

HAクラスタを フェイルオーバー失敗から 救おう！

2013年8月3日 OSC2013 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
- 趣味： マラソン
 - フルマラソン（42.195km）でサブ3をなんとか維持している市民マラソンランナー
 - 2012年の第1回京都マラソンも参加しました




本日のお話

- ① STONITHて何？
- ② 本日のPacemakerデモ環境
- ③ STONITHを設定してみよう
- ④ いろいろ故障デモします！
- ⑤ Linux-HA Japanについて

①

STONITHって何？





HAクラスタで
あつてはならないけど
よく聞く話..



それは フェイルオーバー失敗



フェイルオーバー中に
途中で固まっていた・・・

フェイル
オーバ開始

こんな状態。

故障発生

**サービス
停止**

サービス起動中

unmount

共有ディスク


アンマウント
途中でだんまり!?

さらに、
HAクラスタで
“宿命” ともいえる
よく聞く話…





それは スプリットブレイン



これによって
起こる恐ろしいこと

こんな状態。

あいつ
死んだな..

あいつ
死んだな..

切断

サービス起動中

サービス起動しまーす

mount

mount

共有ディスク

データ破壊

が！これらを救うのが



Pacemakerの

STONITH



STONITHって？




Shoot-**T**he-**O**ther-**N**ode- **I**n-**T**he-**H**ead

制御が利かないサーバをHAクラス
タから「強制的に離脱」させる機能
です。

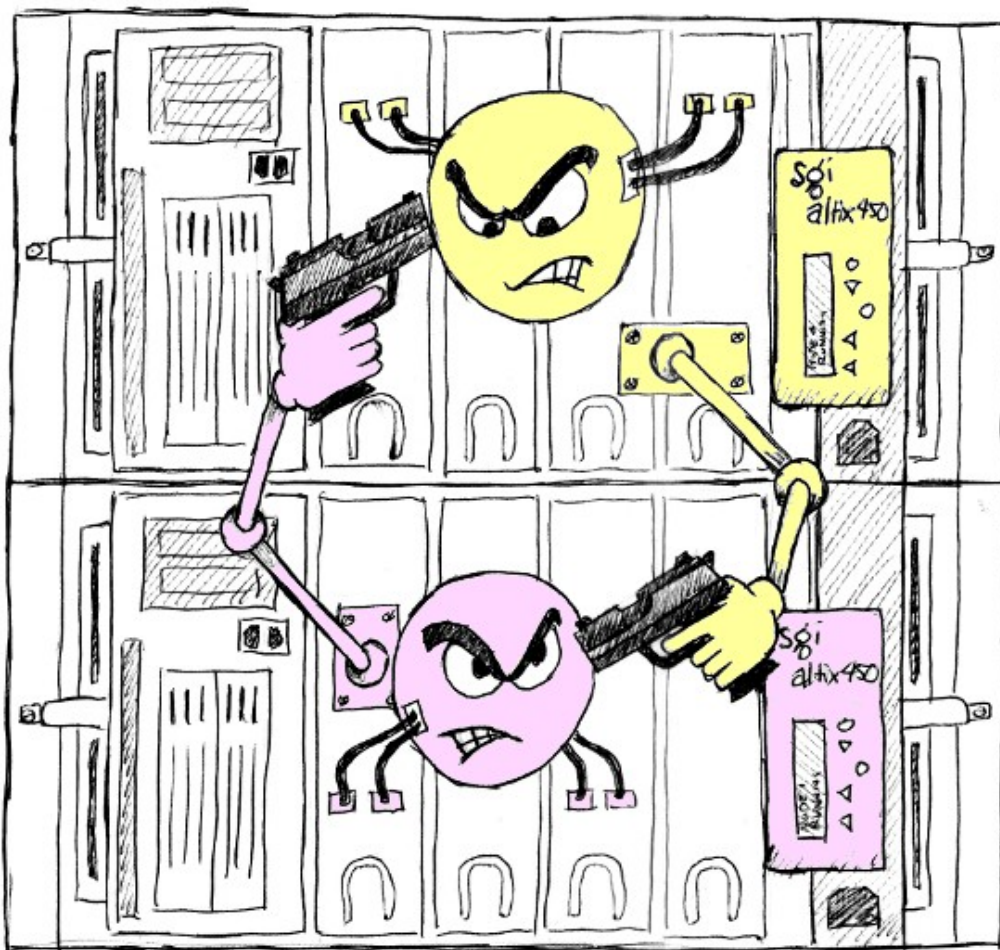
フェンシング





絵で書くと
こんなイメージ..

自殺機能ではなく、他殺機能です。



DON'T ANYBODY MOVE ...

<http://ourobengr.com/ha>

Linux-HA Japan Project



HighAvailability

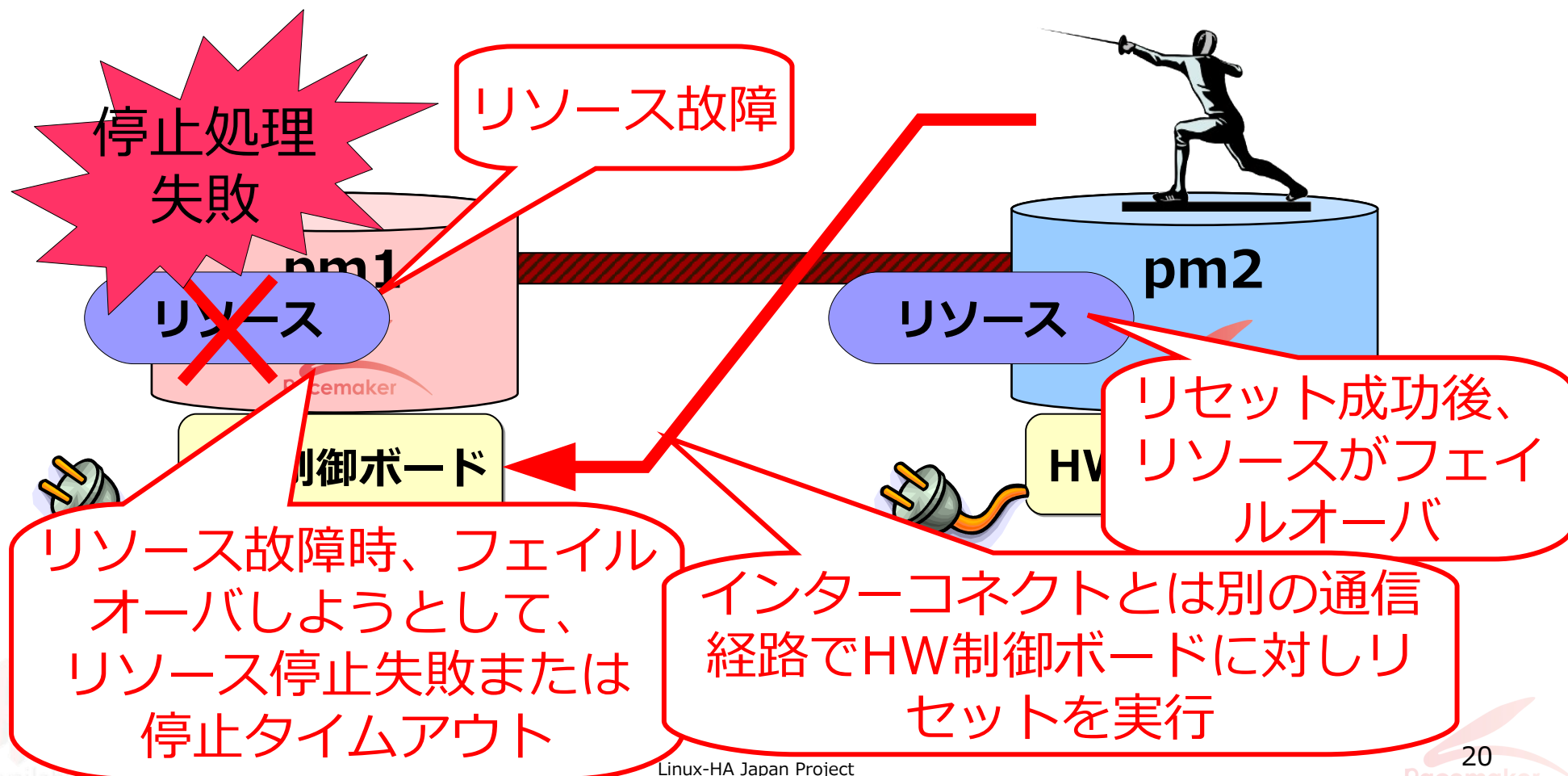
フェンシングが起こるのは・・

- 対向サーバがサービスの停止に失敗したとき
- 対向サーバの応答が無いとき

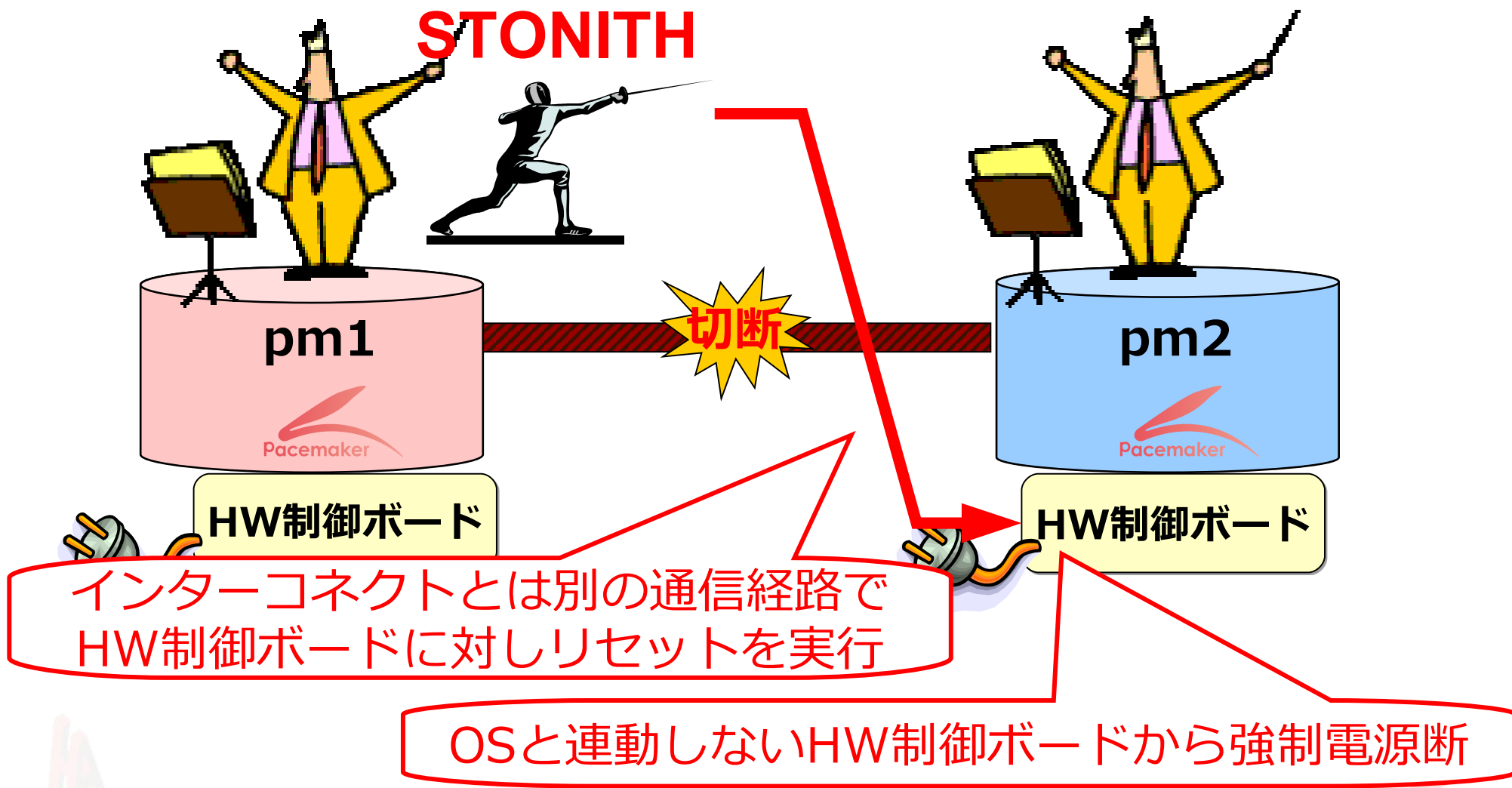
では、
具体的にどうやって
強制離脱させるの？


STONITH実行例(リソース停止失敗)

STONITH



STONITH実行例(スプリットブレイン)





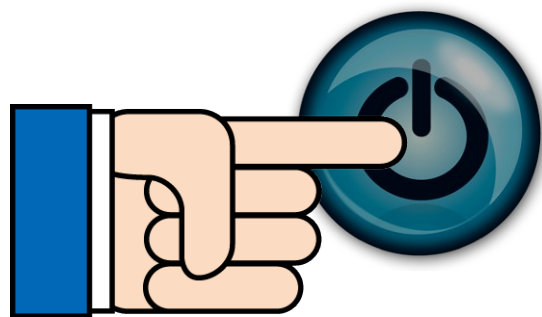
不具合があったとき、
OSシャットダウン処理を
行わずに、

対向サーバの 電源OFFを Pacemaker行うのです



強引だと
思うでしょうが・・・

最終手段としては、
一般的ですよね？



HW制御ボードは例えばこんなの。

高価なサーバ
のみ搭載!?

