Pacemakerって何？
～あなたもできる、簡単高信頼システムの構築～

2011年2月5日　OSC2011 Kagawa

Linux-HA Japan
三井 一能

HighAvailability Pacemaker

本日のお話

① HAクラスタって何？Pacemakerって何？
② システム基本構成
③ Pacemakerの基本動作
④ Pacemakerのコンポーネント構成
⑤ Pacemakerでクラスタリングに挑戦！
⑥ Linux-HA Japan について
⑦ 参考情報

①

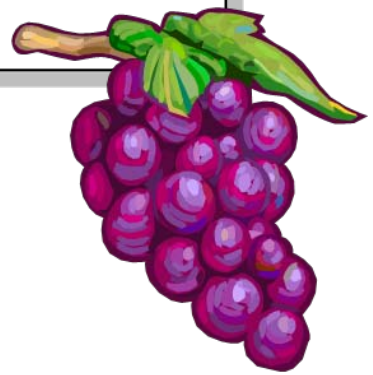
HAクラスタって何？
Pacemakerって何？

そもそもクラスタって何？

コンピュータの世界で

クラスタというと

複数のコンピュータを結合し、
果実・花などの房のように
ひとまとまりとしたシステムのこと
(Wikipediaより)



HAクラスタ
HPCクラスタ
負荷分散クラスタ　　・・・・

さらに**HAクラスタ**っていうと・・

High Availability = 高可用性

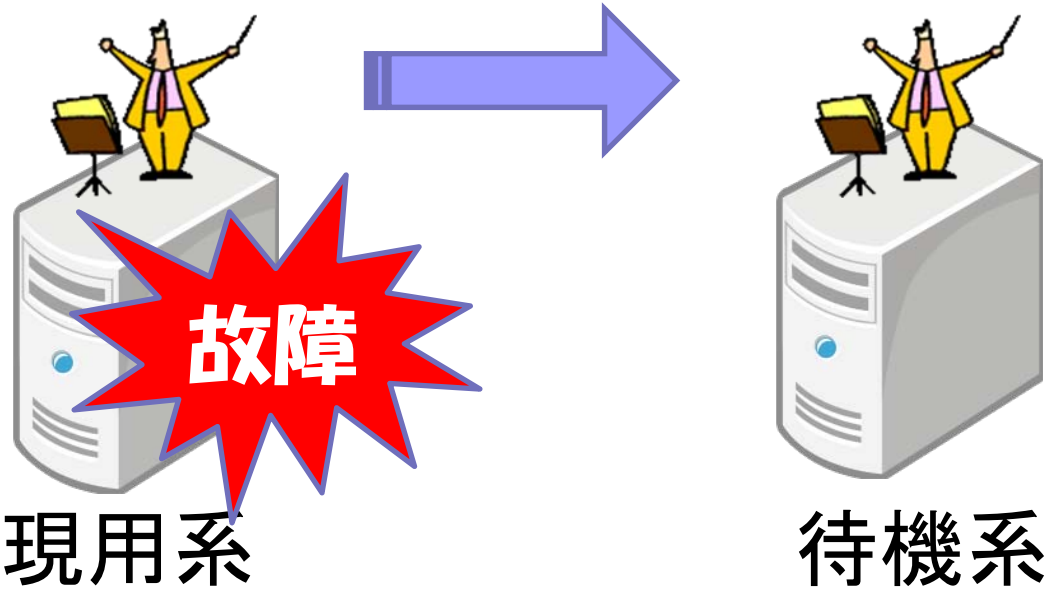
サービス継続性が高い

つまり

一台のコンピュータでは得られない
高い信頼性を狙うために、
複数のコンピュータを結合し、
ひとまとまりとしたシステムのこと

HAクラスタを導入すると、

現用系で**故障**が発生しサービスができなくなったときに、待機系でサービスを**自動起動**し、サービス中断を最小限にすることができます



今回ご紹介する

は
このHAクラスタ
という部類のソフトウェアで、
実績のある「Heartbeat」と
呼ばれていたHAクラスタの
後継ソフトウェアです。

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。

Pacemaker は

ご存知ですか？

同じくHAクラスタである

Heartbeatは

知っていますか？

「Pacemaker」は
「Heartbeat」の
後継というだけあって、
密接な関係があります。
詳細は後ほどお話します。

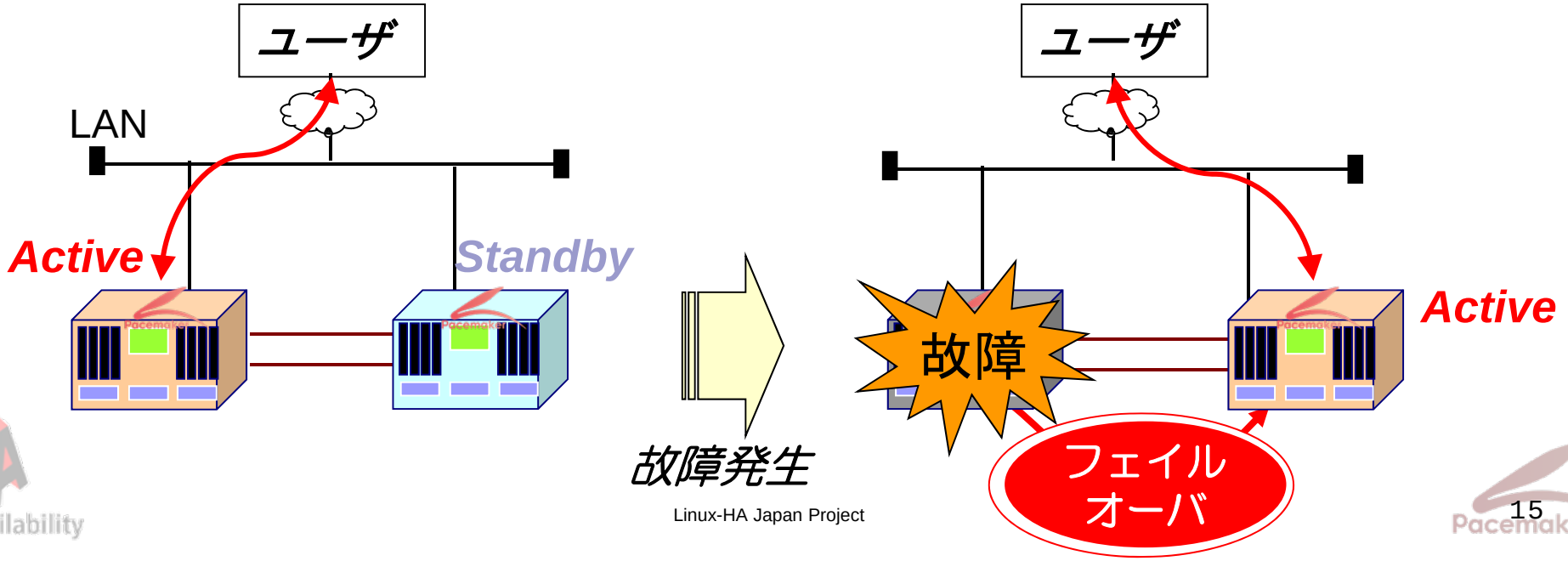
②

システム基本構成

基本構成

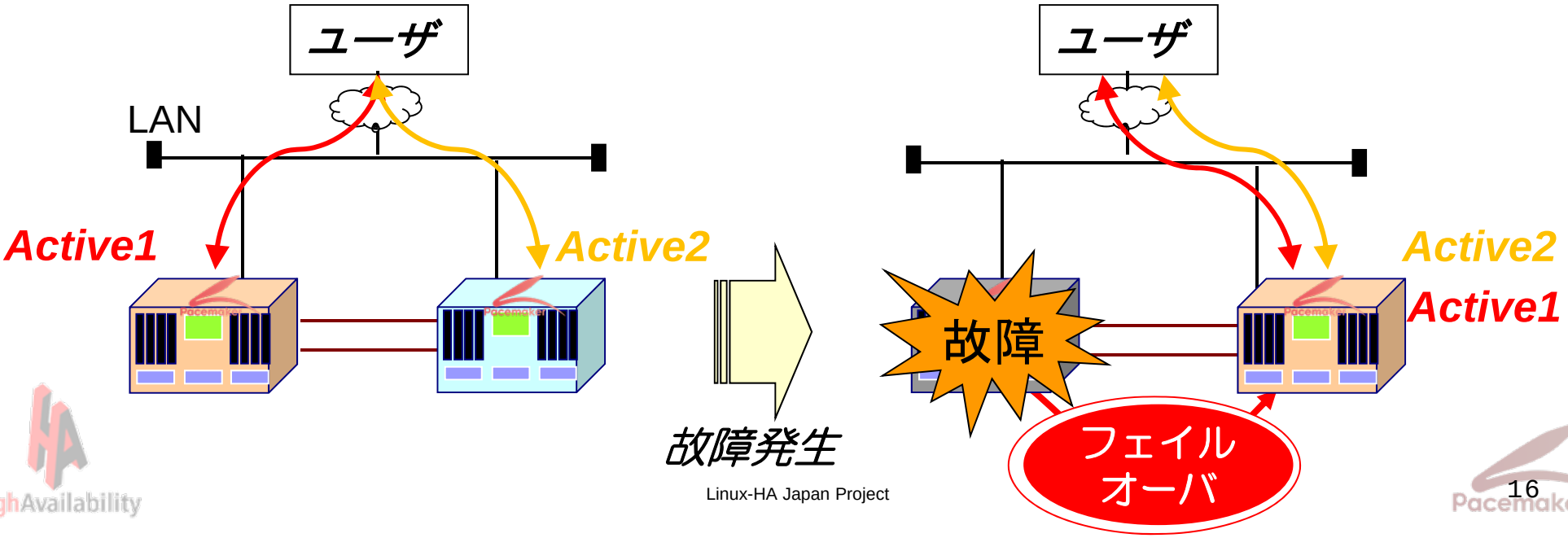
Active/Standby（1+1）構成

- 通常はActiveノードと呼ばれるサーバでサービスを提供します。
- Activeノードが故障した場合は、StandbyノードがActiveになりサービスを引き継ぎます。
  これをフェイルオーバと呼びます。



応用構成
Active/Active（1+1）構成

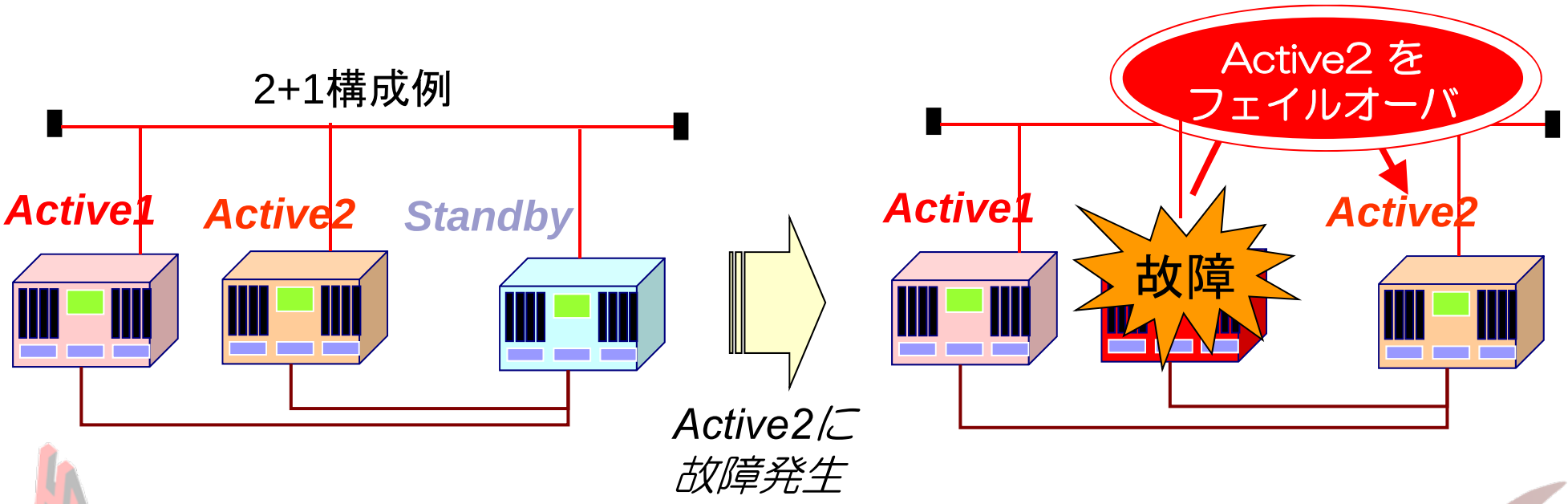
- Active/Standby構成を2セット組み合わせた構成で、両方のサーバでそれぞれサービスが稼働します。
- あるノードが故障した場合は、別のノードにサービスをフェイルオーバします。



