挑戦！ Pacemakerで自由自在にクラスタリング

2010年9月11日　OSC2010 Tokyo/Fall

Linux-HA Japan プロジェクト

田中崇幸

本日の話題

① Pacemakerって何？
② Pacemakerのコンポーネント構成
③ Pacemakerでクラスタリングに挑戦しよう！
④ Linux-HA Japanプロジェクトについて

①

Pacemakerって何？

簡単に言うと・・・

Pacemakerとは？

オープンソースの
HAクラスタリングソフトウェアで
実績のある「Heartbeat」の後継ソフトウェアです

Pacemakerは、サービスの可用性向上ができるHAクラスタを可能とした、コストパフォーマンスに優れたオープンソースのクラスタリングソフトウェアです。

ここで本日の客層を知るために
皆さんに質問させてください。

そのHAクラスタソフトである

Pacemaker は

知っていましたか？

同じくHAクラスタソフトウェアである

Heartbeatバージョン1 は

知っていましたか？

さらに
同じくHAクラスタソフトウェアである

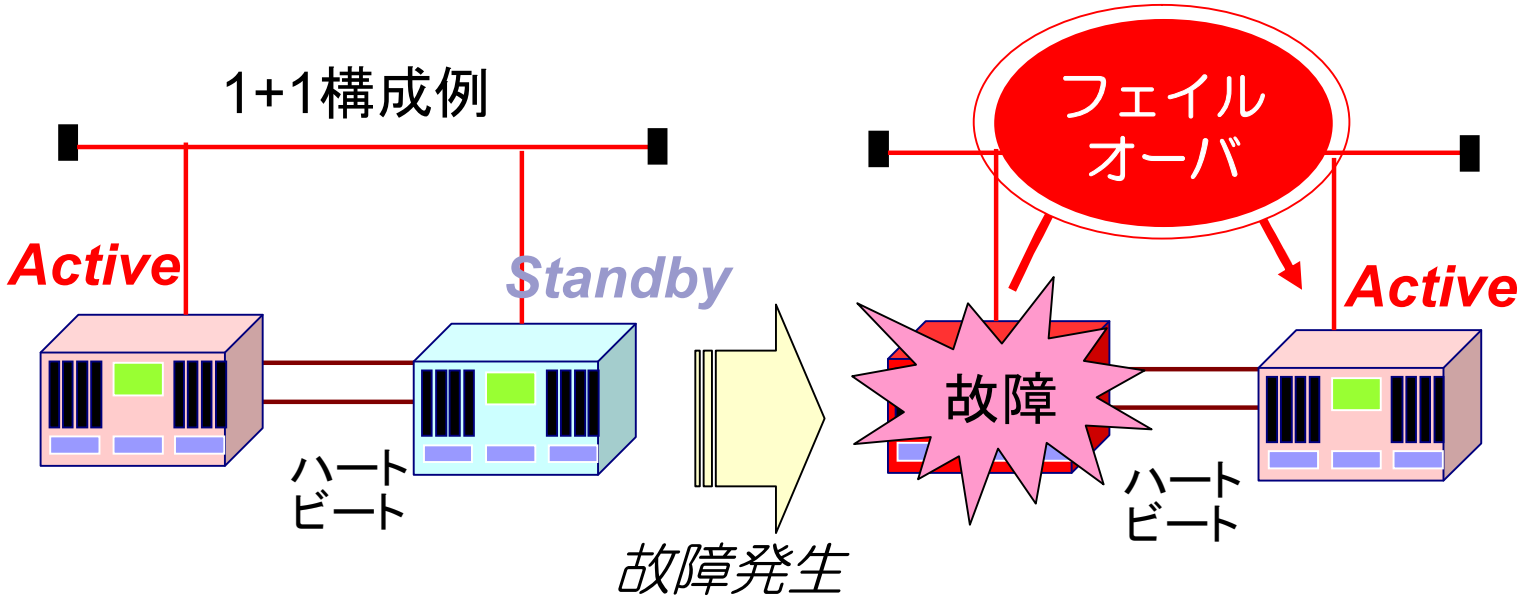
Heartbeatバージョン2 は

知っていましたか？

「Pacemaker」と「Heartbeat」は
密接な関係があるため
詳しいお話は
後ほどお話します。

PacemakerによるHAクラスタの基本構成 Active/Standby（1+1）構成

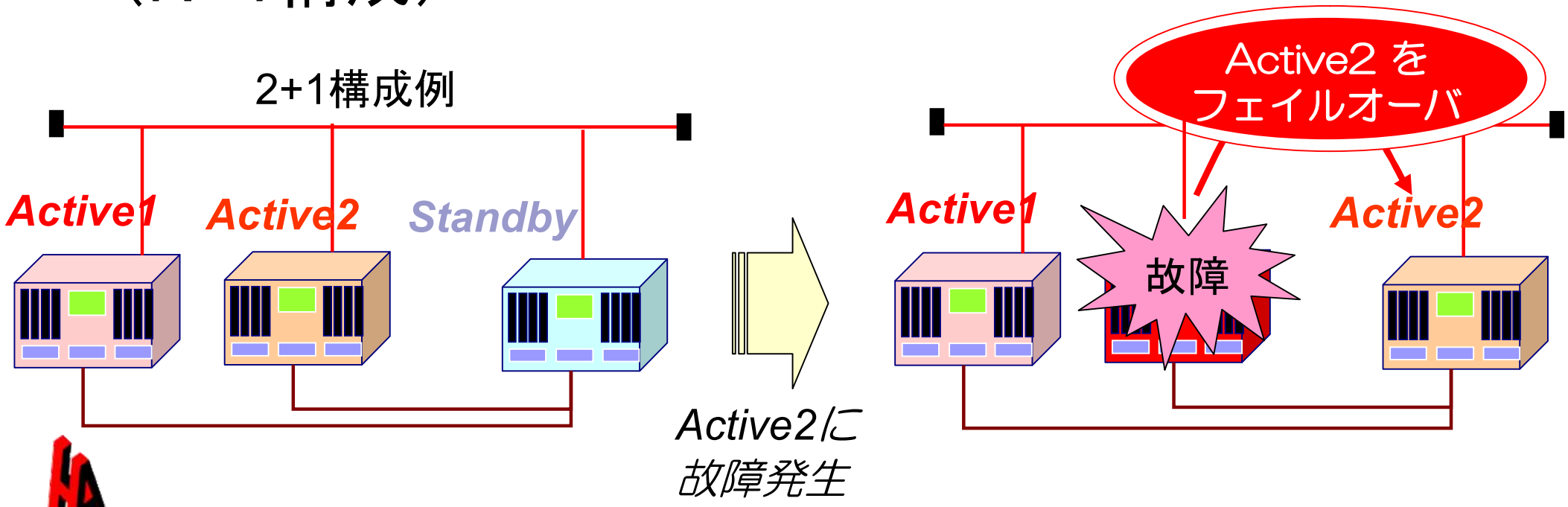
- Pacemakerは、故障発生を検知した場合、待機系へフェイルオーバさせることによってサービスの継続が可能になります。



Pacemakerでは複数台構成も可能です

※ Heartbeatバージョン1では実現できませんでした

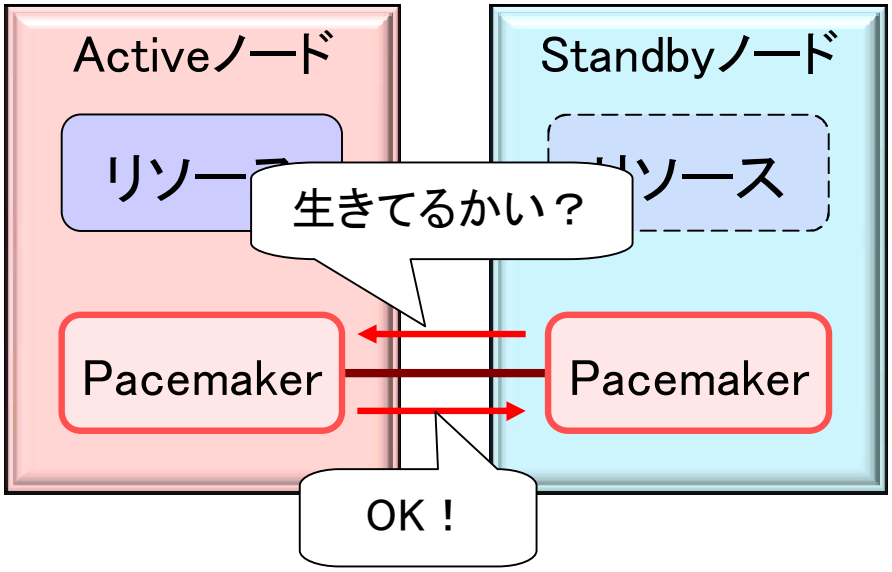
- Pacemakerでは、2台など複数台の運用系ノードに対し、待機系ノードを1台にする事も可能です。（N+1構成）