■ D社 【iDRAC6】

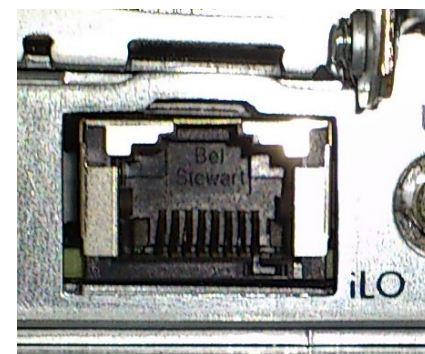
□ Integrated Dell Remote Access Controller 6

- PowerEdge R610 等に標準搭載
- **IPMI** 2.0 対応

■ H社 【iLO4】

□ Integrated Lights-Out 4

- ProLiant DL360 Gen8 等に標準搭載
- **IPMI** 2.0 対応



▲ iLO4

安価なサーバでも搭載されてます。

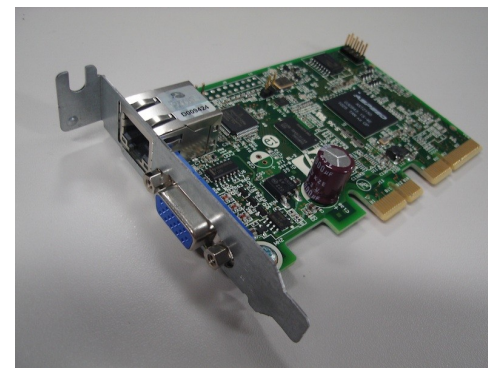
■ 例) H社 MicroServer

- MicroServerにリモート管理オプションのリモートアクセスカードキットを搭載すればSTONITHが使用可能です。

Linux-HA
Japan展示
ブースのデモ
機で使ってい
ます



▲ H社 MicroServer



▲ リモートアクセスカード
(**IPMI** 2.0 対応)

IPMI対応ならば
STONITHできます！

IPMIとは...

サーバ・プラットフォームの状態（温度、電圧、ファン、バスなど）監視や復旧、リモート制御を行うための標準インターフェイス仕様です。

使用するにはまずハードウェアが対応していることが前提である上、それなりの設定が必要です。

このIPMIデバイスにアクセスするためのソフトウェアが **IPMITool** で、PacemakerからもSTONITHプラグインからこのコマンドを使用します。

```
# ipmitool [options...] <command>
```

STONITHプラグイン

Pacemakerには、様々なSTONITHプラグインが標準装備されています。プラグインは、シェルスクリプト、Perl、Python等で作成されています。

プラグインの詳細は後ほど説明します!!

目的	プラグイン名
フェンシング(電源断)制御	ipmi (IPMIデバイス用) libvirt (KVM,Xen等 仮想制御用) riloe (H社 iLO1, iLO2用) ibmrsa-telnet (I社 RSA2用)
サーバ生死確認、相撃ち防止	stonith-helper
停止通知	meatware

ipmi プラグイン

[/usr/lib64/stonith/plugins/external/ipmi](#)

IPMIに準拠したHW制御ボードを制御するプラグインです。
ipmiプラグインでは、ipmitool コマンドをプラグインから実行しています。

reset処理で実行されるコマンド

```
# ipmitool -I lanplus -H <IPMIボードIPアドレス>  
-U <ユーザ名> -P <パスワード> power reset
```

```
Chassis Power Control: Reset
```

status(状態確認)処理で実行されるコマンド

```
# ipmitool -I lanplus -H <IPMIボードIPアドレス>  
-U <ユーザ名> -P <パスワード> power status
```

```
Chassis Power is on
```



ipmi STONITHプラグイン設定例

```
primitive prmSt1-2 stonith:external/ipmi ¥
params ¥
    priority="2" ¥
    stonith-timeout="60s" ¥
    hostname="pm1" ¥
    ipaddr="172.20.24.147" ¥
    user id="pacemaker" ¥
    passwd="hoge hoge" ¥
    interface="lanplus" ¥
op start interval="0s" timeout="60s" ¥
op monitor interval="3600s" timeout="60s" ¥
op stop interval="0s" timeout="60s"
```

この設定により
reset処理時に
実行される
コマンド

```
# ipmitool -I lanplus -H 172.20.24.147 -U pacemaker -P
hoge hoge power reset
```


MicroServerでの例

リモートアクセスカード管理画面

HP ProLiant MicroServer Remote Access Card Properties

- Configuration
 - Network
 - Network Security
 - Security
 - Users**
 - Services
 - IPMI
- Sessions
- Update
- Utilities
- Server Information
 - Power
 - Control
 - Thermal
 - Fan
 - Temperatures
 - System Event Log
 - Event Management
 - Platform Events
 - Trap Settings
 - Email Settings
 - vKVM & vMedia

User Configuration

General

User ID	4
Enable User	<input checked="" type="checkbox"/>
User Name	pacemaker
Change Password	<input type="checkbox"/>
New Password	
Confirm New Password	

User Privileges

User Role	User
IPMI LAN Privilege	Administrator

“userid” に設定
するユーザ名

“ipaddr” に設定
するIPアドレス

IPv4 Settings

Enabled	<input checked="" type="checkbox"/>
Use DHCP	<input type="checkbox"/>
IP Address	172.20.24.147
Subnet Mask	255.255.192.0
Gateway	172.20.0.1
Use DHCP to obtain DNS server addresses	<input type="checkbox"/>
Preferred DNS Server	0.0.0.0
Alternate DNS Server	0.0.0.0

“passwd” に設定
するパスワード

resetのために
必要なユーザ権限は
HWによって様々..



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 4G)

- OS

- CentOS 6.4 x86_64

- HAクラスタ

- Pacemaker-1.0.13
 - 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 9.2.4

