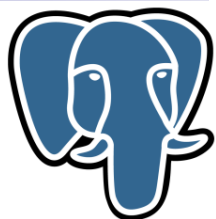


試して覚えるPacemaker入門 『PG-REX構築』

(Pacemaker+PostgreSQLによるシェアードナッシング構成構築)

Linux-HA Japan

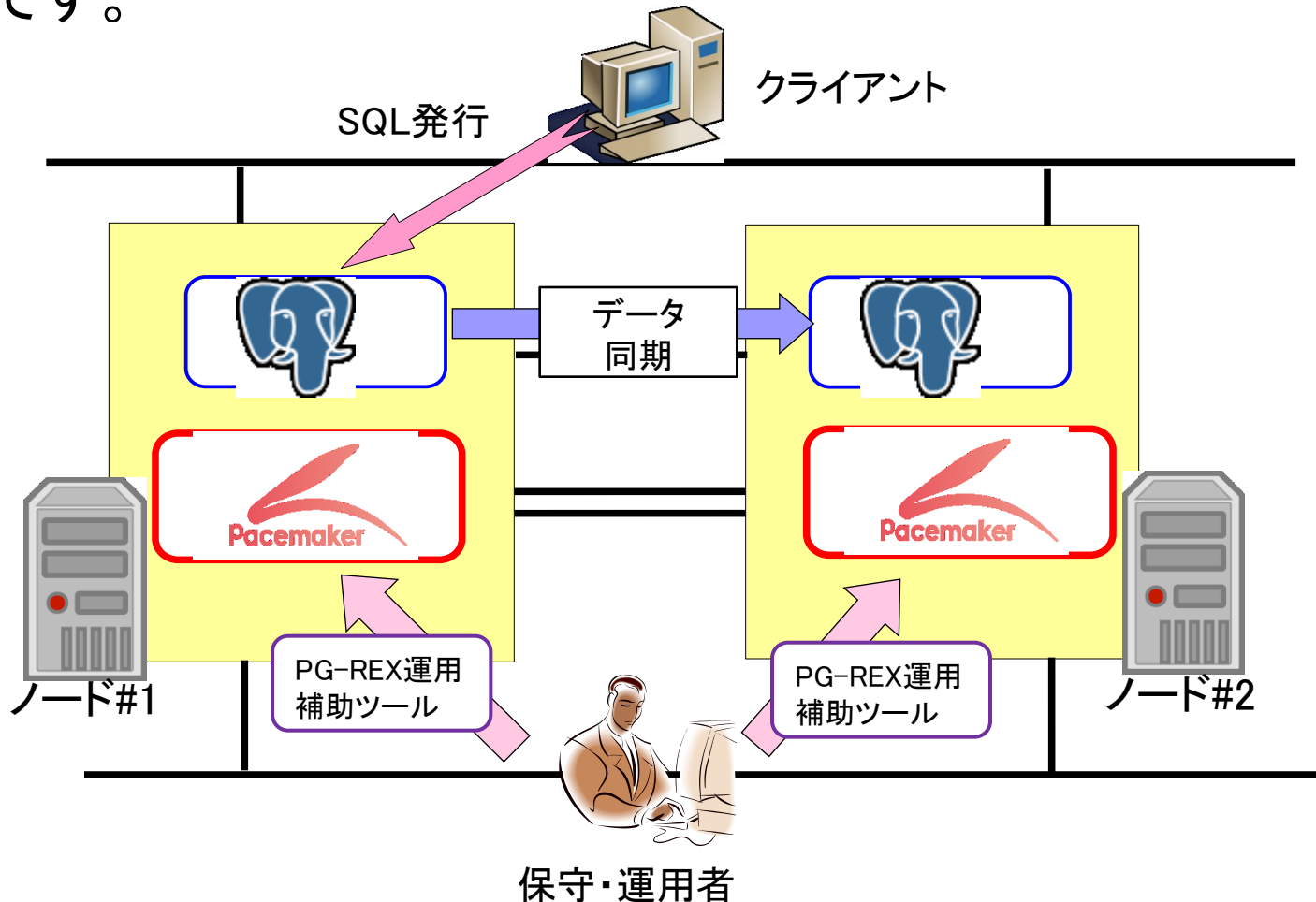


さっそくですが

PG-REXとは

PG-REXとは構成モデルの名称である！

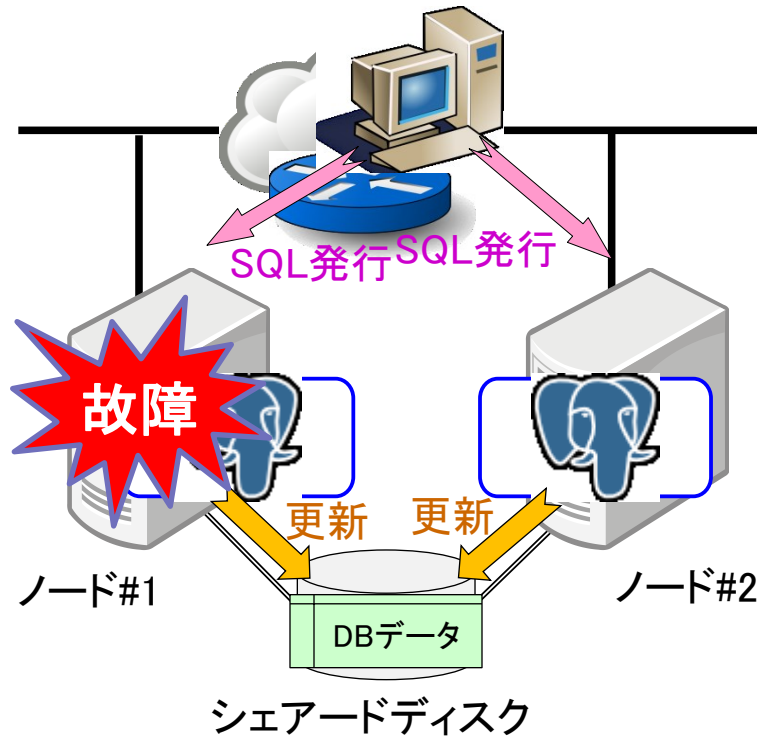
PG-REXとはPostgreSQLとPacemakerとPG-REX運用補助ツールを
組み合わせてシェアドナッシング構成のHAクラスタを実現した構成
モデルです。



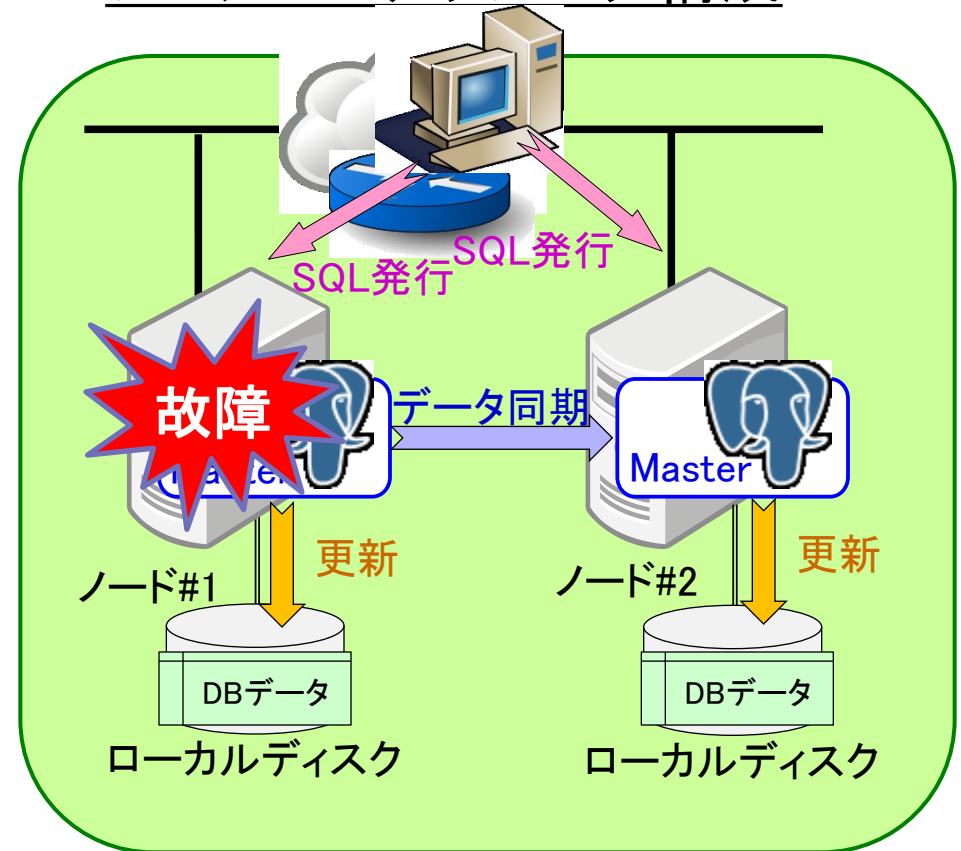
PostgreSQLのHAクラスタ構成

PostgreSQLのHAクラスタを構築する場合、以下の2パターンが考えられます。

シェアードディスク構成



シェアードナッシング構成



PG-REXではこちらの構成を実現します

本セミナーではPG-REXの構築方法を紹介します。

PG-REX構成ではPostgreSQLやPacemakerに特殊な設定が必要で、シェアードディスク構成より構築が難しくなっているため、本セミナーで分かりやすく説明します。

