※DRBDはLINBIT Information Technologies GmbHの登録商標です。

【入門】PostgreSQL＋Pacemaker＋DRBDで高可用性構成を構築してみよう

オープンソースカンファレンス2014 Nagoya
2014年7月4日(金)

SRA OSS, Inc. 日本支社
松坂 大地
matsusaka@sraoss.co.jp

はじめに

- どのような人に聞いてほしいか
  - PostgreSQLを使っている人
  - PostgreSQLの可用性を上げたいと思っている人
  - 可用性構成は難しそうだと思っている人

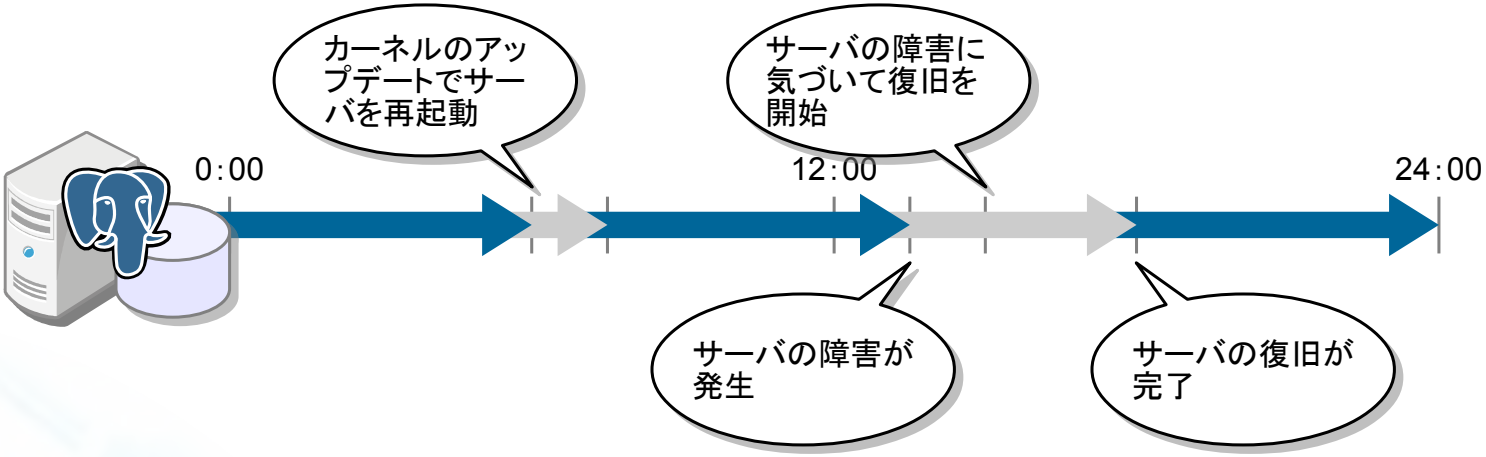
- PostgreSQLの高可用性構成の構築手順を説明
  - インストールについては概要だけにしておく
  - とりあえず＝細かいことは置いておく
    スプリットブレイン、STONITHなど
  - Pacemaker、HeartbeatについてはLinux-HA Japan Project、DRBDについてはサードウェアにも話を聞こう
    - PostgreSQLについてはSRA OSSも忘れないで

45分ということで

- リソース設定と動作デモ
  - PostgreSQL+DRBD+Pacemaker構成の必要最低限の設定
  - フェイルオーバなどの動作を確認

高可用性とは

- システムをいかに止めずに動かし続けられるかの度合い＝可用性が高いこと
  - HA＝High Availabilityとも言う
- 可用性は稼働率や停止時間で計られる



  - 稼働率は➡÷(➡＋➡)＝18÷(18＋6)＝75%
  - 停止時間は➡＝6時間／日≒91.25日／年
    - 日単位の停止時間を年単位に換算するのには無理があるが

SRA OSS, INC.

高可用性構成とは？（9）

- PostgreSQL高可用性の実現方法めやす

稼働率	年間停止時間	実現方法
90%	36.5 日	バックアップ～リストアだけで十分。オンラインバックアップ取得を実施。
99%	3.65 日	オンプレミスなら予備マシンが必要。大データならバックアップのリストア所要時間を把握しておく。
99.9%	8.7 時間	保守停電のないクラウド～ハウジングが必要。平日日中のみ障害検知だとむずかしい。
99.99%	52 分	バックアップのリストアがほぼ不可能。レプリケーション（データ同期）された待機サーバが必要。
99.999%	5 分	HAクラスタソフトウェア等が必要。5分は自動切り替え1～2回分。

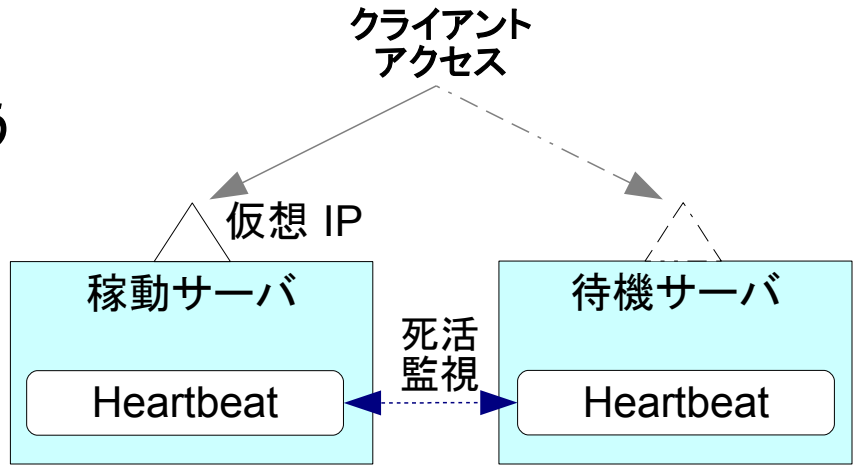
PostgreSQL高可用性構成の選択肢とトレンド
http://www.sraoss.co.jp/event_seminar/2013/20130212_pgha_seminar_sraoss.pdf

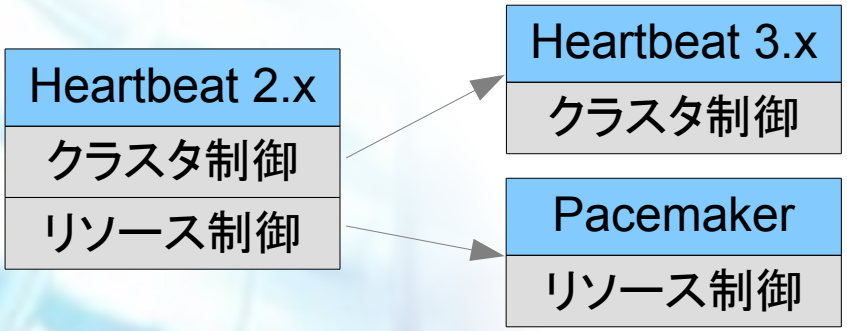
PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