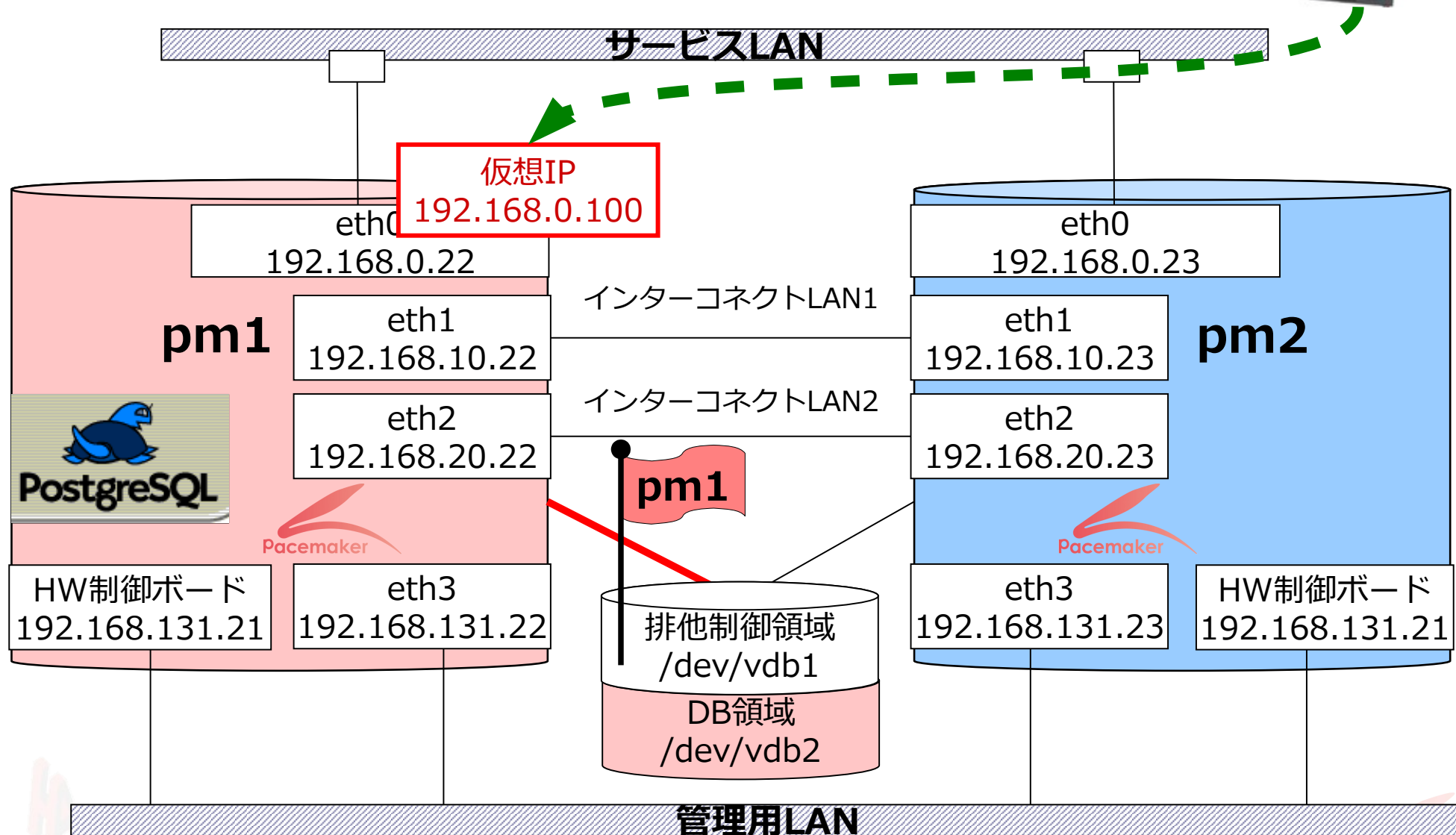
- 仮想環境

- KVM (ホスト×1、ゲスト×2)
 - 各ゲストOSには、CPU×1・メモリ1024M を割り当て

Pacemakerデモ構成

pm1: アクティブ
pm2: スタンバイ

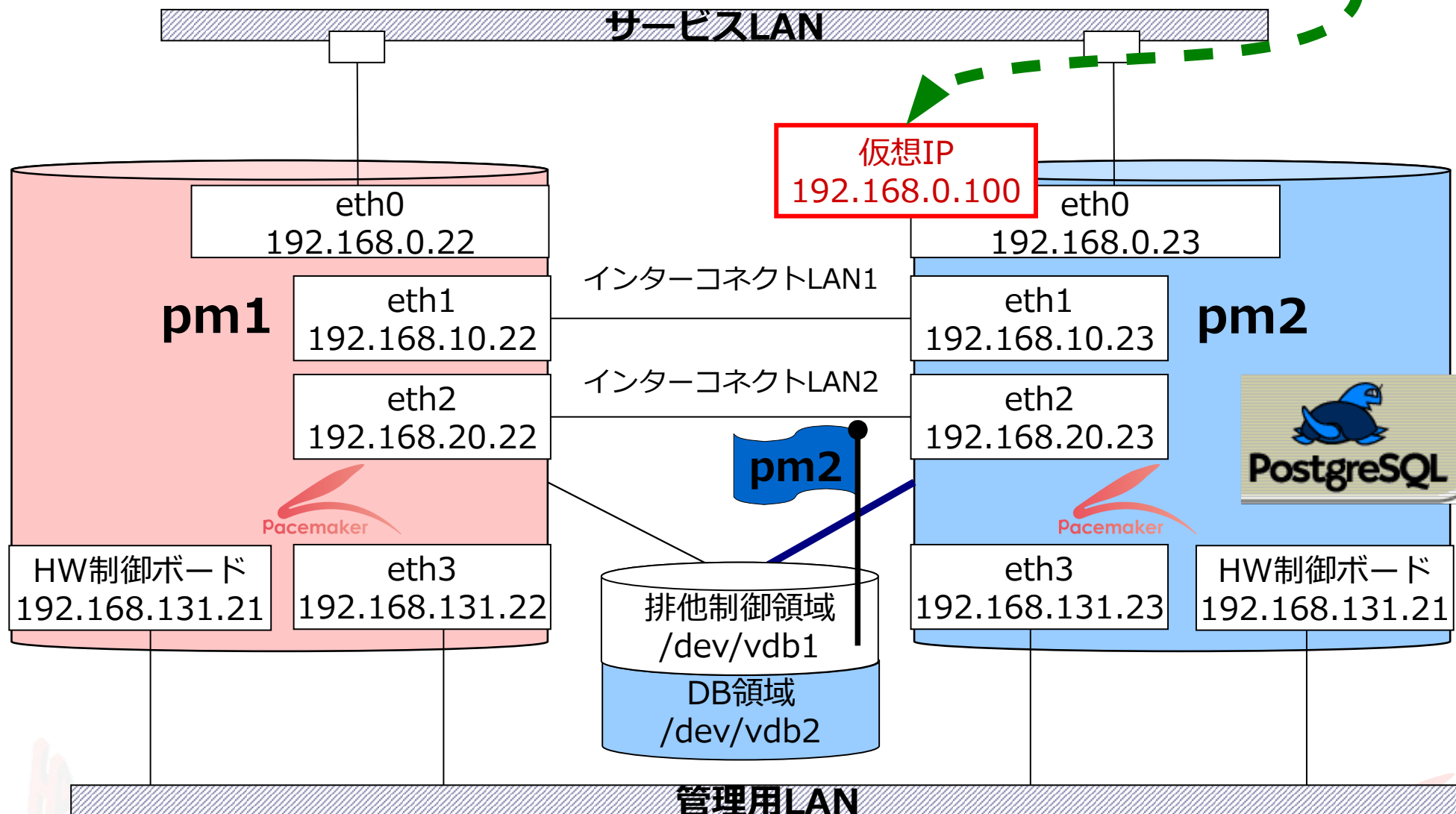
demo
(ホスト)



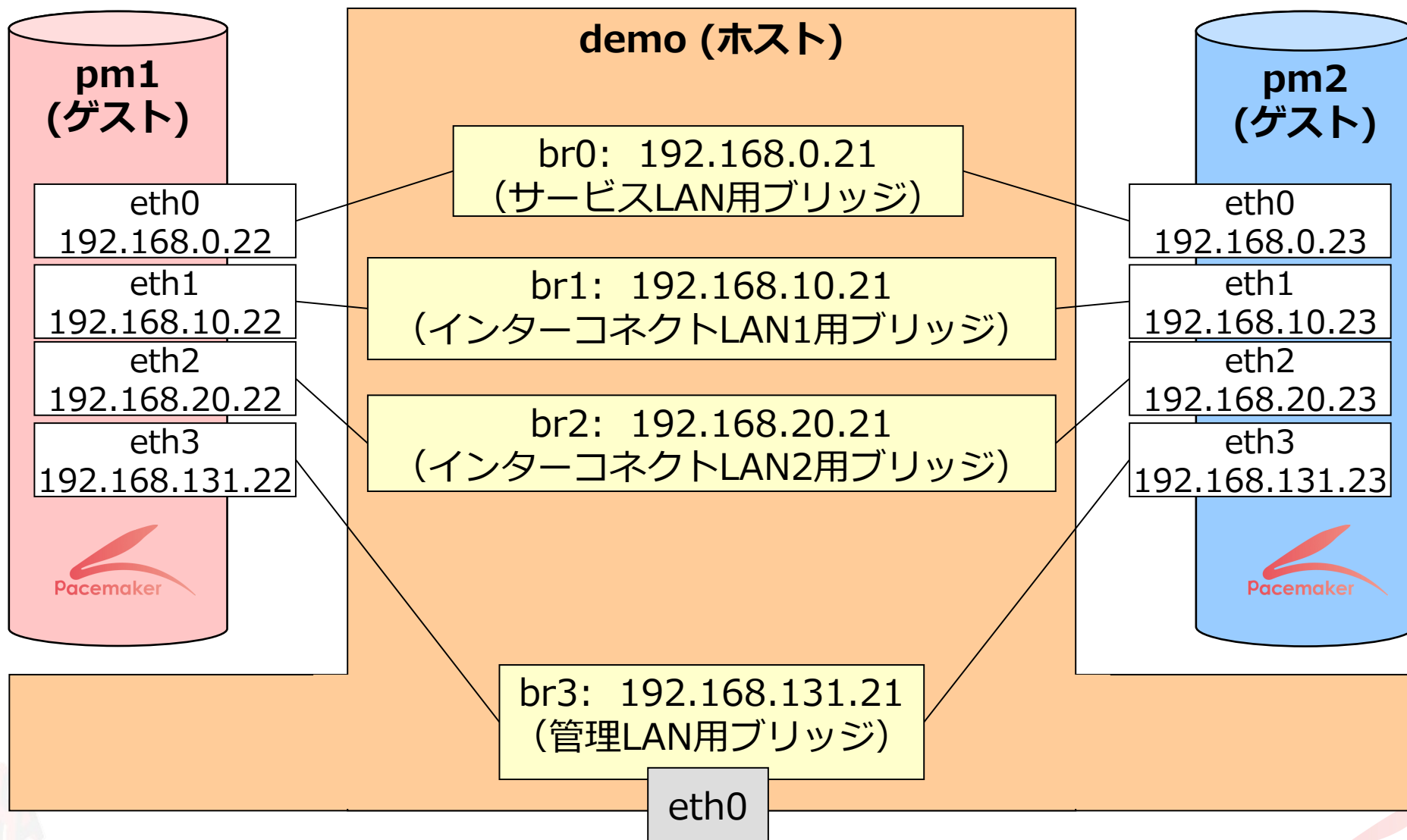
Pacemakerデモ構成

pm1: スタンバイ
pm2: アクティブ

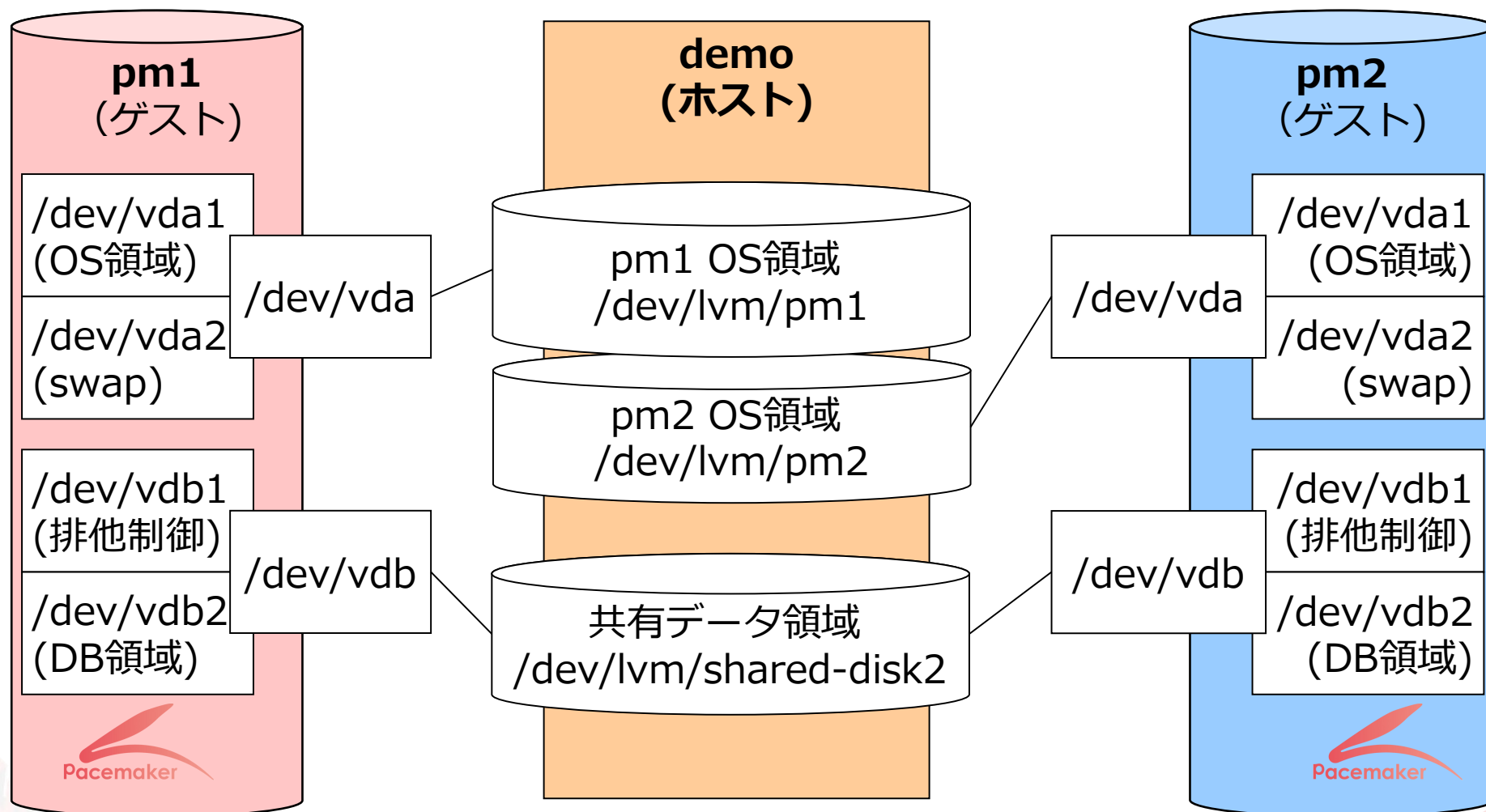
demo
(ホスト)