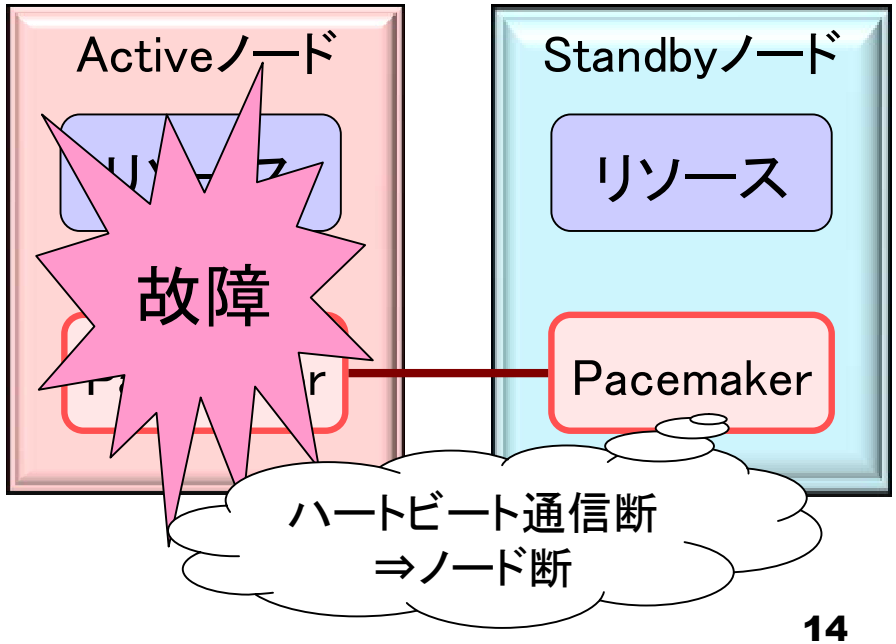


ここからPacemakerの説明は、
Active / Standby（1+1構成）の
単純構成を例としてお話します。

基本的動作：ノード監視

- 相手ノードの監視
  - 一定間隔で相手ノードと通信し、相手ノードの生死を確認します。（ハートビート通信）
  - 相手ノードと通信できなくなった場合に、相手はダウンしたと判断し、フェイルオーバなどのクラスタ制御の処理を行います。



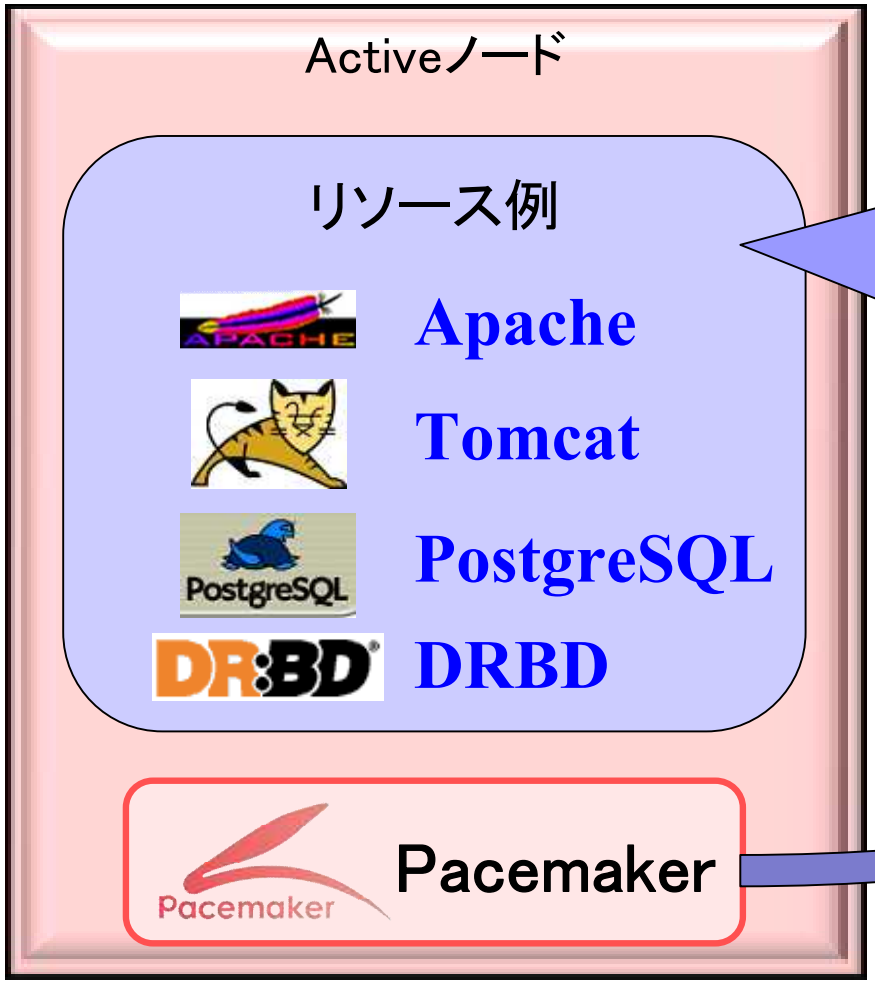


「リソース」とは？

Pacemakerではよく出てくる
言葉なのでおぼえてください！

HAクラスタにおけるリソースとは、サービスを提供するために必要な構成要素の事で、Pacemakerが起動、停止、監視等の制御対象とするアプリケーション、NIC、ディスク等を示します。

例えばこんなのが Pacemaker から見た「リソース」になります



Pacemaker から見ると、PostgreSQL などのアプリケーションは、「リソース」となります。

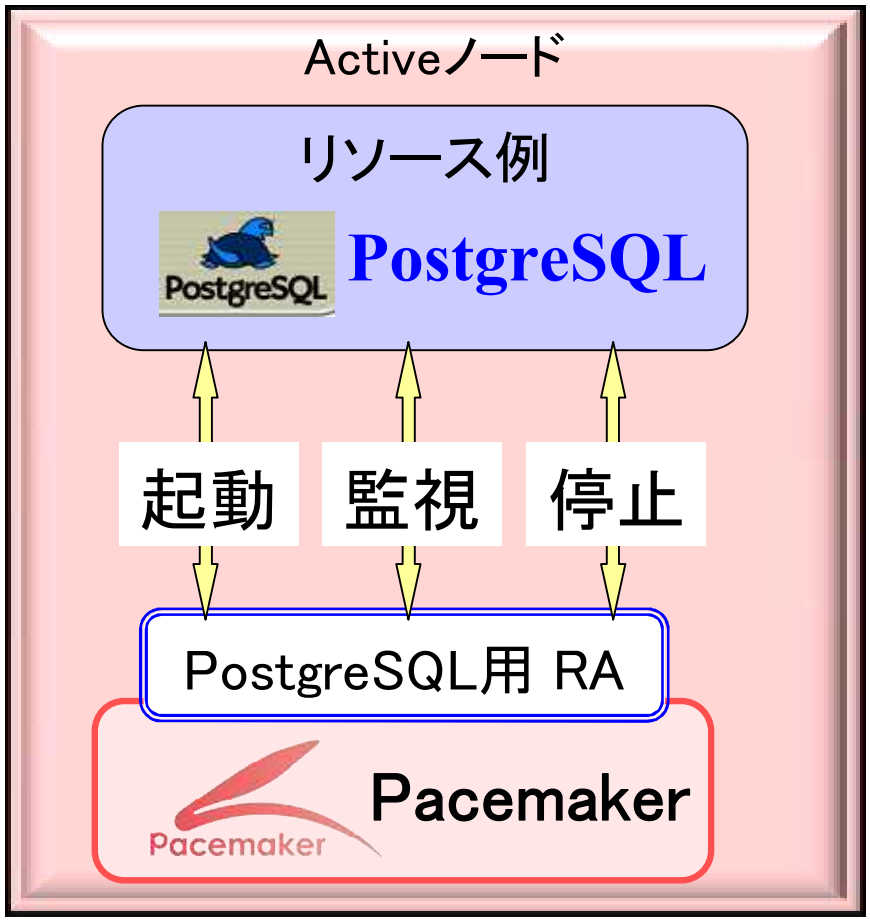
「リソースエージェント」とは？

これまたPacemakerではよく出てくる
言葉なのでおぼえてください！

リソースエージェント（RA）とは、そのリソースとPacemakerを仲介するプログラムになり、主にシェルスクリプトで作成されています。

Pacemakerは、リソースエージェントに対して指示を出し、リソースの起動(start)、停止(stop)、監視(monitor)の制御を行います。

「リソース」と「リソースエージェント」は こんな関係 になります

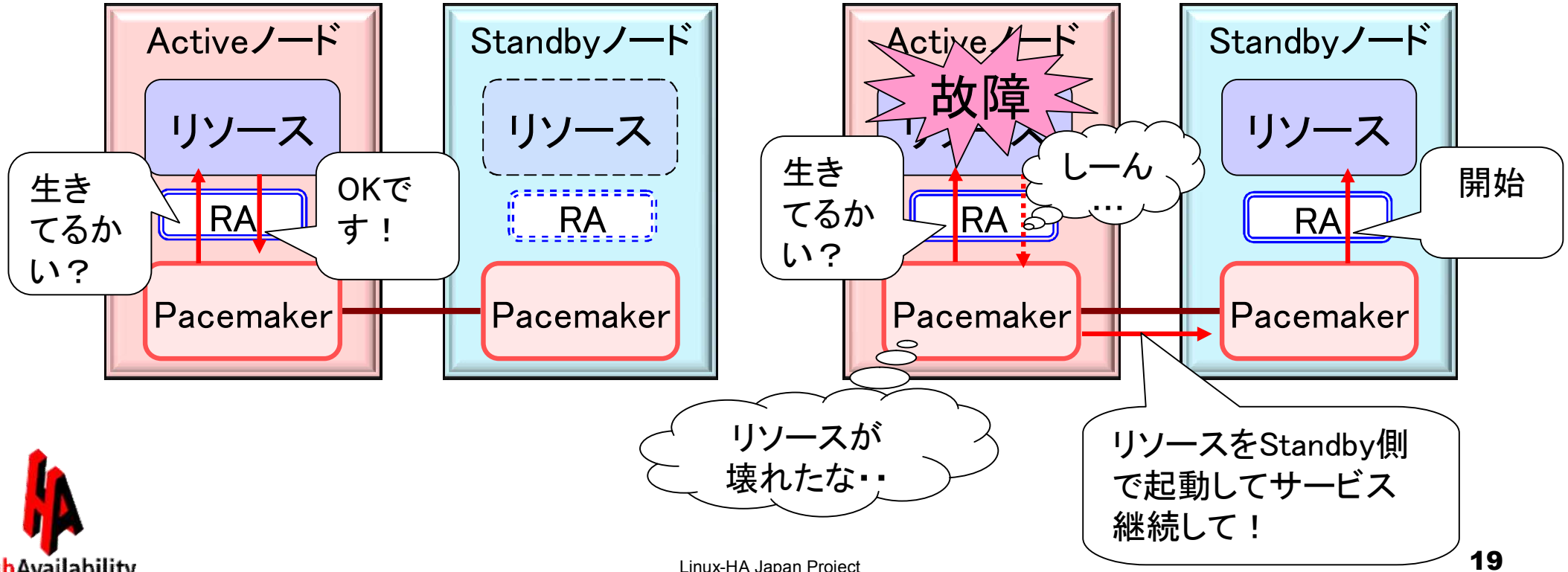


Pacemaker は、PostgreSQL などのリソースを、リソースエージェントを介して起動、停止、監視等の制御をおこなうことができます。

※ Heartbeatバージョン1ではリソース監視の機能はありませんでした

基本的動作：リソース制御

- リソースの制御：起動(start)、停止(stop)、監視(monitor)
  - 起動後は一定間隔でリソースエージェント（RA）を介してリソースを監視し、正しく動作していないと判断した場合にはフェイルオーバなどのリソース制御の処理を行います。