Pacemakerとは

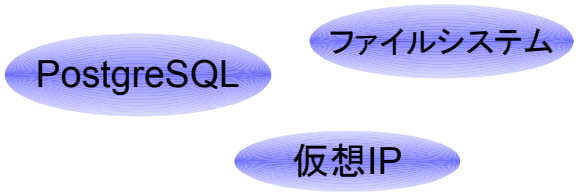
Pacemaker誕生

- Heatbeatのリソース制御部分が独立
  - Heartbeat
    HAクラスタにおいて連結した
    複数サーバの死活監視を行う
    ソフトウェア
  - クラスタ制御
    HAクラスタのノードを管理
    「ハートビート」と呼ばれる
    パケットを送信し死活監視



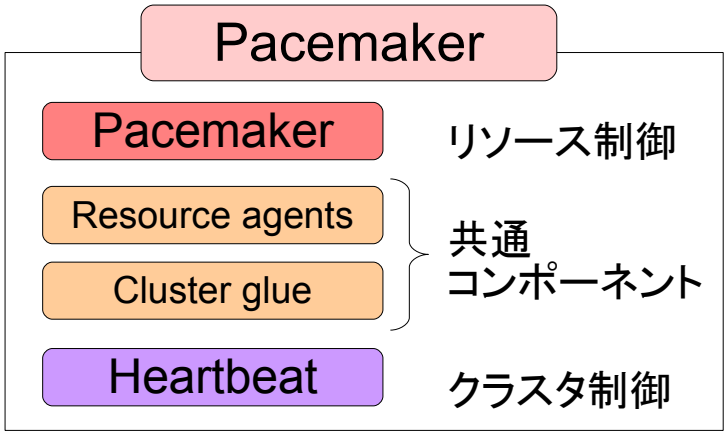


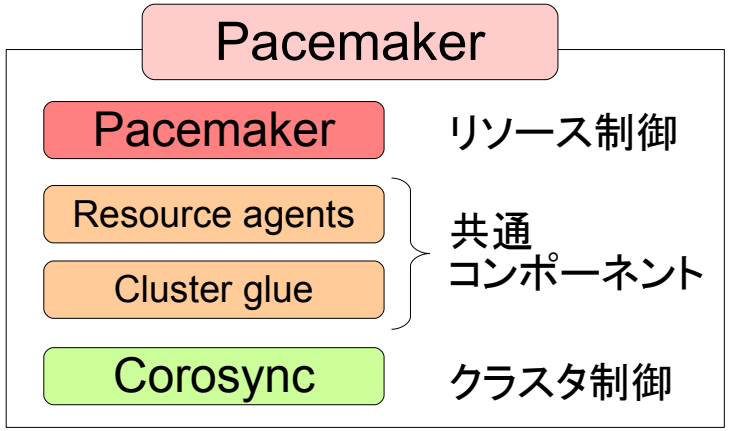
  - リソース制御
    Activeなノードにて定義された
    リソースを有効化