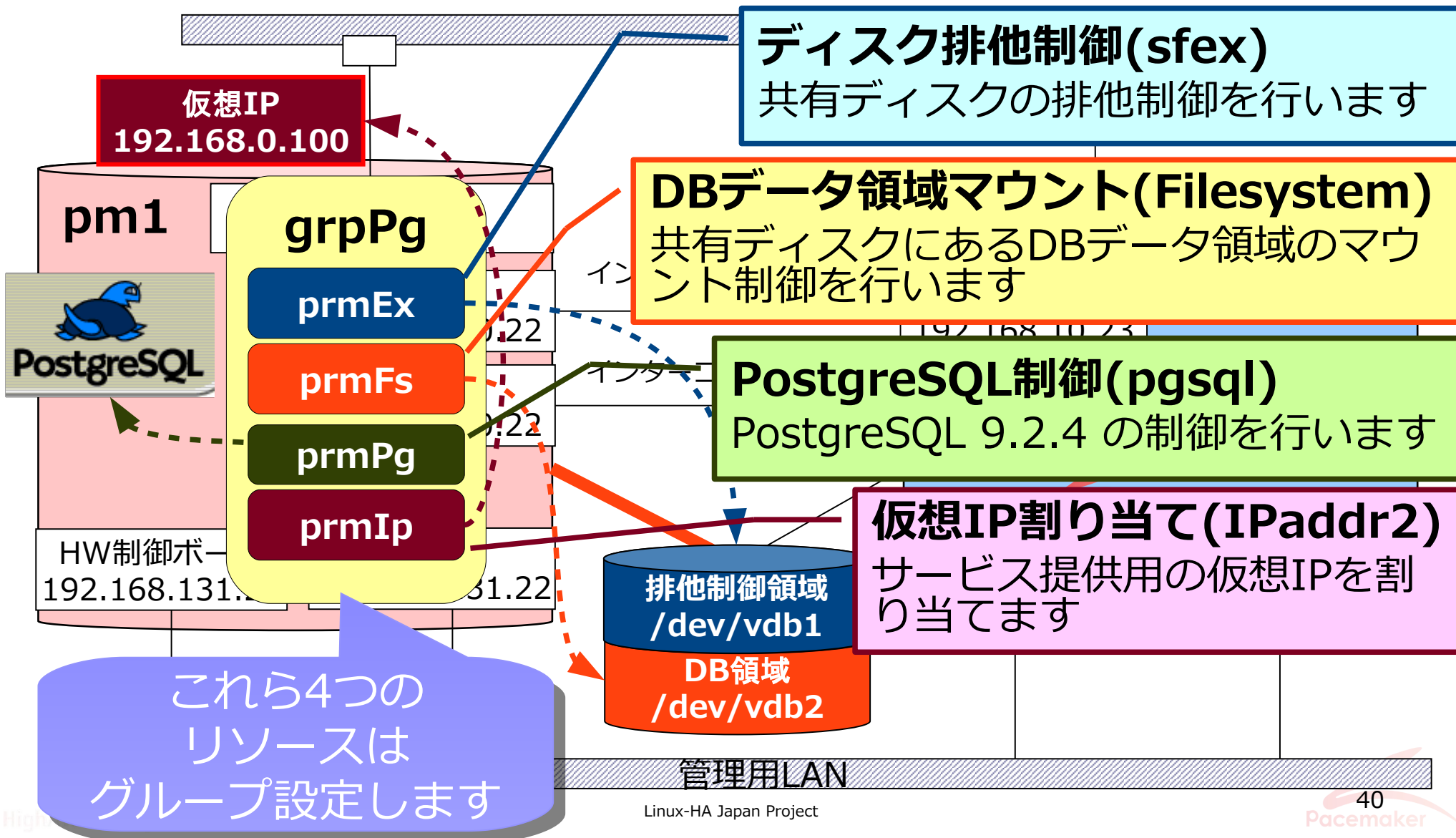
Pacemakerデモ機構成（仮想NW）



Pacemakerデモ機構成（仮想ディスク）



Pacemakerデモリソース構成



リソース状態表示

状態表示コマンド(crm_mon)にて、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。

```
# crm_mon
```

```
Resource Group: grpPg
```

```
    prmEx      (ocf::heartbeat:sfex):   Started pm1
```

```
    prmFs      (ocf::heartbeat:Filesystem): Started pm1
```

```
    prmPg      (ocf::heartbeat:pgsql):   Started pm1
```

```
    prmIp      (ocf::heartbeat:IPaddr2):  Started pm1
```

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

[デモ1]

STONITH無環境で
リソース停止
タイムアウトの
故障デモします！

故障デモに あたって...

状態表示コマンド **crm_mon**
の結果はデモでは表示しきれ
ないので、crm_monコマン
ドのワンショットから一部を
デモ目的に必要な部分をスク
リプトで抜き出し1秒毎に表
示してデモを行います。

推奨ではないですが、デモの
関係上Pacemakerは自動起動
設定しています。

```
=====
Last updated: Thu Aug  1 09:01:14 2013
Stack: Heartbeat
Current DC: pm2 (e470e941-7c11-42a4-9180-1256f0c1f22d)
- partition with quorum
Version: 1.0.13-30bb726
2 Nodes configured, unknown expected votes
4 Resources configured.
=====

Online: [ pm1 pm2 ]

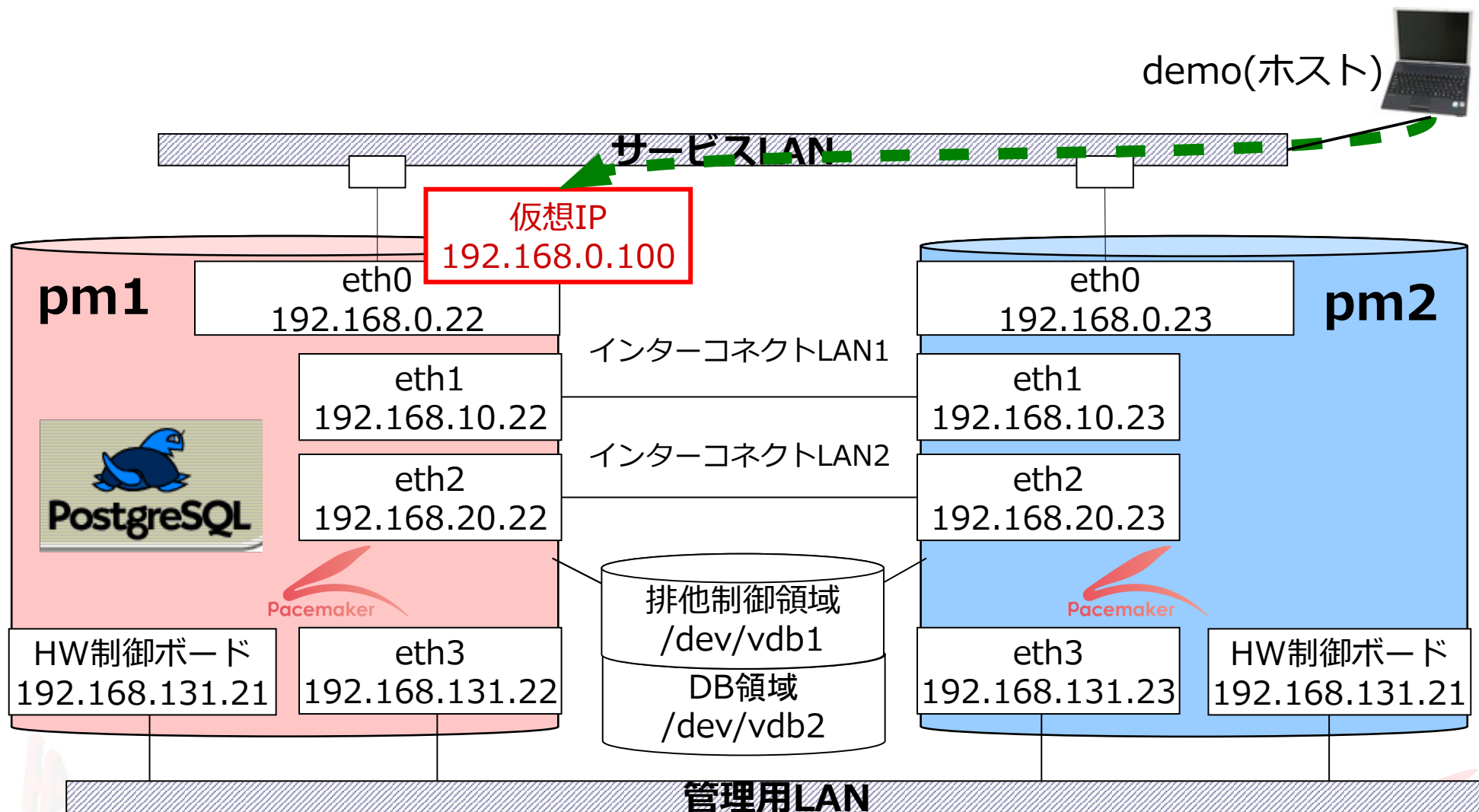
Resource Group: grpPg
  prmEx (ocf::heartbeat:sfex): Started pm1
  prmFs (ocf::heartbeat:Filesystem): Started pm1
  prmPg (ocf::heartbeat:pgsql): Started pm1
  prmIp (ocf::heartbeat:IPaddr2): Started pm1
Clone Set: clnDiskd1
  Started: [ pm1 pm2 ]
Clone Set: clnDiskd2
  Started: [ pm1 pm2 ]
Clone Set: clnPingd
  Started: [ pm1 pm2 ]

Node Attributes:
* Node pm1:
  + default_ping_set : 100
  + diskcheck_status : normal
  + diskcheck_status_internal : normal
  + pm2-eth1 : up
  + pm2-eth2 : up
* Node pm2:
  + default_ping_set : 100
  + diskcheck_status : normal
  + diskcheck_status_internal : normal
  + pm1-eth1 : up
  + pm1-eth2 : up

Migration summary:
* Node pm1:
* Node pm2:
```

PostgreSQLに接続...

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



停止タイムアウトデモのために、 こんな仕掛けします…

1. postgres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/postgresql

```
779 #postgresql_stop: postgresql_real_stop() wrapper for
780 postgresql_stop() {
781     sleep 60
782     if ! is_replication; then
783         postgresql_real_stop
784         return $?
785     else
```

2.crmコマンドのeditモードで prmPg のストップタイムアウトを 60s から 10s に変更します。

edit は、初期構築後に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-9.2/bin/pg_ctl" psql="/usr/pgsql-  
9.2/bin/psql" pgdata="/var/lib/pgsql/9.2/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="10s" on-fail="block"
```

停止タイムアウト時は、 こんなエラー表示になります。

```
# crm_mon -f
```

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

停止タイムアウト状態
を表示

停止失敗時リソースは、 unmanage状態になります。

