



# Pacemakerで かんたんクラスタリング 体験してみよう！

～仮想環境で運用デモしますよ編～

2011年7月16日 OSC2011 Kansai@Kyoto

Linux-HA Japan プロジェクト

田中 崇幸



# 自己紹介

- 名前：田中崇幸 (Takayuki Tanaka)
  - Twitter: @tanakacchi21
- 所属：Linux-HA Japanプロジェクト
  - コミュニティ旗揚時のメンバー
  - Pacemaker普及促進のため、OSCでの講演で全国行脚中
- 趣味：マラソン
  - 念願のサブスリーを達成したばかりの市民マラソンランナー
  - 8月：北海道マラソン、10月：大阪マラソン 出場予定



# 本日のお話

- ① Linux-HA Japanについて
- ② 本日のPacemakerデモ環境
- ③ crm\_monを使おう！
- ④ ログメッセージ制御機能を使おう！
- ⑤ いろいろデモします！



①

# Linux-HA Japanについて

# Linux-HA Japan URL

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

(一般向け)

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



Pacemaker情報の公開用として  
随時情報を更新中です。

このサイトより、Pacemakerリポジトリ  
パッケージがダウンロード可能です。

本日の資料、デモ用設定ファイルも  
このサイトから公開予定です！

# 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

 Gihyo.jp に連載しました！

『Pacemakerでかんたんクラスタリング  
体験してみよう！』



<http://gihyo.jp/admin/serial/01/pacemaker>

合計5回の連載で、Pacemakerの概要説明から構築方法、保守運用にいたるまで紹介しています。



②

# 本日のPacemakerデモ環境





# 本日のPacemakerデモ環境

## ■ ハードウェア

- ノートPC (Core2Duo 2.26MHz、メモリ 2G)

## ■ OS

- CentOS 5.6 x86\_64

## ■ HAクラスタ

- Pacemaker-1.0.11 (7/5に最新リポジトリパッケージがリリース)
- アクティブ/スタンバイの2台構成

## ■ クラスタ化するアプリケーション

- PostgreSQL 9.0.4

## ■ 仮想環境

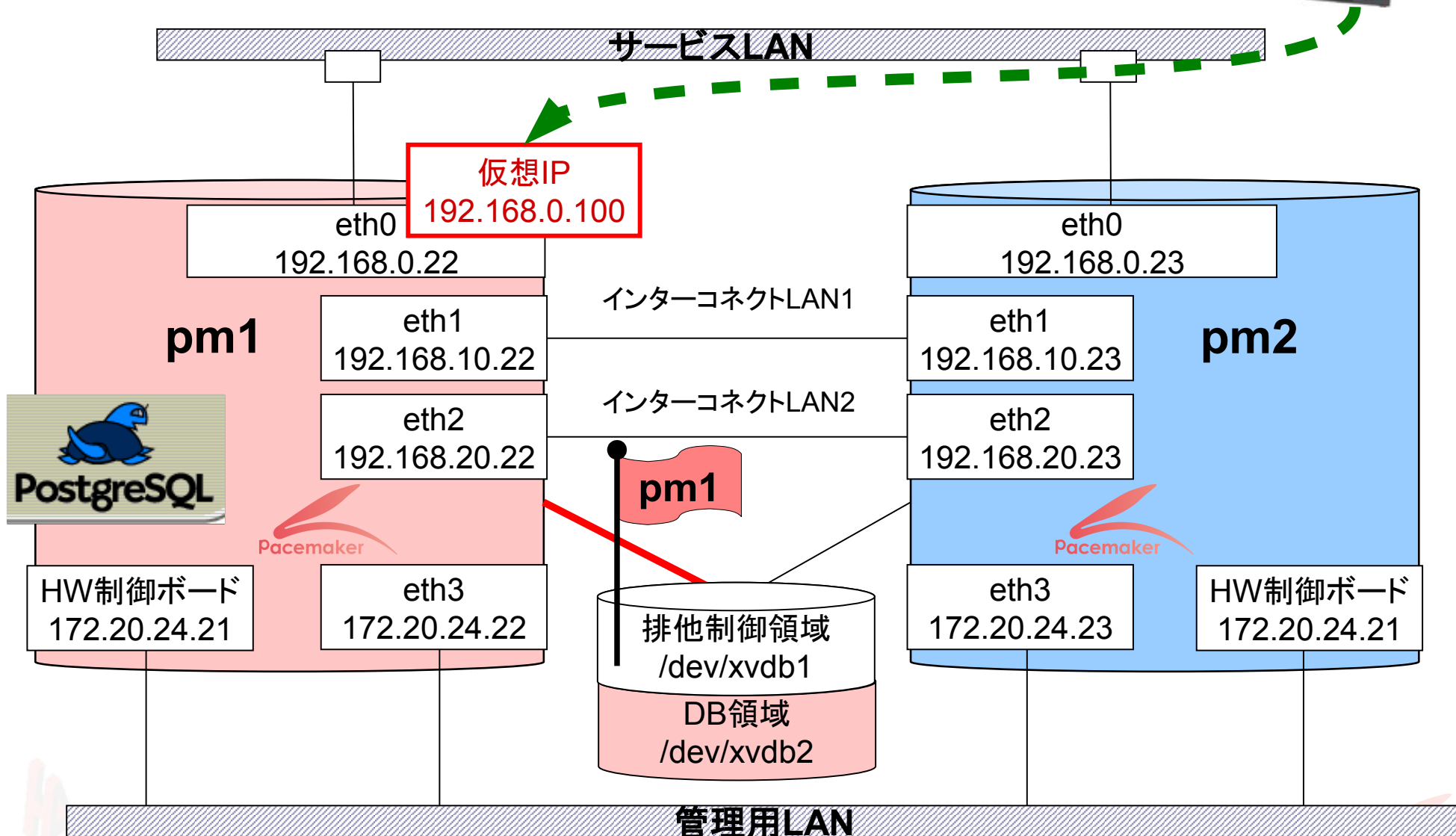
- Xen (CentOS 5.6同梱版)
- Domain-Uは2ドメインで構成
- 各ドメインには、CPU×1・メモリ480M を割り当て

# Pacemakerデモ構成

pm1: アクティブ

pm2: スタンバイ

demo  
(Domain-0)