Pacemakerでは、Web系、DB系、ネットワーク系、ファイルシステム系等のリソースエージェントが標準で多数用意されています。

従来の Heartbeat 2.x 用に作成されたRAも使用が可能です

標準リソースエージェントの一例

分類	リソース	リソースエージェント /usr/lib/ocf/resource.d/heartbeat/ /usr/lib/ocf/resource.d/pacemaker/
ファイルシステム系	ディスクマウント	Filesystem
DB系	PostgreSQL	pgsql
Web系	Apache	apache
ネットワーク系	仮想IPアドレス	IPaddr2

pgsqlリソースエージェント

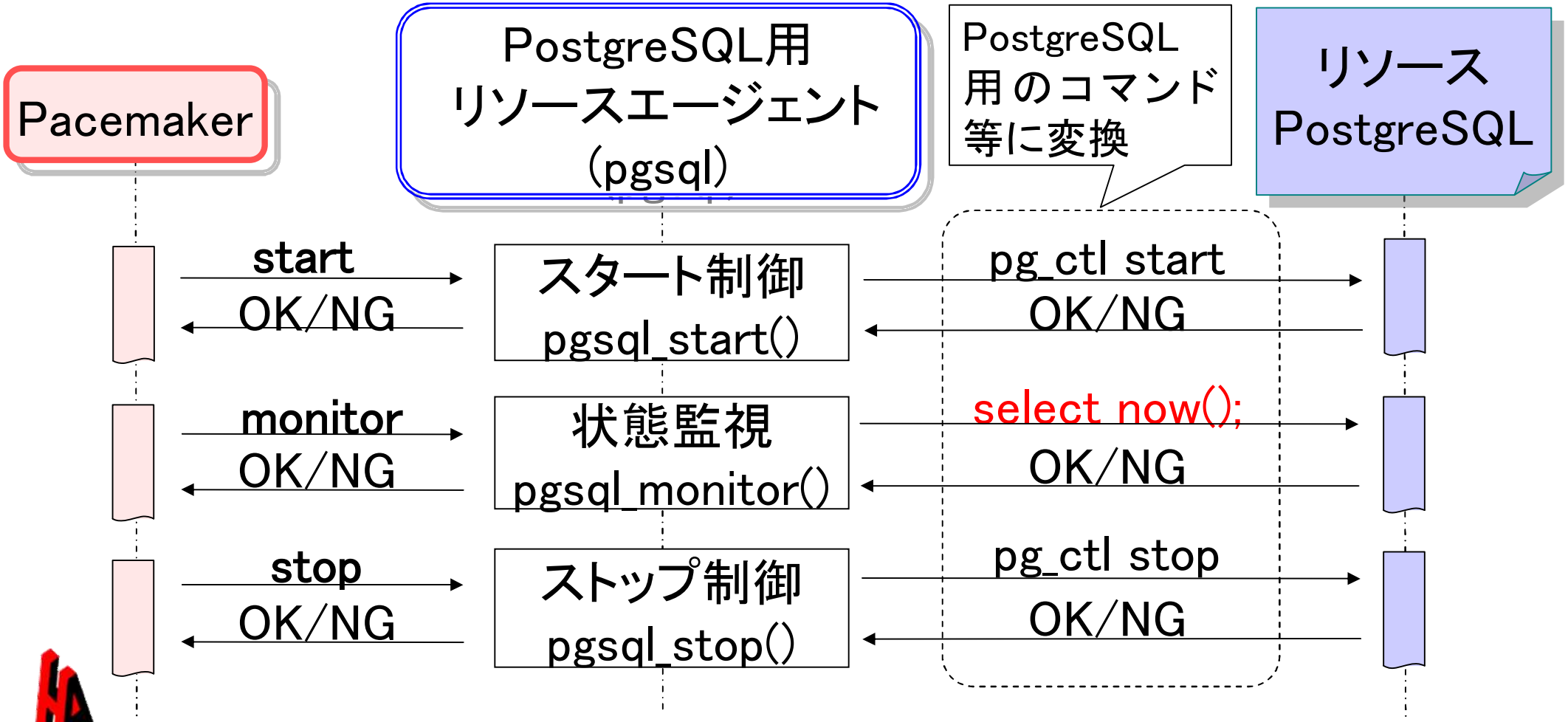
監視(monitor)処理の抜粋

```
#!/bin/sh

 (省略)
pgsql_monitor() {
    if ! pgsql_status
    then
        ocf_log info "PostgreSQL is down"
        return $OCF_NOT_RUNNING
    fi

    if [ "x" = "x$OCF_RESKEY_pghost" ]
    then
       runasowner "$OCF_RESKEY_psql -p $OCF_RESKEY_pgport -U
$OCF_RESKEY_pgdba $OCF_RESKEY_pgdb -c 'select now();' >/dev/null 2>&1"
    else
 (省略)
```

例）Pacemaker と PostgreSQLリソースエージェントの関係



リソースエージェントは自分でも作れます！

```
#!/bin/sh
. ${OCF_ROOT}/resource.d/heartbeat/.ocf-shellfuncs

start処理() {
}
stop処理() {
}
monitor処理 {
}
meta-data処理(){
}
validate-all処理(){
}

case $1 in
  start)        start処理();;
  stop)         stop処理();;
  monitor)      monitor処理();;
  …
esac
```

通常のシェルスクリプトの記述方法ですが、いくつか必須のパラメータ呼び出しに対する処理を行う必要があります。

リソース開始・監視・停止の処理

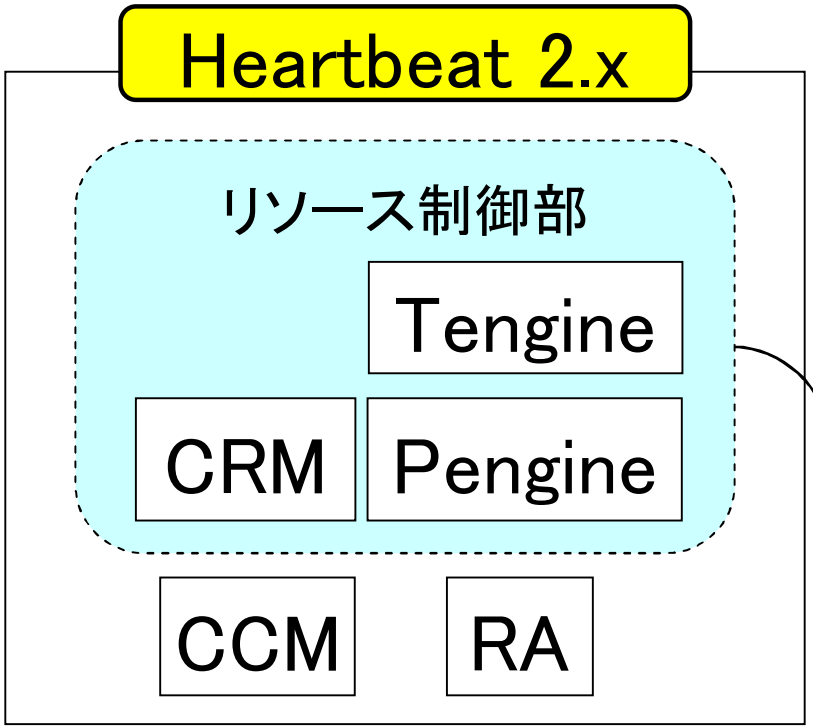
RA処理の振り分け

②

Pacemakerの
コンポーネント構成

Pacemaker のコンポーネント構成は
複数に分かれていて
単純ではないのです…

Pacemaker



Heartbeatバージョン2系の
リソース制御部が
Pacemakerとして切り出されて
リリースされました。

Pacemaker

CRM: Cluster Resource Manager
Tengine: Transition Engine
Pengine: Policy engine
CCM: Cluster Consensus Membership
RA: Resource Agent

ということは・・・
Pacemaker 単独では
HAクラスタソフトとして
動作しない？

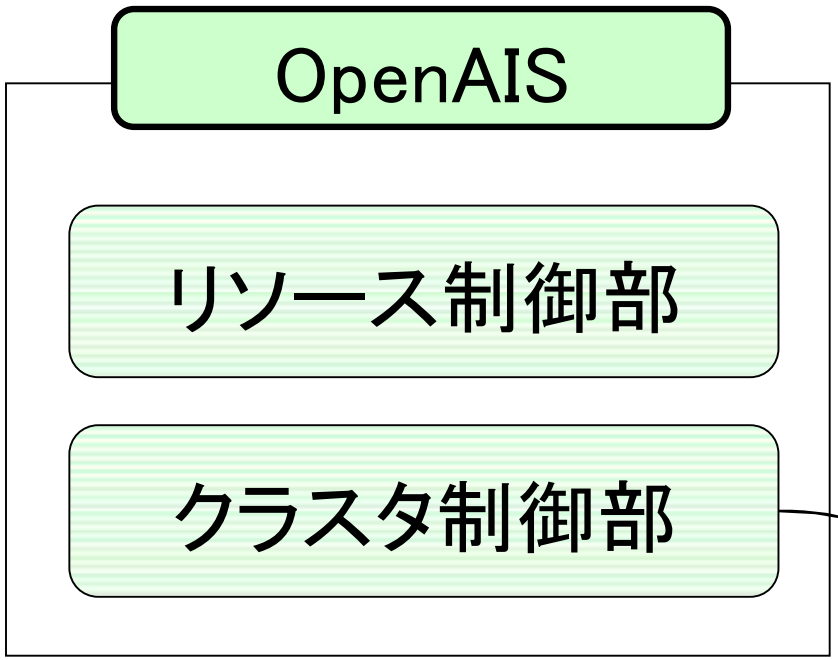
そのとおりです..