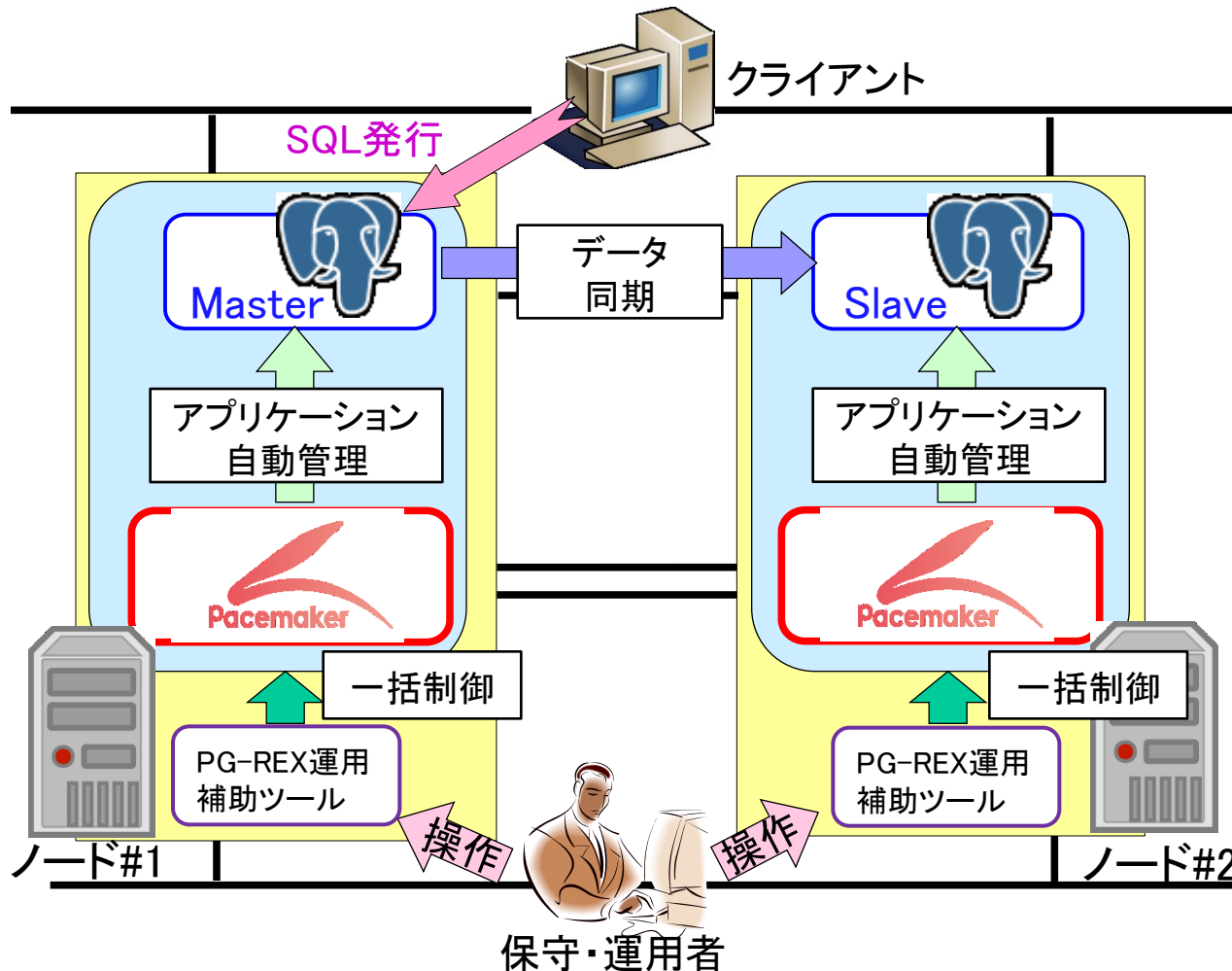
※ シェアードディスク構成の構築方法は以下のセミナー資料で紹介しているため、こちらをご参照ください。

➤ 『試して覚えるPacemaker入門 リソース設定編』

<http://linux-ha.osdn.jp/wp/archives/4532>

PG-REXの概要

PG-REXではPostgreSQLの機能を使用してデータを同期し、PacemakerとPG-REX運用補助ツールにより自動化しています。

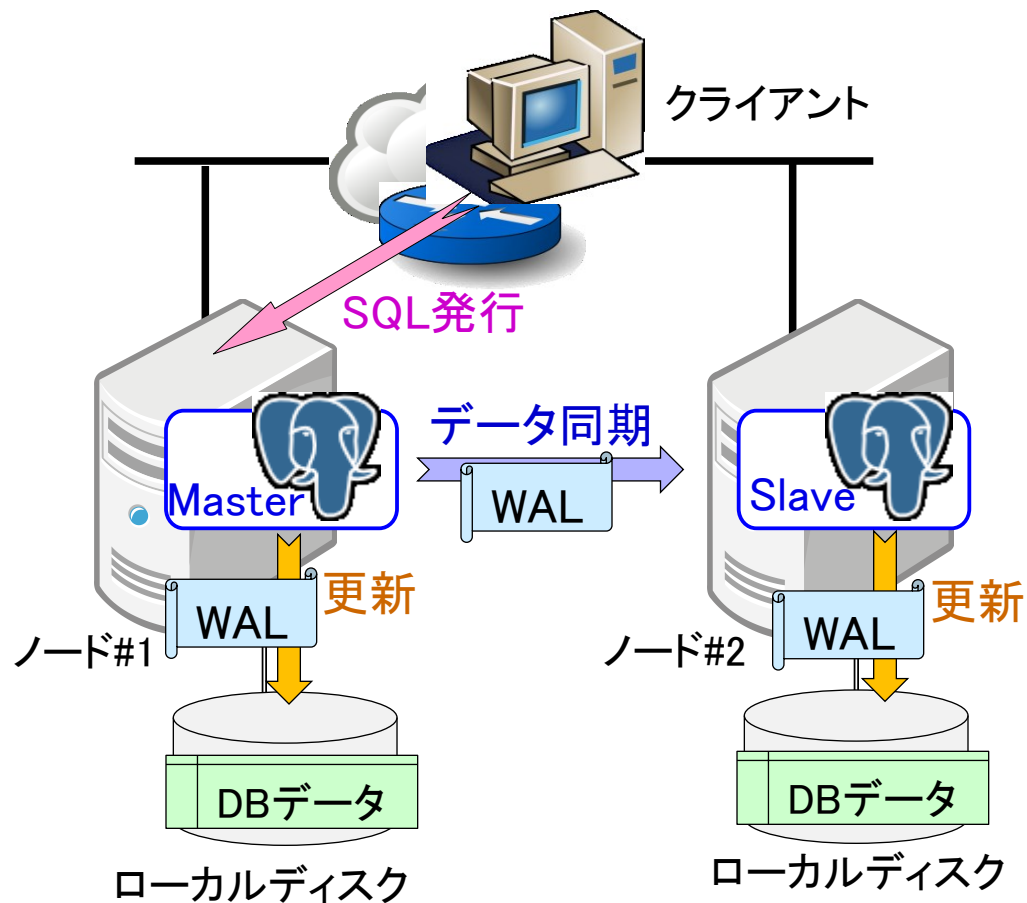


PG-REXではこの3つが組み合わさりシェアードナッシング構成を実現しています。

- ◆ PostgreSQLによる『データ同期』
- ◆ Pacemakerによる『アプリケーション自動管理』
- ◆ PG-REX運用補助ツールによる『一括制御』

