**PG-REX9.4利用マニュアル**

**第1.0版**

**2014年3月19日**

**日本電信電話株式会社  
NTTオープンソースソフトウェアセンタ**

改版履歴

|  |  |  |
| --- | --- | --- |
| 日付 | 版数 | 変更内容 |
| 2015/3/19 | 1.0 | 初版制定 |

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

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

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

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。

―― 目次 ――

[1章. はじめに 1](#_Toc414608072)

[1.1. 本書の目的 1](#_Toc414608073)

[1.2. 対象読者 1](#_Toc414608074)

[1.3. 表記規則 2](#_Toc414608075)

[1.4. 前提条件 2](#_Toc414608076)

[2章. 概要 3](#_Toc414608077)

[2.1. PG-REXの概要 3](#_Toc414608078)

[2.2. システム構成 4](#_Toc414608079)

[2.3. 制約 5](#_Toc414608080)

[2.3.1. PG-REXの制約 5](#_Toc414608081)

[2.3.2. PG-REX運用補助ツールの制約 8](#_Toc414608082)

[2.4. 推奨設定 9](#_Toc414608083)

[3章. 環境構築 10](#_Toc414608084)

[3.1. ネットワーク 10](#_Toc414608085)

[3.2. ディレクトリ構成 13](#_Toc414608086)

[3.3. Pacemaker 14](#_Toc414608087)

[3.3.1. インストール 14](#_Toc414608088)

[3.3.2. クラスタの基本設定（corosync.conf） 17](#_Toc414608089)

[3.3.3. Corosync設定ファイルの確認 18](#_Toc414608090)

[3.3.4. クラスタ認証設定ファイルの生成（authkey） 18](#_Toc414608091)

[3.3.5. Pacemaker設定ファイルの追加設定（pacemaker） 18](#_Toc414608092)

[3.3.6. クラスタ起動スクリプト設定の追加設定 19](#_Toc414608093)

[3.3.7. ログの出力設定（rsyslog.conf） 20](#_Toc414608094)

[3.3.8. ログメッセージ制御機能設定（pm\_logconv.conf） 20](#_Toc414608095)

[3.3.9. ログローテーションの追加設定（pacemaker） 21](#_Toc414608096)

[3.3.10. pm\_logconvのプロセス起動 22](#_Toc414608097)

[3.3.11. ifcheckdプロセスの起動 22](#_Toc414608098)

[3.4. PostgreSQL 24](#_Toc414608099)

[3.4.1. PostgreSQLのインストール 24](#_Toc414608100)

[3.4.2. 環境変数の登録 27](#_Toc414608101)

[3.4.3. DBクラスタ用ディレクトリの作成 27](#_Toc414608102)

[3.4.4. DBクラスタの初期化 28](#_Toc414608103)

[3.4.5. postgresql.confの編集 28](#_Toc414608104)

[3.4.6. レプリケーションユーザの作成 31](#_Toc414608105)

[3.4.7. pg\_hba.confの編集 33](#_Toc414608106)

[3.4.8. .pgpassの編集 33](#_Toc414608107)

[3.5. PG-REX運用補助ツール 35](#_Toc414608108)

[3.5.1. インストール 35](#_Toc414608109)

[3.5.2. pg-rex\_tools.confの編集 36](#_Toc414608110)

[3.5.3. ネットワーク接続登録 37](#_Toc414608111)

[3.6. リソースの設定 38](#_Toc414608112)

[3.6.1. pm\_crmgenの概要 38](#_Toc414608113)

[3.6.2. クラスタ・プロパティ、リソース・デフォルトの設定 38](#_Toc414608114)

[3.6.3. リソース構成の設定 38](#_Toc414608115)

[3.6.4. リソース・パラメータの設定 38](#_Toc414608116)

[3.6.5. STONITH実行順序の設定 38](#_Toc414608117)

[3.6.6. リソース（仮想IP）の設定 39](#_Toc414608118)

[3.6.7. リソース（PostgreSQL）の設定 39](#_Toc414608119)

[3.6.8. リソース（ネットワーク監視）の設定 41](#_Toc414608120)

[3.6.9. リソース（ディスク監視）の設定 41](#_Toc414608121)

[3.6.10. リソース（STONITH）の設定 41](#_Toc414608122)

[3.6.11. リソース配置制約の設定 41](#_Toc414608123)

[3.6.12. リソース同居制約の設定 41](#_Toc414608124)

[3.6.13. リソース起動順序制約の設定 42](#_Toc414608125)

[3.6.14. リソースチケット制約の設定 42](#_Toc414608126)

[3.6.15. crmファイルの作成 43](#_Toc414608127)

[3.6.16. crmファイルの反映 46](#_Toc414608128)

[3.7. アンインストール 47](#_Toc414608129)

[4章. 起動と停止 51](#_Toc414608130)

[4.1. 両系の起動 51](#_Toc414608131)

[4.2. Masterの起動 51](#_Toc414608132)

[4.3. Slaveの起動 54](#_Toc414608133)

[4.4. 両系の停止 56](#_Toc414608134)

[4.5. Slaveの停止 56](#_Toc414608135)

[4.6. Masterの停止 56](#_Toc414608136)

[4.7. PostgreSQL停止中の系の停止 57](#_Toc414608137)

[5章. 故障対応 58](#_Toc414608138)

[5.1. 前提条件 58](#_Toc414608139)

[5.2. クラスタ状態定義 59](#_Toc414608140)

[5.3. 監視コマンド表示確認方法 59](#_Toc414608141)

[5.3.1. 表示部説明 61](#_Toc414608142)

[5.3.2. 正常状態確認方法 64](#_Toc414608143)

[5.4. 故障箇所特定手順 65](#_Toc414608144)

[5.4.1. ノード情報・リソース情報・故障回数表示部の確認 66](#_Toc414608145)

[5.4.2. 属性情報表示部の確認 69](#_Toc414608146)

[5.4.3. 制御エラー情報表示部の確認 71](#_Toc414608147)

[5.4.4. /var/log/pm\_logconv.outの確認 74](#_Toc414608148)

[5.5. 故障一覧 76](#_Toc414608149)

[5.6. ルータ故障 77](#_Toc414608150)

[5.6.1. 故障時のクラスタ状態 77](#_Toc414608151)

[5.6.2. 復旧 79](#_Toc414608152)

[5.7. S-LAN故障 83](#_Toc414608153)

[5.7.1. 故障時のクラスタ状態 83](#_Toc414608154)

[5.7.2. 復旧 85](#_Toc414608155)

[5.8. D-LAN故障 89](#_Toc414608156)

[5.8.1. 故障時のクラスタ状態 89](#_Toc414608157)

[5.8.2. 復旧 91](#_Toc414608158)

[5.9. リソース故障(start/promote) 94](#_Toc414608159)

[5.9.1. 故障時のクラスタ状態 94](#_Toc414608160)

[5.9.2. 復旧 95](#_Toc414608161)

[5.10. リソース故障(monitor) 99](#_Toc414608162)

[5.10.1. 故障時のクラスタ状態 99](#_Toc414608163)

[5.10.2. 復旧 101](#_Toc414608164)

[5.11. リソース故障(demote/stop) 105](#_Toc414608165)

[5.11.1. 故障時のクラスタ状態 105](#_Toc414608166)

[5.11.2. 復旧 108](#_Toc414608167)

[5.12. リソース故障(vip-master) 112](#_Toc414608168)

[5.12.1. 故障時のクラスタ状態 112](#_Toc414608169)

[5.12.2. 復旧 114](#_Toc414608170)

[5.13. リソース故障(vip-rep) 118](#_Toc414608171)

[5.13.1. 故障時のクラスタ状態 118](#_Toc414608172)

[5.13.2. 復旧 120](#_Toc414608173)

[5.14. リソース故障(vip-slave) 123](#_Toc414608174)

[5.14.1. 故障時のクラスタ状態 123](#_Toc414608175)

[5.14.2. 復旧 125](#_Toc414608176)

[5.15. 内蔵ディスク故障 128](#_Toc414608177)

[5.15.1. 故障時のクラスタ状態 128](#_Toc414608178)

[5.15.2. 復旧 132](#_Toc414608179)

[5.16. ノード故障 136](#_Toc414608180)

[5.16.1. 故障時のクラスタ状態 136](#_Toc414608181)

[5.16.2. 復旧 139](#_Toc414608182)

[5.17. IC-LAN故障 143](#_Toc414608183)

[5.17.1. 故障時のクラスタ状態 143](#_Toc414608184)

[5.17.2. 復旧 146](#_Toc414608185)

[6章. メンテナンス時の対応 152](#_Toc414608186)

[6.1. PostgreSQLのバックアップ 152](#_Toc414608187)

[6.1.1. 前提条件 152](#_Toc414608188)

[6.1.2. PostgreSQLのバックアップ 152](#_Toc414608189)

[6.2. アーカイブログの削除 153](#_Toc414608190)

[6.2.1. 前提条件 153](#_Toc414608191)

[6.2.2. PostgreSQLアーカイブログの削除 153](#_Toc414608192)

[6.3. 計画的な系切り替え 156](#_Toc414608193)

[6.3.1. 前提条件 156](#_Toc414608194)

[6.3.2. 系切り替え 156](#_Toc414608195)

[6.4. 計画的なメンテナンス 159](#_Toc414608196)

[6.4.1. 前提条件 159](#_Toc414608197)

[6.4.2. Masterのメンテナンス 159](#_Toc414608198)

[6.4.3. Slaveのメンテナンス 160](#_Toc414608199)

[付録A 起動と停止 161](#_Toc414608200)

[A.1. 両系の起動 161](#_Toc414608201)

[A.2. Masterの起動 161](#_Toc414608202)

[A.3. Slaveの起動 168](#_Toc414608203)

[A.4. 両系の停止 173](#_Toc414608204)

[A.5. Slaveの停止 173](#_Toc414608205)

[A.6. Masterの停止 173](#_Toc414608206)

[付録B アーカイブログの削除 175](#_Toc414608207)

[B.1. 前提条件 175](#_Toc414608208)

[B.2. PostgreSQLアーカイブログの削除 175](#_Toc414608209)

[付録C 計画的な系切り替え 178](#_Toc414608210)

[C.1. 前提条件 178](#_Toc414608211)

[C.2. 系切り替え 178](#_Toc414608212)

[付録D 運用補助ツール概要 182](#_Toc414608213)

[D.1. ツール概要 182](#_Toc414608214)

[D.2. 環境設定ファイルについて 183](#_Toc414608215)

[D.3. 各コマンド機能概要 188](#_Toc414608216)

[D.4. FAQ 198](#_Toc414608217)

[用語集 199](#_Toc414608218)

# はじめに

## 本書の目的

本書は、PostgreSQL-REX9.4（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-pcs/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.sourceforge.jp/> (Linux-HA Japanプロジェクトのサイト)

http://sourceforge.jp/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 9.4付属ドキュメント

http://www.postgresql.jp/document/9.4/

⇒本書では、『PostgreSQLドキュメント』と呼びます[[1]](#footnote-1)。

## 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。

表 1‑1　表記規則

|  |  |  |
| --- | --- | --- |
| 表記 | 説明 | 表記例 |
| 『』 | 本書内で参照する書名  （本書の章見出し等も含む） | 『PostgreSQLドキュメント』 |
| *斜体* | 環境に応じて変更する項目 | *eth1* |
| <ver> | パッケージのバージョン | postgresql94-<ver>-1PGDG.rhel6.x86\_64.rpm |
| # | rootユーザの操作記号 | # crm\_mon |
| $ | 一般ユーザの操作記号 | $ psql |
| **黒太字** | 端末操作においてユーザが編集すべき箇所 | **pacemaker on** |
| **赤太字** | 端末操作においてユーザが確認すべき箇所 | **PGDATA=/dbfp/pgdata/data** |

## 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が、LANG=Cの場合は英語で出力します。

本書では、PG-REXの動作環境を構築する2つのサーバをpg-rex01、pg-rex02と呼びます。それぞれのサーバのホスト名（uname -nの結果）はpg-rex01、pg-rex02とします。実際の環境に応じて、ホスト名は適宜読み替えてください。

# 概要

## PG-REXの概要

PG-REXは、PostgreSQL9.4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、PostgreSQL9.4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の両系にトランザクションが存在することを保証します。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

本書で前提とする、ハードウェアとソフトウェアの構成例を以下に示します。

表 2‑1　ハードウェア構成例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | 用途 | 数量 | ベンダ | ハードウェア | | |
| ① | DBサーバ | 2 | HP | DL360 G7 | CPU | Xeon E5640 2.66GHz (1P/4C) |
| Memory | 18GB |
| HDD | 146GB(15k)×4 <RAID1+0> |
| 電源 | 2 |
| NIC | 8ポート |

表 2‑2　ソフトウェア構成例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | 用途 | 項目 | ベンダ | 製品名/エディション | バージョン |
| ① | 共通 | OS | RedHat | RedHat Enterprise Linux | 6.5 |
| ② | DBサーバ | DBMS | OSS | PostgreSQL | 9.4 |
| クラスタリング | OSS | Pacemaker リポジトリパッケージ | 1.1.12-1.1 |

## 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『3.4.6 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。

表 2‑3　動作環境の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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） * IC-LAN用とデータ転送用（以下D-LAN）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。 |
| 3 | PG-REXで制御可能なサーバ数は2台である。 |
| 4 | RAIDはWriteBack設定を行うこと。 |

表 2‑4　運用中の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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。 |
| 4 | 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。 |
| 5 | 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。 |
| 6 | SlaveでSQLを実行するとき、ハッシュインデックスを利用できない[[5]](#footnote-5)。フェイルオーバ後、ハッシュインデックスを新Masterで利用するには、REINDEXを実行する必要がある。 |
| 7 | 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ていない可能性がある。これは、SlaveがpromoteすることでタイムラインIDが切り替わるが、切り替わり後のWALファイルをpg\_basebackupが受信しない場合があるためである。 |

表 2‑5　フェイルオーバ動作時の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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[[6]](#footnote-6)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。 |

表 2‑6　故障対応時の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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pg-rex02が単独でMasterとして稼働しているときに、旧Masterのpg-rex01をSlaveとして組み込みしても、自動的にpg-rex01がMaster、pg-rex02がSlaveに切り替わることはない。 |

表 2‑7　メンテナンス時の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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。

表 2‑8　PG-REX運用補助ツールの制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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、『5章 故障対応』を参照して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。 |

## 推奨設定

本節では、PG-REX運用時の推奨設定を示します。

表 2‑9　運用時の推奨設定

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 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ァイルオーバ防止や、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図 3‑1に示します。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。本書では表 3‑1のIPアドレスを使用します。

表 3‑1　ネットワーク構成例

|  |  |  |  |
| --- | --- | --- | --- |
| LAN | NIC | サーバ | IPアドレス |
| S-LAN(bonding) | bond0(eth0,eth5) | pg-rex01 | 192.168.0.11 |
| pg-rex02 | 192.168.0.12 |
| ― | ネットワーク監視先（ping） | 192.168.0.254 |
| D-LAN(bonding) | bond1(eth2,eth4) | pg-rex01 | 192.168.2.1 |
| pg-rex02 | 192.168.2.2 |
| IC-LAN1 | eth1 | pg-rex01 | 192.168.1.1 |
| pg-rex02 | 192.168.1.2 |
| IC-LAN2 | eth3 | pg-rex01 | 192.168.3.1 |
| pg-rex02 | 192.168.3.2 |
| STONITH-LAN | eth6 | pg-rex01 | 172.20.144.41 |
| pg-rex02 | 172.20.144.42 |
| iLO3 | pg-rex01 | 172.20.144.43 |
| pg-rex02 | 172.20.144.44 |

※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。

PING監視先(GW等)

内蔵ディスク

HW制御ボード

eth5

eth0

bond0

eth0

eth5

bond0

PostgreSQL

PostgreSQL

内蔵ディスク

eth2

eth4

bond1

eth2

eth4

bond1

eth6

eth6

HW制御ボード

eth1

eth1

eth3

eth3

Pacemaker

Pacemaker

pg-rex01

pg-rex02

D-LAN

STONITH-LAN

IC-LAN

S-LAN

bonding

bonding

bonding

アーカイブ

アーカイブ

図 3‑1　ネットワーク構成例

※図 3‑1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Master側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アドレスを表 3‑2のように設定することを前提とします。

表 3‑2　仮想IPアドレス

|  |  |  |  |
| --- | --- | --- | --- |
| 用途 | LAN | NIC | 仮想IPアドレス |
| Master側接続用 | S-LAN(bonding) | bond0(eth0,eth5) | 192.168.0.10 |
| Slave側接続用 | 192.168.0.20 |
| Master側レプリケーション受付用 | D-LAN(bonding) | bond1(eth2,eth4) | 192.168.2.3 |

PG-REX（pg-rex01およびpg-rex02）とユーザまたはアプリケーション間や両系間でファイアウォールを有効にする場合、以下のポート、プロトコルを許可する必要があります。

表 3‑3　ファイアウォールの設定

|  |  |  |
| --- | --- | --- |
| 用途 | ポート番号 | プロトコル |
| S-LAN | 5432（※1） | tcp |
| D-LAN | 5432（※1） | tcp |
| 22 | ssh |
| IC-LAN | 5405（※2） | udp |
| STONITH-LAN | 623（ipmiの場合[[7]](#footnote-7)） | udp |
| PING監視用 | － | icmp |

1. PostgreSQLパラメータportの設定値を指定します。
2. 『3.3.2 クラスタの基本設定（corosync.conf）』で設定するパラメータmcastportの設定値を指定します。

## ディレクトリ構成

PG-REXのディレクトリ構成例を図 3‑2に示します。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（pg-rex01とpg-rex02）のディレクトリ構成は同一にしてください。

本書では、図 3‑2のディレクトリ構成を前提に記述します。

|  |
| --- |
| /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  └-pgxlog  └-pg\_xlog …………… PostgreSQLのWAL格納先  └-pgarch  └-arc1 ………………… PostgreSQLのアーカイブログ格納先 |

図 3‑2　ディレクトリ構成例

## 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。

### インストール

『インストール手順書 [Pacemaker]』を参考にpg-rex01とpg-rex02へPacemakerをインストールします[[8]](#footnote-8)。

1. mountコマンドで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  # df  Filesystem 1K-blocks Used Available Use% Mounted on  (略)  /dev/sr0 3762278 3762278 0 100% /media  **↑正常にマウントされたことを確認する** |

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。以下の通りに、/etc/yum.repos.d配下にrheldvd.repoを新規に作成します。

|  |
| --- |
| # vi /etc/yum.repos.d/rheldvd.repo  [rhel-server]  name=Red Hat Enterprise Linux $releasever - $basearch - server  baseurl=file:///media/Server  enabled=1  gpgcheck=1  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

1. rpmコマンドを実行し、Pacemakerリポジトリパッケージ[[9]](#footnote-9)を/opt/linux-ha配下に展開します。

|  |
| --- |
| # rpm –ivh pacemaker-repo-<ver>.el6.x86\_64.rpm |

1. yumのキャッシュをクリアします。

|  |
| --- |
| # yum clean all  (略)  Cleaning up Everything |

1. Pacemakerおよびipmitool[[10]](#footnote-10)をインストールします。

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

以下のメッセージが出力されます。

|  |
| --- |
| Loaded plugins: product-id, refresh-packagekit, security, subscription-manager  This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.  Repository rhel-server is listed more than once in the configuration  linux-ha-ja-pacemaker | 2.9 kB 00:00 ...  linux-ha-ja-pacemaker/primary\_db | 23 kB 00:00 ...  rhel-server | 3.9 kB 00:00 ...  rhel-server/primary\_db | 3.1 MB 00:00 ...  Setting up Install Process  Resolving Dependencies  --> Running transaction check：  (略)  Complete! |

1. 以下のコマンドで全てのパッケージがインストールされていることを確認します。

|  |
| --- |
| # 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  **pacemaker-libs-<ver>**  **pacemaker-cli-<ver>**  **pacemaker-cluster-libs-<ver>**  **pacemaker-<ver>**  **corosynclib-<ver>**  **corosync-<ver>**  **cluster-glue-<ver>**  **cluster-glue-libs-<ver>**  **resource-agents-<ver>**  **crmsh-<ver>**  **libqb-<ver>**  **pm\_diskd-<ver>**  **pm\_extras-<ver>**  **pm\_crmgen-<ver>**  **pm\_logconv-cs-<ver>**  **ipmitool-<ver>** |

1. /etc/yum.repos.d/rheldvd.repoを無効化します。当該ファイルを削除するか、内容を以下のように変更してください。

・/etc/yum.repos.d/rheldvd.repoの「enabled」の値を「1」から「0」に修正

|  |
| --- |
| # vi /etc/yum.repos.d/rheldvd.repo  #  [rhel-server]  name=Red Hat Enterprise Linux $releasever - $basearch - server  baseurl=file:///media/Server  enabled=**0** **← 「1」から「0」に修正**  gpgcheck=1  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

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

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

### クラスタの基本設定（corosync.conf）

pg-rex01とpg-rex02でクラスタの基本設定ファイル（corosync.conf）を、『Pacemaker関連サイト』を参考に作成します。

本作業はrootユーザで行います。

|  |
| --- |
| # vi /etc/corosync/corosync.conf **←新規作成**  **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: 239.255.1.1**  **mcastport: 5405**  **}**  **}**  **logging {**  **syslog\_facility: local1**  **debug: off**  **}**  **quorum {**  **provider: corosync\_votequorum**  **expected\_votes: 2**  **}** |

（※1） 「bindnetaddr」には、IC-LANのネットワークアドレスを設定します。

（※2） 「syslog\_facility」は、『3.3.7 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。

### Corosync設定ファイルの確認

pg-rex01とpg-rex02で、作成したCorosyncの設定ファイルを確認します。

本作業はrootユーザで行います。

|  |
| --- |
| # cd /etc/corosync/  # ls –l corosync.conf  **-rw-r--r-- 1 root root** 211 9月 17 11:50 **corosync.conf** |