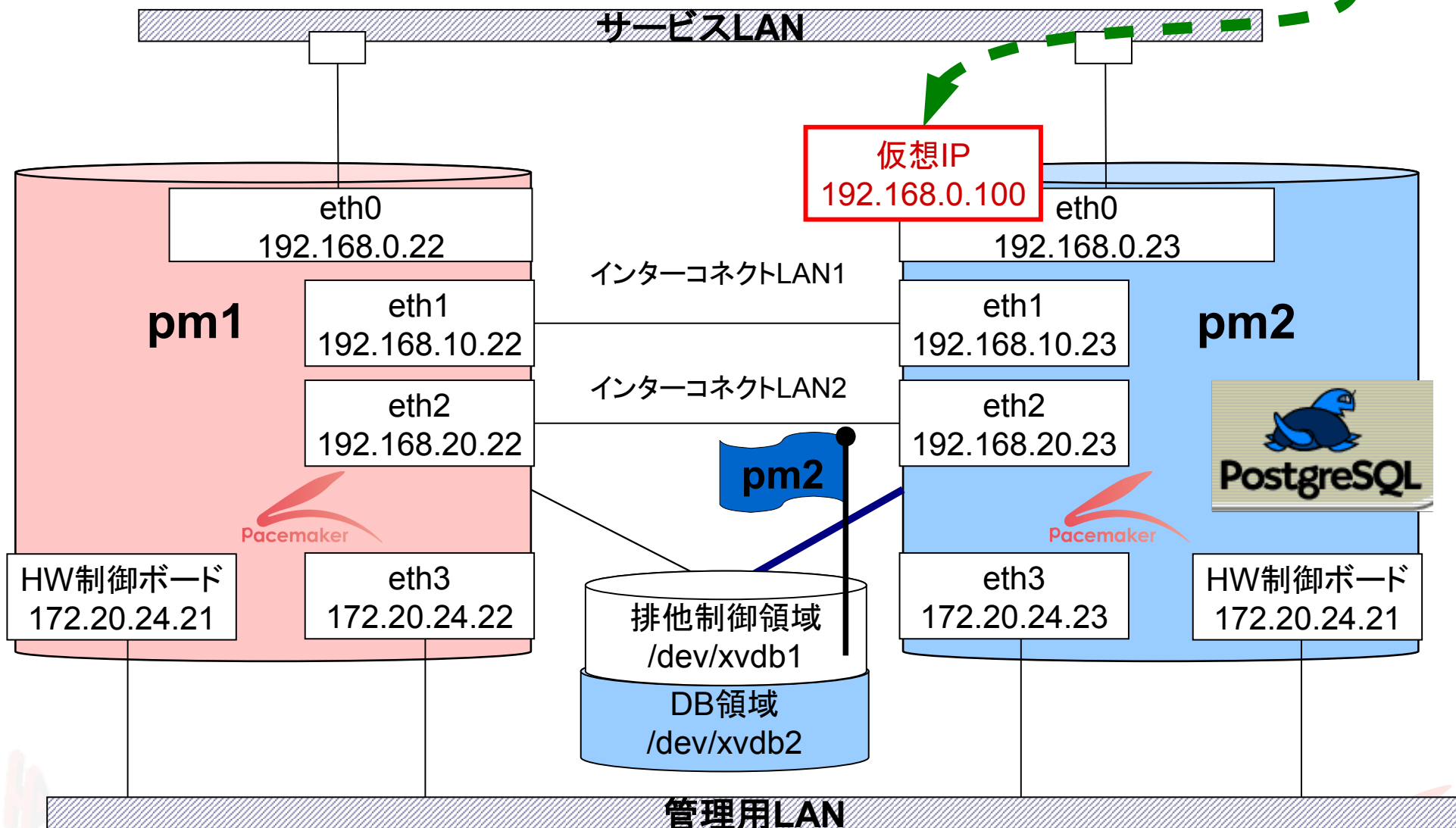


# Pacemakerデモ構成

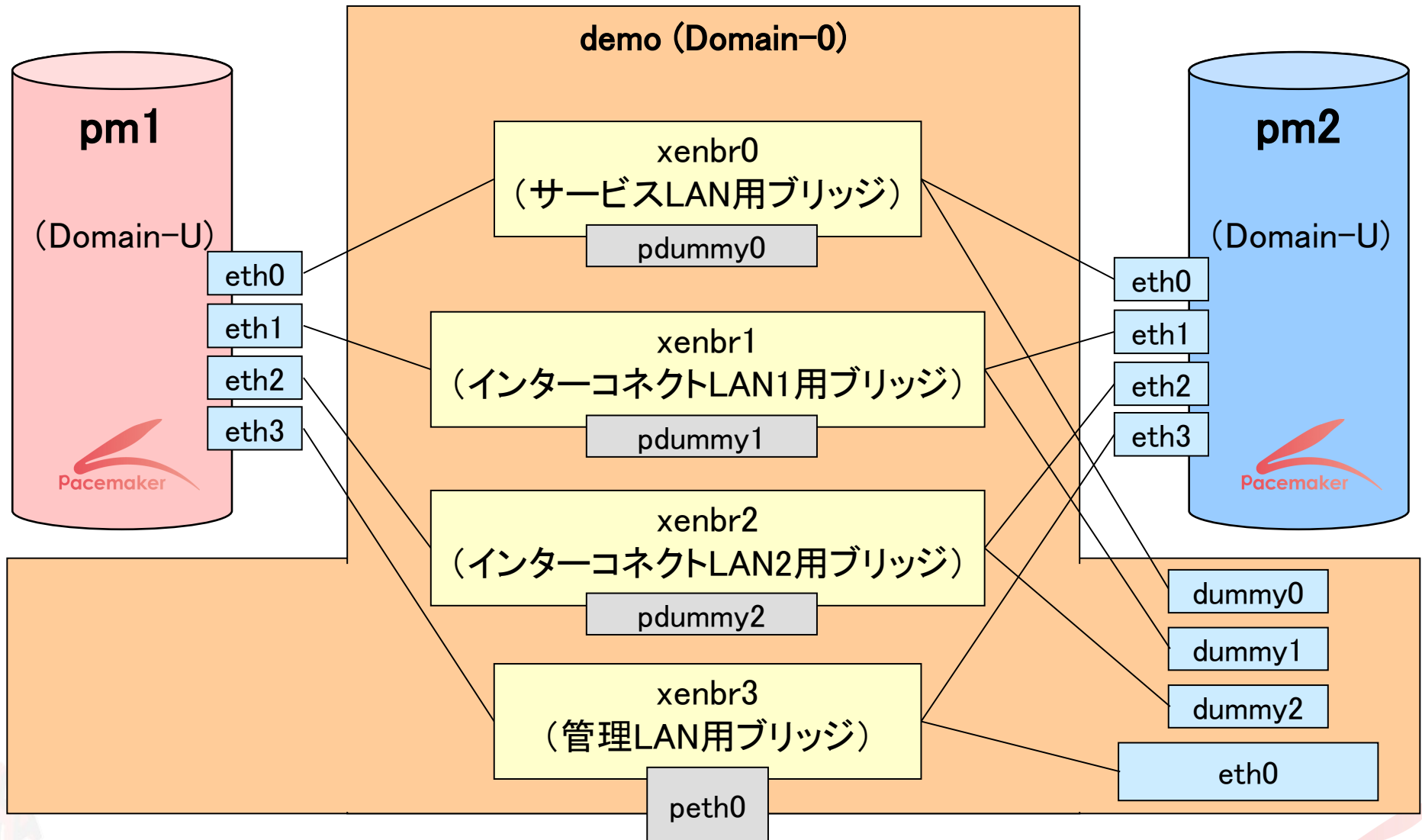
pm1: スタンバイ

**pm2: アクティブ**

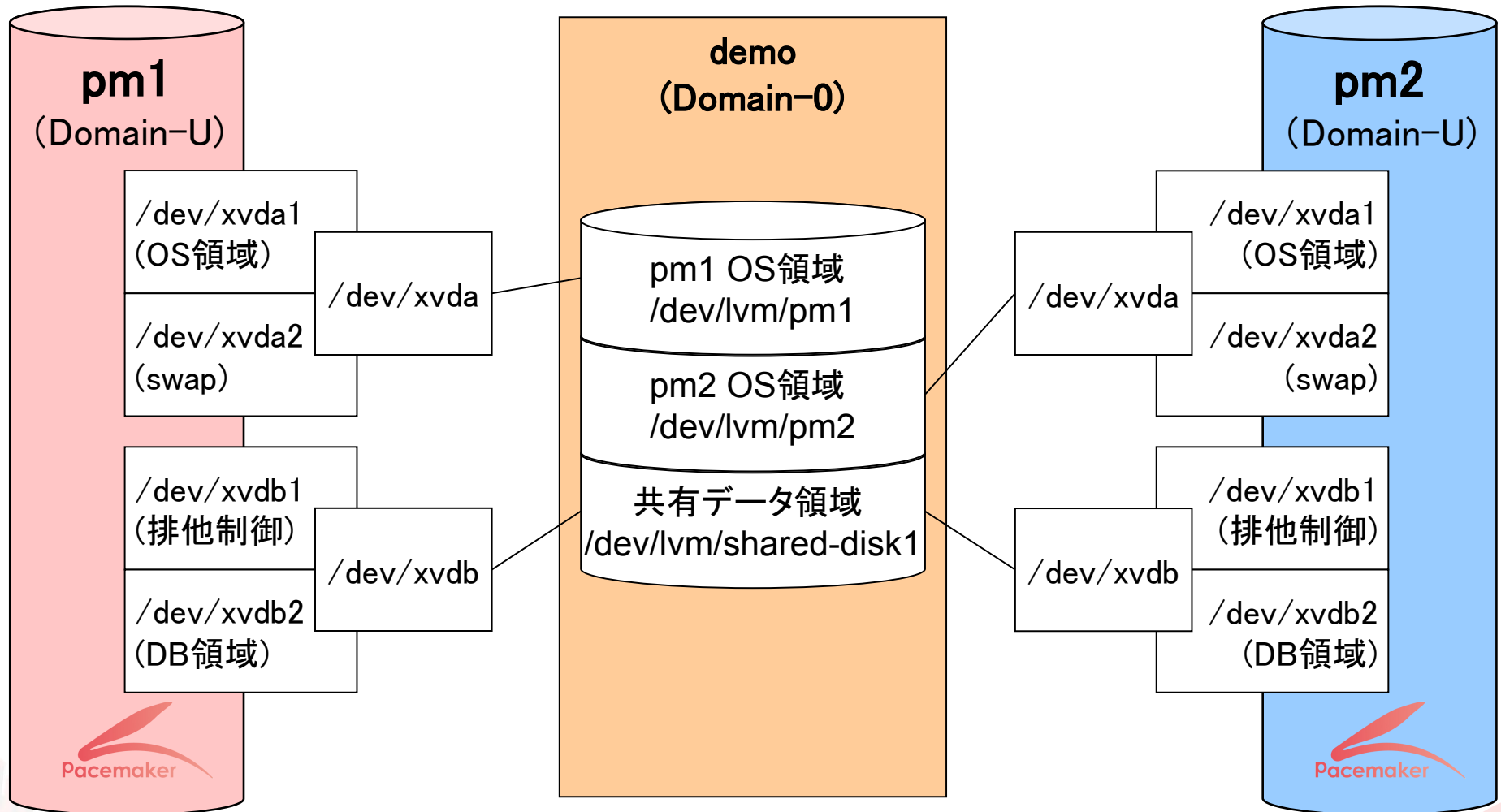
demo  
(Domain-0)



# Pacemakerデモ機構成(Xen仮想NW)



# Pacemakerデモ機構成(Xen仮想ディスク)



# Pacemakerデモ クラスタ制御部設定

## /etc/ha.d/ha.cf

```
pacemaker on
debug 0
udpport 694
keepalive 2
warntime 7
deadtime 10
initdead 48
logfacility local1

bcast eth1
bcast eth2

node pm1
node pm2

watchdog /dev/watchdog
respawn root /usr/lib64/heartbeat/ifcheckd
```

クラスタ制御部の基本設定ファイルで、クラスタ内の全サーバに同じ内容のファイルを設置します。

pm\_extrasをインストールしifcheckdの設定を追加すればインターコネクトLANの接続情報等が確認可能になります。

# Pacemakerデモ リソース構成

これら4つの  
リソースは  
グループ設定します

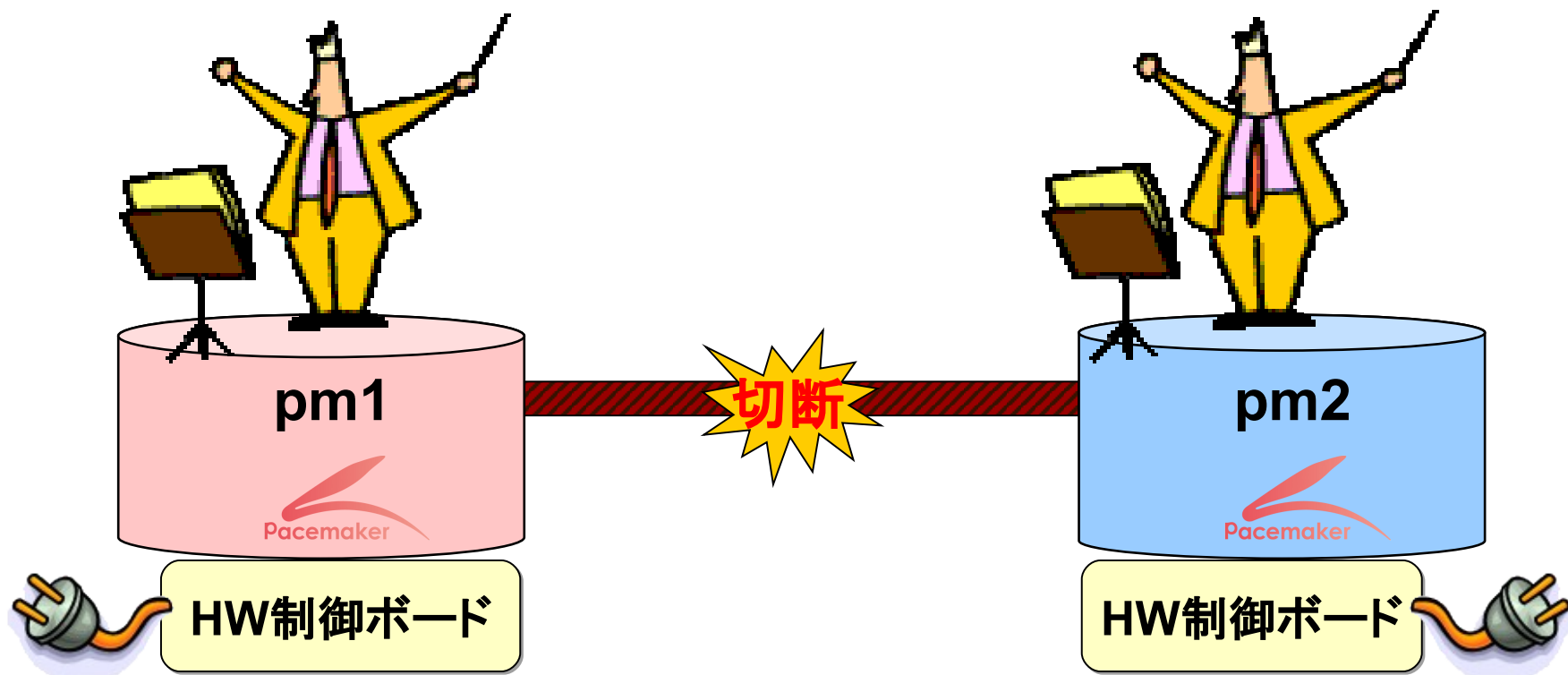
- ディスク排他制御 (sfex)
  - 共有ディスクの排他制御を行います
- DBデータ領域マウント (Filesystem)
  - 共有ディスクにあるDBデータ領域のマウント制御を行います
- 仮想IP割り当て (IPaddr2)
  - サービス提供用の仮想IPを割り当てます
- PostgreSQL制御 (pgsql)
  - PostgreSQL 9.0.4 の制御を行います

本日はSTONITH  
のデモも行います

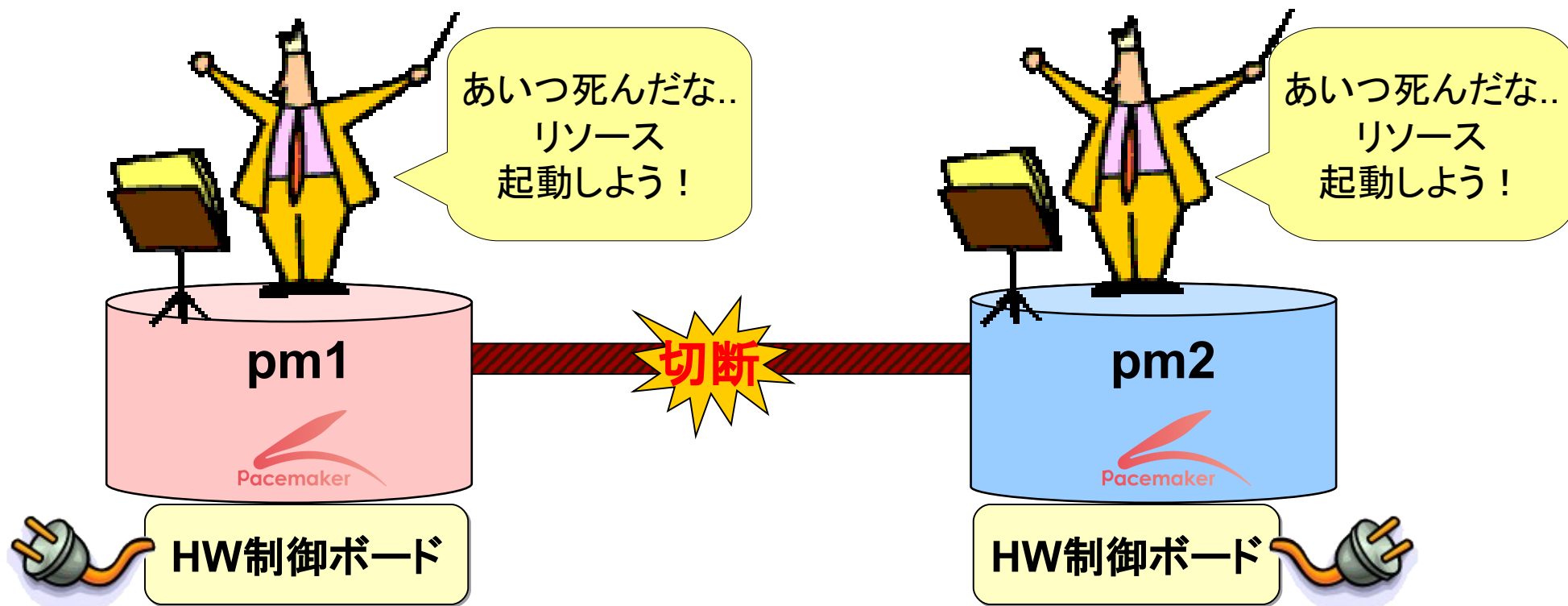
- STONITH (stonith-helper, xen0, meatclient)
  - STONITHは「**S**hoot **T**he **O**ther **N**ode **I**n **T**he **H**ead」の略で監視対象サーバの異常を検出したときに、強制的にそのサーバをダウンさせるノードフェンシングを行います。
- ネットワーク監視 (pingd)
  - 指定したIPアドレスに ping送信し、ネットワーク疎通があるかどうかの監視を行います。
- ディスク監視 (diskd)
  - 指定したディスクデバイスにアクセスし、ディスクの正常性確認を行います。