Pacemaker は
クラスタ制御部の
アプリケーションと組み合わせて
使用しなければなりません..

ですが、
クラスタ制御部の
選択肢が広がったと
前向きにとらえてください！

Corosync

クラスタ
制御部



OpenAISコミュニティによって開発されたクラスタソフトである【OpenAIS】のクラスタ制御部はCorosyncとして切り出されてリリースされました。

Corosync

つまり Corosyncも単独ではHAクラスタとしては動作しない！？

Heartbeat3

クラスタ
制御部

Heartbeatバージョン2の
クラスタ制御部は、
Heartbeatバージョン3 として切り
出されてリリースされました。

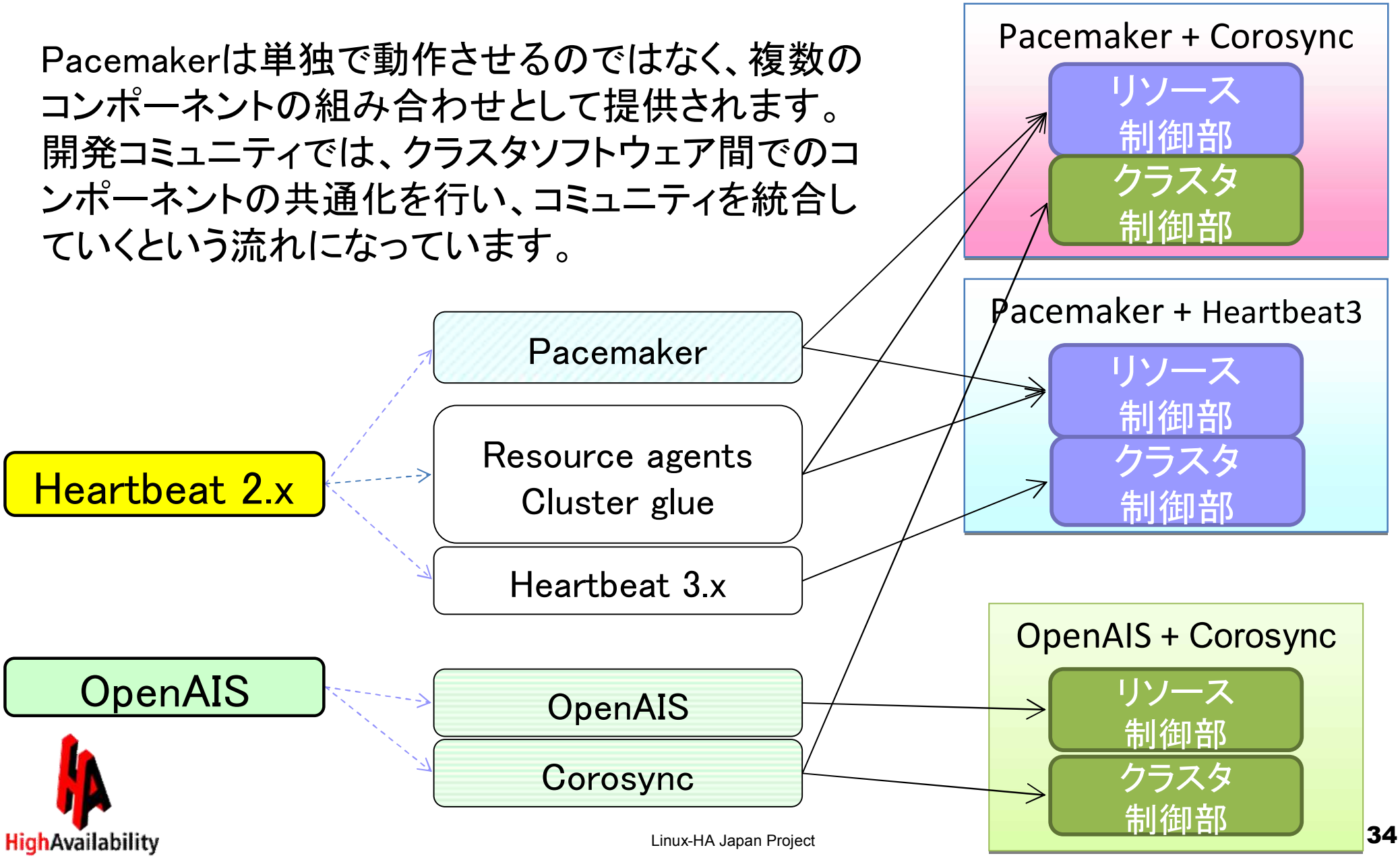
Heartbeat3

切り出されたので“2”から“3”と数字が
上がったのに、機能的にはデグレ！？

Pacemaker は
この「Corosync」と「Heartbeat3」
のクラスタ制御部が
選択可能です。

HAクラスタのリリース形態

Pacemakerは単独で動作させるのではなく、複数のコンポーネントの組み合わせとして提供されます。
開発コミュニティでは、クラスタソフトウェア間でのコンポーネントの共通化を行い、コミュニティを統合していくという流れになっています。