PostgreSQLによる『データ同期』

データ同期はPostgreSQLの同期レプリケーション機能によって実現します。

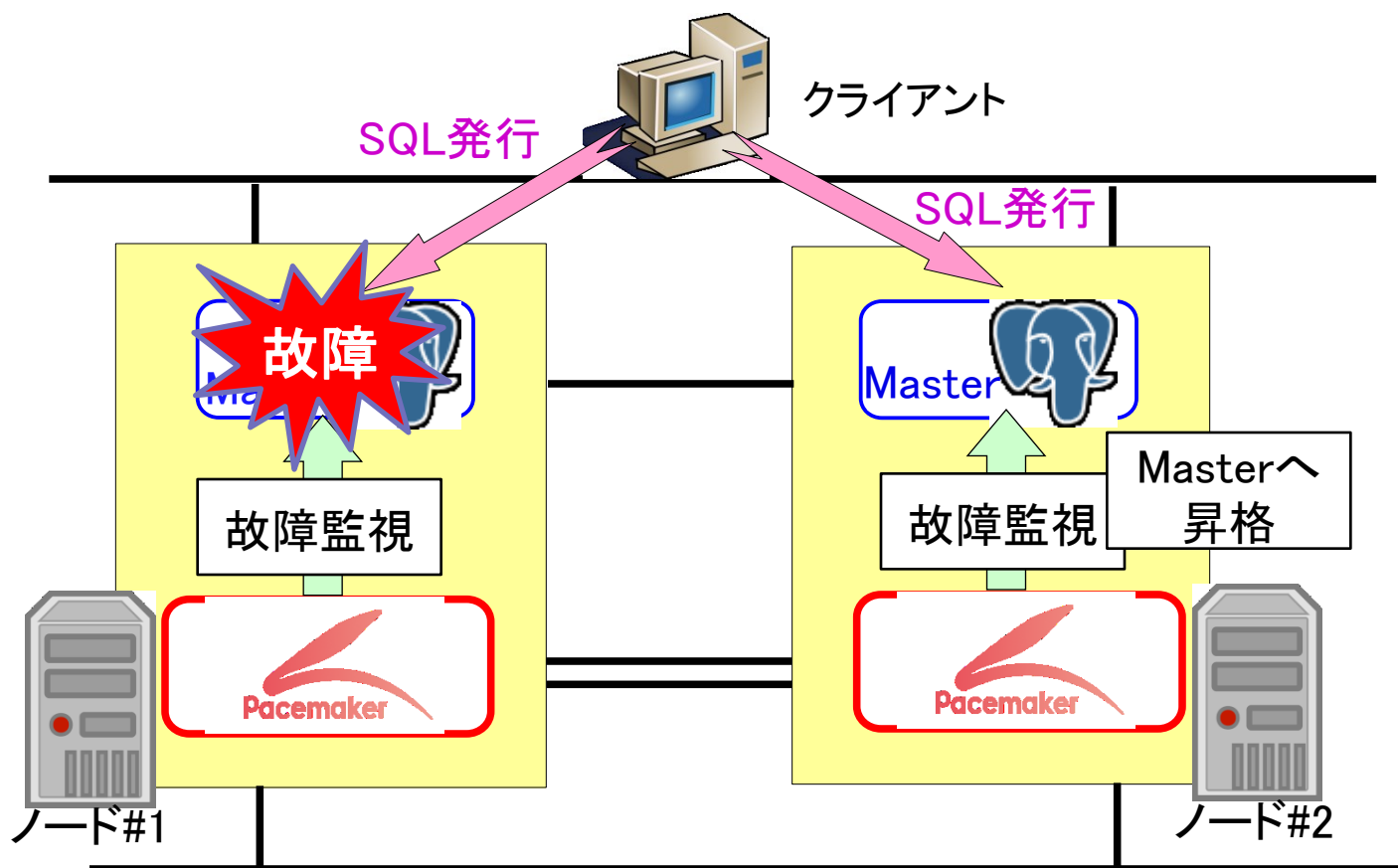


PostgreSQLは更新されたデータをディスクに永続化する際にWALファイルとして書き出します。

同期レプリケーション機能はこのWALファイルを対向ノードへ同期書き込みする機能になります。

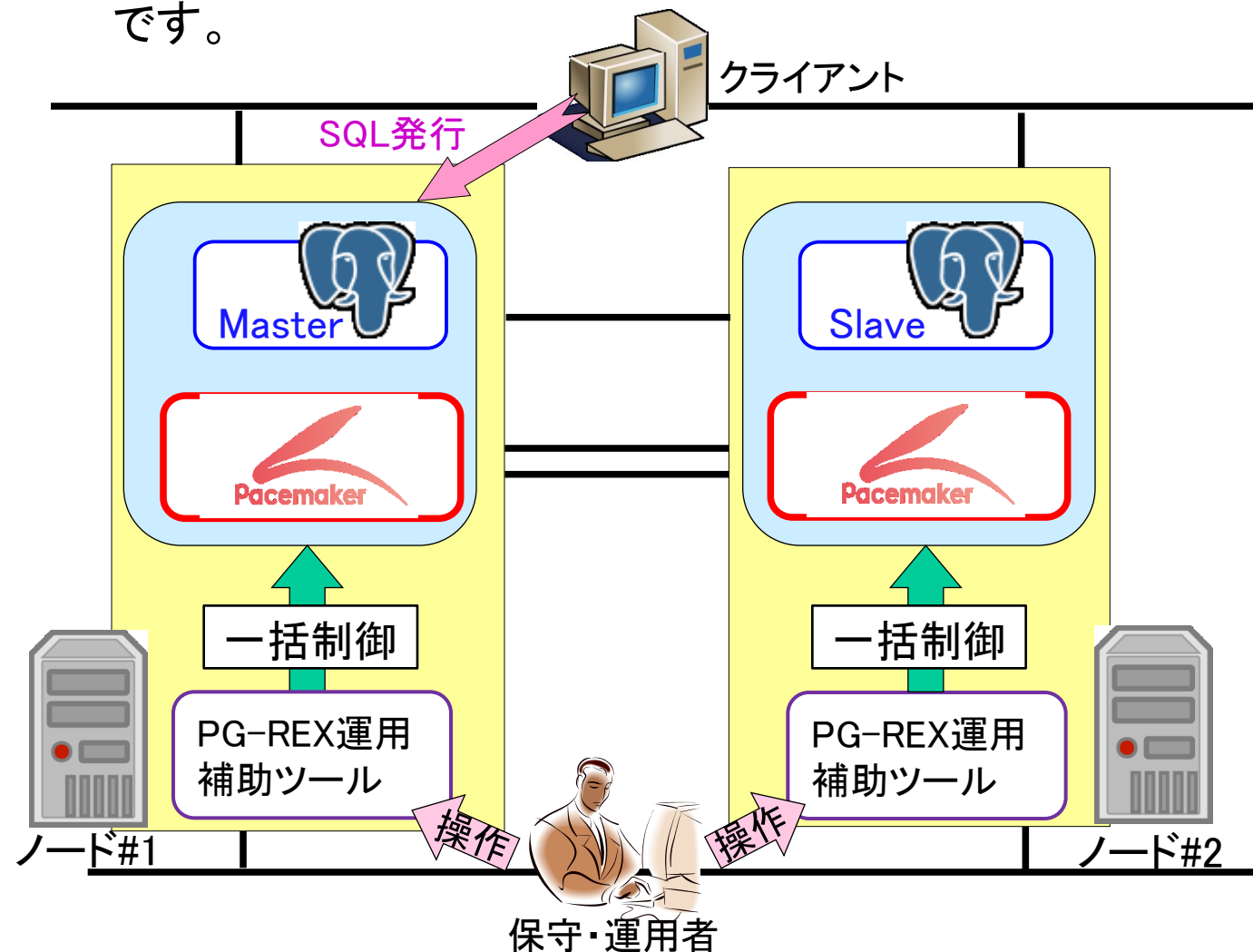
Pacemakerによる『アプリケーション自動管理』

故障発生時にPostgreSQLのMaster/Slave切替(フェイルオーバー)などはPacemakerによって自動管理します。



PG-REX運用補助ツールによる『一括制御』

PG-REX運用補助ツールとはPG-REX構成における保守操作を簡易化するスクリプトです。



PG-REX運用補助ツールはPG-REXの保守時に必要となる操作を簡易に実行できるようにします。

- PG-REX起動
- PG-REX停止
- Master/Slave手動系切替
- PostgreSQLアーカイブログの整理

PacemakerとPostgreSQLのコマンドをラッピングし、操作する上で必要となる複数の手順を一括実行できるようにします。

本セミナーの流れ

以降では以下の順でPG-REX構築手順とその勘所を紹介します。

- **PostgreSQLの同期レプリケーション**
設定方法とPostgreSQL単体でHAクラスタを組む場合に必要手順について紹介
- **Pacemaker、PG-REX運用補助ツール**
設定方法と導入する事で自動化できることについて紹介

設定・構築方法のみをお求めの場合は以下のサイトで配布されているマニュアルをご参照ください。

➤ 『PG-REXプロジェクト』

<https://ja.osdn.net/projects/pg-rex/>

さっそく

PostgreSQLの同期レプリケーション設定 の紹介

PostgreSQLの同期レプリケーション設定の流れ

1. DB初期化

```
$ initdb -D /dbfp/pgdata/data -X /dbfp/pgxlog/pg_xlog --encoding=UTF-8 ¥  
--no-locale --data-checksums
```

2. レプリケーションユーザ作成

```
$ psql -c "CREATE ROLE repuser REPLICATION LOGIN PASSWORD 'repasswd'"
```

3. レプリケーションユーザのアクセス許可設定

```
$ vi $PGDATA/pg_hba.conf
```

4. パスワードファイル設定

```
$ vi ~/.pgpass
```

5. 同期レプリケーション設定

```
$ vi $PGDATA/postgresql.conf
```

※ PostgreSQLのインストール等は両ノードとも完了し、使用可能であることを前提としています。

1. DB初期化

- 各DBファイルを格納するディレクトリを作成します。[ノード1、ノード2]