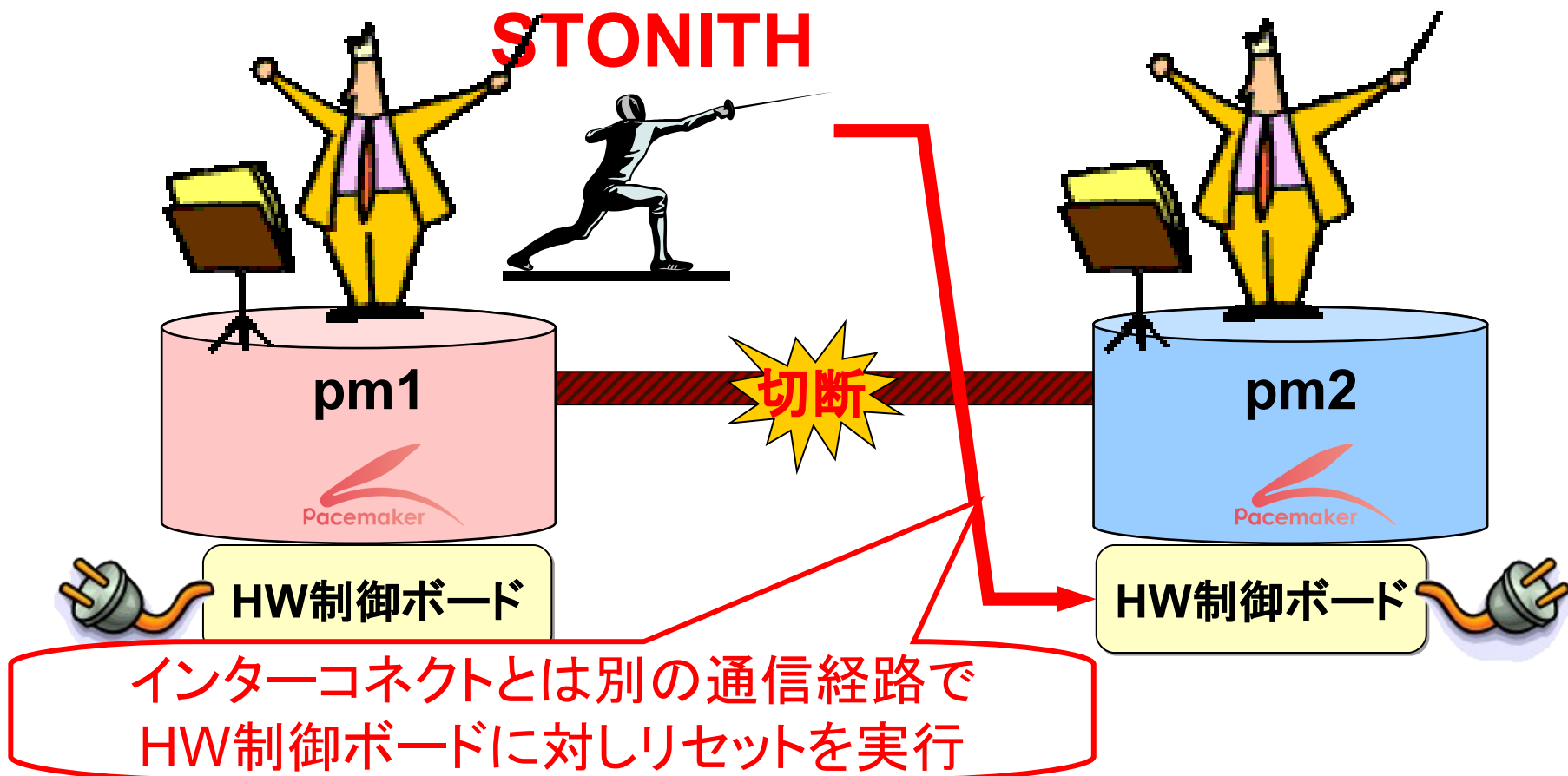
# STONITH実行例(スプリットブレイン)



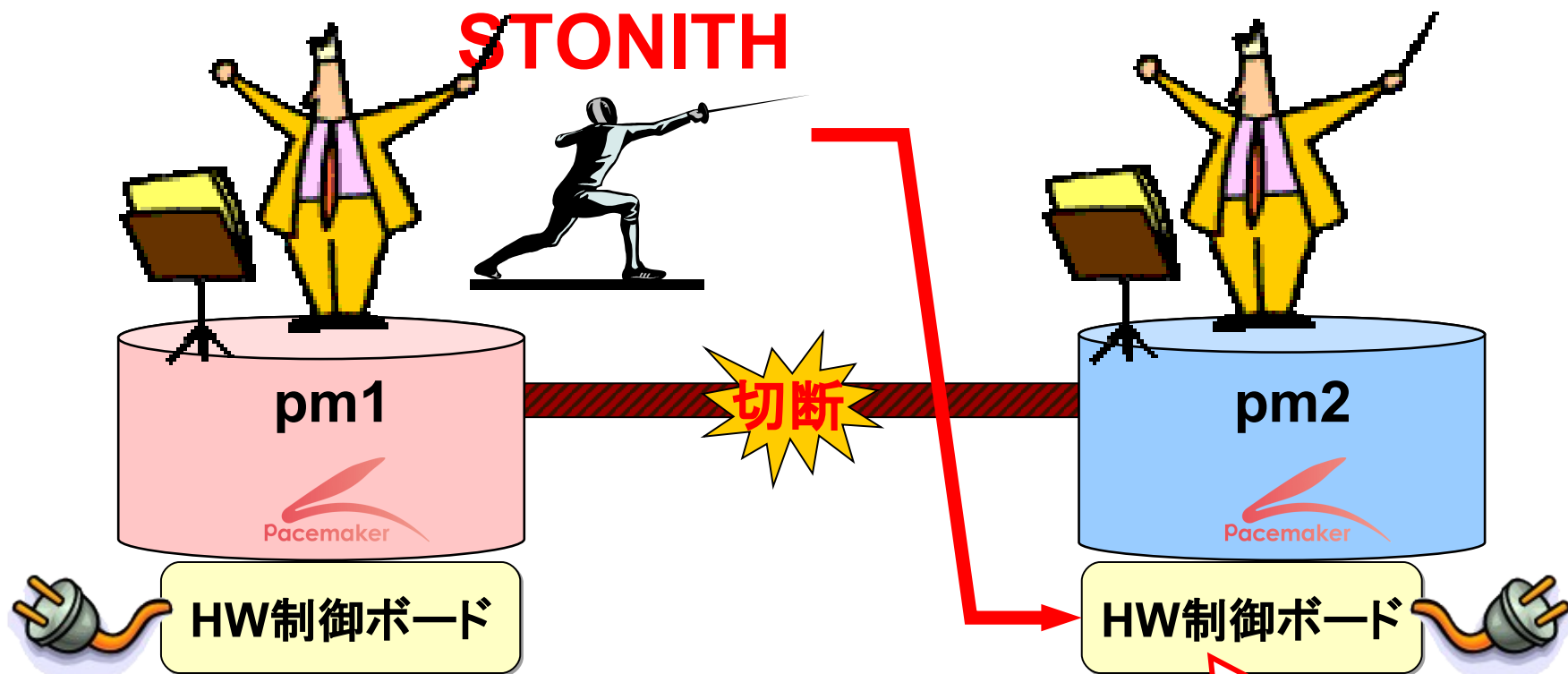
# STONITH実行例(スプリットブレイン)