```
# crm_mon
```

```
Resource Group: grpPg
```

```
  prmEx (ocf::heartbeat:sfex): Started pm1
```

```
  prmFs (ocf::heartbeat:Filesystem): Started pm1
```

```
  prmPg (ocf::heartbeat:pgsql): Started pm1 (unmanaged) FAILED
```

```
  prmIp (ocf::heartbeat:IPaddr2): Stopped
```

unmanage(管理外)状態を表示

unmanage とは？

その名のとおり、Pacemakerから管理外の状態です。

リソース停止の **on-fail** を **block** にした場合、リソース停止失敗時は、unmanage 状態になります。
この場合、サーバの電源OFFを保守者が実施しなければなりません。

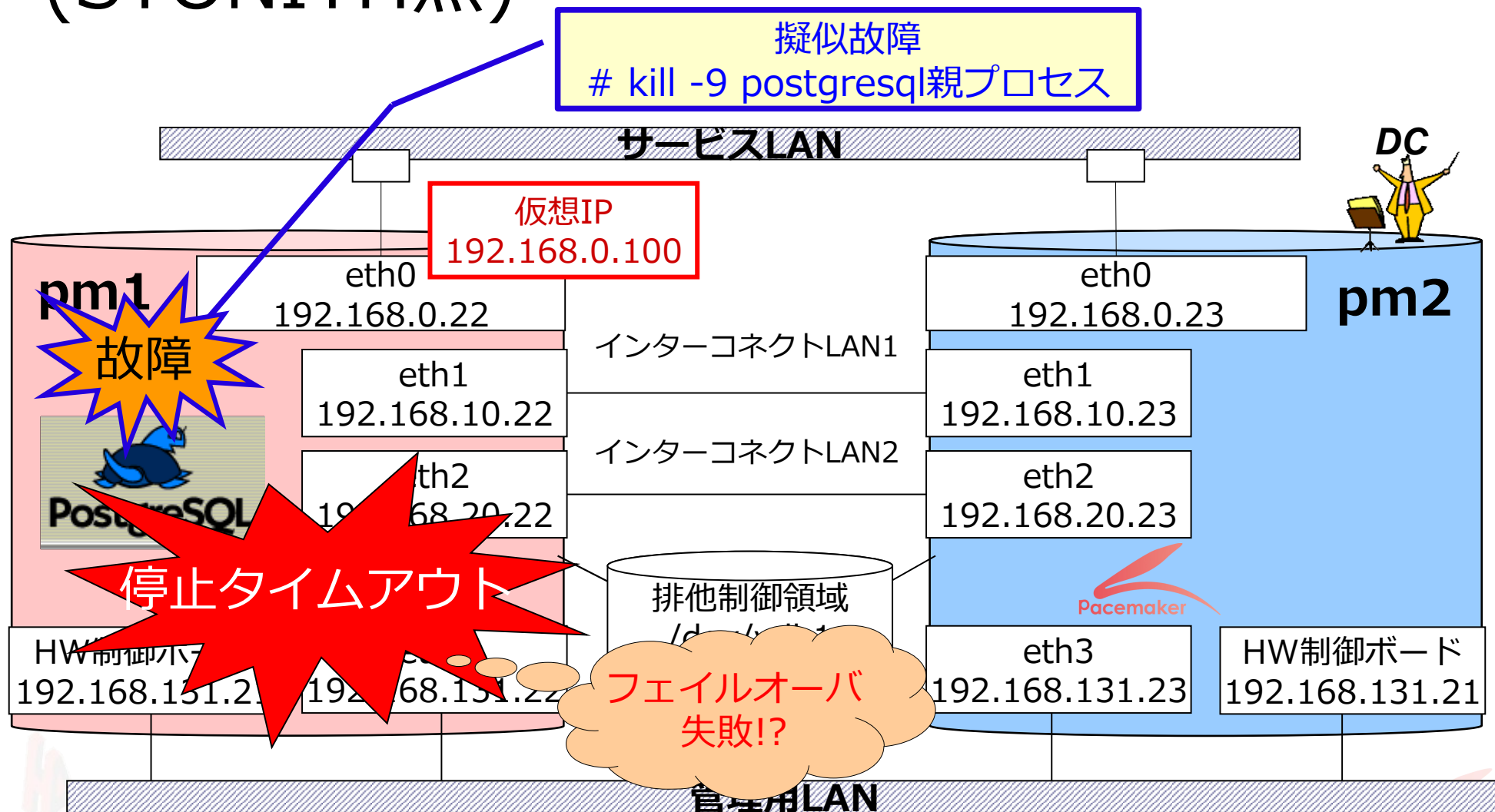
on-fail

リソースが故障した場合の処理は、on-fail 属性に従います。

```
op stop interval="0s" timeout="10s" on-fail="block"
```

- block
 - ・故障が生じたリソースの管理を停止し、何もせず保
守者介入まで待機します。
- fence
 - ・故障が生じたリソースが稼動していたサーバをクラ
スタからSTONITHにより離脱させます。
- ignore
 - ・何も処理を行いません。

リソース故障時、停止タイムアウト… (STONITH無)



擬似故障
kill -9 postgresql親プロセス



③

STONITHを設定してみよう

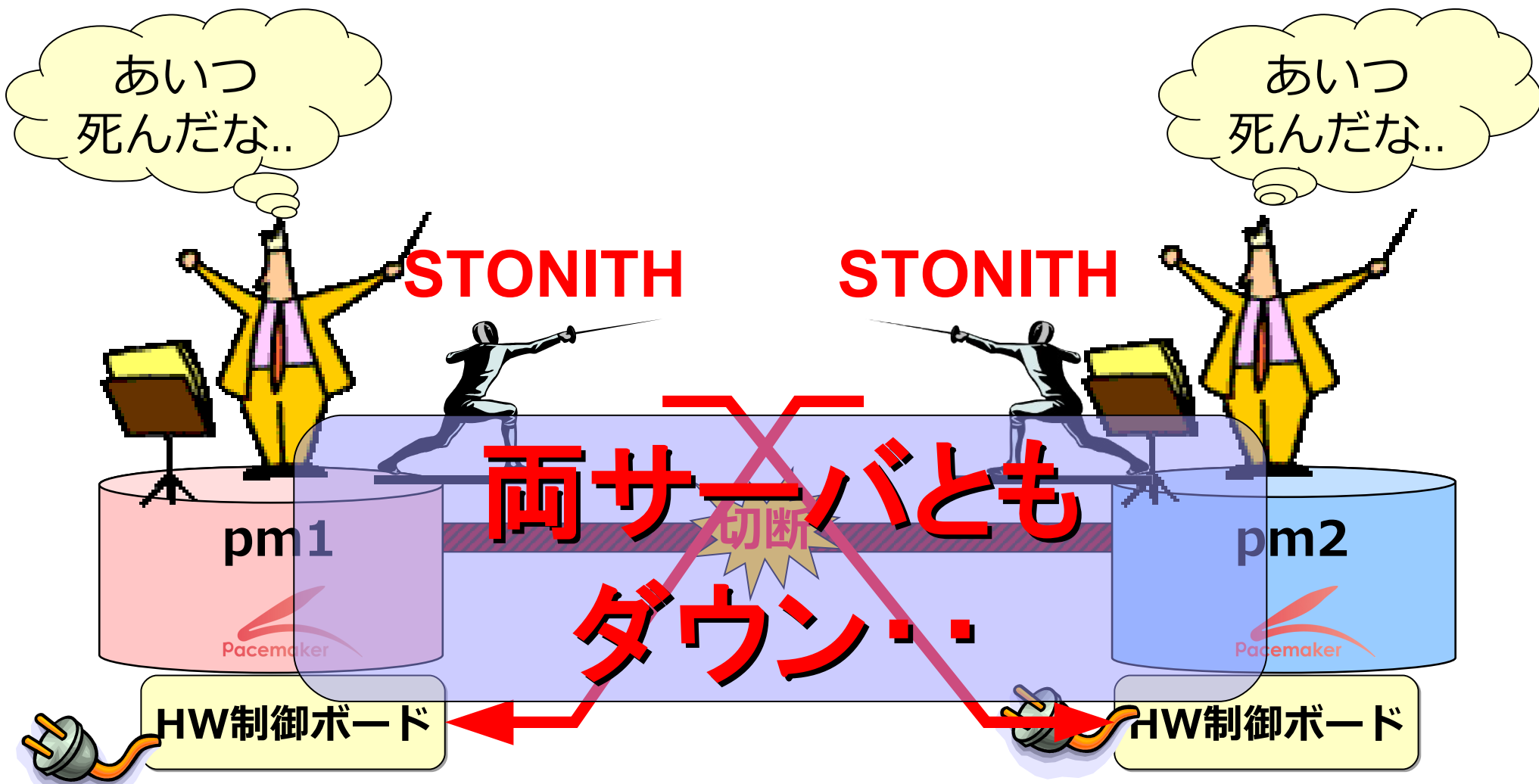
STONITHでよくある 質問




スプリットブレイン時は
相撃ちにならないの？




STONITHによる相撃ち？





これは
残念ながらあり得ます・・



が！
STONITHプラグイン
stonith-helper
を併用すれば起きません

Pacemakerデモリソース構成 (STONITH)

これら3つの
STONITHリソースを
グループ設定し、
エスカレーション処理
させます。

- stonith-helper

- 相撃ち防止対策、サーバ断確認を行います。

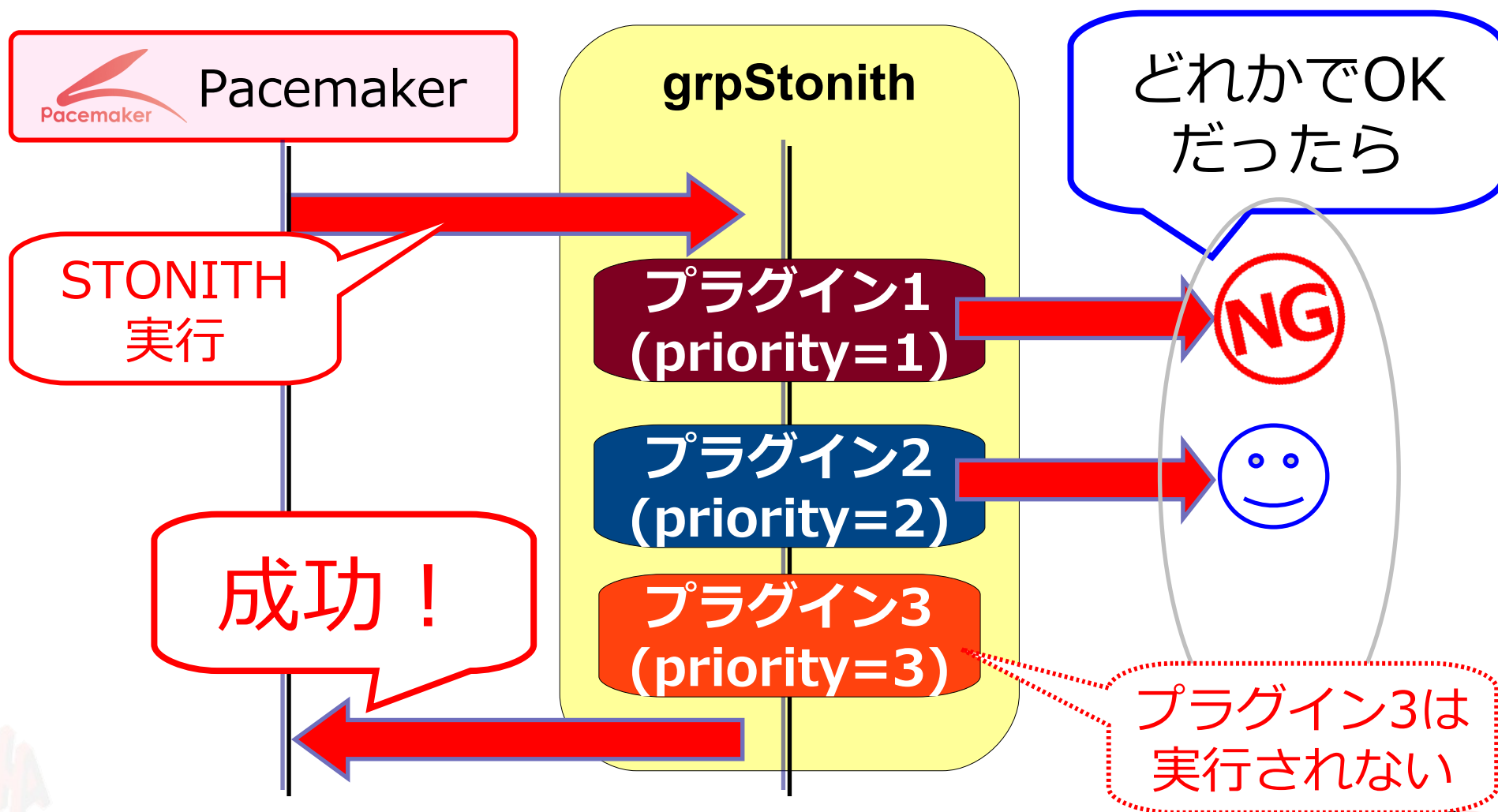
- libvirt

- フェンシング対象のゲストOSをvirshコマンドによりホストOS経由で強制断します。

- meatware

- ユーザによる停止通知を行います。

STONITHエスケーレーション処理



STONITH処理 エスカレーション順番

grpStonith



stonith-helper
(priority=1)

libvirt
(priority=2)

meatware
(priority=3)

1. 生死確認と相撃ち防止対策
2. フェンシング
3. 停止通知

各プラグインの 詳細 説明します！



stonith-helperプラグイン

[/usr/lib64/stonith/plugins/external/stonith-helper](#)

標準STONITH機能で足りない機能をお助けするプラグインです。

Linux-HA Japan で開発。
pm_extras としてPacemaker
リポジトリパッケージに同梱。

■ サーバ生死確認

- 対向サーバの死活確認として pingによるネットワーク疎通が確認できるか否かでノードの停止を判定。

■ 相撃ち防止

- 指定されたコマンドの結果から STONITH実行サーバがリソースを稼働させているサーバか否かを判断し、reset 処理実行を遅延させることで相撃ちを回避。

stonith-helper サーバ生死確認

pingにより対向サーバの生死を確認します。
対向サーバに付与されているすべてのIPを設定します。



stonith-helper 設定例