Pacemakerの構造

- 広義の意味でのPacemaker
  - Pacemaker と Heartbeat3 を使った構成をPacemakerと呼ぶ





  - Heartbeat3の代わりにCorosyncを使う構成もある
    - OpenAISのクラスタ制御部分

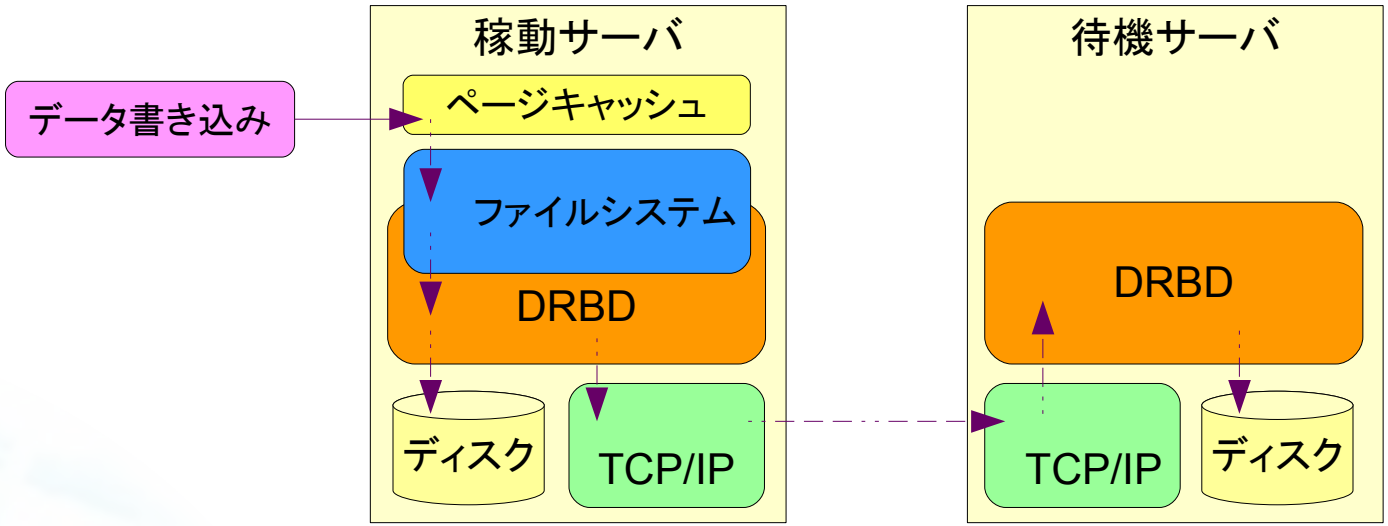
⇒ 本日はHeatbeat3でのお話

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

DRBDとは

DRBDとは

- DRBDとは
  - ネットワーク越しにHDDのミラーリングを行うソフトウェア
  - ブロックデバイス（/dev/sda1など）の変更内容をミラーリング



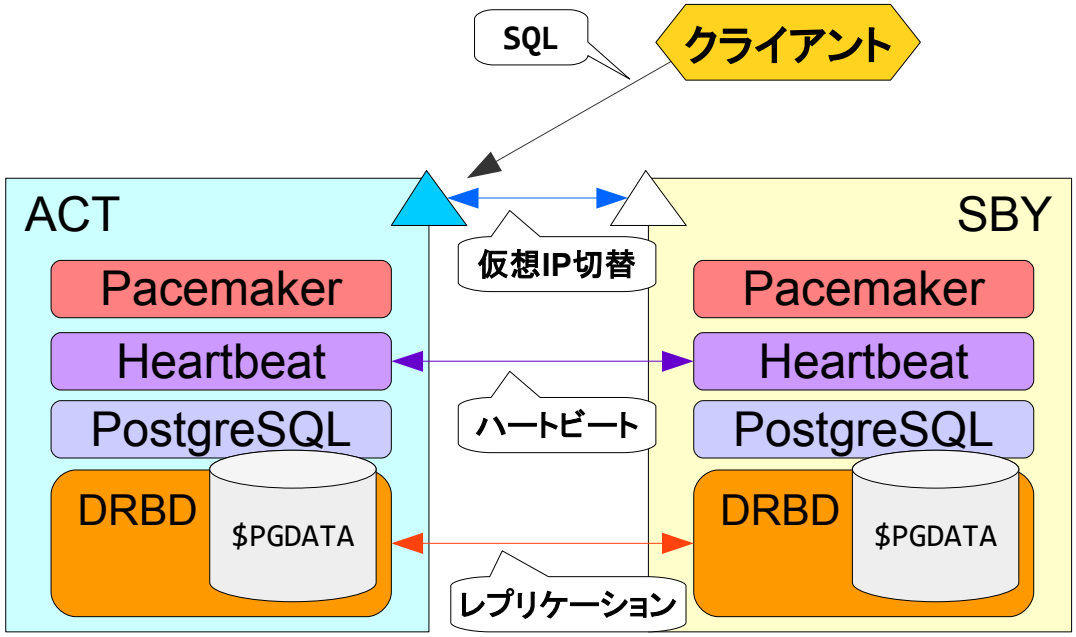
  - 稼働サーバへの変更をTCP/IP経由でミラーリング
  - ネットワークを介したRAID1のようなもの
  - 待機サーバではマウントできない

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

構成例

ソフトウェア構成図

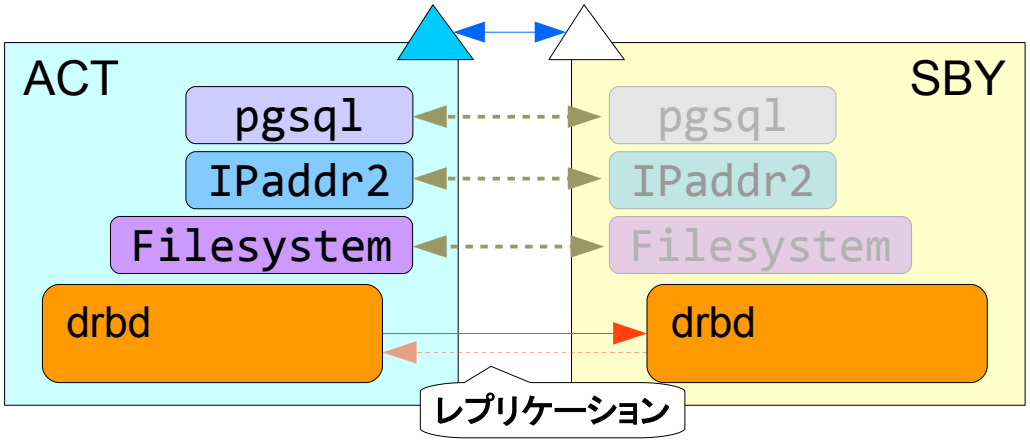
- PostgreSQL＋Pacemaker＋DRBD



  - DRBDは同期モードでレプリケーション
  - DRBDデバイス上にファイルシステムを作成しPostgreSQLデータベースクラスタを作成
  - Pacemakerが全てのリソースを監視

リソース構成図

- PostgreSQL＋Pacemaker＋DRBD



  - 4つのリソースは全て同じノードで起動するよう定義
    - IPaddr2　　：仮想IPアドレスの監視／付与／剥奪
    - pgsql　　：PostgreSQLサーバの監視／起動／停止
    - Filesystem　：ファイルシステムの監視／マウント／アンマウント
    - drbd　　：DRBDの監視／稼働ノード切り替え／起動／停止
  - いずれかに障害が発生するとPacemakerがそれを検知して、全てのサービスをSBYへ切り替えることでサービスを継続

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

インストール

細かい説明は割愛します

PostgreSQLのインストール

ACT
SBY
SRA OSS,INC.

- PGDGのリポジトリでPostgreSQLをインストール
  - PGDGとは
    - PostgreSQL Global Development Group
    - PostgreSQLの開発コミュニティ
    - http://www.postgresql.org/

1. リポジトリのパッケージをインストール

```
# yum -y install http://yum.postgresql.org/9.3/redhat/rhel-6-x86_64/pgdg-centos93-9.3-1.noarch.rpm
（省略）
Complete!
```

  - CentOS 6 x64以外については以下のURLを参照
    - http://yum.postgresql.org/repopackages.php

2. PostgreSQLのパッケージをインストール

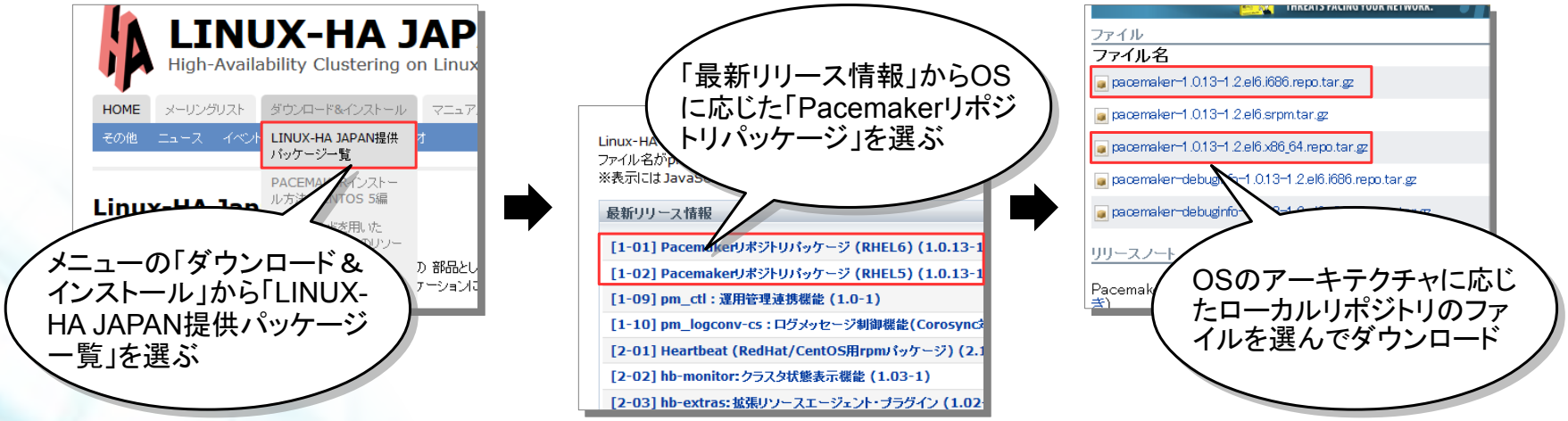
```
# yum -y groupinstall "PostgreSQL Database Server 9.3 PGDG"
（省略）
Complete!
```

Linux-HA Japanリポジトリのダウンロード

ACT SBY SRA OSS, INC.