# STONITH実行例(スプリットブレイン)

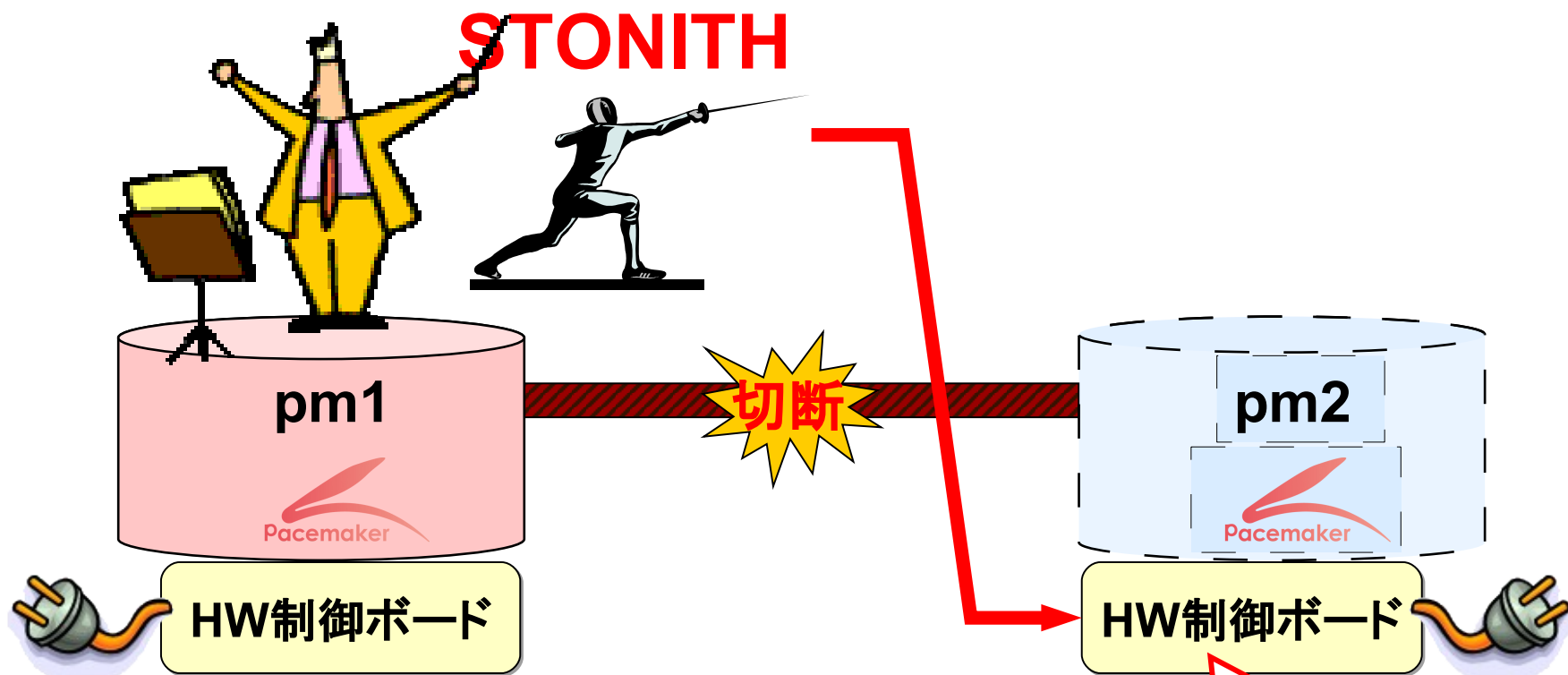


# STONITH実行例(スプリットブレイン)



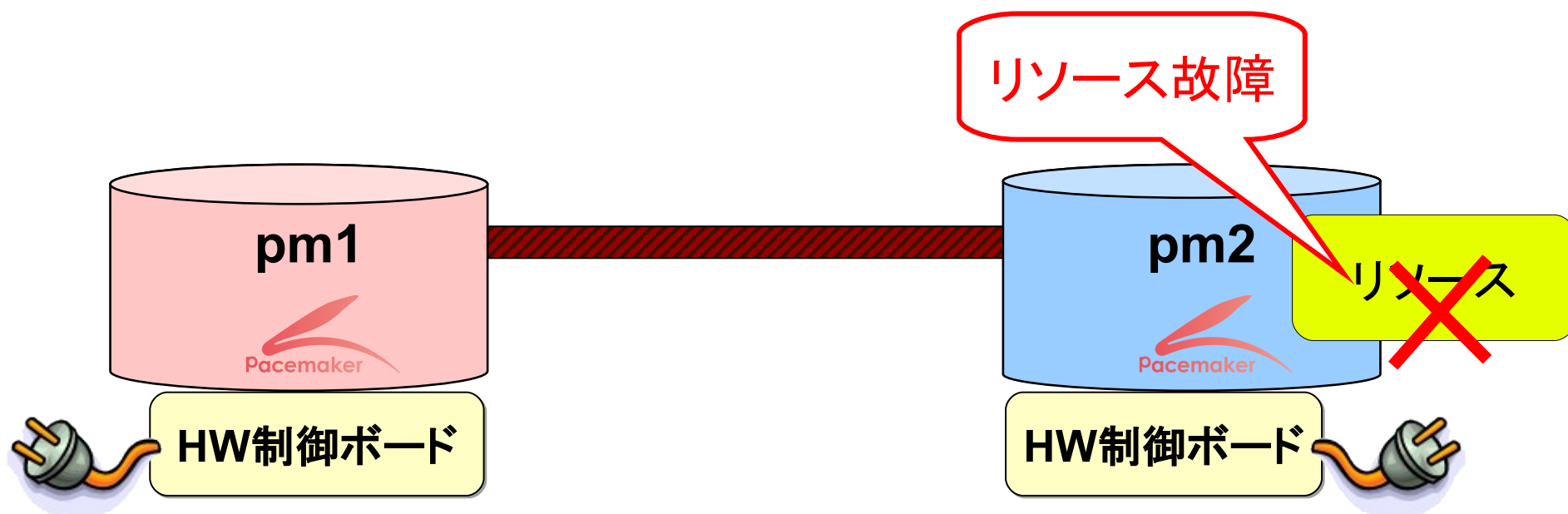
OSと連動しないHW制御ボードから強制電源断

# STONITH実行例(スプリットブレイン)

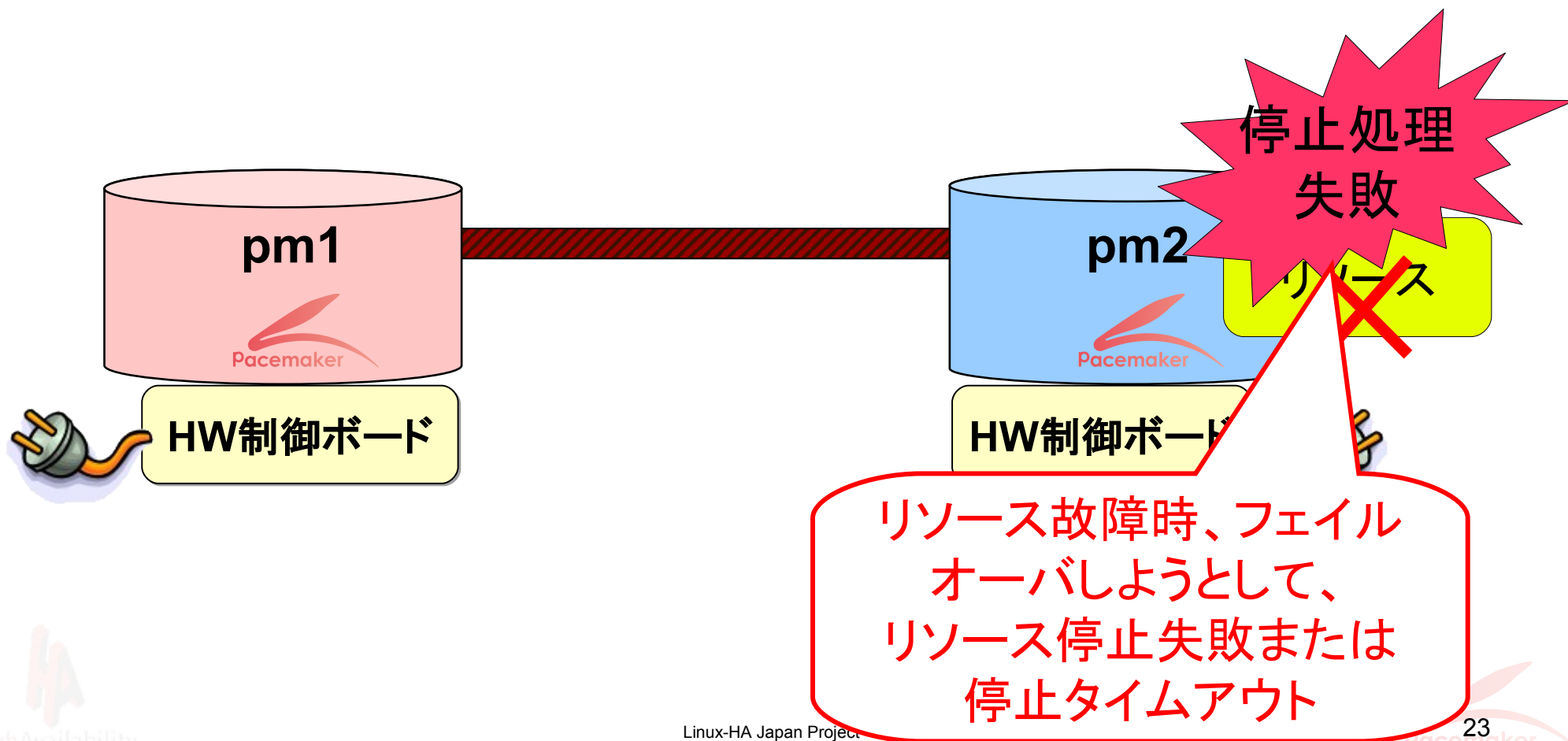


OSと連動しないHW制御ボードから強制電源断

# STONITH実行例(リソース停止失敗)

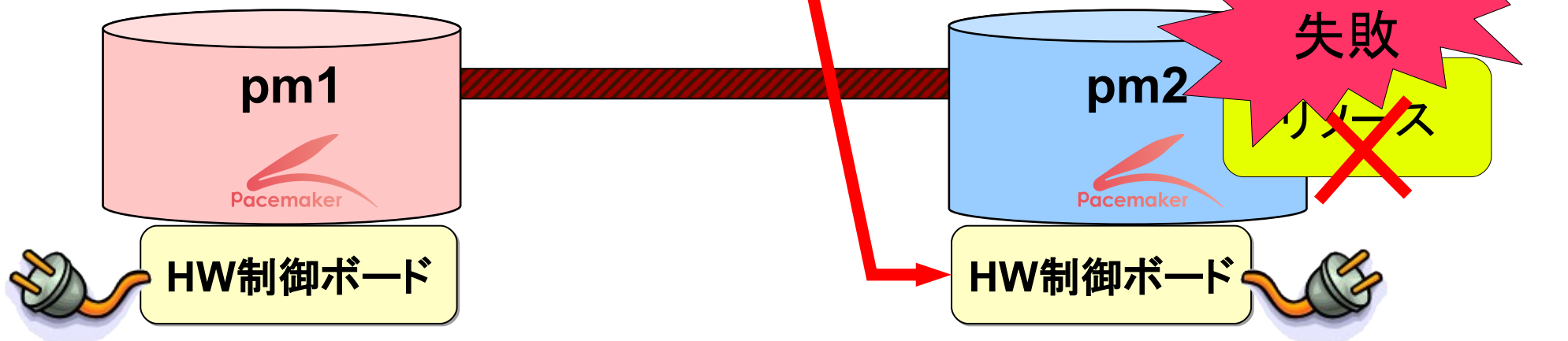


# STONITH実行例(リソース停止失敗)



# STONITH実行例(リソース停止失敗)

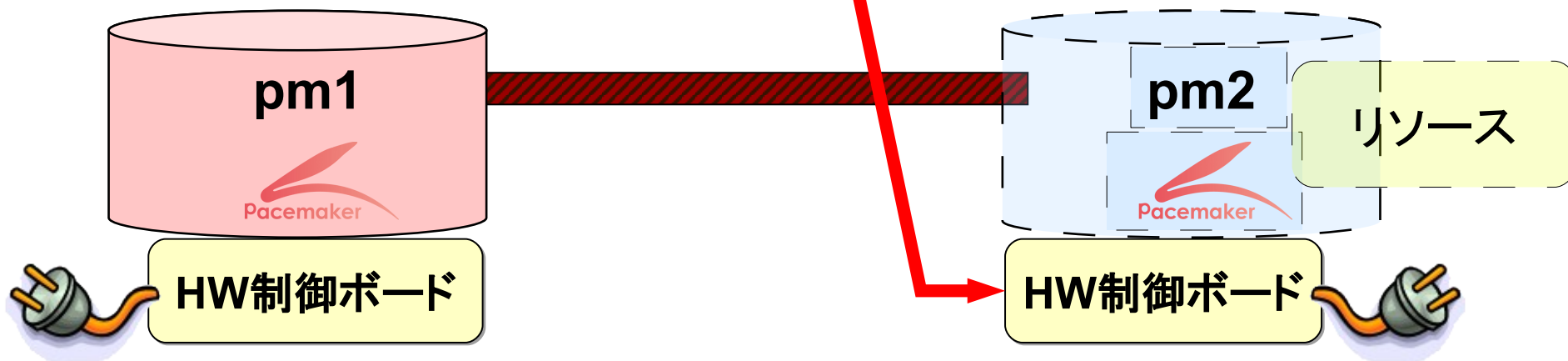
**STONITH**





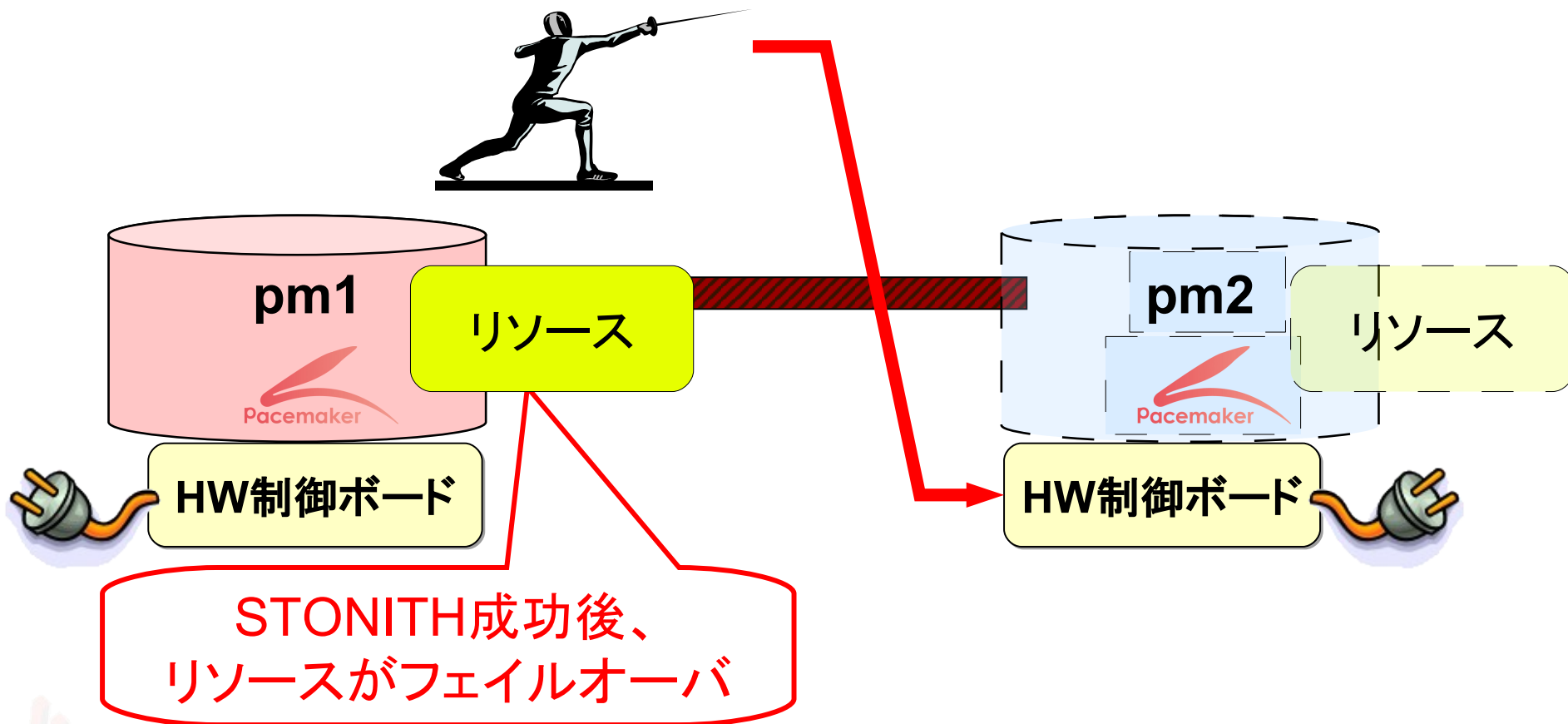
# STONITH実行例(リソース停止失敗)

**STONITH**

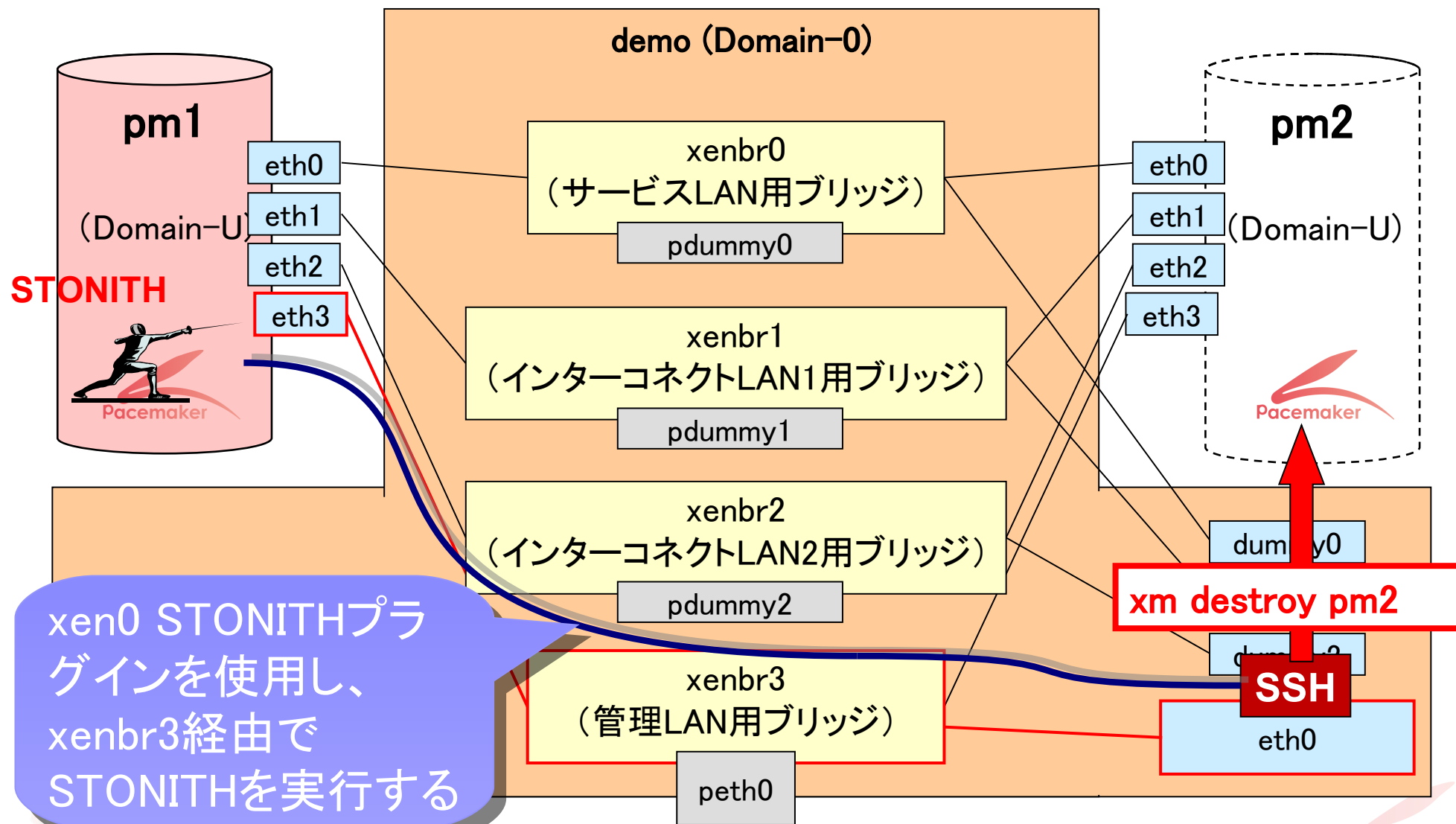


# STONITH実行例（リソース停止失敗）

# STONITH



# Pacemakerデモ機フェンシング (STONITH) 構成





③

crm\_mon コマンド  
を使おう！



# crm\_monとは？

crm\_monは、Pacemakerが制御しているサーバ状態やリソース状態、インターコネクトLAN、ネットワークの状態を確認できるコマンドです。

```
# crm_mon
```