```
primitive prmSt1-1 stonith:external/stonith-helper ¥
```

```
params ¥
```

```
priority="1" ¥
```

```
stonith-timeout="40" ¥
```

```
hostlist="pm1" ¥
```

```
dead_check_target="192.168.131.21 192.168.0.22
```

```
192.168.10.22 192.168.20.22 192.168.131.22" ¥
```

```
standby_wait_time="10" ¥
```

```
standby_check_command="/usr/sbin/crm_resource -r prmEx
```

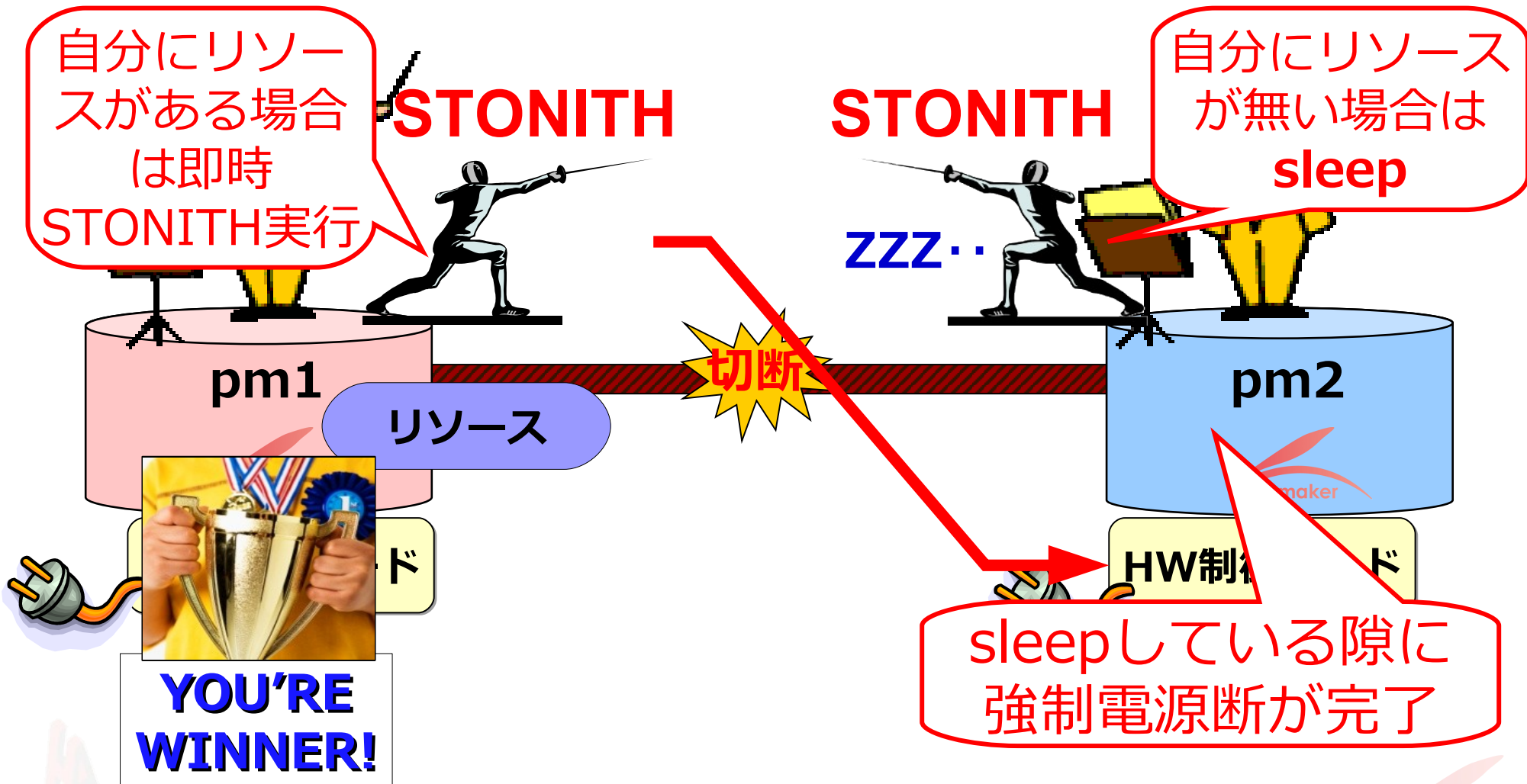
```
-W l ... a `bootname`" ¥
```

priority に優先順位を設定します。
stonith-helperは1が推奨です。

dead_check_target には、サーバに付与されるすべてのIPアドレスを設定します。

OSで設定されるIPのみではなく、HW制御ボードのIPも設定します。

stonith-helper 相撃ち防止方法



stonith-helper 設定例

```
primitive prmSt1-1 stonith:external/stonith-helper ¥
```

```
params ¥
```

```
priority=
```

```
stonith-t
```

```
hostlist=
```

```
dead_chec
```

```
192.168.10.22
```

standby_check_command には、判定コマンドを設定します。

返り値が偽(0以外)だった場合、standby_wait_time 秒 sleep します。

```
standby_wait_time="10" ¥
```

```
standby_check_command="/usr/sbin/crm_resource -r
```

```
prmEx -W | grep -q `hostname`" ¥
```

判定コマンドに、crm_resource コマンドを使用し、prmEx (ディスク排他制御sfex)リソースが自ノードにあるかどうか判定させています。

libvirt プラグイン

[/usr/lib64/stonith/plugins/external/libvirt](#)

ゲストと hypervisor を管理するための コマンド「virsh」を使用し、フェンシングを実現するプラグインです。

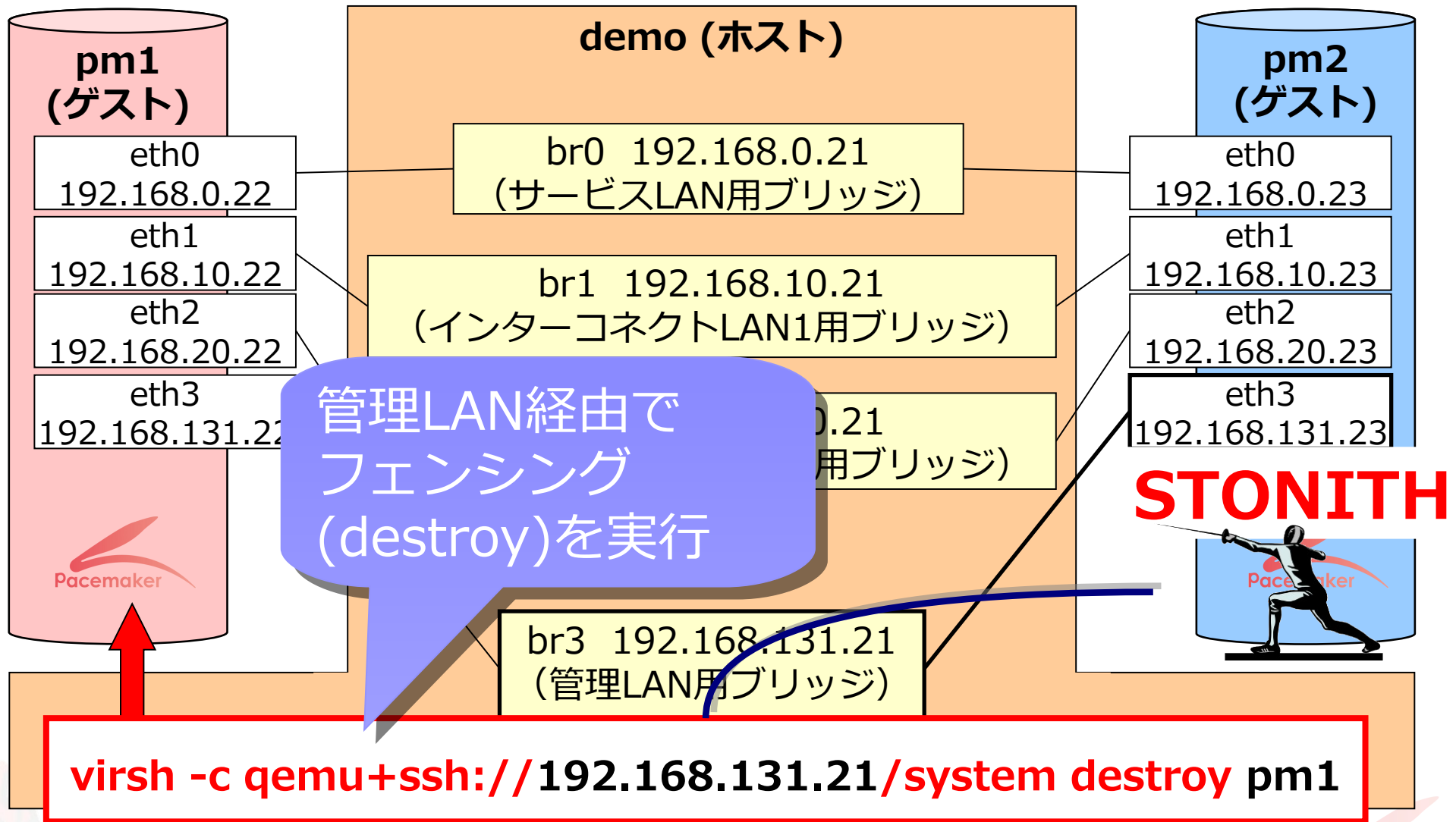
reset処理で実行されるコマンド

```
# virsh -c qemu+ssh://<ホストIP>/system destroy <ゲスト名>  
# virsh -c qemu+ssh://<ホストIP>/system start <ゲスト名>
```

status(状態確認)処理で実行されるコマンド

```
# virsh -c qemu+ssh://<ホストIP>/system version
```

libvirt フェンシング方法



libvirt 設定例