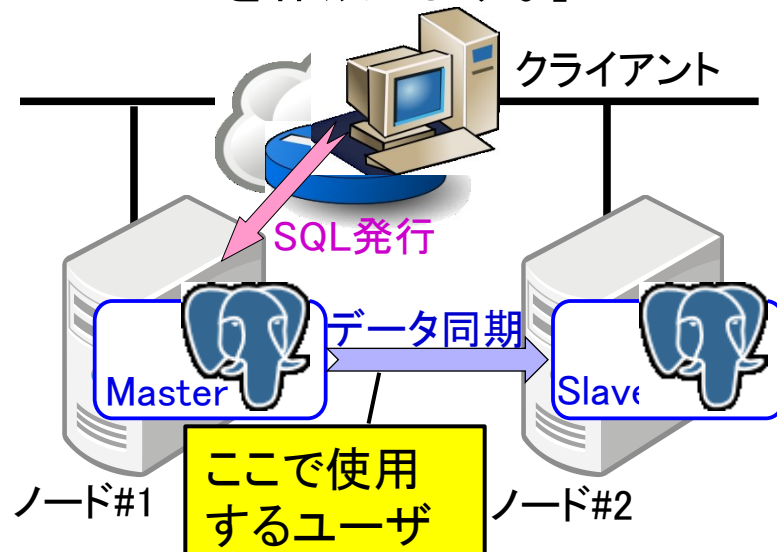
```
# mkdir -p /dbfp/pgdata      ← $PGDATAディレクトリ
# mkdir -p /dbfp/pgxlog      ← WAL格納ディレクトリ
                              (同期レプリケーションではこのWALを転送します。)
# mkdir -p /dbfp/pgarch/arc1 ← アーカイブログディレクトリ
# chown -R postgres:postgres /dbfp
```

- 以下のコマンドでDBを初期化します。[ノード1のみ]
(ノード2は設定完了後、DB起動前にノード1からベースバックアップでコピーします)

```
$ initdb -D /dbfp/pgdata/data -X /dbfp/pgxlog/pg_xlog --encoding=UTF-8 ¥
--no-locale --data-checksums
```

2. レプリケーションユーザ作成

レプリケーションユーザを作成します。[ノード1のみ]



- ❑ ユーザ情報をDBに入力するためPostgreSQLを起動します。

```
$ pg_ctl -w start
```

- ❑ ユーザ情報を入力します。(例: ユーザ名="repuser"、パスワード="reppasswd")

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

- ❑ ユーザ作成後DBを停止します。

```
$ pg_ctl stop
```

3. レプリケーションユーザのアクセス許可設定

- レプリケーションユーザがDBにアクセスすることを許可する設定をします。
[ノード1のみ]

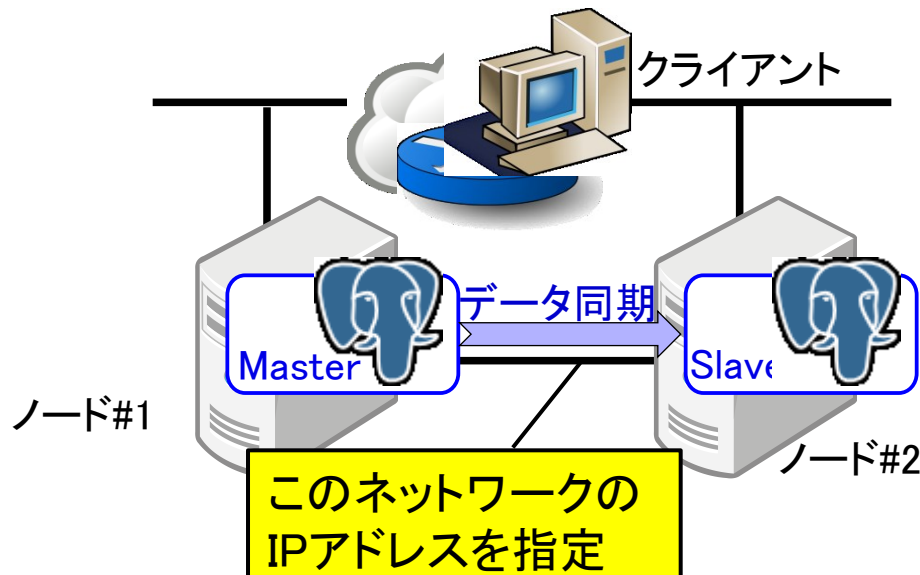
```
$ vi $PGDATA/pg_hba.conf
```

(:省略)

```
host      replication      repuser  192.168.2.1/32      md5
```

```
host      replication      repuser  192.168.2.2/32      md5
```

※ replication データベースに対してレプリケーションユーザの設定をします。
IPアドレスはノード1、ノード2両系の値を設定します。



4. パスワードファイル設定

- ❑ postgresユーザの\$HOME配下にパスワードファイル(.pgpass)を作成し、設定します。[ノード1、ノード2]
(ノード1の設定)

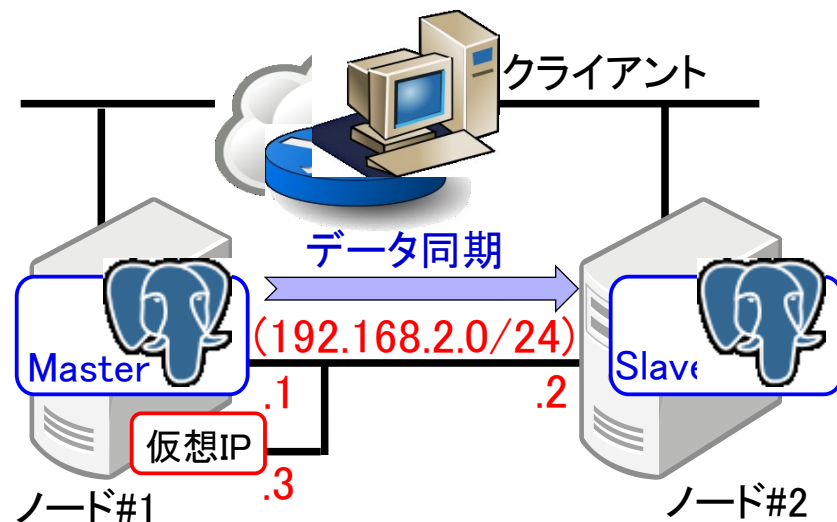
```
$ vi ~/.pgpass
192.168.2.3:5432:replication:repuser:reppasswd
192.168.2.2:5432:replication:repuser:reppasswd

$ chmod 600 ~/.pgpass
```

(ノード2の設定)

```
$ vi ~/.pgpass
192.168.2.3:5432:replication:repuser:reppasswd
192.168.2.1:5432:replication:repuser:reppasswd

$ chmod 600 ~/.pgpass
```



※ Pacemakerで管理する際の処理の都合によりMaster側のノードに仮想IPを割り当てます。
(仮想IPの割当はPacemakerにより行います)

5. 同期レプリケーション設定

□ 同期レプリケーションの設定をします。[ノード1のみ]

```
$ vi $PGDATA/postgresql.conf
```

(:省略)

```
listen_addresses = '*'
```

```
hot_standby = on
```

```
wal_level = hot_standby
```

→ WALに書き込み情報量

```
hot_standby_feedback = on
```

```
max_wal_senders = 4
```

→ WALを送信するためのプロセス数

```
wal_sender_timeout = 20s
```

```
wal_receiver_status_interval = 5s
```

```
wal_keep_segments = 32
```

→ WALを確保しておく量

```
max_standby_streaming_delay = -1
```

```
max_standby_archive_delay = -1
```

```
synchronous_commit = on
```

→ WAL書き込みが完了してからレスポンスを返す

```
archive_mode = on
```

```
archive_command = '/bin/cp %p /dbfp/pgarch/arc1/%f'
```

```
restart_after_crash = off
```

→ クラッシュ時の自動再起動の有効/無効

※ 上記設定は全て必要ですが、ポイントだけここで紹介します。

尚、同期レプリケーションの送信側・受信側の指定は起動時に別途実施します。

PostgreSQL 同期レプリケーション 設定完了！

のハズが……？！

PostgreSQL単体でHAクラスタを組むと思わぬ落とし穴が……！！

PostgreSQL同期レプリケーション 起動停止方法(1/2)

PostgreSQLの同期レプリケーションを有効にした起動方法を紹介します。

HAクラスタでは同期レプリケーションの送信側(Master)と受信側(Slave)は適宜入れ替わることが想定されるため、その都度設定を変更する必要があります。

- ノード1のPostgreSQLを起動します。[ノード1のみ]

```
$ pg_ctl -w start
```

- ノード1からノード2へ\$PGDATAをベースバックアップします。[ノード2のみ]

```
$ pg_basebackup -h 192.168.2.1 -U repuser -D $PGDATA ¥  
--xlogdir=/dbfp/pgxlog/pg_xlog -X stream -P
```

- ノード2をSlaveとして起動するための設定をします。[ノード2のみ]

```
$ vi $PGDATA/recovery.conf  
standby_mode = 'on'  
primary_conninfo = 'user=repuser host=192.168.2.1 port=5432 application_name=node02'  
restore_command = '/bin/cp /dbfp/pgarch/arc1/%f %p'  
recovery_target_timeline = 'latest'
```

PostgreSQL同期レプリケーション 起動停止方法(2/2)