# crm\_mon実行例

```
=====
Last updated: Fri Jul  8 16:47:51 2011
Stack: Heartbeat
Current DC: pm2 (7f1b5dcb-e696-414d-8fca-da79274b0a74) - partition with quorum
Version: 1.0.11-1554a83db0d3c3e546cfd3aaff6af1184f79ee87
2 Nodes configured, unknown expected votes
6 Resources configured.
```

```
=====
Online: [ pm1 pm2 ]
```

Resource Group: grpPg

```
  prmEx      (ocf::heartbeat:sfex):  Started pm1
  prmFs      (ocf::heartbeat:Filesystem):  Started pm1
  prmIp      (ocf::heartbeat:IPaddr2):  Started pm1
  prmPg      (ocf::heartbeat:pgsql):  Started pm1
```

Resource Group: grpStonith1

```
  prmStonith1-1 (stonith:external/stonith-helper):  Started pm2
  prmStonith1-2 (stonith:external/xen0):  Started pm2
  prmStonith1-3 (stonith:meatware):  Started pm2
```

Resource Group: grpStonith2

```
  prmStonith2-1 (stonith:external/stonith-helper):  Started pm1
  prmStonith2-2 (stonith:external/xen0):  Started pm1
  prmStonith2-3 (stonith:meatware):  Started pm1
```

Clone Set: clnDiskd1

```
  Started: [ pm1 pm2 ]
```

Clone Set: clnDiskd2

```
  Started: [ pm1 pm2 ]
```

Clone Set: clnPingd

```
  Started: [ pm1 pm2 ]
```

# サーバ状態表示

Pacemakerが稼動しているサーバは「Online」と表示され、停止しているサーバは「OFFLINE」と表示されます。

=====

Last updated: Fri Jul 8 16:47:51 2011

Stack: Heartbeat

Current DC: pm2 (7f1b5dcb-e696-414d-8fca-da79274b0a74) -  
partition with quorum

Version: 1.0.11-1554a83db0d3c3e546cfd3aaff6af1184f79ee87

2 Nodes configured, unknown expected votes

6 Resources configured.

=====

Online: [ pm1 pm2 ]

クラスタに組み込まれている  
サーバ名が表示されます

# リソース状態表示(サービス系リソース)

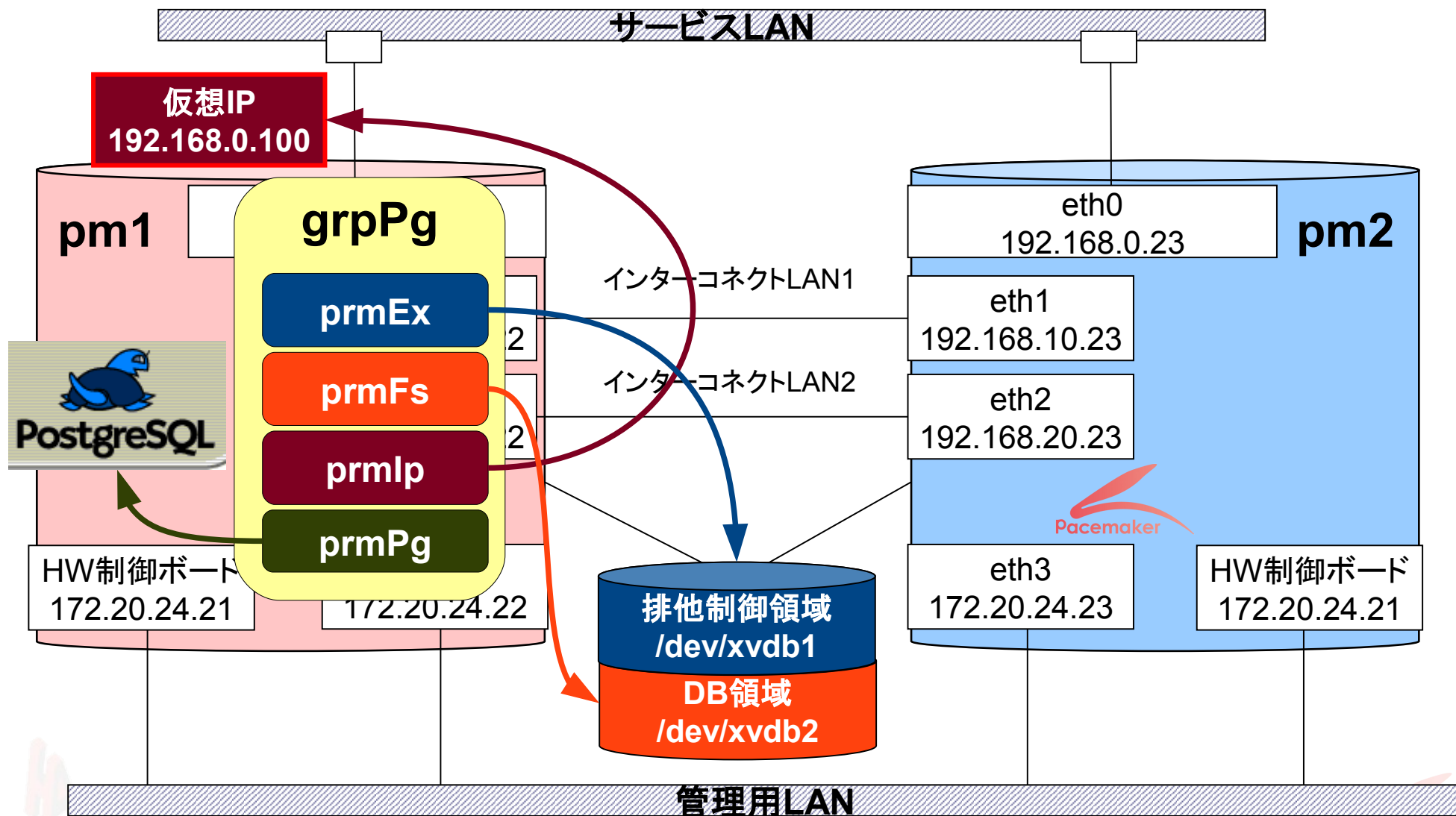
Pacemakerが制御しているリソースの状態が表示されます。  
リソース稼動状態と稼働中のサーバ名が「**Started サーバ名**」  
などと表示されます。

```
Resource Group: grpPg
  prmEx      (ocf::heartbeat:sfex):   Started pm1
  prmFs      (ocf::heartbeat:Filesystem): Started pm1
  prmIp      (ocf::heartbeat:IPaddr2): Started pm1
  prmPg      (ocf::heartbeat:pgsql):  Started pm1
```

- prmEx: ディスク排他制御 (sfex)
- prmFs: DBデータ領域マウント (Filesystem)
- prmIp: 仮想IP割り当て (IPaddr2)
- prmPg: PostgreSQL制御 (pgsql)



# サービス系リソース



# リソース状態表示 (STONITHリソース)

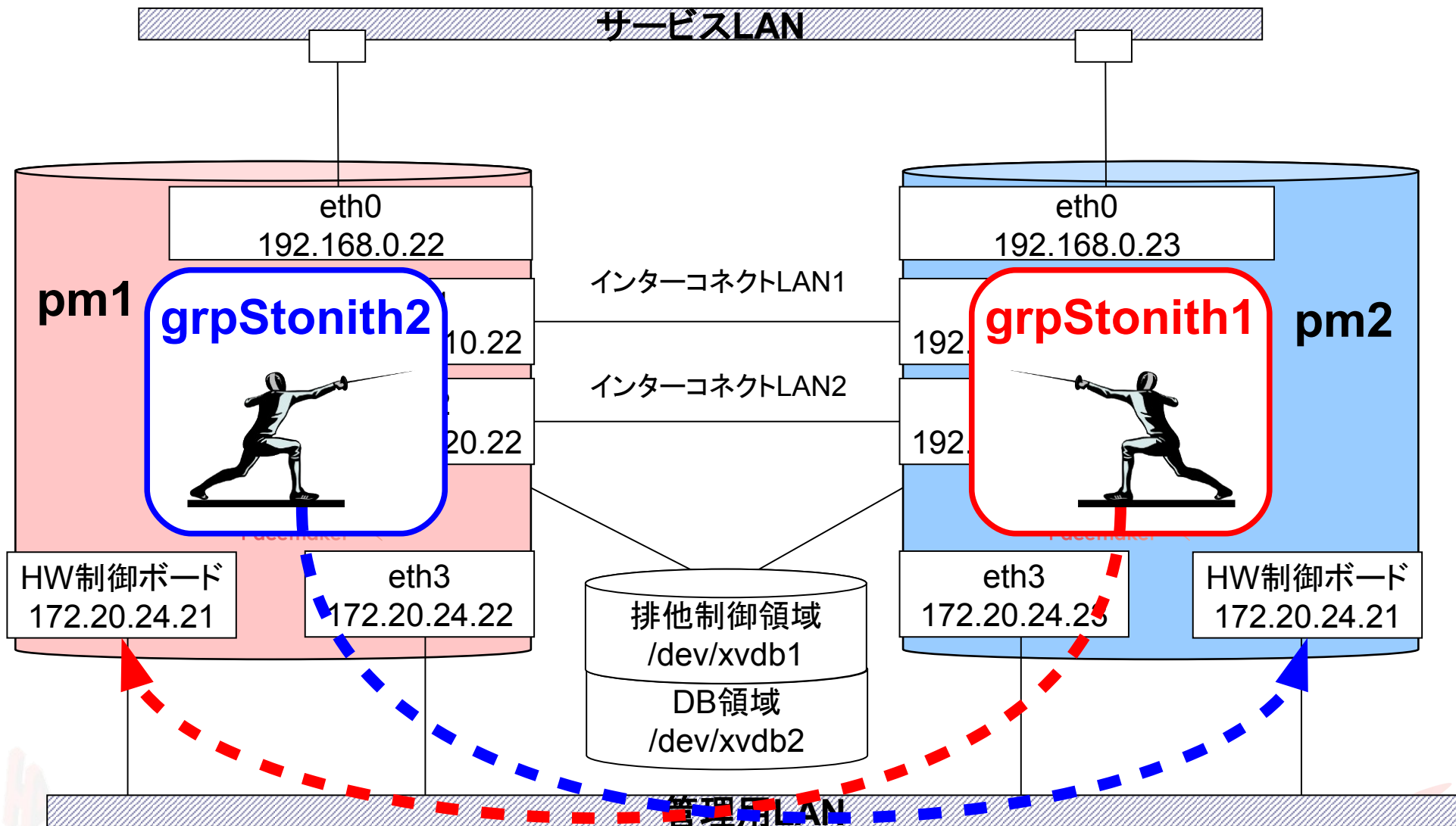
サービス系リソースと同様に、「Started サーバ名」と表示されます。

```
Resource Group: grpStonith1
  prmStonith1-1  (stonith:external/stonith-helper):  Started pm2
  prmStonith1-2  (stonith:external/xen0):           Started pm2
  prmStonith1-3  (stonith:meatware):                 Started pm2
```

- prmStonith1-1: サーバ断確認、相打防止プラグイン (stonith-helper)
- prmStonith1-2: Xen用フェンシングプラグイン (xen0)
- prmStonith1-3: 停止通知用プラグイン (meatware)

このデモでは、grpStonith1 は pm1をフェンシングするSTONITHリソースのため、pm2で稼動しているのが確認できます。

# STONITHリソース



# リソース状態表示(監視系リソース)

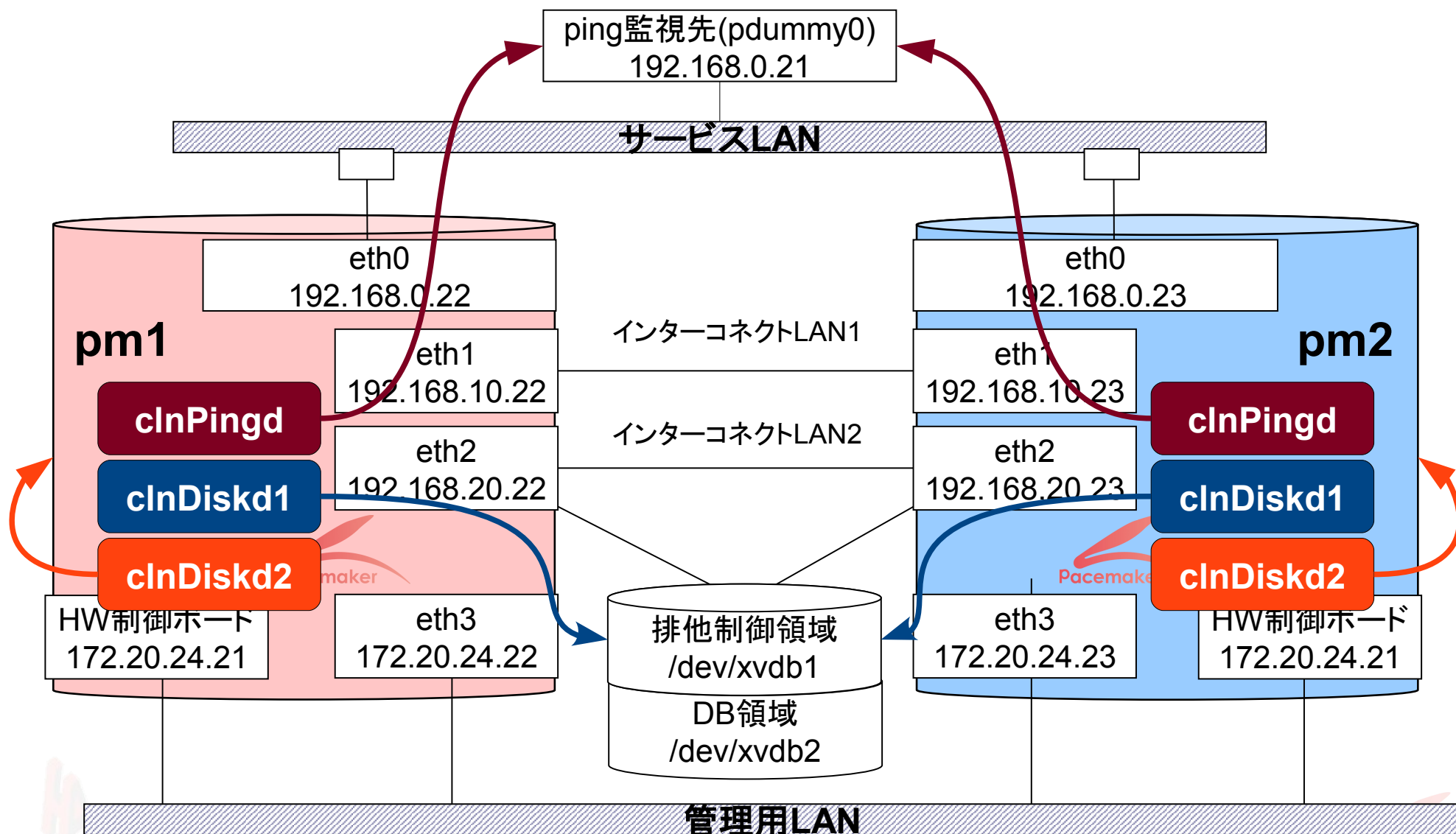
ディスク状態監視、ネットワーク状態監視は、両方のサーバで動作させるように Cloneで登録します。