では、プロダクト名は

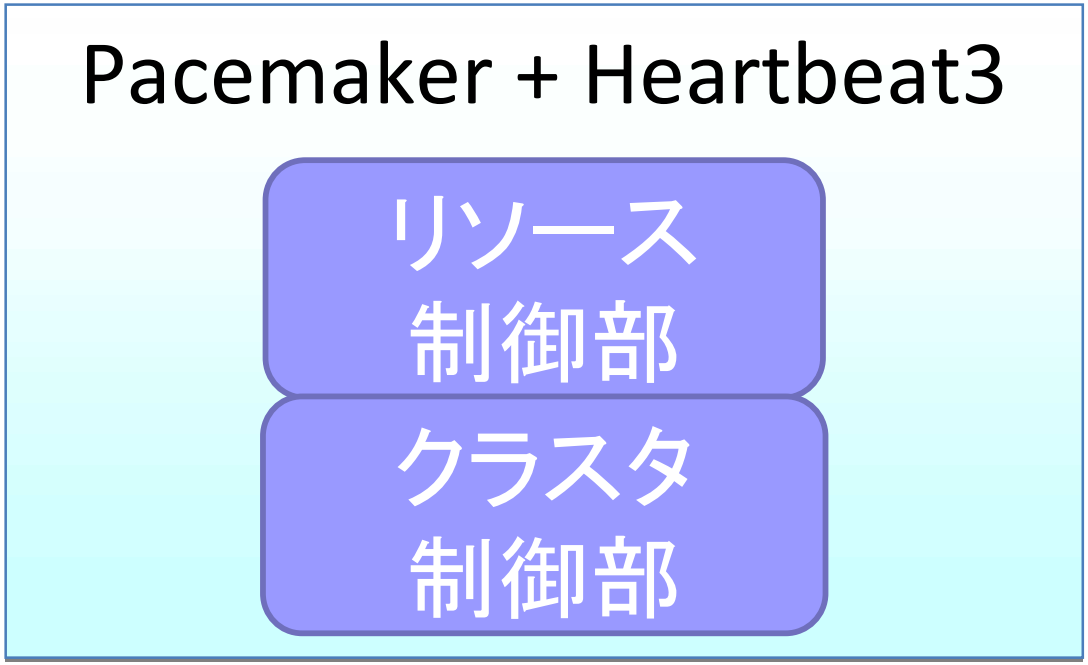
「Pacemaker ぷらす・・・」

って呼ぶの？？

それでは呼びにくいので・・・

Pacemaker + Corosync も

Pacemaker + Heartbeat3 も

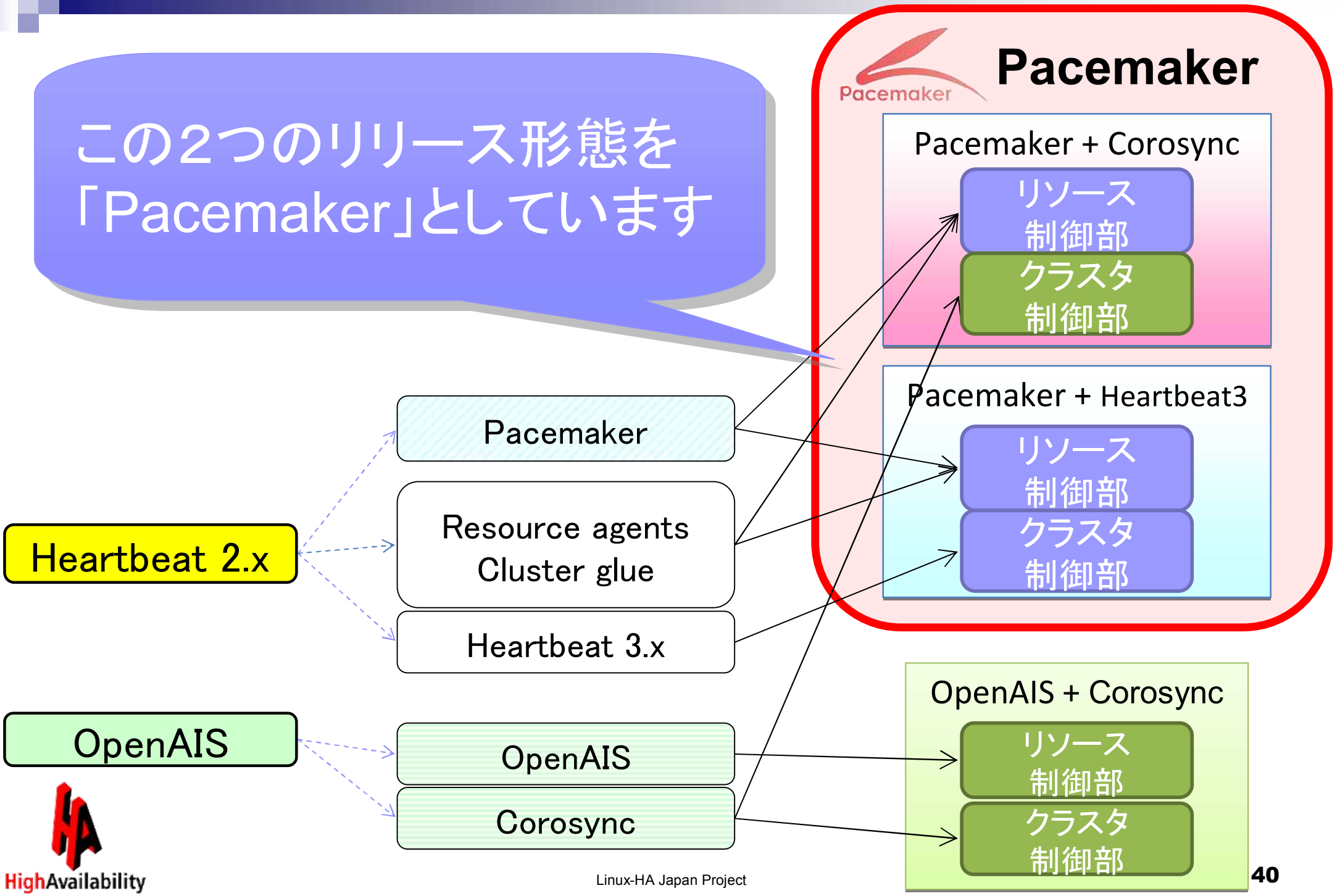


日本のLinux-HAコミュニティである
Linux-HA Japan プロジェクトでは
プロダクト名を

Pacemaker

としています。

この2つのリリース形態を「Pacemaker」としています



Heartbeat3 と Corosync どちらのクラスタ制御部が優れているの？

Corosync のメリット・デメリット

クラスタ制御部

- メリット
  - 多ノード構成に向いている
    - 10+1 構成くらいでも対応可能
  - クラスタの起動時間が短い
  - ノード故障検出時間が短い
  - スプリットブレイン回復時の動作が安定している
  - オンラインによるノード追加・削除時の動作が安定している

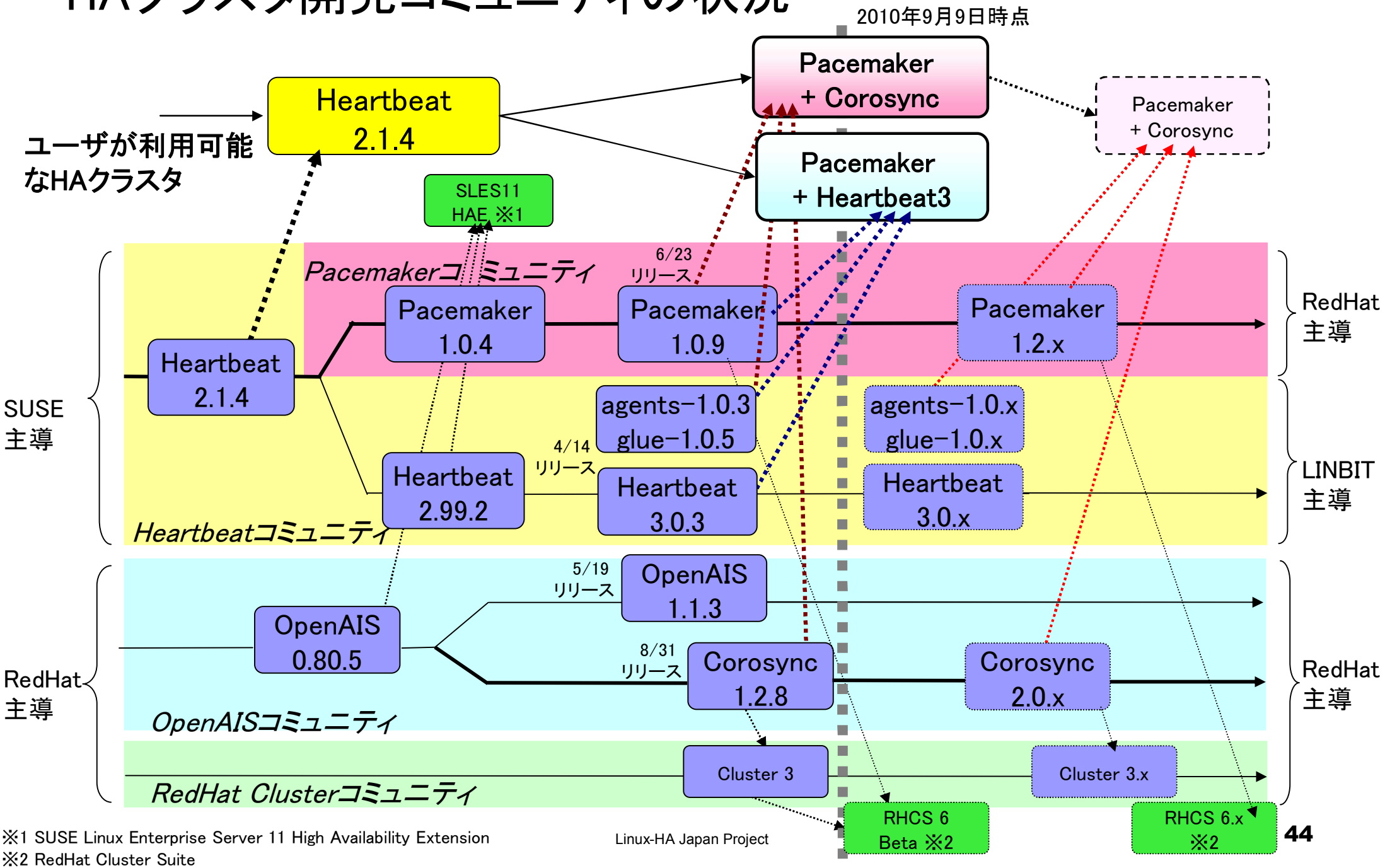
- デメリット
  - まだまだ開発途上である
    - corosync-1系は頻繁にバグフィックス版がリリースされている

Heartbeat3 のメリット・デメリット

クラスタ制御部

- メリット
  - Heartbeat2系のクラスタ制御部のため、これまでの使用方法ならば実績と安定性がある
- デメリット
  - 多ノード構成に向いていない
    - 6+1構成くらいが限界
  - スプリットブレイン回復時の動作が不安定
    - スプリットブレイン回復時のクラスタ復旧手順がやや複雑
  - オンラインによるノード追加・削除時の動作が不安定である

HAクラスタ開発コミュニティの状況



※1 SUSE Linux Enterprise Server 11 High Availability Extension
※2 RedHat Cluster Suite

③

Pacemakerでクラスタリングに挑戦しよう！

まずは
Pacemaker の
インストール方法に挑戦！

Pacemaker rpmパッケージ一覧

CentOS5.5(x86_64)に、HAクラスタを構築する場合の、rpmパッケージ一覧です。
2010年9月9日現在で公開されている最新rpmのバージョンです。

- pacemaker-1.0.9.1-1.15.el5.x86_64.rpm
- pacemaker-libs-1.0.9.1-1.15.el5.x86_64.rpm
- corosync-1.2.7-1.1.el5.x86_64.rpm
- corosynclib-1.2.7-1.1.el5.x86_64.rpm
- cluster-glue-1.0.6-1.el5.x86_64.rpm
- cluster-glue-libs-1.0.6-1.el5.x86_64.rpm
- resource-agents-1.0.3-2.6.el5.x86_64.rpm
- heartbeat-3.0.3-2.3.el5.x86_64.rpm
- heartbeat-libs-3.0.3-2.3.el5.x86_64.rpm

Corosync、Heartbeat3どちらのクラスタ制御部を使用する場合でも、インストールするrpmパッケージは同じです

こーんなに沢山のrpmを
ダウンロード＆インストール
するのは大変・・・

さらに
パッケージの依存関係も
よくわからん・・・

と思い、インストールに
挫折しそうになるでしょうが・・・

CentOS5系（RHEL5系）ならば
yumを使えば
インストールは簡単！

CentOS5.5(x86_64)の場合のPacemakerインストール方法 その1

（ネットワーク接続環境があるのが前提です）

- epel の yumリポジトリを設定

  download.fedora.redhat.com から epel-release の rpmファイルをダウンロードしてインストールします。

```
# wget http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-3.noarch.rpm
# rpm -ivh epel-release-5-3.noarch.rpm
```

clusterlabs.org の yumリポジトリを設定

clusterlabs.org からrepoファイルをダウンロードして、yumリポジトリを設定します。

```
# cd /etc/yum.repo.d
# wget http://clusterlabs.org/rpm/epel-5/clusterlabs.repo
```

※ clusterlabs.repoの内容

```
name=High Availability/Clustering server technologies (epel-5)
baseurl=http://www.clusterlabs.org/rpm/epel-5
type=rpm-md
gpgcheck=0
enabled=1
```

yumで簡単インストール！

これだけでインストールは完成！

```
# yum install pacemaker.x86_64
```

rpmの依存関係で以下のパッケージもネットワークからダウンロードして自動的にインストールされます。

- pacemaker-libs (clusterlabs)
- corosync (clusterlabs)
- corosynclib (clusterlabs)
- cluster-glue (clusterlabs)
- cluster-glue-libs (clusterlabs)
- resource-agents (clusterlabs)
- heartbeat (clusterlabs)
- heartbeat-libs (clusterlabs)
- libesmtp (epel)

