**PG-REX10利用マニュアル**

**第1.0.0版**

**2018年3月22日**

**NTT OSSセンタ**

改版履歴

|  |  |  |
| --- | --- | --- |
| 日付 | 版数 | 変更内容 |
| 2018/3/22 | 1.0.0 | 初版制定 |

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

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

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

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

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

Linuxは、Linus Torvaldsの登録商標です。

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

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

―― 目次 ――

[1章. はじめに 1](#_Toc509583313)

[1.1. 本書の目的 1](#_Toc509583314)

[1.2. 対象読者 1](#_Toc509583315)

[1.3. 表記規則 2](#_Toc509583316)

[1.4. 対象とするソフトウェアとバージョン 2](#_Toc509583317)

[1.5. PG-REX9.6からの主な変更点 2](#_Toc509583318)

[1.6. 前提条件 2](#_Toc509583319)

[2章. 概要 4](#_Toc509583320)

[2.1. PG-REXの概要 4](#_Toc509583321)

[2.2. システム構成 5](#_Toc509583322)

[2.3. 制約 6](#_Toc509583323)

[2.3.1. PG-REXの制約 6](#_Toc509583324)

[2.3.2. PG-REX運用補助ツールの制約 9](#_Toc509583325)

[2.4. 推奨設定 10](#_Toc509583326)

[3章. 環境構築 11](#_Toc509583327)

[3.1. ネットワーク 11](#_Toc509583328)

[3.2. ディレクトリ構成 14](#_Toc509583329)

[3.3. Pacemaker 15](#_Toc509583330)

[3.3.1. 事前作業 15](#_Toc509583331)

[3.3.2. インストール 17](#_Toc509583332)

[3.3.3. Pacemaker関連プロセスの自動起動の無効化 19](#_Toc509583333)

[3.3.4. クラスタの基本設定（corosync.conf） 20](#_Toc509583334)

[3.3.5. Corosync設定ファイルの確認 21](#_Toc509583335)

[3.3.6. クラスタ認証設定ファイルの生成（authkey） 21](#_Toc509583336)

[3.3.7. Pacemaker設定ファイルの追加設定（pacemaker） 21](#_Toc509583337)

[3.3.8. Corosync用Unitファイル(corosync.service)の修正 22](#_Toc509583338)

[3.3.9. Pacemaker用Unitファイル(pacemaker.service)の修正 23](#_Toc509583339)

[3.3.10. ログの出力設定（rsyslog.conf） 24](#_Toc509583340)

[3.3.11. ログメッセージ制御機能設定（pm\_logconv.conf） 25](#_Toc509583341)

[3.3.12. ログローテーションの追加設定（syslog） 25](#_Toc509583342)

[3.3.13. pm\_logconvのプロセス起動 27](#_Toc509583343)

[3.3.14. ifcheckdプロセスの起動 27](#_Toc509583344)

[3.4. PostgreSQL 29](#_Toc509583345)

[3.4.1. PostgreSQLのインストール 29](#_Toc509583346)

[3.4.2. 環境変数の設定 32](#_Toc509583347)

[3.4.3. DBクラスタ用ディレクトリの作成 32](#_Toc509583348)

[3.4.4. DBクラスタの初期化 33](#_Toc509583349)

[3.4.5. postgresql.confの編集 33](#_Toc509583350)

[3.4.6. レプリケーションユーザの作成 36](#_Toc509583351)

[3.4.7. pg\_hba.confの編集 38](#_Toc509583352)

[3.4.8. .pgpassの作成 38](#_Toc509583353)

[3.5. PG-REX運用補助ツール 40](#_Toc509583354)

[3.5.1. インストール 40](#_Toc509583355)

[3.5.2. pg-rex\_tools.confの編集 41](#_Toc509583356)

[3.5.3. ネットワーク接続登録 42](#_Toc509583357)

[3.6. リソースの設定 43](#_Toc509583358)

[3.6.1. pm\_crmgenの概要 43](#_Toc509583359)

[3.6.2. 変更不要の設定 43](#_Toc509583360)

[3.6.3. リソース構成の設定 43](#_Toc509583361)

[3.6.4. リソース（Master側仮想IP）の設定 44](#_Toc509583362)

[3.6.5. リソース（PostgreSQL）の設定 44](#_Toc509583363)

[3.6.6. リソース（ネットワーク監視）の設定 46](#_Toc509583364)

[3.6.7. リソース（ディスク監視）の設定 46](#_Toc509583365)

[3.6.8. リソース配置制約の設定 46](#_Toc509583366)

[3.6.9. リソース同居制約の設定 46](#_Toc509583367)

[3.6.10. リソース起動順序制約の設定 46](#_Toc509583368)

[3.6.11. リソース（Slave側仮想IP）の設定 46](#_Toc509583369)

[3.6.12. crmファイルの作成 47](#_Toc509583370)

[3.6.13. crmファイルの反映 47](#_Toc509583371)

[3.7. アンインストール 48](#_Toc509583372)

[4章. 起動と停止 51](#_Toc509583373)

[4.1. 両系の起動 51](#_Toc509583374)

[4.2. Masterの起動 51](#_Toc509583375)

[4.3. Slaveの起動 54](#_Toc509583376)

[4.4. 両系の停止 57](#_Toc509583377)

[4.5. Slaveの停止 57](#_Toc509583378)

[4.6. Masterの停止 57](#_Toc509583379)

[4.7. PostgreSQL停止中の系の停止 58](#_Toc509583380)

[5章. メンテナンス時の対応 59](#_Toc509583381)

[5.1. PostgreSQLのバックアップ 59](#_Toc509583382)

[5.1.1. 前提条件 59](#_Toc509583383)

[5.1.2. PostgreSQLのバックアップ 59](#_Toc509583384)

[5.2. アーカイブログの削除 60](#_Toc509583385)

[5.2.1. PostgreSQLアーカイブログの削除 60](#_Toc509583386)

[5.3. 計画的な系切り替え 62](#_Toc509583387)

[5.3.1. 前提条件 62](#_Toc509583388)

[5.3.2. 系切り替え 62](#_Toc509583389)

[5.4. 計画的なメンテナンス 65](#_Toc509583390)

[5.4.1. 前提条件 65](#_Toc509583391)

[5.4.2. Masterのメンテナンス 65](#_Toc509583392)

[5.4.3. Slaveのメンテナンス 66](#_Toc509583393)

[6章. 故障対応 67](#_Toc509583394)

[6.1. 前提条件 67](#_Toc509583395)

[6.2. 監視コマンド表示確認方法 67](#_Toc509583396)

[6.2.1. 表示部説明 69](#_Toc509583397)

[6.2.2. 正常状態確認方法 72](#_Toc509583398)

[6.3. 故障箇所特定手順 74](#_Toc509583399)

[6.3.1. ノード情報・リソース情報・故障回数表示部の確認 75](#_Toc509583400)

[6.3.2. 属性情報表示部の確認 78](#_Toc509583401)

[6.3.3. 制御エラー情報表示部の確認 80](#_Toc509583402)

[6.3.4. /var/log/pm\_logconv.outの確認 82](#_Toc509583403)

[6.4. 故障一覧 83](#_Toc509583404)

[6.5. ルータ故障 84](#_Toc509583405)

[6.5.1. 故障時のクラスタ状態 84](#_Toc509583406)

[6.5.2. 復旧 86](#_Toc509583407)

[6.6. S-LAN故障 90](#_Toc509583408)

[6.6.1. 故障時のクラスタ状態 90](#_Toc509583409)

[6.6.2. 復旧 92](#_Toc509583410)

[6.7. D-LAN故障 95](#_Toc509583411)

[6.7.1. 故障時のクラスタ状態 95](#_Toc509583412)

[6.7.2. 復旧 97](#_Toc509583413)

[6.8. リソース故障(monitor) 100](#_Toc509583414)

[6.8.1. 故障時のクラスタ状態 100](#_Toc509583415)

[6.8.2. 復旧 102](#_Toc509583416)

[6.9. リソース故障(demote/stop) 105](#_Toc509583417)

[6.9.1. 故障時のクラスタ状態 105](#_Toc509583418)

[6.9.2. 復旧 106](#_Toc509583419)

[6.10. リソース故障(vip-master) 109](#_Toc509583420)

[6.10.1. 故障時のクラスタ状態 109](#_Toc509583421)

[6.10.2. 復旧 111](#_Toc509583422)

[6.11. リソース故障(vip-rep) 114](#_Toc509583423)

[6.11.1. 故障時のクラスタ状態 114](#_Toc509583424)

[6.11.2. 復旧 116](#_Toc509583425)

[6.12. リソース故障(vip-slave) 119](#_Toc509583426)

[6.12.1. 故障時のクラスタ状態 119](#_Toc509583427)

[6.12.2. 復旧 121](#_Toc509583428)

[6.13. 内蔵ディスク故障 124](#_Toc509583429)

[6.13.1. 故障時のクラスタ状態 124](#_Toc509583430)

[6.13.2. 復旧 126](#_Toc509583431)

[6.14. ノード故障 129](#_Toc509583432)

[6.14.1. 故障時のクラスタ状態 129](#_Toc509583433)

[6.14.2. 復旧 131](#_Toc509583434)

[6.15. IC-LAN故障 134](#_Toc509583435)

[6.15.1. 故障時のクラスタ状態 134](#_Toc509583436)

[6.15.2. 復旧 136](#_Toc509583437)

[付録A 起動と停止 141](#_Toc509583438)

[A.1. 両系の起動 141](#_Toc509583439)

[A.2. Masterの起動 141](#_Toc509583440)

[A.3. Slaveの起動 148](#_Toc509583441)

[A.4. 両系の停止 153](#_Toc509583442)

[A.5. Slaveの停止 153](#_Toc509583443)

[A.6. Masterの停止 153](#_Toc509583444)

[付録B アーカイブログの削除 155](#_Toc509583445)

[B.1. 前提条件 155](#_Toc509583446)

[B.2. PostgreSQLアーカイブログの削除 155](#_Toc509583447)

[付録C 計画的な系切り替え 158](#_Toc509583448)

[C.1. 前提条件 158](#_Toc509583449)

[C.2. 系切り替え 158](#_Toc509583450)

[用語集 162](#_Toc509583451)

# はじめに

## 本書の目的

本書は、PostgreSQL-REX10（以降、PG-REXと呼ぶ）の技術情報を説明します。本書の目的は、読者が、PG-REXの動作環境を構築し、PG-REXを操作できるようになることです。

## 対象読者

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

Pacemaker関連サイト

<http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/index.html> (本家マニュアル)

<http://clusterlabs.org/> (Linux-HAのメインサイト(英語))

<http://linux-ha.osdn.jp/> (Linux-HA Japanプロジェクトのサイト)

<http://ja.osdn.net/projects/linux-ha/> (Linux-HA Japanの開発者向のサイト)

⇒本書では、上記をまとめて『Pacemaker関連サイト』と呼びます。

PostgreSQL10付属ドキュメント

<http://www.postgresql.jp/document/10/>

⇒本書では、『PostgreSQLドキュメント』と呼びます。

## 表記規則

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

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） | 『Pacemaker関連サイト』 |
| *斜体* | 環境に応じて変更する項目 | *eth1* |
| # | rootユーザの操作記号 | # crm\_mon |
| $ | 一般ユーザの操作記号 | $ psql |
| **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** |

## 対象とするソフトウェアとバージョン

本書が対象とする各ソフトウェアのバージョンは以下の通りです。

* RedHat Enterprise Linux 7
* PostgreSQL 10
* Pacemaker 1.1.17

## PG-REX9.6からの主な変更点

PG-REX9.6からの主な変更点は以下の通りです。

* archive\_mode=alwaysに対応
* Slave起動のために取得するpg\_basebackupは-X noneを指定するように変更
* 運用補助ツールのマニュアルを分離し、運用補助ツールのRPMに同梱

## 前提条件

本書に記載されているコマンド実行例は、LANG=ja\_JP.UTF-8を前提にしています。LANGの設定によっては、出力メッセージなど実行例の内容が本書と異なる可能性があるため注意してください。例えば、PostgreSQL関連のツールは、LANG=ja\_JP.UTF-8では日本語でメッセージ等を出力します[[1]](#footnote-1)が、LANG=Cの場合は英語で出力します。

また、コンソールの表示はOSのバージョンによって異なる場合があります。

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

本書の実行例は、単一ディスク構成(OS, DBを同一のディスクに格納)によるものです。

# 概要

## PG-REXの概要

PG-REXは、PostgreSQL10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、PostgreSQL10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

本書の内容は以下の環境で確認しました。実行例も以下の環境でのものです。

表 2‑1　ハードウェア構成例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | 用途 | 数量 | 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 | 製品名 | バージョン |
| ① | 共通 | OS | RedHat Enterprise Linux | 7.4 |
| ② | DBサーバ | DBMS | PostgreSQL | 10.2 |
| クラスタリング | Pacemaker | 1.1.17-1.1 |