※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 corosync.conf  # chmod 644 corosync.conf |

### クラスタ認証設定ファイルの生成（authkey）

pg-rex01とpg-rex02にクラスタ認証設定ファイルを生成します。

本作業はrootユーザで行います。

1. pg-rex01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、pg-rex02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）

pg-rex01とpg-rex02で、Pacemaker設定ファイル（pacemaker）に設定を追加します。

本作業はrootユーザで行います。

|  |
| --- |
| # vi /etc/sysconfig/pacemaker  (略)  **# 以下5行を追加**  **export PCMK\_logfile=none**  **export PCMK\_logfacility=local1**  **export PCMK\_logpriority=info**  **export PCMK\_fail\_fast=yes**  **export HA\_LOGFACILITY=local1** |

### クラスタ起動スクリプト設定の追加設定

pg-rex01とpg-rex02で、クラスタ起動スクリプト（pacemaker.combined.conf）に設定を追加します。

本作業はrootユーザで行います。

1. ハードウェアwatchdogを使用している環境ではsoftdogモジュールのloadが失敗し、クラスタの起動が失敗してしまうため、クラスタ起動スクリプトに下記の修正を行います。

|  |
| --- |
| # vi /etc/init/pacemaker.combined.conf  (略)  **# modprobe softdog soft\_margin=60 # ←コメントアウト**  **[ -c /dev/watchdog ] || modprobe softdog soft\_margin=60 # ←追記** |

1. corosyncプロセスが故障した場合にcorosyncのwatchdogを効かせるために、クラスタ定義スクリプトに下記の修正を行います。

|  |
| --- |
| # vi /etc/init/pacemaker.combined.conf  (略)  # if you use watchdog of corosync, uncomment the line below.  **pidof corosync || false # ←コメントアウトを外す** |

1. runレベルが変更された時にupstart経由で停止が実行されるように、クラスタ定義スクリプトに下記の追加を行います。

|  |
| --- |
| # vi /etc/init/pacemaker.combined.conf  (略)  # Starts Corosync cluster engine and Pacemaker cluster manager.  **stop on runlevel [0123456]** **# ←1行追加**  kill timeout 3600  respawn |

### ログの出力設定（rsyslog.conf）

『Pacemaker関連サイト』を参考にpg-rex01とpg-rex02でPacemakerのログ出力先を設定します。

本作業はrootユーザで行います。

1. ログの出力先を/etc/rsyslog.confに設定します。

|  |
| --- |
| # vi /etc/rsyslog.conf  (略)  \*.info;mail.none;authpriv.none;cron.none;**local1.none** /var/log/messages **# ↑この部分を追記**  (略)  # Save boot messages also to boot.log  local7.\* /var/log/boot.log  **local1.info /var/log/ha-log;RSYSLOG\_TraditionalFileFormat** **# 追記** |

1. 設定を反映するため、rsyslogを再起動します。

|  |
| --- |
| # service rsyslog restart |

1. (1)の設定が反映できたことを確認します。以下の確認したファイルにPacemakerのログが出力されることになります。

|  |
| --- |
| # ls -ld /var/log/ha-log  -**rw**------- 1 **root root** 0 1月 29 15:02 2014 **/var/log/ha-log** |

### 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pg-rex01とpg-rex02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をコピーして作成し、基本設定部（[Setting]セクション）のみを編集します。

本作業はrootユーザで行います。

|  |
| --- |
| # cp /etc/pm\_logconv.conf.sample /etc/pm\_logconv.conf  # vi /etc/pm\_logconv.conf  （略）  [Settings]  #ha\_log\_path = /var/log/ha-log  #output\_path = /var/log/pm\_logconv.out  #syslogformat = True  #reset\_interval = 60  **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  **attribute\_diskd\_inner = not\_defined diskcheck\_status\_internal or diskcheck\_status\_internal eq ERROR** **# 内蔵ディスク監視を行う場合の設定**  #logconv\_logfacility = daemon  **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（pacemaker）

pg-rex01とpg-rex02のログファイルのローテーションの設定を行ないます。

本作業はrootユーザで行います。

|  |
| --- |
| # vi /etc/logrotate.d/pacemaker  **# 以下、pacemaker.logに関する設定をコメントアウト**  **#/var/log/pacemaker.log {**  **# compress**  **# dateext**  **# weekly**  **# rotate 99**  **# maxage 365**  **# notifempty**  **# missingok**  **# copytruncate**  **#}**  **# 以下、ha-log、pm\_logconv.outに関する設定を追記**  **/var/log/ha-log {**  **missingok**  **}**  **/var/log/pm\_logconv.out {**  **missingok**  **}** |

### pm\_logconvのプロセス起動

pg-rex01とpg-rex02でpm\_logconvプロセスを起動します。

本作業はrootユーザで行います。

1. pm\_logconvプロセスが停止していることを確認します。

|  |
| --- |
| # initctl list | grep pm\_logconv\_init  **pm\_logconv\_init stop/waiting** |

* pm\_logconvのプロセスが既に起動している場合は、pm\_logconv.confの設定変更を反映するために、一度停止して(2)以降の手順を実施するか、以下のコマンドで再起動してください。

|  |
| --- |
| # initctl restart pm\_logconv\_init  **pm\_logconv\_init start/running, process *6124*** |

1. pm\_logconvプロセスを起動します。

|  |
| --- |
| # initctl start pm\_logconv\_init  **pm\_logconv\_init start/running, process *6124*** |

1. pm\_logconvプロセスが起動していることを確認します。

|  |
| --- |
| # initctl list | grep pm\_logconv\_init  **pm\_logconv\_init start/running, process *6124*** |

### ifcheckdプロセスの起動

pg-rex01とpg-rex02でifcheckdプロセスを起動します。

本作業はrootユーザで行います。

1. ifcheckdプロセスが停止していることを確認します。

|  |
| --- |
| # initctl list | grep ifcheckd  **ifcheckd stop/waiting** |

* ifcheckdのプロセスが既に起動している場合は、/etc/sysconfig/pacemakerの設定変更を反映するために、一度停止して(2)以降の手順を実施するか、以下のコマンドで再起動してください。

|  |
| --- |
| # initctl restart ifcheckd  **ifcheckd start/running, process *28606*** |

1. ifcheckdプロセスを起動します。

|  |
| --- |
| # initctl start ifcheckd  **ifcheckd start/running, process *28606*** |

1. ifcheckdプロセスが起動していることを確認します。

|  |
| --- |
| # initctl list | grep ifcheckd  **ifcheckd start/running, process *28606*** |

## 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pg-rex01とpg-rex02へPostgreSQLをインストールします。PG-REXで使用するPostgreSQLのバージョンは9.4でなければなりません。

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パッケージは表 3‑4になります。

表 3‑4　インストールするRPMパッケージ

|  |  |
| --- | --- |
| 項番 | パッケージ名 |
| 1 | postgresql94-libs-<ver>-1PGDG.rhel6.x86\_64.rpm |
| 2 | postgresql94-<ver>-1PGDG.rhel6.x86\_64.rpm |
| 3 | postgresql94-server-<ver>-1PGDG.rhel6.x86\_64.rpm |
| 4 | postgresql94-contrib-<ver>-1PGDG.rhel6.x86\_64.rpm |
| 5 | postgresql94-docs-<ver>-1PGDG.rhel6.x86\_64.rpm |

* 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をインストールする前に、対象データベースサーバに古いバージョンのPostgreSQLが既にインストールされていないかを確認します。

|  |
| --- |
| # rpm -qa | grep postgresql  **postgresql-8.4.7-2.el6.x86\_64**  **postgresql-libs-8.4.7-2.el6.x86\_64**  **postgresql-devel-8.4.7-2.el6.x86\_64** |

上記の結果、古いバージョンのPostgreSQLが既にインストールされている場合はアン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。ただし、インストールの順番を間違えると、依存関係にあるパッケージがインストールできないことがあります。

|  |
| --- |
| # rpm -ivh postgresql94-libs-<ver>-1PGDG.rhel6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:postgresql94-libs ########################################### [100%]**  # rpm -ivh postgresql94-<ver>-1PGDG.rhel6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:postgresql94 ########################################### [100%]**  # rpm -ivh postgresql94-server-<ver>-1PGDG.rhel6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:postgresql94-server ########################################### [100%]**  # rpm -ivh postgresql94-docs-<ver>-1PGDG.rhel6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:postgresql94-docs ########################################### [100%]**  # rpm -ivh postgresql94-contrib-<ver>-1PGDG.rhel6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:postgresql94-contrib ########################################### [100%]** |

PostgreSQLのRPMパッケージをインストールすると、表 3‑5に示すOSのユーザと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。

また、/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。

表 3‑5　インストール時に作成されるOSのユーザとグループ

|  |  |
| --- | --- |
| ユーザ名 | postgres |
| グループ名 | postgres |

|  |
| --- |
| /usr  └-pgsql-9.4 …………………… PostgreSQLのインストール先 |

図 3‑3インストール後のディレクトリ構成

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  **postgresql94-server-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-contrib-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-docs-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-libs-<ver>-1PGDG.rhel6.x86\_64** |

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) 所属グループ=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 **← postgresというグループに 26 というidを付与する**  # usermod -u 26 postgres **← postgresユーザのuidを 26 に変更**  # usermod -g 26 postgres **← postgresユーザのgidを 26に変更**  # id postgres  **uid=26(postgres) gid=26(postgres) 所属グループ=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 **← postgresというユーザにパスワードを設定する**  ユーザー postgres のパスワードを変更。  新しいパスワード: **← パスワードを入力する**  新しいパスワードを再入力してください: **← 再度パスワードを入力する**  **passwd: 全ての認証トークンが正しく更新できました。** |

### 環境変数の登録

pg-rex01とpg-rex02で/var/lib/pgsql/.bash\_profileに表 3‑6の環境変数を設定します。

本作業はpostgresユーザで行います。

表 3‑6　環境変数の設定項目

|  |  |  |
| --- | --- | --- |
| 設定項目 | 説明 | 設定値 |
| PATH | PostgreSQLコマンドへのパス | /usr/pgsql-9.4/bin:$PATH |
| PGDATA | DBクラスタのパス | （例）/dbfp/pgdata/data |

* pg-rex01とpg-rex02で同じ設定値を使用してください。

|  |
| --- |
| $ vi ~/.bash\_profile  (略)  **export PATH=/usr/pgsql-9.4/bin:$PATH**  **export PGDATA=/dbfp/pgdata/data**  $ source ~/.bash\_profile |

以下のコマンドにて環境変数が設定されていることを確認します。

|  |
| --- |
| $ env  (略)  **PGDATA=/dbfp/pgdata/data**  (略)  **PATH=/usr/pgsql-9.4/bin:**… |

### DBクラスタ用ディレクトリの作成

pg-rex01とpg-rex02で、DBクラスタを格納するディレクトリ（この例では/dbfp/pgdata）とWALを格納するディレクトリ（この例では/dbfp/pgxlog）、アーカイブログを格納するディレクトリ（この例では/dbfp/pgarch/arc1）を作成します。各ディレクトリの所有者はpostgresでなければなりません。

本作業はrootユーザで行います。

|  |
| --- |
| # mkdir -p /dbfp/pgdata  # mkdir -p /dbfp/pgxlog  # mkdir -p /dbfp/pgarch/arc1  # chown -R postgres:postgres /dbfp |

### DBクラスタの初期化

pg-rex01で、postgresユーザにてDBクラスタを初期化します。

|  |
| --- |
| $ initdb -D /dbfp/pgdata/data -X /dbfp/pgxlog/pg\_xlog --encoding=UTF-8 --no-locale --data-checksums  (略)  **成功しました。以下を使用してデータベースサーバを起動することができます。**  **postmaster -D /dbfp/pgdata/data**  または  **pg\_ctl -D /dbfp/pgdata/data -l logfile start** |

* 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ます。

pg-rex01で、postgresユーザにてDBクラスタディレクトリとWALディレクトリのオーナー（postgres）とパーミッション（700）が正しく設定されていることを確認します。

|  |
| --- |
| $ ls -ld /dbfp/pgdata/data /dbfp/pgxlog/pg\_xlog  **drwx**------ 14 **postgres** postgres 4096 1月 30 09:03 2014 /dbfp/pgdata/data  **drwx**------ 3 **postgres** postgres 4096 1月 30 09:03 2014 /dbfp/pgxlog/pg\_xlog |

### postgresql.confの編集

pg-rex01でpostgresql.confを作成します。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、注意すべき設定は、表 3‑7のとおりです。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。

表 3‑7　PG-REXにおけるpostgresql.confの設定

|  |  |  |  |
| --- | --- | --- | --- |
| パラメータ名 | 推奨値 | 設定範囲[[11]](#footnote-11) | 備考 |
| listen\_addresses | \* | 推奨値固定 | PG-REXでは、接続を待ち受けるアドレスを個別に設定しない。 |
| wal\_level | hot\_standby | 推奨値固定 | ― |
| max\_wal\_senders | 4 | 1以上 | 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4程度を指定することを推奨する。 |
| wal\_keep\_segments | 32 | 0以上 | 推奨値から変更する場合は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 | on | 推奨値固定 | ― |
| max\_standby\_streaming\_delay | -1 | 推奨値固定 | 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 | -1 | 推奨値固定 | 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 | '' | 推奨値固定 | PG-REX RAが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 | on | onまたは remote\_write | onを指定した場合、remote\_write指定時に比べレプリケーションの信頼性が向上する。remote\_writeを指定した場合、on指定時に比べトランザクションの応答時間が短縮される。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。 |
| archive\_mode | on | 推奨値固定 | ― |
| archive\_command | '/bin/cp %p /dbfp/pgarch/arc1/%f' | WALファイルをアーカイブするための任意のコマンド | （参考）  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[[12]](#footnote-12)にもgzipを使用すること。 |
| restart\_after\_crash | off | 推奨値固定 | 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 | 20s | 0より大きい値 | 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。Keepaliveの設定[[13]](#footnote-13)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。ただし、設定値が小さすぎると、誤検知によりSlaveを誤って切り離すなど可用性が下がる。このため、設定値は事前検証等をして注意して決めること[[14]](#footnote-14)。 |
| wal\_receiver\_status\_interval | 5s | 0より大きく、wal\_sender\_timeoutより小さい値 | ― |
| hot\_standby\_feedback | on | 推奨値固定 | ― |

設定例を以下に示します。

|  |
| --- |
| $ vi $PGDATA/postgresql.conf  **listen\_addresses = '\*'**  **wal\_level = hot\_standby**  **max\_wal\_senders = 4**  **wal\_keep\_segments = 32**  **hot\_standby = on**  **max\_standby\_streaming\_delay = -1**  **max\_standby\_archive\_delay = -1**  **archive\_mode = on**  **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\_status\_interval = 5s**  **hot\_standby\_feedback = on** |

### レプリケーションユーザの作成

pg-rex01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。

本書では、表 3‑8のレプリケーションユーザを作成します。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。

表 3‑8　レプリケーションユーザ

|  |  |
| --- | --- |
| データベースユーザ名 | repuser |
| パスワード | reppasswd |

PostgreSQLを一度起動します。

|  |
| --- |
| $ pg\_ctl -w 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** |

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** |

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 | メンバー  **----------+------------------+----------**  **repuser | レプリケーション | {}** |

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

pg-rex01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表 3‑9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。

表 3‑9　PG-REXにおけるpg\_hba.confの追加設定

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 設定値 | 備考 |
| TYPE | host | ― |
| DATABASE | replication | ― |
| USER | repuser | 『3.4.7 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 | 192.168.2.1/32  192.168.2.2/32 | **各系のD-LANのIPアドレス**を設定する。 |
| METHOD | md5 | ― |

|  |
| --- |
| $ vi $PGDATA/pg\_hba.conf  **host replication repuser *192.168.2.1/32* md5**  **host replication repuser *192.168.2.2/32* md5** |

### .pgpassの編集

pg-rex01とpg-rex02それぞれで.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。

postgresユーザの$HOME配下に.pgpassファイルを新規作成し、編集します。

PG-REXを構成するのに必要な設定は表 3‑10のとおりで、レプリケーションの接続についてパスワードを設定しなければなりません。設定必須とされているものは、必ず.pgpass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。

表 3‑10　PG-REXにおける.pgpassの設定

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 設定値 | 備考 |
| hostname | 192.168.2.3 | **Master側レプリケーション受付用の仮想IPアドレス**を指定する。 |
| port | 5432 | **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 | ― |
| username | repuser | 『3.4.7 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 | reppasswd | 『3.4.7 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。 |

|  |
| --- |
| $ vi ~/.pgpass  ***192.168.2.3:5432*:replication:repuser:reppasswd**  $ chmod 600 ~/.pgpass |

## 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。

### インストール

pg-rex01とpg-rex02へPG-REX運用補助ツールをインストールします。

本作業はrootユーザで行います。

PG-REX運用補助ツールのインストール必須のRPMパッケージは表 3‑11になります。

表 3‑11　インストールするRPMパッケージ

|  |  |
| --- | --- |
| 項番 | パッケージ名 |
| 1 | pg-rex\_operation\_tools\_script-<ver>.el6.noarch.rpm |
| 2 | IO\_Tty-<ver>.el6.x86\_64.rpm |
| 3 | Net\_OpenSSH-<ver>.el6.x86\_64.rpm |

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。ただし、インストールの順番を間違えると、依存関係にあるパッケージがインストールできないことがあります。

|  |
| --- |
| # rpm -ivh Net\_OpenSSH-<ver>.el6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:Net\_OpenSSH ########################################### [100%]**  # rpm -ivh IO\_Tty-<ver>.el6.x86\_64.rpm  **準備中... ########################################### [100%]**  **1:IO\_Tty ########################################### [100%]**  # rpm -ivh pg-rex\_operation\_tools\_script-<ver>.el6.noarch.rpm  **準備中... ########################################### [100%]**  **1:pg-rex\_operation\_tools\_########################################### [100%]** |

PG-REX運用補助ツールのRPMパッケージをインストールすると図 3‑4のようなディレクトリ構成になります。

|  |
| --- |
| /usr  └-local  └-bin  └-pg-rex  └-pg-rex\_master\_start  └-pg-rex\_slave\_start  └-pg-rex\_stop  └-pg-rex\_archivefile\_delete  └-pg-rex\_dbcluster\_check  └-pg-rex\_switchover  /etc  └- pg-rex\_tools.conf |

図 3‑4インストール後のディレクトリ構成

以下のコマンドを実行し、PG-REX運用補助ツールの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 IO\_Tty  **IO\_Tty-<ver>.el6.x86\_64**  # rpm -qa | grep Net\_OpenSSH  **Net\_OpenSSH-<ver>.el6.x86\_64**  # rpm -qa | grep pg-rex  **pg-rex\_operation\_tools\_script-<ver>.el6.noarch** |

### pg-rex\_tools.confの編集

pg-rex01とpg-rex02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、注意すべき設定は表 3‑12のとおりです。

本作業はrootユーザで行います。

表 3‑12　pg-rex\_tools.confの設定

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 設定値 | 備考 |
| 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。 |

設定例を以下に示します。

|  |
| --- |
| # vi /etc/pg-rex\_tools.conf  **D-LAN\_IPAddress = *192.168.2.1* , *192.168.2.2***  **Archive\_dir = /dbfp/pgarch/arc1**  （略） |

### 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\_host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。

本作業はrootユーザで行います。

pg-rex01で以下のコマンドを実行します。

|  |
| --- |
| # ssh *192.168.2.2* **←相手先のD-LANのIPアドレスを指定**  The authenticity of host '*192.168.2.2* (*192.168.2.2*)' can't be established.  RSA key fingerprint is \*\*\*\*\*\*\*  Are you sure you want to continue connecting (yes/no)? **yes** **←[yes]を入力し[Enter]キーを押下**  Warning: Permanently added '*192.168.2.2*' (RSA) to the list of known hosts.  root@*192.168.2.2*'s password: **←[Ctrl]キーと[C]キーを同時に押下** |

pg-rex02で以下のコマンドを実行します。

|  |
| --- |
| # ssh *192.168.2.1* **←相手先のD-LANのIPアドレスを指定**  The authenticity of host '*192.168.2.1* (*192.168.2.1*)' can't be established.  RSA key fingerprint is \*\*\*\*\*\*\*  Are you sure you want to continue connecting (yes/no)? **yes** **←[yes]を入力し[Enter]キーを押下**  Warning: Permanently added '*192.168.2.1*' (RSA) to the list of known hosts.  root@*192.168.2.1*'s password: **←[Ctrl]キーと[C]キーを同時に押下** |

## リソースの設定

PG-REX用の『pm\_crmgen環境定義書』(ファイル名:PG-REX9.4\_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、pg-rex01に転送する。
3. pg-rex01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。

『pm\_crmgen環境定義書』の見方、設定方法、注意点などについては、『pm\_crmgen環境定義書』の概要の項目に記述している文章を参照してください。以降、『pm\_crmgen環境定義書』を編集します。

### クラスタ・プロパティ、リソース・デフォルトの設定

『pm\_crmgen環境定義書』の表1-1～表3-2の設定内容は変更不要です。

### リソース構成の設定

DBクラスタを配置するディスクとOSのインストールディスクを違うディスクの構成にしている場合、『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、同じディスクの構成にしている場合、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。

### リソース・パラメータの設定

『pm\_crmgen環境定義書』の表5-1の設定内容は変更不要です。

### STONITH実行順序の設定

『Pacemaker関連サイト』を参考に、『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（仮想IP）の設定

『Pacemaker関連サイト』を参考に、『pm\_crmgen環境定義書』の表7-1-1、7-1-2、7-1-3それぞれに、PostgreSQLのMaster側接続用、Master側レプリケーション受付用、Slave側接続用の仮想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、注意すべき設定は表 3‑13のとおりです。