CentOS5.5(x86_64)の場合の Pacemakerインストール方法 その2

- Pacemaker リポジトリパッケージをダウンロード

  Linux-HA Japanプロジェクト から提供する Pacemaker リポジトリパッケージを sourceforge.jp からダウンロードします。

Pacemaker リポジトリパッケージを展開

sourceforge.jp からダウンロードしたリポジトリパッケージを
/tmp 等のディレクトリで展開します。

```
# cd /tmp
# tar zxvf pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo.tar.gz
                :
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/rpm/heartbeat-3.0.3-2.3.el5.x86_64.rpm
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/rpm/libesmtp-1.0.4-5.el5.x86_64.rpm
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/rpm/pacemaker-1.0.9.1-1.15.el5.x86_64.rpm
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/pacemaker.repo
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/repodata/
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/repodata/primary.xml.gz
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/repodata/other.xml.gz
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/repodata/filelists.xml.gz
pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/repodata/repomd.xml
```

インストールするRPMファイルと
repoファイルが展開されます

ローカルyumリポジトリを設定

展開したrepoファイルをローカルyumリポジトリとして設定します。

```
# cd /tmp/pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/
# vi pacemaker.repo
```

```
[pacemaker]
name=pacemaker
baseurl=file:///tmp/pacemaker-1.0.9.1-1.15.1.el5.x86_64.repo/
gpgcheck=0
enabled=1
```

パッケージを展開したディレクトリを指定
（デフォルトは /tmp ）

- repoファイルを指定して、その1と同様にyumで簡単インストール！

```
# yum –c pacemaker.repo install pacemaker
```

rpmの依存関係で以下のパッケージも /tmp等に展開したディレクトリから自動的にインストールされます。

- pacemaker-libs (pacemaker)
- corosync (pacemaker)
- corosynclib (pacemaker)
- cluster-glue (pacemaker)
- cluster-glue-libs (pacemaker)
- resource-agents (pacemaker)
- heartbeat (pacemaker)
- heartbeat-libs (pacemaker)
- libesmtp (pacemaker)

- リポジトリパッケージから、Linux-HA Japanプロジェクトオリジナルの Pacemaker追加パッケージも同時にインスール可能になる予定です

```
# yum –c pacemaker.repo install pacemaker pm_diskd
```

ディスク監視機能 pm_diskd（予定）も同時にインストールする場合の例です

Pacemaker-1.0.10 リリース時（2010年10月予定）には、Linux-HA Japanプロジェクトオリジナルのディスク監視機能もリポジトリパッケージに入る予定です。

VineSeedの場合のPacemakerインストール方法

2010年7月にVineLinuxの開発版VineSeedにPacemakerを入れてもらいました。
（VineProject様 ありがとうございます！）

- apt-get で一発簡単インストール！

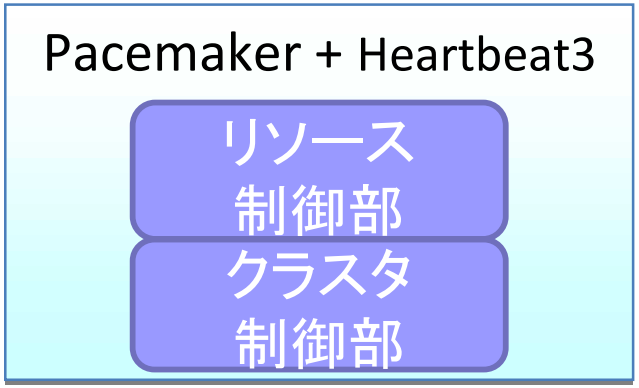
```
# apt-get install pacemaker
```

Pacemakerの設定に挑戦！

Pacemaker では

「クラスタ制御部」「リソース制御部」

それぞれの設定が必要です。



クラスタ制御部の設定（Heartbeat3）

クラスタ
制御部

- /etc/ha.d/ha.cf
  - クラスタの基本的な動作情報
  - クラスタ内の全ノードに同じ内容のファイルを配置

```
pacemaker on

debug 0
udpport 694
keepalive 2
warntime 20
deadtime 24
initdead 48
logfacility local1

bcast eth2
bcast eth3

node pm01
node pm02

watchdog /dev/watchdog
```

基本的に Heartbeat 2.x の
ha.cfと設定は同じです

従来の「crm on」から「pacemaker on」に
変更となります

ロギングには logd ではなく syslog を使用
するため、ログファシリティを設定します

- /etc/ha.d/authkeys
  - クラスタを構成する認証キーを保持するファイル
  - 認証キーが同じノード群でクラスタを構成
  - クラスタ内の全ノードに、同じ内容のファイルを配置
  - 権限・ユーザ/グループは、600・root/root に設定

```
auth 1
1 sha1 hogehoge
```

これも基本的にHeartbeat 2.xと設定は同じです

認証キー：任意の文字列

認証キーの計算方法：sha1, md5, crcを指定可

- /etc/syslog.conf
  - /etc/ha.d/ha.cf で指定したファシリティの設定が必要

/var/log/ha-log にログを出力するように設定します。
また、同内容のログを /var/log/messages に2重出力しないように、「local1.none」の追記も行います。

```
*.info;mail.none;authpriv.none;cron.none;local1.none      /var/log/messages
                :
             (省略)
                :
local1.*                                                   /var/log/ha-log
```

これでとりあえずは Pacemakerが起動します！

```
# service heartbeat start
Starting High-Availability services:                    [  OK  ]
```

起動はクラスタ制御部である
heartbeatを各ノードで起動します

起動状態の確認

Pacemakerのコマンド /usr/sbin/crm_mon を利用して起動状態が確認できます。

```
# crm_mon
============
Last updated: Thu Sep  9 20:24:49 2010
Stack: Heartbeat
Current DC: pm02 (fe705a39-541a-4b10-af22-de27d4c72d23) - partition
with quorum
Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677
2 Nodes configured, unknown expected votes
0 Resources configured.
============

Online: [ pm02 pm01 ]
```

クラスタに組み込まれている
ノード名が表示されます

しかしこれだけでは、
リソース制御部の設定が無いので
リソースは
なーんにも起動していません…

リソース制御部の設定

リソース
制御部

- リソース制御部には次のような設定が必要です。
  - どのようなリソースをどのように扱うか
    Apache、PostgreSQLなど、どのリソース（アプリケーション）を起動するか？
  - 起動、監視、停止時に関連する時間
    リソースの監視（monitor）間隔は何秒にするか？？
  - リソースの配置 などを指定
    リソースをどのノードで起動するか？？？

- 設定方法には主に2通りあります。
  - cib.xml にXML形式で設定を記述
    - 従来のHeartbeat 2.x での方法
  - crmコマンドで設定
    - Pacemakerからの新機能

まずは XML形式に挑戦！

cib.xml

- /var/lib/heartbeat/crm/cib.xml

リソースの定義等を設定するXMLファイルを作成します。

```
(..略..)
<primitive class="ocf" id="prmIp" provider="heartbeat" type="IPaddr2">
  <instance_attributes id="prmIp-instance_attributes">
    <nvpair id="prmIp-instance_attributes-ip" name="ip" value="192.168.0.108"/>
    <nvpair id="prmIp-instance_attributes-nic" name="nic" value="eth1"/>
    <nvpair id="prmIp-instance_attributes-cidr_netmask" name="cidr_netmask"
value="24"/>
  </instance_attributes>
  <operations>
    <op id="prmIp-start-0s" interval="0s" name="start" on-fail="restart" timeout="60s"/>
    <op id="prmIp-monitor-10s" interval="10s" name="monitor" on-fail="restart"
timeout="60s"/>
    <op id="prmIp-stop-0s" interval="0s" name="stop" on-fail="block" timeout="60s"/>
  </operations>
</primitive>
(..略..)
```