## 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『3.4.5 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 | 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。 |
| 2 | PG-REXで制御可能なサーバ数は2台である。 |

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**2**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[[3]](#footnote-3)。   * 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[[4]](#footnote-4)。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ない。 |

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[[5]](#footnote-5)によりフェイルオーバ時間が長くなる可能性がある。これは、競合によりレプリケーションの書込み処理がその間停止するためである。 |

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pgrex02が単独でMasterとして稼働しているときに、旧Masterのpgrex01をSlaveとして組み込みしても、自動的にpgrex01がMaster、pgrex02がSlaveに切り替わることはない。 |

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。 |

表 2‑8　 STONITHなし環境の制約

|  |  |
| --- | --- |
| 項番 | 制約事項 |
| 1 | IC-LANが全て切断されると各Pacemakerが互いの状態を把握できなくなる。この場合には、vip-slaveにアクセスするクライアントは古いデータにアクセスする場合がある。 |
| 2 | ディスク故障発生時やMasterの停止に失敗した場合にリソースが不安定な状態で残る可能性がある。その為フェイルオーバが出来ない場合がある。 |

### 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 2‑9　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、『6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などの固有のリソースにしか確認を行わないため、Apacheなど新しくリソースを追加したとしてもその確認を行わない。 |
| 4 | 両系の状態確認にネットワークの通信を用いるので、ツールが使用するLAN（デフォルトはD-LAN）切断時は、それ以外のLANが繋がっていても実行に失敗する。 |
| 5 | PG-REXでインストールしたファイルのディレクトリ構成が両系で同一であること。 |
| 6 | アーカイブログを圧縮する場合、圧縮方式に対応した拡張子を付与しなければならない。  サポートする圧縮方式はgzip (拡張子.gz)のみである。 |

## 推奨設定

本節では、PG-REX運用時の推奨設定を示します。

表 2‑10　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  (運用LAN) | bond0(eth0,eth5) | pgrex01 | 192.168.0.11 |
| pgrex02 | 192.168.0.12 |
| ― | ネットワーク監視先（ping） | 192.168.0.254 |
| D-LAN  (DBレプリケーションLAN) | bond1(eth2,eth4) | pgrex01 | 192.168.2.1 |
| pgrex02 | 192.168.2.2 |
| IC-LAN1  (インターコネクトLAN①) | eth1 | pgrex01 | 192.168.1.1 |
| pgrex02 | 192.168.1.2 |
| IC-LAN2  (インターコネクトLAN②) | eth3 | pgrex01 | 192.168.3.1 |
| pgrex02 | 192.168.3.2 |

※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)

DB領域ディスク

eth5

eth0

bond0

eth0

eth5

bond0

PostgreSQL

PostgreSQL

DB領域ディスク

eth2

eth4

bond1

eth2

eth4

bond1

eth1

eth1

eth3

eth3

Pacemaker

Pacemaker

pgrex01

pgrex02

D-LAN

IC-LAN

S-LAN

bonding

bonding

bonding

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（pgrex01およびpgrex02）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 |
| PING監視用 | － | icmp |

1. PostgreSQLパラメータportの設定値を指定します。
2. 『3.3.4 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（pgrex01とpgrex02）のディレクトリ構成は同一にしてください。

本書では、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  └-pgwal  └-pg\_wal …………………… 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。

### 事前作業

【watchdogデバイスを使用しているhpwdtモジュール停止の設定】

ハードウェアwatchdogのモジュールを停止します。

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

1. rootユーザにてログインし、以下のコマンドでhpwdtモジュールがロードされているかを確認します。

|  |
| --- |
| # dmesg | grep "hpwdt" |

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

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

1. hpwdtモジュールがロードされている場合は、/etc/modprobe.d/blacklist.conf に以下**太字部分**の設定を追記します。

|  |
| --- |
| # vi /etc/modprobe.d/blacklist.conf  ：(省略)  blacklist hpwdt ←この部分を追記 |

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

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

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

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

1. rootユーザにてログインし、以下の手順でiTCO\_wdtモジュールがロードされているかを確認します。①および②を満たす場合はiTCO\_wdtモジュールがロードされています。
2. コマンド実行結果に以下のメッセージが表示されている。

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

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

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

1. iTCO\_wdtモジュールがロードされている場合は、/etc/modprobe.d/blacklist.conf に以下**太字部分**の設定を追記します。

|  |
| --- |
| # vi /etc/modprobe.d/blacklist.conf  ：(省略)  **blacklist iTCO\_wdt** ←この部分を追記 |

### インストール

『Pacemaker関連サイト』を参考にpgrex01とpgrex02へPacemakerをインストールします[[6]](#footnote-6)。

1. 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  ファイルシス 1K-ブロック 使用 使用可 使用% マウント位置  ：(省略)  /dev/sr0 3516418 3516418 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  enabled=1  gpgcheck=1  gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release |

1. yumコマンドを実行し、Pacemakerリポジトリパッケージ[[7]](#footnote-7)を/opt/linux-ha配下に展開します。

|  |
| --- |
| # yum install -y pacemaker-repo-1.1.17-1.1.el7.x86\_64.rpm |

※下線部分は使用するバージョンに適宜読み替えてください。

1. yumのキャッシュをクリアします。

|  |
| --- |
| # yum clean all  (略)  Cleaning up everything  Maybe you want: rm -rf /var/cache/yum, to also free up space taken by orphaned data from disabled or removed repos |

1. Pacemakerをインストールします。

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

以下のメッセージが出力されます。

|  |
| --- |
| 読み込んだプラグイン: product-id, search-disabled-repos, subscription-manager  This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.  依存性の解決をしています  --> トランザクションの確認を実行しています。  (略)  完了しました! |

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  **pacemaker-libs-1.1.17-1.el7.x86\_64**  **pacemaker-cli-1.1.17-1.el7.x86\_64**  **pacemaker-cluster-libs-1.1.17-1.el7.x86\_64**  **pacemaker-1.1.17-1.el7.x86\_64**  **corosynclib-2.4.2-2.el7.x86\_64**  **corosync-2.4.2-2.el7.x86\_64**  **cluster-glue-1.0.12-3.el7.x86\_64**  **cluster-glue-libs-1.0.12-3.el7.x86\_64**  **resource-agents-4.0.1-2.el7.x86\_64**  **crmsh-2.1-9-1.el7.x86\_64**  **libqb-1.0-2-1.el7.x86\_64**  **pm\_diskd-2.3-1.el7.x86\_64**  **pm\_extras-2.3-1.el7.x86\_64**  **pm\_crmgen-2.2-1.el7.noarch**  **pm\_logconv-cs-2.5-1.el7.noarch** |

※下線部分は使用するバージョンに適宜読み替えてください。

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 |

### Pacemaker関連プロセスの自動起動の無効化

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

本作業はrootユーザで行います。

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

### クラスタの基本設定（corosync.conf）

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

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: 239.255.1.2**  **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、『3.3.10 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

pgrex01とpgrex02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）

pgrex01とpgrex02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. pgrex01でCorosync認証設定ファイルを生成します。

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

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

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

### Pacemaker設定ファイルの追加設定（pacemaker）

pgrex01とpgrex02で、Pacemaker設定ファイル（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4行を追加**  **PCMK\_logfile=none**  **PCMK\_logfacility=local1**  **PCMK\_logpriority=info**  **PCMK\_fail\_fast=yes** |

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

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

本作業はrootユーザで行います。

1. corosyncプロセスが故障した場合にcorosyncのwatchdogを効かせたい場合は、Corosync用Unitファイル(corosync.service)を/usr/lib/systemd/system/corosync.serviceから/etc/systemd/system配下にコピーし、下記の修正を行います。

|  |
| --- |
| # cp -p /usr/lib/systemd/system/corosync.service /etc/systemd/system  # vi /etc/systemd/system/corosync.service  (略)  Restart=on-failure # **←コメントアウトを外す**  RestartSec=70 # **←コメントアウトを外す** |

1. softdogを使用する場合は、"corosync.service"に下記の修正を行います。

|  |
| --- |
| # vi /etc/systemd/system/corosync.service  (略)  ExecStartPre=/sbin/modprobe softdog  **↑コメントアウトを外す** |

※softdog使用時のwatchdog発動待ち時間の設定はCorosyncのデフォルト値(6秒)が有効となる

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

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

本作業はrootユーザで行います。

pacemaker.serviceは、/usr/lib/systemd/system/pacemaker.serviceを/etc/systemd/system配下にコピーして作成します。コピー後、ファイルに下記の修正を行います。

|  |
| --- |
| # cp -p /usr/lib/systemd/system/pacemaker.service /etc/systemd/system  # vi /etc/systemd/system/pacemaker.service  (略)  ExecStopPost=/bin/sh -c 'pidof crmd || killall -TERM corosync' **# ←コメントアウトを外す** |

### ログの出力設定（rsyslog.conf）

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

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  (略)**local1.info /var/log/ha-log;RSYSLOG\_TraditionalFileFormat** **# 追記** |

1. 大量のログ出力時の省略を抑止するように設定をします。

/etc/rsyslog.confに以下の設定を追加します。

|  |
| --- |
| # vi /etc/rsyslog.conf  (略)  $SystemLogRateLimitInterval 0 **# 追記**  $imjournalRatelimitInterval 0 **# 追記** |

/etc/systemd/journald.confに以下の設定を追加します。

|  |
| --- |
| # vi /etc/systemd/journald.conf  (略)  RateLimitInterval=0s **# 追記** |

1. 設定を反映するため、rsyslogを再起動します。

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

1. 設定を反映するため、journaldを再起動します。

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

### ログメッセージ制御機能設定（pm\_logconv.conf）

『Pacemaker関連サイト』を参考にpgrex01とpgrex02のログメッセージ制御機能の設定を行います。pm\_logconv.confは、/etc/pm\_logconv.conf.sampleをコピーして作成し、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  #output\_logfacility = local2  #syslogformat = True  **attribute\_ping = not\_defined default\_ping\_set or default\_ping\_set lt 100** **# ネットワーク監視を行う場合の設定**  **attribute\_diskd = not\_defined diskcheck\_status or diskcheck\_status eq ERROR # ディスク監視を行う場合の設定**  **attribute\_diskd\_inner = not\_defined diskcheck\_status\_internal or diskcheck\_status\_internal eq ERROR** **# ディスク監視を行う場合の設定**  #logconv\_logfacility = daemon  #logconv\_logpriority = info  **act\_rsc = vipCheckMaster, vip-master, vip-rep** **# フェイルオーバの発生と成否の判断基準となる**  **# リソースのIDを設定**  （略） |

### ログローテーションの追加設定（syslog）

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

本作業はrootユーザで行います。

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

|  |
| --- |
| # vi /etc/logrotate.d/syslog  /var/log/cron  /var/log/maillog  /var/log/messages  /var/log/secure  /var/log/spooler  **/var/log/ha-log** ←太字部分を追記  {  missingok  sharedscripts  postrotate  /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true  endscript  } |

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

|  |
| --- |
| # vi /etc/logrotate.d/pm\_logconv  /var/log/pm\_logconv.out {  missingok  } |

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

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

### pm\_logconvのプロセス起動

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

本作業はrootユーザで行います。

1. pm\_logconvプロセスの自動起動を有効に設定します。

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

1. pm\_logconvプロセスを即時、起動します。

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

1. pm\_logconvプロセスが起動していることを確認します。

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

### ifcheckdプロセスの起動

pgrex01とpgrex02でifcheckdプロセスを起動します。

本作業はrootユーザで行います。

1. ifcheckdプロセスの自動起動を有効に設定します。

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

1. ifcheckdプロセスを即時、起動します。

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

1. ifcheckdプロセスが起動していることを確認します。

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

## PostgreSQL

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

### PostgreSQLのインストール

『PostgreSQLドキュメント』を参考にpgrex01とpgrex02へPostgreSQLをインストールします。PG-REXで使用するPostgreSQLのバージョンは10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 | postgresql10-libs-10.2-1PGDG.rhel7.x86\_64.rpm |
| 2 | postgresql10-10.2-1PGDG.rhel7.x86\_64.rpm |
| 3 | postgresql10-server-10.2-1PGDG.rhel7.x86\_64.rpm |
| 4 | postgresql10-contrib-10.2-1PGDG.rhel7.x86\_64.rpm |
| 5 | postgresql10-docs-10.2-1PGDG.rhel7.x86\_64.rpm |

* 下線部分は使用するバージョンに適宜読み替えてください。
* PL/PerlやPL/Tclなどの各種言語インターフェイスが必要な場合は、それぞれに対応するパッケージをインストールしてください。