表 3‑13　PG-REXにおけるPostgreSQL制御の設定

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 設定値 | 備考 |
| start\_opt | -p 5432 | **接続ポート番号（-p）**を指定する。  PG-REXでは、接続先ホスト（-h）を指定することはできない。 |
| rep\_mode | sync | **同期モード**を指定する。 |
| node\_list | pg-rex01 pg-rex02 | **pg-rex01とpg-rex02のホスト名**を指定する（ホスト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**Master側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[[15]](#footnote-15)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 | 『3.4.7 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

『Pacemaker関連サイト』を参考に、『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

『Pacemaker関連サイト』を参考に、『pm\_crmgen環境定義書』の表7-3-1と7-3-2に、OSのインストールディスクの監視とDBクラスタを配置するディスクの監視を設定します。DBクラスタを配置するディスクとOSのインストールディスクを違うディスクの構成にしている場合は、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

『Pacemaker関連サイト』を参考に、『pm\_crmgen環境定義書』の表7-4-1～7-4-2にpg-rex01のSTONITHを設定します。表7-5-1～7-5-2には pg-rex02のSTONITHを設定します。

### リソース配置制約の設定

DBクラスタを配置するディスクとOSのインストールディスクを違うディスクの構成にしている場合、『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クラスタを配置するディスクとOSのインストールディスクを違うディスクの構成にしている場合、『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クラスタを配置するディスクとOSのインストールディスクを違うディスクの構成にしている場合、『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。

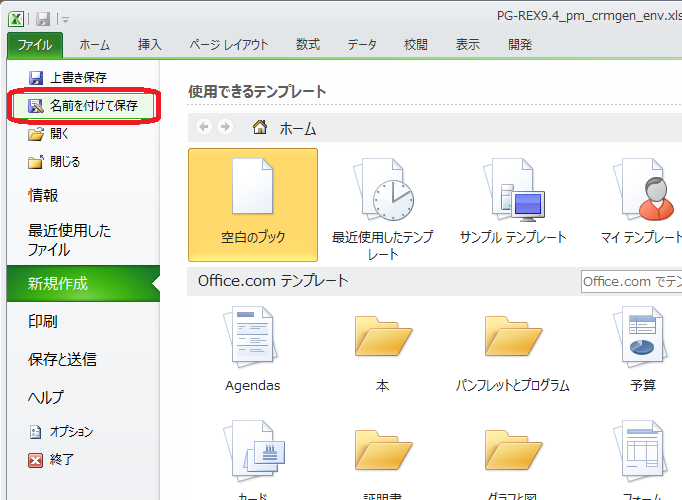
### リソースチケット制約の設定

『pm\_crmgen環境定義書』の表11-1の設定は不要です。

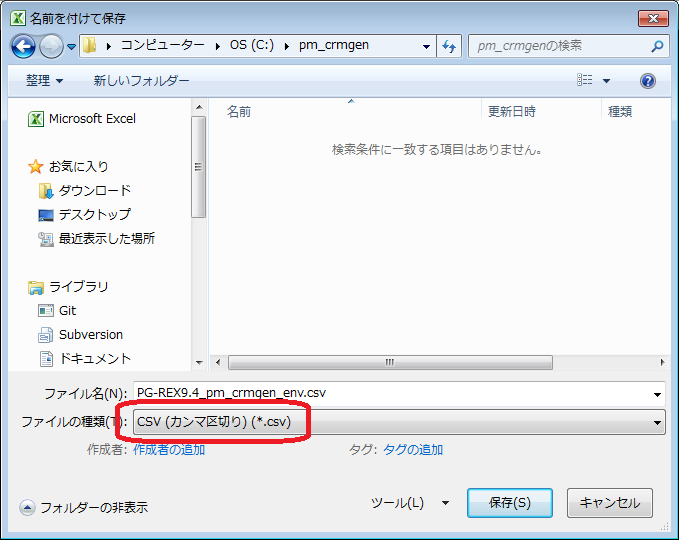
### crmファイルの作成

以下の操作を行い、crmファイルを作成します。以下ではMicrosoft® Excel 2010を使用した例を示します。

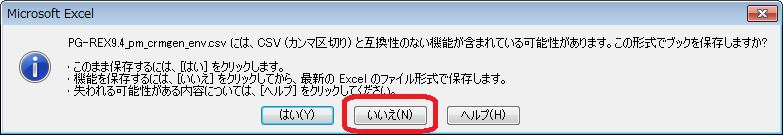
1. 『pm\_crmgen環境定義書』の「pm\_crmgen\_環境定義書(PG-REX9.4)」シートの修正後、修正対象シートを開いたまま、Microsoft® Excelのメニューバーから、「ファイル」→「名前を付けて保存」を選択します。



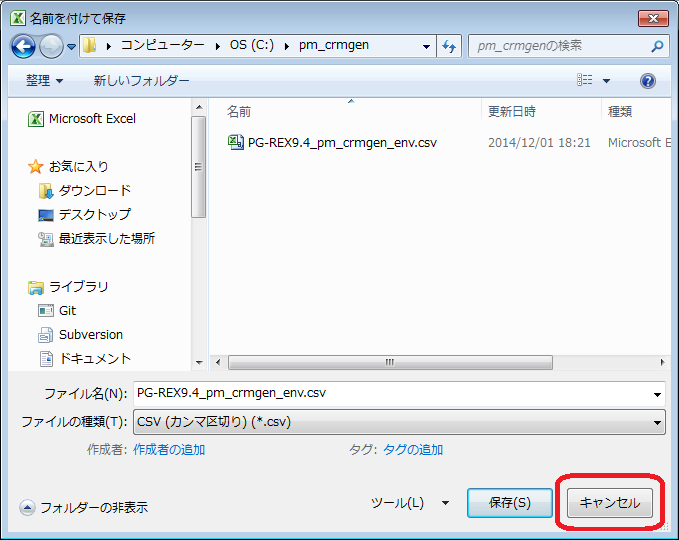
1. 「名前を付けて保存」のダイアログが表示されるので、ファイルの種類の欄で「CSV（カンマ区切り）（\*.csv）」を選択し、任意の場所に保存します。本書では、ファイル名は「PG-REX9.4\_pm\_crmgen\_env.csv」とします。



「互換性のない機能」に関するダイアログが表示されるため、「いいえ」を選択します。このダイアログが表示された時点で、csvファイルは保存されます。



再び「名前を付けて保存」のダイアログが表示されるため、「キャンセル」を選択します。



1. 保存したcsvファイル（PG-REX9.4\_pm\_crmgen\_env.csv）を、pg-rex01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. pg-rex01上で、pm\_crmgenコマンドを使用し、（3）で転送した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-REX9.4\_pm\_crmgen\_env.crm」とします。

|  |
| --- |
| # pm\_crmgen -o PG-REX9.4\_pm\_crmgen\_env.crm PG-REX9.4\_pm\_crmgen\_env.csv |

生成後のcrmファイルを確認します。

|  |
| --- |
| # ls –l PG-REX9.4\_pm\_crmgen\_env.crm  -**rw**-**r**--**r**-- 1 **root root** 6722 1月 30 10:05 2014 **PG-REX9.4\_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-REX9.4\_pm\_crmgen\_env.crm  # chmod 644 PG-REX9.4\_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、『4.2 Masterの起動』の（5）を参照してください。

## アンインストール

PG-REXをpg-rex01とpg-rex02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「4.4 両系の停止」の手順に従って停止してくだい。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。

PostgreSQLがインストール済であることを確認します。

|  |
| --- |
| # rpm -qa | grep postgresql  **postgresql94-server-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-libs-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-contrib-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-<ver>-1PGDG.rhel6.x86\_64**  **postgresql94-docs-<ver>-1PGDG.rhel6.x86\_64** |

上記を確認した後、RPMパッケージのアンインストールを行います。

|  |
| --- |
| # rpm -e postgresql94-contrib-<ver>-1PGDG.rhel6.x86\_64  # rpm -e postgresql94-server-<ver>-1PGDG.rhel6.x86\_64  # rpm -e postgresql94-docs-<ver>-1PGDG.rhel6.x86\_64  # rpm -e postgresql94-<ver>-1PGDG.rhel6.x86\_64  # rpm -e postgresql94-libs-<ver>-1PGDG.rhel6.x86\_64  # rpm -qa | grep postgresql  **（出力なし）** |

不要になったDBクラスタディレクトリ配下のファイルを削除します。

|  |
| --- |
| # rm -r /dbfp/pgdata/data/\* |

作成した環境変数のファイルを削除します。

|  |
| --- |
| # rm /var/lib/pgsql/.bash\_profile |

作成した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 /var/lib/pgsql/.pgpass |

その他、作成した環境に合わせて以下の項目も行う必要があります。ご確認ください。

* 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本体のアンインストールを行います。

|  |
| --- |
| # yum –y erase pacemaker corosync libqb cluster-glue cluster-glue-libs resource-agents pm\_crmgen pm\_logconv-cs pm\_diskd pm\_extras crmsh pssh |

以下のメッセージが出力されます。

|  |
| --- |
| Loaded plugins: product-id, refresh-packagekit, security, subscription-manager  Setting up Remove Process  Resolving Dependencies  --> Running transaction check  (略)  **Complete!** |

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。

|  |
| --- |
| # rpm –e pacemaker-repo-<ver>.x86\_64  # rpm –qa | grep pacemaker-repo  **（出力なし）** |

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/logrotate.d/pacemaker.rpmsave
* /etc/sysconfig/pacemaker.rpmsave
* /etc/init/pacemaker.combined.conf.rpmsave

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
* /usr/lib/ocf/resource.d/heartbeat**18**
* /usr/lib/ocf/resource.d/pacemaker[[16]](#footnote-16)

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

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-<ver>.el6.noarch  # rpm -e Net\_OpenSSH-<ver>.el6.x86\_64  # rpm -e IO\_Tty-<ver>.el6.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）** |

# 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。

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、『付録A起動と停止』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、『4.2 Masterの起動』を参照してください。
* Slaveの起動手順については、『4.3 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、pg-rex01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、pg-rex01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 -w 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。

1. pg-rex01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。本作業はrootユーザで行います。

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

1. 2回目以降のMaster起動の場合、以下の手順を実行します。以下の手順ではpg-rex\_tools.confのPEER\_NODE\_SSH\_PASS\_MODEの設定値がmanualの場合を示しています。本作業はrootユーザで行います。
2. pg-rex01でMasterを起動します。ただし、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(5)に記載のコマンドを実行します。

|  |
| --- |
| # pg-rex\_master\_start  **root@192.168.2.2's password:** **← pg-rex02のrootのパスワードを入力し[Enter]キーを押下**  **パスワードが入力されました**  **1. Pacemaker および Corosync が起動していないことを確認**  **…[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(pg-rex01)が Master として起動しました** |

1. 初回起動時、もしくは新しいcrmファイルを反映させてMasterを起動する場合、以下の手順を実行します。以下の手順ではpg-rex\_tools.confのPEER\_NODE\_SSH\_PASS\_MODEの設定値がmanualの場合を示しています。本作業はrootユーザで行います。
   1. 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-REX9.4\_pm\_crmgen\_env.crm  **root@192.168.2.2's password:** **← pg-rex02のrootのパスワードを入力し[Enter]キーを押下**  **パスワードが入力されました**  **1. Pacemaker および Corosync が起動していないことを確認**  **…[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** **← [y]を入力し[Enter]キーを押下**  **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(pg-rex01)が Master として起動しました** |

## 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、pg-rex02をSlaveとして起動します。

1. pg-rex02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。本作業はrootユーザで行います。

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

1. pg-rex02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『6.2.2 PostgreSQLアーカイブログの削除』を参照してください。本作業はrootユーザで行います。

|  |
| --- |
| # pg-rex\_slave\_start  **root@192.168.2.1's password:** **← pg-rex01のrootのパスワードを入力し[Enter]キーを押下**  **パスワードが入力されました**  **1. Pacemaker および Corosync が起動していないことを確認**  **…[OK]**  **2. 稼働中の Master が存在していることを確認**  **…[OK]**  **3. IC-LAN が接続されていることを確認**  **…[OK]**  **4. 起動禁止フラグの存在を確認**  **…[OK]** |

1. ベースバックアップを取得する場合は[y]、取得しない場合は[n]を入力します。ベースバックアップ取得の要否は、『A.3 Slaveの起動』の(4)を参照してください。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。

|  |
| --- |
| **5. Master からベースバックアップ取得**  **ベースバックアップの取得を行います**  **よろしいでしょうか？ (y/n) y** **← [y]を入力し[Enter]キーを押下**  **20227/20227 kB (100%), 1/1 tablespace**  **…[OK]**  **6. Master のアーカイブディレクトリと同期**  **receiving incremental file list**  **./**  **0000002A000000000000004C**  **0000002B.history**  **0000002B000000000000004C**  **0000002C.history**  **0000002C000000000000004C**  **0000002C000000000000004D**  **0000002C000000000000004D.00000020.backup**  **sent 147 bytes received 67122879 bytes 44748684.00 bytes/sec**  **total size is 83893968 speedup is 1.25**  **…[OK]**  **7. /var/lib/pacemaker/cib 配下のファイルを削除**  **…[OK]**  **8. Slave の起動**  **…[OK]**  **9. Slave の起動確認**  **…[OK]**  **ノード(pg-rex02)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、『4.6 Masterの停止』を参照してください。
* Slaveの停止手順については、『4.5 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にて、rootユーザで行います。

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 (pg-rex02)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にて、rootユーザで行います。

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 (pg-rex01)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。

|  |
| --- |
| **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)** |

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。

* pg-rex\_switchoverコマンドについては、「付録D運用補助ツール概要」を参照してください。

## 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にて、rootユーザで行います。

1. PostgreSQL停止中(Stopped)の系のPacemakerを停止します。

|  |
| --- |
| # pg-rex\_stop  **PostgreSQL が停止しています**  **Pacemaker を停止します**  **1. Pacemaker 停止**  **…[OK]**  **2. Pacemaker 停止確認**  **…[OK]**  **ノード(pg-rex01)で Pacemaker を停止しました** |

# 故障対応

本章では、クラスタシステムに発生した故障の故障箇所特定、および、故障時における、監視者の一次対応について説明します。なお、本章はPG-REXの運用を行う作業者を対象とし、対象読者がPacemakerによって構築されたクラスタシステムの運用手順や用語を理解していることを前提としています。

一次対応の作業には、サービス継続と復旧があります。

サービス継続では、クラスタシステムに故障発生後の一次対応の実施と保守者への報告を実施します。

復旧では、保守者の修復作業終了後、クラスタシステムを故障発生前の状態に戻します。

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。

具体的な作業内容や手順については、それぞれの運用手順を記したドキュメントに従うものとします。

なお、本作業はrootユーザで行い、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。

## 前提条件

本書における、クラスタ運用監視の前提条件を以下に示します。

* 一次対応については、ルータ故障、リソース故障(vip-slave)を除きpg-rex01で発生した故障を想定しています。また、リソース故障(start/promote)を除き両系とも起動され運用中に発生した故障を想定しています。
* 二重故障の対応については、対象外とします。但し、STONITH制御(reset)エラーについては対象とします。
* 復旧作業の手順は、保守者の修復作業によって該当しない場合があります。
* 内蔵ディスク故障時はOS停止状態になりますが、キャッシュ機能により暫くの間通常稼働しているためOS、クラスタ及びサービスアプリケーションは正常に稼働しているように見えます。但し、OS、クラスタ及びサービスアプリケーションは正常状態では無いためログイン、OS再起動は不可とします。
* 保守者による故障復旧においてハードウェア部品の交換が行われた場合、交換前の設定になっていることとします。
* 本書は、他運用監視機能等によるログ監視による保守者への通知を想定しています。
* 保守者による復旧実施後は、故障箇所は正常状態であるものとします。そのため、本書では復旧確認手順は省略します。

## クラスタ状態定義

本書では、ノード毎のリソース起動状態を表すために、クラスタ状態という表現を使用します。各ノードに適用するクラスタ状態を表 5‑1に示します。

表 5‑1　クラスタ状態一覧

|  |  |  |
| --- | --- | --- |
| No. | クラスタ状態 | 説明 |
| 1 | Master | PG-REXリソースがMasterとして起動した状態 |
| 2 | Slave | PG-REXリソースがSlaveとして起動した状態 |
| 3 | OUS | リソース故障、又はネットワーク経路故障/ディスク故障を検知した状態 |
| 4 | NONE | クラスタに組み込まれていない状態（Pacemaker停止中） |
| 5 | UNCLEAN | ノードが異常離脱した状態、又は、STONITH機能によって停止される前の状態 |

## 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、制御エラー情報を取得する必要があります。

以下にcrm\_monコマンド実行時の表示例を示します。

⑤制御エラー情報表示部

(※リソース故障時のみ表示)

④故障回数表示部

①ノード情報表示部

③属性情報表示部

②リソース情報表示部

# crm\_mon -fA -1

Last updated: *日時表示*

Last change: *日時表示*

Stack: corosync

Current DC: pg-rex01 (*UUID表示*) - partition with quorum

：（略）

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): Started pg-rex01

prmStonith2-2 (stonith:external/ipmi): Started pg-rex01

Resource Group: grpStonith1

prmStonith1-1 (stonith:external/stonith-helper): Started pg-rex02

prmStonith1-2 (stonith:external/ipmi): Started pg-rex02

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex01 ]

Slaves: [ pg-rex02 ]

Clone Set: clnDiskd1 [prmDiskd1]

Started: [ pg-rex01 pg-rex02 ]

Clone Set: clnPing [prmPing]

Started: [ pg-rex01 pg-rex02 ]

Node Attributes:

\* Node pg-rex01:

+ default\_ping\_set : 100

+ diskcheck\_status\_internal : normal

+ master-pgsql : 1000

+ pgsql-data-status : LATEST

+ pgsql-master-baseline : 0000000002000270

+ pgsql-status : PRI

+ ringnumber\_0 : 192.168.1.1 is UP

+ ringnumber\_1 : 192.168.3.1 is UP

\* Node pg-rex02:

+ default\_ping\_set : 100

+ 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 pg-rex01:

\* Node pg-rex02:

Failed actions:

### 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。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、そのノードのPacemakerは停止しています。

①ノード情報表示部： 各ノードのホスト名やPacemakerの稼動状態が表示されます。

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

standby状態等のノードについては、UUID(Pacemaker内部でのノード識別子)も表示されます。

②リソース情報表示部： リソースID、リソースの稼動状態や稼動ノードのホスト名が表示されます。

各リソースが示す稼働状況を表 5‑2に示します。

表 5‑2　リソース情報表示部の稼働状況表示一覧

|  |  |  |  |
| --- | --- | --- | --- |
| No. | リソース(リソースID) | 表示 | 稼働状況 |
| 1 | PG-REXリソース (msPostgresql) | Masters: [ pg-rex01 ]  Slaves: [ pg-rex02 ] | 両系でリソースが稼働中 |
| 2 | Masters: [ pg-rex02 ]  Stopped: [ pg-rex01 ] | 片系(pg-rex02)でリソースが稼働中 |
| 3 | Masters: [ pg-rex01 ]  Stopped: [ pg-rex02 ] | 片系(pg-rex01)でリソースが稼働中 |
| 4 | 表示なし | 両系でリソースが停止中 |
| 5 | IPaddr2リソース (vip-master)  (vip-rep) | Started pg-rex01 | pg-rex01でリソースが稼働中 |
| 6 | Started pg-rex02 | pg-rex02でリソースが稼働中 |
| 7 | 表示なし | リソースが停止中 |
| 8 | IPaddr2リソース  (vip-slave) | Started pg-rex02 | pg-rex02でリソースが稼働中 |
| 9 | Started pg-rex01 | pg-rex01でリソースが稼働中 |
| 10 | 表示なし | リソースが停止中 |
| 11 | STONITHリソース (grpStonith1)  (grpStonith2) | Started pg-rex01 | pg-rex01でリソースが稼働中 |
| 12 | 表示なし | リソースが停止中 |
| 13 | cloneリソース  (clnPing) | Started: [ pg-rex01 pg-rex02 ] | 両系でリソースが稼働中 |
| 14 | Started: [ pg-rex02 ]  Stopped: [ pg-rex01 ] | 片系(pg-rex02)でリソースが稼働中 |
| 15 | Started: [ pg-rex01 ]  Stopped: [ pg-rex02 ] | 片系(pg-rex01)でリソースが稼働中 |
| 16 | 表示なし | 両系でリソースが停止中 |
| 17 | cloneリソース  (clnDiskd1) | Started: [ pg-rex01 pg-rex02 ] | 両系でリソースが稼働中 |
| 18 | Started: [ pg-rex02 ]  Stopped: [ pg-rex01 ] | 片系(pg-rex02)でリソースが稼働中 |
| 19 | Started: [ pg-rex01 ]  Stopped: [ pg-rex02 ] | 片系(pg-rex01)でリソースが稼働中 |
| 20 | 表示なし | 両系でリソースが停止中 |

③属性情報表示部： 各ノードにおけるネットワーク経路監視、ディスク監視、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。

各監視先の属性名の正常時/異常時における属性値を表 5‑3に示します。

表 5‑3　属性情報表示部の属性値表示一覧

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 監視先 | 属性名 | 属性値 | |
| 正常時 | 異常時 |
| 1 | ネットワーク経路 | default\_ping\_set | 100 | 0 |
| 2 | 内蔵ディスク | diskcheck\_status\_internal | normal | ERROR |
| 3 | PostgreSQL | master-pgsql | 1000(pg-rex01)  100(pg-rex02) | -INFINITY |
| 4 | pgsql-data-status | LATEST(pg-rex01)  STREAMING|SYNC (pg-rex02) | 左記以外の値 |
| 5 | pgsql-master-baseline[[17]](#footnote-17) | xlog　location[[18]](#footnote-18)(pg-rex01)  表示なし(pg-rex02) | 左記以外の値 |
| 6 | pgsql-status | PRI(pg-rex01)  HS:sync(pg-rex02) | 左記以外の値 |
| 7 | ringnumber\_0[[19]](#footnote-19) | 192.168.1.2 is up | 表示なし[[20]](#footnote-20) |
| 8 | pgsql-xloc-loc | 表示なし | 表示[[21]](#footnote-21) |

