |
| Slave がまだ起動しています 系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停止中の系の停止

本節では、PostgreSQL停止中(Stopped)の系の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)の系の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(MasterまたはSlave)の停止を伴うメンテナンス(PG-REX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、Master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Masterで実行されるオンライン負荷に影響を与えないようにSlave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』[[15]](#footnote-15) 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Masterからバックアップを取得し直してください。旧Master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、vip-repを使用してMaster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 |

## 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[[16]](#footnote-16)、タイムライン履歴ファイル[[17]](#footnote-17))は、MasterおよびSlave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 両系のノード名を取得します  \*\*\*\*\*\* 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/arc1/20181119\_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/arc1/20181119\_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/arc1/20181119\_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。

## 計画的な系切り替え

計画的な系切り替えの手段として、PG-REX運用補助ツールのスイッチオーバ機能(pg-rex\_switchover)があります。本節では当該機能を使用した系切り替えの手順について記述します。

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系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。

### 系切り替え

運用補助ツールを使用した系切り替えは、PG-REXのMasterとSlave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で系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系切り替え後のクラスタ状態を確認します。系切り替えを実行しても良い場合は[y]を入力します。

* なお、系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)の読み込みと両系のノード名を取得 ...[OK] 2. 現在および系切り替え後のクラスタ状態を確認  [ 現在 / 系切り替え後のクラスタ状態 ] Master : pgrex01 -> pgrex02 Slave : pgrex02 -> pgrex01  系切り替え中は可用性が保証されません 系切り替えを実行してもよろしいでしょうか？ (y/N) y |

1. 系切り替えが完了するまで待機します。pg-rex\_switchoverコマンドが正常に完了した場合は、表示された現在のクラスタ状態が正しいことを確認してください。実行途中で異常終了した場合は、クラスタ状態が不確定であり、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。



|  |
| --- |
| 3. CHECKPOINT の実行 ...[OK] \*\*\*\*\*\* 系切り替えを実行 \*\*\*\*\*\* 4. Pacemaker の監視を停止 ...[OK] 5. Master (pgrex01) の PostgreSQL を停止 ...[OK] 6. Pacemaker の監視を再開し系切り替えを実行 ...[OK] 7. pgrex02 が新 Master になったことを確認  \*\*\*\*\*\* pgrex02 が Master として起動しました \*\*\*\*\*\*  8. pgrex01 の Pacemaker を停止 ...[OK] 9. pgrex01 で Slave を起動 00000011000000000000000C 00000012000000000000000E 00000013.history  \*\*\*\*\*\* pgrex01 が Slave として起動しました \*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\* 系切り替えが正常に完了しました \*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  [ 現在のクラスタ状態 ] Master : pgrex02 Slave : 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、crm\_monコマンドを実行しクラスタ状態を確認してください。crm\_mon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。

|  |
| --- |
| # crm configure property maintenance-mode=false |

## 計画的なメンテナンスのための系の停止、起動

本節では、MasterおよびSlaveの計画されたメンテナンス(PG-REX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)を実施する時の系の停止、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系の停止

メンテナンス対象の系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系の停止

* メンテナンス対象の系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. サービス継続の確認

* 起動中の系でサービスが継続していることを、crm\_monコマンドを実行して確認します。
  + ノード情報表示部で、起動中の系の状態が"Online"になっていること。
  + リソース情報表示で、PG-REXリソース(msPostgresql)の"Masters"に、起動中の系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(vip-master、vip-slave、vip-rep)が、起動中の系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。

|  |
| --- |
| # crm\_mon -fA1 ：（略） Last updated:日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

### メンテナンス後の系の起動

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系をSlave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、MasterとSlaveを切り替えたい場合は、『計画的な系切り替え』に従い切り替えます。

# 故障対応

本章では、クラスタシステム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クラスタ状態確認と復旧があります。

* クラスタ状態確認では、クラスタシステム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、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クラスタシステム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。

## 前提条件

この章の前提条件を以下に示します。

* 両系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

クラスタシステムに発生した故障を特定するために、crm\_mon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、制御エラー情報を確認する必要があります。

以下にcrm\_monコマンドの実行時の表示例を示します。

なお、"-1"オプションを付加してcrm\_monコマンドを実行し、"Connection to cluster failed: Transport endpoint is not connected"と表示された場合、crm\_monコマンドを実行したノードのPacemakerは停止しています。

|  |
| --- |
| #01# crm\_mon -fA1 Stack: corosync Current DC: pgrex01 (バージョン) - partition with quorum Last updated: 日付表示 ：（略）  Online: [ pgrex01 pgrex02 ]  Active resources:  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ]  Node Attributes: \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 00000000B1000098  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP \* Node pgrex02: ：（略）  Migration Summary: \* Node pgrex01: \* Node pgrex02: |