```
primitive prmSt1-2 stonith:external/libvirt ¥
```

```
params ¥
```

```
priority="2" ¥
```

```
stonith-timeout="300" ¥
```

```
hostlist="pm1" ¥
```

```
hypervisor_uri="qemu+ssh://192.168.131.21/system" ¥
```

```
op start interval="0s" timeout="60s" on-fail="restart" ¥
```

```
op monitor interval="3600s" timeout="60s" on-fail="restart" ¥
```

```
op stop interval="0s" timeout="60s" on-fail="ignore"
```

priority に優先順位を設定します。
libvirtは2が推奨です。

ハイパーバイザへの接続URIを設定します。
xen+ssh://<ホストIP>/ とURIを設定すれば Xen環境も可能です。

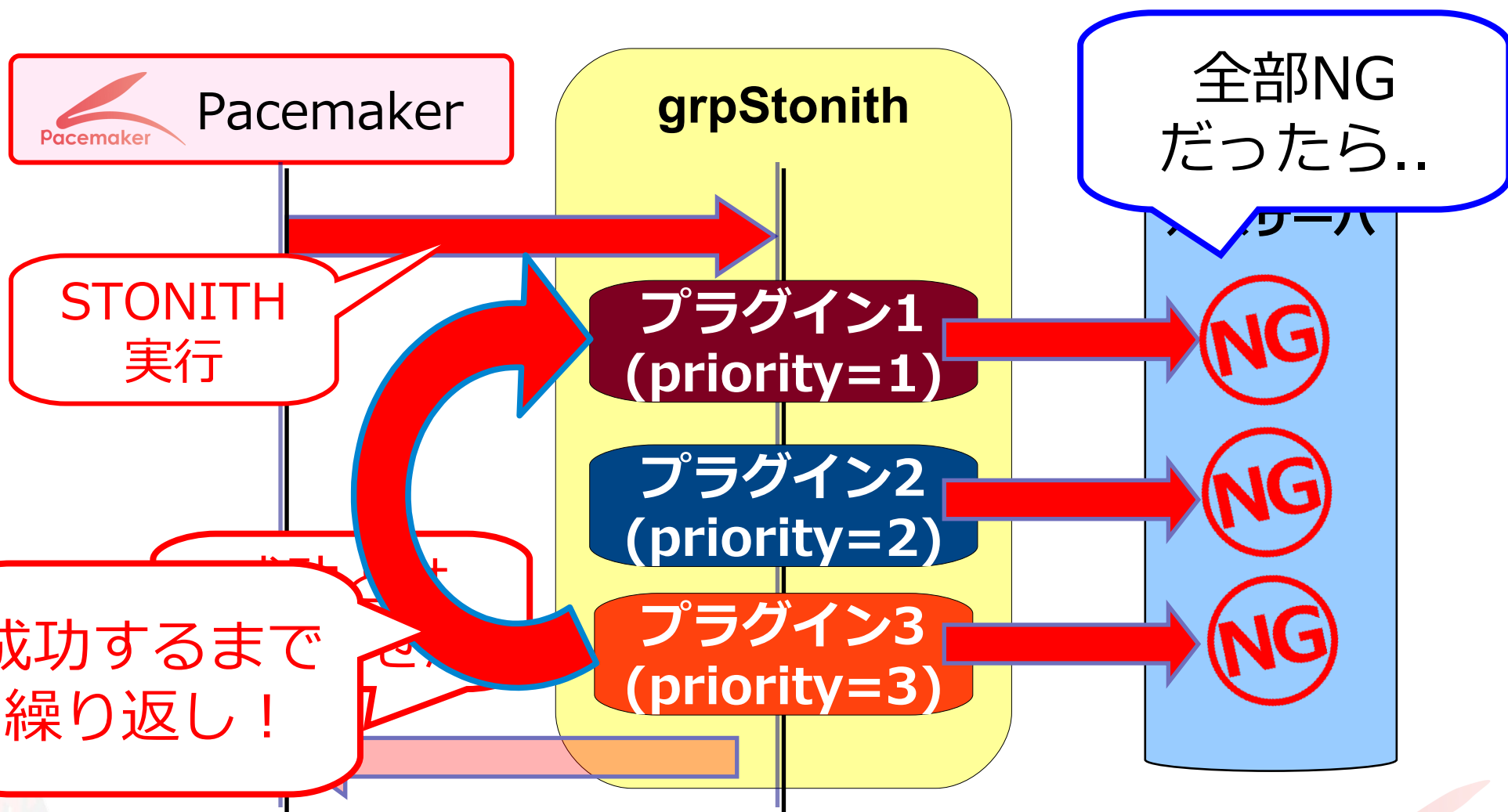
meatwareプラグイン

[/usr/lib64/stonith/plugins/stonith2/meatware.so](#)

保守者から Pacemakerへ対向サーバを
停止した事を通知するインタフェース用
のプラグインです。

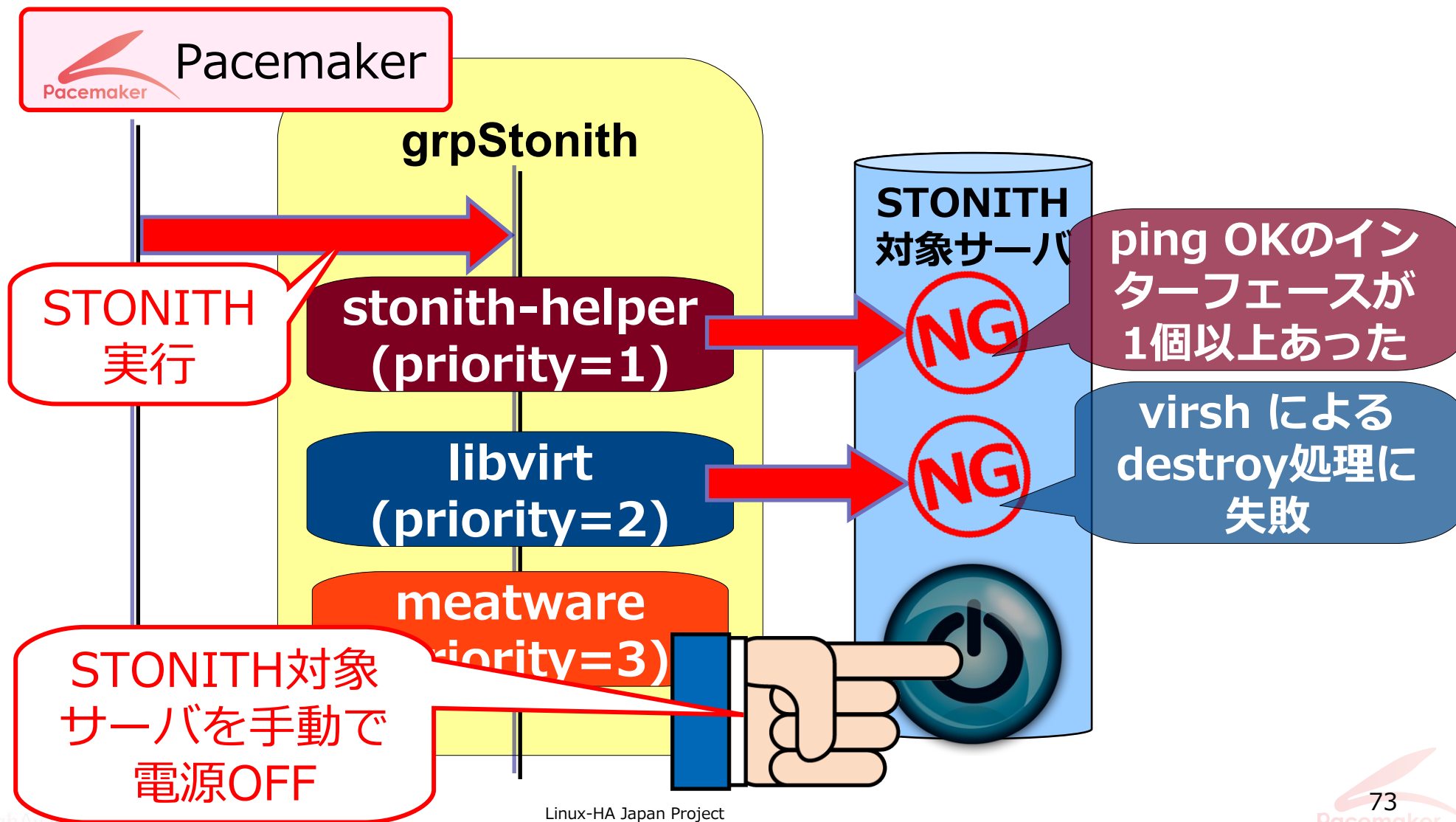


STONITHエスケーレーション処理



つまり
STONITHは
“成功”しなければ
状態遷移しません

meatware処理フロー



手動による電源OFF後、 meatclientコマンドで、Pacemaker に停止完了通知を行います。

```
# meatclient -c <電源OFFしたサーバ名> -w
```

WARNING!

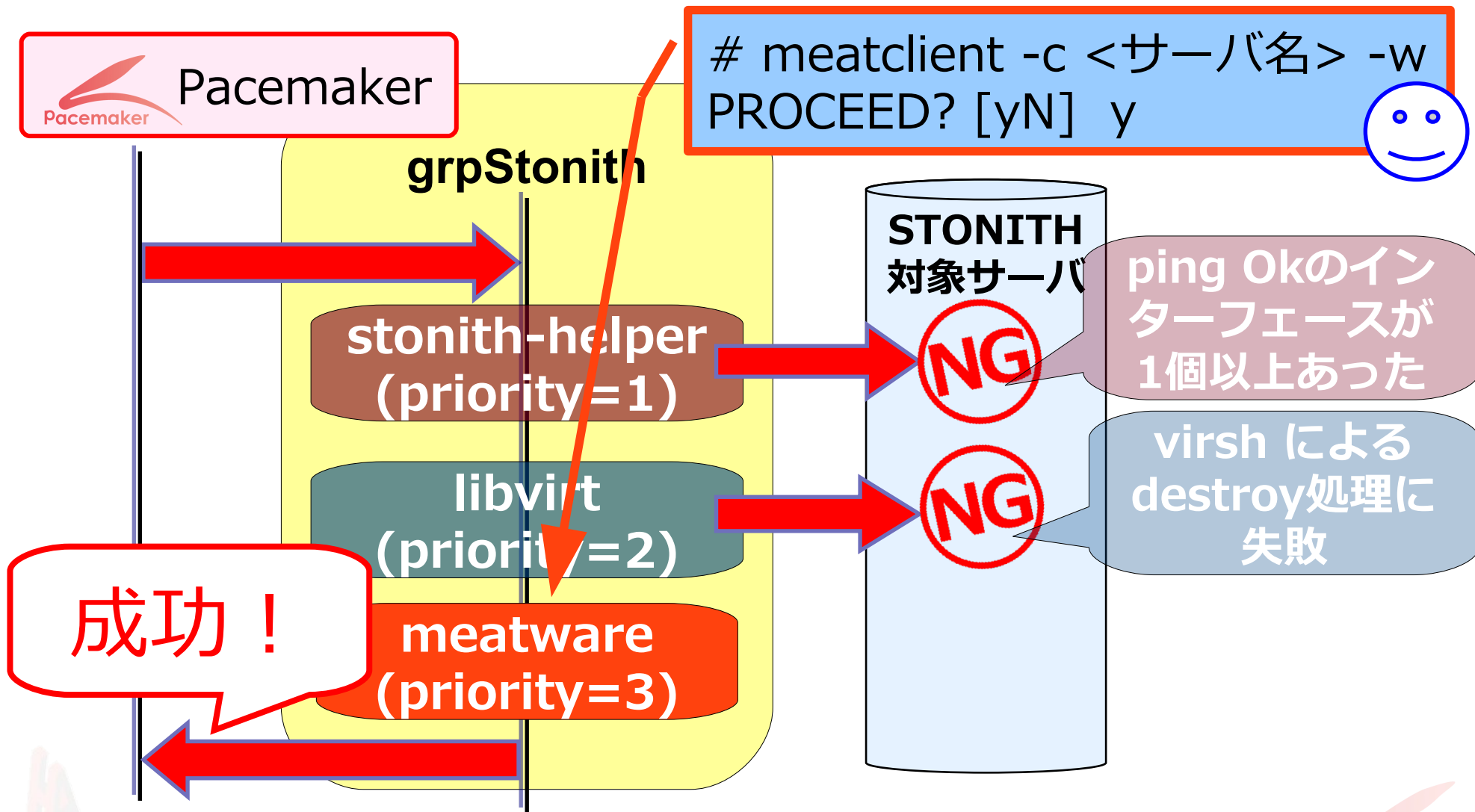
If node "<サーバ名>" has not been manually power-cycled or disconnected from all shared resources and networks, data on shared disks may become corrupted and migrated services might not work as expected.

Please verify that the name or address above corresponds to the node you just rebooted.

PROCEED? [yN]

「y」を入力し
停止完了を通知

meatware処理フロー




meatware 設定例