PostgreSQLをRPMパッケージからインストールします。インストールは以下の順序で行います。

|  |
| --- |
| # rpm -ivh postgresql10-libs-10.2-1PGDG.rhel7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:postgresql10-libs-10.2-1PGDG.rhel################################# [100%]**  # rpm -ivh postgresql10-10.2-1PGDG.rhel7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:postgresql10-10.2-1PGDG.rhel7 ################################# [100%]**  # rpm -ivh postgresql10-server-10.2-1PGDG.rhel7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:postgresql10-server-10.2-1PGDG.rh################################# [100%]**  # rpm -ivh postgresql10-docs-10.2-1PGDG.rhel7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:postgresql10-docs-10.2-1PGDG.rhel################################# [100%]**  # rpm -ivh postgresql10-contrib-10.2-1PGDG.rhel7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:postgresql10-contrib-10.2-1PGDG.r################################# [100%]** |

※下線部分は使用するバージョンに適宜読み替えてください。

PostgreSQLのRPMパッケージをインストールすると、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-10 …………………… 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  **postgresql10-server-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-contrib-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-docs-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-libs-10.2-1PGDG.rhel7.x86\_64** |

※下線部分は使用するバージョンに適宜読み替えてください。

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

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

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

|  |
| --- |
| # groupmod -g 26 postgres **← 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) groups=26(postgres)** |

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

|  |
| --- |
| # passwd postgres **← 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: すべての認証トークンが正しく更新できました。** |

### 環境変数の設定

pgrex01とpgrex02で/var/lib/pgsql/.bash\_profileに表 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-10/bin:$PATH |
| PGDATA | DBクラスタのパス | （例）/dbfp/pgdata/data |

* pgrex01とpgrex02で同じ設定値を使用してください。

|  |
| --- |
| $ vi ~/.bash\_profile  (略)  **export PATH=/usr/pgsql-10/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-10/bin:**… |

### DBクラスタ用ディレクトリの作成

pgrex01とpgrex02で、DBクラスタを格納するディレクトリ（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/pgwal）、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。なお、ディレクトリが既に存在する場合は、ディレクトリを空にします。

本作業はrootユーザで行います。

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

### DBクラスタの初期化

pgrex01で、postgresユーザにてDBクラスタを初期化します。

|  |
| --- |
| $ initdb -D /dbfp/pgdata/data -X /dbfp/pgwal/pg\_wal --encoding=UTF-8 --no-locale --data-checksums  (略)  Success. You can now start the database server using:    **pg\_ctl -D /dbfp/pgdata/data -l logfile start** |

* 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。

pgrex01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/pgwal/pg\_wal  **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/pgwal/pg\_wal |

### postgresql.confの編集

pgrex01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 | 設定値 | 備考 |
| listen\_addresses | \* | ― |
| wal\_level | replica | logicalを指定することも可能。 |
| max\_wal\_senders | 10 | PG-REXでは、レプリケーション機能を利用するため、1以上を指定する必要がある。ただし、pg\_basebackupによる運用中のバックアップ取得など、Slave以外からの接続に備え、余裕を持たせて指定することを推奨する。 |
| wal\_keep\_segments | 32 | 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 | 必須の設定。 |
| max\_standby\_streaming\_delay | -1 | 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 | PG-REXでは、Slaveの監視クエリがキャンセルされるのを回避するために、-1（キャンセルを無効）を設定する。 |
| synchronous\_standby\_names | '' | PG-REX 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 | 本マニュアルではレプリケーションの信頼性を重視するため、onを推奨する。remote\_writeを指定した場合、故障発生時にデータを損失する可能性がある。  remote\_applyを指定した場合、on指定時と同様の信頼性となるが、トランザクションの応答時間が長くなる。 |
| archive\_mode | always | 必須の設定。 |
| archive\_command | '/bin/cp %p /dbfp/pgarch/arc1/%f' | 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[[8]](#footnote-8)にもgzipを使用すること。 |
| restart\_after\_crash | off | 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 | Slaveの故障や両系間の通信断をMasterがすぐに検知できるように、タイムアウトを有効にすることを推奨する。postgresql.confで設定するTCP keepaliveに関する設定[[9]](#footnote-9)だけでは、検知に時間がかかることがあり、異常時のダウンタイムが長くなることがある。ただし、設定値が小さすぎると誤検知によりSlaveが切り離されることがあるため、設定値は事前検証等をして注意して決めること。 |
| wal\_receiver\_timeout | 20s | wal\_sender\_timeoutと同程度に揃える必要がある。 |
| hot\_standby\_feedback | on | 必須の設定。 |
| max\_replication\_slots | 10 | pg\_basebackupによるバックアップ取得などに備え、余裕を持たせて指定することを推奨する。 |

設定例を以下に示します。

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

### レプリケーションユーザの作成

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

本作業はpostgresユーザで行います。

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 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"**  List of roles  Role name | Attributes | Member of  -----------+-------------+-----------  **repuser | Replication | {}** |

PostgreSQLを停止します。

|  |
| --- |
| $ pg\_ctl stop  **サーバ停止処理の完了を待っています.......完了**  **サーバは停止しました** |

### pg\_hba.confの編集

pgrex01でpg\_hba.confを作成します。本節は、PG-REXを構成するのに必要な設定のみ説明しています。PostgreSQL一般の設定については『PostgreSQLドキュメント』を参照してください。

PG-REXを構成するのに必要な設定は表 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 | 『3.4.6 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の作成

pgrex01とpgrex02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**および**相手ノードのD-LANの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 | 『3.4.6 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 | 『3.4.6 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。 |

pgrex01で.pgpassを作成します。

|  |
| --- |
| $ vi ~/.pgpass  ***192.168.2.3:5432*:replication:repuser:reppasswd**  ***192.168.2.2:5432*:replication:repuser:reppasswd**  $ chmod 600 ~/.pgpass |

pgrex02で.pgpassを作成します。

|  |
| --- |
| $ vi ~/.pgpass  ***192.168.2.3:5432*:replication:repuser:reppasswd**  ***192.168.2.1:5432*: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

pgrex01とpgrex02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-10.0-1.el7.noarch.rpm |
| 2 | IO\_Tty-1.11-1.el7.x86\_64.rpm |
| 3 | Net\_OpenSSH-0.62-1.el7.x86\_64.rpm |

※下線部分は使用するバージョンに適宜読み替えてください。

PG-REX運用補助ツールをRPMパッケージからインストールします。インストールは以下の順序で行います。

|  |
| --- |
| # rpm -ivh Net\_OpenSSH-0.62-1.el7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:Net\_OpenSSH-0.62-1.el7 ################################# [100%]**  # rpm -ivh IO\_Tty-1.11-1.el7.x86\_64.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:IO\_Tty-1.11-1.el7 ################################# [100%]**  # rpm -ivh pg-rex\_operation\_tools\_script-10.0-1.el7.noarch.rpm  **準備しています... ################################# [100%]**  **更新中 / インストール中...**  **1:pg-rex\_operation\_tools\_script-10.################################# [100%]** |

※下線部分は使用するバージョンに適宜読み替えてください。

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-1.11-1.el7.x86\_64**  # rpm -qa | grep Net\_OpenSSH  **Net\_OpenSSH-0.62-1.el7.x86\_64**  # rpm -qa | grep pg-rex  **pg-rex\_operation\_tools\_script-10.0-1.el7.noarch** |

※下線部分は使用するバージョンに適宜読み替えてください。

### pg-rex\_tools.confの編集

pgrex01とpgrex02で/etc/pg-rex\_tools.confの設定を行います。運用補助ツールを使用するために必要な設定、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。 |
| STONITH | disable | STONITH導入環境の場合は**enable**、それ以外の場合は**disable**を指定する。 |
| VIP\_SLAVE | disable | vip\_slave を使用する環境の場合は**enable**、それ以外の場合は**disable**を指定する。 |
| DISKD\_ResourceID | clnDiskd1, clnDiskd2 | **diskdのリソースID**を複数設定する。 |

設定例を以下に示します。

|  |
| --- |
| # vi /etc/pg-rex\_tools.conf  **D-LAN\_IPAddress = *192.168.2.1* , *192.168.2.2***  **Archive\_dir = /dbfp/pgarch/arc1**  **STONITH = disable**  **VIP\_SLAVE = disable**  **DISKD\_ResourceID = clnDiskd1, clnDiskd2**  （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\_hostsに相手先のD-LANのIPアドレスに対する接続登録をする必要があります。

本作業はrootユーザで行います。

pgrex01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.  ECDSA 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*' (ECDSA) 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** |

pgrex02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.  ECDSA 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*' (ECDSA) 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-REX10\_NoSTONITH\_pm\_crmgen.xlsx)を用いて、クラスタ内のリソース構成、動作条件、パラメータ、配置制約などを設定します。

### pm\_crmgenの概要

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

PG-REXでのcrmファイル生成までの手順は以下のとおりです。

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

以降、『pm\_crmgen環境定義書』を編集します。

### 変更不要の設定

『pm\_crmgen環境定義書』の下記の設定は変更不要です。

* 表1-1 クラスタ・ノード属性
* 表2-1 クラスタ・プロパティ
* 表3-1 リソース・デフォルト
* 表3-2 オペレーション・デフォルト
* 表5-1 リソース・パラメータ
* 表6-1 STONITHの実行順序
* 表11-1 リソースチケット制約
* 表12-1 追加設定
* 表13-1 ALERT設定

### リソース構成の設定

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

### リソース（Master側仮想IP）の設定

『pm\_crmgen環境定義書』の表7-1-1、7-1-4にPostgreSQLのMaster側接続用の仮想IPを設定します。

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

### リソース（PostgreSQL）の設定

『pm\_crmgen環境定義書』の表7-1-6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 |
| rep\_mode | sync | **同期モード**を指定する。 |
| node\_list | pgrex01 pgrex02 | **pgrex01とpgrex02のノード名**を指定する（ノード名の間はスペース区切り）。 |
| master\_ip | 192.168.2.3 | 『pm\_crmgen環境定義書』の表7-1-2に設定した**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[[10]](#footnote-10)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 | 『3.4.6 レプリケーションユーザの作成』で作成した、**レプリケーションユーザ**を指定する。 |
| primary\_conninfo\_opt | keepalives\_idle=60 keepalives\_interval=5 keepalives\_count=5 | **TCPキープアライブ用の制御パラメータを設定する。**PG-REXでは左記の設定値を推奨する。 |
| stop\_escalate | 0 | このパラメータは、Master停止時にPostgreSQLの高速シャットダウンを実行してから即時シャットダウンにエスカレーションするまでの待ち時間を指定する。**0**を指定した場合、高速シャットダウンは実行されずに、即座に即時シャットダウンが実行される。 |
| xlog\_check\_count | 0 | このパラメータは、2台同時に起動する際、どちらがMasterとして起動するかをチェックする回数を指定する（デフォルトは3）。  このパラメータを設定すると、Master起動の際にmonitor間隔×設定値の時間だけSlaveで待機することになり、その分Masterとして起動する時間が遅くなる。  本マニュアルでは、2台同時に起動するケースがないため、**0**を指定する。 |

### リソース（ネットワーク監視）の設定

『pm\_crmgen環境定義書』の表7-2-1にネットワーク監視を設定します。なお、PG-REXではS-LANを監視します。監視するネットワークのIPアドレスとしてS-LANのデフォルトゲートウェイ等を設定してください。

### リソース（ディスク監視）の設定

『pm\_crmgen環境定義書』に表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監視の設定をします。DBクラスタとOSとを異なるディスクに配置している場合は、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17行）を削除します。

### リソース配置制約の設定

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

### リソース同居制約の設定

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

### リソース起動順序制約の設定

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

### リソース（Slave側仮想IP）の設定

Slaveへのデータベース接続を利用する場合は、『pm\_crmgen環境定義書』の表7-1-3、表7-1-5にSlave側接続用の仮想IPを設定します。

Slaveへのデータベース接続を利用しない場合は、『pm\_crmgen環境定義書』の表4-1、表7-1-3、表7-1-5、表8-1からvip-slaveの設定値をコメントアウトします。

### crmファイルの作成

以下の操作を行い、crmファイルを作成します。

1. Microsoft® Excelを使って『pm\_crmgen環境定義書』の「pm\_crmgen\_環境定義書(PG-REX10)」シートを修正し、修正したシートをCSV形式で保存します。

本書ではファイル名は「PG-REX10\_pm\_crmgen\_env.csv」とします。

1. 保存したcsvファイル（PG-REX10\_pm\_crmgen\_env.csv）を、pgrex01に転送します。

ファイル転送の際に文字コード変換を行わないよう注意してください。

1. pgrex01上で、pm\_crmgenコマンドを使用し、（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-REX10\_pm\_crmgen\_env.crm」とします。

|  |
| --- |
| # pm\_crmgen -o PG-REX10\_pm\_crmgen\_env.crm PG-REX10\_pm\_crmgen\_env.csv |