### 表示部説明

crm\_mon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 pgrex02 ] |

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、クラスタを構成しているノードが"[]"内に表示されます。

【リソース情報表示部】

|  |
| --- |
| Active resources:   vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] |

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。

■ msPostgreSQL(PG-REXリソース)

PG-REXリソースの稼働状況表示一覧

|  |  |
| --- | --- |
| crm\_mon実行結果の表示 | 稼働状況の説明 |
| Masters: [ pgrex01 ] Slaves: [ pgrex02 ] | 両系でリソースが稼働中 |
| Masters: [ pgrex01 ] | 片系(pgrex01)でリソースが稼働中 |
| 表示なし | 両系でリソースが停止中 |

■ vip-master, vip-rep, vip-slave(IPaddr2リソース), grpStonith1, grpStonith2(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

|  |  |
| --- | --- |
| crm\_mon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 |
| 表示なし | リソースが停止中 |

■ clnPing, clnDiskd1, clnDiskd2(cloneリソース)

cloneリソースの稼働状況表示一覧

|  |  |
| --- | --- |
| crm\_mon実行結果の表示 | 稼働状況の説明 |
| Started: [ pgrex01 pgrex02 ] | 両系でリソースが稼働中 |
| Started: [ pgrex01 ] | 片系(pgrex01)でリソースが稼働中 |
| 表示なし | 両系でリソースが停止中 |

【属性情報表示部】

|  |
| --- |
| Node Attributes: \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 00000000B1000098  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP \* Node pgrex02: ：（略） |

各ノードにおけるネットワーク経路、ディスクの監視状態、およびPG-REXリソース、IC-LANの状態を表示します。 IC-LANの状態についてはクラスタ内の他のノードのIC-LAN用インターフェイスからの通信状態を表示します。

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 |
| default\_ping\_set | 100 | ネットワーク経路の状況を示す |
| diskcheck\_status | normal | DB領域ディスクの監視状態を示す |
| diskcheck\_status\_internal | normal | OS領域ディスクの監視状態を示す |
| master-pgsql | 1000(pgrex01), 100(pgrex02) | PG-REXリソースの属性で、 PacemakerがMasterおよびSlave 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(Master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 |
| ringnumber\_0[[18]](#footnote-18) | 192.168.1.2 is up | IC-LANの通信状態を示す |
| 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Masterが存在しない場合 に、Masterになれるかどうかを 決定するために設定される |

【故障回数表示部】

|  |
| --- |
| Migration Summary: \* Node pgrex01: \* Node pgrex02: |

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 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 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 |

### 正常状態確認方法

crm\_monコマンド実行時の表示からクラスタシステムが正常状態であることを確認する際の項目を示します。

crm\_mon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、クラスタシステムは正常状態です。

【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(msPostgresql)、IPaddr2リソース(vip-master、vip-slave、vip-rep)が、以下のとおりに稼働していることを確認する。

* pgrex01側 ： msPostgresql(Master), vip-master, vip-rep
* pgrex02側 ： msPostgresql(Slave), vip-slave

【『リソース情報表示部』の『STONITHリソース』】

STONITHリソース(grpStonith1、grpStonith2)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『cloneリソース』】

ネットワーク経路監視用cloneリソース(clnPing)、ディスク監視用cloneリソース(clnDisk1、clnDisk2)が全て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、ディスクの監視状態、IC-LANの通信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。

## 故障箇所特定手順

本節では、クラスタシステム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

* crm\_mon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、内蔵ディスク故障、IC-LAN (一部インターフェイス通信異常)のハードウェア故障を特定します。

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/pm\_logconv.out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crm\_mon -fA1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crm\_monの実行結果のノード表示部が以下に該当する場合、『/var/log/pm\_logconv.outの確認』へ進みます。

|  |
| --- |
| Online(pgrex02) OFFLINE(pgrex01)  ※ pgrex01がOFFLINEになっている |

【D-LAN故障の特定】

pgrex01でcrm\_mon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:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : -INFINITY  + pgsql-data-status : DISCONNECT  + pgsql-status : HS:alone  + ringnumber\_0 : 192.168.1.2 is UP  + ringnumber\_1 : 192.168.3.2 is UP |

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[[19]](#footnote-19) 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/postgresql Mar 12 16:39:46 pgrex01 postgres[3174]: [33-1] ~ LOG: 00000: terminating walsender process due to replication timeout |