Pacemakerでは複数台構成も可能

※ Heartbeatバージョン1では実現できませんでした

- 複数台のActiveノードや、
  複数台のStandbyノードを設定可能です。
  （N+M構成）



今回は、
話を単純にするために、
Active / Standby（1+1構成）
の構成を例に話を進めます。

③

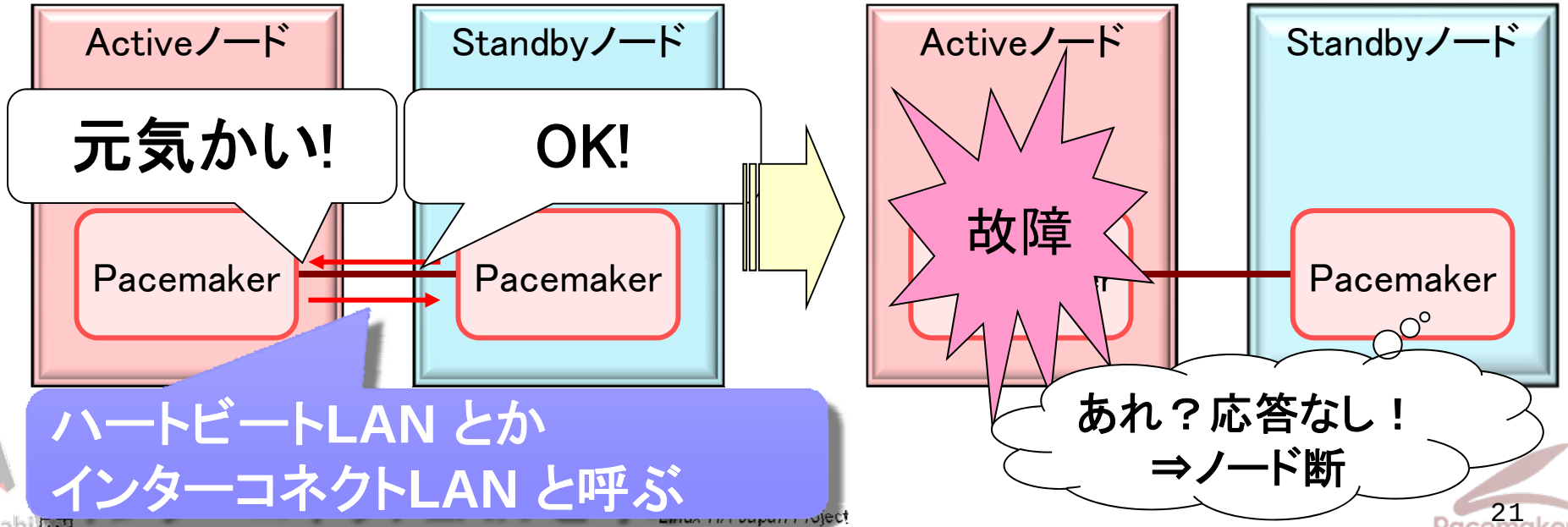
Pacemakerの基本動作

Pacemaker 基本動作は主にこの二つ

1. ノード監視
2. リソース制御

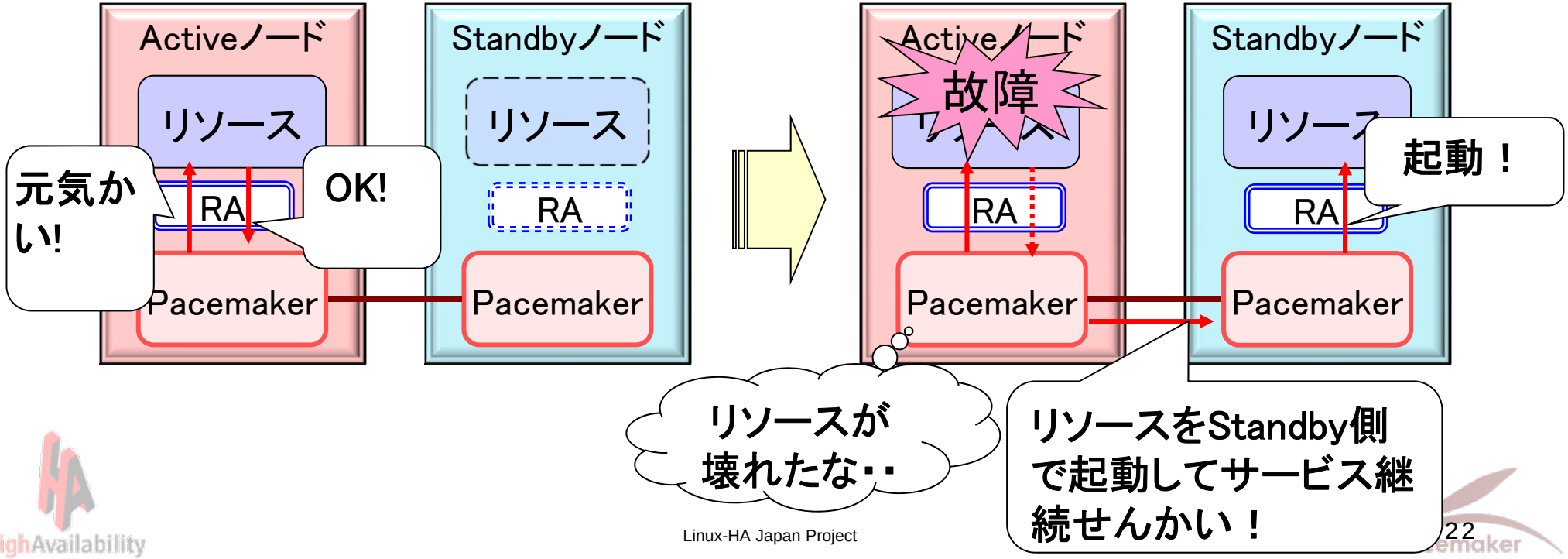
基本動作1：ノード監視

- 相手ノードの生死を確認するために、一定間隔で相手ノードを監視します。（ハートビート通信と呼ぶ）
- 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。



基本動作2:リソース制御

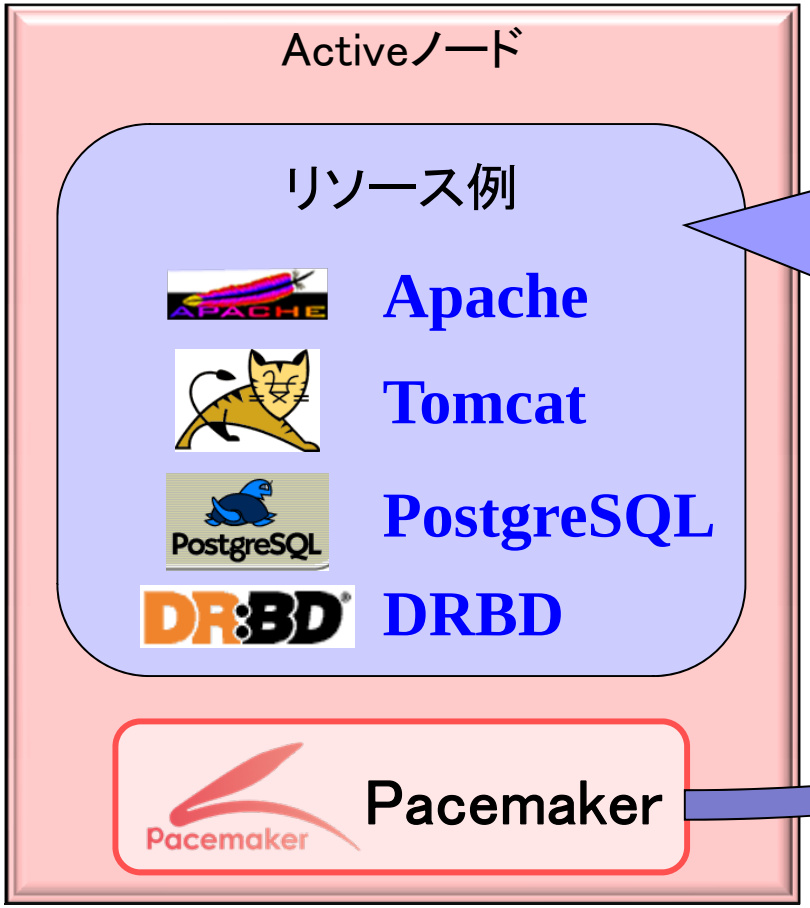
- リソースと呼ばれる物をリソースエージェント(RA)を介して起動(start)、停止(stop)、監視(monitor)します。
- リソースが故障した場合にはフェイルオーバといったリソース制御の処理を行います。



「リソース」って何？

- クラスタが管理するものすべて
  - ■ノード間でサービスを引き継ぐために制御が必要なもの
    - サーバプログラム、コンピュータ資源
  - ■サービスの故障を検知するために監視が必要なもの
    - ネットワーク監視、ディスク監視

簡単に言うと、Pacemakerが起動、停止、監視するものがリソースになります。



Pacemaker から見ると、
PostgreSQL などの
サーバプログラムは、
「リソース」となります。

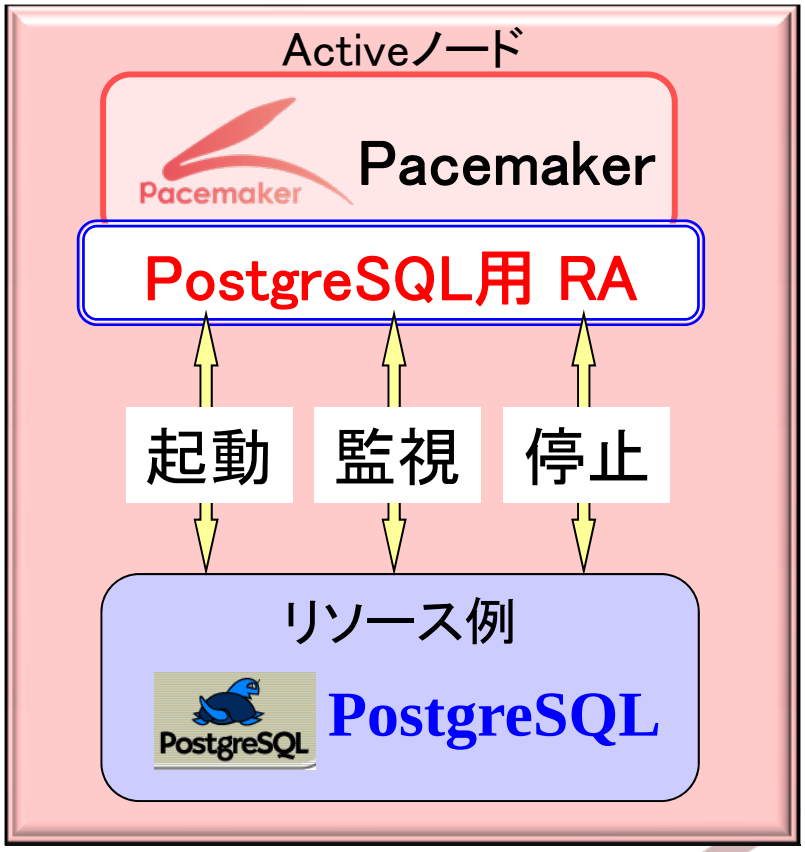
「リソースエージェント(RA)」とは？

リソースとPacemakerを仲介するプログラム

主にシェルスクリプトで記述

Pacemakerは、
リソースエージェントに対し
リソースの
- 起動(start)
- 停止(stop)
- 監視(monitor)

を指示します。



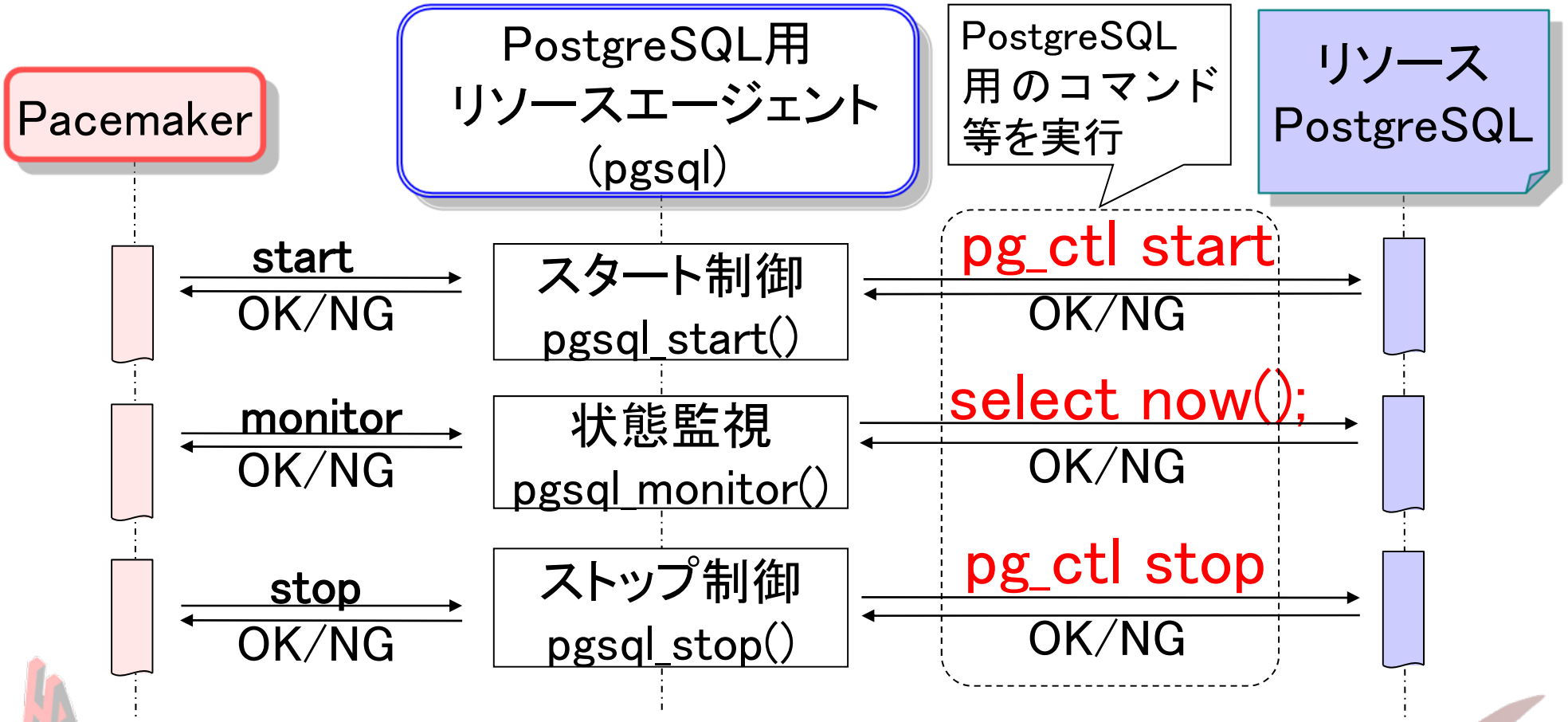
リソースエージェント実装例

PostgreSQL(pgsql RA)監視(monitor)処理の抜粋

```
pgsql_monitor() {                                  ←PostgreSQLの監視のメイン関数
    ......
    if ! pgsql_status                              ←PostgreSQLプロセスの存在を確認
    then                                           ←PostgreSQLプロセスがいなければ
        ocf_log info "PostgreSQL is down"
        return $OCF_NOT_RUNNING                    ←PostgreSQLは停止していると判断
    fi
    ......
   runasowner -q $loglevel  "$OCF_RESKEY_psql $psql_options -c
 '$OCF_RESKEY_monitor_sql' "
            ↑実際にSQL(select now())を実行してPostgreSQLの正常性を確認
    ......
    return $OCF_SUCCESS                            ←PostgreSQLは動作していると判断
}
```