- ノード2をSlaveとして起動します。[ノード2のみ]

```
$ pg_ctl -w start
```

- ノード1からノード2への同期レプリケーションを開始します。[ノード1のみ]

```
$ vi $PGDATA/postgresql.conf  
(:省略)  
synchronous_standby_names = 'node2'  
  
$ pg_ctl reload
```

PostgreSQL停止する際はSlave側から停止する必要があります。

- SlaveのPostgreSQLを停止します。[ノード2のみ]

```
$ pg_ctl stop  
$ rm $PGDATA/recovery.conf  
$ rm -rf $PGDATA/data /dbfp/pgxlog/pg_xlog /dbfp/pgarch/arc1/*
```

- MasterのPostgreSQLを停止します。[ノード1のみ]

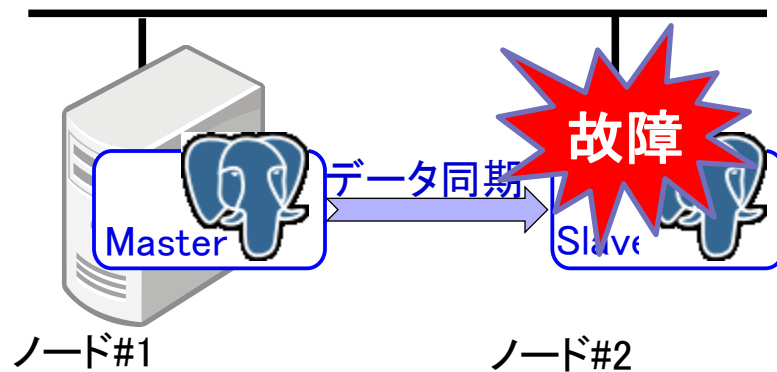
```
$ pg_ctl stop  
$ vi $PGDATA/postgresql.conf  
(:省略)  
# synchronous_standby_names = 'node2'
```

← 設定削除

PostgreSQL同期レプリケーション 故障時の設定変更

同期レプリケーションしている状態でDBやノード等が故障した際は以下の様に設定を変える必要があります。

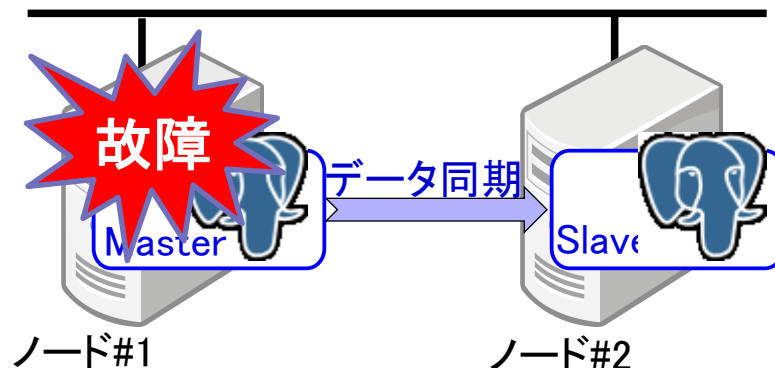
(Slave故障時)



Slaveへの送信設定削除[ノード1のみ]

```
$ vi $PGDATA/postgresql.conf  
(:省略)  
# synchronous_standby_names = 'node2'  
↑ 設定削除  
$ pg_ctl reload
```

(Master故障時)



Masterへ昇格[ノード2のみ]

```
$ pg_ctl promote
```

※ recovery.confは自動的にファイル名変更(recovery.done)され、無視されます。

PostgreSQL同期レプリケーション 故障時の設定変更

同期レプリケーションして
設定を変え

た際は以下の様に設

起動の毎にベースバックアップを行い、
MasterとSlaveはそれぞれ別々の設定を
し、...

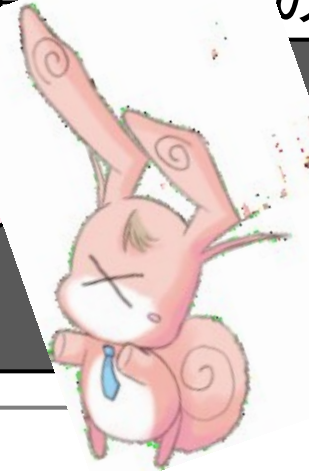
故障時はまた設定を変えるの...?!

Master

ノード#1

設定削除
\$ pg_ctl reload

のみ]



(Ma 人類にはまだ...

早過ぎたんだ!!

昇格[ノード2のみ]

\$ pg_ctl promote

※ recovery.confは自動的にファイル名変
(recovery.done)され、無視されます。

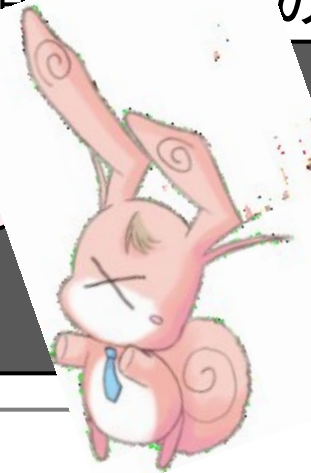
ノード#1

同期レプリケーション
設定を変え

た際は以下の様に設

大丈夫。

また設定を変えるの...?!



私が自動実行
するもの。

設定削除
\$ pg_ctl reload

格[ノード2のみ]

note

onflは自動的にファイル名変
(done)され、無視されます。

このPostgreSQLの管理の難しさ

Pacemakerと
PG-REX運用補助ツールにより

解決します！

PacemakerとPG-REX運用補助ツールを使用した操作(1/4)

□ PacemakerとPG-REX運用補助ツールを使用した場合、以下のコマンドを使用して保守に必要な操作を実行します。

- Master起動

```
# pg-rex_master_start
```

- Slave起動

```
# pg-rex_slave_start
```

- PG-REX停止

```
# pg-rex_stop
```

- Master/Slave手動系切替

```
# pg-rex_switchover
```

- アーカイブログの整理(不要となったアーカイブログを削除)

```
# pg-rex_archivefile_delete
```

PacemakerとPG-REX運用補助ツールを使用した操作(2/4)

PacemakerとPG-REX運用補助ツールを使用した場合、起動手順は以下のようになります。

起動時

ノード1(Master)

1. PostgreSQL起動

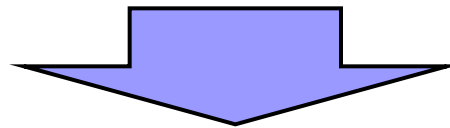
5. synchronous_standby_names設定

ノード2(Slave)

2. ベースバックアップ

3. recovery.conf設定

4. PostgreSQL起動



PacemakerとPG-REXを
導入すると

ノード1(Master)

1. Master起動(# pg-rex_master_start)

ノード2(Slave)

2. Slave起動(# pg-rex_slave_start)

PacemakerとPG-REX運用補助ツールを使用した操作(3/4)

PacemakerとPG-REX運用補助ツールを使用した場合、停止手順は以下のようになります。

停止時

ノード1(Master)

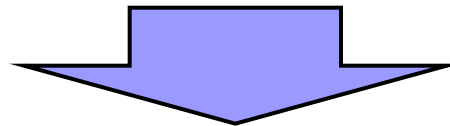
ノード2(Slave)

1. PostgreSQL停止

2. recovery.conf削除

3. PostgreSQL停止

4. synchronous_standby_names削除



PacemakerとPG-REXを
導入すると

ノード1(Master)

ノード2(Slave)

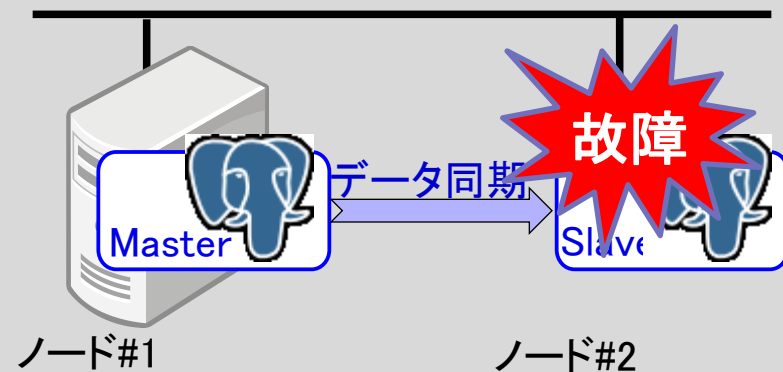
1. PG-REX停止(# pg-rex_stop)

2. PG-REX停止(# pg-rex_stop)

PacemakerとPG-REX運用補助ツールを使用した操作(4/4)

故障時に必要だった以下の手順は全て自動で行うため、操作不要になります。

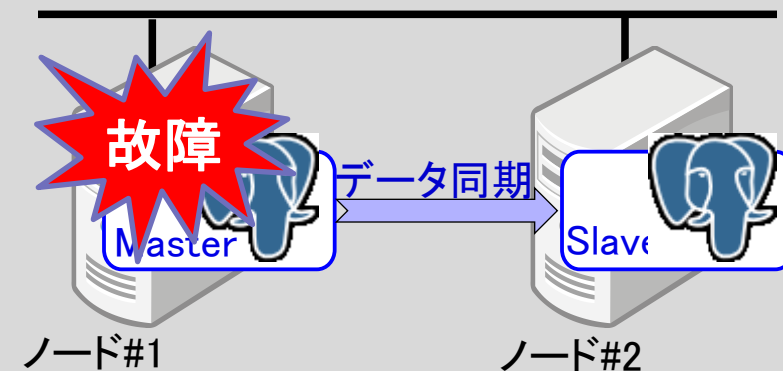
(Slave故障時)