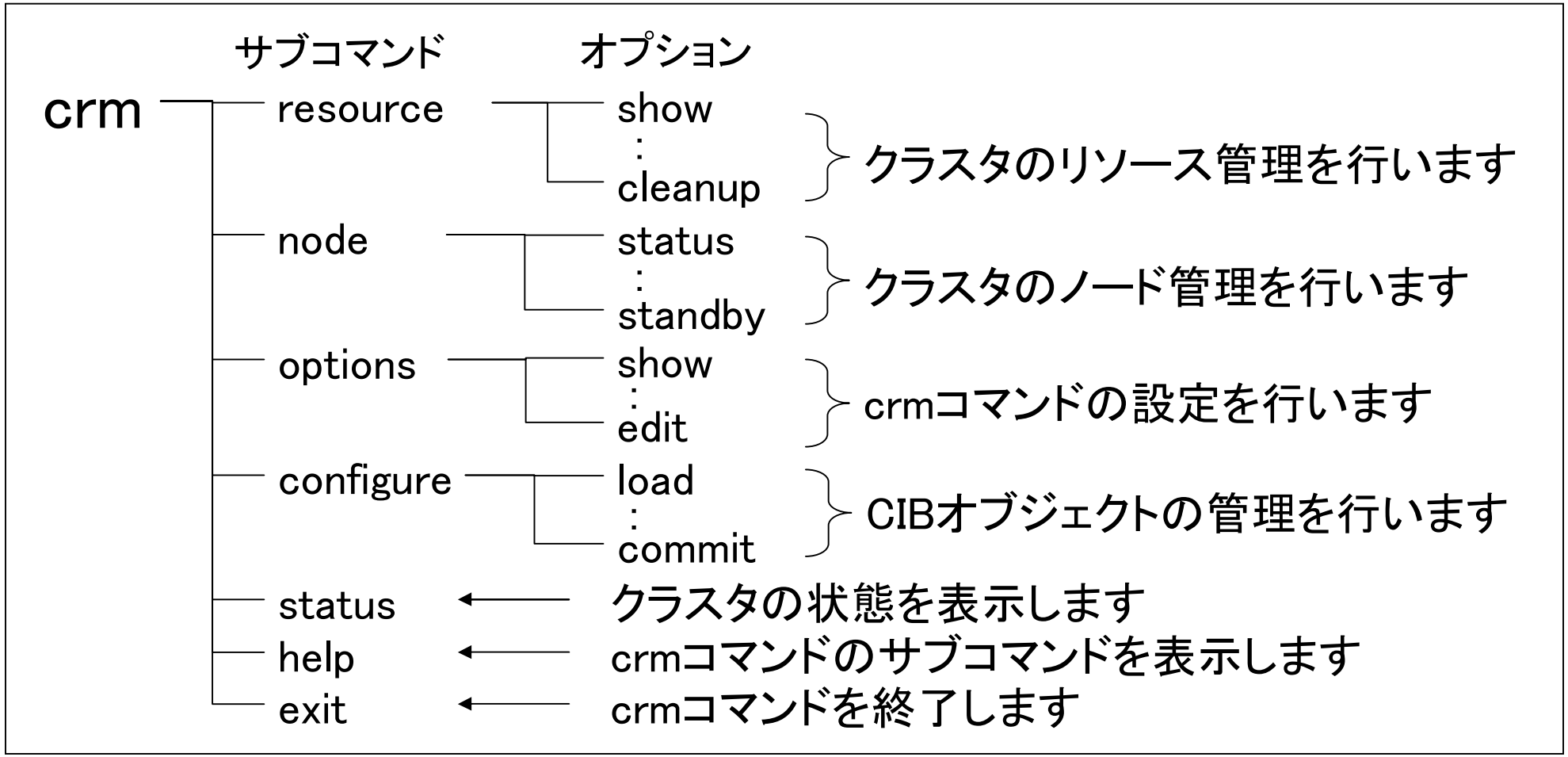
XMLの記法を知る
必要があり難しい…

Heartbeatバージョン2を
使おうとして、
このXMLで挫折した人は
多いはずです…

Pacemaker での新機能
crmコマンドに挑戦！

crmコマンド

crmコマンドは、クラスタ状態を管理するためのコマンドラインインターフェイスです。



crmコマンドで、
仮想IPアドレス設定と
Apacheを起動する
クラスタ設定に挑戦！

crmコマンド実行例

- crmコマンドを起動し、リソース設定モードに入ります

```
# crm
crm(live)# configure
```

- Pacemakerには「STONITH」という強制電源断機能がありますが、ここでは使用しない設定を行います

```
crm(live)configure# property no-quorum-policy="ignore" ¥
        stonith-enabled="false" ¥
        startup-fencing="false"
```

- 「IPaddr2」リソースエージェントを使用し、仮想IPアドレスのリソース設定を行います

仮想IP設定のリソースIDを「**prmlp**」とします

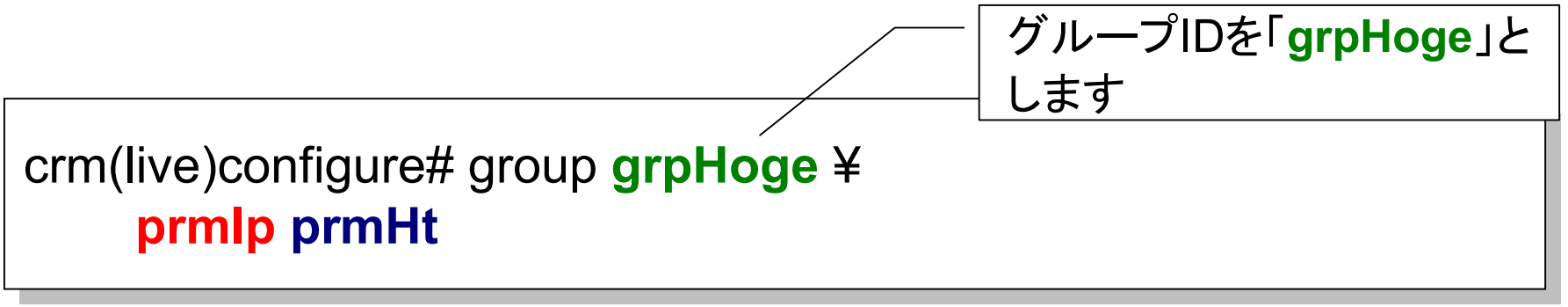
```
crm(live)configure# primitive prmlp ocf:heartbeat:IPaddr2 ¥
        params ¥
                ip="192.168.0.108" ¥
                nic="eth1" ¥
                cidr_netmask="24" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="block"
```

- 続けて「apache」リソースエージェントを使用し、Apacheのリソース設定を行います

Apache設定のリソースIDを「**prmHt**」とします

```
crm(live)configure# primitive prmHt ocf:heartbeat:apache ¥
        params ¥
                statusurl="http://localhost/test.html" ¥
                testregex="hogehoge" ¥
                httpd="/usr/sbin/httpd" ¥
                configfile="/etc/httpd/conf/httpd.conf" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="block“
```

- 設定した「IPaddr2」「apache」の2つのリソースをグルーピングする設定を行います



- ここまで設定リソース・リソースグループを確認します

```
crm(live)configure# show
node $id="a0dacbcf-346f-4003-ab5b-15422e0e4697" pm01
node $id="fe705a39-541a-4b10-af22-de27d4c72d23" pm02
primitive prmHt ocf:heartbeat:apache ¥
        params statusurl="http://localhost/test.html" testregex="hogehoge"
httpd="/usr/sbin/httpd" configfile="/etc/httpd/conf/httpd.conf" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="block"
primitive prmIp ocf:heartbeat:IPaddr2 ¥
        params ip="192.168.0.108" nic="eth1" cidr_netmask="24" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="block"
group grpHoge prmIp prmHt
property $id="cib-bootstrap-options" ¥
        dc-version="1.0.9-89bd754939df5150de7cd76835f98fe90851b677" ¥
        cluster-infrastructure="Heartbeat" ¥
        no-quorum-policy="ignore" ¥
        stonith-enabled="false" ¥
        startup-fencing="false“
```

- コミットを実行するとリソースが起動されます

```
crm(live)configure# commit
```

コミットされると、cib.xmlに反映されてリソースが起動されます。
つまりリソース設定の根っこは、どちらにしろ cib.xml なのです。

というように、
crmコマンドを熟知すれば、
自由自在に
クラスタリングできます！

しかしこの例では、
リソース配置制約等の設定が
まだおこなわれていないなど、

まだまだ、
リソース制御部設定完了までの
イバラの道は続きます・・・

が、しかし、
crmコマンドがわからなくても
まとめて設定できる
簡単ツールを紹介します！

crmは恐くない！

- 複雑なリソース制御の設定も crmファイル編集ツール pm_crmgenで解決！

pm_crmgenを使用すれば、テンプレートExcelファイルから簡単にリソース制御部を設定する事が可能です。

Linux-HA Japanプロジェクトで
crmファイル編集ツールを開発中！

開発版は、Linux-HA Japanプロジェクトのリポジトリよりダウンロード可能です。
http://hg.sourceforge.jp/view/linux-ha/

crmファイル編集ツールで簡単設定！

※ 9/9 時点での開発版での状況です

① pm_crmgenをインストール

rpmパッケージ名は予定名です。プログラム自体は pythonで作成されています。

rpmコマンドでインストールする場合

```
# rpm –ivh pm_crmgen-1.0.noarch.rpm
```

sourceforge.jpから配布予定のPacemakerリポジトリパッケージに含めるため、yumコマンドでもインストール可能になる予定です

```
# yum –c pacemaker.repo pm_crmgen
```

② テンプレートExcelファイルにリソース定義を記載

青枠線の中に値を記入します。
仮想IPをActiveノードに付与する場合の例です。

53	#表 5-2 クラスタ設定 … Primitiveリソース					
54	PRIMITIVE					
55	P	id	class	provider	type	
56	#	リソースID	class	provider	type	
57		prmIp	ocf	heartbeat	IPaddr2	
58	A	type	name	value		
59	#	パラメータ種別	項目	設定内容		
60		params	ip	192.168.0.108		
61			nic	eth1		
62			cidr_netmask	24		
63	O	type	timeout	interval	on-fail	
64	#	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)	備考
65		start	60s	0s	restart	
66		monitor	60s	10s	restart	
67		stop	60s	0s		

「IPaddr2」のリソースエージェントを使用

付与する仮想IPのIPアドレス等を入力

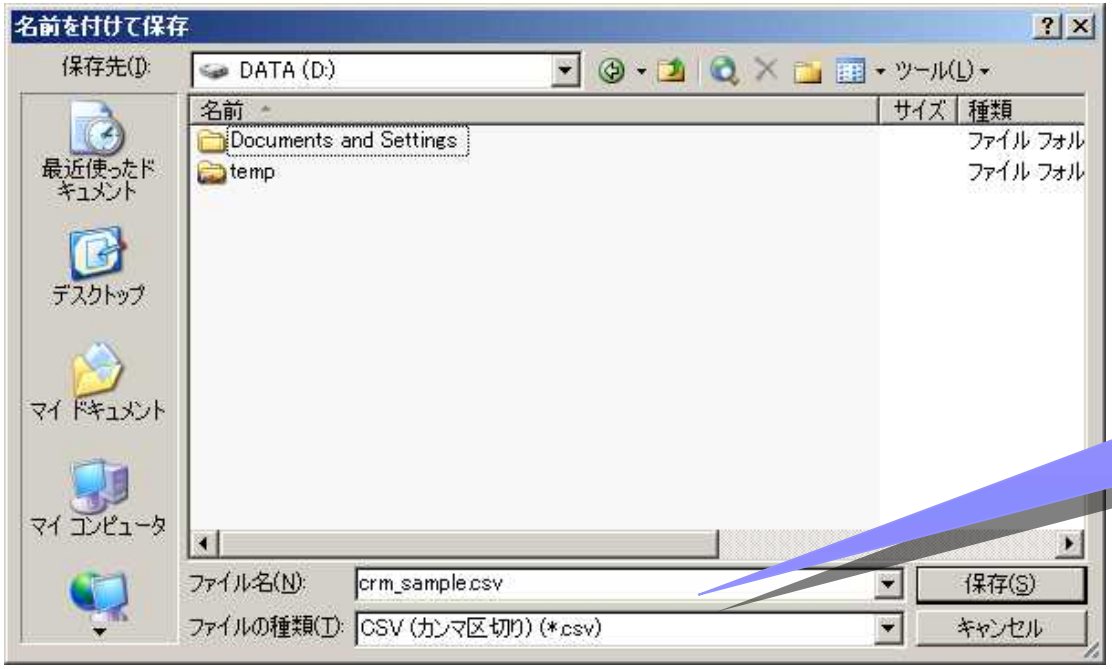
監視間隔などを入力

リソースグループをどのノードで起動させるかのリソース配置制約の設定も可能です。

```
#表 6-1 クラスタ設定 --- リソース配置制約
LOCATION
rsc             score:200   score:100   score:-inf   p
#  リソースID   Activeノード  Standbyノード  非稼働ノード  -
   grpHoge      pm01        pm02                     y
```