$OCF_SUCCESS, $OCF_NOT_RUNNINGはPacemakerで定義済みの変数

（例）Pacemaker と PostgreSQLリソースエージェントの関係



Pacemakerでは、様々なリソースエージェントが用意されています。

従来の Heartbeat2用に作成されたRAも使用が可能

標準リソースエージェントの一例

目的	リソース	リソースエージェント名 （/usr/lib/ocf/resource.d/ に存在）
サーバプログラム	データベース インターネットサーバ	pgsql, oracle, oralsnr, mysql apache, tomcat, jboss, postfix
コンピュータ資源	ファイルシステム	Filesystem （複数のファイルシステムに対応）
	仮想IPアドレス	IPaddr2, IPv6addr
異常監視	ネットワーク監視 ディスク監視	pingd diskd （Linux-HA Japan提供）

リソースエージェントは自作可能!

```
#!/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で実装できます。

いくつか必須のパラメータ呼び出しに対する処理と、定義済みの戻り値を返すように実装する必要があります。

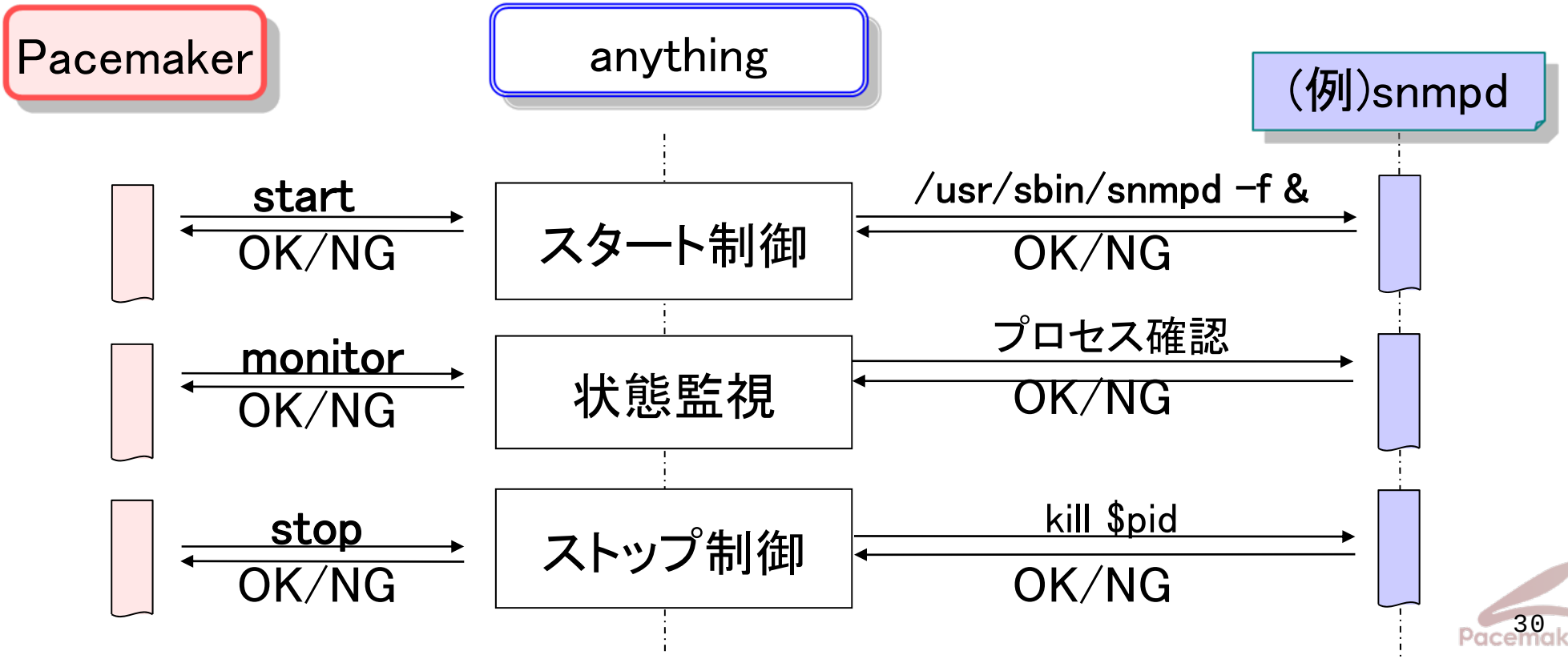
リソース開始・監視・停止の処理

シェルに渡されるパラメータを元にRA処理を振り分け

プログラミングはちょっと、という人のために

汎用RA(**anything**)というものもあります

フォアグラウンドプロセスで動作するプログラムであれば、anything を利用することで、Pacemakerで制御可能になります。

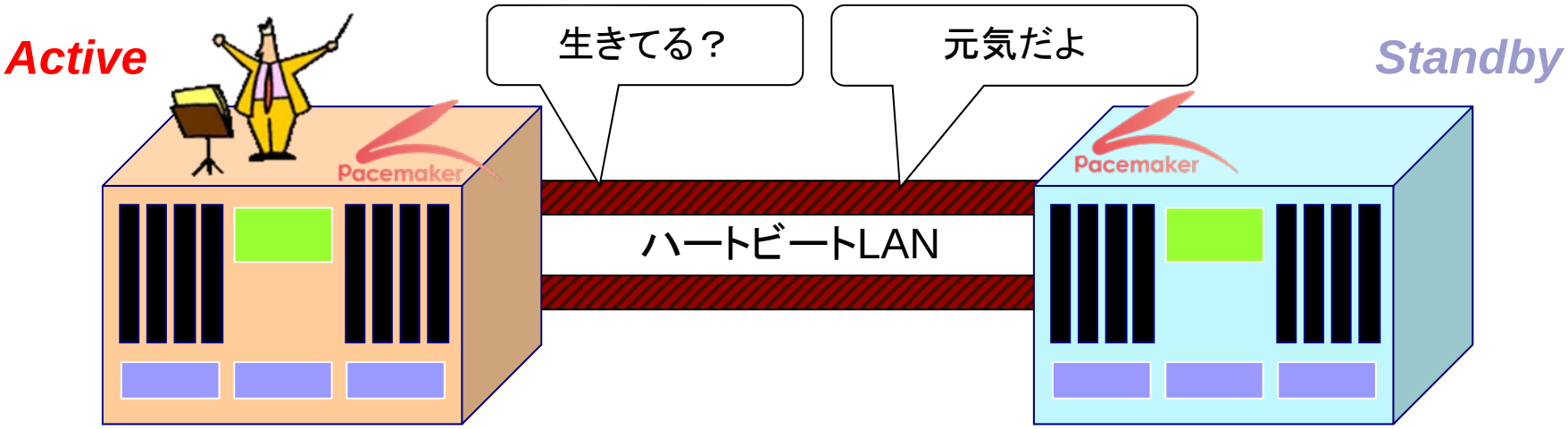


さらに、HAクラスタとして重要な機能

3. スプリットブレイン対策

基本動作3:スプリットブレイン対策

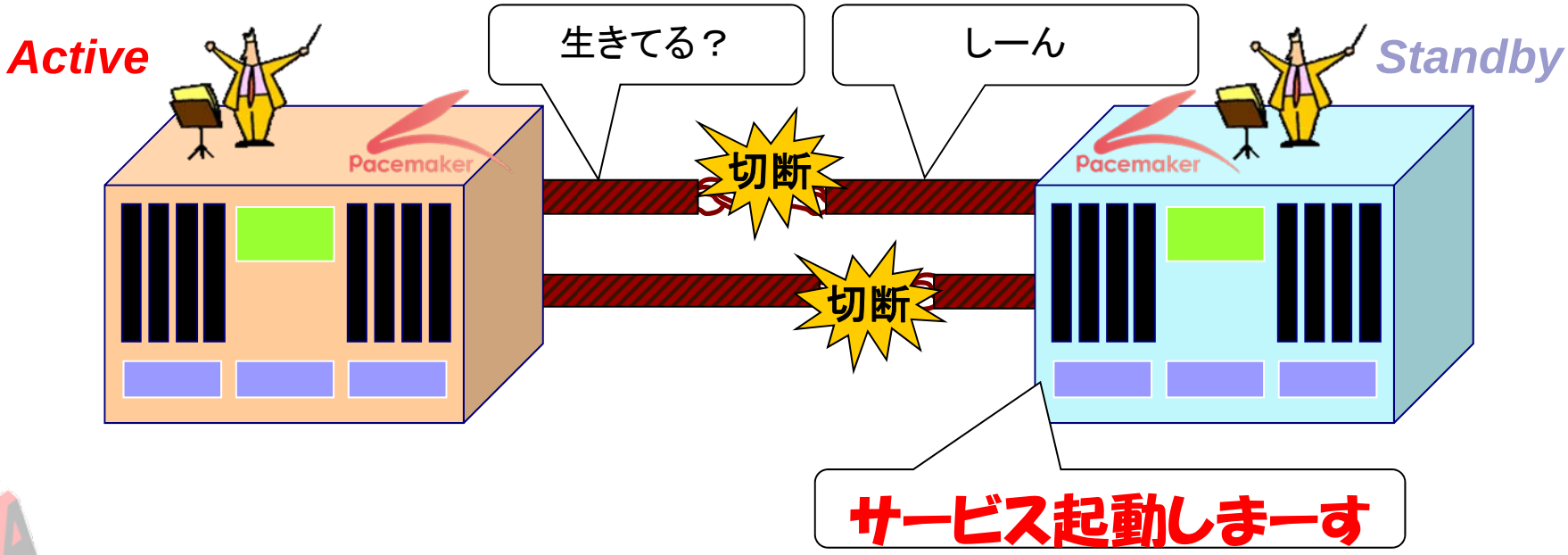
全てのハートビートLANが切れてしまった場合



基本動作3:スプリットブレイン対策

全てのハートビートLANが切れてしまった場合

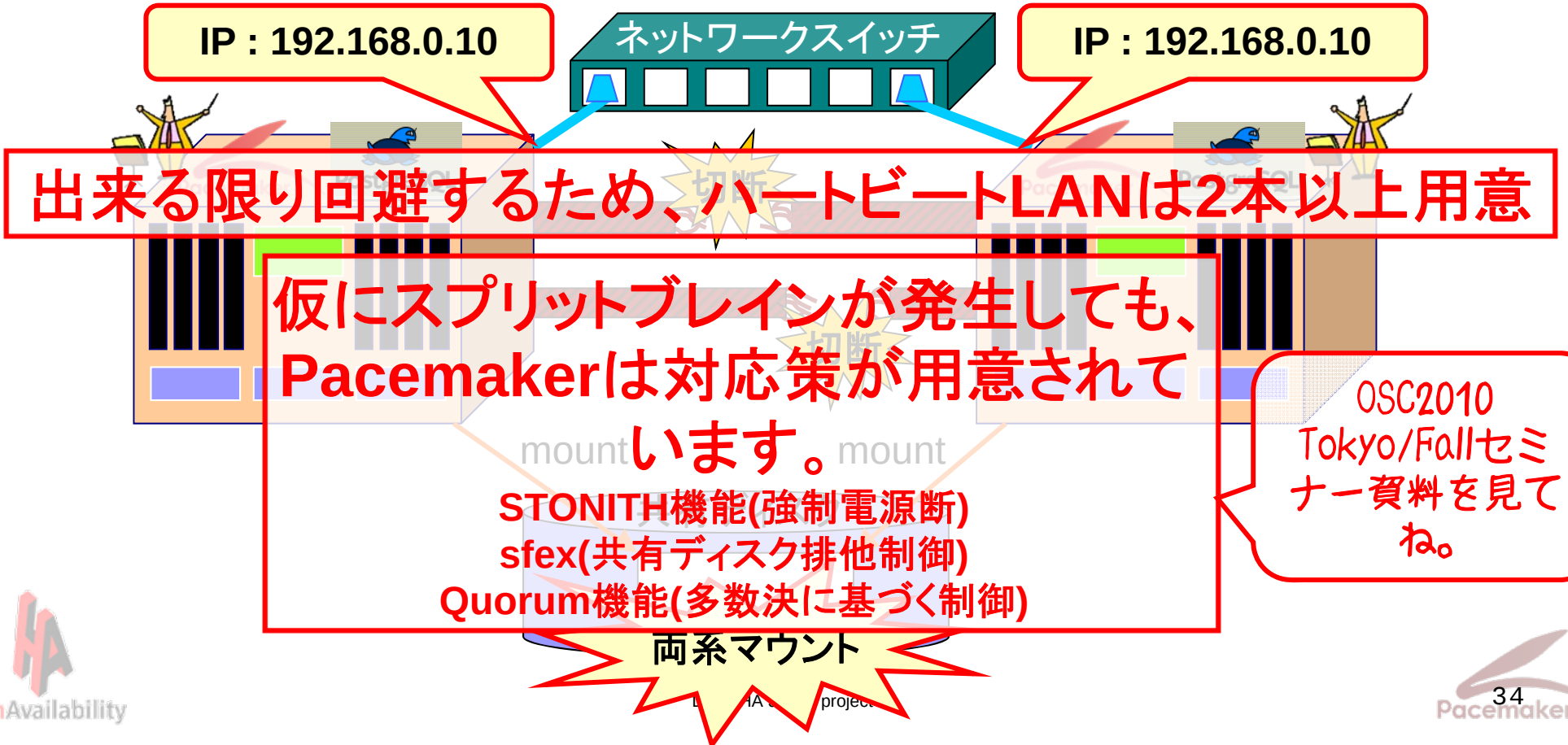
お互いが相手が故障したと判断し、サービスを引き継ごうとします。これを**スプリットブレイン**と呼びます。



両サーバが勝手に動き始めると

（例えば）データを共有していると → **データ破壊発生**

IPを共有していると → **IP競合発生**

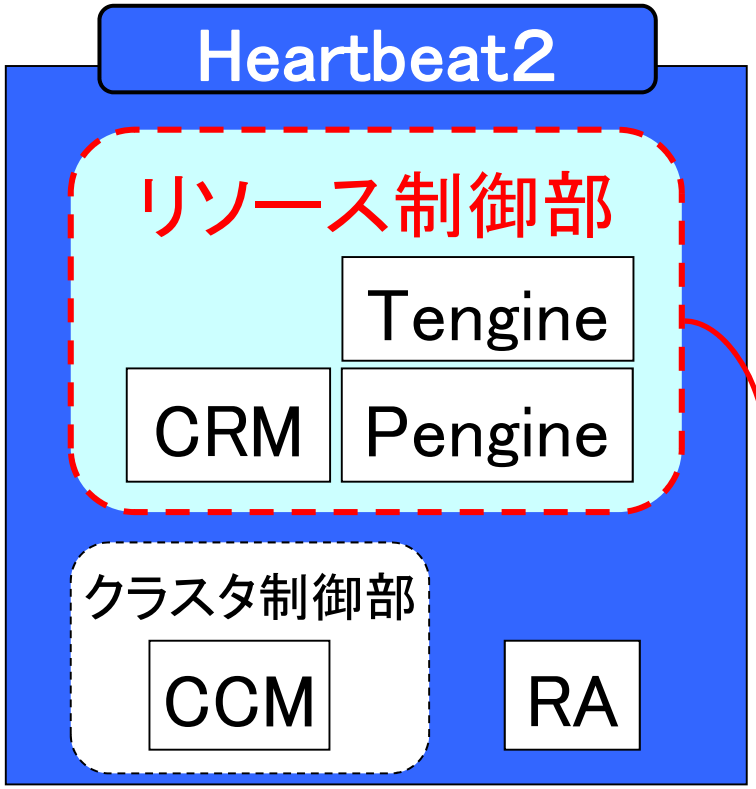


④

Pacemakerのコンポーネント構成

Pacemaker のコンポーネント構成は少々複雑なのです...

リソース制御部：Pacemaker



他のクラスタソフトウェア間とのコンポーネントの共通化のために、Heartbeat2のリソース制御部がPacemakerとして切り出されました

Pacemaker

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

※リソース制御機能は主にこのコンポーネントに含まれる

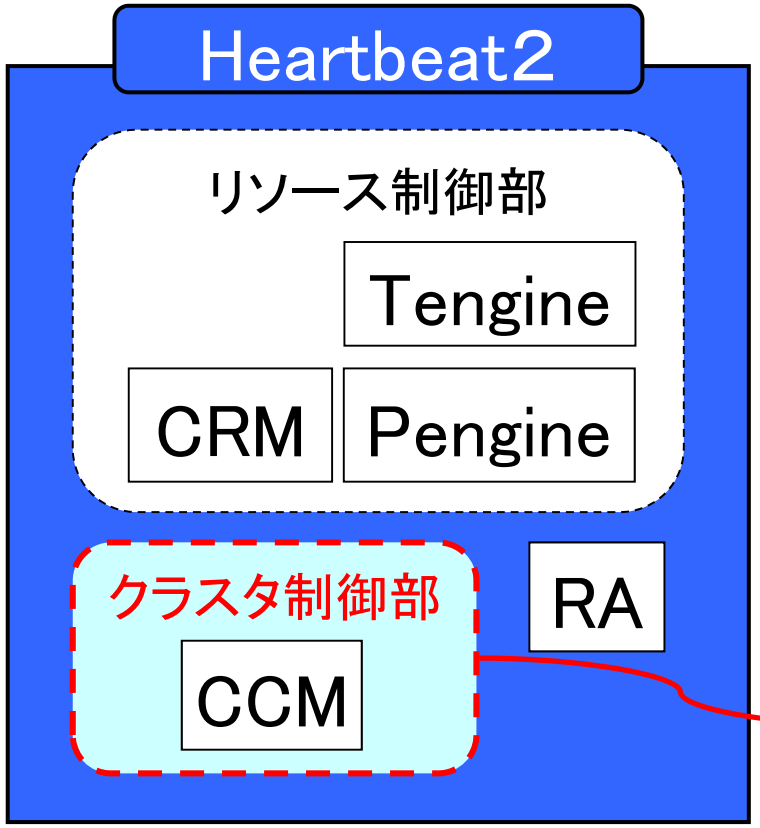
切り出されたということは・・・
Pacemaker 単独では
HAクラスタソフトとして
動作しない？

そのとおりです..

Pacemaker は
クラスタ制御部の
プログラムと組み合わせて
使用しなければなりません..

ですが、
クラスタ制御部の
候補がいくつもあると
前向きにとらえてください！

クラスタ制御部 候補1：Heartbeat3



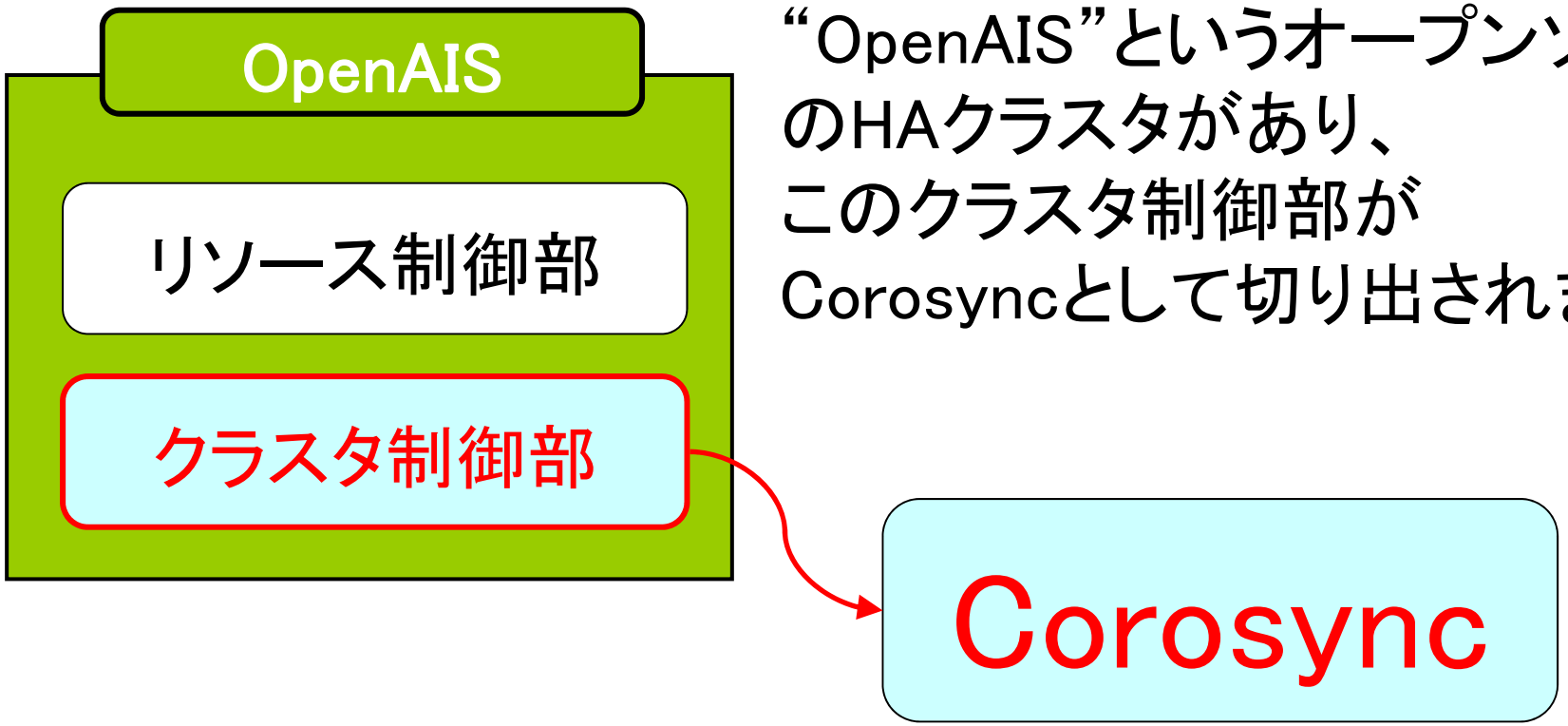
Heartbeat2の
クラスタ制御部が、
Heartbeat3 として切り
出されました

Heartbeat3

※ノード監視は主にこちらの
コンポーネントに含まれる

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！？

クラスタ制御部 候補2：Corosync

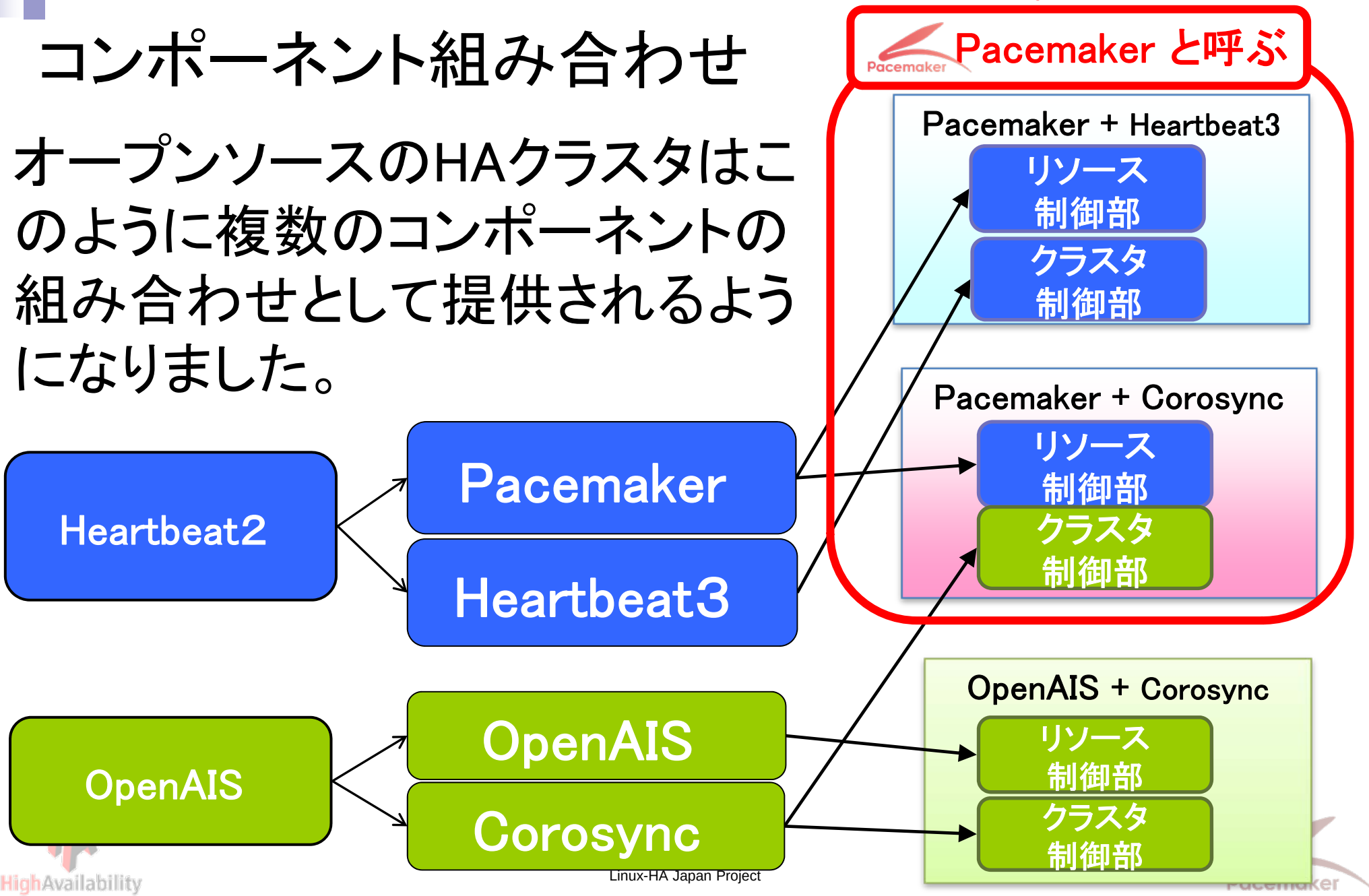


“OpenAIS”というオープンソースのHAクラスタがあり、
このクラスタ制御部が