Slaveへの送信設定削除[ノード1のみ]

```
$ vi $PGDATA/postgresql.conf  
(:省略)  
# synchronous_standby_names = 'node2'  
↑ 設定削除  
$ pg_ctl reload
```

(Master故障時)



Masterへ昇格[ノード2のみ]

```
$ pg_ctl promote
```

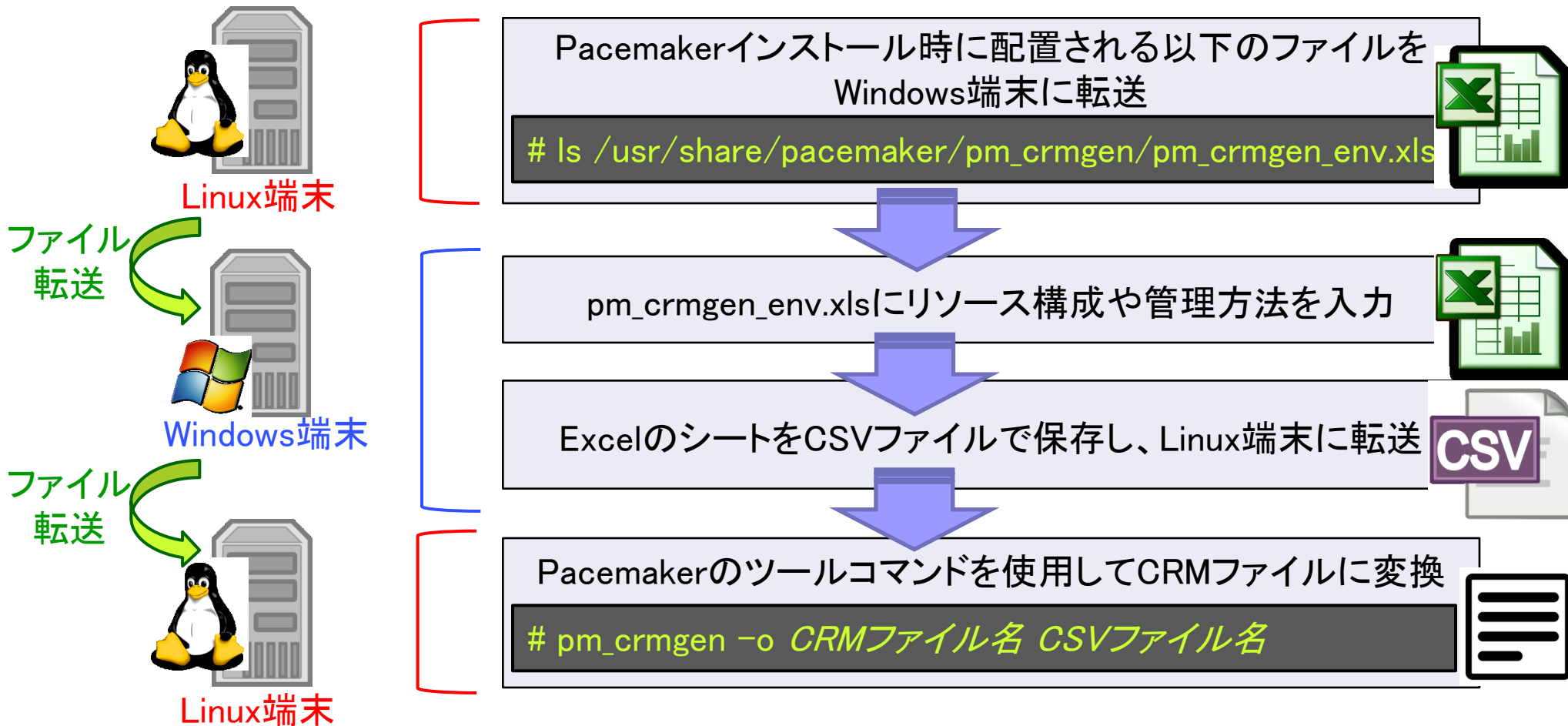
これらを実現するには
アプリケーションの管理方法を
設定する必要があります。

まずは Pacemakerから

- ※ Pacemakerのインストールや基本設定は以下をご参照ください。
- 『試して覚えるPacemaker入門 リソース設定編』
<http://linux-ha.osdn.jp/wp/archives/4532>

Pacemakerのリソース管理設定の流れ

Pacemakerのリソース(アプリケーション)管理は**CRMファイル**と呼ばれるファイルを作成して設定します。CRMファイルは以下の手順で作成できます。



Pacemakerのリソース管理設定

- pm_crmgen_env.xlsは以下の様な表が記述されており、青の枠線の中を埋めて設定します。

| | | | | | | |
|------------------------------------|------------------|-------|---------------------|------|-------|----|
| # pm_crmgen 環境定義書 ファイル形式バージョン: 2.2 | | | | | | |
| #表 1-1 クラスタ設定 ... クラスタ・ノード属性 | | | | | | |
| NODE | | | | | | |
| | uname | ntype | ptype | name | value | |
| # | ノード名 | ノード種別 | パラメータ種別 | 項目 | 設定内容 | 備考 |
| | | | | | | |
| | | | | | | |
| #表 2-1 クラスタ設定 ... クラスタ・プロパティ | | | | | | |
| PROPERTY | | | | | | |
| | name | value | | | | |
| # | 項目 | 設定内容 | 概要 | | | |
| | no-quorum-policy | | ノード数によるリソース割当て | | | |
| | stonith-enabled | | 障害ノード対処 (STONITH制御) | | | |
| | startup-fencing | | 起動時に状態不明ノードへSTONITH | | | |

- 詳細については以下のOSCセミナー資料で説明しているのでご参照ください。

➤ 『試して覚えるPacemaker入門 リソース設定編』

<http://linux-ha.osdn.jp/wp/archives/4532>

PG-REXリソース管理設定

- PG-REXの場合は以下のサイトから設定済みのpm_crmgen_env.xlsが入手できるため、こちらを修正するのが簡単です。

- PG-REXプロジェクト: <https://ja.osdn.net/projects/pg-rer/>

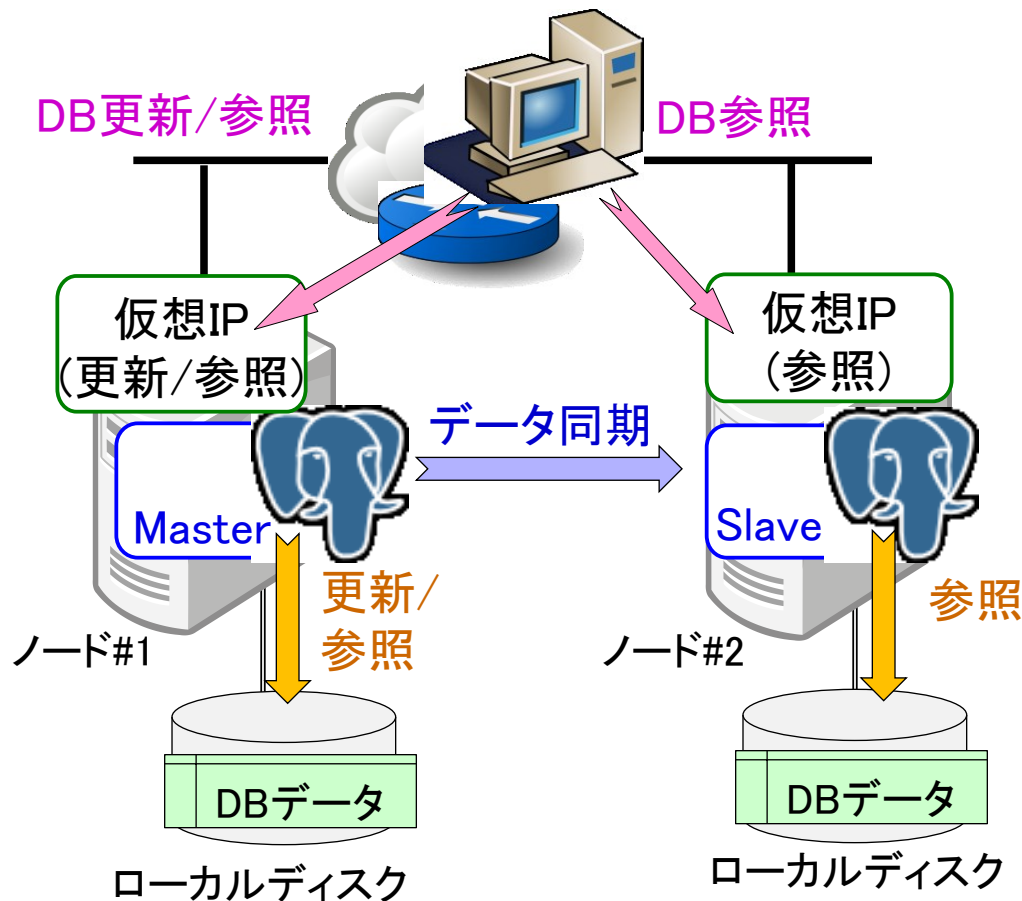
- 以下の**橙色**で塗りつぶされているセルを環境に合わせて修正してください。