④故障回数表示部： ホスト名毎に故障したリソースIDと故障した回数が表示されます。  
monitor故障時以外は、故障回数に1000000 (INFINITY)が表示されます。  
demote故障発生時は、故障回数表示部に表示がされません。

⑤制御エラー情報表示部： 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)、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。

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。

制御エラー情報表示部の出力フォーマットを以下に示します。

Failed actions:

pgsql\_monitor\_9000 on pg-rex01 'not running' (7): call=79, status=complete,

① ② ③ ④ ⑤

last-rc-change='Mon Oct 27 16:10:04 2014', 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表 5‑4に示します。

表 5‑4に示す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。

表 5‑4　正常状態確認項目一覧

|  |  |  |  |
| --- | --- | --- | --- |
| No. | 表示部 | 正常状態の確認項目 | |
| 1 | ①ノード情報表示部 | pg-rex01とpg-rex02のノード状態が"Online"状態になっていることを確認する。 | |
| 2 | ②リソース情報表示部 | 【サービス系リソース】 | |
|  |  |  | PG-REXリソース(msPostgresql)、IPaddr2リソース(vip-master、vip-slave、vip-rep)が以下のとおり稼動していることを確認する。  pg-rex01： msPostgresql (Master)  vip-master  vip-rep  pg-rex02： msPostgresql (Slave)  vip-slave |
|  |  | 【STONITHリソース】 | |
|  |  |  | STONITHリソースが、それぞれのノードで稼動していることを確認する。 |
|  |  | 【cloneリソース】 | |
|  |  |  | ネットワーク経路監視用cloneリソース、内蔵ディスク監視用cloneリソースが全てのノードで稼動していることを確認する。 |
| 3 | ③属性情報表示部 | ネットワーク経路、内蔵ディスク、PG-REXリソース、IC-LAN監視のそれぞれの状態について異常値を示している属性名が無いことを確認する。 | |
| 4 | ⑤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際の手順について説明します。

故障箇所特定の手順を以下に示します。

* ノード情報・リソース情報・故障回数表示部の確認 (『5.4.1 ノード情報・リソース情報・故障回数表示部の確認』)

Pacemaker停止ノードが1台以上存在する場合は、STONITH機能reset処理実行結果を調べます。

全ノードのPacemakerが起動中の場合は、IC-LAN故障の特定、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。

* 属性情報表示部の確認 (『5.4.2 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。

* 制御エラー情報表示部の確認 (『5.4.3 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。

* /var/log/pm\_logconv.outの確認 (『5.4.4 /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。

故障箇所特定までのフローを図 5‑1に示します。

属性情報表示部の確認

制御エラー情報表示部の確認

D-LAN故障

Pacemaker停止ﾉｰﾄﾞが2台の場合

Pacemaker停止ﾉｰﾄﾞが1台の場合

D-LAN  
故障の場合

ハードウェア故障の場合

リソース故障の場合

ﾙｰﾀ故障/  
S-LAN故障/内蔵ﾃﾞｨｽｸ故障

リソース故障(start/promote/monitor/vip-master/vip-slave/vip-rep)

リソース故障(demote/stop)

ノード故障

IC-LAN  
故障の場合

ノード情報・リソース情報・故障回数表示部の確認

IC-LAN故障

/var/log/pm\_logconv.outの確認

図 5‑1　故障特定手順

### 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、以下の状態に応じてそれぞれの確認を実行します。

* Pacemaker停止ノード数が2台の場合

IC-LAN故障が発生し、pg-rex01とpg-rex02から、STONITH機能のreset処理が実施され相撃ちしています。

『5.5 故障一覧』のIC-LAN故障を参照します。

* Pacemaker停止ノード数が1台のみの場合

Pacemakerが起動しているノードでcrm\_monコマンドを実行し、ノード情報表示部に表示されているPacemakerが停止しているノードの状態から、STONITH機能のreset処理実行結果を確認します。

|  |  |
| --- | --- |
| Pacemaker停止 ノードの状態 | Pacemaker停止ノードの状態reset処理実行結果： 実行結果特定後遷移先 |
| UNCLEAN (offline) | STONITH機能reset処理失敗：『5.4.4 /var/log/pm\_logconv.outの確認』 |
| OFFLINE | STONITH機能reset処理成功：『5.4.4 /var/log/pm\_logconv.outの確認』 |

* 全ノードのPacemakerが起動中の場合

1. IC-LAN故障の特定

pg-rex01とpg-rex02でcrm\_monコマンドを実行し、ノード情報表示部を確認します。

以下の表示例のように、pg-rex01とpg-rex02でノード情報表示部の表示が異なる場合、IC-LAN故障と特定できます。

IC-LAN故障と特定された場合は、『5.5 故障一覧』のIC-LAN故障を参照します。

pg-rex01におけるCurrent DCとノード状態表示部の表示例

← Current DCがpg-rex01

pg-rex02がOFFLINE

Current DC: pg-rex01 (*UUID表示*) - partition with quorum

：（略）

Online: [ pg-rex01 ]

OFFLINE: [ pg-rex02 ]

pg-rex02におけるノード状態表示部の表示例

← Current DCがpg-rex02

pg-rex01がOFFLINE

Current DC: pg-rex02 (*UUID表示*) - partition with quorum

：（略）

Online: [ pg-rex02 ]

OFFLINE: [ pg-rex01 ]

1. D-LAN故障の特定

pg-rex01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 pg-rex02:

+ default\_ping\_set : 100

+ 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、pg-rex02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、pg-rex01のPostgreSQLのサーバログファイル[[22]](#footnote-22)を以下に示す正規表現を条件に検索します。

正規表現：.\*terminating.\*walsender.\*

# grep .\*terminating.\*walsender.\* /var/log/postgresql22

Mar 12 16:39:46 **pg-rex01** **postgres**[3174]: [33-1] ~ LOG: 00000: **terminating walsender process due to replication timeout**

該当のログが出力された場合、『5.5 故障一覧』のD-LAN故障を参照します。

1. ハードウェア故障・リソース故障の切り分け

クラスタシステム内のいずれかのノードでcrm\_monコマンドを実行し、ノード情報、リソース情報、故障回数表示部から、ハードウェア故障(ネットワーク、ディスク等)、リソース故障の切り分けを行います。表 5‑5に示す状態に応じて該当する項へ遷移します。

表 5‑5　故障箇所別遷移先対応表

|  |  |  |  |
| --- | --- | --- | --- |
| 表示部 | | | 故障切り分け： 故障箇所特定遷移先 |
| ①ノード情報 | ②リソース情報 | ④故障回数 |
| 全ノード：  Online | 表示なし | 表示なし | ハードウェア故障：『5.4.2 属性情報表示部の確認』 |
|  | PG-REXリソース (msPostgresql)： Masters: [ pg-rex01 ]  (リソースF/O無) | 表示なし |  |
| pg-rex01： fail-count=1 | リソース故障：『5.4.3 制御エラー情報表示部の確認』 |
|  | PG-REXリソース (msPostgresql)： Masters: [ pg-rex02 ]  (リソースF/O有) | 表示なし | ハードウェア故障：『5.4.2 属性情報表示部の確認』 |
|  |  | pg-rex01： fail-count=1000000 | リソース故障：『5.4.3 制御エラー情報表示部の確認』 |
|  |  | pg-rex01： fail-count=1 |  |
|  | 表示なし | pg-rex02： fail-count=1 |  |
| pg-rex01： UNCLEAN (online)  pg-rex02： Online |  | pg-rex01： fail-count=1000000 |  |

### 属性情報表示部の確認

属性情報表示部の確認では、pg-rex01またはpg-rex02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、クラスタシステムを構成しているノードのネットワーク経路、内蔵ディスク、IC-LAN (一部インターフェイス通信異常)に発生した故障を特定します。

以下にcrm\_monコマンド実行時の表示例を示します。

pg-rex01でdefault\_ping\_setの属性値が異常

： (略)

\* Node pg-rex01:

+ default\_ping\_set : 0 : Connectivity is lost

+ diskcheck\_status\_internal : normal

+ master-pgsql : -INFINITY

+ pgsql-data-status : DISCONNECT

+ pgsql-status : STOP

+ ringnumber\_0 : 192.168.1.1 is UP

+ ringnumber\_1 : 192.168.3.1 is UP

\* Node pg-rex02:

+ default\_ping\_set : 100

+ 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）

上記の例では、pg-rex01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、pg-rex01にてS-LANの故障を検知したことを示しています。

属性情報表示部の出力と対応する故障内容を表 5‑6に示します。実施後は『5.5 故障一覧』に遷移し、特定された故障を参照します。

表 5‑6　属性情報別故障箇所対応表

|  |  |  |  |
| --- | --- | --- | --- |
| ④属性情報表示部 | | | 故障内容  (故障発生ノードまたは異常個所) |
| 属性名 | 属性値 (pg-rex01) | 属性値 (pg-rex02) |
| default\_ping\_set | 0 | 100 | S-LAN故障 (pg-rex01) |
|  | 100 | 0 | S-LAN故障 (pg-rex02) |
|  | 0 | 0 | ルータ故障 (pg-rex01、pg-rex02) |
| diskcheck\_status\_internal | ERROR | normal | 内蔵ディスク故障 (pg-rex01) |
|  | normal | ERROR | 内蔵ディスク故障 (pg-rex02) |
|  | ERROR | ERROR | 内蔵ディスク故障 (pg-rex01、pg-rex02) |
| pgsql-data-status | LATEST | DISCONNECT | D-LAN故障 |
| pgsql-status | PRI | HS:alone | D-LAN故障 |
| ringnumber-0[[23]](#footnote-23) | 192.168.1.1 is FAULTY[[24]](#footnote-24) | 192.168.1.1 is FAULTY | IC-LAN故障 (eth1インターフェイスによる通信異常) |

### 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、pg-rex01またはpg-rex02でcrm\_monコマンドを実行し、故障リソースID、検知オペレーション、故障ノードによって故障内容を特定し、クラスタシステム内のリソースに発生した故障を特定します。実施後は『5.5 故障一覧』に遷移し、特定された故障を参照します。

1. start故障の特定

以下にstart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**\_**start**\_0 on **pg-rex01** 'not installed' (**5**): call=52, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

上記の例では、故障リソースIDが"pgsql"、検知オペレーションが"start"、故障ノードが"pg-rex01"、リターンコードが"5"と表示されていることから、pg-rex01のノードで、PG-REXリソース(pgsql)のstart故障(リターンコード5)が発生したことを示しています。

startでは、以下に示す原因でも故障が発生します。

* 起動禁止フラグ(/var/lib/pgsql/tmp/PGSQL.lock)消し忘れ。

1. promote故障の特定

以下にpromote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**\_**promote**\_0 on **pg-rex01** 'unknown exec error' (**-2**): call=16, status=Timed Out, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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 **pg-rex01** 'unknown error' (1): call=28, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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 **pg-rex01** 'unknown error' (1): call=52, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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 **pg-rex01** 'unknown error' (1): call=46, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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 **pg-rex01** 'unknown error' (1): call=22, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

**pgsql**\_**monitor**\_10000 on **pg-rex01** 'not running' (7): call=56, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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 **pg-rex02** 'unknown error' (1): call=16, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

1. 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。

# crm\_mon -fA -1

:（略）

Migration summary:

\* Node **pg-rex01**:

**vip-rep**: migration-threshold=0 **fail-count=1** last-failure='日時表示'

\* Node pg-rex02:

上記の例では、"vip-rep"が1回故障したことを示しています。

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:

**pgsql**\_**monitor**\_9000 on **pg-rex01** 'unknown error' (7): call=28, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

**pgsql**\_**stop**\_0 on **pg-rex01** 'unknown error' (1): call=32, status=complete, last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

### /var/log/pm\_logconv.outの確認

Pacemakerが停止しているノードに対して正規表現をキーワードに/var/log/pm\_logconv.outから該当するエラーメッセージを検索し、故障を特定します。

Pacemaker停止ノードに対して表 5‑7に示す確認項目を、手順に従って実行します。実施後は『5.5 故障一覧』に遷移し、特定された故障を参照します。

表 5‑7　Pacemaker停止ノードに対する確認項目

|  |  |  |  |
| --- | --- | --- | --- |
| Pacemaker 故障ノード | 確認項目 | | |
| 手順1 | 手順2 | 手順3 |
| pg-rex01 | リソース故障(stop)の確認 | IC-LAN故障の確認 | ノード故障の確認 |
| pg-rex02 | － | IC-LAN故障の確認 | ノード故障の確認 |

1. リソース故障(stop)の確認

正規表現：".\*Resource.\*monitor.\*"Timed Out".\*"

# grep .\*Resource.\*stop.\*rc=1.\* /var/log/pm\_logconv.out

**→以下はpg-rex01で検索した場合の出力例**

Jun 18 12:59:23 pg-rex01 ERROR: Resource **pgsql** failed to **stop**. (**rc=1**)

該当のログが出力された場合、pg-rex01で、PG-REXリソース(msPostgresql)のstop故障(リターンコード1)が発生したことを示しています。

正規表現：".\*Resource.\*stop.\*"Timed Out".\* "

# grep .\*Resource.\*stop.\*"Timed Out".\* /var/log/pm\_logconv.out

**→以下はpg-rex01で検索した場合の出力例**

Jun 18 13:03:22 pg-rex01 ERROR: Resource **pgsql** failed to **stop**. (**Timed Out**)

該当のログが出力された場合、pg-rex01で、PG-REXリソース(msPostgresql)のstop故障(Timed Out)が発生したことを示しています。

1. IC-LAN故障の確認

正規表現：".\*Ring.\*number.\*FAULTY.\*"

# grep .\*Ring.\*number.\*FAULTY.\* /var/log/pm\_logconv.out

**→以下はpg-rex01で検索した場合の出力例**

Nov 19 11:14:53 pg-rex01 warning: **Ring number 0 is FAULTY (interface 192.168.1.1)**.

Nov 19 11:14:53 pg-rex01 warning: **Ring number 1 is FAULTY (interface 192.168.1.3)**.

※全てのIC-LANが切断したメッセージが出力されていることを確認します。

該当のログが出力された場合、IC-LAN故障が発生したことを示しています。

1. ノード故障の確認

リソース故障(stop)またはIC-LAN故障の確認で該当ログが出力されなかった場合、ノード故障 (Pacemaker故障またはOS故障) が発生したことを示しています。

## 故障一覧

本書で対応する対象故障一覧を表 5‑8に示します。

表 5‑8　故障一覧

|  |  |  |
| --- | --- | --- |
| No. | 故障 | 参照節番号 |
| 1 | ルータ故障 | 『5.6 ルータ故障』 |
| 2 | S-LAN故障 | 『5.7 S-LAN故障』 |
| 3 | D-LAN故障 | 『5.8 D-LAN故障』 |
| 4 | リソース故障(start/promote) | 『5.9 リソース故障(start/promote)』 |
| 5 | リソース故障(monitor) | 『5.10 リソース故障(monitor)』 |
| 6 | リソース故障(demote/stop) | 『5.11 リソース故障(demote/stop)』 |
| 7 | リソース故障(vip-master) | 『5.12 リソース故障(vip-master)』 |
| 8 | リソース故障(vip-rep) | 『5.13 リソース故障(vip-rep)』 |
| 9 | リソース故障(vip-slave) | 『5.14 リソース故障(vip-slave)』 |
| 10 | 内蔵ディスク故障 | 『5.15 内蔵ディスク故障』 |
| 11 | ノード故障 | 『5.16 ノード故障』 |
| 12 | IC-LAN故障 | 『5.17 IC-LAN故障』 |

## ルータ故障

この節では、ルータ故障時における一次対応について説明します。

### 故障時のクラスタ状態

ルータ故障時における、各ノードの状態を表 5‑9に示します。

表 5‑9　ルータ故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| OUS | OUS | S-LAN故障を検知し、PG-REXリソース (msPostgresql)が正常終了  PG-REXリソースがMasterから降格したことを受け、IPaddr2リソース(vip-master、vip-rep)が正常終了 | S-LAN故障を検知し、PG-REXリソース(msPostgresql)が正常終了  PG-REXリソースが終了したことを受け、IPaddr2リソース(vip-slave)が正常終了 | 『5.6.2 復旧』 STEP1 |
| 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith2】  ・正常状態  【clnPing】  ・S-LAN故障検知状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・S-LAN故障検知状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例を図 5‑2に示します。

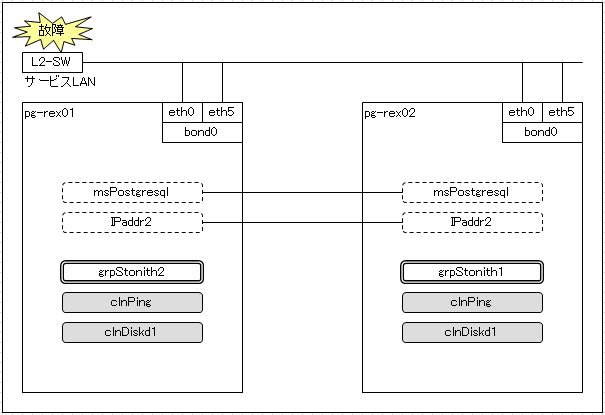


図 5‑2　ルータ故障状況

### 復旧

pg-rex01とpg-rex02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、pg-rex01とpg-rex02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、pg-rex02が先に停止した場合はpg-rex01(Master) - pg-rex02(Slave)、pg-rex01が先に停止した場合はpg-rex01(Slave) - pg-rex02(Master)となります。

復旧手順を図 5‑3に示します。

クラスタ状態

復旧手順

pg-rex01(OUS) - pg-rex02(OUS)

STEP2

Pacemaker停止[pg-rex02]

STEP3

Pacemaker停止[pg-rex01]

STEP4

保守者へ報告

STEP5

STEP5

STEP7

Pacemaker起動[pg-rex01]

STEP8

ノード状態・リソース状態確認[pg-rex01]

Pacemaker起動[pg-rex02]

ノード状態・リソース状態確認[pg-rex02]

保守者による故障復旧

STEP6

リソース状態確認[pg-rex01]

STEP1

図 5‑3　ルータ故障復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex01]

データが進んでいるノードを確認する。crm\_monの結果を確認し、pgsql-data-statusがLATESTとなっているノードをデータが進んでいる系とする。以下pg-rex01がデータの進んでいるノードの場合を示します。pg-rex02がデータの進んでいるノードの場合はpg-rex01とpg-rex02を読み替えて下さい。

STEP2： Pacemaker停止 [pg-rex02]

保守者の作業中に、PG-REXリソースが再起動しないようpg-rex02のPacemakerを停止します。停止する手順は『4.7 PostgreSQL停止中の系の停止』を参照してください。

STEP3： Pacemaker停止 [pg-rex01]

保守者の作業中に、PG-REXリソースが再起動しないようpg-rex01のPacemakerを停止します。停止する手順は『4.7 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。

* 報告時点でのサービス稼働状況  
  全ノードでサービス停止中
* 報告時点でのクラスタ状態  
  pg-rex01 (NONE) - pg-rex02 (NONE)  
  全ノードでPacemaker停止中
* 故障箇所  
  ルータ故障が発生

STEP5： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.2 Masterの起動』を参照してください。

STEP6： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01 ]**

OFFLINE: [ pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex01**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

Stopped: [ pg-rex02 ]

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01 ]**

Stopped: [ pg-rex02 ]

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01 ]**

Stopped: [ pg-rex02 ]

：（略）

STEP7： Pacemaker起動 [pg-rex02]

pg-rex02のPacemakerを起動します。起動する手順は『4.3 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 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [** pg-rex01 **pg-rex02 ]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Resource Group: grpStonith1

prmStonith1-1 (stonith:external/stonith-helper): **Started pg-rex02**

prmStonith1-2 (stonith:external/ipmi): **Started pg-rex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex01 ]

**Slaves: [ pg-rex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pg-rex01 **pg-rex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pg-rex01 **pg-rex02 ]**

：（略）

## S-LAN故障

この節では、S-LAN故障時における一次対応について説明します。

### 故障時のクラスタ状態

S-LAN故障時における、各ノードの状態を表 5‑10に示します。

表 5‑10　S-LAN故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| OUS | Master | S-LAN故障を検知し、PG-REXリソース(msPostgresql)が正常終了  PG-REXリソースがMasterから降格したことを受けて、IPaddr2リソース(vip-master、vip-rep)が正常終了 | pg-rex01のPG-REXリソース(msPostgresql)の停止を受け、pg-rex02のPG-REXリソース(msPostgresql)がMasterへ昇格  PG-REXリソース(msPostgresql)の昇格を受け、IPaddr2リソース(vip-master、vip-rep)が起動 | 『5.7.2 復旧』 STEP1 |
| 【msPostgresql】  ・pg-rex02へフェイルオーバ  【vip-master】  ・pg-rex02へフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02へフェイルオーバ  【grpStonith2】  ・正常状態  【clnPing】  ・故障検知状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例を図 5‑4に示します。

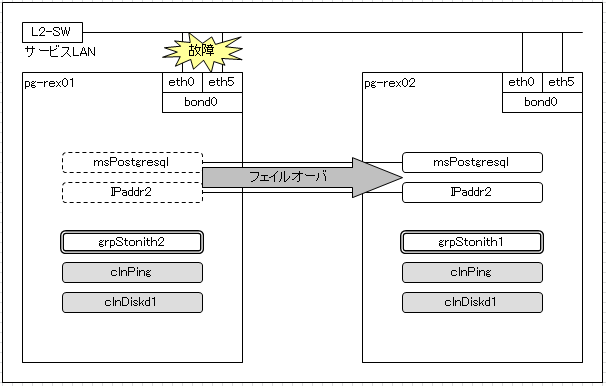


図 5‑4　S-LAN故障状況

### 復旧

サービス状況の確認とpg-rex01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、pg-rex01のPacemakerを再起動します。