生成後のcrmファイルを確認します。

|  |
| --- |
| # ls -l PG-REX10\_pm\_crmgen\_env.crm  -**rw**-**r**--**r**-- 1 **root root** 6722 1月 30 10:05 2014 **PG-REX10\_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-REX10\_pm\_crmgen\_env.crm  # chmod 644 PG-REX10\_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pgrex01とpgrex02の両方からアンインストールします。PG-REXが起動している場合は「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  **postgresql10-server-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-libs-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-contrib-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-10.2-1PGDG.rhel7.x86\_64**  **postgresql10-docs-10.2-1PGDG.rhel7.x86\_64** |

※下線部分は使用するバージョンに適宜読み替えてください。

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 postgresql10-contrib-10.2-1PGDG.rhel7.x86\_64  # rpm -e postgresql10-server-10.2-1PGDG.rhel7.x86\_64  # rpm -e postgresql10-docs-10.2-1PGDG.rhel7.x86\_64  # rpm -e postgresql10-10.2-1PGDG.rhel7.x86\_64  # rpm -e postgresql10-libs-10.2-1PGDG.rhel7.x86\_64  # rpm -qa | grep postgresql  **（出力なし）** |

※下線部分は使用するバージョンに適宜読み替えてください。

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 erase -y 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。

|  |
| --- |
| 読み込んだプラグイン: product-id, search-disabled-repos, subscription-manager  This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.  --> トランザクションの確認を実行しています。  (略)  **完了しました!** |

Pacemakerリポジトリパッケージのアンインストールを行います。

|  |
| --- |
| # yum -y pacemaker-repo  # yum list installed | grep pacemaker-repo  **（出力なし）** |

※下線部分は使用するバージョンに適宜読み替えてください。

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

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**11**
* /usr/lib/ocf/resource.d/pacemaker[[11]](#footnote-11)

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-10.0-1.el7.noarch  # rpm -e Net\_OpenSSH-0.62-1.el7.x86\_64  # rpm -e IO\_Tty-1.11-1.el7.x86\_64  # rpm -qa | grep IO\_Tty  **（出力なし）**  # rpm -qa | grep Net\_OpenSSH  **（出力なし）**  # rpm -qa | grep pg-rex  **（出力なし）** |

※下線部分は使用するバージョンに適宜読み替えてください。

# 起動と停止

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

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、pgrex01をMasterとして起動します。

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

|  |
| --- |
| $ pg\_ctl start  **サーバの起動完了を待っています....**  ：(略)  **完了**  **サーバ起動完了**  $ pg\_ctl stop  **サーバ停止処理の完了を待っています....完了**  **サーバは停止しました** |

【注意】

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

1. pgrex01で起動禁止フラグのファイルが存在する場合削除します。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. pgrex01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:**  **↑pgrex02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 が停止していることを確認**  **…[OK]**  **2. 稼働中の Master が存在していないことを確認**  **…[OK]**  **3. Master として稼働することが出来るかを確認**  **…[OK]**  **4. 起動禁止フラグの存在を確認**  **…[OK]**  **5. Pacemaker 起動**  **…[OK]**  **6. Master の起動確認**  **…[OK]**  **ノード(pgrex01)が Master として起動しました** |

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

max\_connections、max\_worker\_processes、max\_prepared\_transactions、max\_locks\_per\_transactions

より詳しい説明については、PostgreSQLドキュメント[[12]](#footnote-12)を参照してください。

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

1. 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-REX10\_pm\_crmgen\_env.crm  **root@192.168.2.2's password:**  **↑pgrex02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 が停止していることを確認**  **…[OK]**  **2. 稼働中の Master が存在していないことを確認**  **…[OK]**  **3. 起動禁止フラグの存在を確認**  **…[OK]**  **/var/lib/pacemaker/cib 配下に既に cib ファイルがあります**  **削除して反映しても宜しいでしょうか？ (y/N) y** **← [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(pgrex01)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、pgrex02をSlaveとして起動します。

1. pgrex02で起動禁止フラグのファイルが存在する場合削除します。本作業はrootユーザで行います。

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

1. pgrex02でSlaveを起動します。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『5.2.1 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:**  **↑pgrex01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 が停止していることを確認**  **…[OK]**  **2. 稼働中の Master が存在していることを確認**  **…[OK]**  **3. 起動禁止フラグが存在しないことを確認**  **…[OK]** |

1. 起動方法を選択します。以下の手順では、ベースバックアップを取得する場合を示します。

|  |
| --- |
| **4. DB クラスタの状態を確認**  **4.1 現在のDBクラスタのまま起動が可能か確認**  **DB クラスタが存在していません**  **4.2 巻き戻しを実行することで起動が可能か確認**  **DB クラスタが存在していません**  **4.3 ベースバックアップを取得することが可能か確認**  **以下の方法で起動が可能です**  **b) ベースバックアップを取得してSlaveを起動**  **q) Slaveの起動を中止する**  **起動方法を選択してください(b/q) b** **← [b]を入力し[Enter]キーを押下**  **5. IC-LAN が接続されていることを確認**  **...[OK]**  **6. Master からベースバックアップ取得**  **22631/22631 kB (100%), 1/1 tablespace**  **NOTICE: pg\_stop\_backup complete, all required WAL segments have been archived**  **...[OK]**  **7. Master のアーカイブディレクトリと同期**  **receiving incremental file list**  **./**  **000000010000000000000002.partial**  **00000002.history**  **000000020000000000000003.00000028.backup**  **sent 125 bytes received 16779892 bytes 11186678.00 bytes/sec**  **total size is 16777559 speedup is 1.00**  **receiving incremental file list**  **000000010000000000000001**  **000000020000000000000002**  **000000020000000000000003**  **sent 68 bytes received 50338126 bytes 33558796.00 bytes/sec**  **total size is 67109207 speedup is 1.33**  **...[OK]**  **8. /var/lib/pacemaker/cib 配下のファイルを削除**  **...[OK]**  **9. Slave の起動 (アーカイブリカバリ対象 WAL セグメント数: 1)**  **...[OK]**  **10. Slave の起動確認**  **...[OK]**  **ノード(pgrex02)が Slave として起動しました** |

## 両系の停止

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

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

* Masterの停止手順については、『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 (pgrex02)を停止しました** |

## Masterの停止

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

本作業は停止対象のノードにて、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 (pgrex01)を停止しました** |

【注意】

Slave稼働中の場合、以下の問い合わせが出力されます。

|  |
| --- |
| **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。

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

# メンテナンス時の対応

本章は、PostgreSQLのバックアップ取得の操作手順、アーカイブログ削除の操作手順、およびPG-REX(MasterまたはSlave)の停止を伴うメンテナンス(PG-REXのマイナーバージョンアップやハードウェアの増設等の作業)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pgrex01、Slaveをpgrex02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』[[13]](#footnote-13)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。

### PostgreSQLアーカイブログの削除

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

PostgreSQLのアーカイブログ(および付随するバックアップ履歴ファイル[[14]](#footnote-14)、タイムライン履歴ファイル[[15]](#footnote-15))は、MasterおよびSlaveのリカバリに加え、『5.1.2 PostgreSQLのバックアップ』の方法で取得したバックアップからのリカバリのいずれでも必要とならない場合に削除できます。

pg-rex\_archivefile\_delete ツールを使うことで、これらの不要なファイルの削除または移動を自動的に行うことができます。

本作業はroot、もしくはPG-REXで管理しているDBクラスタを作成したユーザのみ実行可能です。pg-rex\_tools.conf での設定によって手順中のパスワードの与え方等は異なります。また手順中に現れる*/somewhere/basebackups/oldest*は保存すべき最古のベースバックアップを格納しているディレクトリを示します。

|  |
| --- |
| $ pg-rex\_archivefile\_delete -m */somewhere/basebackups/oldest*  **\*\*\*\* 1. 実行準備 \*\*\*\***  **移動モードで実行します**  **環境設定ファイル (pg-rex\_tool.conf) を読み込みます**  **postgres@192.168.2.2's password:**  **↑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 "pgrex01" の現時点の PGDATA "/dbfp/pgdata/data " からリカバリに必要な最初の WAL ファイル名を取得します**  **"00000052000000010000003E"**  **相手のノード "pgrex02" の現時点の PGDATA "/dbfp/pgdata/data " からリカバリに必要な最初の WAL ファイル名を取得します**  **"00000052000000010000003D"**  **\*\*\*\* 3. 削除基準の算出 \*\*\*\***  **削除基準を "00000051000000010000003A" としました**  **\*\*\*\* 4. アーカイブログの削除 \*\*\*\***  **削除対象のリストに "0000004F0000000100000035.00000020.backup" を追加します**  **削除対象のリストに "0000004D0000000100000034" を追加します**  **:(略)**  **移動先ディレクトリ "/dbfp/pgarch/arc1/20140224\_150226" を作成しました**  **-- 移動 -- 0000004F0000000100000035.00000020.backup**  **:(略)**  **アーカイブログの移動に成功しました**  **移動モード実行のため、移動したファイルは "/dbfp/pgarch/arc1/20140224\_150226" に格納されています** |

この実行例ではコマンドに -m オプションを指定しているため、作業の終了後には /dbfp/pgarch/arc1/20140224\_150226 ディレクトリに不要となったファイルが格納されています。これを削除するか別のディスク等に移動させるなどすることで作業は完了します。

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

## 計画的な系切り替え

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

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pgrex01、Slaveをpgrex02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pgrex01で系切り替えを実行する場合の手順を示します。

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

|  |
| --- |
| # pg-rex\_switchover  **root@192.168.2.2's password:**  **↑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 : pgrex01 -> pgrex02**  **Slave : pgrex02 -> pgrex01** **← 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。『6章 故障対応』に従い、手動復旧を試みてください。

|  |
| --- |
| **3. CHECKPOINT の実行**  **…[OK]**  **\*\*\*\* 系切り替えを実行 \*\*\*\***  **4. Pacemaker の監視を停止**  **…[OK]**  **5. Master (pgrex01) の PostgreSQL を停止**  **…[OK]**  **6. Pacemaker の監視を再開し系切り替えを実行**  **…[OK]**  **7. pgrex02 が新 Master になったことを確認**  **\*\*\*\* pgrex02 が Master として起動しました \*\*\*\***  **8. pgrex01 の Pacemaker を停止**  **…[OK]**  **9. pgrex01 で Slave を起動**  **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**  **\*\*\*\* pgrex01 が Slave として起動しました \*\*\*\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\*\*\*\* 系切り替えが正常に完了しました \*\*\*\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **[ 現在のクラスタ状態 ]**  **Master : pgrex02**  **Slave : pgrex01** **← 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4～6の間)にコマンドが異常終了した場合は、Pacemakerの監視が停止している可能性があるため、crm\_monコマンドを実行しクラスタ状態を確認してください。crm\_monコマンドの実行結果のリソース情報表示部に”unmanaged”が表示されている場合はPacemakerの監視が停止しています。Pacemakerの監視が停止している場合は、Pacemakerの監視を再開してください。Pacemakerの監視を再開するには以下のコマンドを実行します。

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

## 計画的なメンテナンス

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pgrex01、Slaveをpgrex02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pgrex02に切り替えるため、pgrex01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

pgrex02でcrm\_monコマンドを実行し、以下を確認します。

* pgrex01の状態が"OFFLINE"になっていること。
* clnPing、clnDiskdリソースが、pgrex02上だけ"Started"になっていること。
* msPostgresqlリソースが、pgrex02上でMasterとして動作していること。

|  |
| --- |
| [pgrex02] # crm\_mon -fA  ：（略）  Last updated: *日時表示*  Last change: *日時表示* by root via crm\_attribute on *ノード名*  ：（略）  Online: [ pgrex02 ]  **OFFLINE: [ pgrex01 ]**  ：（略）  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pgrex02 ]**  Clone Set: clnPing [prmPing]  **Started: [ pgrex02 ]**  Clone Set: clnDiskd1 [prmDiskd1]  **Started: [ pgrex02 ]**  Clone Set: clnDiskd2 [prmDiskd2]  **Started: [ pgrex02 ]**  ：（略） |

1. pgrex01のメンテナンス作業

pgrex01で、計画されたメンテナンス作業を実施します。

1. pgrex01のPacemakerを起動

pgrex01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、『5.3 計画的な系切り替え』に従い、pgrex01をMaster、pgrex02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. pgrex02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. pgrex02のメンテナンス作業

pgrex02で、計画されたメンテナンス作業を実施します。

1. pgrex02のPacemakerを起動

『4.3 Slaveの起動』の手順に従い、pgrex02のPacemakerを起動します。

# 故障対応

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

監視者による作業にはクラスタ状態確認と復旧があります。

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

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

なお、本作業にはスーパーユーザ権限が必要となります。

## 前提条件

この章の前提条件を以下に示します。

* 両系が稼働している状態でpgrex01の故障が起きたと仮定しています。

## 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。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。

以下にcrm\_monコマンド実行時の表示例を示します。

# crm\_mon -fA -1

Stack: corosync

Current DC: pgrex01 (*バージョン*) - partition with quorum

Last updated: *日時表示*

Last change: *日時表示 by root via crm\_attribute on ノード名*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): Started pgrex01