- Linux-HA Japanのローカルリポジトリでpacemakerをインストール
  - Linux-HA Japanとは
    - LinuxでHAクラスタを実現するソフトに関する日本のコミュニティ
    - http://linux-ha.sourceforge.jp/

1. ローカルリポジトリをダウンロード



RHEL 6系 x64	pacemaker-1.0.13-1.2.el6.x86_64.repo.tar.gz
RHEL 6系 x86	pacemaker-1.0.13-1.2.el6.i686.repo.tar.gz
RHEL 5系 x64	pacemaker-1.0.13-1.2.el5.x86_64.repo.tar.gz
RHEL 5系 x86	pacemaker-1.0.13-1.2.el5.i386.repo.tar.gz

Linux-HA Japanリポジトリの設定

2. ローカルリポジトリを/tmpディレクトリに展開

```
# cd /tmp
# tar -xzf ~/Downloads/pacemaker-1.0.13-1.2.el6.x86_64.repo.tar.gz
```

- ほかのディレクトリに展開した場合には、展開先のディレクトリに合わせてリポジトリの設定を変更

```
# vim ./pacemaker-1.0.13-1.2.el6.x86_64.repo/pacemaker.repo
```

```
[pacemaker]
name=pacemaker
baseurl=file:///tmp/pacemaker-1.0.13-1.2.el6.x86_64.repo/
```

3. OS標準のリポジトリからPacemaker関連のパッケージがインストールされないようにリポジトリの設定を変更

```
# vim /etc/yum.repos.d/CentOS-Base.repo
```

```
[base]
(省略)
exclude=cluster-glue* corosync* pacemaker* resource-agents*

[updates]
(省略)
exclude=cluster-glue* corosync* pacemaker* resource-agents*
```

PacemakerとHeartbeatパッケージのインストール

4. PacemakerとHeartbeatのパッケージをインストール

```
# cd /tmp/pacemaker-1.0.13-1.2.el6.x86_64.repo
# yum -c pacemaker.repo -y install pacemaker heartbeat
（省略）
Complete!
```

- リポジトリに含まれるそのほかのパッケージ

パッケージ名	説明
pm_crmgen	Excelなどの表計算ソフトで編集したCSVファイルからPacemakerの設定ファイルを生成するツール
pm_diskd	ディスクにアクセスして読み取りを行うことでディスクの監視を行うリソースエージェント
pm_extras	NetVault Backupクライアント用のリソースエージェント（NVclient）、仮想IPアドレスの排他制御によるスプリットブレイン防止用のリソースエージェント（VIPcheck）、ハートビート通信用のネットワークインタフェースの状態を表示する機能（ifcheckd）、STONITH実行時に停止するノードを判断するSTONITHモジュール（stonith-helper）
pm_kvm_tools	KVM上のPacemakerと仮想マシン上のPacemakerを連携する機能
pm_logconv-hb	Pacemaker、Heartbeatのログメッセージを読みやすく変換する機能
vm-ctl	仮想マシンリソースを制御するツール

ELRepoリポジトリの設定

- ELRepoのリポジトリでDRBDをインストール
  - ELRepoとは
    - RHEL系のOS向けに標準でないパッケージを提供するプロジェクト
    - http://elrepo.org/

1. 公開キーをインポート

```
# rpm --import http://www.elrepo.org/RPM-GPG-KEY-elrepo.org
```

   - パッケージの署名をチェックするのに使う

2. リポジトリのパッケージをインストール

```
# rpm -ivh http://www.elrepo.org/elrepo-release-6-5.el6.elrepo.noarch.rpm
http://www.elrepo.org/elrepo-release-6-5.el6.elrepo.noarch.rpm を取得中
準備中...                ########################################### [100%]
   1:elrepo-release      ########################################### [100%]
```

RHEL 6系 `elrepo-release-6-6.el6.elrepo.noarch.rpm`

RHEL 5系 `elrepo-release-5-5.el5.elrepo.noarch.rpm`

DRBDパッケージのインストール

3. DRBDのパッケージをインストール

```
# yum -y kmod-drbd84 drbd84-utils
(省略)
Complete!
```

4. DRBDはPacemakerによって起動されるので、OSの起動時に自動的に起動されないように設定を変更

```
# chkconfig drbd off
$ chkconfig --list drbd
drbd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

OSの設定

細かい説明は割愛します

ファイアウォールの設定

ACT SBY SRA OSS,INC.

- PostgreSQL、Heartbeat、DRBDで使うポートを開く

サービス名	ポート	プロトコル
PostgreSQL	5432	TCP
Heartbeat	694	UDP
DRBD	7788	TCP

1. ファイアウォールの設定を行う

```
# vim /etc/sysconfig/iptables
```

```
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 694 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 7788 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

直接ファイルを編集せずにsystem-config-firewallを使ってもOK

2. ファイアウォールを再起動

```
# service iptables restart
iptables: ファイアウォールルールを消去中:                  [  OK  ]
iptables: チェインをポリシー ACCEPT へ設定中filter         [  OK  ]
iptables: モジュールを取り外し中:                          [  OK  ]
iptables: ファイアウォールルールを適用中:                  [  OK  ]
```

SELinuxの設定

- SELinuxの設定は難しいので無効にする
  - 無効にしないと動かないわけではないが、ちゃんと設定してあげないと動かないこともある

1. SELinuxによるアクセス制御を一時的に無効にする

```
# setenforce 0
```

2. OSの起動時に完全に無効になるようにSELinuxの設定を変更

```
# vim /etc/selinux/config
```

```
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
```

  - アクセス制御自体は無効になっているので、OSを再起動しなくてもOK

Syslogの設定

- PostgreSQL、Pacemaker、DRBDのログが専用のファイルに出力されるようにSyslog(rsyslog)の設定を行う
- Syslogの設定を行う

```
# vim /etc/rsyslog.conf
```

```
#### RULES ####

local0.*                                                /var/log/ha-log
& ~
:msg, contains, "drbd"                                  /var/log/ha-log
& ~
(省略)
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
```

  - ファシリティがlocal0のログと「drbd」を含むログを取得

- Syslogを再起動

```
# service rsyslog restart
システムロガーを停止中:                                    [  OK  ]
システムロガーを起動中:                                    [  OK  ]
```