【ハードウェア故障・リソース故障の特定】

pgrex02のcrm\_mon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、ディスク等)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crm\_mon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 |
| default\_ping\_set | 0 | 100 | S-LAN故障(pgrex01) |
| 同上 | 100 | 0 | S-LAN故障(pgrex02) |
| 同上 | 0 | 0 | ルータ故障(pgrex01, pgrex02) |
| diskcheck\_status diskcheck\_status\_internal | ERROR | normal | 内蔵ディスク故障(pgrex01) |
| 同上 | normal | ERROR | 内蔵ディスク故障(pgrex02) |
| pgsql-data-status | LATEST | DISCONNECT | D-LAN故障 |
| pgsql-status | PRI | HS:alone | D-LAN故障 |
| ringnumber\_0[[20]](#footnote-20) | 192.168.1.1 is FAULTY[[21]](#footnote-21) | 192.168.1.1 is FAULTY[[22]](#footnote-22) | IC-LAN故障(eth1インタ フェースの通信異常) |

以下にcrm\_mon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default\_ping\_set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:  + default\_ping\_set : 0 : Connectivity is lost  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : -INFINITY  + pgsql-data-status : DISCONNECT  + pgsql-status : DISCONNECT  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP \* Node pgrex02:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 00000139210029D8  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.2 is UP  + ringnumber\_1 : 192.168.3.2 is UP ：（略） |

### 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crm\_mon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crm\_monコマンド実行時の表示例を示します。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Failed 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crm\_monコマンド実行時の表示例を示します。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Failed 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crm\_monコマンド実行時の表示例を示します。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Failed 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 |

【vip-master故障の特定】

以下にvip-master故障発生時のcrm\_monコマンド実行時の表示例を示します。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Failed Actions: \* vip-master\_monitor\_10000 on pgrex01 'unknown error' (1): call=22, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【vip-slave故障の特定】

以下にvip-slave故障発生時のcrm\_monコマンド実行時の表示例を示します。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Failed Actions: \* vip-slave\_monitor\_10000 on pgrex02 'unknown error' (1): call=16, status=complete, exitreason='', last-rc-change='日時表示', queued=XXms, exec=XXms |

【vip-rep故障の特定】

以下にvip-rep故障発生時のcrm\_monコマンド実行時の表示例を示します。

vip-rep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、"vip-rep"が1回故障したことを示しています。

|  |
| --- |
| # crm\_mon -fA -1 :（略） Migration Summary: \* Node pgrex01:   vip-rep: migration-threshold=0 fail-count=1 last-failure='日時表示' \* Node pgrex02: |

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 Actions: \* vip-master\_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/pm\_logconv.outの確認

Pacemakerが停止しているノードのログファイル(/var/log/pm\_logconv.out)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、PG-REXリソース(msPostgresql)のstop故障が発生したことを示しています。

|  |
| --- |
| # grep 'Resource.\*failed to stop' /var/log/pm\_logconv.out Jun 18 13:03:22 pgrex01 error: Resource pgrex01 failed to stop. (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 'Node.\*is lost' /var/log/pm\_logconv.out Dec 4 14:22:56 pgrex01 warning: Node pgrex02 is 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。

### 故障時のクラスタ状態

S-LANのルータ故障を検知し、両系のPG-REXリソース(msPostgresql)、IPadd2リソース(vip-master、vip-slave、vip-rep)が停止した状態となっています。

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex02が先に停止した場合はpgrex01(Master)-pgrex02(Slave)、pgrex01が先に停止した場合はpgrex01(Slave)-pgrex02(Master)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]】

crm\_mon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、PG-REX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停止中の系の停止』を参照してください。

【STEP3：Pacemaker停止 [pgrex01]】

保守者の作業中に、PG-REX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停止中の系の停止』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。

* 故障時のクラスタ状態
* 報告時点でのサービス稼働状況(サービス停止中)
* 報告時点でのクラスタ状態(全ノードで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『Master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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 ] OFFLINE: [ pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 ] ：（略） |

【STEP8：Pacemaker起動 [pgrex02]】

pgrex02のPacemakerを起動します。起動する手順は『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02 ：（略） Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 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。

### 故障時のクラスタ状態

S-LAN故障を検知し、pgrex01でPG-REXリソース(msPostgresql)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PG-REXリソースがMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。

サービスは片系(pgrex02)で継続しています。

### 復旧

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。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

保守者の作業中に、PG-REX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停止中の系の停止』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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(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PG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlaveとして起動します。

起動する手順は『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（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(vip-slave)がpgrex01へ移動します。

サービスは片系(pgrex01)で継続しています。

### 復旧

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、故障発生前のクラスタ状態に戻します。

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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ] ：（略） |

【STEP2：Pacemaker停止 [pgrex02]】

pgrex02のPacemakerを停止します。停止する手順は『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 ] OFFLINE: [ pgrex02 ] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex01でサービス継続中)
* 報告時点でのクラスタ状態(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『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略）  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 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。