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pgrex01

vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): Started pgrex02

vip-slave (ocf::heartbeat:IPaddr2): Started pgrex02

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex01 ]

Slaves: [ pgrex02 ]

Clone Set: clnPing [prmPing]

Started: [ pgrex01 pgrex02 ]

Clone Set: clnDiskd1 [prmDiskd1]

Started: [ pgrex01 pgrex02 ]

Clone Set: clnDiskd2 [prmDiskd2]

Started: [ pgrex01 pgrex02 ]

Node Attributes:

\* Node pgrex01:

+ default\_ping\_set : 100

+ diskcheck\_status : normal

+ diskcheck\_status\_internal : normal

+ master-pgsql : 1000

+ pgsql-data-status : LATEST

+ pgsql-master-baseline : 0000000002000270

+ pgsql-status : PRI

+ ringnumber\_0 : 192.168.1.1 is UP

+ ringnumber\_1 : 192.168.3.1 is UP

\* Node pgrex02:

：（略）

Migration Summary:

\* Node pgrex01:

\* Node pgrex02:

Failed Actions:

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

③属性情報表示部

②リソース情報表示部

①ノード情報表示部

### 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。

①ノード情報表示部： 各ノードの名前や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。なお、稼働状態にはOnline、OFFLINEが存在します。

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名前が表示されます。

各リソースが示す稼働状況を表 6‑1に示します。

表 6‑1　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: [ pgrex01 ]  Slaves: [ pgrex02 ] | 両系でリソースが稼働中 |
| 2 | Masters: [ pgrex02 ] | 片系(pgrex02)でリソースが稼働中 |
| 3 | Masters: [ pgrex01 ] | 片系(pgrex01)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 | VIPcheckリソース  (vipCheckMaster) | Started pgrex01 | pgrex01でリソースが稼働中 |
| 6 | Started pgrex02 | pgrex02でリソースが稼働中 |
| 7 | 表示なし | リソースが停止中 |
| 8 | VIPcheckリソース  (vipCheckSlave) | Started pgrex01 | pgrex01でリソースが稼働中 |
| 9 | Started pgrex02 | pgrex02でリソースが稼働中 |
| 10 | 表示なし | リソースが停止中 |
| 11 | IPaddr2リソース (vip-master)  (vip-rep) | Started pgrex01 | pgrex01でリソースが稼働中 |
| 12 | Started pgrex02 | pgrex02でリソースが稼働中 |
| 13 | 表示なし | リソースが停止中 |
| 14 | IPaddr2リソース  (vip-slave) | Started pgrex02 | pgrex02でリソースが稼働中 |
| 15 | Started pgrex01 | pgrex01でリソースが稼働中 |
| 16 | 表示なし | リソースが停止中 |
| 17 | cloneリソース  (clnPing) | Started: [ pgrex01 pgrex02 ] | 両系でリソースが稼働中 |
| 18 | Started: [ pgrex02 ] | 片系(pgrex02)でリソースが稼働中 |
| 19 | Started: [ pgrex01 ] | 片系(pgrex01)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 |
| 21 | cloneリソース  (clnDiskd1)  (clnDiskd2) | Started: [ pgrex01 pgrex02 ] | 両系でリソースが稼働中 |
| 22 | Started: [ pgrex02 ] | 片系(pgrex02)でリソースが稼働中 |
| 23 | Started: [ pgrex01 ] | 片系(pgrex01)でリソースが稼働中 |
| 24 | 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 6‑2に示します。

表 6‑2　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(DB領域) | diskcheck\_status | normal | ERROR |
| 3 | 内蔵ディスク(OS領域) | diskcheck\_status\_internal | normal | ERROR |
| 4 | PostgreSQL | master-pgsql | 1000(pgrex01)  100(pgrex02) | -INFINITY |
| 5 | pgsql-data-status | LATEST(pgrex01)  STREAMING|SYNC (pgrex02) | 左記以外の値 |
| 6 | pgsql-master-baseline[[16]](#footnote-16) | LSN (pgrex01)  表示なし(pgrex02) | 左記以外の値 |
| 7 | pgsql-status | PRI(pgrex01)  HS:sync(pgrex02) | 左記以外の値 |
| 8 | ringnumber\_0[[17]](#footnote-17) | 192.168.1.2 is up | 表示なし[[18]](#footnote-18) |
| 9 | pgsql-xloc-loc | 表示なし | 表示[[19]](#footnote-19) |

④故障回数表示部： ノード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 pgrex01 'not running' (7): call=79, status=complete,

① ② ③ ④ ⑤

exitreason='none', 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 6‑3に示します。

表 6‑3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 6‑3　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 | pgrex01とpgrex02のノード状態が"Online"状態になっていることを確認する。 | |
| 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。  pgrex01： msPostgresql (Master)  vipCheckMaster  vip-master  vip-rep  pgrex02： msPostgresql (Slave)  vipCheckSlave  vip-slave |
|  |  | 【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 (『6.3.1 ノード情報・リソース情報・故障回数表示部の確認』)

両系起動状態[[20]](#footnote-20)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。

片系起動状態[[21]](#footnote-21)の場合は、停止ノードに発生した故障を特定します。

* 属性情報表示部の確認 (『6.3.2 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 (『6.3.3 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 (『6.3.4 /var/log/pm\_logconv.outの確認』)

IC-LAN故障であることを確認します。

故障箇所特定までのフローを図 6‑1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

片系起動状態の場合

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(monitor/vip-master/vip-slave/vip-rep)

ノード故障

6.3.1 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 6‑1　故障特定フロー

### 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。

* 片系起動状態の場合

『6.3.4 /var/log/pm\_logconv.outの確認』に遷移します。

* 両系起動状態の場合

1. IC-LAN故障の特定

pgrex01とpgrex02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、pgrex01とpgrex02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、『6.4 故障一覧』のIC-LAN故障を参照します。

pgrex01におけるCurrent DCとノード状態表示部の表示例

←Current DCがpgrex01

pgrex02がOFFLINE

Current DC: pgrex01 (*バージョン*) - partition with quorum

：（略）

Online: [ pgrex01 ]

OFFLINE: [ pgrex02 ]

pgrex02におけるノード状態表示部の表示例

←Current DCがpgrex02

pgrex01がOFFLINE

Current DC: pgrex02 (*バージョン*) - partition with quorum

：（略）

Online: [ pgrex02 ]

OFFLINE: [ pgrex01 ]

1. D-LAN故障の特定

pgrex01でcrm\_monコマンドを実行し、リソース情報、属性情報を確認します。

：（略）

\* Node pgrex02:

+ default\_ping\_set : 100

+ diskcheck\_status : normal

+ diskcheck\_status\_internal : normal

+ master-pgsql : -INFINITY

+ pgsql-data-status : **DISCONNECT**

+ pgsql-status : **HS:alone**

+ ringnumber\_0 : 192.168.1.2 is UP

+ ringnumber\_1 : 192.168.3.2 is UP

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

次に、pgrex01のPostgreSQLのサーバログファイル[[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/postgresql**21**

Mar 12 16:39:46 **pgrex01** **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、『6.4 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 6‑4に示す状態に応じて該当する項へ遷移します。

表 6‑4　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：『6.3.2 属性情報表示部の確認』 |
|  | PG-REXリソース (msPostgresql)： Masters: [ pgrex01 ]  (リソースF/O無) | 表示なし |  |
| pgrex01： fail-count=1 | リソース故障：『6.3.3 制御エラー情報表示部の確認』 |
|  | PG-REXリソース (msPostgresql)： Masters: [ pgrex02 ]  (リソースF/O有) | 表示なし | ハードウェア故障：『6.3.2 属性情報表示部の確認』 |
|  |  | pgrex01： fail-count=1000000 | リソース故障：『6.3.3 制御エラー情報表示部の確認』 |
|  |  | pgrex01： fail-count=1 |  |
|  | 表示なし | pgrex02： fail-count=1 |  |

### 属性情報表示部の確認

属性情報表示部の確認では、pgrex01またはpgrex02でcrm\_monコマンドを実行し、異常値を示している属性名を調べ、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。

pgrex01でdefault\_ping\_setの属性値が異常

： (略)

\* Node pgrex01:

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

+ diskcheck\_status : normal

+ diskcheck\_status\_internal : normal

+ master-pgsql : -INFINITY

+ pgsql-data-status : DISCONNECT

+ pgsql-status : STOP

+ ringnumber\_0 : 192.168.1.1 is UP

+ ringnumber\_1 : 192.168.3.1 is UP

\* Node pgrex02:

+ default\_ping\_set : 100

+ diskcheck\_status : normal

+ diskcheck\_status\_internal : normal

+ master-pgsql : 1000

+ pgsql-data-status : LATEST

+ pgsql-master-baseline : 00000139210029D8

+ pgsql-status : PRI

+ ringnumber\_0 : 192.168.1.2 is UP

+ ringnumber\_1 : 192.168.3.2 is UP

:（略）

上記の例では、pgrex01のNode Attributesに表示されているdefault\_ping\_setの属性値が異常を示していることから、pgrex01にてS-LANの故障を検知したことを示しています。

属性情報表示部の出力と対応する故障内容を表 6‑5に示します。実施後は『6.4 故障一覧』に遷移し、特定された故障を参照します。

表 6‑5　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 (pgrex01) | 属性値 (pgrex02) |
| default\_ping\_set | 0 | 100 | S-LAN故障 (pgrex01) |
|  | 100 | 0 | S-LAN故障 (pgrex02) |
|  | 0 | 0 | ルータ故障 (pgrex01、pgrex02) |
| diskcheck\_status,  diskcheck\_status\_internal | ERROR | normal | 内蔵ディスク故障 (pgrex01) |
|  | normal | ERROR | 内蔵ディスク故障 (pgrex02) |
|  | ERROR | ERROR | 内蔵ディスク故障 (pgrex01、pgrex02) |
| 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**24** | 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、pgrex01またはpgrex02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『6.4 故障一覧』に遷移し、特定された故障を参照します。

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 **pgrex01** 'not running' (7): call=79, status=complete, exitreason='none',

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 **pgrex01** 'unknown error' (5): call=182, status=Timed Out, exitreason='none',

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 **pgrex01** 'unknown error' (1): call=87, status=Timed Out, exitreason='none',

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 **pgrex01** 'unknown error' (1): call=22, status=complete, exitreason='none', last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

\* **pgsql**\_**monitor**\_10000 on **pgrex01** '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 **pgrex02** 'unknown error' (1): call=16, status=complete, exitreason='none', 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 **pgrex01**:

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

\* Node pgrex02:

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:

\* **vip-master**\_**monitor**\_10000 on **pgrex01** 'unknown error' (1): call=22, status=complete, exitreason='none', last-rc-change='*日時表示*', queued=*XX*ms, exec=*XX*ms

\* **pgsql\_stop\_0** on **pgrex01** 'unknown error' (1): call=87, status=Timed Out, exitreason='none',

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。故障を特定後は、『6.4 故障一覧』に遷移し、特定された故障を参照します。

1. IC-LAN故障の確認

正規表現： ‘.\*Node.\*is lost.\*’

# grep ’.\*Node.\*is lost.\*’ /var/log/pm\_logconv.out

**→以下はpgrex01で検索した場合の出力例**

Dec 4 14:22:56 pgrex01 warning: **Node pgrex02 is lost.**

※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 6‑6に示します。

表 6‑6　故障一覧

|  |  |  |
| --- | --- | --- |
| No. | 故障 | 参照節番号 |
| 1 | ルータ故障 | 『6.5 ルータ故障』 |
| 2 | S-LAN故障 | 『6.6 S-LAN故障』 |
| 3 | D-LAN故障 | 『6.7 D-LAN故障』 |
| 4 | リソース故障(monitor) | 『6.8 リソース故障(monitor)』 |
| 5 | リソース故障(demote/stop) | 『6.9 リソース故障(demote/stop)』 |
| 6 | リソース故障(vip-master) | 『6.10 リソース故障(vip-master)』 |
| 7 | リソース故障(vip-rep) | 『6.11 リソース故障(vip-rep)』 |
| 8 | リソース故障(vip-slave) | 『6.12 リソース故障(vip-slave)』 |
| 9 | 内蔵ディスク故障 | 『6.13 内蔵ディスク故障』 |
| 10 | ノード故障 | 『6.14 ノード故障』 |
| 11 | IC-LAN故障 | 『6.15 IC-LAN故障』 |

## ルータ故障

この節では、ルータ故障時の対処について説明します。

### 故障時のクラスタ状態

ルータ故障時における、各ノードの状態を表 6‑7に示します。

表 6‑7　ルータ故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.5.2 復旧』 STEP1 |
| 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】故障検知  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】故障検知  【clnDiskd1, clnDiskd2】起動 |
| 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)、VIPcheckリソース(vipCheckMaster)が正常終了が正常終了 | 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)、VIPcheckリソース(vipCheckSlave)が正常終了 |