Corosyncとして切り出されました

Pacemaker は
この「Heartbeat3」と「Corosync」
2つのクラスタ制御部が
選択可能

コンポーネント組み合わせ

オープンソースのHAクラスタはこのように複数のコンポーネントの組み合わせとして提供されるようになりました。



⑤

Pacemakerでクラスタリングに挑戦

～ CentOS 5.5(x86_64)編 ～

インストール方法の種類

1. yum を使ってネットワークインストール
   - Pacemaker本家(clusterlabs) の yumのリポジトリを使用
   - サーバにインターネット接続必須
2. ローカルリポジトリ + yum を使ってインストール
   - Linux-HA Japan 提供のリポジトリパッケージを使用
   - Linux-HA Japan オリジナルパッケージも含まれる
3. rpm を手動でインストール
   - 沢山のrpmを個別にダウンロードする必要あり
4. ソースからインストール
   - 最新の機能をいち早く試せる
   - コンポーネントが多いので、コンパイルは面倒

Pacemaker rpmパッケージ

現在公開されているHAクラスタを構築するのに
必要なCentOS5(x86_64)用のrpmパッケージ 計9個
(Linux-HA Japan公開のリポジトリパッケージの場合)

- pacemaker-1.0.10-1.4.el5.x86_64.rpm
- pacemaker-libs-1.0.10-1.4.el5.x86_64.rpm
- corosync-1.2.5-1.3.el5.x86_64.rpm
- corosynclib-1.2.5-1.3.el5.x86_64.rpm
- cluster-glue-1.0.6-1.6.el5.x86_64.rpm
- cluster-glue-libs-1.0.6-1.6.el5.x86_64.rpm
- resource-agents-1.0.3-3.el5.x86_64.rpm
- heartbeat-3.0.4-1.el5.x86_64.rpm
- heartbeat-libs-3.0.4-1.el5.x86_64.rpm

11/12 に 1.0.10 が
リリースされました！

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
パッケージの組み合わせも
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
リポジトリパッケージを使えば
インストールは簡単！

～ ローカルリポジトリ + yum を使ってインストール ～

（サーバにインターネット接続環境がなくてもOK！）

- 1. Pacemakerリポジトリパッケージをダウンロード

Linux-HA Japan 提供の 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。

2. 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-1.0.10-1.4.1.el5.x86_64.repo.tar.gz
                :
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/rpm/pacemaker-1.0.10-1.4.1.el5.x86_64.rpm
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/pacemaker.repo
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/repodata/
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/repodata/primary.xml.gz
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/repodata/other.xml.gz
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/repodata/filelists.xml.gz
pacemaker-1.0.10-1.4.1.el5.x86_64.repo/repodata/repomd.xml
```