### 故障時のクラスタ状態

リソース故障(monitor)を検知し、pgrex01でPG-REXリソース(msPostgresql)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PG-REXリソースがMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。

サービスは片系(pgrex02)で継続しています。

### 復旧

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

pgrex01のPacemakerを停止します。停止する手順は『PostgreSQL停止中の系の停止』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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(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PG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlaveとして起動します。

起動する手順は『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（略） |

## リソース故障(demote/stop)

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

### 故障時のクラスタ状態

リソース故障(demote/stop)を検知し、pgrex02からpgrex01へのreset処理を実施しています。

reset処理が成功した場合、フェイルオーバが発生し、pgrex02のPG-REXリソース(msPostgresql)がMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。

reset処理が失敗した場合、pgrex01の状態は不定となり、pgrex02はSlaveのままで、pgrex01に対してreset処理を繰り返しています。

reset処理が成功した場合、サービスは片系(pgrex02)で継続しています。

reset処理が失敗した場合、サービスの稼働状況は不定です。

### 復旧

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP3：保守者介在処理 [pgrex02]】

手動でpgrex01を停止させたことを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".\*error.\*"Failed to STONITH".\*reboot.\*"をキーワードとし、/var/log/pm\_logconv.outに以下のログが出力されていることを確認します。

|  |
| --- |
| Feb 15 13:38:54 pgrex02    error: Failed to STONITH (reboot) pgrex01 by pgrex02. |

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：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。

【STEP5：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(pgrex01のリソース故障(demote/stop)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 [pgrex01]】

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

※ pgrex02でPG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（略） |

## リソース故障(vip-master)

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

### 故障時のクラスタ状態

リソース故障(vip-master)を検知し、pgrex01でPG-REXリソース(msPostgresql)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PG-REXリソースがMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。

サービスは、片系(pgrex02)で継続しています。

### 復旧

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP2：Pacemaker停止 [pgrex01]】

保守者の作業中に、PG-REX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停止中の系の停止』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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略） |

【STEP4：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(IPaddr2(vip-master)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『Slaveの起動』を参照してください。

※ pgrex02でPG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（略） |

## リソース故障(vip-rep)

この節では、リソース故障(vip-rep)時の対処について説明します。

### 故障時のクラスタ状態

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

サービスは両系で継続しています。

### 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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ] ：（略） Migration Summary: \* Node pgrex01:   (故障回数表示) \* Node pgrex02: |

【STEP2：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex01でサービス継続中)
* 報告時点でのクラスタ状態(両系でPacemaker起動中)
* 故障箇所(IPaddr2(vip-rep)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(vip-rep)のフェイルカウントクリア [pgrex01]】

pgrex01のvip-repのリソースのフェイルカウントをクリアします。

|  |
| --- |
| # crm\_resource -C -r vip-rep -N pgrex01 Cleaning up vip-rep on pgrex01 Waiting for 1 replies from the CRMd. 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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略） Migration Summary: \* Node pgrex01:   (表示無し) \* Node pgrex02: |

## リソース故障(vip-slave)

この節では、リソース故障(vip-slave)時の対処について説明します。

### 故障時のクラスタ状態

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

サービスは両系で継続しています。

### 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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ] ：（略） Migration Summary: \* Node pgrex01: \* Node pgrex02:   (故障回数表示)  Failed Actions:   (vip-slave制御エラー情報表示) |

【STEP2：保守者へ報告】

以下の内容を報告します。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex01でサービス継続中)
* 報告時点でのクラスタ状態(両系でPacemaker起動中)
* 故障箇所(IPaddr2(vip-slave)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(vip-slave)の切り替え [pgrex02]】

pgrex02のvip-slaveの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。

|  |
| --- |
| # crm\_resource -C -r vip-slave -N pgrex02 Cleaning up vip-slave on pgrex02 Waiting for 1 replies from the CRMd. 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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略） Migration Summary: \* Node pgrex01: \* Node pgrex02:   (表示無し) |

## 内蔵ディスク故障

この節では、内蔵ディスク故障時の対処について説明します。

### 故障時のクラスタ状態

内蔵ディスク故障を検知し、pgrex01でPG-REXリソース(msPostgresql)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PG-REXリソースがMasterへ昇格し、pgrex01のIPaddr2リソース(vip-mster, vip-rep)がpgrex02へ移動します。

サービスは片系(pgrex02)で継続しています。

### 復旧

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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]】

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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP3：保守者介在処理 [pgrex02]】

手動でpgrex01を停止させたことを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".\*error.\*"Failed to STONITH".\*reboot.\*"をキーワードとし、/var/log/pm\_logconv.outに以下のログが出力されていることを確認します。

|  |
| --- |
| Feb 15 13:38:54 pgrex02    error: Failed to STONITH (reboot) pgrex01 by pgrex02. |