| #表 7-1-1 クラスタ設定 ... Primitiveリソース (id=vip-master) | | | | | | |
|---|---|------------|--------------|--------------|-----------------|-------------------------------|
| PRIMITIVE | | | | | | |
| # | P | id | class | provider | type | |
| | | リソースID | class | provider | type | 概要 |
| | | vip-master | ocf | heartbeat | IPAddr2 | PostgreSQL(Master)接続用仮想IP割当 |
| # | A | type | name | value | | |
| | | パラメータ種別 | 項目 | 設定内容 | | 概要 |
| | | params | ip | 192.168.0.10 | | PostgreSQL(Master)接続用仮想IPアドレス |
| | | | nic | bond0 | | 同デバイス名 |
| | | | cidr_netmask | 24 | | 同ネットマスク |
| # | O | type | timeout | interval | on-fail | start-delay |
| | | オペレーション | タイムアウト値 | 監視間隔 | on_fail(障害時の動作) | 起動前待機時間 |
| | | start | 60s | 0s | restart | |
| | | monitor | 60s | 10s | restart | |
| | | stop | 60s | 0s | block | |

※ pm_crmgen_env.xlsはSTONITH構成とNoSTONITH構成のものが用意されています。運用環境ではSTONITH構成を推奨しますが、動作確認用であればNoSTONITH構成が環境の制約もなく構築しやすいです。

PostgreSQLのSlaveへのデータベース接続

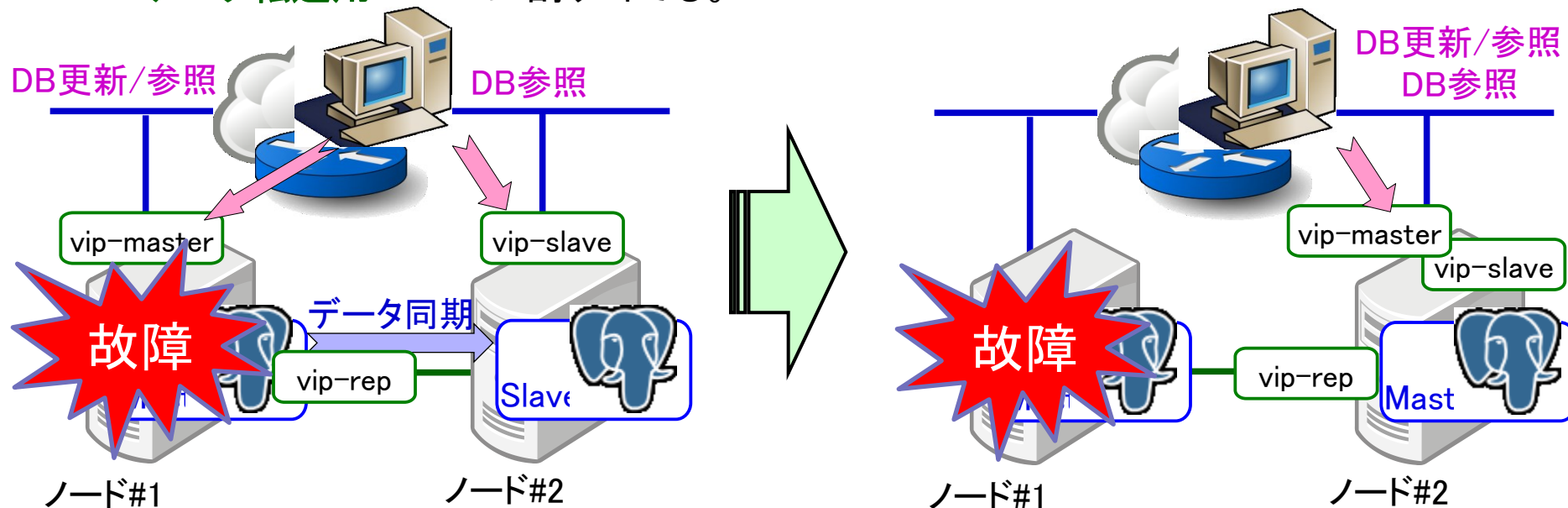
- PostgreSQLはSlaveでもDBを参照することが可能です。
 - PG-REX用設定済みpm_crmgen_env.xlsではDB更新/参照用の仮想IPとは別にDB参照専用の仮想IPを割り当てて、上記機能を活用できるようにしています。



PG-REXにおける仮想IPの管理

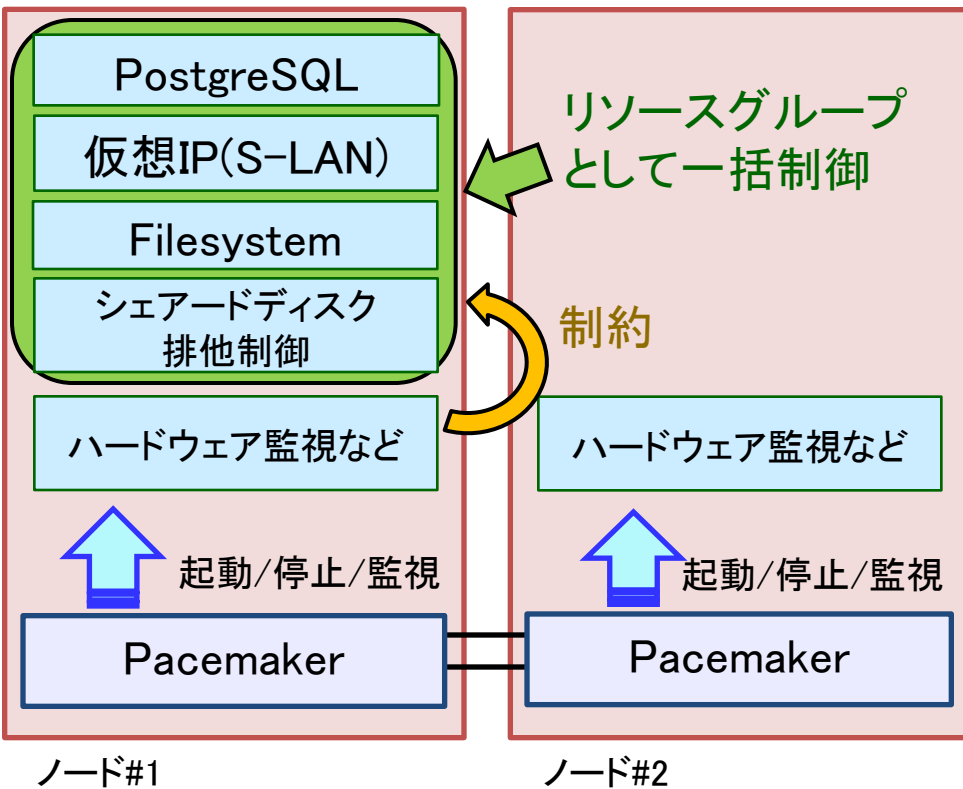
□ PG-REXにてPacemakerが割り当てる仮想IPは合計で以下の3つであり、それぞれvip-master、vip-slave、vip-repと名称しています。

- vip-master:
DBを更新/参照する際にアクセスするための仮想IP。**サービスLAN**上に割り当てる。
- vip-slave:
DBを参照する際にアクセスするための仮想IP。**サービスLAN**上に割り当てる。
- vip-rep:
データレプリケーションする際にMasterにアクセスするための仮想IP。
データ転送用LAN上に割り当てる。