故障発生時におけるクラスタ状態の例を図 6‑2に示します。



図 6‑2　ルータ故障状況

### 復旧

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

保守者による故障復旧後、pgrex01とpgrex02のPacemakerを再起動し、故障発生前のクラスタ状態に戻します。

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

復旧手順を図 6‑3に示します。

クラスタ状態

復旧手順

表 6‑7の状態

STEP2

Pacemaker停止[pgrex02]

STEP3

Pacemaker停止[pgrex01]

STEP4

保守者へ報告

STEP5

STEP5

STEP7

Pacemaker起動[pgrex01]

STEP8

ノード状態・リソース状態確認[pgrex01]

Pacemaker起動[pgrex02]

ノード状態・リソース状態確認[pgrex02]

保守者による故障復旧

STEP6

リソース状態確認[pgrex01]

STEP1

図 6‑3　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 [pgrex01]

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pgrex01がデータの進んでいるノードの場合を示します。pgrex02がデータの進んでいるノードの場合はpgrex01とpgrex02を読み替えて下さい。

STEP2： Pacemaker停止 [pgrex02]

保守者の作業中に、PG-REXリソースが再起動しないようpgrex02のPacemakerを停止します。停止する手順は『4.7 PostgreSQL停止中の系の停止』を参照してください。

STEP3： Pacemaker停止 [pgrex01]

保守者の作業中に、PG-REXリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『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。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑7を参照
* 報告時点でのクラスタ状態  
  全ノードでPacemaker停止中
* 故障箇所  
  ルータ故障が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01 ]**

OFFLINE: [ pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex01**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex01**

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01 ]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01 ]**

：（略）

STEP7： Pacemaker起動 [pgrex02]

pgrex02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 [pgrex02]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [** pgrex01 **pgrex02 ]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex01 ]

**Slaves: [ pgrex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [** pgrex01 **pgrex02 ]**

：（略）

## S-LAN故障

この節では、S-LAN故障時の対処について説明します。

### 故障時のクラスタ状態

S-LAN故障時における、各ノードの状態を表 6‑8に示します。

表 6‑8　S-LAN故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.6.2 復旧』 STEP1 |
| 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】故障検知  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| 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)、VIPcheckリソース(vipCheckMaster)が正常終了が正常終了 | pgrex01のPG-REXリソース(msPostgresql)の停止を受け、pgrex02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)、VIPcheckリソース(vipCheckMaster)が起動 |

故障発生時におけるクラスタ状態の例を図 6‑4に示します。



図 6‑4　S-LAN故障状況

### 復旧

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

保守者による故障復旧後、pgrex01のPacemakerを再起動します。

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

復旧手順を図 6‑5に示します。

クラスタ状態

復旧手順

表 6‑8の状態

STEP2

Pacemaker停止[pgrex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

リソース状態確認[pgrex02]

ノード状態確認[pgrex02]

STEP4

図 6‑5　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 [pgrex02]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP2： Pacemaker停止 [pgrex01]

保守者の作業中に、PG-REXリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『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 [pgrex02]

ノード状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑8を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所  
  S-LAN故障が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP6： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## D-LAN故障

この節では、D-LAN故障時の対処について説明します。

### 故障時のクラスタ状態

D-LAN故障時における、各ノードの状態を表 6‑9に示します。

表 6‑9　D-LAN故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.7.2 復旧』 STEP1 |
| 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Slave)  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| D-LAN故障を検知し、pgrex01がpgrex02のpgsql-data-statusの値を"DISCONNECT"に変更 | pgrex02のpgsql-data-statusの変更により、master-pgsql、pgsql-statusの値を"HS:alone"に変更  vip-slave、vipCheckSlaveがpgrex01に切り替わる |

故障発生時におけるクラスタ状態の例を図 6‑6に示します。



図 6‑6　D-LAN故障状況

### 復旧

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

保守者による故障復旧後、pgrex02のPacemakerを再起動し、故障発生前のクラスタ状態に戻します。

復旧手順を図 6‑7に示します。

クラスタ状態

復旧手順

表 6‑9の状態

STEP2

Pacemaker停止[pgrex02]

STEP4

保守者へ報告

STEP5

STEP6

Pacemaker起動[pgrex02]

ノード状態・リソース状態確認[pgrex02]

保守者による故障復旧

STEP1

リソース状態確認[pgrex01]

STEP3

ノード状態確認[pgrex01]

図 6‑7　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 [pgrex01]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex01 ]**

**Slaves: [ pgrex02 ]**

：（略）

STEP2： Pacemaker停止 [pgrex02]

pgrex02のPacemakerを停止します。停止する手順は『4.5 Slaveの停止』を参照してください。

STEP3： ノード状態確認 [pgrex01]

ノード状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 ]

**OFFLINE: [ pgrex02 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑9を参照
* 報告時点でのクラスタ状態  
  pgrex02でPacemaker停止中
* 故障箇所  
  D-LAN故障が発生

STEP5： Pacemaker起動 [pgrex02]

pgrex02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 [pgrex02]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [** pgrex01 **pgrex02 ]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex01 ]

**Slaves: [ pgrex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [** pgrex01 **pgrex02 ]**

：（略）

## リソース故障(monitor)

この節では、リソース故障(monitor)時の対処について説明します。

### 故障時のクラスタ状態

リソース故障(monitor)時における、各ノードの状態を表 6‑10に示します。

表 6‑10　リソース故障(monitor)時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.8.2 復旧』 STEP1 |
| 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| 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)、VIPcheckリソース(vipCheckMaster)が正常終了 | pgrex01のPG-REXリソースの停止を受け、pgrex02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)、VIPcheckリソース(vipCheckMaster)が起動 |

故障発生時におけるクラスタ状態の例を図 6‑8に示します。



図 6‑8　リソース故障(monitor)状況

### 復旧

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

保守者による故障復旧後、pgrex01のPacemakerを再起動し、故障発生前のクラスタ状態に戻します。

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

復旧手順を図 6‑9に示します。

クラスタ状態

復旧手順

表 6‑10の状態

STEP2

Pacemaker停止[pgrex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

リソース状態確認[pgrex02]

ノード状態確認[pgrex02]

STEP4

図 6‑9　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 [pgrex02]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP2： Pacemaker停止 [pgrex01]

pgrex01のPacemakerを停止します。停止する手順は『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 [pgrex02]

ノード状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑10を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所  
  pgrex01のリソース故障(monitor)が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP6： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## リソース故障(demote/stop)

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

### 故障時のクラスタ状態

リソース故障(demote/stop)時における、各ノードの状態を表 6‑11に示します。

表 6‑11　リソース故障(demote/stop)時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.9.2 復旧』 STEP1 |
| 【msPostgresql】故障検知 | 【msPostgresql】起動(Slave) |
| PG-REXリソースを停止させたが停止処理に失敗 | IPaddr2リソース(vip-slave)、VIPcheckリソース(vipCheckSlave)が停止 |

故障発生時におけるクラスタ状態の例を図 6‑10に示します。



図 6‑10　リソース故障(stop/demote)状況

### 復旧

復旧手順を図 6‑11に示します。

クラスタ状態

復旧手順

STEP2

STEP3

ノード起動[pgrex01]

STEP6

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

強制電源断[pgrex01]

STEP4

STEP5

表 6‑11の状態

ノード状態・リソース状態確認[pgrex02]

保守者へ報告

図 6‑11　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 [pgrex01]

pgrex01の電源ボタンを押下し、電源を停止します。

STEP2： ノード状態・リソース状態確認 [pgrex02]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP3： ノード起動 [pgrex01]

pgrex01の電源ボタンを押下し、ノードを起動します。

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑11を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所

pgrex01のリソース故障(demote/stop)が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP6： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## リソース故障(vip-master)

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

### 故障時のクラスタ状態

リソース故障(vip-master)時における、各ノードの状態を表 6‑12に示します。

表 6‑12　リソース故障(vip-master)時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.10.2 復旧』 STEP1 |
| 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| リソース故障(vip-master)を検知し、PG-REXリソース(msPostgresql)が正常終了 | pgrex01のPG-REXリソース(msPostgresql)の停止を受け、pgrex02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)、VIPcheckリソース(vipCheckMaster)が起動 |

故障発生時におけるクラスタ状態の例を図 6‑12に示します。



図 6‑12　リソース故障(vip-master)状況

### 復旧

復旧手順を図 6‑13に示します。

クラスタ状態

復旧手順

表 6‑12の状態

STEP2

Pacemaker停止[pgrex01]

STEP3

保守者へ報告

STEP5

STEP6

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

リソース状態確認[pgrex02]

ノード状態確認[pgrex02]

STEP4

図 6‑13　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 [pgrex02]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): Started pgrex02

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP2： Pacemaker停止 [pgrex01]

保守者の作業中に、PG-REXリソースが再起動しないようpgrex01のPacemakerを停止します。停止する手順は『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 [pgrex02]

ノード状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑12を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所  
  IPaddr2(vip-master)故障が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP6： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## リソース故障(vip-rep)

この節では、リソース故障(vip-rep)時の対処について説明します。

### 故障時のクラスタ状態

リソース故障(vip-rep)時における、各ノードの状態を表 6‑13に示します。

表 6‑13　リソース故障(vip-rep)時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.11.2 復旧』 STEP1 |
| 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】停止  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Slave)  【vip-master】停止  【vip-slave】起動  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| 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 |  |

故障発生時におけるクラスタ状態の例を図 6‑14に示します。



図 6‑14　リソース故障(vip-rep)状況

### 復旧

復旧手順を図 6‑15に示します。

クラスタ状態

復旧手順

表 6‑13の状態

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[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

リソース状態確認[pgrex01]

STEP3

図 6‑15　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 [pgrex01]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pgrex01

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex01 ]**

**Slaves: [ pgrex02 ]**

：（略）

Migration Summary:

\* Node pgrex01:

***故障回数表示***

\* Node pgrex02:

STEP2： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑13を参照
* 報告時点でのクラスタ状態  
  両系でPacemaker起動中
* 故障箇所  
  IPaddr2(vip-rep)故障が発生

STEP3： リソース(vip-rep)のフェイルカウントクリア [pgrex01]

pgrex01のvip-repのリソースのフェイルカウントをクリアします。

# crm\_resource -C -r vip-rep -N pgrex01 -f

Cleaning up vip-rep on pgrex01

Waiting for 1 replies from the CRMd. OK

STEP4： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01pgrex02]

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): Started pgrex01

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): Started pgrex01

vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01

：（略）

Migration Summary:

\* Node pgrex01:

***表示無し***

\* Node pgrex02:

## リソース故障(vip-slave)

この節では、リソース故障(vip-slave)時の対処について説明します。

### 故障時のクラスタ状態

リソース故障(vip-slave)時における、各ノードの状態を表 6‑14に示します。

表 6‑14　リソース故障(vip-slave)時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.12.2 復旧』 STEP1 |
| 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Slave)  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| pgrex02のIPaddr2リソース(vip-slave)の停止を受け、IPaddr2リソース(vip-slave)、VIPcheckリソース(vipCheckSlave)がpgrex01に切り替わる | 故障を検知する |

故障発生時におけるクラスタ状態の例を図 6‑16に示します。



図 6‑16　リソース故障(vip-slave)状況

### 復旧

復旧手順を図 6‑17に示します。

クラスタ状態

復旧手順

表 6‑14の状態

STEP2

保守者へ報告

STEP4

リソース(vip-slave)の切り替え[pgrex02]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

リソース状態確認[pgrex01]

STEP3

図 6‑17　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 [pgrex01]

リソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 pgrex02 ]

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex01 ]**

**Slaves: [ pgrex02 ]**

：（略）

Migration Summary:

\* Node pgrex01:

\* Node pgrex02:

***故障回数表示***

**Failed Actions:**

***vip-slave制御エラー情報表示***

STEP2： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑14を参照
* 報告時点でのクラスタ状態  
  両系でPacemaker起動中
* 故障箇所  
  IPaddr2(vip-slave)故障が発生

STEP3： リソース(vip-slave)の切り替え [pgrex02]

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

# crm\_resource -C -r vip-slave -N pgrex02 -f

Cleaning up vip-slave on pgrex02

Waiting for 1 replies from the CRMd. OK

STEP4： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01pgrex02]

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**

Migration Summary:

\* Node pgrex01:

\* Node pgrex02:

***表示無し***

## 内蔵ディスク故障

この節では、内蔵ディスク故障時の対処について説明します。

### 故障時のクラスタ状態

内蔵ディスク故障時における、各ノードの状態を表 6‑15に示します。

表 6‑15　内蔵ディスク故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| Online | Online | 『6.13.2 復旧』 STEP1 |
| 【msPostgresql】停止  【vip-master】停止  【vip-slave】停止  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】故障検知 | 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| 内蔵ディスク故障を検知し、PG-REXリソースを正常に停止 | pgrex01のPG-REXリソース(msPostgresql)の停止を受け、pgrex02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)、VIPcheckリソース(vipCheckMaster)が起動 |

故障発生時におけるクラスタ状態の例を図 6‑18に示します。



図 6‑18　内蔵ディスク故障状況

### 復旧

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

保守者による故障復旧後、pgrex01のPacemakerを再起動し、故障発生前のクラスタ状態に戻します。

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

復旧手順を図 6‑19に示します。

クラスタ状態

復旧手順

STEP2

STEP3

ノード起動[pgrex01]

STEP5

STEP6

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

強制電源断[pgrex01]

STEP4

ノード状態・リソース状態確認[pgrex02]

保守者へ報告

表 6‑15の状態

図 6‑19　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 [pgrex01]

pgrex01の電源ボタンを押下し、電源を停止します。

STEP2： ノード状態・リソース状態確認 [pgrex02]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP3： ノード起動 [pgrex01]

pgrex01の電源ボタンを押下し、ノードの起動を試みます。

STEP4： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑15を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所

pgrex01の内蔵ディスク故障が発生

STEP5： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP6： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## ノード故障

この節では、ノード故障時の対処について説明します。

### 故障時のクラスタ状態

ノード故障時における、各ノードの状態を表 6‑16に示します。

表 6‑16　ノード故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| OFFLINE | Online | 『6.14.2 復旧』 STEP1 |
| Pacemaker停止中のため、表示なし | 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】起動  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| OS故障、電源断、Pacemakerのプロセス故障により、Pacemaker停止状態 | pgrex01のPacemaker停止を受け、pgrex02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)、VIPcheckリソース(vipCheckMaster)が起動 |

故障発生時におけるクラスタ状態の例を図 6‑20に示します。

図 6‑20　ノード故障状況

### 復旧

復旧手順を図 6‑21に示します。

クラスタ状態

復旧手順

STEP2

ノード起動[pgrex01]

STEP5

Pacemaker起動[pgrex01]

ノード状態・リソース状態確認[pgrex01]

保守者による故障復旧

STEP1

STEP3

STEP4

ノード状態・リソース状態確認[pgrex02]

保守者へ報告

表 6‑16の状態

図 6‑21　ノード故障復旧手順

各STEPについて説明します。

STEP1： ノード状態・リソース状態確認 [pgrex02]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex02 ]**

：（略）

STEP2： ノード起動 [pgrex01]

ノードが停止している場合、pgrex01の電源ボタンを押下し、ノードの起動を試みます。

STEP3： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑16を参照
* 報告時点でのクラスタ状態  
  pgrex01でPacemaker停止中
* 故障箇所  
  pgrex01のノードまたはPacemaker故障が発生

STEP4： Pacemaker起動 [pgrex01]

pgrex01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。

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

STEP5： ノード状態・リソース状態確認 [pgrex01]

ノード状態とリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [ pgrex01** pgrex02 **]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex02 ]

**Slaves: [ pgrex01 ]**

Clone Set: clnPing [prmPing]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [ pgrex01** pgrex02 **]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [ pgrex01** pgrex02 **]**

：（略）

## IC-LAN故障

この節では、IC-LAN故障時の対処について説明します。

### 故障時のクラスタ状態

IC-LAN故障時における、各ノードの状態を表 6‑17に示します。

表 6‑17　IC-LAN故障時のクラスタ状態

|  |  |  |
| --- | --- | --- |
| pgrex01 | pgrex02 | 復旧手順遷移先 |
| crm\_monを実行したノードがOnline、もう一方のノードがOFFLINE | | 『6.15.2 復旧』 STEP1 |
| 【msPostgresql】起動(Master)  【vip-master】起動  【vip-slave】停止  【vip-rep】起動  【vipCheckMaster】起動  【vipCheckSlave】停止  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 | 【msPostgresql】起動(Slave)  【vip-master】停止  【vip-slave】起動  【vip-rep】停止  【vipCheckMaster】停止  【vipCheckSlave】起動  【clnPing】起動  【clnDiskd1, clnDiskd2】起動 |
| IC-LAN故障を受け、Slaveを切り離すが、VIPcheckリソース(vipCheckSlave)の起動に失敗 | IC-LAN故障を受け、Masterへ昇格しようとするが、VIPcheckリソース(vipCheckMaster)の起動に失敗 |

故障発生時におけるクラスタ状態の例を図 6‑22に示します。



図 6‑22　IC-LAN故障状況

### 復旧

復旧手順を図 6‑23に示します。

クラスタ状態

復旧手順

STEP2

STEP3

STEP6

STEP7

Pacemaker起動[pgrex02]

ノード状態・リソース状態確認[pgrex02]

保守者による故障復旧

STEP1

STEP4

STEP5

ノード状態・リソース状態確認[pgrex01, pgrex02]

表 6‑17の状態

保守者へ報告

Pacemaker停止[pgrex02]

リソース(vipCheckSlave)のフェイルカウントクリア

ノード状態・リソース状態確認[pgrex01]

図 6‑23　IC-LAN故障復旧手順①

各STEPについて説明します。

STEP1： ノード状態・リソース状態確認 [pgrex01, pgrex02]

pgrex01のリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex01 ]

**OFFLINE: [ pgrex02 ]**

：（略）

vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex01**

Resource Group: master-group

vip-master (ocf::heartbeat:IPaddr2): **Started pgrex01**

vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex01**

：（略）

Master/Slave Set: msPostgresql [pgsql]

**Masters: [ pgrex01 ]**

：（略）

Migration Summary:

\* Node pgrex01:

**故障回数表示**

Failed Actions:

**vipCheckSlave制御エラー情報表示**

pgrex02のリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Online: [ pgrex02 ]

**OFFLINE: [ pgrex01 ]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**

Master/Slave Set: msPostgresql [pgsql]

**Slaves: [ pgrex02 ]**

：（略）

Migration Summary:

\* Node pgrex02:

**故障回数表示**

Failed Actions:

**vipCheckMaster制御エラー情報表示**

STEP2： Pacemaker停止 [pgrex02]

保守者の作業中に、PG-REXリソースが再起動しないようpgrex02のPacemakerを停止します。停止する手順は『4.5 Slaveの停止』を参照してください。

STEP3： リソース(vipCheckSlave)のフェイルカウントクリア [pgrex01]

pgrex01のvipCheckSlaveのフェイルカウントをクリアします。

# crm\_resource -C -r vipCheckSlave -N pgrex01 -f

Cleaning up vipCheckSlave on pgrex01

Waiting for 1 replies from the CRMd. OK

STEP4： ノード状態・リソース状態確認 [pgrex01]

pgrex01のリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**

：（略）

Migration Summary:

\* Node pgrex01:

**表示無し**

STEP5： 保守者へ報告

以下の内容を報告します。

* 復旧開始時点のクラスタ状態  
  クラスタ状態は表 6‑17を参照
* 報告時点でのクラスタ状態  
  pgrex02でPacemaker停止中
* 故障箇所  
  IC-LAN故障が発生

STEP6： Pacemaker起動 [pgrex02]

pgrex02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。

STEP7： ノード状態・リソース状態確認 [pgrex02]

ノードとリソース状態が以下のとおりとなっていることを確認します。

# crm\_mon -fA -1

：（略）

Last updated: *日時表示*

：（略）

**Online: [** pgrex01 **pgrex02 ]**

：（略）

Resource Group: slave-group

vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**

vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**

Master/Slave Set: msPostgresql [pgsql]

Masters: [ pgrex01 ]

**Slaves: [ pgrex02 ]**

Clone Set: clnPing [prmPing]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd1 [prmDiskd1]

**Started: [** pgrex01 **pgrex02 ]**

Clone Set: clnDiskd2 [prmDiskd2]

**Started: [** pgrex01 **pgrex02 ]**

：（略）

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、pgrex01をMasterとして起動します。

1. (Masterとして起動する)pgrex01で、Pacemakerが停止していることを確認します。起動している場合は、『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。

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

1. 稼働中のMasterが存在しないことを確認するために、pgrex02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: [ pgrex02 ]**  ：(略) |

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

|  |
| --- |
| $ pg\_ctl start  **サーバの起動完了を待っています....**  ：(略)  **完了**  **サーバ起動完了**  $ pg\_ctl stop  **サーバ停止処理の完了を待っています.......完了**  **サーバは停止しました。** |

【注意】

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

1. pgrex01で起動禁止フラグのファイルが存在する場合は削除します。本作業はrootユーザで行います。

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

1. 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. pgrex01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。

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

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. pgrex01とpgrex02で、既存のPacemakerの設定を削除します。

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

* + 1. pgrex01でMasterを起動します。

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

* + 1. pgrex01のPacemakerが起動したことを確認します。

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

* + 1. pgrex01でcrmファイルを反映させるコマンドを実行します。本作業はrootユーザで行います。

|  |
| --- |
| # crm options sort-elements no  # crm configure load update PG-REX10\_pm\_crmgen\_env.crm |

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

|  |
| --- |
| WARNING: pgsql: specified timeout 60s for notify is smaller than the advised 90 |

1. pgrex01でcrm\_mon -fA コマンドを実行し、MasterのPacemakerが正常に起動したことを確認します。本作業はrootユーザで行います。

|  |
| --- |
| # crm\_mon -fA  ：(略)  Last updated: *日時表示*  Last change: *日時表示* by root via crm\_attribute on pgrex01  ：(略)  **Online: [ pgrex01 ]**  ：(略)  vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex01**  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): **Started pgrex01**  vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex01**  Resource Group: slave-group  vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**  vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pgrex01 ]**  Clone Set: clnPing [prmPing]  **Started: [ pgrex01 ]**  Clone Set: clnDiskd1 [prmDiskd1]  **Started: [ pgrex01 ]**  Clone Set: clnDiskd2 [prmDiskd2]  **Started: [ pgrex01 ]**  Node Attributes:  **\* Node pgrex01:**  + default\_ping\_set : **100**  + diskcheck\_status : **normal**  + diskcheck\_status\_internal : **normal**  + master-pgsql : **1000**  + pgsql-data-status : **LATEST**  + pgsql-master-baseline : 0000000008000090  + pgsql-status : **PRI**  + ringnumber\_0 : **192.168.1.1 is UP**  + ringnumber\_1 : **192.168.3.1 is UP**  Migration Summary:  \* **Node pgrex01**: |

* 一度もSlaveを起動したことがない場合、pgrex02の情報は表示されません。

1. pgrex01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-10/bin/postgres** -D /dbfp/pgdata/data -c config\_file=/dbfp/pgdata/data/postgresql.conf  **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  **postgres** 3272 2414 0 15:11 ? 00:00:00 **postgres: bgworker: logical replication launcher**  ：(略) |

(MasterのPostgreSQLに接続できることを確認)

|  |
| --- |
| # su - postgres -c "psql -l"  List of databases  Name | Owner | Encoding | Collate | Ctype | Access privileges  -----------+----------+----------+---------+-------+-----------------------  **postgres | postgres | UTF8 | C | C |**  **template0 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/postgres**  **template1 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/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)  ~ pgrex01 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[[25]](#footnote-25)

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、pgrex02をSlaveとして起動します。

1. (Slaveとして起動する)pgrex02で、Pacemakerが停止していることを確認します。起動している場合は、『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。

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

1. 稼働中のMasterが存在することをpgrex01で確認します。確認手順は、『A.2 Masterの起動』の(8)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。

* pgrex02が初回起動時の場合。
* pgrex02にDBクラスタ(または展開されたベースバックアップ)が存在しない場合。
* pgrex02に起動禁止フラグのファイル(/var/lib/pgsql/tmp/PGSQL.lock)が存在する場合。

起動禁止フラグのファイルが存在するということは、前回停止時(異常終了も含む)に系がMasterとして稼働していたことを意味します。このような系をSlaveとして起動する場合(例えば、異常終了した旧Masterの系を、フェイルオーバ後に再組み込みする場合)は、Masterからベースバックアップを取得する必要があります。一方、例えば、Slaveとして稼働していた系を、停止後に再び起動する場合は、起動禁止フラグのファイルが存在せず、ベースバックアップの取得は不要となります。

* pgrex02に存在するDBクラスタが非常に古い場合。

Slaveを長期間停止していたなどで、pgrex02のDBクラスタがpgrex01のDBクラスタと比べて非常に古い場合。この場合、pgrex02のDBクラスタをそのまま使用するとリカバリが大量に必要になります。