「Started: [ pm1 pm2 ]」などと表示され、リソース稼動状態と稼働中のサーバ名がわかります。

```
Clone Set: clnDiskd1
  Started: [ pm1 pm2 ]
Clone Set: clnDiskd2
  Started: [ pm1 pm2 ]
Clone Set: clnPingd
  Started: [ pm1 pm2 ]
```

- clnDiskd1: 共有ディスク監視 (diskd)
- clnDiskd2: 内蔵ディスク監視 (diskd)
- clnPingd: ネットワーク監視 (pingd)

# 監視系リソース



**-fA** オプションを付与すると、インターコネクト LAN等の状況も確認可能です。

```
# crm_mon -fA
```

```
* Node pm1:
```

+ default_ping_set	: 100
+ diskcheck_status	: normal
+ diskcheck_status_internal	: normal
+ pm2-eth1	: up
+ pm2-eth2	: up

ネットワーク監視  
の状況

ディスク監視  
の状況

インターコネクトがUPされ  
ているのが確認可能

④

ログメッセージ制御機能  
を使おう！

# ログメッセージ制御機能

pm\_logconv-hb

6/6 に pm\_logconv-hb 1.1版  
をリリース

Linux-HA Japanで  
ログメッセージ制御機能を提供中！

Pacemaker標準ログ(ha-log)は出力が多くわかりにくいですが、pm\_logconv-hb を使用すると、運用上必要なログだけを出力することができます。

さらにフェイルオーバーが発生した際に、「Start Fail-over」のログが出力されるようになります。

※クラスタ制御部がHeartbeat3である必要性があります。(Corosyncは未対応です)



# インストール

リポジトリパッケージから yum コマンドで pacemaker インストールの際に、pm\_logconv-hb も指定してインストールします。

```
# cd /tmp
# tar zxvf pacemaker-1.0.11-1.2.1.el5.x86_64.repo.tar.gz
# cd /tmp/pacemaker-1.0.11-1.2.1.el5.x86_64.repo/
# yum -c pacemaker.repo install pacemaker pm_crmgen pm_diskd  
pm_logconv-hb pm_extras
```

- pm\_crmgen-1.1-1.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-1.1-1.el5.noarch.rpm ... ログメッセージ制御機能
- pm\_extras-1.1-1.el5.x86\_64.rpm ... その他オリジナルRA 等

# 動作設定

/etc/pm\_logconv.conf

[Settings]

**ha\_log\_path** = /var/log/ha-log

**output\_path** = /var/log/pm\_logconv.out

#hostcache\_path = /var/lib/heartbeat/hostcache

#syslogformat = True

#reset\_interval = 60

attribute\_pingd = not\_defined default\_ping\_set or default\_ping\_set lt 100

attribute\_diskd = not\_defined diskcheck\_status or diskcheck\_status eq ERROR

attribute\_diskd\_inner = not\_defined diskcheck\_status\_internal or diskcheck\_status\_internal eq ERROR

#logconv\_logfacility = daemon

**act\_rsc** = prmEx, prmPg

変換元のログファイル名を指定

変換後のログファイル名を指定

サービスリソースの最上位と最下位のリソースIDを設定

pm\_logconv-hb

# 起動設定

inittabに pm\_logconv-hb 起動設定を追加し、  
respawnで起動させます。

## /etc/inittab

(省略)

:

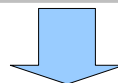
```
logc:2345:respawn:/usr/share/pacemaker/pm_logconv/pm_logconv.py
```

# ログ変換例 (PostgreSQL起動時)

/var/log/ha-log

```
Jul 11 18:53:34 pm1 crmd: [1996]: info: do_lrm_rsc_op: Performing key=18:14:0:54ec38e9-bfac-4b29-9256-a9b9587456c6  
op=prmPg_start_0 )  
Jul 11 18:53:34 pm1 lrmd: [1993]: info: rsc:prmPg:63: start  
Jul 11 18:53:34 pm1 crmd: [1996]: info: process_lrm_event: LRM operation prmIp_monitor_10000 (call=62, rc=0, cib-update=68,  
confirmed=false) ok  
Jul 11 18:53:35 pm1 pgsql[19130]: INFO: server starting  
Jul 11 18:53:35 pm1 pgsql[19130]: INFO: PostgreSQL start command sent.  
Jul 11 18:53:35 pm1 pgsql[19130]: WARNING: psql: could not connect to server: No such file or directory Is the server  
running locally and accepting connections on Unix domain socket "/tmp/.s.PGSQL.5432"?  
Jul 11 18:53:35 pm1 pgsql[19130]: WARNING: PostgreSQL template1 isn't running  
Jul 11 18:53:35 pm1 pgsql[19130]: WARNING: Connection error (connection to the server went bad and the session was not  
interactive) occurred while executing the psql command.  
Jul 11 18:53:37 pm1 pgsql[19130]: INFO: PostgreSQL is started.  
Jul 11 18:53:37 pm1 crmd: [1996]: info: process_lrm_event: LRM operation prmPg_start_0 (call=63, rc=0, cib-update=69,  
confirmed=true) ok
```

/var/log/pm\_logconv.out



運用上必要なログだけを出力

```
Jul 11 18:53:34 pm1 info: Resource prmPg tries to start.  
Jul 11 18:53:37 pm1 info: Resource prmPg started. (rc=0)
```

# フェイルオーバー時のログ出力例

/var/log/pm\_logconv.out

```
Jul 11 19:02:15 pm2 ERROR: Start to fail-over.  
Jul 11 19:02:23 pm2 info: Resource prmEx tries to start.  
Jul 11 19:02:24 pm2 info: Resource prmEx started. (rc=0)  
Jul 11 19:02:24 pm2 info: Resource prmFs tries to start.  
Jul 11 19:02:24 pm2 info: Resource prmFs started. (rc=0)  
Jul 11 19:02:24 pm2 info: Resource prmIp tries to start.  
Jul 11 19:02:24 pm2 info: Resource prmIp started. (rc=0)  
Jul 11 19:02:24 pm2 info: Resource prmPg tries to start.  
Jul 11 19:02:26 pm2 info: Resource prmPg started. (rc=0)  
Jul 11 19:02:26 pm2 info: Resource prmEx : Move pm1 -> pm2  
Jul 11 19:02:26 pm2 info: Resource prmPg : Move pm1 -> pm2  
Jul 11 19:02:26 pm2 info: fail-over succeeded.
```