復旧後のクラスタ状態は、pg-rex01(Slave) - pg-rex02(Master)となります。

復旧手順を図 5‑5に示します。

クラスタ状態

復旧手順

pg-rex01(OUS) - pg-rex02(Master)

STEP2

Pacemaker停止[pg-rex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex02]

ノード状態確認[pg-rex02]

STEP4

図 5‑5　S-LAN故障復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP2： Pacemaker停止 [pg-rex01]

保守者の作業中に、PG-REXリソースが再起動しないようpg-rex01のPacemakerを停止します。停止する手順は『4.7 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 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01 (NONE) - pg-rex02 (Master)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  S-LAN故障が発生

STEP5： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01のPG-REXリソースをSlaveとして起動します。

STEP6： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## D-LAN故障

この節では、D-LAN故障時における一次対応について説明します。

### 故障時のクラスタ状態

D-LAN故障時における、各ノードの状態を表 5‑11に示します。

表 5‑11　D-LAN故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| Master | Slave | D-LAN故障を検知し、pg-rex01がpg-rex02のpgsql-data-statusの値を変更 | pg-rex02のpgsql-data-statusの変更により、master-pgsql、pgsql-statusの値が変更  vip-slaveがpg-rex01に切り替わる | 『5.8.2 復旧』 STEP1 |
| 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例を図 5‑6に示します。

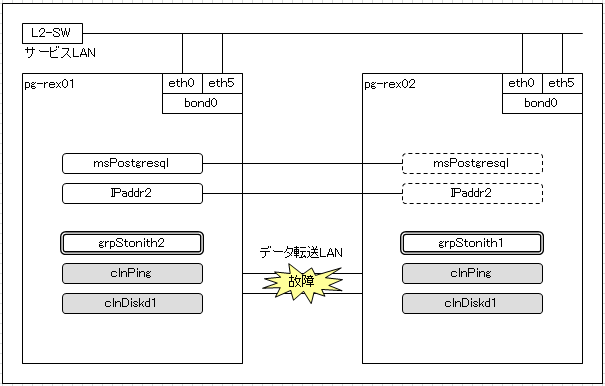


図 5‑6　D-LAN故障状況

### 復旧

サービス状況の確認とpg-rex02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、pg-rex02のPacemakerを再起動し、故障発生前のクラスタ状態に戻します。

復旧手順を図 5‑7に示します。

クラスタ状態

復旧手順

pg-rex01(Master) - pg-rex02(Slave)

STEP2

Pacemaker停止[pg-rex02]

STEP4

保守者へ報告

STEP5

STEP6

Pacemaker起動[pg-rex02]

ノード状態・リソース状態確認[pg-rex02]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex01]

STEP3

ノード状態確認[pg-rex01]

図 5‑7　D-LAN故障復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

**Slaves: [ pg-rex02 ]**

：（略）

STEP2： Pacemaker停止 [pg-rex02]

pg-rex02のPacemakerを停止します。停止する手順は『4.5 Slaveの停止』を参照してください。

STEP3： ノード状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 ]

**OFFLINE: [ pg-rex02 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex01でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01 (Master) - pg-rex02 (NONE)  
  pg-rex02でPacemaker停止中
* 故障箇所  
  D-LAN故障が発生

STEP5： Pacemaker起動 [pg-rex02]

pg-rex02のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

STEP6： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [** pg-rex01 **pg-rex02 ]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Resource Group: grpStonith1

prmStonith1-1 (stonith:external/stonith-helper): **Started pg-rex02**

prmStonith1-2 (stonith:external/ipmi): **Started pg-rex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex01 ]

**Slaves: [ pg-rex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pg-rex01 **pg-rex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pg-rex01 **pg-rex02 ]**

：（略）

## リソース故障(start/promote)

この節では、リソース故障(start/promote)時における一次対応について説明します。

リソース故障(start/promote)は他の節とは異なり、起動する際に発生する故障となります。

以下、pg-rex01起動時で発生した故障について記載します。

### 故障時のクラスタ状態

リソース故障(start/promote)時における、各ノードの状態を表 5‑12に示します。

表 5‑12　リソース故障(start/promote)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| OUS | NONE | リソース故障を検知し、PG-REXリソース(msPostgresql)が正常終了 |  | 『5.9.2 復旧』 STEP1 |
| 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith1】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 |

故障発生時におけるクラスタ状態の例を図 5‑8に示します。

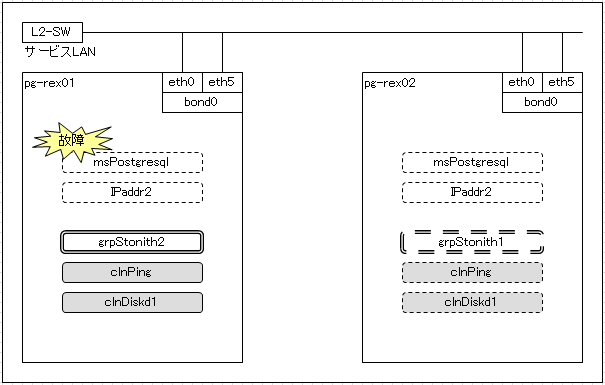


図 5‑8　リソース故障(start/promote)状況

### 復旧

pg-rex01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、pg-rex01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、pg-rex01(Master) - pg-rex02(NONE)となります。

復旧手順を図 5‑9に示します。

クラスタ状態

復旧手順

pg-rex01(OUS) - pg-rex02(NONE)

STEP2

Pacemaker停止[pg-rex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex01]

ノード状態確認[pg-rex01]

STEP4

図 5‑9　リソース故障(start/promote)復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 ]

OFFLINE: [ pg-rex02 ]

：（略）

Clone Set: clnDiskd1 [prmDiskd1]

Started: [ pg-rex01 ]

Stopped: [ pg-rex02 ]

Clone Set: clnPing [prmPing]

Started: [ pg-rex01 ]

Stopped: [ pg-rex02 ]

：（略）

Migration summary:

\* Node pg-rex01:

***故障回数表示***

**Failed actions:**

***msPostgresql制御エラー情報表示***

STEP2： Pacemaker停止 [pg-rex01]

pg-rex01のPacemakerを停止します。停止する手順は『4.7 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 [pg-rex01]

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

**Connection to cluster failed: Transport endpoint is not connected**

STEP4： 保守者へ報告

以下の内容を報告します。

* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(NONE)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  pg-rex01のリソース故障(start/promote)が発生

STEP5： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.2 Masterの起動』を参照してください。

STEP6： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01 ]**

OFFLINE: [ pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex01**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

Stopped: [ pg-rex02 ]

：（略）

## リソース故障(monitor)

この節では、リソース故障(monitor)時における一次対応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)時における、各ノードの状態を表 5‑13に示します。

表 5‑13　リソース故障(monitor)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| OUS | Master | リソース故障を検知し、PG-REXリソース(msPostgresql)が正常終了  PG-REXリソースがMasterから降格したことを受けて、IPaddr2リソース(vip-master、vip-rep)が正常終了 | pg-rex01のPG-REXリソースの停止を受け、pg-rex02のPG-REXリソース(msPostgresql)がMasterへ昇格  PG-REXリソース(msPostgresql)の昇格を受け、IPaddr2リソース(vip-master、vip-rep)が起動 | 『5.10.2 復旧』 STEP1 |
| 【msPostgresql】  ・pg-rex02へフェイルオーバ  【vip-master】  ・pg-rex02へフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02へフェイルオーバ  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例を図 5‑10に示します。

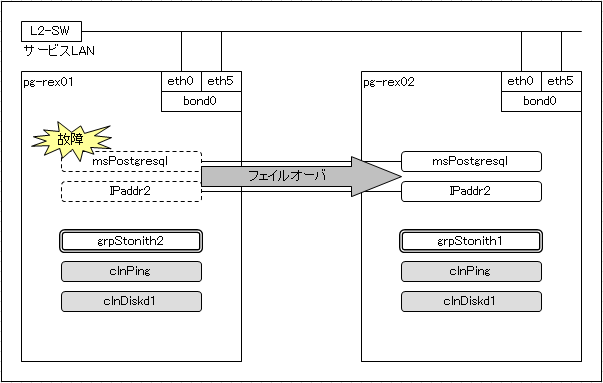


図 5‑10　リソース故障(monitor)状況

### 復旧

サービス状況の確認とpg-rex01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、pg-rex01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、pg-rex01(Slave) - pg-rex02(Master)となります。

復旧手順を図 5‑11に示します。

クラスタ状態

復旧手順

pg-rex01(OUS) - pg-rex02(Master)

STEP2

Pacemaker停止[pg-rex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex02]

ノード状態確認[pg-rex02]

STEP4

図 5‑11　リソース故障(monitor)復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP2： Pacemaker停止 [pg-rex01]

pg-rex01のPacemakerを停止します。停止する手順は『4.7 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 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  pg-rex01のリソース故障(monitor)が発生

STEP5： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01のPG-REXリソースをSlaveとして起動します。

STEP6： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## リソース故障(demote/stop)

この節では、リソース故障(demote/stop)時における一次対応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)時における、各ノードの状態を表 5‑14に示します。

表 5‑14　リソース故障(demote/stop)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| UNCL EAN | Slave | PG-REXリソースを停止させたが停止処理に失敗 | pg-rex01のPG-REXリソース停止処理失敗を受け、reset処理を実施したが失敗し、pg-rex01に対するreset処理を繰り返し実行 | 『5.11.2 復旧』 STEP1 |
| 【msPostgresql】  ・停止状態[[25]](#footnote-25)  【vip-master】  ・停止状態25  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態25  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態[[26]](#footnote-26)  【clnDiskd1】  ・停止状態26 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・正常状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |
| NONE | Master | PG-REXリソースを停止させたが停止処理に失敗 | pg-rex01のPG-REXリソース停止処理失敗を受け、pg-rex02からpg-rex01へのreset処理を実施し、pg-rex01停止後PG-REXリソースをMasterへ昇格  PG-REXリソースの昇格を受けIPaddr2(vip-master、vip-rep)を起動 | 『5.11.2 復旧』 STEP3 |
| 【msPostgresql】  ・pg-rex02にフェイルオーバ  【vip-master】  ・pg-rex02にフェイルオーバ  【vip-slave】  ・pg-rex02にフェイルオーバ  【vip-rep】  ・停止状態  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(NONE)-pg-rex02(Master)の場合を図 5‑12に示します。

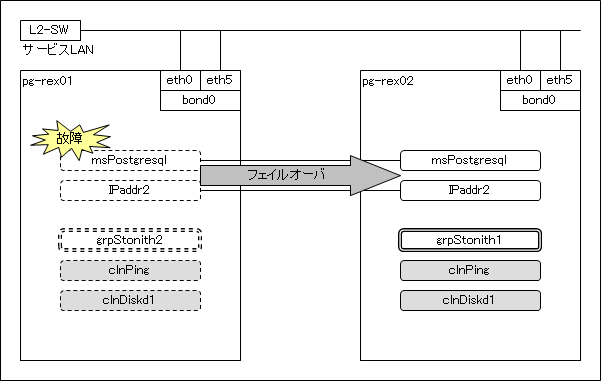


図 5‑12　リソース故障(stop/demote)状況

### 復旧

復旧手順を図 5‑13に示します。

クラスタ状態

復旧手順

STEP2

保守者介在処理[pg-rex02]

STEP3

ノード起動[pg-rex01]

STEP6

STEP7

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

強制電源断[pg-rex01]

STEP4

STEP5

pg-rex01(UNCLEAN) - pg-rex02(Slave)

ノード状態・リソース状態確認[pg-rex02]

pg-rex01(NONE) - pg-rex02(Master)

保守者へ報告

reset処理成功の場合

reset処理失敗の場合

図 5‑13　リソース故障(demote/stop)復旧手順

各STEPについて説明します。

STEP1： 強制電源断 [pg-rex01]

pg-rex01の電源ボタンを押下し、電源を停止します。

STEP2： 保守者介在処理 [pg-rex02]

手動でpg-rex01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。

pg-rex02で".\*Succeeded to STONITH.\*"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 **pg-rex02 info: Succeeded to STONITH (off) pg-rex01 by a human.**

pg-rex02でstonith\_adminコマンドを以下のとおり実施します。

# stonith\_admin -C *pg-rex01*

STEP3： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP4： ノード起動 [pg-rex01]

pg-rex01の電源ボタンを押下し、ノードを起動します。

STEP5： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所

パターン1： 故障発生時のpg-rex01の状態が"NONE"の場合  
pg-rex01のリソース停止処理失敗を受け、STONITH機能によるreset処理が実施され、正常に終了

パターン2： 故障発生時のpg-rex01の状態が"UNCLEAN"の場合  
pg-rex01のリソース停止処理失敗を受け、STONITH機能によるreset処理が実施されたが、reset処理が失敗

STEP6： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01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 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## リソース故障(vip-master)

この節では、リソース故障(vip-master)時における一次対応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)時における、各ノードの状態を表 5‑15に示します。

表 5‑15　リソース故障(vip-master)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| OUS | Master | リソース故障(vip-master)を検知し、PG-REXリソース(msPostgresql)が正常終了 | pg-rex01のPG-REXリソース(msPostgresql)の停止を受け、pg-rex02のPG-REXリソース(msPostgresql)がMasterへ昇格  PG-REXリソース(msPostgresql)の昇格を受け、IPaddr2リソース(vip-master、vip-rep)が起動 | 『5.12.2 復旧』 STEP1 |
| 【msPostgresql】  ・pg-rex02へフェイルオーバ  【vip-master】  ・故障検知してpg-rex02へフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02へフェイルオーバ  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(OUS)-pg-rex02(Master)の場合を図 5‑14に示します。

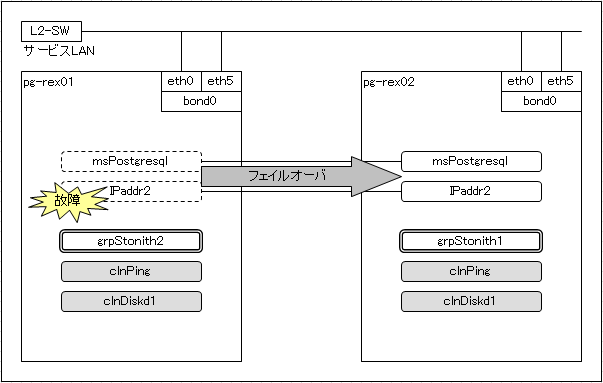


図 5‑14　リソース故障(vip-master)状況

### 復旧

復旧手順を図 5‑15に示します。

クラスタ状態

復旧手順

pg-rex01(OUS) - pg-rex02(Master)

STEP2

Pacemaker停止[pg-rex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex02]

ノード状態確認[pg-rex02]

STEP4

図 5‑15　リソース故障(vip-master)復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP2： Pacemaker停止 [pg-rex01]

保守者の作業中に、PG-REXリソースが再起動しないようpg-rex01のPacemakerを停止します。停止する手順は『4.7 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 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  IPaddr2(vip-master)故障が発生

STEP5： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01のPG-REXリソースをSlaveとして起動します。

STEP6： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## リソース故障(vip-rep)

この節では、リソース故障(vip-rep)時における一次対応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)時における、各ノードの状態を表 5‑16に示します。

表 5‑16　リソース故障(vip-rep)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| Master | Slave | リソース故障(vip-rep)を検知し、IPaddr2リソース(vip-rep)が再起動する |  | 『5.13.2 復旧』 STEP1 |
| 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・停止状態  【vip-rep】  ・故障検知して再起動  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・正常状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(Master)-pg-rex02(Slave)の場合を図 5‑16に示します。

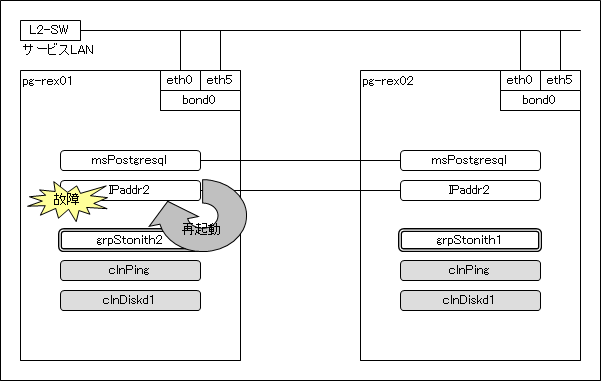


図 5‑16　リソース故障(vip-rep)状況

### 復旧

復旧手順を図 5‑17に示します。

クラスタ状態

復旧手順

pg-rex01(Master) - pg-rex02(Slave)

STEP2

保守者へ報告

STEP4

リソース(vip-rep)のフェイルカウントクリア[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex01]

STEP3

図 5‑17　リソース故障(vip-rep)復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

**Slaves: [ pg-rex02 ]**

：（略）

Migration summary:

\* Node pg-rex01:

***故障回数表示***

\* Node pg-rex02:

STEP2： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex01でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(Master)-pg-rex02(Slave)
* 故障箇所  
  IPaddr2(vip-rep)故障が発生

STEP3： リソース(vip-rep)のフェイルカウントクリア [pg-rex01]

pg-rex01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 pg-rex01

Cleaning up vip-rep on pg-rex01

Waiting for 1 replies from the CRMd...

STEP4： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01pg-rex02]

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Migration summary:

\* Node pg-rex01:

***表示無し***

\* Node pg-rex02:

## リソース故障(vip-slave)

この節では、リソース故障(vip-slave)時における一次対応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)時における、各ノードの状態を表 5‑17に示します。

表 5‑17　リソース故障(vip-slave)時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| Master | Slave | pg-rex02のIPaddr2リソース(vip-slave)の停止を受け、IPaddr2リソース(vip-slave)がpg-rex01に切り替わる | 故障を検知する | 『5.14.2 復旧』 STEP1 |
| 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・故障検知してpg-rex01へ切り替わる  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(Master)-pg-rex02(Slave)の場合を図 5‑18に示します。

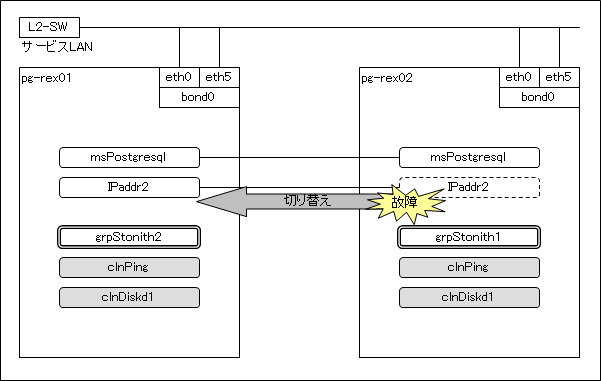


図 5‑18　リソース故障(vip-slave)状況

### 復旧

復旧手順を図 5‑19に示します。

クラスタ状態

復旧手順

pg-rex01(Master) - pg-rex02(Slave)

STEP2

保守者へ報告

STEP4

リソース(vip-slave)の切り替え[pg-rex02]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

リソース状態確認[pg-rex01]

STEP3

図 5‑19　リソース故障(vip-slave)復旧手順

各STEPについて説明します。

STEP1： リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 pg-rex02 ]

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

**Slaves: [ pg-rex02 ]**

：（略）

Migration summary:

\* Node pg-rex01:

\* Node pg-rex02:

***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。

* 報告時点でのサービス稼働状況  
  pg-rex01でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(Master)-pg-rex02(Slave)
* 故障箇所  
  IPaddr2(vip-slave)故障が発生

STEP3： リソース(vip-slave)の切り替え [pg-rex02]

pg-rex02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 pg-rex02

Cleaning up vip-slave on pg-rex02

Waiting for 1 replies from the CRMd...