インストールするRPMファイルと
repoファイル等が展開されます

3. 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-1.0.10-1.4.1.el5.x86_64.repo/
# vi pacemaker.repo
```

```
[pacemaker]
name=pacemaker
baseurl=file:///tmp/pacemaker-1.0.10-1.4.1.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 なので、/tmpに tar.gzファイルを展開したのならば修正不要）

4. yumでローカルからインストール！

```
# 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)

- Linux-HA Japanオリジナルパッケージも同時にインストール可能！

```
# yum –c pacemaker.repo install pacemaker pm_crmgen pm_diskd
pm_logconv-hb pm_extras
```

- pm_crmgen-1.0-1.el5.noarch.rpm ・・・ crm用設定ファイル編集ツール
- pm_diskd-1.0-1.el5.x86_64.rpm ・・・ ディスク監視アプリとRA
- pm_logconv-hb-1.0-1.el5.noarch.rpm ・・・ ログ変換ツール
- pm_extras-1.0-1.el5.x86_64.rpm ・・・ その他オリジナルRA 等

ぜひぜひ使ってみてください！

ちょっと注意..

- CentOS5.5（RHEL5.5）に付属している
  libxml2 ではタイミングによってPacemakerが
  正常に動作しないバグがありました。

  libxml2-2.6.26-2.1.2.8.el5_5.1
  で修正されているので、
  アップデートすることをお勧めします。

  詳細：https://rhn.redhat.com/errata/RHBA-2010-0764.html

Pacemakerの設定に挑戦！

～ Pacemaker + Heartbeat3 編 ～

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

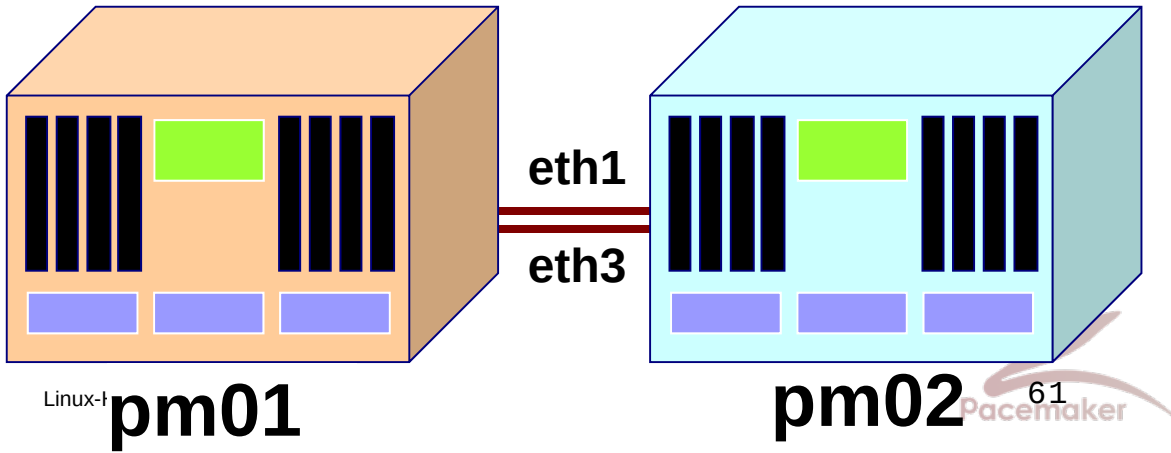
/etc/ha.d/ha.cf

- クラスタ制御部の基本設定ファイル
- クラスタ内の**全ノードに同じ内容のファイルを設置**

```
pacemaker on

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

bcast eth1
bcast eth3

node pm01
node pm02

watchdog /dev/watchdog
```

基本的に Heartbeat2 のha.cf と設定は同じ

Heartbeat2 での「crm on」から「pacemaker on」に変更となります



/etc/ha.d/authkeys

- ノード間の「認証キー」を設定するファイル
- クラスタ内の**全ノードに同じ内容のファイルを配置**
- 所有ユーザ/グループ・パーミッションは root/root ・ rw---- に設定

```
auth 1
1 sha1 secret
```

これも基本的に Heartbeat2 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

- 必須の設定ではないが、多くのログが/var/log/messagesに出力されるため出力先を個別のファイルに変更するのがお勧め
- 以下は /var/log/ha-log への出力例

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

ha.cf で設定したlogfacility 名

これでとりあえず Pacemakerのクラスタ制御部が 起動します！

```
# /etc/init.d/heartbeat start                ← 2ノードで実行
Starting High-Availability services:                [  OK  ]
```

起動確認

Pacemakerの状態表示コマンド

crm_monコマンドを利用

```
# crm_mon
============
Last updated: Wed Nov 10 14:28:55 2010
Stack: Heartbeat
Current DC: pm02 (a59a9306-d6e7-4357-bb0c-a5aea0615e61) - partition
with quorum
Version: 1.0.10-da7075976b5ff0bee71074385f8fd02f
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(ホスト名)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
なにも起動していません…

計画

- リソース制御するには事前に計画が必要
  - リソースの選択

    Apache、PostgreSQL、NW監視など、何を使用するか？
    リソースエージェント(RA)がなければ、予め自作してみるか？

  - リソースの動作の定義

    リソースの監視（monitor）間隔は何秒にするか？タイムアウトは？
    故障時はどのように動作させるか？
    リソースエージェント(RA)に与えるパラメータは？

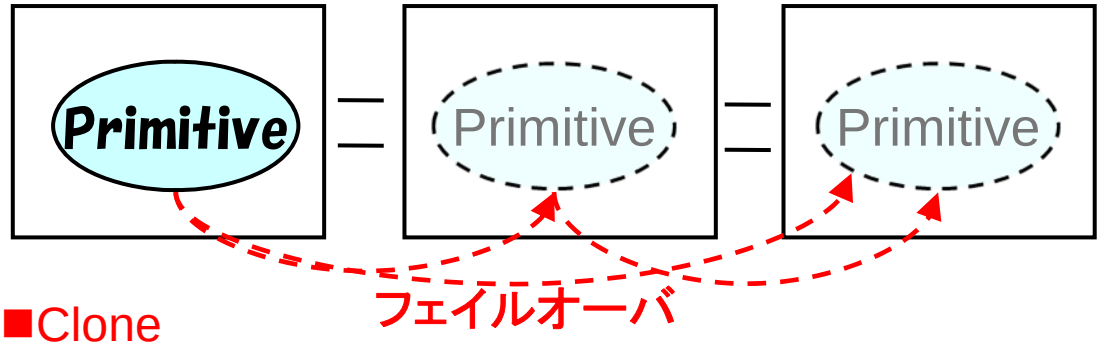
  - リソース配置・連携の定義

    リソースをどのノードで起動させるか？
    リソースの起動順番は？

クラスタ制御部の設定

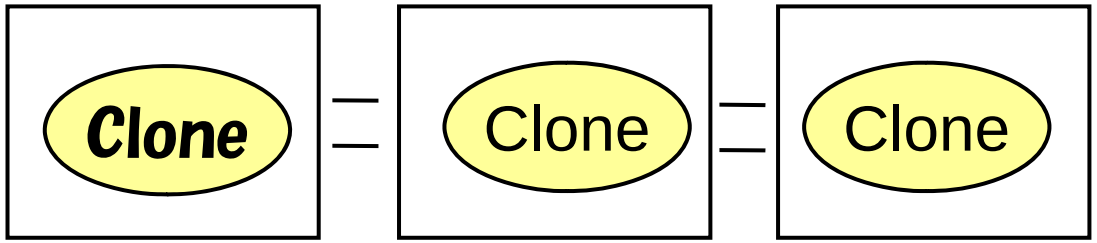
リソース定義の種類

■Primitive



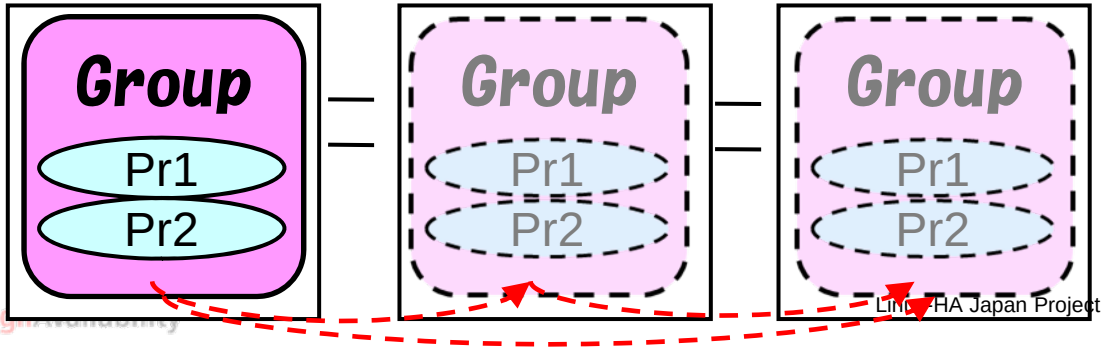
全てのリソース定義の最小単位。
1つのサーバプログラム、コンピュータ資源に対応する。
(例) PostgreSQL, 仮想IPアドレス

■Clone



同じ設定のリソースを複数のノードで動作させたい場合に使用。
Primitive を定義した後Clone化する。
(例) NW監視, ディスク監視

■Group



複数のリソースをまとめてフェイルオーバさせるために使用。Group内のPrimitiveには、起動／停止の順序制約も付与される。
(例)Apacheと仮想IPアドレスをグループ化する

設定方法

- 主に2通り
  - cib.xml ファイルにXML形式で設定を記述
    - 従来のHeartbeat 2での方法
    - XMLを手で書く必要があり面倒
  - crmコマンドで設定
    - Pacemakerからの新機能

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="172.20.24.110"/>
     <nvpair id="prmIp-instance_attributes-nic" name="nic" value="eth0"/>
     <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…