* pgrex01で稼働中のMasterにレプリケーション接続ができない場合。

たとえば、システムIDが異なる、タイムライン履歴が一致しないなどの場合。

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

|  |
| --- |
| $ rm -rf $PGDATA  $ rm -rf /dbfp/pgwal/pg\_wal  $ pg\_basebackup -h *192.168.2.3* -U repuser -D $PGDATA -X none -P  **← -hには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. pgrex01からpgrex02へアーカイブディレクトリを同期します。本作業はpostgresユーザで行います。アーカイブディレクトリの同期は必ずMasterからSlaveに行ってください。誤ってSlaveからMasterに同期すると、DBデータの一部が失われる可能性があります。

|  |
| --- |
| $ rsync -av --size-only *192.168.2.3*:/dbfp/pgarch/arc1/ /dbfp/pgarch/arc1/  **← 接続先はMaster側レプリケーション受付用の仮想IPアドレス** |

1. pgrex02で起動禁止フラグのファイルを(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. pgrex02でSlaveを起動します。本作業はrootユーザで行います。

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

1. pgrex02で以下の操作を行い、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  ：(略)  Last updated: *日時表示*  Last change: *日時表示* by root via crm\_attribute on pgrex02  ：(略)  **Online**: [ pgrex01 **pgrex02** ]  ：(略)  vipCheckMaster (ocf::heartbeat:VIPcheck): Started pgrex01  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): Started pgrex01  vip-rep (ocf::heartbeat:IPaddr2): Started pgrex01  Resource Group: slave-group  vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex02**  vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex02**  Master/Slave Set: msPostgresql [pgsql]  Masters: [ pgrex01 ]  **Slaves: [ pgrex02 ]**  Clone Set: clnPing [prmPing]  Started: [ pgrex01 **pgrex02** ]  Clone Set: clnDiskd1 [prmDiskd1]  Started: [ pgrex01 **pgrex02** ]  Node Attributes:  \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 1000  + pgsql-data-status : LATEST  + pgsql-master-baseline : 0000000002000090  + pgsql-status : PRI  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_1 : 192.168.3.1 is UP  **\* Node pgrex02:**  + default\_ping\_set : **100**  + diskcheck\_status : **normal**  + diskcheck\_status\_internal : **normal**  + master-pgsql : **100**  + pgsql-data-status : **STREAMING|SYNC**  + pgsql-status : **HS:sync**  + ringnumber\_0 : **192.168.1.2 is UP**  + ringnumber\_1 : **192.168.3.2 is UP**  Migration Summary:  \* Node pgrex01:  **\* Node pgrex02:** |

1. pgrex02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-10/bin/postgres -D /pgdata/data -c config\_file=/pgdata/data/postgresql.conf**  **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**  **postgres** 22523 22476 0 17:01 ? 00:00:00 **postgres: bgworker: logical replication launcher**  ：(略) |

(SlaveのPostgreSQLに接続できることを確認)

|  |
| --- |
| # su - postgres -c "psql -l"  List of databases  Name | Owner | Encoding | Collate | Ctype | Access privileges  -----------+----------+----------+---------+-------+-----------------------  **postgres | postgres | UTF8 | C | C |**  **template0 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/postgres**  **template1 | postgres | UTF8 | C | C | =c/postgres +**  **| | | | | postgres=CTc/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)  ~ **pgrex02** postgres[2019]: [5-1] ~ LOG: 00000: **started streaming WAL from primary at 0/4000000 on timeline *2***  ：(略) |

1. pgrex01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)  ~ **pgrex01** **postgres**[16409]: [30-1] ~ LOG: 00000: **standby "pgrex02" is now a 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。

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

1. SlaveのPacemakerが正常に停止されたことを確認します。

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

1. SlaveのPostgreSQLが正常に停止されたことを確認します。

|  |
| --- |
| # ps -ef | grep postgres  root 3252 17408 0 17:18 pts/0 00:00:00 grep postgres  **↑PostgreSQLのプロセスが存在しない** |

* 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。

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

1. MasterのPacemakerが正常に停止されたことを確認します。

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

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

|  |
| --- |
| # ps -ef | grep postgres  root 20518 24450 0 17:28 pts/0 00:00:00 grep postgres  **↑PostgreSQLのプロセスが存在しない** |

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、『5.2 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[[26]](#footnote-26)、タイムライン履歴ファイル[[27]](#footnote-27)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

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

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

1. 削除対象の計算

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

1. データベースの復旧に必要のないファイルの削除

両系で、(3)で決定したWALファイル名よりも古いファイルを削除します。

以下のコマンドはpgrex01を想定した例になります。以下の手順をpgrex01で実行後、必ずpgrex02でも同じ手順を実行してください。

ls -lコマンドを用いてアーカイブディレクトリを辞書順にソートして出力します。このとき、(3)で決定したWALファイル名と同じ名前のファイルをチェックします。そのファイルより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。

|  |
| --- |
| [pgrex01] $ 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、『5.3 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pgrex01、Slaveをpgrex02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pgrex02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。

|  |
| --- |
| [pgrex01] # crm\_resource -M -r msPostgresql -N pgrex02 -f -Q |

1. フェイルオーバ確認

pgrex01でcrm\_monコマンドを実行し、以下を確認します。

* pgrex01のmsPostgresqlのリソースの状態が表示されていないこと。
* msPostgresqlリソースが、pgrex02上でMasterとして動作していること。
* vipCheckMasterリソース、vip-masterリソース、vip-repリソースが、pgrex02上で起動していること。

|  |
| --- |
| [pgrex01] # crm\_mon -fA  ：（略）  Last updated: *日時表示*  Last change: *日時表示* by root via crm\_attribute on pgrex01  ：（略）  Online: [ pgrex01 pgrex02 ]  ：(略)  vipCheckMaster (ocf::heartbeat:VIPcheck): **Started pgrex02**  Resource Group: master-group  vip-master (ocf::heartbeat:IPaddr2): **Started pgrex02**  vip-rep (ocf::heartbeat:IPaddr2): **Started pgrex02**  ：（略）  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pgrex02 ]**  ：（略）  Node Attributes:  \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + 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 pgrex01:  \* Node pgrex02: |

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、pgrex02からベースバックアップの取得、pg\_walの構成の修正、pgrex02のアーカイブログとの同期、起動禁止フラグの削除があります。

|  |
| --- |
| [pgrex01] # su - postgres  [pgrex01] $ rm -rf $PGDATA  [pgrex01] $ rm -rf /dbfp/pgwal/pg\_wal  [pgrex01] $ pg\_basebackup -h *192.168.2.3* -U repuser -D $PGDATA -X none -P  **← -hには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**  [pgrex01] $ rsync -av --size-only *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**  [pgrex01] $ exit  [pgrex01] # 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。

|  |
| --- |
| [pgrex01] # crm\_resource -U -r msPostgresql |

1. 系切り替え確認

pgrex01でcrm\_monコマンドを実行し、以下を確認します。

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

|  |
| --- |
| [pgrex01] # crm\_mon -fA  ：（略）  Last updated: *日時表示*  Last change: *日時表示* by root via crm\_attribute on pgrex01  ：（略）  Online: [ pgrex01 pgrex02 ]  ：（略）  Resource Group: slave-group  vipCheckSlave (ocf::heartbeat:VIPcheck): **Started pgrex01**  vip-slave (ocf::heartbeat:IPaddr2): **Started pgrex01**  Master/Slave Set: msPostgresql [pgsql]  **Masters: [ pgrex02 ]**  **Slaves: [ pgrex01 ]**  ：（略）  Node Attributes:  \* Node pgrex01:  + default\_ping\_set : 100  + diskcheck\_status : normal  + diskcheck\_status\_internal : normal  + master-pgsql : 100  + pgsql-data-status : **STREAMING|SYNC**  + pgsql-status : **HS:sync**  + ringnumber\_0 : 192.168.1.1 is UP  + ringnumber\_0 : 192.168.3.1 is UP  ：（略）  Migration Summary:  \* **Node pgrex01:**  \* Node pgrex02:  + ringnumber\_1 : 192.168.3.1 is UP  ：（略）  Migration Summary:  \* **Node pgrex01:**  \* Node pgrex02: |

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-REX10 | PacemakerとPostgreSQL10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-REX10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。 |
| pgrex01  (現用機) | クラスタシステムにおいて、初めから処理やサービスを稼動させているサーバ。 |
| pgrex02  (予備機) | クラスタシステムにおいて、pgrex01（現用機）の故障発生時に処理やサービスを稼動させているサーバ。 |
| スプリットブレイン | pgrex01とpgrex02上の各Pacemakerが互いの状態を把握できなくなり、両系ともMasterになること。主にIC-LANが切断されたときに遷移される状態。 |
| IC-LAN  (インターコネクトLAN) | インターコネクトLANのこと。pgrex01とpgrex02間のネットワーク。pgrex01とpgrex02上の各Pacemakerが互いに死活監視するために使用される。 |
| D-LAN  (DBレプリケーションLAN) | データ転送用LANのこと。pgrex01とpgrex02間のネットワーク。PostgreSQLのレプリケーション機能で使用される。 |
| S-LAN  (運用LAN) | サービス提供用LANのこと。pgrex01またはpgrex02とユーザまたはアプリケーション間のネットワーク。ユーザやアプリケーションがPostgreSQLにアクセスするために使用される。 |
| STONITH-LAN | STONITH機能を使用するためのLANのこと。pgrex01とpgrex02間のネットワーク。各系がスプリットブレイン状態に陥りそうなとき、各系のPacemakerは、このLANを介して、一方の系のマシンを強制的に停止（STONITH機能）し、スプリットブレインが発生するのを回避する。 |
| 同期モード | MasterのWALをSlaveに書き込み後に、Masterがコミットを返すモード。 |
| 非同期モード | トランザクションのコミットのタイミングとは異なり、一定期間でMasterのWALをSlaveに反映していくモード。 |
| Master | RAの状態を表す: PG-REXの場合、同一ノードでPostgreSQLがプライマリサーバとして稼働する。 |
| Slave | RAの状態を表す: PG-REXの場合、同一ノードでPostgreSQLがスタンバイサーバとして稼働する。 |
| Stopped | RAの状態を表す: RAが停止している状態。 |
| start | RAの動作を表す: 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. PostgreSQL関連のツールの対応状況によって日本語が表示されないこともあります。 [↑](#footnote-ref-1)
2. 参照先：<http://www.postgresql.jp/document/10/html/functions-admin.html> [↑](#footnote-ref-2)
3. 参照先：<http://www.postgresql.jp/document/10/html/warm-standby.html#STANDBY-PLANNING> [↑](#footnote-ref-3)
4. 参照先：<http://www.postgresql.jp/document/10/html/hot-standby.html#HOT-STANDBY-CAVEATS> [↑](#footnote-ref-4)
5. 参照先：<http://www.postgresql.jp/document/10/html/hot-standby.html#HOT-STANDBY-CONFLICT> [↑](#footnote-ref-5)
6. その他のインストール方法についてはLinux-HA Japanプロジェクトのサイト(<http://linux-ha.osdn.jp/>)を参照してください。 [↑](#footnote-ref-6)
7. PacemakerリポジトリパッケージはLinux-HA Japanプロジェクトのサイトからダウンロードしてください。 [↑](#footnote-ref-7)
8. 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『3.6.7 リソース（PostgreSQL）の設定』を参照してください。 [↑](#footnote-ref-8)
9. tcp\_keepalives\_idle、tcp\_keepalives\_interval、tcp\_keepalives\_countの設定を指します。 [↑](#footnote-ref-9)
10. アーカイブコマンドの設定は『3.4.5 postgresql.confの編集』を参照してください。 [↑](#footnote-ref-10)
11. /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-11)
12. 参照先: <http://www.postgresql.jp/document/10/html/hot-standby.html#HOT-STANDBY-ADMIN> [↑](#footnote-ref-12)
13. 参照先：<http://www.postgresql.jp/document/10/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-13)
14. 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/10/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-14)
15. 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/10/html/continuous-archiving.html#BACKUP-TIMELINES> [↑](#footnote-ref-15)
16. PostgreSQLがpromoteされる直前のLSNの値を表示する属性です。 [↑](#footnote-ref-16)
17. pgrex01のeth1からの通信状態の場合の例を示します。 [↑](#footnote-ref-17)
18. 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-18)
19. 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-19)
20. crm\_monコマンド実行結果のノード情報表示部にOnlineのノードが2つ表示されている状態です。 [↑](#footnote-ref-20)
21. crm\_monコマンド実行結果のノード情報表示部にOnlineのノードが1つ表示されている状態です。 [↑](#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pgrex02の双方で互いのeth1からの通信状態が"FAULTY"と表示されていることを確認します。 [↑](#footnote-ref-24)
25. 「XXX」には、「No route to host」や「Connection refused」などの文字列が入ります。 [↑](#footnote-ref-25)
26. 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/10/html/continuous-archiving.html#BACKUP-BASE-BACKUP> [↑](#footnote-ref-26)
27. 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/10/html/continuous-archiving.html#BACKUP-TIMELINES> [↑](#footnote-ref-27)