※ fail-overのログは、DCノード側のみ出力されます。

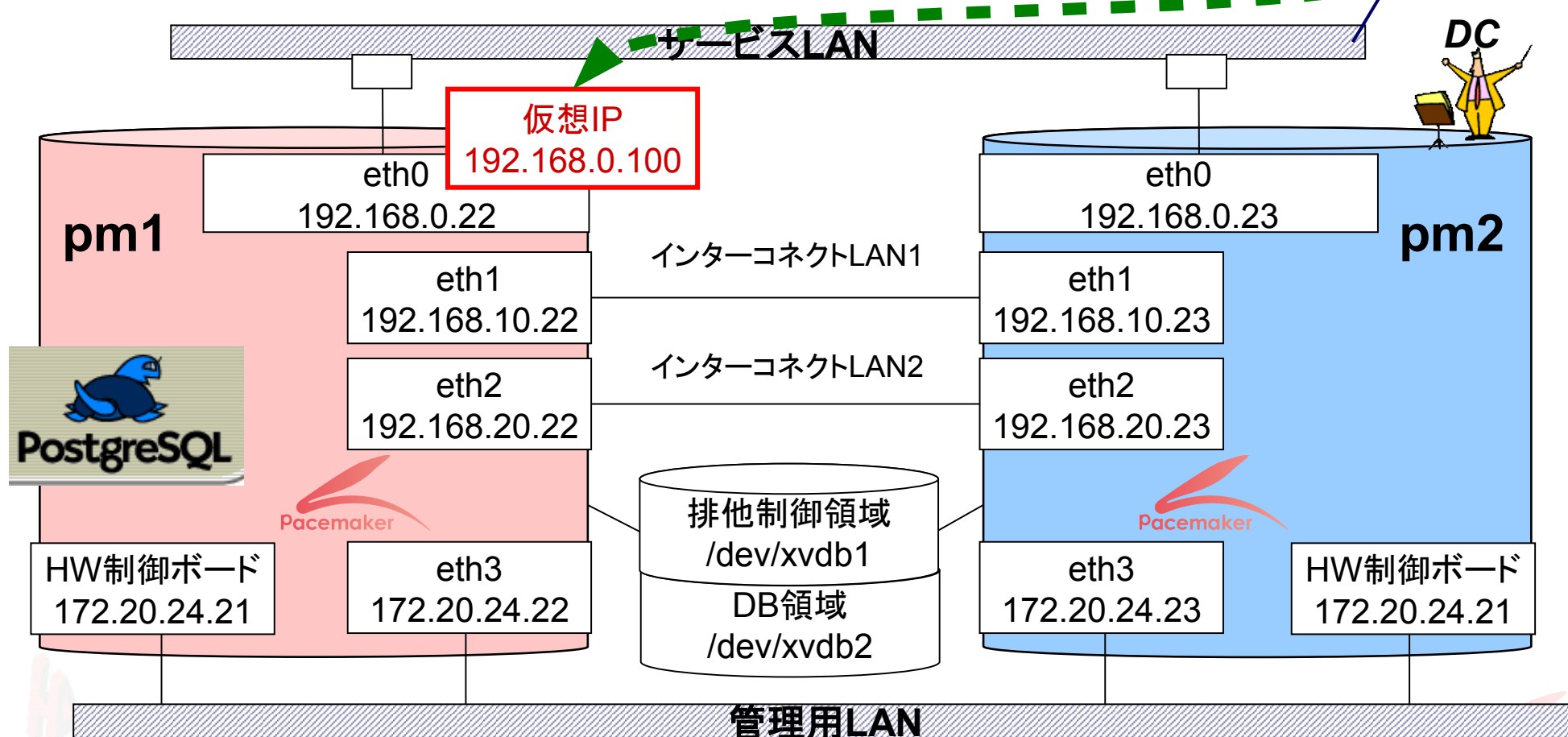
⑤

いろいろデモします！

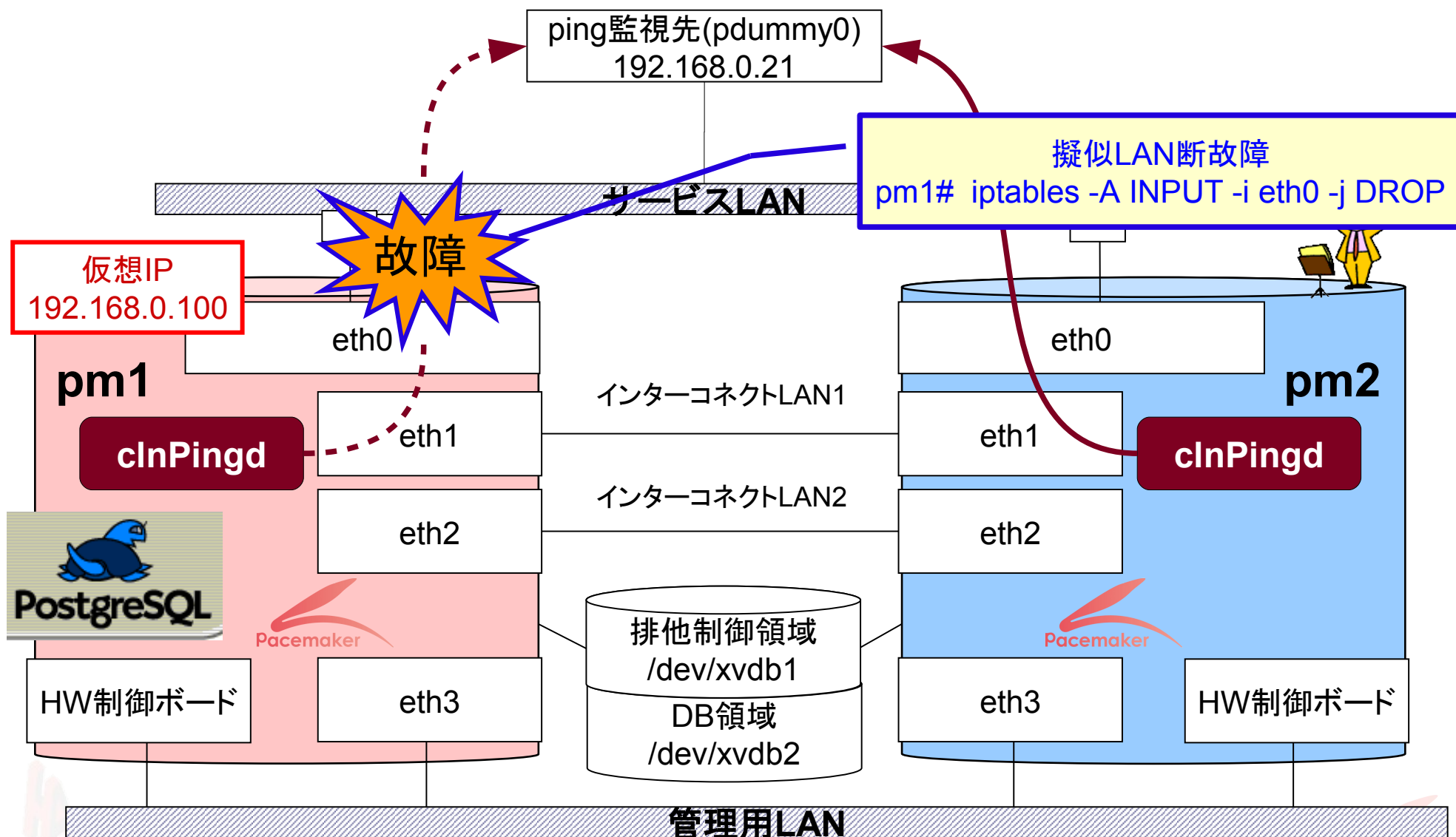
# リソース設定して PostgreSQLに接続してみる…

```
demo# pgsql -U postgres -h 192.168.0.100 -c "SELECT now();"
```

demo(Domain-0)



# サービスLAN故障させてみる...





```
# crm_mon -fA
```

～ 省略 ～

Node Attributes:

\* Node pm1:

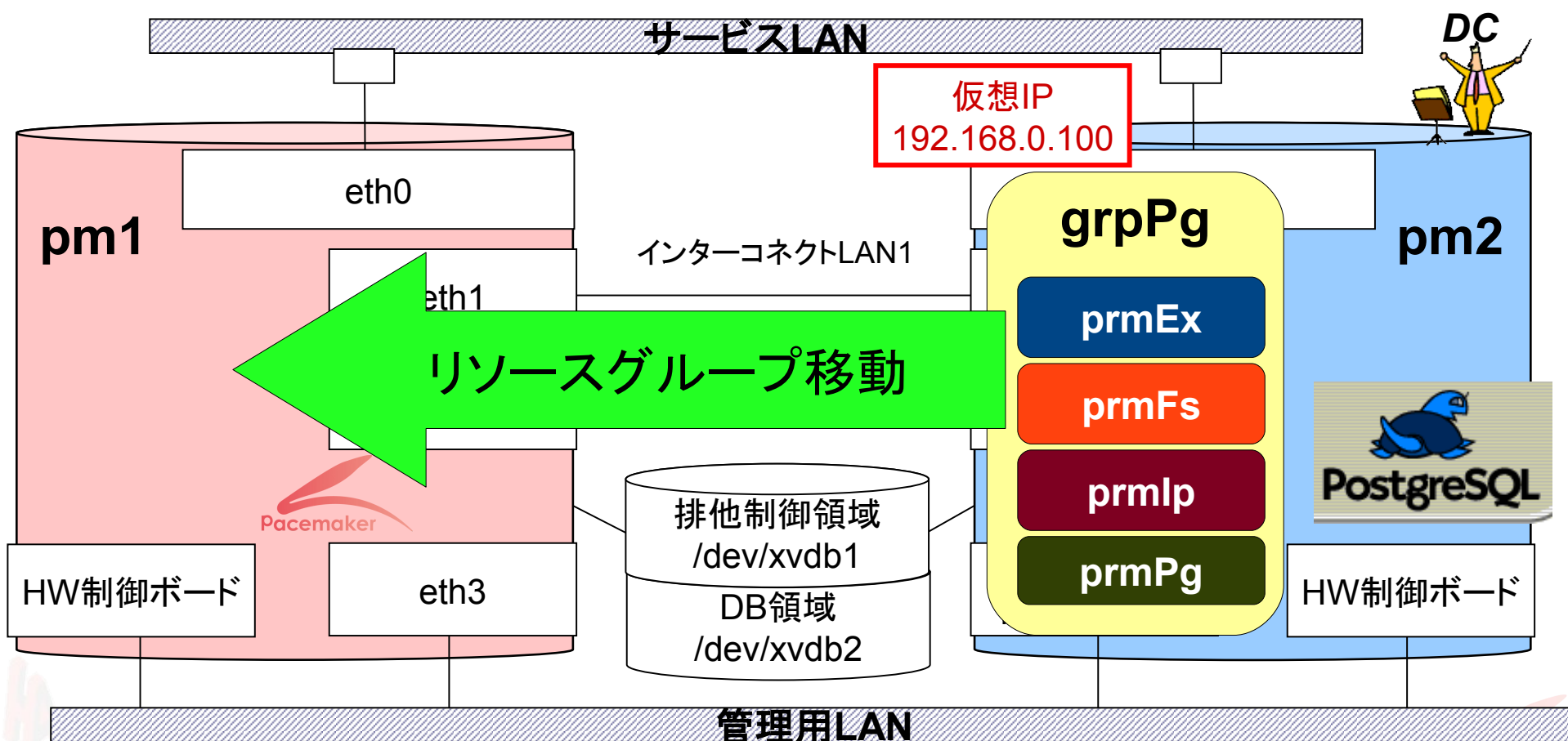
+ default_ping_set	: 0	: Connectivity is lost
+ diskcheck_status	: normal	
+ diskcheck_status_internal	: normal	

サービスLAN故障  
を表示

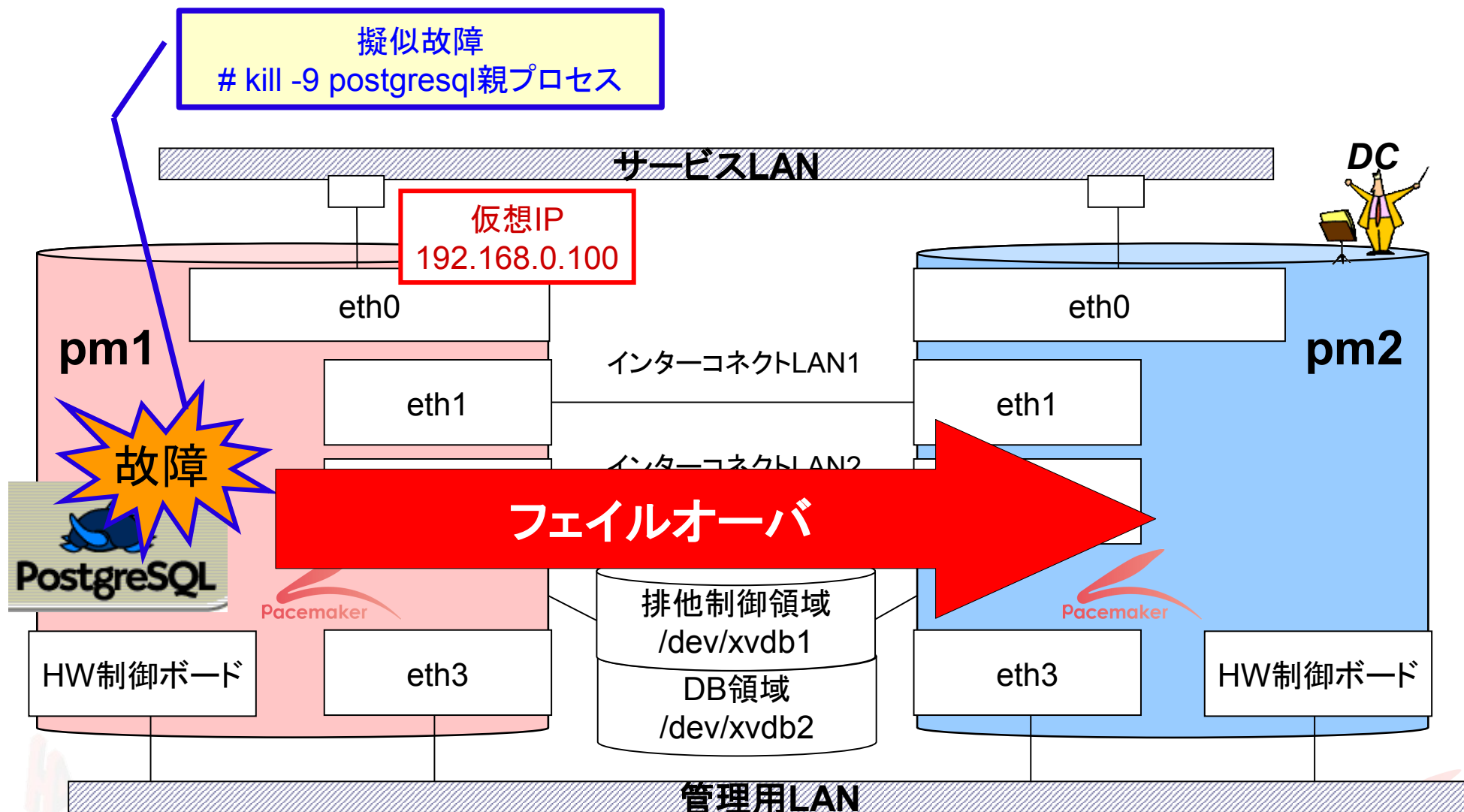
# リソースグループ移動させてみる…

```
# crm resource move grpPg pm1 force
```

```
# crm resource unmove grpPg
```



# リソース故障させてみる…



```
# crm_mon -fA
```

～ 省略 ～

Online: [ pm1 pm2 ]

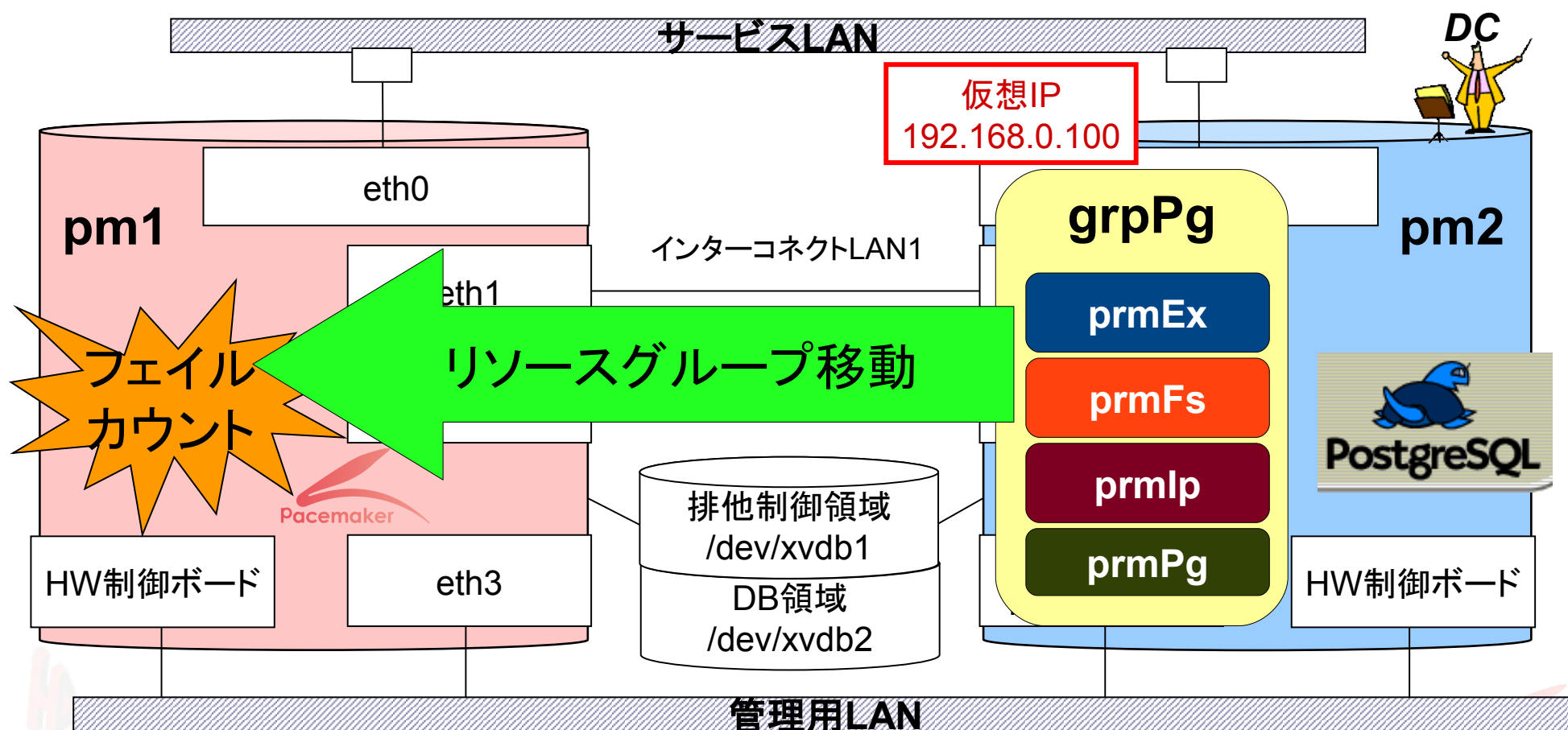
Resource Group: grpPg

prmEx	(ocf::heartbeat:sfex):	Started pm1
prmFs	(ocf::heartbeat:Filesystem):	Started pm1
prmIp	(ocf::heartbeat:IPaddr2):	Started pm1
prmPg	(ocf::heartbeat:pgsql):	Started pm1 FAILED