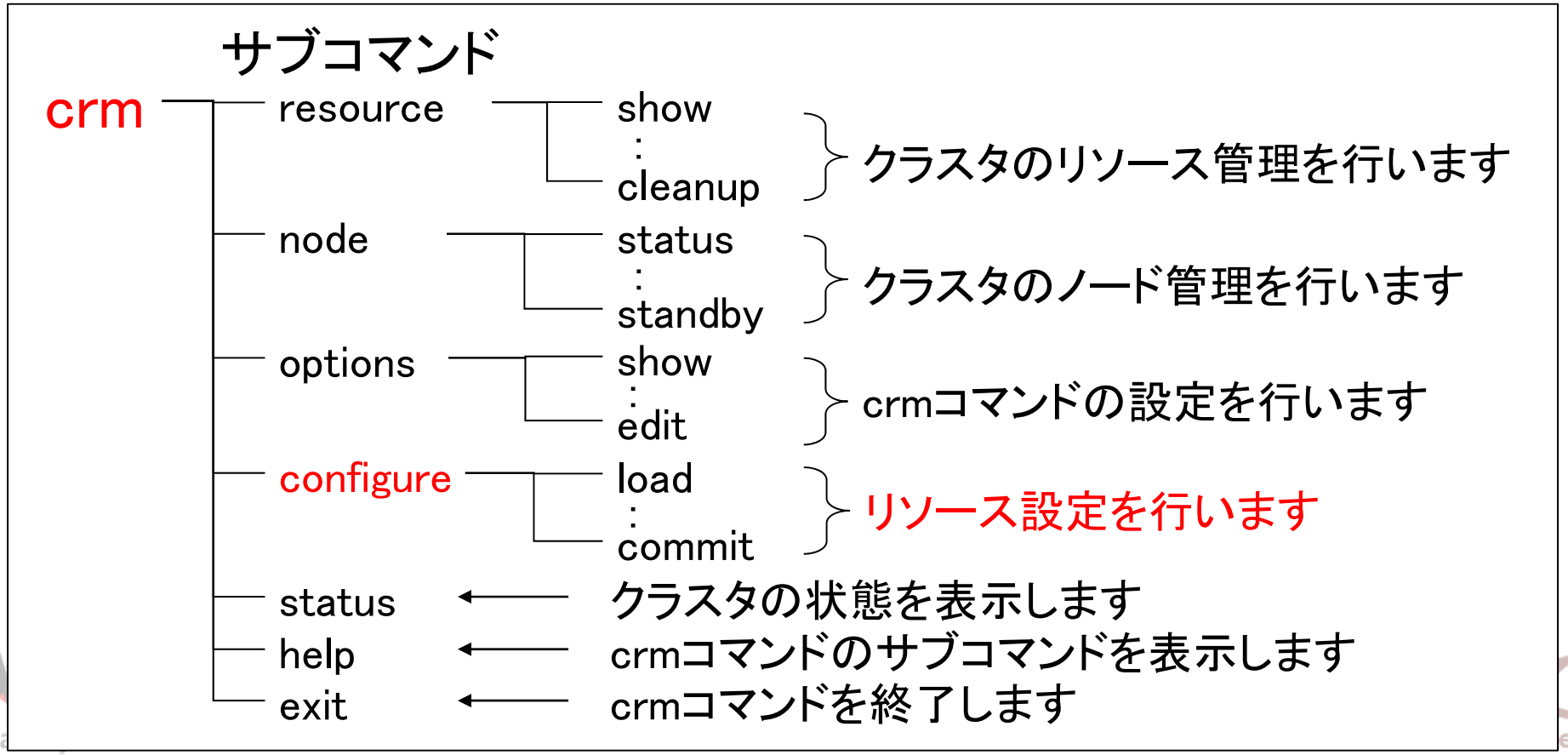
そこで、

Pacemaker での新機能

crmコマンドを使ってみよう！

crm コマンド

- 階層構造をもったコマンドラインインターフェイス
- 設定だけでなく、Pacemakerの状態把握や管理も可能
- TABキーで入力内容の補完可能



crmコマンド実行例

「IPaddr2」リソースエージェントを使用して仮想IPを設定をするcrmコマンド例です

```
shell# crm
crm(live)# configure
crm(live)configure# primitive MyIp ocf:heartbeat:IPaddr2 ¥
 params ip=”172.20.24.110" nic="eth0“ ¥
 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“
              :
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 cib.xml なのです）

これでも設定方法が
わかりにくいって人には、

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！

設定ファイル編集ツール

pm_crmgen

2010/11/26 に pm_crmgen 1.0 版をリリース

Linux-HA Japanで
crmファイル編集ツールを開発！

Excelのテンプレートファイルから簡単に
crm用設定ファイルを生成してくれるツール

編集、CSV出力ができればよいので、OpenOffice, GoogleDocsでもOK!

リポジトリパッケージに含まれていますし、
個別にダウンロードも可能です。

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

- どのノードが優先的にActive？
- NW監視は？
- NWが壊れた時の挙動は？

など細かい挙動の設定も
可能です！

インストール

① pm_crmgenをインストール

rpmコマンドでインストールする場合

```
# rpm –ivh pm_crmgen-1.0-1.el5.noarch.rpm
```

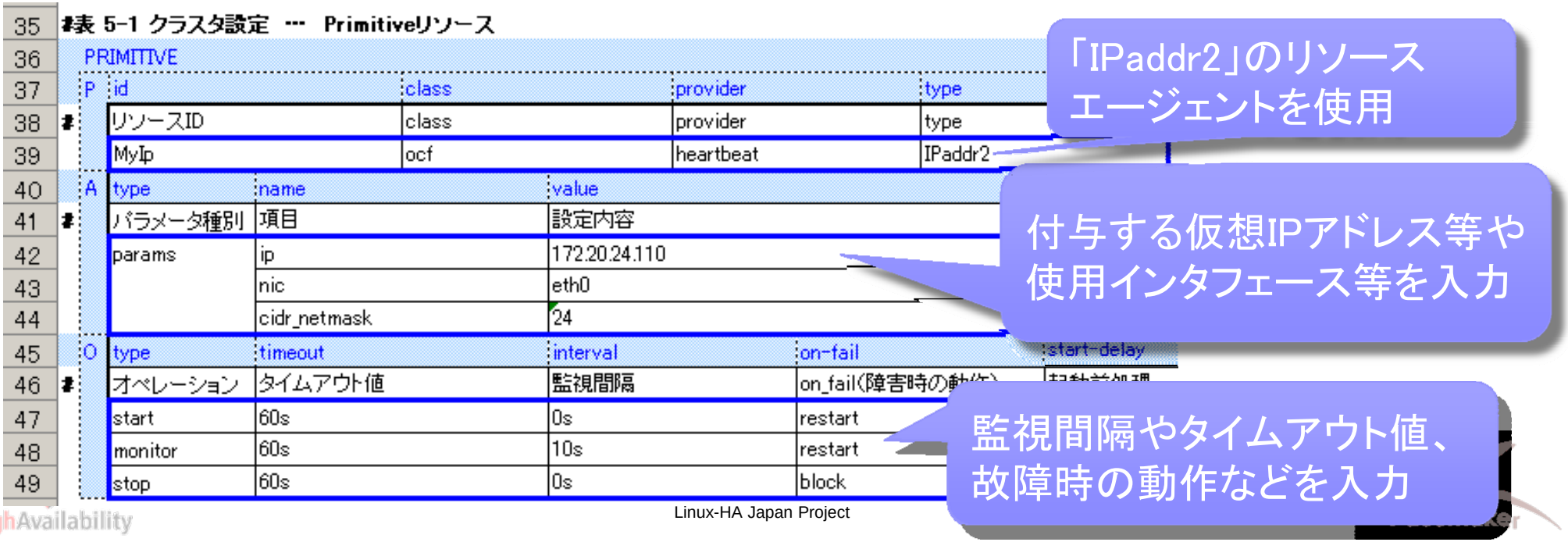
ローカルリポジトリの場合

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

設定イメージ

② Excelのテンプレートファイルにリソース定義を記載

/usr/share/pacemaker/pm_crmgen/pm_crmgen_env.xls ファイルを
Excel が使用できるPCにコピーします。
テンプレートは青枠の中に値を記入していきます。



35	#表 5-1 クラスタ設定 --- Primitiveリソース				
36	PRIMITIVE				
37	P	id	class	provider	type
38	#	リソースID	class	provider	type
39		MyIp	ocf	heartbeat	IPaddr2
40	A	type	name	value	
41	#	パラメータ種別	項目	設定内容	
42		params	ip	172.20.24.110	
43			nic	eth0	
44			cidr_netmask	24	
45	O	type	timeout	interval	on-fail
46	#	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)
47		start	60s	0s	restart
48		monitor	60s	10s	restart
49		stop	60s	0s	block

どのノードをActiveにするかといった
リソース配置制約の設定も、ノード名を記述
するだけで可能です。

67 #表 6-1 クラスタ設定 --- リソース配置制約
68 LOCATION

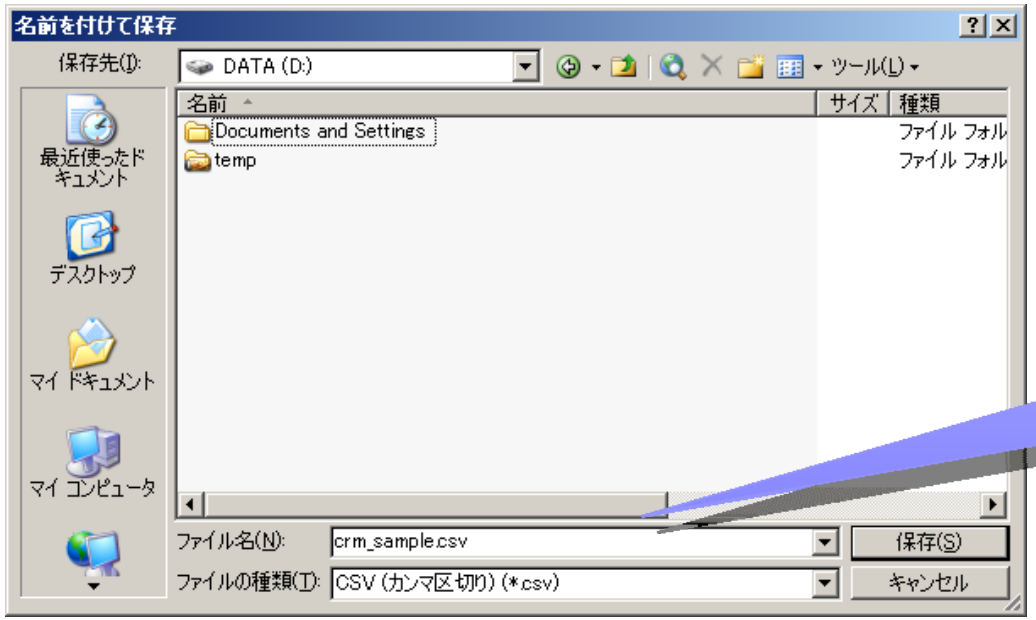
69	rsc	score:200	score:100	score:-inf
70 #	リソースID	Activeノード	Standbyノード	非稼働ノード
71	MyGrp	pm02	pm01	
72				
73				

リソースID

ActiveとStandbyノードを指定

crm用設定ファイルに変換

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やFTP等でpm_crmgenがインストール
されたサーバへ転送

crm用設定ファイルに変換

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
```

生成する設定ファイル名

③で転送した
CSVファイル

⑥ crmコマンドを実行してリソース設定を反映

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

出来上がった crmファイル例

```
(..略..)

### Primitive Configuration ###
primitive MyIp ocf:heartbeat:IPaddr2 ¥
        params ¥
                ip=”172.20.24.110" ¥
                nic="eth0" ¥
                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コマンドが
ファイルに記述されます

リソース起動の確認

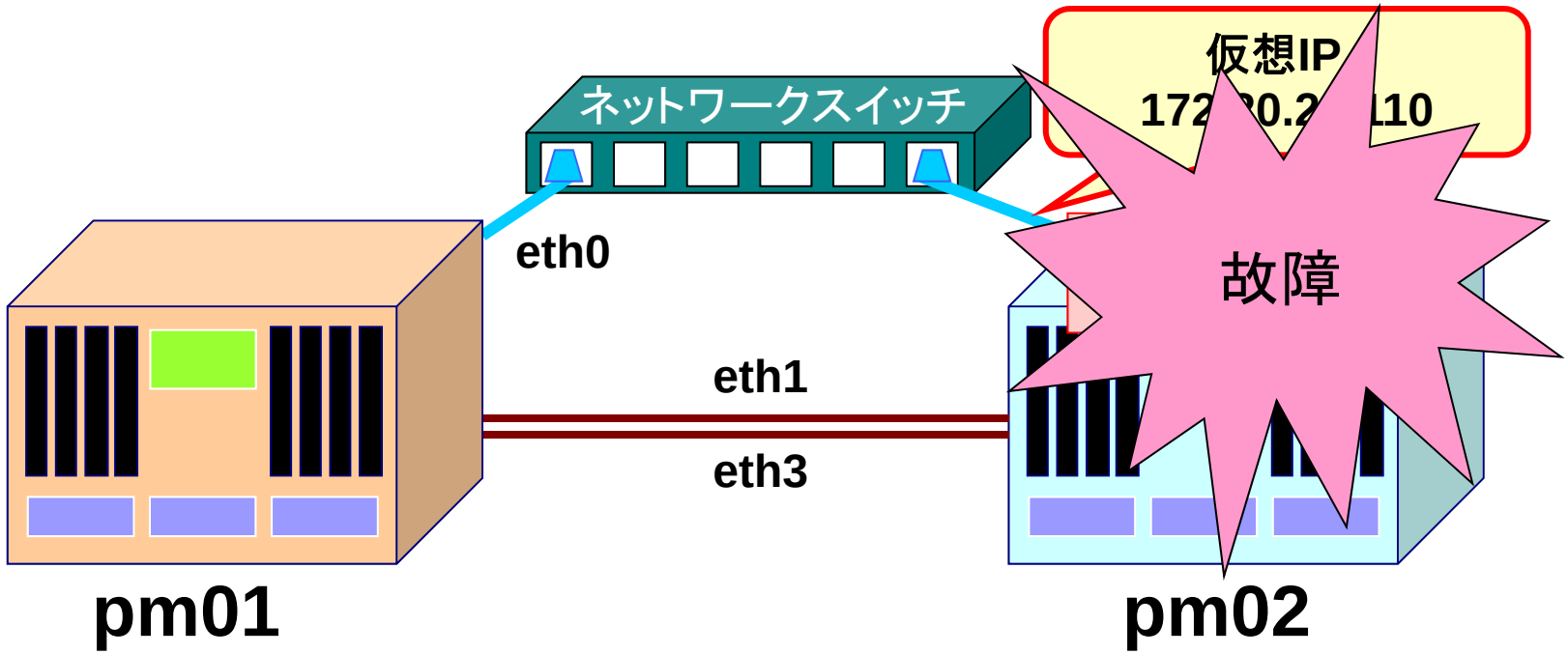
- crm_mon コマンドでリソース起動を確認

```
# crm_mon
```

```
============
～省略～
============

Online: [ pm02 pm01 ]