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

Heartbeatの設定

細かい説明は割愛します

Heartbeatの設定

- テンプレートをコピーして設定ファイルを編集

```
# cp /usr/share/doc/heartbeat-3.0.5/ha.cf /etc/ha.d
# vim /etc/ha.d/ha.cf
```

```
pacemaker respawn
logfacility local0
#udpport 694
ucast eth0 192.168.137.101    # alice
ucast eth0 192.168.137.102    # bob
node alice
node bob
（抜粋）
```

「#(番号記号)」から行末まではコメント

- おもなパラメータ
  - pacemaker respawn
    - Pacemakerといっしょに動かすかどうか(動かすならrespawn)を指定
    - テンプレートに書かれていないのでパラメータ自体を追加
  - log_facility local0
    - Syslogファシリティを指定
  - udpport 694
    - ハートビート通信に使うポートを指定

Heartbeatのおもなパラメータ

- おもなパラメータ(続き)
  - `ucast eth0 192.168.137.101  # alice`
    `ucast eth0 192.168.137.102  # bob`
    - ユニキャストによるハートビート通信の設定としてデバイスと相手のIPアドレスを指定
    - 自分のIPアドレスを指定する必要はないが、ノード間で設定ファイルを同じにしておいたほうが管理しやすい
    - マルチキャスト(mcast)、ブロードキャスト(bcast)でも指定できる
  - `node alice`
    `node bob`
    - クラスタに加えるノードを指定
    - ノード名は「uname -n」で返ってくるホスト名と同じでなければならない
- そのほかのパラメータについてはテンプレートのコメントや「man ha.cf」でマニュアルを参照

認証用キーの設定

- ハートビート通信でノードの認証に使うキーの設定を行う

1. 認証用キーファイルのテンプレートをコピー

```
# cp /usr/share/doc/heartbeat-3.0.5/authkeys /etc/ha.d
```

2. rootユーザにしか読めないようにアクセス権を変更

```
# chmod 0600 /etc/ha.d/authkeys
```

3. 認証用キーファイルを編集

```
# vim /etc/ha.d/authkeys
```

```
auth 1
1 sha1 1d55b4aa7fe26469aee71baacd6f8e26e51b7d5c
```

- authの後にどの設定を使うかの番号を指定
- 設定には番号、ハッシュ化方式、認証用キーを指定
- 認証用キーは推測しにくいものであればOK

```
# head -c 8k /dev/urandom | sha1sum
d04675e6abdcb49913a0f94a08d760953cbd9430  -
```

Heartbeatの起動

- 設定ファイルを待機系ノードにコピーし、両方のノードでHeartbeatを起動

ACT

```
# scp /etc/ha.d/ha.cf /etc/ha.d/authkeys bob:/etc/ha.d
root@bob's password: （パスワードを入力）
ha.cf                                         100%   10KB  10.4KB/s   00:00
authkeys                                      100%  700     0.7KB/s   00:00
```

ACT
SBY

```
# service heartbeat start
Starting High-Availability services: Done.
```

- Heartbeatが起動していることを確認

ACT

```
# crm_mon
Version: 1.0.13-30bb
2 Nodes configured,
0 Resources configur
============

Online: [ alice bob ]

（Ctrl＋Cで終了）
```

しばらく待って「Online」の後にすべてのノードが表示されればOK

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

DRBDの設定

細かい説明は割愛します

DRBD用パーティションの準備

- DRBD用のパーティションを準備
  - ディスクまたはパーティション（/dev/sdb1など）
  - とりあえず試してみたいだけなら、ループバックデバイス（/dev/loop0など）でもOK

```
# mkdir /srv/images
# dd if=/dev/zero of=/srv/images/pgsql-data.img bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 5.30452 s, 202 MB/s
# losetup /dev/loop0 /srv/images/pgsql-data.img
# vim /etc/rc.local
```

```
touch /var/lock/subsys/local
losetup /dev/loop0 /srv/images/pgsql-data.img
```

DRBDリソースの設定

- DRBDリソースの設定ファイルを作成し、待機系ノードにコピー

```
# vim /etc/drbd.d/pgsql.res
```

```
resource pgsql {
  protocol C;
  meta-disk internal;
  disk {
    resync-rate 40M;
  }
  on alice {
    device /dev/drbd0;
    disk /dev/loop0;
    address 192.168.137.101:7788;
  }
  on bob {
    device /dev/drbd0;
    disk /dev/loop0;
    address 192.168.137.102:7788;
  }
}
```

```
# scp /etc/drbd.d/pgsql.res bob:/etc/drbd.d
root@bob's password: （パスワードを入力）
pgsql.res                                     100%  321     0.3KB/s   00:00
```

DRBDリソースの起動とデータの初期同期

- メタデータを作成し、DRBDリソースを起動

ACT
SBY

```
# drbdadm create-md pgsql
Writing meta data...
initializing activity log
NOT initializing bitmap
New drbd meta data block successfully created.
success
# drbdadm up pgsql
```

- データの初期同期を行う

ACT

```
# drbdadm disk-options --resync-rate=110M pgsql
# drbd-overview
  0:pgsql/0  Connected Secondary/Secondary Inconsistent/Inconsistent C
r-----
# drbdadm primary --force pgsql
# drbd-overview
  0:pgsql/0  SyncSource Primary/Secondary UpToDate/Inconsistent C r-----
        [=========>.........] sync'ed: 50.4% (524092/1048508)K
# drbd-overview
  0:pgsql/0  Connected Primary/Secondary UpToDate/UpToDate C r-----
# drbdadm adjust pgsql
```

ファイルシステムの作成とマウント

- DRBDデバイスにファイルシステムを作成し、マウント

```
# mkfs.ext4 /dev/drbd0
mke2fs 1.41.12 (17-May-2010)
Discarding device blocks: done
Filesystem label=
(省略)

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
# mkdir /mnt/pgsql-data
# mount -t ext4 /dev/drbd0 /mnt/pgsql-data
# df
Filesystem           1K-ブロック     使用    使用可 使用% マウント位置
/dev/mapper/VolGroup-lv_root
                      14006192   5254108   8040612  40% /
tmpfs                   506208        72    506136   1% /dev/shm
/dev/sda1               495844     73912    396332  16% /boot
/dev/drbd0             1032020     17668    961928   2% /mnt/pgsql-data
```

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

PostgreSQLの設定

細かい説明は割愛します

データベースクラスタの作成

- DRBDデバイス上のファイルシステムにデータベースクラスタを作成

```
# mkdir /mnt/pgsql-data/data
# chmod 0700 /mnt/pgsql-data/data
# chown postgres:postgres /mnt/pgsql-data/data
# su - postgres
$ /usr/pgsql-9.3/bin/initdb -A md5 -D /mnt/pgsql-data/data -E UTF8 \
>     --locale=C -W
（省略）
新しいスーパーユーザのパスワードを入力してください：（パスワードを入力）
再入力してください：（もう一度パスワードを入力）
（省略）
成功しました。以下を使用してデータベースサーバを起動することができます。

    /usr/pgsql-9.3/bin/postmaster -D /mnt/pgsql-data/data
または
    /usr/pgsql-9.3/bin/pg_ctl -D /mnt/pgsql-data/data -l logfile start

$ ls /mnt/pgsql-data/data
PG_VERSION  pg_hba.conf    pg_serial     pg_subtrans  postgresql.conf
base        pg_ident.conf  pg_snapshots  pg_tblspc
global      pg_multixact   pg_stat       pg_twophase
pg_clog     pg_notify      pg_stat_tmp   pg_xlog
```