STEP4： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01pg-rex02]

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Migration summary:

\* Node pg-rex01:

\* Node pg-rex02:

***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における一次対応について説明します。

### 故障時のクラスタ状態

内蔵ディスク故障時における、各ノードの状態を表 5‑18に示します。

表 5‑18　内蔵ディスク故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| UNCL  EAN | Slave | 内蔵ディスク故障を検知し、PG-REXリソースを停止させたが停止処理に失敗 | pg-rex01のPG-REXリソース停止処理失敗を受け、reset処理を実施したが失敗し、pg-rex01に対するreset処理を繰り返し実行 | 『5.15.2 復旧』 STEP1 |
| 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・故障検知状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・正常状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |
| NONE | Master | 内蔵ディスク故障を検知し、PG-REXリソースを停止させたが停止処理に失敗 | pg-rex01のPG-REXリソース停止処理失敗を受け、pg-rex02からpg-rex01へのreset処理を実施し、pg-rex01停止後PG-REXリソースをMasterへ昇格  PG-REXリソースの昇格を受けIPaddr2(vip-master、vip-rep)を起動 | 『5.15.2 復旧』 STEP3 |
| 【msPostgresql】  ・pg-rex02にフェイルオーバ  【vip-master】  ・pg-rex02にフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02にフェイルオーバ  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |
| OUS | Master | 内蔵ディスク故障を検知し、PG-REXリソースを正常に停止 | pg-rex01のPG-REXリソース(msPostgresql)の停止を受け、pg-rex02のPG-REXリソース(msPostgresql)がMasterへ昇格  PG-REXリソース(msPostgresql)の昇格を受け、IPaddr2リソース(vip-master、vip-rep)が起動 | 『5.15.2 復旧』STEP1 |
| 【msPostgresql】  ・pg-rex02にフェイルオーバ  【vip-master】  ・pg-rex02にフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02にフェイルオーバ  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・故障検知状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例を図 5‑20に示します。

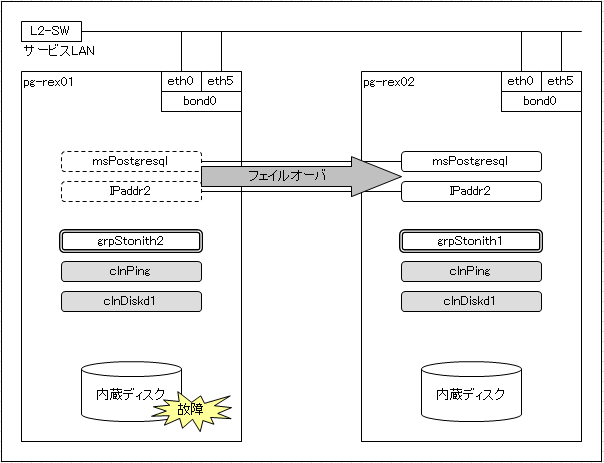


図 5‑20　内蔵ディスク故障状況

### 復旧

サービス状況の確認とpg-rex01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、pg-rex01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、pg-rex01(Slave) - pg-rex02(Master)となります。

復旧手順を図 5‑21に示します。

クラスタ状態

復旧手順

STEP2

保守者介在処理[pg-rex02]

STEP3

ノード起動[pg-rex01]

STEP6

STEP7

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

強制電源断[pg-rex01]

STEP4

STEP5

pg-rex01(UNCLEAN) - pg-rex02(Slave)

ノード状態・リソース状態確認[pg-rex02]

pg-rex01(NONE) - pg-rex02(Master)

保守者へ報告

reset処理成功の場合

reset処理未実施、

または失敗の場合

pg-rex01(OUS) - pg-rex02(Master)

図 5‑21　内蔵ディスク故障復旧手順

各STEPについて説明します。

STEP1： 強制電源断 [pg-rex01]

pg-rex01の電源ボタンを押下し、電源を停止します。

STEP2： 保守者介在処理 [pg-rex02]

手動でpg-rex01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。

pg-rex02で".\*Succeeded to STONITH.\*"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 **pg-rex02 info: Succeeded to STONITH (off) pg-rex01 by a human.**

pg-rex02でstonith\_adminコマンドを以下のとおり実施します。

# stonith\_admin -C *pg-rex01*

STEP3： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex02 ]

：（略）

STEP4： ノード起動 [pg-rex01]

pg-rex01の電源ボタンを押下し、ノードの起動を試みます。

STEP5： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所

pg-rex01の内蔵ディスク故障が発生

パターン1： 故障発生時のpg-rex01の状態が"NONE"の場合  
pg-rex01のリソース停止処理失敗を受け、STONITH機能によるreset処理が実施され、正常に終了

パターン2： 故障発生時のpg-rex01の状態が"UNCLEAN"の場合  
pg-rex01のリソース停止処理失敗を受け、STONITH機能によるreset処理が実施されたが、reset処理が失敗

パターン3： 故障発生時のpg-rex01の状態が"OUS"の場合  
pg-rex01のリソース停止処理が成功し正常終了。reset処理は実施していない。

STEP6： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01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 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## ノード故障

この節では、ノード故障時における一次対応について説明します。

### 故障時のクラスタ状態

ノード故障時における、各ノードの状態を表 5‑19に示します。

表 5‑19　ノード故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| UNCL EAN | Slave | OS故障、電源断、Pacemakerのプロセス故障により、Pacemaker停止状態 | pg-rex01のPacemaker停止を受け、reset処理を実施したが失敗し、pg-rex01に対するreset処理を繰り返し実行 | 『5.16.2 復旧』 STEP1 |
| 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・正常状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |
| NONE | Master | OS故障、電源断、Pacemakerのプロセス故障により、Pacemaker停止状態 | pg-rex01のPacemaker停止を受け、pg-rex02からpg-rex01へのreset処理を実施し、pg-rex01停止後PG-REXリソースをMasterへ昇格  PG-REXリソースの昇格を受けて、IPaddr2リソースが起動(vip-master、vip-rep) | 『5.16.2 復旧』 STEP3 |
| 【msPostgresql】  ・pg-rex02にフェイルオーバ  【vip-master】  ・pg-rex02にフェイルオーバ  【vip-slave】  ・停止状態  【vip-rep】  ・pg-rex02にフェイルオーバ  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(NONE)-pg-rex02(Master)の場合を図 5‑22に示します。

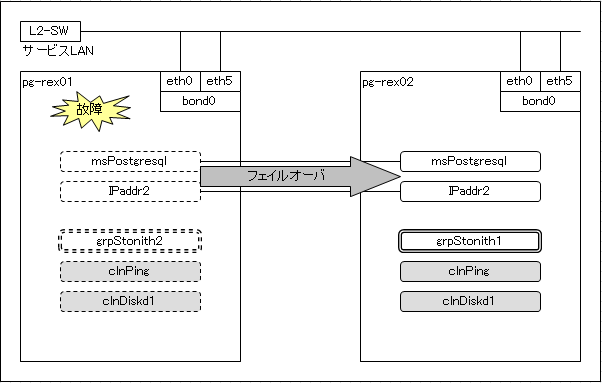


図 5‑22　ノード故障状況

### 復旧

復旧手順を図 5‑23に示します。

クラスタ状態

復旧手順

STEP2

保守者介在処理[pg-rex02]

STEP3

ノード起動[pg-rex01]

STEP6

STEP7

Pacemaker起動[pg-rex01]

ノード状態・リソース状態確認[pg-rex01]

保守者による故障復旧

STEP1

強制電源断[pg-rex01]

STEP4

STEP5

pg-rex01(UNCLEAN) - pg-rex02(Slave)

ノード状態・リソース状態確認[pg-rex02]

保守者へ報告

reset処理成功の場合

reset処理失敗の場合

pg-rex01(NONE) - pg-rex02(Master)

図 5‑23　ノード故障復旧手順

各STEPについて説明します。

STEP1： 強制電源断 [pg-rex01]

pg-rex01の電源ボタンを押下し、電源を停止します。

STEP2： 保守者介在処理 [pg-rex02]

手動でpg-rex01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。

pg-rex02で".\*Succeeded to STONITH.\*"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 **pg-rex02 info: Succeeded to STONITH (off) pg-rex01 by a human.**

pg-rex02でstonith\_adminコマンドを以下のとおり実施します。

# stonith\_admin -C *pg-rex01*

STEP3： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP4： ノード起動 [pg-rex01]

pg-rex01の電源ボタンを押下し、ノードの起動を試みます。

STEP5： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  pg-rex01のノードまたはPacemaker故障が発生

STEP6： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01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 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

## IC-LAN故障

この節では、IC-LAN故障時における一次対応について説明します。

### 故障時のクラスタ状態

IC-LAN故障時における、各ノードの状態を表 5‑20に示します。

表 5‑20　IC-LAN故障時のクラスタ状態

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| クラスタ状態 | | pg-rex01の状態 | pg-rex02の状態 | 復旧手順 遷移先STEP |
| pg-rex01 | pg-rex02 |
| Master | Slave | IC-LAN故障を受け、pg-rex01からpg-rex02へのreset処理を実施したが失敗 | IC-LAN故障を受け、pg-rex02からpg-rex01へのreset処理を実施したが失敗 | 『5.17.2 復旧』 STEP1-1 |
| 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・停止状態  【vip-rep】  ・正常状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・正常状態 (Slave)  【vip-master】  ・停止状態  【vip-slave】  ・正常状態  【vip-rep】  ・停止状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |
| Master | NONE | IC-LAN故障を受け、pg-rex01からpg-rex02へのreset処理を実施し正常終了 | pg-rex01からのreset処理が実施されPacemaker停止 | 『5.17.2 復旧』 STEP1-3 |
| 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith2】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 | 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith1】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 |
| NONE | Master | IC-LAN故障を受け、pg-rex01からpg-rex02へのreset処理を実施したが失敗し、pg-rex02からのreset処理が実施されPacemaker停止 | IC-LAN故障を受け、pg-rex02からpg-rex01へのreset処理を実施し、pg-rex01停止後リソースを正常に起動 | 『5.17.2 復旧』 STEP2-1 |
| 【msPostgresql】  ・停止状態  【vip-master】  ・停止状態  【vip-slave】  ・停止状態  【vip-rep】  ・停止状態  【grpStonith2】  ・停止状態  【clnPing】  ・停止状態  【clnDiskd1】  ・停止状態 | 【msPostgresql】  ・正常状態 (Master)  【vip-master】  ・正常状態  【vip-slave】  ・正常状態  【vip-rep】  ・正常状態  【grpStonith1】  ・正常状態  【clnPing】  ・正常状態  【clnDiskd1】  ・正常状態 |

故障発生時におけるクラスタ状態の例としてpg-rex01(Master)-pg-rex02(NONE)の場合を図 5‑24に示します。

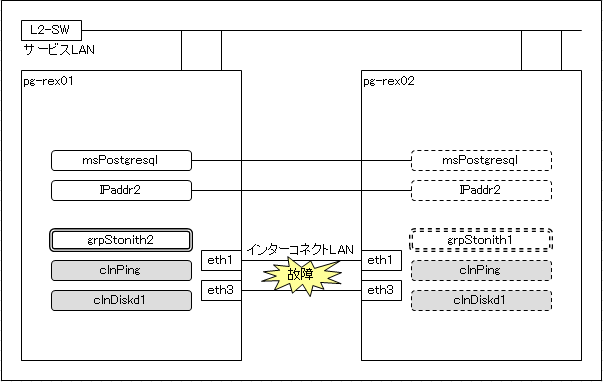


図 5‑24　IC-LAN故障状況

### 復旧

クラスタ状態がpg-rex01(Master)-pg-rex02(Slave)、または、pg-rex01(Master)-pg-rex02(NONE)の場合の復旧手順を図 5‑25に示します。

クラスタ状態

復旧手順

STEP1-2

保守者介在処理[pg-rex01]

STEP1-3

ノード起動[pg-rex02]

STEP1-6

STEP1-7

Pacemaker起動[pg-rex02]

ノード状態・リソース状態確認[pg-rex02]

保守者による故障復旧

STEP1-1

強制電源断[pg-rex02]

STEP1-4

STEP1-5

pg-rex01(Master) - pg-rex02(Slave)

ノード状態・リソース状態確認[pg-rex01]

pg-rex01(Master) - pg-rex02(NONE)

保守者へ報告

pg-rex01のreset  
処理成功の場合

pg-rex01のreset  
処理失敗の場合

図 5‑25　IC-LAN故障復旧手順①

各STEPについて説明します。

STEP1-1： 強制電源断 [pg-rex02]

pg-rex02の電源ボタンを押下し、電源を停止します。

STEP1-2： 保守者介在処理 [pg-rex01]

手動でpg-rex02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。

pg-rex01で".\*Succeeded to STONITH.\*"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 **pg-rex01 info: Succeeded to STONITH (off) pg-rex02 by a human.**

pg-rex01でstonith\_adminコマンドを以下のとおり実施します。

# stonith\_admin -C *pg-rex02*

STEP1-3： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex01 ]

**OFFLINE: [ pg-rex02 ]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex01**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex01 ]**

Stopped: [ pg-rex02 ]

：（略）

STEP1-4： ノード起動 [pg-rex02]

pg-rex02の電源ボタンを押下し、ノードを起動します。

STEP1-5： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex01でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(Master)-pg-rex02(NONE)  
  pg-rex02でPacemaker停止中
* 故障箇所  
  IC-LAN故障が発生

STEP1-6： Pacemaker起動 [pg-rex02]

pg-rex02のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

STEP1-7： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [** pg-rex01 **pg-rex02 ]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01

：（略）

Resource Group: grpStonith1

prmStonith1-1 (stonith:external/stonith-helper): **Started pg-rex02**

prmStonith1-2 (stonith:external/ipmi): **Started pg-rex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex01 ]

**Slaves: [ pg-rex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pg-rex01 **pg-rex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pg-rex01 **pg-rex02 ]**

：（略）

クラスタ状態がpg-rex01(NONE)-pg-rex02(Master)の場合の復旧手順を図 5‑26に示します。

クラスタ状態

復旧手順

STEP2-2

Pacemaker起動[pg-rex01]

保守者による故障復旧

STEP2-1

STEP2-3

STEP2-4

pg-rex01(NONE) - pg-rex02(Master)

ノード状態・リソース状態確認[pg-rex01]

保守者へ報告

ノード状態・リソース状態確認[pg-rex02]

図 5‑26　IC-LAN故障復旧手順②

各STEPについて説明します。

STEP2-1： ノード状態・リソース状態確認 [pg-rex02]

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

============

Last updated: *日時表示*

：（略）

============

Online: [ pg-rex02 ]

**OFFLINE: [ pg-rex01 ]**

vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pg-rex02 ]**

Stopped: [ pg-rex01 ]

：（略）

STEP2-2： 保守者へ報告

以下の内容を報告します。

* 報告時点でのサービス稼働状況  
  pg-rex02でサービス稼働中
* 報告時点でのクラスタ状態  
  pg-rex01(NONE)-pg-rex02(Master)  
  pg-rex01でPacemaker停止中
* 故障箇所  
  IC-LAN故障が発生

STEP2-3： Pacemaker起動 [pg-rex01]

pg-rex01のPacemakerを起動します。起動する手順は『4.3 Slaveの起動』を参照してください。

※pg-rex02でPG-REXリソースがMasterとして稼働中のため、pg-rex01のPG-REXリソースをSlaveとして起動します。

STEP2-4： ノード状態・リソース状態確認 [pg-rex01]

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

============

Last updated: *日時表示*

：（略）

============

**Online: [ pg-rex01** pg-rex02 **]**

vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02

vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02

Resource Group: grpStonith2

prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**

prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pg-rex02 ]

**Slaves: [ pg-rex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pg-rex01** pg-rex02 **]**

Clone Set: clnPing [prmPing]

**Started: [ pg-rex01** pg-rex02 **]**

：（略）

# 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)の時に監視者が行う操作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時に、監視者が行う手順について記述します。

### 前提条件

* 監視者は、バックアップ取得対象サーバで作業を実施します。
* 各コマンドはpostgresユーザで実行します。
* ここでは、Masterをpg-rex01、Slaveをpg-rex02として説明します。

### 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』[[27]](#footnote-27)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。以下のコマンドでは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。

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、『付録B アーカイブログの削除』を参照してください。

### 前提条件

* 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。

### PostgreSQLアーカイブログの削除

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

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

運用を続けていくうちにアーカイブディレクトリのファイルがディスク容量を圧迫する可能性があります。定期的に以下の手順に従って、不要なファイルを削除してください。

1. 不要なアーカイブログを移動

アーカイブディレクトリに存在する不要なアーカイブログをアーカイブログ格納ディレクトリ直下にディレクトリを作成して移動します。

本作業はrootユーザ、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で実行します。以下の手順では、pg-rex\_tools.confのPEER\_NODE\_SSH\_PASS\_MODEとBACKUP\_NODE\_SSH\_PASS\_MODEの設定値がmanualでpostgresユーザでの実行の場合を示します。

|  |
| --- |
| $ pg-rex\_archivefile\_delete -m $OLDEST\_BACKUP  **\*\*\*\* 1. 実行準備 \*\*\*\***  **mv 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:** **← 相手ノードのpostgresのパスワードを入力し[Enter]キーを押下**  **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**  **postgres@172.20.144.38's password:** **← バックアップを格納しているノードのpostgresのパスワードを入力し[Enter]キーを押下**  **パスワードが入力されました**  **"00000051000000010000003A"**  **自身のノード "pg-rex01" 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"**  **相手先のノード "pg-rex02" 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/arc1/20140224\_150226" を作成しました**  **-- mv -- 0000004F0000000100000035.00000020.backup**  **:(略)**  **アーカイブログの移動に成功しました**  **mv モード実行のため、移動したファイルは "/dbfp/pgarch/arc1/20140224\_150226" に格納されています** |

【注意】

「-m」オプションではなく「-r」オプションを使用すると、このツール実行時にアーカイブログが削除されるため、(2)の手順が不要になります。

1. 不要なアーカイブログが格納されているディレクトリの削除

(1)で不要なアーカイブログを移動したディレクトリを削除します。

本作業はpostgresユーザで行います。

|  |
| --- |
| $ rm -rf /dbfp/pgarch/arc1/20140224\_150226 |

## 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。

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、『付録C 計画的な系切り替え』を参照してください。

### 前提条件

* ここでは作業開始時点のMasterをpg-rex01、Slaveをpg-rex02として説明します。
* 各コマンドはrootユーザで実行します。

### 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、pg-rex\_tools.confのPEER\_NODE\_SSH\_PASS\_MODEの設定値がmanualの場合にpg-rex01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:** **← 相手ノードのrootのパスワードを入力し[Enter]キーを押下**  **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 : pg-rex01 -> pg-rex02**  **Slave : pg-rex02 -> pg-rex01** **← 系切り替え後のクラスタ状態を確認**  **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** **← [y]を入力し[Enter]キーを押下** |

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。『5章 故障対応』に従い、手動復旧を試みてください。

|  |
| --- |
| **3. vacuumdb コマンドの実行**  **…[OK]**  **4. CHECKPOINT の実行**  **…[OK]**  **\*\*\*\* 系切り替えを実行 \*\*\*\***  **5. Pacemaker の監視を停止**  **…[OK]**  **6. Master (pg-rex01) の PostgreSQL を停止**  **…[OK]**  **7. Pacemaker の監視を再開し系切り替えを実行**  **…[OK]**  **8. pg-rex02 が新 Master になったことを確認**  **\*\*\*\* pg-rex02 が Master として起動しました \*\*\*\***  **9. pg-rex01 の Pacemaker を停止**  **…[OK]**  **10. pg-rex01 で Slave を起動**  **receiving incremental file list**  **./**  **00000011000000000000000C**  **00000012000000000000000E**  **00000013.history**  **sent 71 bytes received 33561066 bytes 7458030.44 bytes/sec**  **total size is 503327160 speedup is 15.00**  **\*\*\*\* pg-rex01 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 : pg-rex02**  **Slave : pg-rex01** **← 現在(系切り替え完了後)のクラスタ状態を確認** |

【注意】

Pacemakerの監視を停止中(上記の5～7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

本節では、MasterおよびSlaveの計画されたメンテナンスを実施する時の操作手順を記述します。

### 前提条件

* ここでは作業開始時点のMasterをpg-rex01、Slaveをpg-rex02として説明します。
* 各コマンドはrootユーザで実行します。

### Masterのメンテナンス

Masterの計画的なメンテナンスにおける操作手順を以下に示します。

1. 系切り替え実行

Masterをpg-rex02に切り替えるため、pg-rex01の運用を停止します。

Masterの停止手順については、『4.6 Masterの停止』を参照してください。

1. 系切り替え確認

pg-rex02でcrm\_monコマンドを実行し、以下を確認します。

* pg-rex01の状態が"OFFLINE"になっていること。
* clnPing、clnDiskdリソースが、pg-rex02上だけ"Started"になっていること。
* msPostgresqlリソースが、pg-rex02上でMasterとして動作していること。

|  |
| --- |
| [pg-rex02] # crm\_mon -fA  Last updated: *日時表示*  Last change: *日時表示*  Stack: corosync  ：（略）  Online: [ pg-rex02 ]  **OFFLINE: [ pg-rex01 ]**  ：（略）  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pg-rex02 ]**  Stopped: [ pg-rex01 ]  Clone Set: clnDiskd1 [prmDiskd1]  **Started: [ pg-rex02 ]**  Stopped: [ pg-rex01 ]  Clone Set: clnPing [prmPing]  **Started: [ pg-rex02 ]**  Stopped: [ pg-rex01 ]  ：（略） |

1. pg-rex01のメンテナンス作業

pg-rex01で、計画されたメンテナンス作業を実施します。

1. pg-rex01のPacemakerを起動

pg-rex01をSlaveとして起動します。Slaveの起動手順については、『4.3 Slaveの起動』を参照してください。

1. 系の切り替え(戻し)

作業開始時点に系の状態を戻すため、『6.3 計画的な系切り替え』に従い、pg-rex01をMaster、pg-rex02をSlaveとなるように切り替えます。

### Slaveのメンテナンス

Slaveの計画的なメンテナンスにおける操作手順を以下に示します。

1. pg-rex02のPacemakerの停止

Slaveの運用を停止させます。

Slaveの停止手順については、『4.5 Slaveの停止』を参照してください。

1. pg-rex02のメンテナンス作業

pg-rex02で、計画されたメンテナンス作業を実施します。

1. pg-rex02のPacemakerを起動

『4.3 Slaveの起動』の手順に従い、pg-rex02のPacemakerを起動します。

1. 起動と停止

本付録では、コマンドを直接実行して起動と停止を行う手順を説明します。PG-REX運用補助ツールを用いて起動と停止を行いたい場合は、『4章 起動と停止』をご確認ください。

* 1. 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、『A.2 Masterの起動』を参照してください。
* Slaveの起動手順については、『A.3 Slaveの起動』を参照してください。
  1. 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、pg-rex01をMasterとして起動します。

1. (Masterとして起動する)pg-rex01で、Pacemakerが起動していないことを確認します。起動している場合は、『A.6 Masterの停止』の手順どおりにPacemakerを停止します。本作業はrootユーザで行います。

|  |
| --- |
| # initctl status pacemaker.combined  **pacemaker.combined stop/waiting** |

1. 稼働中のMasterが存在しないことを確認するために、pg-rex02でcrm\_monコマンドを実行します。稼働中のMasterが存在する場合は、以下のような出力結果となります。その場合は、『A.6 Masterの停止』の手順どおりにMasterを停止します。本作業はrootユーザで行います。

|  |
| --- |
| # crm\_mon –fA  ：(略)  **Master/Slave Set: msPostgresql [pgsql]**  **Masters: [ pg-rex02 ]**  **Stopped: [ pg-rex01 ]**  ：(略) |

1. ベースバックアップからMasterを起動する場合に限り、pg-rex01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 -w 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。