 Resource Group: MyGrp
     MyIp          (ocf::heartbeat:IPaddr2):       Started pm02
     MyApache      (ocf::heartbeat:apache):        Started pm02
```

仮想IPとApacheが
pm02で起動

もしノード故障が発生すると・・・



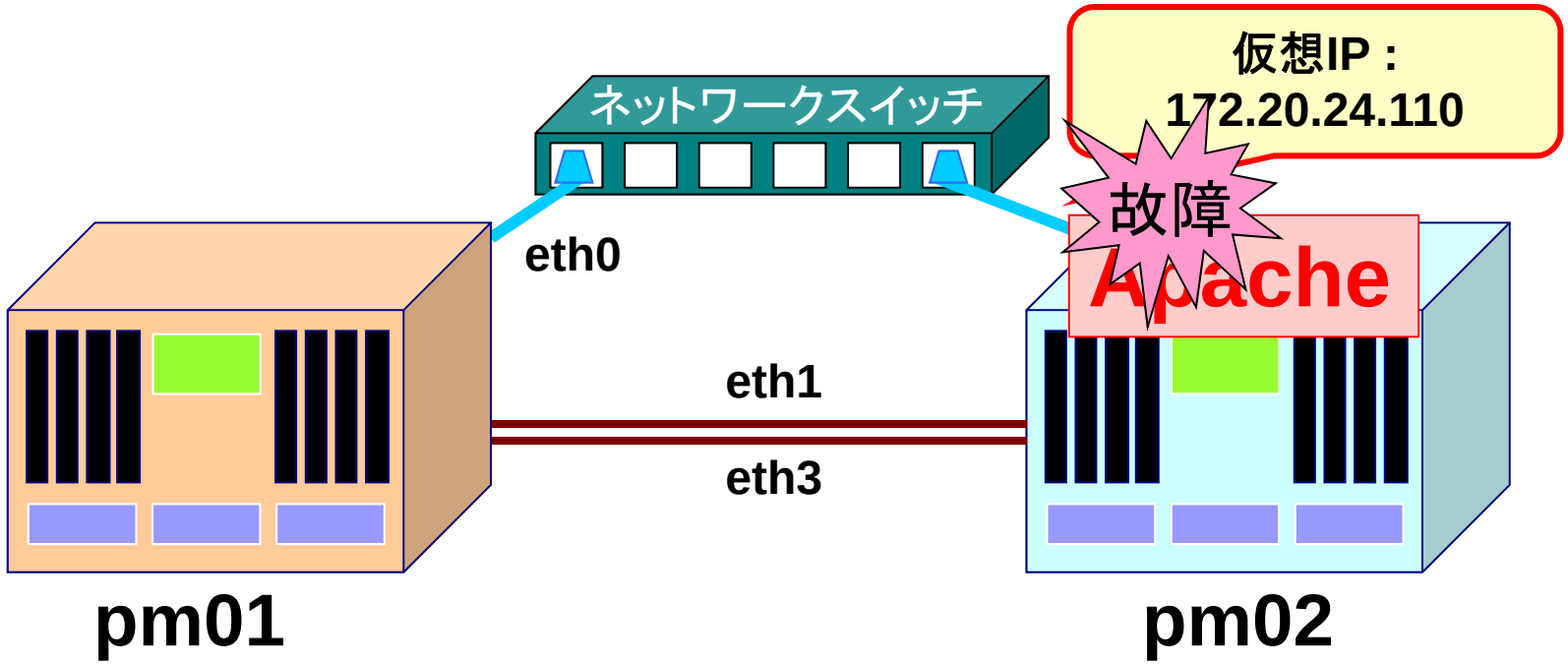
ノード故障時の状態表示

```
============
～ 省略 ～
============

Online: [ pm01 ]
OFFLINE: [ pm02 ]

 Resource Group: MyGrp
     MyIp            (ocf::heartbeat:IPaddr2):       Started pm01
     MyApache        (ocf::heartbeat:apache):        Started pm01
```

ノードpm01からはノードpm02が
見えなくなったので「OFFLINE」と表示

pm01にフェイルオーバ

もしリソース故障が発生すると・・・



リソース故障時の状態表示

```
============
～ 省略 ～
============

Online: [ pm02 pm01 ]

 Resource Group: MyGrp
     MyIp       (ocf::heartbeat:IPaddr2):       Started pm01
     MyApache   (ocf::heartbeat:apache):   Started pm01

Failed actions:
   MyApache_monitor_10000 (node=pm02, call=13, rc=1, status=complete): unknown error
   MyApache_start_0 (node=pm02, call=15, rc=1, status=complete): unknown error
```

pm01にフェイルオーバ

リソース故障状況が表示されます
※ノードpm02でApacheが
monitor, start 故障

⑥

Linux-HA Japanについて

(ころ)40分経過ぐらいだよ。しっかりまとめてよ。

Linux-HA Japan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、Heartbeat2のrpmバイナリと、オリジナルのHeartbeat機能追加用パッケージを提供してきました。

Linux-HA Japan URL

http://linux-ha.sourceforge.jp/ （一般向け）

http://sourceforge.jp/projects/linux-ha/ （開発者向け）

情報の公開用として
新しい一般向けウェブサイトが
2010/6/25にオープンしました。

随時情報を更新しています！

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
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.sourceforge.jp/
の「メーリングリスト」をクリック

・MLアドレス

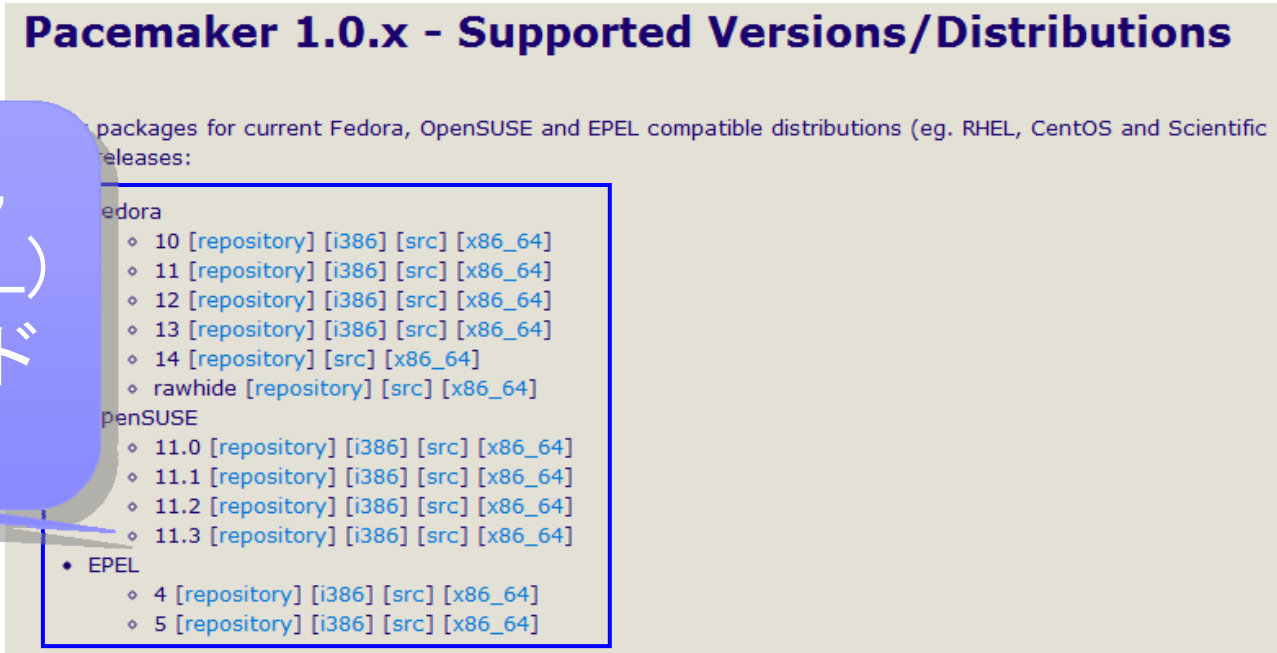
linux-ha-japan@lists.sourceforge.jp

※スパム防止のために、登録者以外の投稿は許可制です

本家Pacemakerサイト

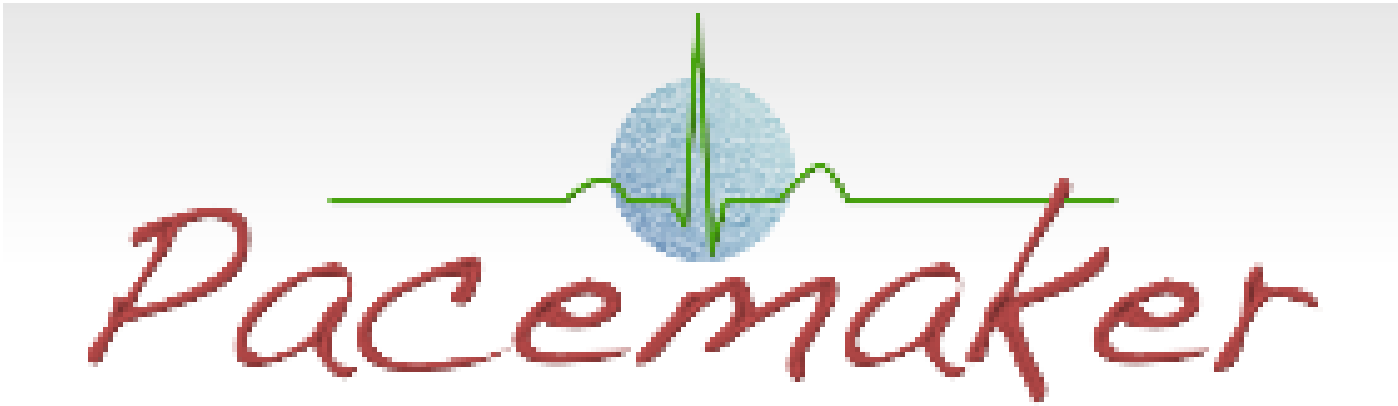
http://clusterlabs.org/

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



http://clusterlabs.org/rpm

ところで、
昨年12月まで
実は本家の
Pacemakerのロゴはこれでした



しかし

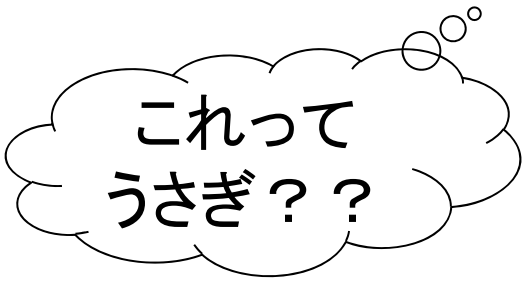
これ 

では、

いかにも医療機器っぽいので…

Pacemakerロゴ

Linux-HA Japan では、
Pacemakerのロゴ・バナーを独自に作成



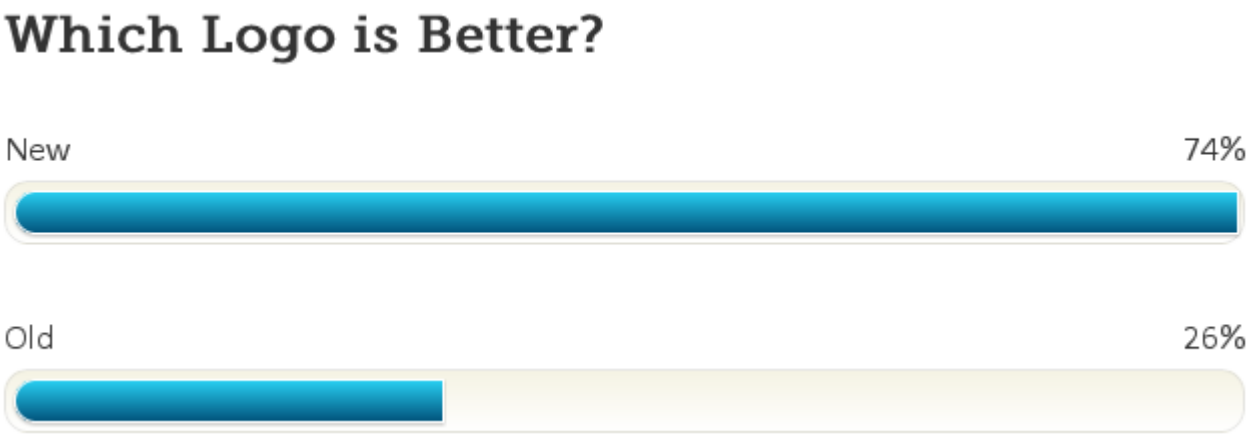
本家Pacemakerロゴに勝負！

http://theclusterguy.clusterlabs.org/post/1551578523/new-logo

Cluster Guy New logo　検索

VS

YOU'RE WINNER!



本家Pacemaker新ロゴに採用!

さいごに…

HAクラスタを全く知らなかった人へ・・・

会社や学校の

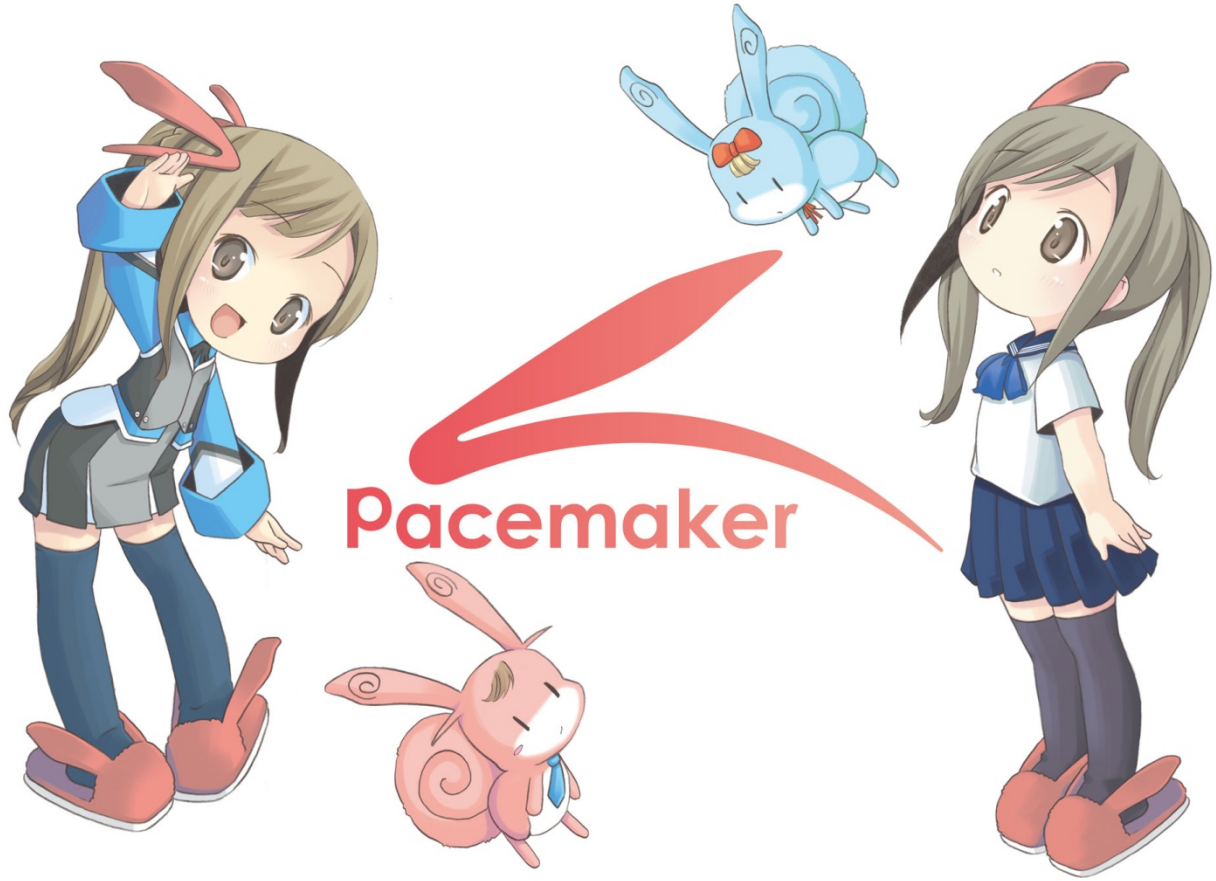
- ・メールサーバ
- ・ファイルサーバ
- ・Webサーバ



といった情報システムが止まって「いらいら」したり、管理者として「ドキドキ」、「ハラハラ」した経験はありませんか？

Pacemakerは、この「いらいら」や「ドキドキ」、「ハラハラ」を和らげてくれるソフトウェアになってくれれば幸いです！

ご清聴ありがとうございました



⑦

参考情報

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

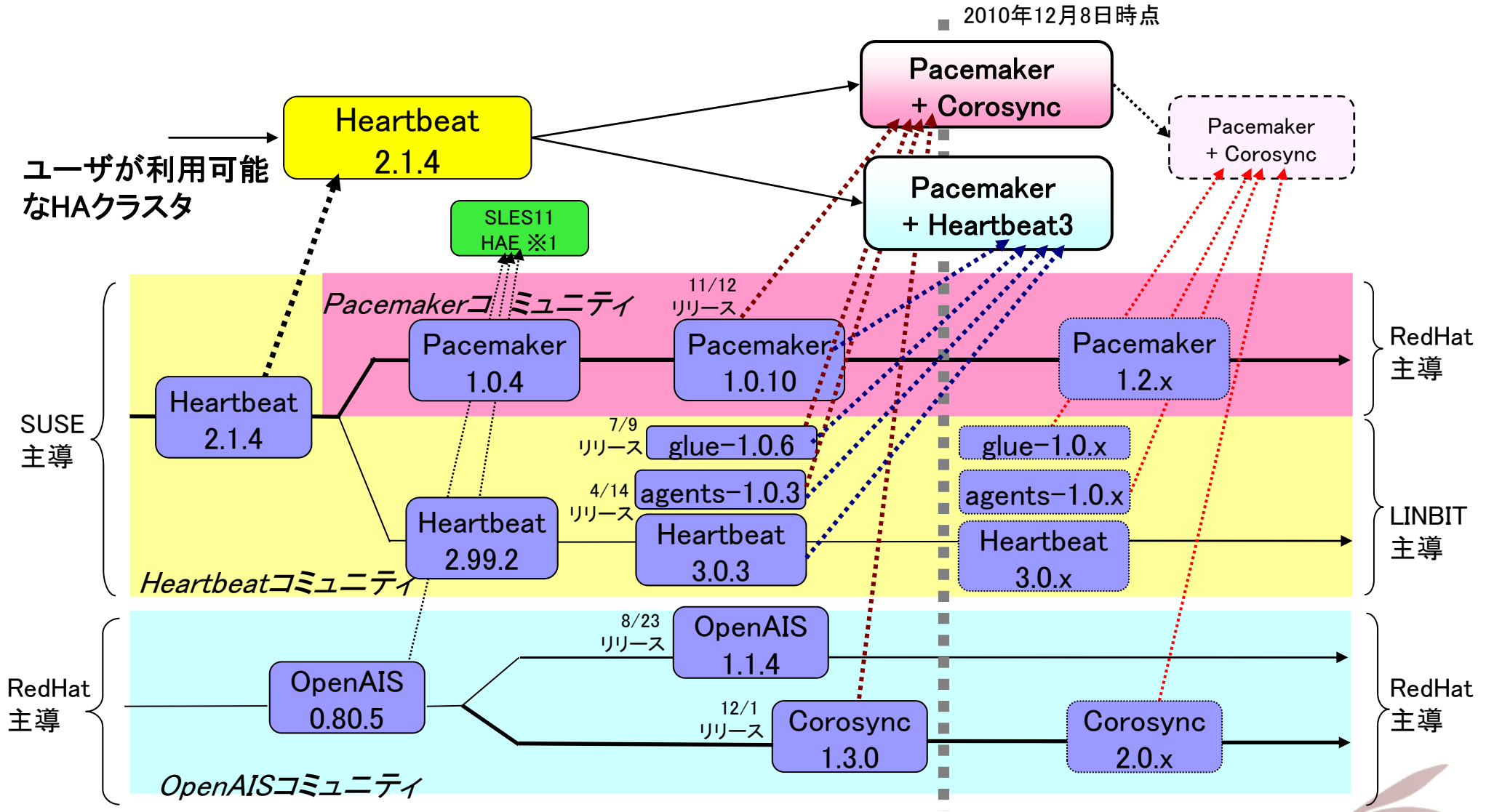
Heartbeat3 のメリット・デメリット

- メリット
  - 安定
    - Heartbeat2系のクラスタ制御部を引き継いでいるため、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
    - リソース数にもよるが、7ノードくらいが限界
  - ハートビートLAN切断時(スプリットブレイン) に弱い
    - スプリットブレイン時の復旧手順がやや複雑
    - クォーラム制御(3ノード以上時に使用)が不安定
  - オンラインによるノード追加・削除時の動作が不安定

Corosync のメリット・デメリット

- メリット
  - 多ノード構成に向いている
    - 11ノードで動いた！
    - 次期バージョンでは16ノード以上でも動くという情報も…
  - 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
  - オンラインによるノード追加・削除時の動作が安定
- デメリット
  - 開発途上で不安定
    - 頻繁にバグフィックス版がリリース

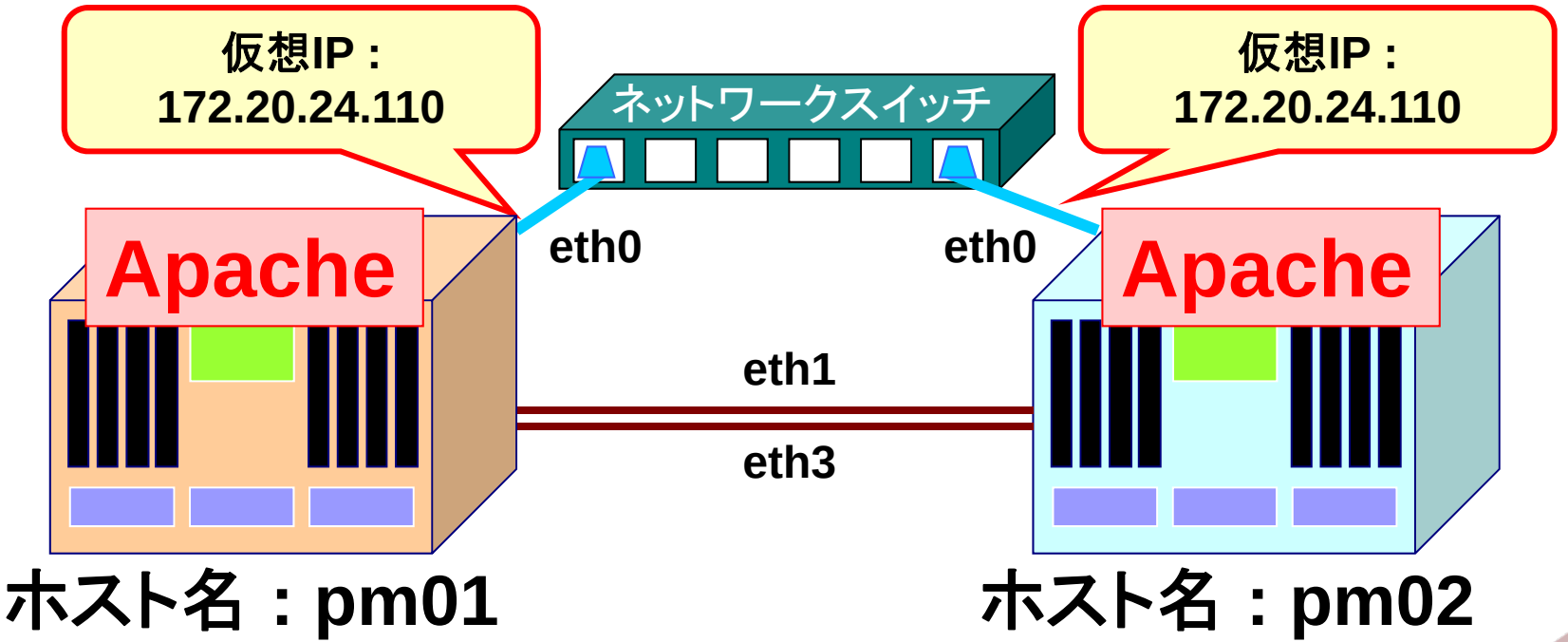
HAクラスタ開発コミュニティの状況



※1 SUSE Linux Enterprise Server 11 High Availability Extension

crmコマンドで、
仮想IPアドレス + Apache
のリソース設定に挑戦！

※apacheはインストール済み前提



crmコマンド：実行

- 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
crm(live)configure#                    ← ここから設定を入力していく
```