ActiveとStandbyノードを指定

③ CSV形式でファイルを保存



「crm_sample.csv」としてCSV形式で保存

④ CSVファイルをノードへ転送

CSVファイル保存後、SCPコマンド等でActive系ノードへ転送
→ Active系、Standby系どちらか片方のノードに転送すればOK！

⑤ pm_crmgenコマンドでcrmファイルを生成

```
# pm_crmgen –o crm_sample.crm  crm_sample.csv
```

③で転送したCSVファイル

生成するcrmファイル名

出来上がった crmファイル例

```
(..略..)

### Primitive Configuration ###
primitive prmIp ocf:heartbeat:IPaddr2 ¥
        params ¥
                ip="192.168.0.108" ¥
                nic="eth1" ¥
                cidr_netmask="24" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="block“

(..略..)
```

Excelファイルで記述した
仮想IPを設定する
crmサブコマンドが
ファイルに記述されます

⑥ crmコマンドを実行してリソース設定を反映

```
# crm
crm(live)# configure
crm(live)configure# load update crm_sample.crm
crm(live)configure# commit
```

⑤で生成したcrmファイル名

commitで設定が反映される

または以下のようにcrmコマンド一発で反映も可能です。

```
# crm configure load update crm_sample.crm
```

これでリソースも起動しました！

/usr/sbin/crm_mon を利用して起動したリソースが確認できます。

```
============
Last updated: Thu Sep  9 21:13:40 2010
Stack: Heartbeat
Current DC: pm02 (fe705a39-541a-4b10-af22-de27d4c72d23) - partition
with quorum
Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ pm02 pm01 ]

Resource Group: grpHoge
    prmIp      (ocf::heartbeat:IPaddr2):       Started pm01
    prmHt      (ocf::heartbeat:apache):        Started pm01
 Clone Set: clnPingd
    Started: [ pm02 pm01 ]
```

ノード1でリソースグループが起動されました。（仮想IPが付与されてApacheが起動）

もしノード故障が発生すると・・・

```
============
Last updated: Thu Sep  9 21:15:27 2010
Stack: Heartbeat
Current DC: pm02 (fe705a39-541a-4b10-af22-de27d4c72d23) - partition with
quorum
Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ pm02 ]
OFFLINE: [ pm01 ]

 Resource Group: grpHoge
     prmIp      (ocf::heartbeat:IPaddr2):       Started pm02
     prmHt      (ocf::heartbeat:apache):        Started pm02
 Clone Set: clnPingd
     Started: [ pm02 ]
     Stopped: [ prmPingd:1 ]
```

ノード2からはノード1が見えなくなったので「OFFLINE」と表示されます

フェイルオーバしてノード2でリソースグループが起動されます

もしリソース故障が発生すると・・・

```
============
Last updated: Thu Sep  9 21:41:18 2010
Stack: Heartbeat
Current DC: pm01 (a0dacbcf-346f-4003-ab5b-15422e0e4697) - partition with
quorum
Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ pm02 pm01 ]

 Resource Group: grpHoge
     prmIp      (ocf::heartbeat:IPaddr2):       Started pm02
     prmHt      (ocf::heartbeat:apache):        Started pm02
 Clone Set: clnPingd
     Started: [ pm02 pm01 ]

Failed actions:
    prmHt_monitor_10000 (node=pm01, call=10, rc=7, status=complete): not running
```

フェイルオーバしてノード2でリソースグループが起動されます

リソース故障状況が表示されます
※ ノード1でprmHt(Apache)が故障中です。

④

Linux-HA Japan プロジェクトについて

Linux-HA Japan プロジェクトの経緯

『Heartbeat(ハートビート)』の日本における更なる普及展開を目的として、2007年10月5日「Linux-HA (Heartbeat) 日本語サイト」を設立しました。

その後、日本でのLinux-HAコミュニティ活動として、Heartbeat-2.x のrpmバイナリと、Heartbeat機能追加パッケージを提供しています。

Pacemaker の
情報やパッケージも
Linux-HA Japanプロジェクトから
提供中です。

Linux-HA JapanプロジェクトURL

http://linux-ha.sourceforge.jp/

Pacemaker関連情報の
公開用として SourceForge.jp に
新しいウェブサイトが 6/25に
オープンしました。

これから随時情報を更新していきます！

Linux-HA Japan開発者向けサイト

http://sourceforge.jp/projects/linux-ha/

Pacemakerリポジトリパッケージが公開されています。

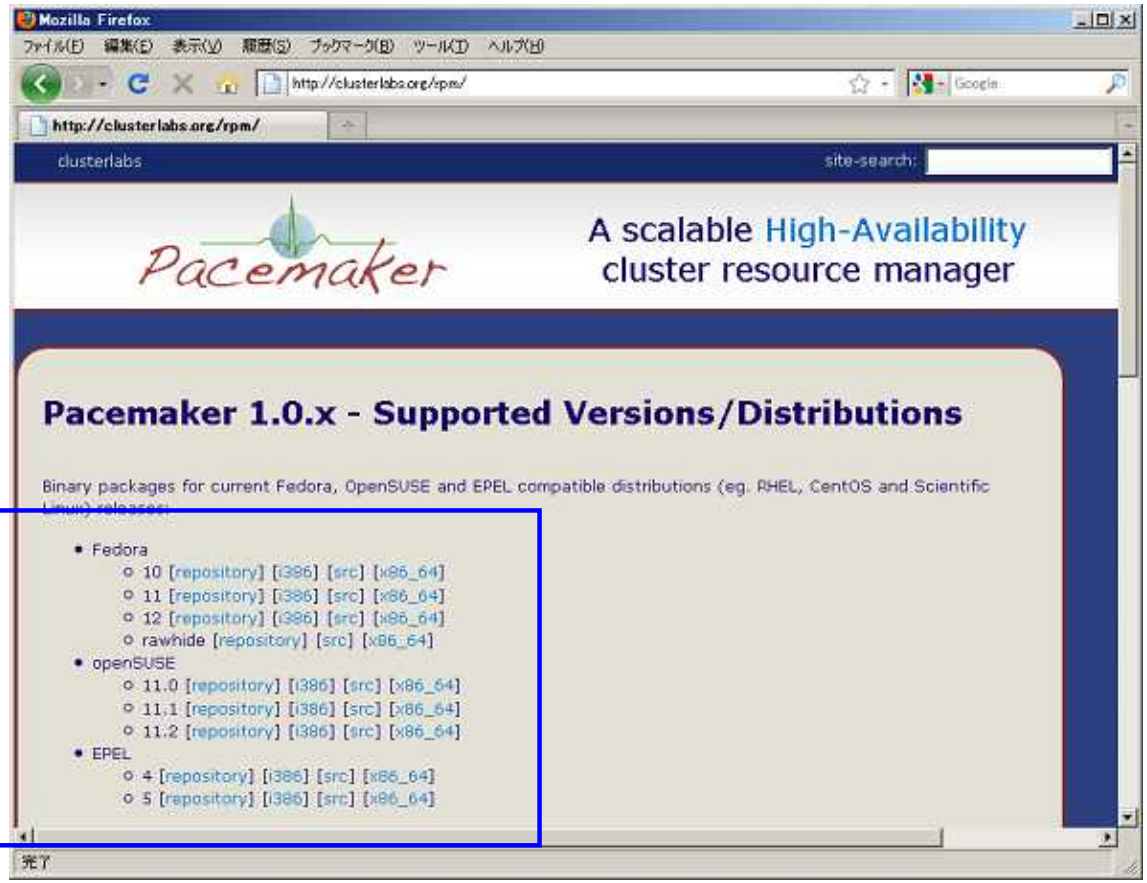
pm_crmgenなどのPacemaker追加パッケージの開発ソースコードも参照可能です。

RHEL/CentOS用 Heartbeat-2.x rpmバイナリの提供や、機能追加パッケージ類も、GPLライセンスにて公開しています。

clusterlabs.org
本家Pacemakerサイト

http://clusterlabs.org/

Fedora, openSUSE, EPEL(CentOS/RHEL)のrpmがダウンロード可能です。



実は
本家Pacemakerのロゴは
これ 

です。

しかし

これ 

では、

いかにも医療機器なので…

Pacemakerロゴ

Linux-HA Japan プロジェクトでは、
Pacemakerのロゴを作成しました。

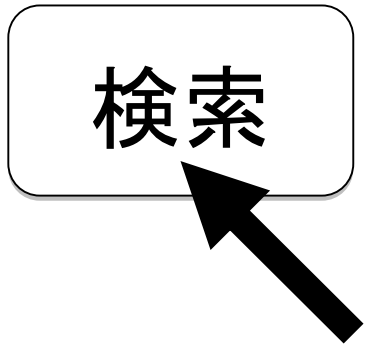
Linux-HA Japanメーリングリスト

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
その他DRBDなど、HAクラスタに関連する話題は全て歓迎します！

- ML登録用URL
  http://lists.sourceforge.jp/mailman/listinfo/linux-ha-japan
- MLアドレス
  linux-ha-japan@lists.sourceforge.jp

Linux-HA Japan



http://linux-ha.sourceforge.jp/

ご清聴ありがとうございました。