1. pg-rex01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. 2回目以降のMaster起動の場合、以下の手順を実行します。本作業はrootユーザで行います。
   1. pg-rex01でMasterを起動します。ただし、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(7)以降記載のコマンドを実行します。

|  |
| --- |
| # initctl start pacemaker.combined  **pacemaker.combined start/running, process 52430** |

1. 初回起動時、もしくは新しいcrmファイルを反映させてMasterを起動する場合、以下の手順を実行します。本作業はrootユーザで行います。
   * 1. pg-rex01とpg-rex02で、既存のPacemakerの設定を削除します。

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

* + 1. pg-rex01でMasterを起動します。

|  |
| --- |
| # initctl start pacemaker.combined  **pacemaker.combined start/running, process 52430** |

* + 1. pg-rex01のPacemakerが起動したことを確認します。

|  |
| --- |
| # crm\_mon -fA  Last updated: *日時表示*  Last change: *日時表示*  Stack: corosync  Current DC: pg-rex01 (*UUID表示*) - partition WITHOUT quorum  ：(略)  1 Nodes configured  0 Resources configured  **Online: [ pg-rex01 ]**  Node Attributes:  **\* Node pg-rex01:**  **+ ringnumber\_0 : 192.168.1.1 is UP**  **+ ringnumber\_1 : 192.168.3.1 is UP**  Migration summary:  **\* Node pg-rex01:** |

* + 1. pg-rex01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-REX9.4\_pm\_crmgen\_env.crm |

コマンド実行時に以下のようなWARNメッセージとERROR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 90  WARNING: fencing\_topology: target pg-rex02 not a node  WARNING: fencing\_topology: target pg-rex02 not a node  WARNING: rsc\_location-grpStonith2-4: referenced node pg-rex02 does not exist |

1. pg-rex01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  Last updated: *日時表示*  Last change: *日時表示*  Stack: corosync  Current DC: pg-rex01 (*UUID表示*) - partition WITHOUT quorum  ：(略)  2 Nodes configured  13 Resources configured  **Online: [ pg-rex01 ]**  OFFLINE: [ pg-rex02 ]  vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex01**  vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex01**  Resource Group: grpStonith2  prmStonith2-1 (stonith:external/stonith-helper): **Started pg-rex01**  prmStonith2-2 (stonith:external/ipmi): **Started pg-rex01**  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pg-rex01 ]**  Stopped: [ pg-rex02 ]  Clone Set: clnDiskd1 [prmDiskd1]  **Started: [ pg-rex01 ]**  Stopped: [ pg-rex02 ]  Clone Set: clnPing [prmPing]  **Started: [ pg-rex01 ]**  Stopped: [ pg-rex02 ]  Node Attributes:  **\* Node pg-rex01:**  + default\_ping\_set : **100**  + diskcheck\_status\_internal : **normal**  + master-pgsql : **1000**  + pgsql-data-status : **LATEST**  + pgsql-master-baseline : 0000000008000090  + pgsql-status : **PRI**  + ringnumber\_0 : **192.168.1.1 is UP**  + ringnumber\_1 : **192.168.3.1 is UP**  Migration summary:  **\* Node pg-rex01:** |

* 一度もSlaveを起動したことがない場合、pg-rex02の情報や、Stoppedの行は表示されません。

1. pg-rex01で以下の操作を行い、MasterのPostgreSQL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。

(MasterのPostgreSQLプロセスの存在を確認)

|  |
| --- |
| # ps -ef | grep postgres  ：(略)  **postgres** 2414 1 0 15:11 ? 00:00:00 **/usr/pgsql-9.4/bin/postgres** -D /pgdata/data -c config\_file=/pgdata/data/postgresql.conf -p 5432  **postgres** 2439 2414 0 15:11 ? 00:00:00 **postgres: logger process**  **postgres** 2459 2414 0 15:11 ? 00:00:00 **postgres: checkpointer process**  **postgres** 2460 2414 0 15:11 ? 00:00:00 **postgres: writer process**  **postgres** 2461 2414 0 15:11 ? 00:00:00 **postgres: stats collector process**  **postgres** 3269 2414 0 15:11 ? 00:00:00 **postgres: wal writer process**  **postgres** 3270 2414 0 15:11 ? 00:00:00 **postgres: autovacuum launcher process**  **postgres** 3271 2414 0 15:11 ? 00:00:00 **postgres: archiver process** last was 00000004.history  ：(略) |

(MasterのPostgreSQLに接続できることを確認)

|  |
| --- |
| # su - postgres -c "psql -l"  データベース一覧  名前 | 所有者 | エンコーディング | 照合順序 | Ctype(変換演算子) | アクセス権  -----------+----------+------------------+----------+-------------------+-----------------------  **postgres | postgres | UTF8 | C | C |**  **template0 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/postgres**  **template1 | postgres | UTF8 | C | C | =c/postgres +**  ：(略) |

(正常起動を意味するログメッセージが、MasterのPostgreSQLのサーバログに出力されていることを確認)

|  |
| --- |
| # tail */var/log/messages* **←ログファイルのパスは、設定に応じて読み替えること**  ：(略)  ~ pg-rex01 <local0.info> postgres[2414]: [5-1] ~ LOG: 00000: **database system is ready to accept connections**  ：(略) |

* 正常起動時であっても、/var/log/messagesに上記ログメッセージが表示される前に、

「FATAL: ~ the database system is starting up」

「FATAL: ~ could not connect to the primary server: could not connect to server: XXX[[30]](#footnote-30)

Is the server running on host "192.168.2.3" and accepting

TCP/IP connections on port 5432 ?」

というFATALメッセージが出力されることがあります。

* 1. 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、pg-rex02をSlaveとして起動します。

1. (Slaveとして起動する)pg-rex02で、Pacemakerが起動していないことを確認します。起動している場合は、『A.5 Slaveの停止』の手順どおりにPacemakerを停止します。本作業はrootユーザで行います。

|  |
| --- |
| # initctl status pacemaker.combined  **pacemaker.combined stop/waiting** |

1. 稼働中のMasterが存在することをpg-rex01で確認します。確認手順は、『A.2 Masterの起動』の(8)、(9)を参照してください。
2. 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。
3. 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。

* pg-rex02が初回起動時の場合。
* pg-rex02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。
* pg-rex02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。

* pg-rex02に存在するDBクラスタが非常に古い場合。

リカバリに必要なWALファイルがどこにも存在しないほどDBクラスタが古い場合(例えば、アーカイブログの保管期間3日のシステムで、1週間前のDBクラスタからSlaveを起動する場合)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。

* pg-rex01で稼働中のMasterが、ベースバックアップから起動されていた場合。

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、pg-rex01からバックアップを取得し、pg-rex02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、pg-rex02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。

|  |
| --- |
| $ rm -rf $PGDATA  $ rm -rf /dbfp/pgxlog/pg\_xlog  $ pg\_basebackup -h *192.168.2.3* -U repuser -D $PGDATA  –-xlogdir=/dbfp/pgxlog/pg\_xlog -X stream -P  **← -hにはMaster側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. pg-rex01からpg-rex02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 *192.168.2.3*:/dbfp/pgarch/arc1/ /dbfp/pgarch/arc1/  **← 接続先はMaster側レプリケーション受付用の仮想IPアドレス** |

1. pg-rex02で起動禁止フラグのファイルを(存在する場合)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. pg-rex02でSlaveを起動します。本作業はrootユーザで行います。

|  |
| --- |
| # initctl start pacemaker.combined  **pacemaker.combined start/running, process 70114** |

1. pg-rex02で以下の操作を行い、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 -fA1  Last updated: *日時表示*  Last change: *日時表示*  Stack: corosync  Current DC: pg-rex01 (*UUID表示*) - partition with quorum  ：(略)  **Online**: [ pg-rex01 **pg-rex02** ]  vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex02**  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): Started pg-rex01  vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex01  Resource Group: grpStonith2  prmStonith2-1 (stonith:external/stonith-helper): Started pg-rex01  prmStonith2-2 (stonith:external/ipmi): Started pg-rex01  Resource Group: grpStonith1  prmStonith1-1 (stonith:external/stonith-helper): **Started pg-rex02**  prmStonith1-2 (stonith:external/ipmi): **Started pg-rex02**  Master/Slave Set: msPostgresql [pgsql]  Masters: [ pg-rex01 ]  **Slaves: [ pg-rex02 ]**  Clone Set: clnDiskd1 [prmDiskd1]  Started: [ pg-rex01 **pg-rex02** ]  Clone Set: clnPing [prmPing]  Started: [ pg-rex01 **pg-rex02** ]  Node Attributes:  \* Node pg-rex01:  + default\_ping\_set : 100  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 0000000002000090  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP  **\* Node pg-rex02:**  + default\_ping\_set : **100**  + 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 pg-rex01:  **\* Node pg-rex02:** |

1. pg-rex02で以下の操作を行い、SlaveのPostgreSQL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。

(SlaveのPostgreSQLプロセスの存在を確認)

|  |
| --- |
| # ps -ef | grep postgres  ：(略)  **postgres** 22476 1 0 17:01 ? 00:00:00 **/usr/pgsql-9.4/bin/postgres -D /pgdata/data -c config\_file=/pgdata/data/postgresql.conf -p 5432**  **postgres** 22501 22476 0 17:01 ? 00:00:00 **postgres: logger process**  **postgres** 22504 22476 0 17:01 ? 00:00:00 **postgres: startup process waiting for 000000020000000000000006**  **postgres** 22518 22476 0 17:01 ? 00:00:00 **postgres: checkpointer process**  **postgres** 22519 22476 0 17:01 ? 00:00:00 **postgres: writer process**  **postgres** 22520 22476 0 17:01 ? 00:00:00 **postgres: stats collector process**  **postgres** 22522 22476 0 17:01 ? 00:00:00 **postgres: wal receiver process**  ：(略) |

(SlaveのPostgreSQLに接続できることを確認)

|  |
| --- |
| # su - postgres -c "psql -l"  データベース一覧  名前 | 所有者 | エンコーディング | 照合順序 | Ctype(変換演算子) | アクセス権  -----------+----------+------------------+----------+-------------------+-----------------------  **postgres | postgres | UTF8 | C | C |**  **template0 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/postgres**  **template1 | postgres | UTF8 | C | C | =c/postgres +**  ：(略) |

(PostgreSQLのストリーミングレプリケーション開始を意味するログメッセージが、SlaveのPostgreSQLのサーバログに出力されていることを確認)

|  |
| --- |
| # tail */var/log/messages* **←ログファイルのパスは、設定に応じて読み替えること**  ：(略)  ~ **pg-rex02** <local0.info> postgres[2019]: [5-1] ~ LOG: 00000: **started streaming WAL from primary at 0/4000000 on timeline *2***  ：(略) |

1. pg-rex01で以下の操作を行い、レプリケーションが開始されたことを確認します。本作業はrootユーザで行います。

(MasterのPostgreSQLのwal senderプロセスの存在を確認)

|  |
| --- |
| # ps -ef | grep postgres  ：(略)  **postgres** **16409 23710 0 17:01 ? 00:00:00 postgres: wal sender process repuser 192.168.2.2(38794) streaming 0/60000C8**  ：(略) |

(SlaveがMasterに追いついて、同期レプリケーションが開始したことを意味するログメッセージが、MasterのPostgreSQLのサーバログに出力されていることを確認)

|  |
| --- |
| # tail */var/log/messages* ←ログファイルのパスは、設定に応じて読み替えること  ：(略)  ~ **pg-rex01** <local0.info> **postgres**[16409]: [30-1] ~ LOG: 00000: **standby "pg-rex02" is now the synchronous standby with priority 1**  ：(略) |

* 1. 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、『A.6 Masterの停止』を参照してください。
* Slaveの停止手順については、『A.5 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。

* 1. 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。

|  |
| --- |
| # initctl stop pacemaker.combined  **pacemaker.combined stop/waiting** |

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** |

* 1. 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。

|  |
| --- |
| # initctl stop pacemaker.combined  **pacemaker.combined stop/waiting** |

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** |

1. アーカイブログの削除

本付録では、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。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したい場合は、『6.2 アーカイブログの削除』をご確認ください。

* 1. 前提条件
* 各コマンドはpostgresユーザで実行します。
* ここでは管理する最古のバックアップの納先を、$OLDEST\_BACKUPとします。
  1. PostgreSQLアーカイブログの削除

PostgreSQLのアーカイブログ、バックアップ履歴ファイル[[31]](#footnote-31)、タイムライン履歴ファイル[[32]](#footnote-32)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

運用を続けていくうちにアーカイブディレクトリのファイルがディスク容量を圧迫する可能性があります。定期的に以下の手順に従って、不要なファイルを削除してください。

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。

|  |
| --- |
| [~] $ 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: 2014-02-17 17:48:19 JST  LABEL: pg\_basebackup base backup |

1. Slaveに必要な最も古いWALファイル名を取得

pg-rex02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。

|  |
| --- |
| [pg-rex02] $ 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ファイルを削除します。

以下のコマンドはpg-rex01を想定した例になります。以下の手順をpg-rex01で実行後、必ずpg-rex02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。そのファイルより先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ファイル名と前半8桁が同じタイムライン履歴ファイルは削除しないものとします。

(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。

|  |
| --- |
| [pg-rex01] $ ls -l /dbfp/pgarch/arc1  ~ **000000710000006A0000004B**  ~ **000000710000006A0000004B.00000020.backup**  ~ **00000072.history**  ~ **000000720000006A0000004C**  ~ **000000720000006A0000004D**  ~ **000000720000006A0000004E**  ~ **000000720000006A0000004F**  ~ 00000073.history **←削除対象としない**  ~ **000000730000006A0000004C**  ~ **000000730000006A0000004D**  ~ 000000730000006A0000004E **←チェックしたファイル**  ~ 000000730000006A0000004E.000013D8.backup  ~ 000000730000006A0000004F  ~ 000000730000006A00000050  ~ 00000074.history  ~ 000000740000006A00000050  ~ 000000740000006A00000051  ~ 00000075.history  ~ 000000750000006A00000051 |

1. 計画的な系切り替え

本付録では、PG-REX運用補助ツールを使用せずに、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。PG-REX運用補助ツールを用いて系切り替えを実施したい場合は、『6.3 計画的な系切り替え』をご確認ください。なお、本付録では、Slaveの再組み込みを実施する前にMasterのベースバックアップを取得する手順について記述します。

* 1. 前提条件
* ここでは作業開始時点のMasterをpg-rex01、Slaveをpg-rex02として説明します。
  1. 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

Masterをpg-rex02に切り替えます。msPostgresqlのリソースをフェイルオーバさせます。

|  |
| --- |
| [pg-rex01] # crm\_resource -M -r msPostgresql -N pg-rex02 -f -Q |

1. フェイルオーバ確認

pg-rex01でcrm\_monコマンドを実行し、以下を確認します。

* pg-rex01のmsPostgresqlのリソースの状態が"Stopped"になっていること。
* msPostgresqlリソースが、pg-rex02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、pg-rex02上で起動していること。

|  |
| --- |
| [pg-rex01] # crm\_mon -fA   * Last updated: *日時表示* * Last change: *日時表示* * Stack: corosync * Current DC: pg-rex01 (*UUID表示*) - partition with quorum   ：（略）  Online: [ pg-rex01 pg-rex02 ]  vip-slave (ocf::heartbeat:IPaddr2): Started pg-rex02  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): **Started pg-rex02**  vip-rep (ocf::heartbeat:IPaddr2): **Started pg-rex02**  ：（略）  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pg-rex02 ]**  **Stopped: [ pg-rex01 ]**  ：（略）  Node Attributes:  \* Node pg-rex01:  + default\_ping\_set : 100  + diskcheck\_status\_internal : normal  + master-pgsql : -INFINITY  + pgsql-data-status : DISCONNECT  + pgsql-status : **STOP**  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP  ：（略）  Migration summary:  \* Node pg-rex01:  \* Node pg-rex02: \* Node pg-rex02: |

1. msPostgresqlのリソースの再組み込み準備

msPostgresqlの再組み込みの準備をします。再組み込みの準備には、pg-rex02からベースバックアップの取得、pg\_xlogの構成の修正、pg-rex02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。

|  |
| --- |
| [pg-rex01] # su - postgres  [pg-rex01] $ rm -rf $PGDATA  [pg-rex01] $ rm -rf /dbfp/pgxlog/pg\_xlog  [pg-rex01] $ pg\_basebackup -h *192.168.2.3* -U repuser -D $PGDATA  --xlogdir=/dbfp/pgxlog/pg\_xlog -X stream -P  **← -hにはMaster側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**  [pg-rex01] $ rsync -av *192.168.2.3*:/dbfp/pgarch/arc1/ /dbfp/pgarch/arc1/  **← 接続先はMaster側レプリケーション受付用の仮想IPアドレス**  [pg-rex01] $ exit  [pg-rex01] # rm -f /var/lib/pgsql/tmp/PGSQL.lock |

1. msPostgresqlのリソース再組み込み

(1)のコマンドによりmsPostgresqlのリソースが自動的にフェイルオーバできない状態になっているため、状態を元に戻すコマンドを実行します。

|  |
| --- |
| [pg-rex01] # crm\_resource -U -r msPostgresql |

1. 系切り替え確認

pg-rex01でcrm\_monコマンドを実行し、以下を確認します。

* msPostgresqlリソースが、pg-rex01上でSlaveとして動作していること。
* vip-slaveリソースが、pg-rex01上で起動していること。

|  |
| --- |
| [pg-rex01] # crm\_mon -fA  Last updated: *日時表示*  Last change: *日時表示*  Stack: corosync  Current DC: pg-rex01 (*UUID表示*) - partition with quorum  ：（略）  Online: [ pg-rex01 pg-rex02 ]  vip-slave (ocf::heartbeat:IPaddr2): **Started pg-rex01**  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): Started pg-rex02  vip-rep (ocf::heartbeat:IPaddr2): Started pg-rex02  ：（略）  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pg-rex02 ]**  **Slaves: [ pg-rex01 ]**  ：（略）  Node Attributes:  \* Node pg-rex01:  + default\_ping\_set : 100  + 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）  Migration summary:  \* **Node pg-rex01:**  \* Node pg-rex02: |

1. 運用補助ツール概要

本付録では、PG-REX運用補助ツールについての概要を説明します。

* 1. ツール概要

PG-REX運用補助ツールとは、PG-REXの運用手順の簡易化を目的としたコマンド群です。

各コマンドの概要を以下に示します。

表 付録D-1　運用補助ツールのコマンド一覧

|  |  |  |
| --- | --- | --- |
| No. | コマンド名 | 概要 |
| 1 | pg-rex\_master\_start | PG-REXをMasterとして起動する |
| 2 | pg-rex\_slave\_start | PG-REXをSlaveとして起動する |
| 3 | pg-rex\_stop | PG-REXのMasterまたはSlaveを停止する |
| 4 | pg-rex\_archivefile\_delete | データベースの復旧に必要のないアーカイブログを削除する |
| 5 | pg-rex\_dbcluster\_check | PG-REXのSlave起動時にベースバックアップを取得してDBクラスタを再構築する必要があるかを確認する |
| 6 | pg-rex\_switchover | PG-REXの系切り替えを実施する |

* 1. 環境設定ファイルについて

運用補助ツールで利用する環境設定ファイル（/etc/pg-rex\_tools.conf）の設定値について説明します。環境設定ファイルで設定する内容は、以下のとおりです。

表 付録D-2　pg-rex\_tools.confのパラメータ

|  |  |  |
| --- | --- | --- |
| パラメータ名 | 初期値[[33]](#footnote-33) | 説明 |
| D-LAN\_IPAddress | ― | **両系のD-LAN IPアドレス**を指定する。IPアドレスはカンマで区切って指定する。 |
| Archive\_dir | ― | **アーカイブディレクトリの絶対パス**を指定する。 |
| STONITH | enable | STONITH導入環境であれば**enable**を指定する。そうでなければ**disable**（非推奨）を指定する。 |
| VIP\_SLAVE | enable | vip\_slave[[34]](#footnote-34)を使用する環境であれば**enable**を指定する。そうでなければ、**disable**を指定する。 |
| PGPATH | /usr/pgsql-9.4/bin | **PostgreSQLコマンドへの絶対パス**を指定する。postgresユーザログイン時に設定される環境変数のPATHにPostgreSQLコマンドへのパスの指定が無ければ、この設定が有効になる。なお、本パラメータには運用補助ツールの各パッケージのデフォルト値が予め設定されている。 |
| PEER\_NODE\_SSH\_PASS\_MODE | manual | **manual、passfile、nopass**のいずれかを指定する。運用補助ツールは実行時に相手ノードにsshで接続する場合があり、このパラメータでssh接続のパスワード取得方法を変更することが出来る。設定値はセキュリティの観点から**manual**を推奨する。   * manual ： パスワードを手動で入力する。 * passfile ： PEER\_NODE\_SSH\_ PASS\_FILEに指定したファイル内のパスワードを参照する。この設定の場合、PEER\_NODE\_ SSH\_PASS\_FILEの指定が必須となる。 * nopass ： パスワードを空文字にする。 |
| PEER\_NODE\_SSH\_PASS\_FILE | ― | **相手ノードへのssh接続に必要なパスワードが記述されたファイルの絶対パス**を指定する。  **PEER\_NODE\_SSH\_PASS\_MODEにpassfileを指定している場合、設定が必須となる**。passfile以外を指定している場合、この設定は参照されない。 |
| BACKUP\_NODE\_SSH\_PASS\_MODE | manual | **manual、passfile、nopass**のいずれかを設定する。運用補助ツールは実行時にDBクラスタのバックアップ格納先ノードにsshで接続する場合があり、このパラメータでssh接続のパスワード取得方法を変更することが出来る。設定値はセキュリティの観点から**manual**を推奨する。   * manual ： パスワードを手動で入力する。 * passfile ： BACKUP\_NODE\_SSH\_ PASS\_FILEに指定したファイル内のパスワードを参照する。この設定の場合、BACKUP\_NODE\_SSH\_ PASS\_FILEの指定が必須となる。 * nopass ： パスワードを空文字にする。 |
| BACKUP\_NODE\_SSH\_PASS\_FILE | ― | **DBクラスタのバックアップ格納先ノードへのssh接続に必要なパスワードが記述されたファイルの絶対パス**を指定する。**BACKUP\_NODE\_SSH\_PASS\_ MODEにpassfileを指定している場合、設定が必須となる**。passfile以外を指定している場合、この設定は参照されない。 |
| PG-REX\_Master\_ResourceID | msPostgresql | **PG-REXのMasterのリソースID**を指定する。 |
| PG-REX\_Primitive\_ResourceID | pgsql | **PG-REXのPrimitiveのリソースID**を指定する。 |
| VIP-MASTER\_ResourceID | vip-master | **vip-masterのPrimitiveのリソースID**を指定する。**省略した場合はvip-masterの起動確認が行われない。** |
| VIP-REP\_ResourceID | vip-rep | **vip-repのPrimitiveのリソースID**を指定する。**省略した場合はvip-repの起動確認が行われない。** |
| VIP-SLAVE\_ResourceID | vip-slave | **vip-slaveのPrimitiveのリソースID**を指定する。**省略した場合はvip-slaveの起動確認が行われない。** |
| PING\_ResourceID | clnPing | **pingのリソースID**を指定する。複数設定する場合はカンマ区切りで指定する。**省略した場合はpingの起動確認が行われない。** |
| DISKD\_ResourceID | clnDiskd1 | **diskdのリソースID**を指定する。複数設定する場合はカンマ区切りで指定する。**省略した場合はdiskdの起動確認が行われない。** |
| STONITH\_ResourceID | grpStonith1, grpStonith2 | **STONITHのGroupのリソースIDをカンマ区切りで**指定する。STONITHにdisableが指定されている場合、この設定は無視され、STONITHのGroupの起動確認が行われない。また、この**設定を省略した場合も起動確認が行われない**。 |
| ARCHIVE\_DELETE\_DBCLUSTER\_DIR | ― | 運用補助ツールをrootユーザで実行した時に自動的に追加されるパラメータであり、**指定する必要なし**。PG-REXが管理しているDBクラスタディレクトリの絶対パスが設定される。本パラメータはpg-rex\_archivefile\_deleteコマンド実行時に、削除するアーカイブログの基準を決定するために参照される。**PG-REX環境の再構成等で、DBクラスタディレクトリの場所を変更した場合、この設定は削除する必要がある**。 |