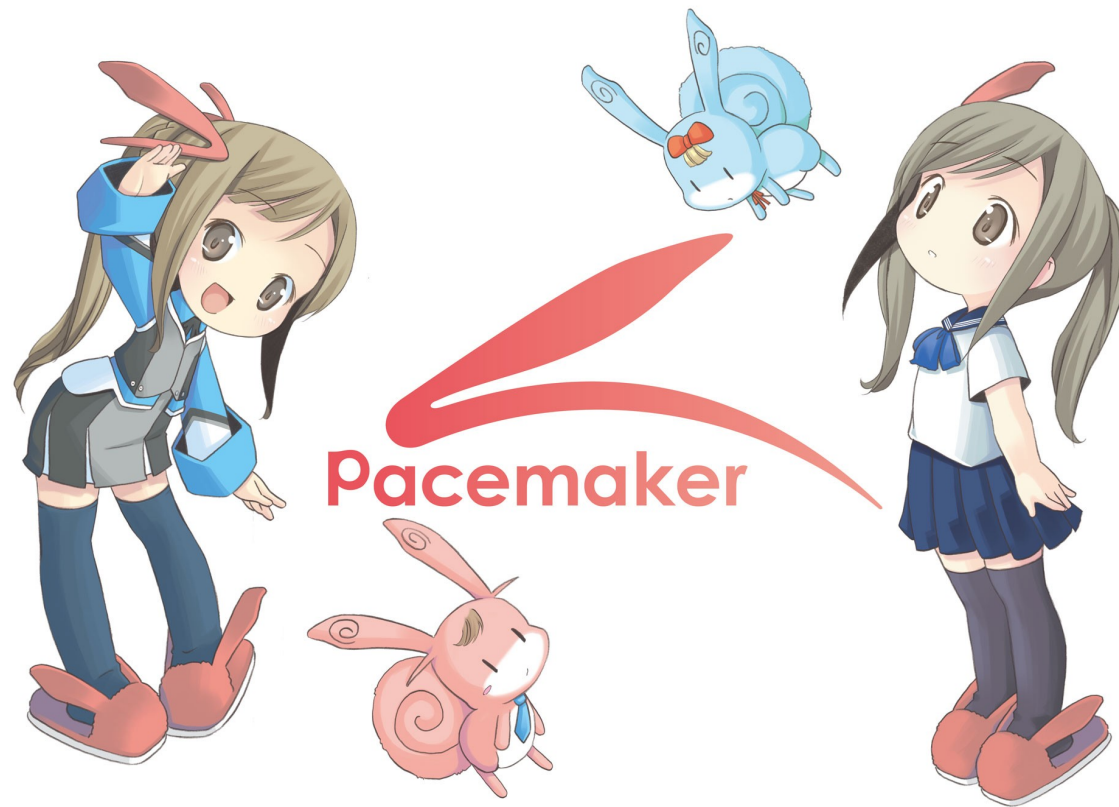
故障検出を表示

# この状態で リソースグループ移動させてみる...

```
# crm resource move grpPg pm1 force
```



切り替わらないのは  
ミスではありません！



フェイルカウントがカウントアップされているため、  
クリアしなければ切り替わりません。

```
# crm_mon -fA
```

```
=====
~ 省略 ~
=====
```

Migration summary:

\* Node pm1:

prnPg: migration-threshold=1 fail-count=1

\* Node pm2:

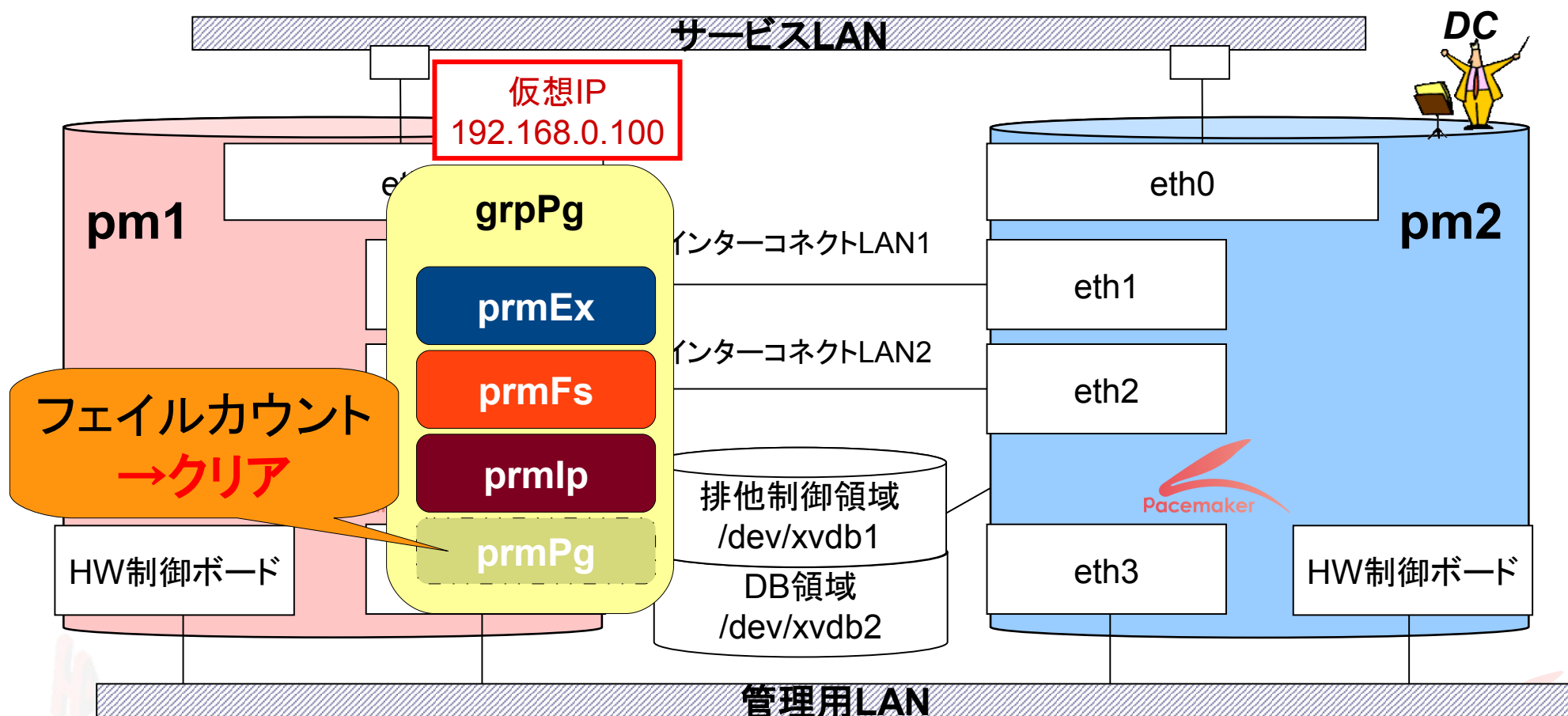
Failed actions:

prnPg\_monitor\_10000 (node=pm1, call=34, rc=7, status=complete):  
not running

# フェイルカントをクリアしてみる…

```
# crm resource cleanup prmPg pm1
```

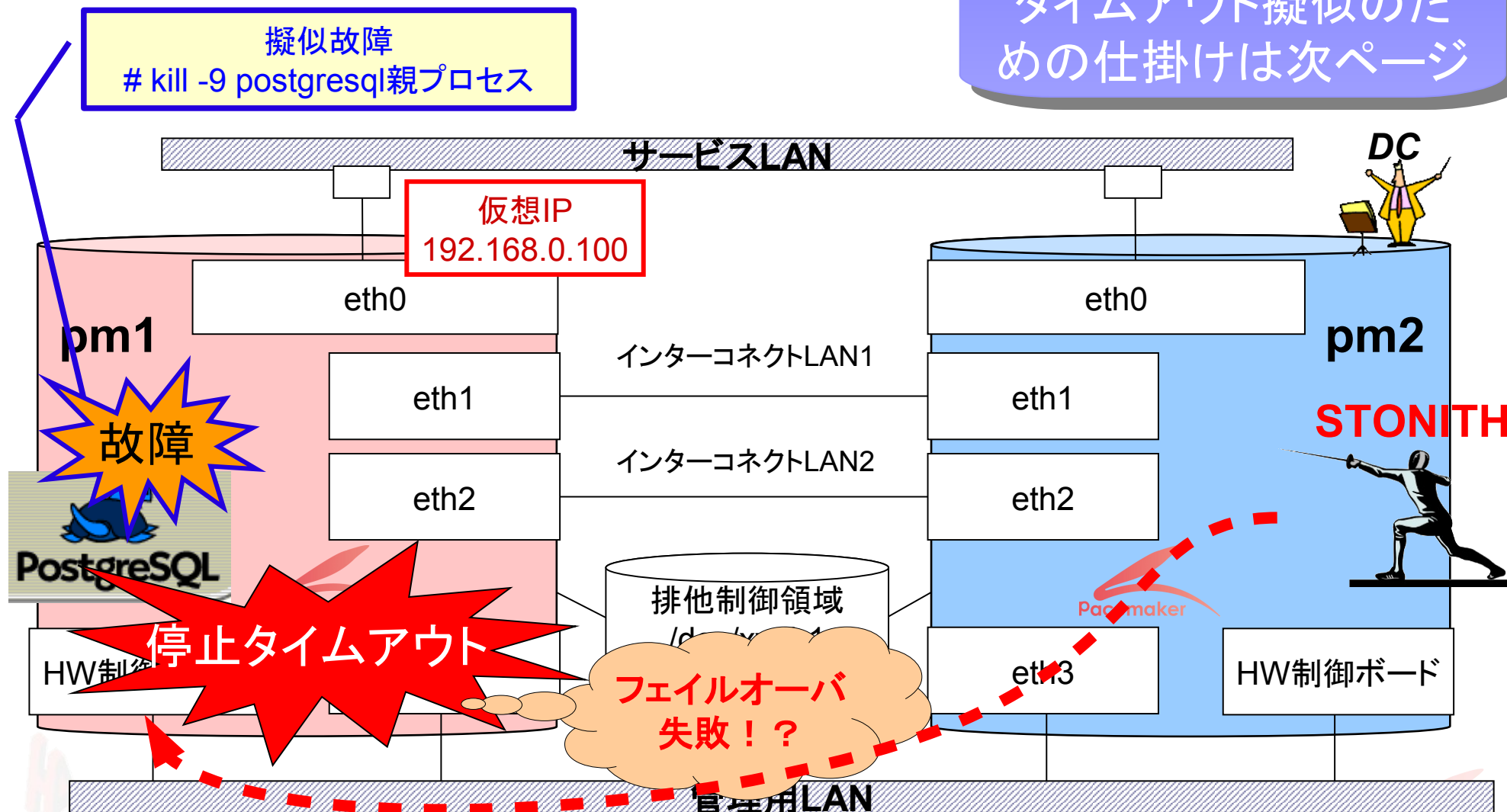
```
# crm resource unmove grpPg
```





# リソース故障時、停止タイムアウト...

タイムアウト擬似のための仕掛けは次ページ



# 停止タイムアウトデモのために、 こんな仕掛けします…

1. pgsqlリソースエージェントのstop制御部に sleep 60 をわざと入れます。

/usr/lib/ocf/resource.d/heartbeat/pgsql

```
381 #pgsql_stop: Stop PostgreSQL
382 pgsql_stop() {
383     local rc
384     sleep 60
385     if ! pgsql_status
386     then
387         #Already stopped
388         return $OCF_SUCCESS
389     fi
```

## 2. crmコマンドで prmPg のストップタイムアウトを 60s から 20s に変更します。

初期構築後、値を変更したい場合に便利です

```
# crm configure edit
```

```
primitive prmPg ocf:heartbeat:pgsql ¥  
  params pgctl="/usr/pgsql-9.0/bin/pg_ctl" start_opt="-p  
5432 -h 192.168.0.100" psql="/usr/pgsql-9.0/bin/psql"  
pgdata="/var/lib/pgsql/9.0/data" pgdba="postgres"  
pgport="5432" pgdb="template1" ¥  
  op start interval="0s" timeout="60s" on-fail="restart" ¥  
  op monitor interval="10s" timeout="60s" on-  
fail="restart" ¥  
  op stop interval="0s" timeout="20s" on-fail="fence"
```

```
# crm_mon -fA
```

Migration summary:

\* Node pm1:

prnPg: migration-threshold=1 fail-count=1

\* Node pm2:

Failed actions:

prnPg\_monitor\_10000 (node=pm1, call=34, rc=7,  
status=complete): not running

prnPg\_stop\_0 (node=pm1, call=35, rc=-2, status=Timed Out):  
unknown exec error

ストップタイムアウト状態  
を表示

# スプリットブレイン...