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。

* 故障時のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(pgrex01でPacemaker停止中)
* 故障箇所(pgrex01の内蔵ディスク故障が発生)

【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]】

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『Slaveの起動』を参照してください。

※ pgrex02でPG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（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、pgrex01のPacemakerの異常を検知し、pgrex02からpgrex01へのreset処理を実施してます。 reset処理が成功した場合、フェイルオーバが発生し、pgrex02のPG-REXリソース(msPostgresql)がMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。 reset処理が失敗した場合、pgrex02はSlave状態のままで、pgrex01に対してreset処理を繰り返しています。

reset処理が成功した場合、サービスは片系(pgrex02)で継続しています。 reset処理が失敗した場合、サービス稼動状況は不定です。

### 復旧

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、故障発生前のクラスタ状態に戻します。

復旧後のクラスタ状態は、pgrex01(Slave) - pgrex02(Master)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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [pgrex02 ] OFFLINE: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ] ：（略） |

【STEP3：保守者介在処理 [pgrex02]】

手動でpgrex01を停止させたことを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".\*error.\*"Failed to STONITH".\*reboot.\*"をキーワードとし、/var/log/pm\_logconv.outに以下のログが出力されている ことを確認します。

|  |
| --- |
| Feb 15 13:38:54 pgrex02    error: Failed to STONITH (reboot) pgrex01 by pgrex02. |

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。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex02でサービス継続中)
* 報告時点でのクラスタ状態(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『Slaveの起動』を参照してください。

※ pgrex02でPG-REXリソースがMasterとして稼働中のため、pgrex01のPG-REXリソースをSlave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ] ：（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、クラスタおよび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(vip-slave)がpgrex01へ移動します。
* サービスは片系(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PG-REXリソース(msPostgresql)がMasterへ昇格し、pgrex01のIPaddr2リソース(vip-master、vip-rep)がpgrex02へ移動します。
* サービスは片系(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Masterのままで、pgrex02はSlave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。
* サービスは両系で継続しています。

### 復旧

pgrex01がMasterの場合の復旧手順を以下に説明します。

pgrex02がMaster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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 ] OFFLINE: [ pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ] ：（略） |

【STEP3：保守者介在処理 [pgrex01]】

手動でpgrex02を停止させたことを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".\*error.\*"Failed to STONITH".\*reboot.\*"をキーワードとし、/var/log/pm\_logconv.outに以下のログが出力されていることを確認します。

|  |
| --- |
| Feb 15 13:38:54 pgrex01    error: Failed to STONITH (reboot) pgrex02 by pgrex01. |

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。

* 復旧開始時点のクラスタ状態
* 報告時点でのサービス稼働状況(pgrex01でサービス継続中)
* 報告時点でのクラスタ状態(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『Slave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。

|  |
| --- |
| # crm\_mon -fA -1 ：（略） Last updated: 日時表示 ：（略）  Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 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。

### 両系の起動

両系を起動するには、一方の系でMasterを起動させ、起動完了後、もう一方の系でSlaveを起動させます。どちらの系をMaster、Slaveとして稼働させるかは、ユーザが決定します。

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

### Masterの起動

本節では、Masterの起動手順を説明します。

1. どの系をMasterとして起動するか決定します。

* PG-REXでは、最新のDBデータを持つ系をMasterとして起動しなければなりません。古いDBデータを持つ系をMaster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、Masterとして起動する系を決めるときの考え方の例です。
  + DBクラスタが片系のみに存在し、そのDBクラスタを使ってPG-REXを起動する場合(初めてMasterを起動する場合を含む)は、DBクラスタが存在する系をMasterとして起動する。
  + DBクラスタが両系に存在する場合は、直前までMasterとして稼働していた系をMaster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系をMasterとして起動する。

以降の手順では、pgrex01をMaster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。



|  |
| --- |
| # systemctl status pacemaker ● pacemaker.service - Pacemaker High Availability Cluster Manager    Loaded: loaded (/etc/systemd/system/pacemaker.service; disabled; vendor preset: disabled)    Active: inactive (dead) |

1. ベースバックアップからMaster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Master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、Master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、Masterの起動の手順を行います。これにより、Master起動時のアーカイブリカバリは必要なくなるため、Master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。

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. 新しいcrmファイルを反映させてMasterを起動する場合は、既存のPacemakerの設定を削除します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # rm -f /var/lib/pacemaker/cib/\* |

1. pgrex01でMasterを起動します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # systemctl start pacemaker |

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。

|  |
| --- |
| # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by hacluster via crmd on pgrex01 ：（略）  Online: [ pgrex01 ] ：（略）  Node Attributes: \* Node pgrex01:  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP  Migration Summary: \* Node pgrex01: |