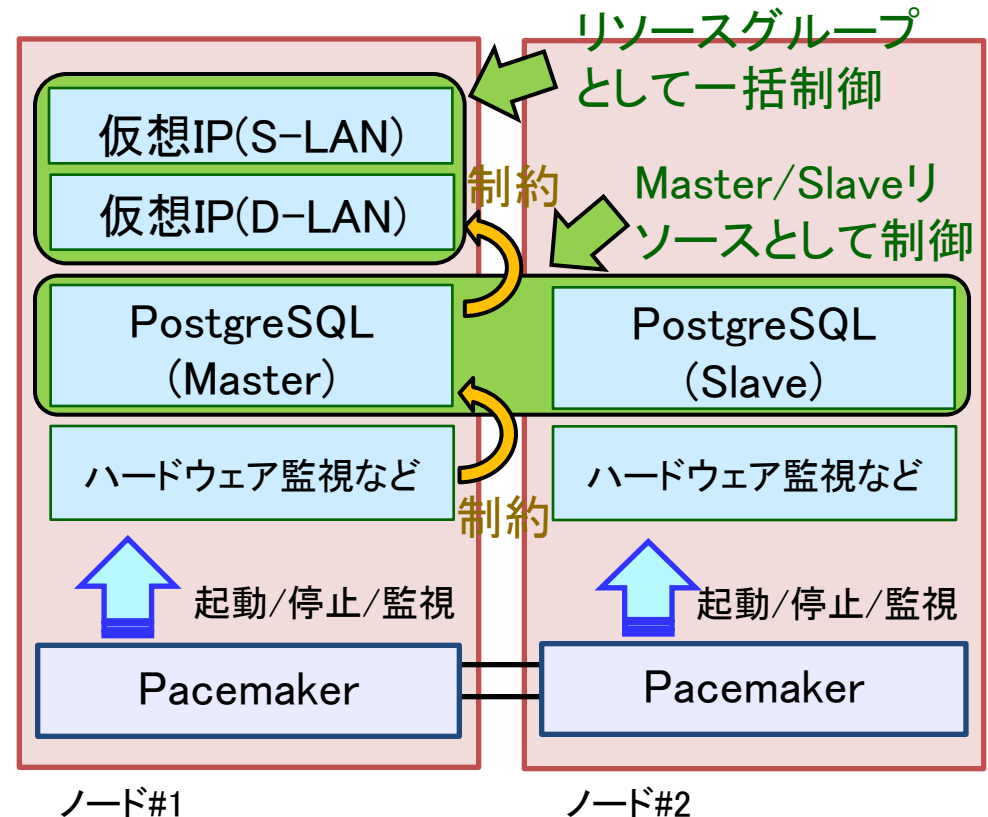
(参考) PG-REXでPacemakerが管理するリソース構成

シェアードディスク構成



※『試して覚えるPacemaker入門 リソース設定編』ではこの構成を構築しています。

PG-REX (シェアードナッシング構成)



※ 上記では以下の様な略称で記述しています。
サービスLAN ⇒ S-LAN
データ転送用LAN ⇒ D-LAN

続いて

PG-REX運用補助ツール

PG-REX運用補助ツールのインストール・設定(1/2)

□ PG-REX運用補助ツールも先ほどのpm_crmgen_env.xlsと同様に以下から入手できます。

- PG-REXプロジェクト: <https://ja.osdn.net/projects/pg-rex/>
- 必要となるパッケージ(バージョンは入手時の最新のものを選択してください)
 - ・ Net_OpenSSH-*.rpm
 - ・ IO_Tty-*.rpm
 - ・ pg-rex_operation_tools_script-*.rpm

□ ダウンロードしたパッケージを以下の順にインストールします。[ノード1、ノード2]

```
# rpm -ivh Net_OpenSSH-*.rpm
# rpm -ivh IO_Tty-*.rpm
# rpm -ivh pg-rex_operation_tools_script-*.rpm
```

PG-REX運用補助ツールのインストール・設定(2/2)

□ インストールすると以下のファイルが配置されます。

/usr/local/bin

└ pg-rex_master_start

← MasterノードとしてPacemakerを起動するコマンド

└ pg-rex_slave_start

← SlaveノードとしてPacemakerを起動するコマンド

└ pg-rex_stop

← 自ノードのPacemakerを停止するコマンド

└ pg-rex_switchover

← MasterとSlaveを系切替するコマンド

└ pg-rex_archivefile_delete

← 不要なアーカイブログを選定して削除するコマンド

/etc

└ pg-rex_tools.conf

← PG-REX運用補助ツールの設定ファイル

(その他、/usr/local/share/perl5/PGRex等にcommonファイル等が配置されます。)

□ PG-REX運用補助ツールの設定ファイルを修正します。[ノード1、ノード2]

```
# vi /etc/pg-rex_tools.conf
```

```
D-LAN_IPAddress = 192.168.2.1 , 192.168.2.2
```

← 両系のデータ転送用LANのIP

```
Archive_dir = /dbfp/pgarch/arc1
```

← アーカイブログディレクトリ

```
STONITH = disable
```

← STONITHの使用有無

```
VIP_SLAVE = enable
```

← 仮想IP vip-slaveの使用有無

```
(:省略)
```



PG-REX完成！！



PG-REXの動作確認方法(1/3)

PG-REXはMasterを先に起動し、起動完了後にSlaveを起動します。

□ Masterを起動します。[ノード1のみ]

```
# rm /var/lib/pgsql/tmp/PGSQL.lock
↑ 前起動時に正常停止できなかった場合に残存される起動禁止フラグです。
# pg-rex_master_start PG-REX9.6_pm_crmgen_env.crm
```

pm_crmgen_env.xlsから生成したCRMファイル名を指定します。

□ 上記Master起動完了を確認後、Slaveを起動します。[ノード2のみ]

```
# rm /var/lib/pgsql/tmp/PGSQL.lock
# pg-rex_slave_start
```

PG-REXの動作確認方法(2/3)

- 動作確認用のDBとユーザ、テーブルを作成します。[ノード1のみ]

```
$ createdb testdb
$ createuser user1
$ psql -d testdb -U user1 -c "CREATE TABLE bar (id int, name varchar)"
$ psql -d testdb -U user1 -c "INSERT INTO bar VALUES (1, 'apple')"
$ psql -d testdb -U user1 -c "SELECT * FROM bar"
 id | name
----+-----
  1 | apple
```

- ノード2で入力したテーブルが参照できることを確認します。[ノード2のみ]

```
$ psql -d testdb -U user1 -c "SELECT * FROM bar"
 id | name
----+-----
  1 | apple
```

PG-REXの動作確認方法(3/3)

停止時は以下のコマンドを使用してSlaveから順に停止します。

- ❑ Slaveを停止します。[ノード2]

```
# pg-rex_stop
```

- ❑ Masterを停止します。[ノード1]

```
# pg-rex_stop
```

Linux-HA Japanのブースにて
PG-REXのデモ環境を用意しています。

是非お越しください！！

さいごに(宣伝1)

Linux-HA Japan URL

<http://linux-ha.osdn.jp/>

<https://ja.osdn.net/projects/linux-ha/>



The screenshot shows the Linux-HA Japan website. At the top is the logo and title "LINUX-HA JAPAN High-Availability Clustering on Linux". Below this is a navigation bar with links: HOME, メーリングリスト, ダウンロード&インストール, マニュアル, デスクトップテーマ・壁紙等, コミュニティ概要. A secondary bar contains: その他, ニュース, イベント情報, 読み物, WEBラジオ. The main content area is titled "Linux-HA Japan プロジェクト" and includes social media links (Facebook, Twitter, 125, Check). The text describes the project as a Linux-based high-availability (HA) cluster system, mentioning its components like Pacemaker, Heartbeat, Corosync, and DRBD. It also lists resources such as manuals, mailing lists, event information, and a developer site. At the bottom, there is a Twitter link and contact information for the mailing list manager.

Pacemaker関連の最新情報を
日本語で発信

Pacemakerのダウンロードもこ
ちらからどうぞ
(インストールが楽なリポジトリパッケージ
を公開しています)

さいごに(宣伝2)

日本におけるHAクラスタについての活発な意見交換の場として「Linux-HA Japan 日本語メーリングリスト」も開設しています。

Linux-HA-Japan MLでは、Pacemaker、Heartbeat3、Corosync DRBDなど、HAクラスタに関連する話題は歓迎！

- ・ ML登録用URL

<http://linux-ha.osdn.jp/>
の「メーリングリスト」をクリック

- ・ MLアドレス

linux-ha-japan@lists.osdn.me

※スパム防止のために、登録者以外の投稿は許可制です



PG-REX URL

<https://ja.osdn.net/projects/pg-rex/>

PG-REX Fork

概要 ▾ ダウンロード ソースコード ▾ チケット ▾ 文書 ▾ コミュニケーション ▾ ニュース

KDE Plasma 5を搭載した「Mageia 6」が登場 [Magazine]

プロジェクトの説明





PostgreSQLとPacemakerを組み合わせた高可用ソリューション


 画像一覧

ダウンロード

 Windows [pg-rex96-1.0.0-1.tar.gz](#) (日付: 2017-01-25, サイズ: 1.46 MB)

 Mac [pg-rex96-1.0.0-1.tar.gz](#) (日付: 2017-01-25, サイズ: 1.46 MB)

 Linux [pg-rex96-1.0.0-1.tar.gz](#) (日付: 2017-01-25, サイズ: 1.46 MB)

 UNIX [pg-rex96-1.0.0-1.tar.gz](#) (日付: 2017-01-25, サイズ: 1.46 MB)

最新リリース

[PG-REX9.6 1.0.2_CentOS7](#) (日付: 2017-07-21)

[pg-rex_operation_tools 1.8.1](#) (日付: 2017-07-21)

[pg-rex_operation_tools 1.8.0](#) (日付: 2017-01-25)

[PG-REX9.6 1.0.0](#) (日付: 2017-01-25)

[PG-REX9.5 1.1.1](#) (日付: 2016-09-01)

PG-REXの構築手順書や
pm_crmgen_env.xls、PG-REX運
用補助ツールを提供しています。

本セミナーでは説明できなかった
詳細内容も上記手順書に詳しく
書いてあるので是非読んでくださ
い！

ご清聴、ありがとうございました。