PostgreSQLの設定

- リモートホストからの接続を監視し、Syslogでログを取得するようにPostgreSQLの設定を変更

```
$ vim /mnt/pgsql-data/data/postgresql.conf
```

```
listen_addresses = '*'                  # what IP address(es) to listen on;
#port = 5432                            # (change requires restart)
log_destination = 'syslog'              # Valid values are combinations of
logging_collector = off                 # Enable capturing of stderr and csvlog
#syslog_facility = 'LOCAL0'
log_line_prefix = ''                    # special values:
（抜粋）
```

- 同じネットワーク内から接続できるようにクライアント認証の設定を変更

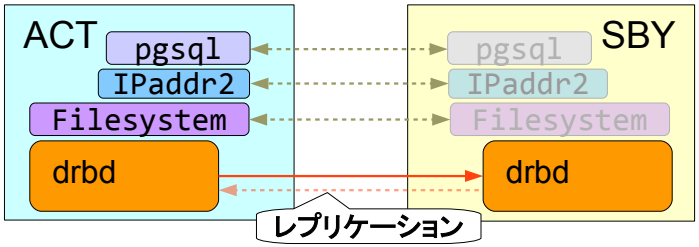
```
$ vim /mnt/pgsql-data/data/pg_hba.conf
```

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             192.168.137.0/24        md5

# "local" is for Unix domain socket connections only
local   all             all                                     md5
```

PostgreSQLの起動と停止

- PostgreSQLを起動し、ネットワーク越しに接続できることを確認し、停止

```
$ /usr/pgsql-9.3/bin/pg_ctl start -w -D /mnt/pgsql-data/data
サーバの起動完了を待っています....LOG:  ending log output to stderr
HINT:  Future log output will go to log destination "syslog".
完了
サーバ起動完了
$ psql -h alice
パスワード:（パスワードを入力）
psql (9.3.3)
"help" でヘルプを表示します.

=# \q
$ /usr/pgsql-9.3/bin/pg_ctl stop -D /mnt/pgsql-data/data
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
$ exit
```



PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

Pacemakerの設定

リソース全体の設定

ACT

SRA OSS, INC.

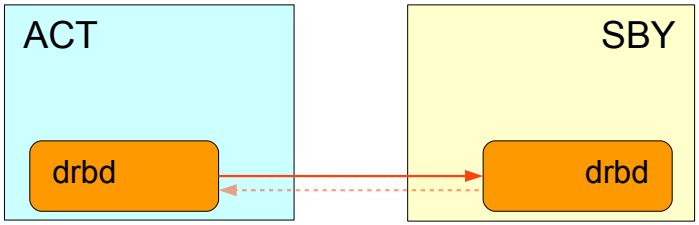
- Pacemakerとリソースのデフォルト値の設定
  - ファイルに記述してcrmコマンドで読み込む

```
# vim pgsql.crm
property \
    no-quorum-policy="ignore" \
    stonith-enabled="false" \
    startup-fencing="false"

rsc_defaults \
    resource-stickiness="INFINITY" \
    migration-threshold="2" \
    failure-timeout="60s"

（次のページに続く）
```

- no-quorum-policy
  HAクラスタに参加しているノードの過半数のノードとハートビート通信できない場合のポリシー
  2台構成ではignoreを指定する必要がある
- stonith-enabled
  STONITH機能を有効にするか
- startup-fencing
  Paemaker起動時に相手が確認できない場合相手をフェンシングするか
- resource-stickness
  自動フェイルバックの設定
  （無効：INFINITY, 有効：1）
- migration-threshold
  フェイルオーバーを実行するリソース故障検知数
  （フェイルオーバしない：0）
- failure-timeout
  リソース故障判定のデフォルトタイムアウト時間

DRBDリソースの設定

- 「primitiveリソース」
  - すべてのリソースの最小単位
  - クラスタ全体のうち1ノードのみで動作させるリソースを定義
  - リソースの設定オプションは「crm ra info リソース名」で確認



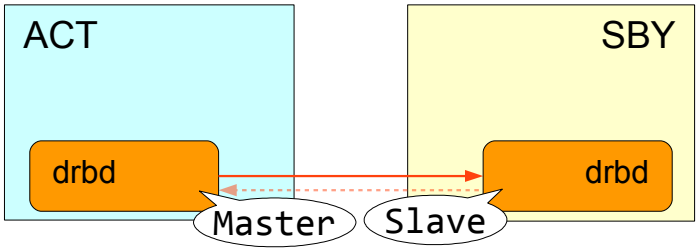
- DRBDリソース

```
primitive drbd ocf:linbit:drbd \
    params \
        drbd_resource="pgsql" \
    op start timeout="240s" \
    op stop timeout="100s" \
    op monitor interval="10s" timeout="20s" on_fail="restart" role="Master"\
    op monitor interval="20s" timeout="20s" on_fail="restart" role="Slave" \
    op promote timeout="90s" \
    op demote timeout="90s"

(次のページに続く)
```

  - op には各監視項目の設定を記述

マスタスレーブリソースの設定

- 「master/slaveリソース」
  - 稼働系と待機系の状態があるリソースはprimitiveリソースとして定義した後にmaster/slave化する

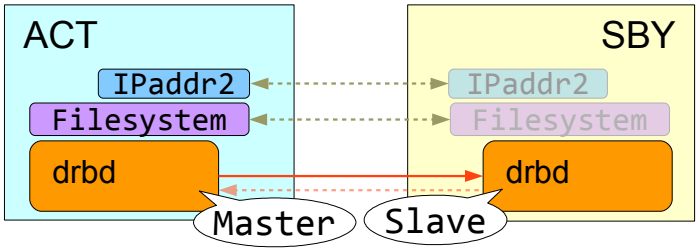



```
ms ms_drbd drbd \
    meta \
        master-max="1" \
        master-node-max="1" \
        clone-max="2" \
        clone-node-max="1" \
        notify="true"