1. 初回起動時、もしくは新しいcrmファイルを反映させる場合、crmファイルを反映させます。

* pgrex01でcrm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。

|  |
| --- |
| # crm options sort-elements no # crm configure load update PG-REX11\_pm\_crmgen\_env.crm |

【注意】

コマンド実行時に以下のようなメッセージが表示されることがありますが、Pacemakerの動作に問題はありません。

|  |
| --- |
| WARNING: pgsql: specified timeout 60s for notify is smaller than the advised 90s WARNING: rsc\_location-grpStonith2-4: referenced node pgrex02 does not exist WARNING: fencing\_topology: target pgrex02 not a node WARNING: fencing\_topology: target pgrex02 not a node |

1. pgrex01でcrm\_mon -fA コマンドを実行し、Master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。

|  |
| --- |
| # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by root via crm\_attribute on pgrex01 ：（略） Online: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 ]  Node Attributes: \*Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 00000000B1000098  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP  Migration Summary: \* Node pgrex01:  ※ 一度もSlaveを起動したことがない場合、pgrex02の情報は表示されません。 |

### Slaveの起動

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

1. pgrex02で、Pacemakerが停止していることを確認します。

* 起動している場合は、Pacemakerを停止します。
* 本作業はrootユーザで行います。

|  |
| --- |
| # systemctl status pacemaker ● pacemaker.service - Pacemaker High Availability Cluster Manager    Loaded: loaded (/etc/systemd/system/pacemaker.service; disabled; vendor preset: disabled)    Active: inactive (dead) |

1. pgrex01でcrm\_mon -fA コマンドを実行し、Masterが稼働中であることを確認します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by root via crm\_attribute on pgrex01 ：（略） Online: [ pgrex01 ] ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ] ：（略） |

1. IC-LANが正常に接続されていることを確認します。

* pingコマンド等を用いて確認してください。

1. 以下のいずれかの場合に限り、Master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)に系がMasterとして稼働していたことを意味します。このような系をSlaveとして起動する場合(例えば、異常終了した旧Masterの系を、フェイルオーバ後に再組み込みする場合)は、Master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、Slaveとして稼働していた系を、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。
   * Slave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Master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。

* Master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  ※ -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MasterからSlaveに行ってください。誤ってSlaveからMaster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/  ※ 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Slaveを起動します。

* 本作業はrootユーザで行います。

|  |
| --- |
| # systemctl start pacemaker |

1. pgrex02でcrm\_mon -fA コマンドを実行し、Slave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。

|  |
| --- |
| # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by root via crm\_attribute on pgrex01 ：（略） Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex01   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: grpStonith1   prmStonith1-1 (stonith:external/stonith-helper): Started pgrex02   prmStonith1-2 (stonith:external/ipmi): Started pgrex02  Resource Group: grpStonith2   prmStonith2-1 (stonith:external/stonith-helper): Started pgrex01   prmStonith2-2 (stonith:external/ipmi): Started pgrex01  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex01 ]   Slaves: [ pgrex02 ]  Clone Set: clnPing [prmPing]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd1 [prmDiskd1]   Started: [ pgrex01 pgrex02 ]  Clone Set: clnDiskd2 [prmDiskd2]   Started: [ pgrex01 pgrex02 ]  Node Attributes: \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 00000000B1000098  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP \* Node pgrex02:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 100  + pgsql-data-status : STREAMING|SYNC  + pgsql-status : HS:sync  + ringnumber\_0 : 192.168.1.2 is UP  + ringnumber\_1 : 192.168.3.2 is UP  Migration Summary: \* Node pgrex01: \* Node pgrex02: |

### 両系の停止

両系を停止するには、Slaveを停止させ、Slaveの停止が完了後にMasterを停止させます。

Master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。

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

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

### Slaveの停止

本節では、Slave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. Slaveを停止します。



|  |
| --- |
| # systemctl stop pacemaker |

1. SlaveのPacemakerが正常に停止されたことを確認します。



|  |
| --- |
| # crm\_mon -fA Attempting connection to the cluster.... |

1. Slave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。 |

### Masterの停止

本節では、Masterの停止手順を説明します。

Slave稼働中にMaster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. Masterを停止します。



|  |
| --- |
| # systemctl stop pacemaker |

1. MasterのPacemakerが正常に停止されたことを確認します。



|  |
| --- |
| # crm\_mon -fA Attempting connection to the cluster.... |

1. Master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[[23]](#footnote-23)、タイムライン履歴ファイル[[24]](#footnote-24)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
* Slave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. Slave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、Slave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

* 両系で、(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 |

## 計画的な系切り替え

本節では、手動による計画的な系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。

### 系切り替え

手動による系切り替えの手順を以下に示します。系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、Masterをpgrex02に切り替えます。