設定例を以下に示します。

|  |
| --- |
| $ vi /etc/pg-rex\_tools.conf  **D-LAN\_IPAddress = *192.168.2.1* , *192.168.2.2***  **Archive\_dir = /dbfp/pgarch/arc1**  **STONITH = enable**  **VIP\_SLAVE = enable**  **PGPATH = /usr/pgsql-9.4/bin**  **PEER\_NODE\_SSH\_PASS\_MODE = manual**  **PEER\_NODE\_SSH\_PASS\_FILE = /root/.ssh/peer\_node\_passfile**  **BACKUP\_NODE\_SSH\_PASS\_MODE = manual**  **BACKUP\_NODE\_SSH\_PASS\_FILE = /root/.ssh/backup\_node\_passfile**  **PG-REX\_Master\_ResourceID = msPostgresql**  **PG-REX\_Primitive\_ResourceID = pgsql**  **VIP-MASTER\_ResourceID = vip-master**  **VIP-REP\_ResourceID = vip-rep**  **VIP-SLAVE\_ResourceID = vip-slave**  **PING\_ResourceID = clnPing**  **DISKD\_ResourceID = clnDiskd1**  **STONITH\_ResourceID = grpStonith1 , grpStonith2**  **ARCHIVE\_DELETE\_DBCLUSTER\_DIR = /dbfp/pgdata** # **運用補助ツールをrootユーザで一度でも実行している場合に記述されている** |

* 1. 各コマンド機能概要

各コマンドの概要、Usage、実行例を示します。

* pg-rex\_master\_start
  + 概要

本コマンドを実行したノードで、PG-REXをMasterとして起動します。

* + Usage

pg-rex\_master\_start [-h][-v][CrmFilePath]

-h, --help Usageを表示し終了します

-v, --version バージョン情報を表示し終了します

CrmFilePath 初回起動時のみcrmファイルのファイルパスを指定します

* + 実行例（引数なし）

|  |
| --- |
| # 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 が起動していないことを確認  …[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(pg-rex01)が Master として起動しました |

* + 実行例（引数あり）

|  |
| --- |
| # pg-rex\_master\_start PG-REX9.4\_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 が起動していないことを確認  …[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(pg-rex01)が Master として起動しました |

* pg-rex\_slave\_start
  + 概要

本コマンドを実行したノードで、PG-REXをSlaveとして起動します。

* + Usage

pg-rex\_slave\_start [-h][-v][-n][-f]

-h, --help Usageを表示し終了します

-v, --version バージョン情報を表示し終了します

-n, --nobasebackup Masterからベースバックアップを取得せずに

Slaveを起動します

-f, --force スクリプトの問い合わせ無しで実行します

* + 実行例

|  |
| --- |
| # 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 が起動していないことを確認  …[OK]  2. 稼働中の Master が存在していることを確認  …[OK]  3. IC-LAN が接続されていることを確認  …[OK]  4. 起動禁止フラグの存在を確認  …[OK]  5. Master からベースバックアップ取得  ベースバックアップの取得を行います  よろしいでしょうか？ (y/n) y  20227/20227 kB (100%), 1/1 tablespace  …[OK]  6. Master のアーカイブディレクトリと同期  receiving incremental file list  ./  0000002A000000000000004C  0000002B.history  0000002B000000000000004C  0000002C.history  0000002C000000000000004C  0000002C000000000000004D  0000002C000000000000004D.00000020.backup  sent 147 bytes received 67122879 bytes 44748684.00 bytes/sec  total size is 83893968 speedup is 1.25  …[OK]  7. /var/lib/pacemaker/cib 配下のファイルを削除  …[OK]  8. Slave の起動  …[OK]  9. Slave の起動確認  …[OK]  ノード(pg-rex02)が Slave として起動しました |

* pg-rex\_stop
  + 概要

本コマンドを実行したノードで、PG-REXのMasterまたはSlaveを停止します。

* + Usage

pg-rex\_stop [-f][-h][-v]

-f, --fast 停止前にCHECKPOINTとsyncコマンドを実行しません

-h, --help Usageを表示し終了します

-v, --version バージョン情報を表示し終了します

|  |
| --- |
| # 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 (pg-rex01)を停止しました |

* + 実行例
* pg-rex\_archivefile\_delete
  + 概要

本コマンドを実行したノードで、PG-REXのMasterとSlaveのDBクラスタ、およびコマンド実行時に指定したベースバックアップのリカバリに不要なアーカイブログを削除します。指定したベースバックアップの取得時点よりも過去に取得したベースバックアップは使用できなくなることに注意してください。

コマンド実行時にベースバックアップの指定を省略した場合は、対話形式での指定となります。対話形式でも省略した場合は、MasterとSlaveのみを対象にして不要なアーカイブログを削除します。

* + Usage

pg-rex\_archivefile\_delete {-m|-r}[-f][-h][-v][-D DBclusterFilepath]

[[Hostname:]BasebackupPath]

-m, --move mvモードで実行します

-r, --remove rmモードで実行します

(mvモードまたはrmモードはどちらか片方を必ず指定してください)

-f, --force スクリプトの問い合わせ無しで実行します

-h, --help Usage を表示し終了します

-v, --version バージョン情報を表示し終了します

-D, --dbcluster=DBclusterFilepath

DBクラスタの場所を絶対パスで指定します

Hostname ベースバックアップが存在するリモートサーバを指定します

BasebackupPath ベースバックアップの場所を絶対パスで指定します

注) 指定したベースバックアップよりも過去に取得したベースバックアップは、

アーカイブログが削除されてしまうため、使用することができなくなります

* + 実行例（mvモードでベースバックアップの指定をしない場合）

|  |
| --- |
| # pg-rex\_archivefile\_delete -m  \*\*\*\* 1. 実行準備 \*\*\*\*  mv モードで実行します  ベースバックアップが存在するリモートサーバを入力してください  (入力しなければ "localhost" を設定します)  >  ベースバックアップの場所の絶対パスを入力してください  (入力しなければバックアップ指定無しとして実行されアーカイブが削除されるため、  以前に取得したベースバックアップが使用できなくなります)  >  環境設定ファイル (pg-rex\_tools.conf) を読み込みます  root@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  cib.xml ファイルを読み込みます  ベースバックアップの場所を指定せずに実行すると、  自身のノード "pg-rex01" と相手のノード "pg-rex02" の  現時点の PGDATA "/pgdata/data" を基準にしてアーカイブログを削除することになります  アーカイブログを削除しますか (y/N) : y  \*\*\*\* 2. WAL ファイル名の取得 \*\*\*\*  自身のノード "pg-rex01" の現時点の PGDATA "/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  "00000002000000000000000C"  相手のノード "pg-rex02" の現時点の PGDATA "/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  "000000020000000000000003"  \*\*\*\* 3. 削除基準の算出 \*\*\*\*  削除基準を "000000020000000000000003" としました  \*\*\*\* 4. アーカイブログの移動 \*\*\*\*  削除対象のリストに "000000010000000000000002" を追加します  削除対象のリストに "000000020000000000000002" を追加します  削除対象のリストに "000000010000000000000001" を追加します  移動先ディレクトリ "/pgdata/pgarch/arc1/20130826\_163510" を作成しました  -- mv -- 000000010000000000000002  -- mv -- 000000020000000000000002  -- mv -- 000000010000000000000001  アーカイブログの移動に成功しました  mv モード実行のため、移動したファイルは"/pgdata/pgarch/arc1/20130826\_163510" に格納されています |

* + 実行例（rmモードで、対象とするベースバックアップにリモートサーバpg-rex03の/pgdata/backup\_dataを指定した場合）

|  |
| --- |
| # pg-rex\_archivefile\_delete -r pg-rex03:/pgdata/backup\_data  \*\*\*\* 1. 実行準備 \*\*\*\*  rm モードで実行します  環境設定ファイル (pg-rex\_tools.conf) を読み込みます  root@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  cib.xml ファイルを読み込みます  \*\*\*\* 2. WAL ファイル名の取得 \*\*\*\*  指定されたバックアップからリカバリを行うために必要な最初の WAL ファイル名を取得します  root@pg-rex03's password:  パスワードが入力されました  "000000020000000000000004"  自身のノード "pg-rex01" の現時点の PGDATA "/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  "000000020000000000000003"  相手のノード "pg-rex02" の現時点の PGDATA "/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  "00000002000000000000000C"  \*\*\*\* 3. 削除基準の算出 \*\*\*\*  削除基準を "000000020000000000000003" 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 "000000010000000000000001" を追加します  削除対象のリストに "000000010000000000000002" を追加します  削除対象のリストに "000000020000000000000002" を追加します  -- rm -- 000000010000000000000001  -- rm -- 000000010000000000000002  -- rm -- 000000020000000000000002  アーカイブログの削除に成功しました |

* pg-rex\_dbcluster\_check
  + 概要

Slave起動時にベースバックアップからDBクラスタを再構築する必要があるかを確認します。本コマンドは、実行したノードのDBクラスタについて以下の点を確認します。

* + - 初回起動ではない
    - DBクラスタが存在している
    - 起動禁止フラグが存在しない
    - 稼働中のMasterとデータを失わずにレプリケーション接続が確立できる状態であること

上記の点を一つでも満たさない場合、本コマンドは、Slaveをベースバックアップから再構築する必要があることを通知します。

ただし、上記の点を全て満たしていても、反映すべきWALの量が非常に多い場合はSlaveの起動処理がタイムアウトする可能性があります。その場合、Slave起動時にMasterからベースバックアップを取得する等の対策が必要となります。

本コマンドはSlaveとして起動したいノードで実行してください。

* + Usage

pg-rex\_dbcluster\_check [-h][-v]

-h, --help Usageを表示し終了します

-v, --version バージョン情報を表示し終了します

* + 実行例

|  |
| --- |
| # pg-rex\_dbcluster\_check  \*\*\*\* 実行準備 \*\*\*\*  root@192.168.2.1's password:  パスワードが入力されました  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. 自身のノードで Pacemaker および Corosync が停止していることを確認  …[OK]  3. 相手のノードが Master として既に起動していることを確認  …[OK]  \*\*\*\* Slave をベースバックアップから再構築する必要があるかを確認 \*\*\*\*  1. 初回起動でないことを確認  …[OK]  2. DB クラスタの存在およびバージョンの確認  …[OK]  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  最終チェックポイント日時 : Fri Jun 19 16:42:22 2014  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  3. 起動禁止フラグがあることを確認  …[OK]  4. 両系の DB クラスタの状況を確認  …[OK]  ベースバックアップから再構築せずに Slave を起動することができます  ただし、DBクラスタが古い場合はリカバリに時間がかかり、  Pacemaker による Slave の起動処理が中止される可能性があります |

* pg-rex\_switchover
  + 概要

Slaveの再組み込み時にベースバックアップを取得せずにPG-REXの系切り替えを実行します。ベースバックアップを取得しないことで、系切り替え時間の短縮を実現します。

本コマンドは、PG-REXのMasterとSlaveのどちらのノードでも実行することができます。

* + Usage

pg-rex\_switchover [-h][-v]

-h, --help Usageを表示し終了します

-v, --version バージョン情報を表示し終了します

* + 実行例

|  |
| --- |
| # pg-rex\_switchover  root@192.168.2.2's password:  パスワードが入力されました  \*\*\*\* 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 : pg-rex01 -> pg-rex02  Slave : pg-rex02 -> pg-rex01  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  3. vacuumdb コマンドの実行  …[OK]  4. CHECKPOINT の実行  …[OK]  \*\*\*\* 系切り替えを実行 \*\*\*\*  5. Pacemaker の監視を停止  …[OK]  6. Master (pg-rex01) の PostgreSQL を停止  …[OK]  7. Pacemaker の監視を再開し系切り替えを実行  …[OK]  8. pg-rex02 が新 Master になったことを確認  \*\*\*\* pg-rex02 が Master として起動しました \*\*\*\*  9. pg-rex01 の Pacemaker を停止  …[OK]  10. pg-rex01 で Slave を起動  receiving incremental file list  ./  00000011000000000000000C  00000012000000000000000E  00000013.history  sent 71 bytes received 33561066 bytes 7458030.44 bytes/sec  total size is 503327160 speedup is 15.00  \*\*\*\* pg-rex01 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 : pg-rex02  Slave : pg-rex01 |

* 1. FAQ

運用補助ツール使用時における、よくある質問について以下に示します。

表 付録D-3　FAQ一覧

|  |  |  |
| --- | --- | --- |
| 項番 | 事象 | 原因・解決方法 |
| 1 | 運用補助ツールコマンドのタイムアウトの時間を変更したい。 | pg-rex\_master\_start、pg-rex\_slave\_start、pg-rex\_stop、pg-rex\_switchoverコマンドの起動/停止確認時間のタイムアウト値は300秒に設定してあります。これを変更したい場合は、/usr/local/bin配下にある上記コマンドの”my $timeout = 300”と記述されている箇所の数値(秒単位)を変更して下さい。 |
| 2 | Slave起動時にベースバックアップを取得しなくてよいパターンとは具体的にどういう場合か。 | 例えば、フェイルオーバ後に旧MasterをSlaveとして起動する場合や、Slaveを一旦停止して再び起動する場合はベースバックアップを取得する必要はありません。ただし、どちらの場合でもSlave起動時のPostgreSQLのリカバリに必要なWALファイルを消去していないことが前提となります。なお、Slave起動時のベースバックアップ取得要否はpg-rex\_dbcluster\_checkを使用することで確認することができます。 |
| 3 | PG-REXでインストールするファイルのディレクトリ構成を両系で異なる構成にしたい。 | PG-REXではディレクトリ構成が両系で同一であることを前提としています。ディレクトリ構成が両系で同一でない場合、運用補助ツールは正常に動作しません。 |
| 4 | 運用補助ツールが表示するメッセージに日本語と英語が混在する。 | 運用補助ツールではPostgreSQLのコマンドをpostgresユーザに切り替えて実行しています。そのため、PostgreSQLのコマンドが出力するメッセージはpostgresユーザの言語設定に沿って表示されます。 |

用語集

本付録では、資料に使用されている用語について以下に示します。

表 1　用語集

|  |  |
| --- | --- |
| 用語・略語 | 説明 |
| PG-REX9.4 | PacemakerとPostgreSQL9.4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-REX9.4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RAの構成のうちの一つ。PG-REXのRAではこの構成を用いる。 |
| pg-rex01  (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動させているサーバ。 |
| pg-rex02  (予備機) | クラスタシステムにおいて、pg-rex01（現用機）の故障発生時に処理やサービスを稼動させているサーバ。 |
| スプリットブレイン | pg-rex01とpg-rex02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状態。 |
| IC-LAN | インターコネクトLANのこと。pg-rex01とpg-rex02間のネットワーク。pg-rex01とpg-rex02上の各Pacemakerが互いに死活監視するために使用される。 |
| D-LAN | データ転送用LANのこと。pg-rex01とpg-rex02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 | サービス提供用LANのこと。pg-rex01またはpg-rex02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。pg-rex01とpg-rex02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が起動していない状態。 |
| start | RAの動作を表す: StoppedのリソースをSlaveにすること。 |
| stop | RAの動作を表す: SlaveのリソースをStoppedにすること。 |
| promote | RAの動作を表す: SlaveのリソースをMasterにすること。 |
| demote | RAの動作を表す: MasterのリソースをSlaveにすること。 |
| monitor | RAの動作を表す: 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. 本マニュアル発行の時点では、日本語訳が未公開ですが、上記のURLで公開されるものと想定しました。『PostgreSQLドキュメント』を参照する他の箇所も同様です。ご了承ください。 [↑](#footnote-ref-1)
2. 参照先：<http://www.postgresql.jp/document/9.4/html/functions-admin.html> [↑](#footnote-ref-2)
3. 参照先：<http://www.postgresql.jp/document/9.4/html/functions-admin.html> [↑](#footnote-ref-3)
4. 参照先：<http://www.postgresql.jp/document/9.4/html/warm-standby.html#STANDBY-PLANNING> [↑](#footnote-ref-4)
5. 参照先：<http://www.postgresql.jp/document/9.4/html/hot-standby.html#HOT-STANDBY-CAVEATS> [↑](#footnote-ref-5)
6. 参照先：<http://www.postgresql.jp/document/9.4/html/hot-standby.html#HOT-STANDBY-CONFLICT> [↑](#footnote-ref-6)
7. ipmi以外をSTONITHに使用する場合は、ハードウェアのベンダに設定方法をお問い合わせください。 [↑](#footnote-ref-7)
8. 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.sourceforge.jp/>)を参照してください。 [↑](#footnote-ref-8)
9. 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-9)
10. 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-10)
11. PG-REXの設定値として設定可能な範囲を表します。 [↑](#footnote-ref-11)
12. 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『3.6.6 リソース(PostgreSQL)の設定』を参照してください。 [↑](#footnote-ref-12)
13. tcp\_keepalives\_idle、tcp\_keepalives\_interval、tcp\_keepalives\_countの設定を指します。 [↑](#footnote-ref-13)
14. checkpoint\_segmentsを200～300に設定した場合、wal\_sender\_timeoutを20sに設定しているとSlaveが切り離される可能性があるため、wal\_sender\_timeoutを40s～60s程度に設定してください。 [↑](#footnote-ref-14)
15. アーカイブコマンドの設定は『3.4.6 postgresql.confの編集』を参照してください。 [↑](#footnote-ref-15)
16. /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-16)
17. PostgreSQLがpromoteされる直前のxlog locationの値を表示する属性です。 [↑](#footnote-ref-17)
18. PostgreSQLのlast xlog replay location または last xlog receive location を比較して新しい方の値が表示されます。 [↑](#footnote-ref-18)
19. pg-rex01のeth1からの通信状態の場合の例を示します。 [↑](#footnote-ref-19)
20. IC-LANを冗長化している環境において、いずれかのインターフェイスのみ通信が異常となった場合、異常となったインターフェイスの属性値は" FAULTY"が表示されます。 [↑](#footnote-ref-20)
21. 状態遷移時に一時的に表示される属性であり一時的に表示されても問題ありません。 [↑](#footnote-ref-21)
22. 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-22)
23. 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-23)
24. 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pg-rex02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-24)
25. crm\_monコマンドの実行結果は停止状態 (Stoppedが表示、または表示されていない状態) ですが、リソースは稼働している状態です。 [↑](#footnote-ref-25)
26. crm\_monコマンドの実行結果は停止状態 (Stoppedが表示されている状態) ですが、cloneリソースは稼働している状態です。 [↑](#footnote-ref-26)
27. 参照先：<http://www.postgresql.jp/document/9.4/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-27)
28. バックアップ履歴ファイルは、拡張子が.backupのファイルを指します。

    <http://www.postgresql.jp/document/9.4/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-28)
29. タイムライン履歴ファイルは、拡張子が.historyのファイルを指します。

    <http://www.postgresql.jp/document/9.4/html/continuous-archiving.html#BACKUP-TIMELINES> [↑](#footnote-ref-29)
30. 「XXX」には、「No route to host」や「Connection refused」などの文字列が入ります。 [↑](#footnote-ref-30)
31. 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/9.4/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-31)
32. 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/9.4/html/continuous-archiving.html#BACKUP-TIMELINES> [↑](#footnote-ref-32)
33. 運用補助ツールインストール直後のpg-rex\_tools.confに設定されている値を表します。 [↑](#footnote-ref-33)
34. 本書ではSlave側の仮想IPを意味します。 [↑](#footnote-ref-34)