（次のページに続く）
```

- master-max
  クラスタ全体のマスタ最大数
- master-node-max
  ノードあたりのマスタ最大数
- clone-max
  primitiveリソースをクローンする数
- clone-node-max
  ノードあたりのクローン最大数
- notify
  DRBDリソースではtrueに

リソース設定

- ファイルシステムリソース
- 仮想IPリソース

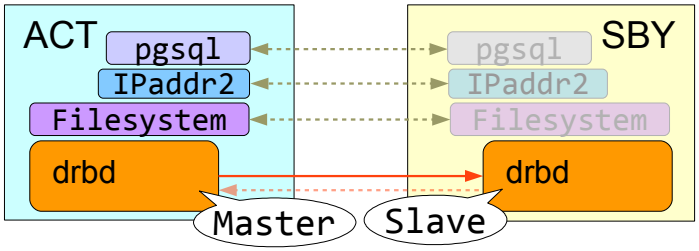



```
primitive filesystem ocf:heartbeat:Filesystem \
    params \
        device="/dev/drbd0" \
        directory="/mnt/pgsql-data" \
        fstype="ext4" \
    op start timeout="60s" \
    op stop timeout="60s" \
    op monitor interval="20s" timeout="40s" on_fail="restart"

primitive ipaddr ocf:heartbeat:IPaddr2 \
    params \
        ip="192.168.137.201" \
        nic="eth0" \
        cidr_netmask="24" \
    op start timeout="20s" \
    op stop timeout="20s" \
    op monitor interval="10s" timeout="20s" on_fail="restart"
```

(次のページに続く)

PostgreSQLグループリソース設定

- PostgreSQLリソースを作成
- グループ制約
  - 指定したリソースは同ノードで起動
  - 指定した順序で起動

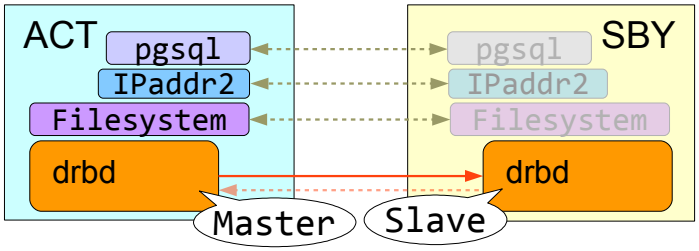



```
primitive pgsql ocf:heartbeat:pgsql \
    params \
        pgctl="/usr/pgsql-9.3/bin/pg_ctl" \
        start_opt="-p 5432" \
        psql="/usr/bin/psql" \
        pgdba="postgres" \
        pgdata="/mnt/pgsql-data/data" \
        pgport="5432" \
        monitor_user="postgres" \
        monitor_password="password" \
    op start timeout="120s" \
    op stop timeout="120s" \
    op monitor interval="30s" timeout="30s" on_fail="restart"

group group_pgsql \
    filesystem ipaddr pgsql
```

(次のページに続く)

DRBDリソースの制約設定

- コロケーション制約
  - グループリソースと
    DRBDリソースのマスタが
    同一ノードで起動するよう設定



- オーダー制約
  - リソースの起動順序をdrbdマスタ、グループリソースに設定

```
colocation group_pgsql_and_ms_drbd_master \
    inf: group_pgsql ms_drbd:Master

order ms_drbd_promote_before_group_pgsql_start \
    inf: ms_drbd:promote group_pgsql:start
```

  - msリソースは、グループ制約に含められないため、
    上記2つの制約にて設定
    - 起動順序：DRBD → Filesystem → IPaddr2 → pgsql
    - 停止順所：起動の逆順

設定のロード

- ファイルから設定をロードしてリソースを作成

```
# crm configure load replace pgsql.crm
crm_verify[2199]: 2014/02/27_14:50:29 WARN: unpack_nodes: Blind faith:
not fencing unseen nodes
# crm_mon -r
```

```
============
Last updated: Thu Feb 27 14:51:25 2014
Stack: Heartbeat
Current DC: bob (e6cb12d9-acdc-45c7-ab3c-6f969e9d1665) - partition with quorum
Version: 1.0.13-30bb726
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ alice bob ]

 Resource Group: group_pgsql
     filesystem (ocf::heartbeat:Filesystem):    Started bob
     ipaddr     (ocf::heartbeat:IPaddr2):       Started bob
     pgsql      (ocf::heartbeat:pgsql): Started bob
 Master/Slave Set: ms_drbd
     Masters: [ bob ]
     Slaves: [ alice ]
```

PostgreSQL＋Pacemaker＋DRBDで
高可用性構成を構築してみよう

デモ

PowerGres HA Pacemaker版

PowerGres HA Pacemaker版とは

- PowerGres＋Pacemaker for PowerGres
- オープンソースのHAクラスタソフトを採用し、PowerGres向けに機能を絞り、リーズナブルな価格で提供
- 24時間365日のサポートも提供

11/20プレスリリース：http://www.sraoss.co.jp/company/press/2013/1120.php

PowerGresでPostgreSQLの信頼性をより高めよう
http://www.sraoss.co.jp/event_seminar/2013/20131127_powergres-plus-and-ha.pdf

heartbeat設定

- PowerGres HA Pacemaker版管理ツールによる構成
  - PowerGres HA Pacemaker版がインストールされた状態にてheartbeatの設定までを対話的に

```
[root@alice ~]# pwg_clumgr init-cluster
Heartbeat 設定ファイルを作成しています...
syslog ファシリティを入力してください (local0 - local7) [local0]:
ノードを追加しています...
ノード名を入力してください ("." でキャンセルします): alice
ノードの追加を継続しますか (y/n) [y]:
ノード名を入力してください ("." でキャンセルします): bob
ノードの追加を継続しますか (y/n) [y]: n
SSH 鍵を作成しています...
パスフレーズを入力してください (パスフレーズなしにするには空にしてください):
同じパスフレーズを再度入力してください:
SSH 鍵をコピーしています...

(省略)

コミュニケーションパスを追加しています...
ネットワークインタフェース名を入力してください ("." でキャンセルします) [eth0]:
コミュニケーションパスの追加を継続しますか (y/n) [y]: y
ネットワークインタフェース名を入力してください ("." でキャンセルします) [eth1]:
コミュニケーションパスの追加を継続しますか (y/n) [y]: n
```

heartbeat設定

```
============
Last updated: Tue Jun 17 17:03:24 2014
Stack: Heartbeat
Current DC: bob (cef37169-ad1a-4b63-ad9b-b915b7c890be) - partition with
quorum
Version: 1.0.13-30bb726
2 Nodes configured, unknown expected votes
0 Resources configured.
============

Online: [ alice bob ]

Full list of resources:


Node Attributes:
* Node alice:
    + bob-eth0                          : up
    + bob-eth1                          : up
* Node bob:
    + alice-eth0                        : up
    + alice-eth1                        : up