1. フェイルオーバ確認

* pgrex02でcrm\_mon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。
  + msPostgresqlリソースが、pgrex02上でMasterとして動作していること。
  + vip-masterリソース、vip-repリソースが、pgrex02上で起動していること。

|  |
| --- |
| [pgrex02] # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by root via crm\_attribute on pgrex02 ：（略） Online: [ pgrex02 ] OFFLINE: [ pgrex01 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]  Clone Set: clnPing [prmPing] ：（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 [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  ※ -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/  ※ 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  [pgrex01] # rm -f /var/lib/pgsql/tmp/PGSQL.lock |

1. pgrex01の再組み込み

* pgrex01をSlaveとして起動します。

1. 系切り替え確認

* pgrex01でcrm\_mon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。
  + msPostgresqlリソースが、pgrex01上でSlaveとして動作していること。
  + vip-slaveリソースが、pgrex01上で起動していること。

|  |
| --- |
| [pgrex01] # crm\_mon -fA ：（略） Last updated: 日時表示 Last change: 日時表示 by root via crm\_attribute on pgrex02 ：（略） Online: [ pgrex01 pgrex02 ] ：（略）  vip-slave (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: master-group   vip-master (ocf::heartbeat:IPaddr2): Started pgrex02   vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02 ：（略）  Master/Slave Set: msPostgresql [pgsql]   Masters: [ pgrex02 ]   Slaves: [ pgrex01 ]  Clone Set: clnPing [prmPing] ：（略） Node Attributes: \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 100  + pgsql-data-status : STREAMING|SYNC  + pgsql-status : HS:sync  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP ：（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 |
| PG-REX11 | PacemakerとPostgreSQL11を組み合わせたソリューションやその 開発プロジェクト名およびそのバージョン。 PG-REX11はMaster/Slave構成から成っている。 |
| 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。 |
| Master/Slave構成 | Pacemakerがリソースの高可用性を実現するための構成の うちの一つ。PG-REXではこの構成を用いる。 |
| pgrex01(現用機) | クラスタシステムにおいて、初めから処理やサービスを稼働 させているサーバ。 |
| pgrex02(予備機) | クラスタシステムにおいて、pgrex01(現用機)の故障発生時に 処理やサービスを稼働させているサーバ。 |
| スプリットブレイン | pgrex01とpgrex02上の各Pacemakerが互いの状態を把握できなく なり、両系ともMasterになること。主にIC-LANが切断された ときに遷移される可能性がある状態。 |
| IC-LAN (インターコネクトLAN) | インターコネクトLANのこと。pgrex01とpgrex02間のネット ワーク。pgrex01とpgrex02上の各Pacemakerが互いに死活監視 するために使用される。 |
| D-LAN (DBレプリケーションLAN) | データ転送用LANのこと。pgrex01とpgrex02間のネットワーク。 PostgreSQLのレプリケーション機能で使用される。 |
| S-LAN(運用LAN) | サービス提供用LANのこと。pgrex01またはpgrex02とユーザまた はアプリケーション間のネットワーク。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される。 |
| STONITH-LAN | STONITH機能を使用するためのLANのこと。pgrex01とpgrex02間 のネットワーク。各系がスプリットブレイン状態に陥りそうな とき、 各系のPacemakerは、このLANを介して、一方の系のマシンを 強制的に停止（STONITH機能）し、スプリットブレインが発生 する のを回避する。 |
| 同期モード | MasterのWALをSlaveに書き込み後に、Masterがコミットを返す モード。 |
| 非同期モード | トランザクションのコミットのタイミングとは異なり、一定 期間でMasterのWALをSlaveに反映していくモード。 |
| Master | RAの状態を表す:PG-REXの場合、同一ノードでPostgreSQLが プライマリサーバとして稼働する。 |
| Slave | RAの状態を表す:PG-REXの場合、同一ノードでPostgreSQLが スタンバイサーバとして稼働する。 |
| Stopped | RAの状態を表す:RAが停止している状態。 |
| start | RAの状態を表す:StoppedのリソースをSlaveにすること。 |
| stop | RAの状態を表す:SlaveのリソースをStoppedにすること。 |
| promote | RAの状態を表す:SlaveのリソースをMasterにすること。 |
| demote | RAの状態を表す:MasterのリソースをSlaveにすること。 |
| monitor | RAの状態を表す:MasterおよびSlaveのリソースを一定間隔で 監視すること。 |
| 昇格 | PostgreSQLの動作を表す:SlaveからMasterにすること。 |
| 起動禁止フラグ | PacemakerがPostgreSQLをpromoteする際に作成するファイル。 (デフォルト: /var/lib/pgsql/tmp/PGSQL.lock) 本ファイルが残っていると次回PostgreSQLを起動でき なくなる。 Master停止時にSlaveが存在していない場合は、停止時に削除 される。 |
| フェイルオーバ | Master/SlaveをStopped/Masterにすること。 |
| フェイルバック | Stopped/MasterをMaster/Slaveにすること。 |
| スイッチオーバ | Master/SlaveをSlave/Masterにすること。 |
| スイッチバック | Slave/MasterをMaster/Slaveにすること。 |