- サブコマンドを同時に指定することで、一気にリソース設定モードに入ることも可能

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

crm コマンド：基本動作設定

- クォーラムの設定
  - 2ノードでは基本的にignoreを設定
- STONITH設定
  - 今回は設定例を簡単にするために無効(false)に設定
  - 商用環境では有効にし、STONITHを設定することを推奨

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

crm コマンド：仮想IPアドレス設定

- 仮想IPアドレスを制御するリソースエージェント「IPaddr2」のリソース設定を行います

仮想IPのリソースIDを「**MyIp**」とします (任意の文字列)

```
crm(live)configure# primitive MyIp ocf:heartbeat:IPaddr2 ¥
        params ¥
                ip="172.20.24.110" ¥
                nic="eth0" ¥
                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"
```

IPaddr2に渡すパラメータ

監視間隔やタイムアウト故障時の動作

crm コマンド：Apache設定

- 「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「**MyApache**」とします(任意の文字列)

```
crm(live)configure# primitive MyApache 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"
```

apacheに渡すパラメータ

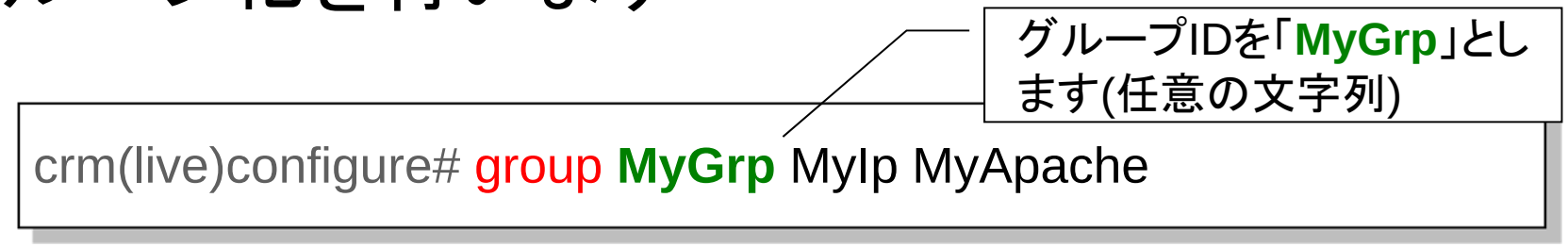
監視間隔やタイムアウト故障時の動作

※Apacheの監視URLに指定したファイルを作成する必要あり

```
# echo "hogehoge" > /var/www/html/test.html
```

crm コマンド：リソースのグループ化

- 設定した「IPaddr2」「apache」リソースのグループ化を行います



なぜグループ化？？

- グループ内のどれか1つでも壊れたら
  グループ全体がフェイルオーバ
- グループ化の順番でリソースを起動

crm コマンド：設定の確認

```
crm(live)configure# show
node $id="a0dacbcf-346f-4003-ab5b-15422e0e4697" pm01
node $id="fe705a39-541a-4b10-af22-de27d4c72d23" pm02
primitive MyApache 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 MyIp ocf:heartbeat:IPaddr2 ¥
        params ip="172.20.24.110" nic="eth0" 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 MyGrp MyIp MyApache
property $id="cib-bootstrap-options" ¥
        dc-version="1.0.10-da7075976b5ff0bee71074385f8fd02f296ec8a3" ¥
        cluster-infrastructure="Heartbeat" ¥
        no-quorum-policy="ignore" ¥
        stonith-enabled="false" ¥
        startup-fencing="false"
```

apache設定

仮想IP設定

グループ設定

共通設定

crm コマンド：設定の反映

- commitを実行すると設定が反映され、リソースが起動されます

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

- 設定は外部ファイルに保存もできます

```
crm(live)configure# save /root/config.crm
```

保存した設定は読み込めます

```
crm(live)configure# load update /root/config.crm
```

設定を全て破棄したい場合は、Pacemakerを停止後、/var/lib/heartbeat/crm/ 内のファイルを全て削除します