Migration summary:
* Node alice:
* Node bob:
```

リソース設定

- 各リソース設定を対話的に

```
[root@alice ~]# pwg_clumgr create-service powergres
サービスタイプを入力してください [pgsql-drbd]:
pg_ctl コマンドのパスを入力してください [/opt/powergres91/bin/pg_ctl]:
psql コマンドのパスを入力してください [/opt/powergres91/bin/psql]:
PostgreSQL のスーパーユーザ名を入力してください [postgres]:
スーパーユーザのパスワードを入力してください (パスワードなしにするには空にしてください):
同じパスワードを再度入力してください:
マウント先ディレクトリを入力してください: /disk
データディレクトリを入力してください [/disk/data]:
PostgreSQL のポート番号を入力してください (1024 - 65535) [5432]:
syslog ファシリティを入力してください (local0 - local7) [local0]:
DRBD サービス "powergres_drbd" を作成しています...
ファイルシステムタイプを入力してください [ext4]:
DRBD ブロックデバイスのマイナー番号を入力してください (0 - 1048576) [0]:
ディスクのブロックデバイス名を入力してください [/dev/sdb]:
レプリケーションパスのネットワークインタフェース名を入力してください [eth0]: eth1
レプリケーションパスのポート番号を入力してください (7788 - 7799) [7788]:
ファイルをノード "alice" にコピーしています...
powergres_drbd.res                                    100%  443     0.4KB/s   00:00
ファイルをノード "bob" にコピーしています...
powergres_drbd.res                                    100%  443     0.4KB/s   00:00
仮想 IP アドレスサービス "powergres_ipaddr" を作成しています...
IP アドレスを入力してください: 192.168.100.110
ファイルシステムがマウントされるのを待機しています... 完了。
データベースクラスタを作成しています... 完了。
```

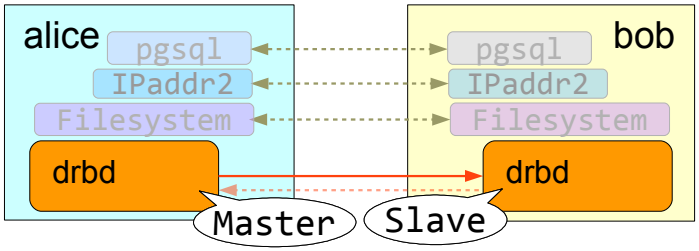
heartbeat設定

```
(省略)

Online: [ alice bob ]

Full list of resources:

 Master/Slave Set: powergres_drbd.drbd.ms_drbd
     Masters: [ alice ]
     Slaves: [ bob ]
 Resource Group: powergres.pgsql-drbd.group_pgsql
     powergres_drbd.drbd.filesystem     (ocf::heartbeat:Filesystem):    Started alice
     powergres_ipaddr.ipaddr    (ocf::heartbeat:IPaddr2):       Started alice
     powergres.pgsql-drbd.pgsql (ocf::heartbeat:pgsql): Started alice

Node Attributes:
* Node alice:
    + bob-eth0                          : up
    + bob-eth1                          : up
    + master-powergres_drbd.drbd.drbd:1 : 10000
* Node bob:
    + alice-eth0                        : up
    + alice-eth1                        : up
    + master-powergres_drbd.drbd.drbd:0 : 10000

Migration summary:
* Node alice:
* Node bob:
```

リソースを動かしてみる

- 稼働系のファイルシステム停止

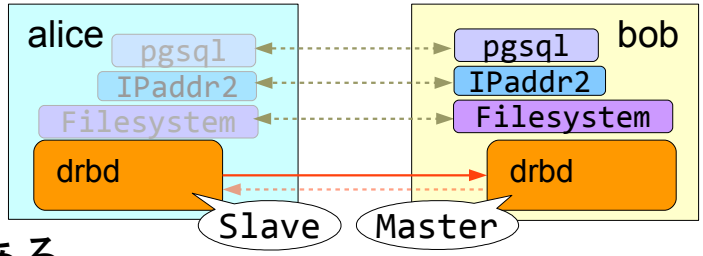
```
[root@alice ~]# crm
crm(live)# resource stop リソース名
crm(live)# resource start リソース名
```



  - group制約によりIPaddr2とpgsqlも停止

- 稼働系をaliceからbobに移行
  - location制約を追加する操作

```
crm(live)# resource migrate リソース名
```

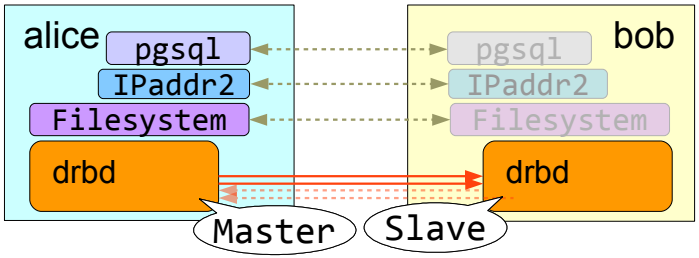


  - スイッチオーバ後は制約を削除する必要がある

```
crm(live)# resource unmigrate リソース名
```

  - 再びaliceを稼働系へ

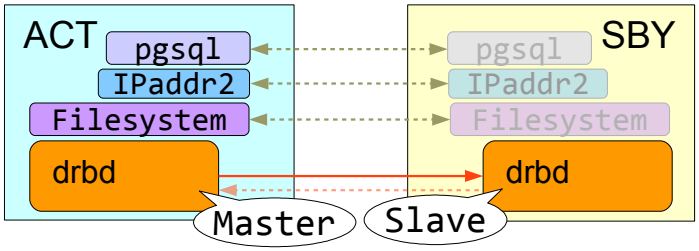
```
crm(live)# resource migrate リソース名
crm(live)# resource unmigrate リソース名
```



障害を発生させてみる

- PostgreSQLのマスタープロセスを落としてみる

```
[root@alice ~]# kill 41157
```



  - 1回目の故障検知は再起動
    - fail-count: 故障回数
    - migration-threshold: フェイルオーバ実行閾値

```
Migration summary:
* Node alice:
   powergres.pgsql-drbd.pgsql: migration-threshold=2 fail-count=1 last-failure='Wed Jun 18 14:03:53 2014'
* Node bob:
```

  - 2回目の故障でフェイルオーバ

```
Migration summary:
* Node alice:
   powergres.pgsql-drbd.pgsql: migration-threshold=2 fail-count=2 last-failure='Wed Jun 18 14:13:26 2014'
* Node bob:

Failed actions:
    powergres.pgsql-drbd.pgsql_monitor_30000 (node=alice, call=77, rc=7, status=complete): not running
```

  - Failed actionsを消すにはcleanup
    - fail-countもゼロに

```
[root@alice ~]# crm
crm(live)# resource cleanup
```

Pacemaker for PowerGres のご紹介

Pacemaker for PowerGres

オープンソースソフトウェアPacemaker、Heartbeat、DRBDによるHAクラスタ構成が簡単に構築できる!!

SRA OSSが責任を
持って対応します。

オープンソースとともに

SRA OSS, INC.