1. 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-1)
2. 参照先：[<http://www.postgresql.jp/document/11/html/functions-admin.html>](http://www.postgresql.jp/document/11/html/functions-admin.html) [↑](#footnote-ref-2)
3. 参照先：[<http://www.postgresql.jp/document/11/html/functions-admin.html>](http://www.postgresql.jp/document/11/html/functions-admin.html) [↑](#footnote-ref-3)
4. 参照先：[<http://www.postgresql.jp/document/11/html/warm-standby.html#STANDBY-PLANNING>](http://www.postgresql.jp/document/11/html/warm-standby.html#STANDBY-PLANNING) [↑](#footnote-ref-4)
5. 参照先：[<http://www.postgresql.jp/document/11/html/hot-standby.html#HOT-STANDBY-CONFLICT>](http://www.postgresql.jp/document/11/html/hot-standby.html#HOT-STANDBY-CONFLICT) [↑](#footnote-ref-5)
6. その他のインストール方法についてはLinux-HA Japanプロジェクトのサイト([<http://linux-ha.osdn.jp/>](http://linux-ha.osdn.jp/))を参照してください。 [↑](#footnote-ref-6)
7. PacemakerリポジトリパッケージはLinux-HA Japanプロジェクトのサイトからダウンロードしてください。 [↑](#footnote-ref-7)
8. ipmitoolパッケージはRHELのインストールDVDに含まれており、STONITH機能で使用する。 [↑](#footnote-ref-8)
9. リストアコマンドはpm\_crmgen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-9)
10. tcp\_keepalives\_idle、tcp\_keepalives\_interval、tcp\_keepalives\_countの設定を指します。 [↑](#footnote-ref-10)
11. 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-11)
12. /usr/lib/ocf/resource.d/heartbeatまたは/usr/lib/ocf/resource.d/pacemakerは、ユーザが独自に作成、編集を行ったリソースエージェント(シェルスクリプト)等のファイルが配下に存在する場合のみ残存します。不要な場合は削除してください。 [↑](#footnote-ref-12)
13. /usr/lib/ocf/resource.d/heartbeatまたは/usr/lib/ocf/resource.d/pacemakerは、ユーザが独自に作成、編集を行ったリソースエージェント(シェルスクリプト)等のファイルが配下に存在する場合のみ残存します。不要な場合は削除してください。 [↑](#footnote-ref-13)
14. 参照先: [<http://www.postgresql.jp/document/11/html/hot-standby.html#HOT-STANDBY-ADMIN>](http://www.postgresql.jp/document/11/html/hot-standby.html#HOT-STANDBY-ADMIN) [↑](#footnote-ref-14)
15. 参照先：[<http://www.postgresql.jp/document/11/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/11/html/continuous-archiving.html#BACKUP-BASE-BACKUP) [↑](#footnote-ref-15)
16. 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/11/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/11/html/continuous-archiving.html#BACKUP-BASE-BACKUP) [↑](#footnote-ref-16)
17. 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/11/html/continuous-archiving.html#BACKUP-TIMELINES>](http://www.postgresql.jp/document/11/html/continuous-archiving.html#BACKUP-TIMELINES) [↑](#footnote-ref-17)
18. eth1のインターフェイスによるハートビート通信の場合の例を示します。 [↑](#footnote-ref-18)
19. 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-19)
20. eth1のインターフェイスによるハートビート通信の場合の例を示します。 [↑](#footnote-ref-20)
21. IC-LANの状態についてはクラスタ内の他のノードのインターフェイスからの通信状態を表示するため、現用機とpgrex02の双方で互いのeth1からの通信状態が"FAULTY"と表示されていることを確認します。 [↑](#footnote-ref-21)
22. IC-LANの状態についてはクラスタ内の他のノードのインターフェイスからの通信状態を表示するため、現用機とpgrex02の双方で互いのeth1からの通信状態が"FAULTY"と表示されていることを確認します。 [↑](#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/11/html/continuous-archiving.html#BACKUP-BASE-BACKUP>](http://www.postgresql.jp/document/11/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/11/html/continuous-archiving.html#BACKUP-TIMELINES>](http://www.postgresql.jp/document/11/html/continuous-archiving.html#BACKUP-TIMELINES) [↑](#footnote-ref-24)