```
primitive prmSt1-3 stonith:meatware ¥  
  params ¥  
    priority="3" ¥  
    stonith-timeout="600" ¥  
    hostlist="pm1" ¥  
  op start interval="0s" timeout="60s" ¥  
  op monitor interval="3600s" timeout="60s" ¥  
  op stop interval="0s" timeout="60s"
```

priority に優先順位を設定します。
meatwareは3が推奨です。

このタイムアウト経過後、次のプラグインにエスカレーションされてしまうため、電源OFFするタイミングがあるように長めに設定してください。



その他の STONITH基本設定 説明します！

STONITH設定(Cluster Option)

```
### Cluster Option ###  
property no-quorum-policy="ignore" ¥  
  stonith-enabled="true" ¥  
  startup-fencing="false" ¥  
  stonith-timeout="710s" ¥  
  crmd-transition-delay="2s"
```

STONITHを有効にする場合は、
true に変更します。

STONITH処理の全体タイムアウトを設定します。
各STONITHプラグインのタイムアウトを合算した値
(例: 40(stonith-helper) + 60(libvirt)
+ 600(meatware) = 700) より大きい値を設定します。

STONITH設定(Group Configuration)

```
### Group Configuration ###
```

```
group grpStonith1 ¥
```

```
    prmSt1-1 ¥
```

```
    prmSt1-2 ¥
```

```
    prmSt1-3
```

```
group grpStonith2 ¥
```

```
    prmSt2-1 ¥
```

```
    prmSt2-2 ¥
```

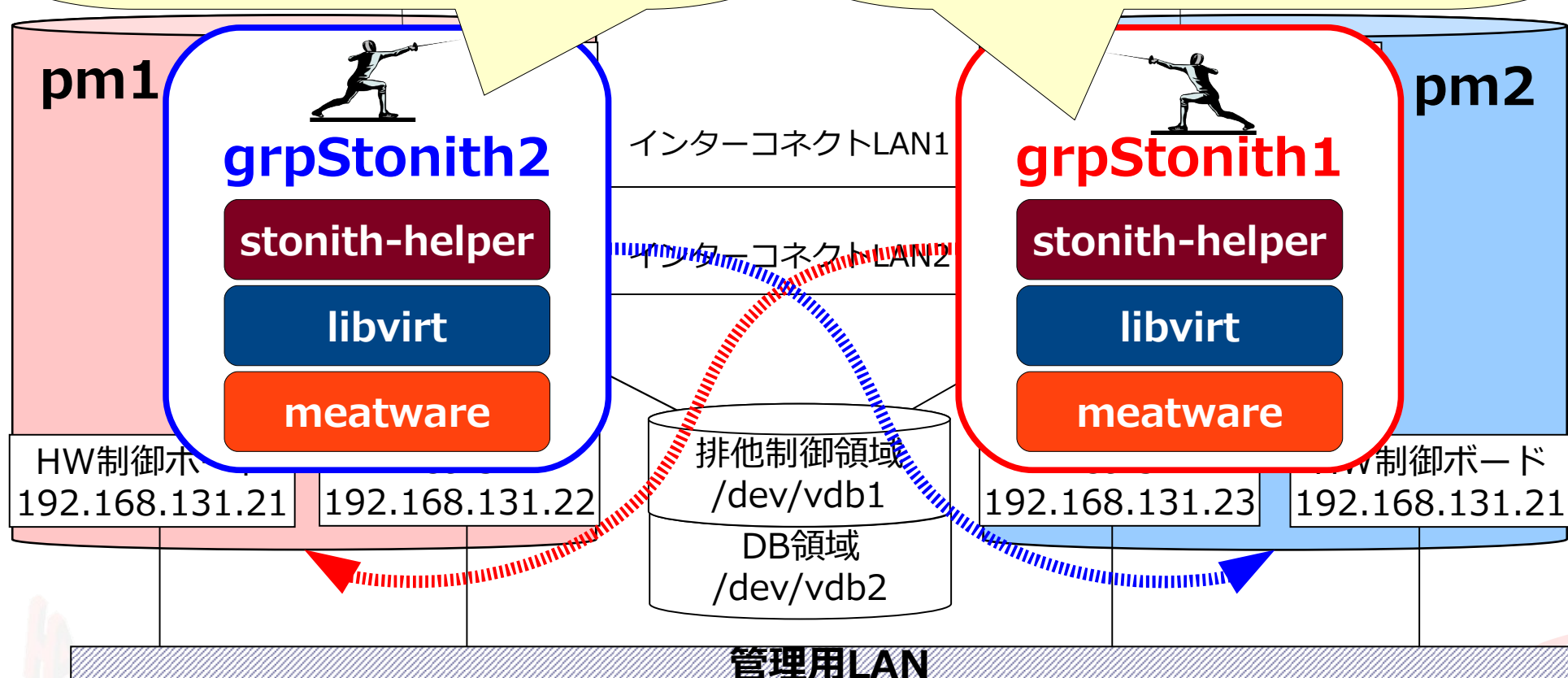
```
    prmSt2-3
```

pm1 を STONITHするグループ名を、grpStonith1 とし、prmSt1-1(stonith-helper)、prmSt1-2(libvirt)、prmSt1-3(meatware)をグループ設定します。

STONITHリソースの配置

grpStonith2 は pm2をフェンシングするSTONITHリソースのためpm1で稼動

grpStonith1 は pm1をフェンシングするSTONITHリソースのためpm2で稼動



STONITH設定(Resource Location)

```
### Resource Location ###  
location rsc_location-grpStonith1-2 grpStonith1 ¥  
    rule -INFINITY: #uname eq pm1  
location rsc_location-grpStonith2-3 grpStonith2 ¥  
    rule -INFINITY: #uname eq pm2
```

-INFINITYの重み付けを付与することにより、
pm2 STONITH用リソースグループ
「grpStonith2」を、pm2で**動作させない**よう
に Location設定します。

STONITH設定(stop on-fail属性)

```
primitive prmPg ocf:heartbeat:pgsql ¥  
  params pgctl="/usr/pgsql-9.2/bin/pg_ctl" psql="/usr/pgsql-  
9.2/bin/psql" pgdata="/var/lib/pgsql/9.2/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="10s" on-fail="fence"
```

サービス系リソースの **stop** の on-fail 属性を **fence**に変更します。

STONITHリソース状態表示

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。

```
# crm_mon
```

```
Resource Group: grpStonith1
```

```
    prmSt1-1 (stonith:external/stonith-helper): Started pm2
```

```
    prmSt1-2 (stonith:external/libvirt):      Started pm2
```

```
    prmSt1-3 (stonith:meatware): Started pm2
```

- prmSt1-1: stonith-helper
- prmSt1-2: libvirt
- prmSt1-3: meatware

[デモ2]

デモ環境を
STONITH有設定に
変更してみます！



④

いろいろ故障デモします！

[デモ3]

STONITH有環境で
リソース停止
タイムアウトの
故障デモします！

停止タイムアウト時、
STONITH実行直前は、
こんな表示になります。

```
pm2 # crm_mon
```

```
Node pm1 (73200749-ec8b-4861-9786-f6475d8ae8c8): UNCLEAN (online)  
Online: [ pm2 ]
```

STONITH対象サーバ
が、UNCLEANと表示されます。

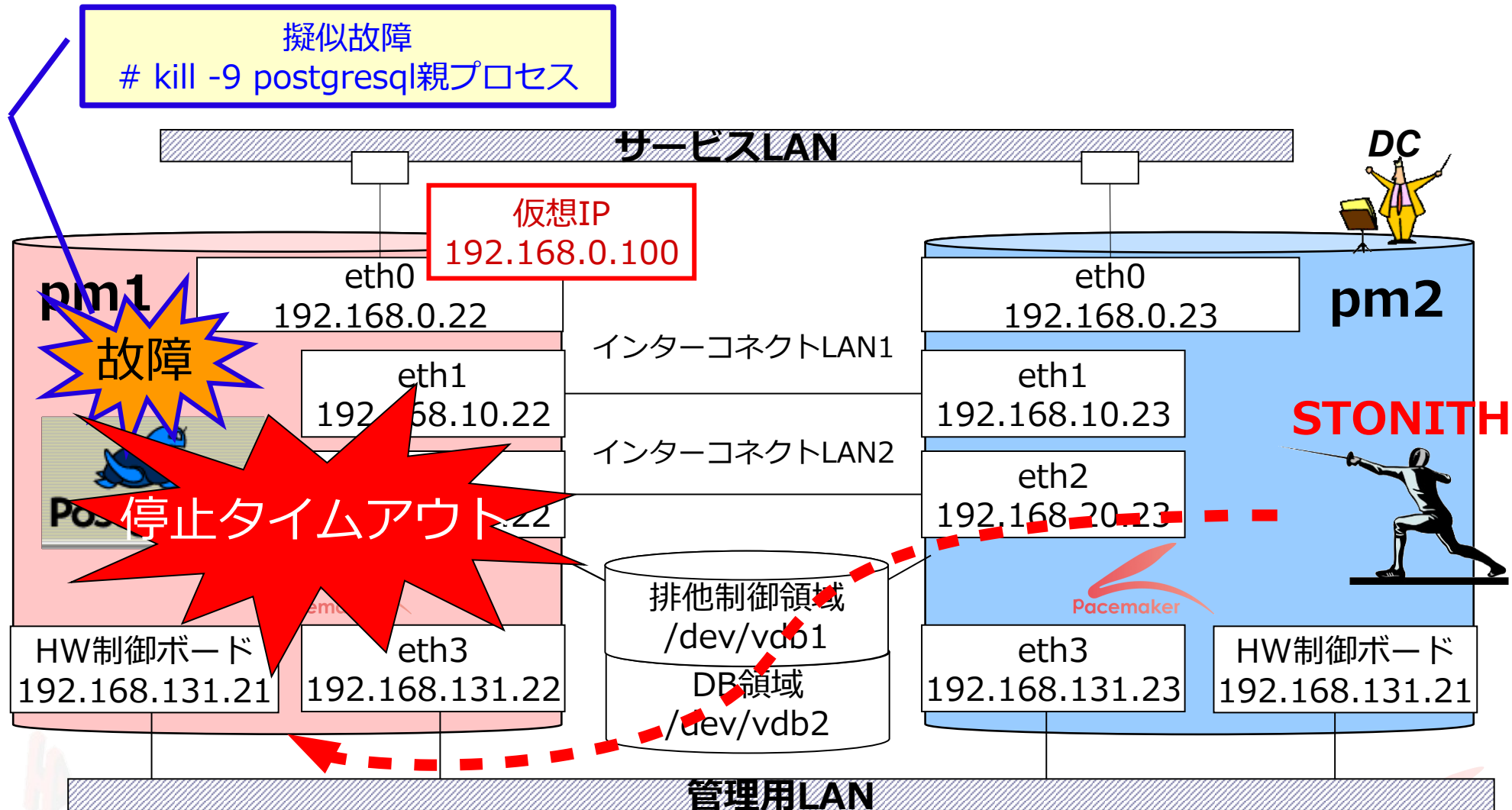
STONITH実行後に、 OFFLINEとなります。

```
pm2 # crm_mon
```

```
Online: [ pm2 ]
```

```
OFFLINE: [ pm1 ]
```


リソース故障時、停止タイムアウト...



[デモ4]

スプリットブレインの
故障デモします！



スプリットブレイン時、
STONITH実行直前は、
こんな表示になります。

お互いに対向サーバを、UNCLEANと表示します。

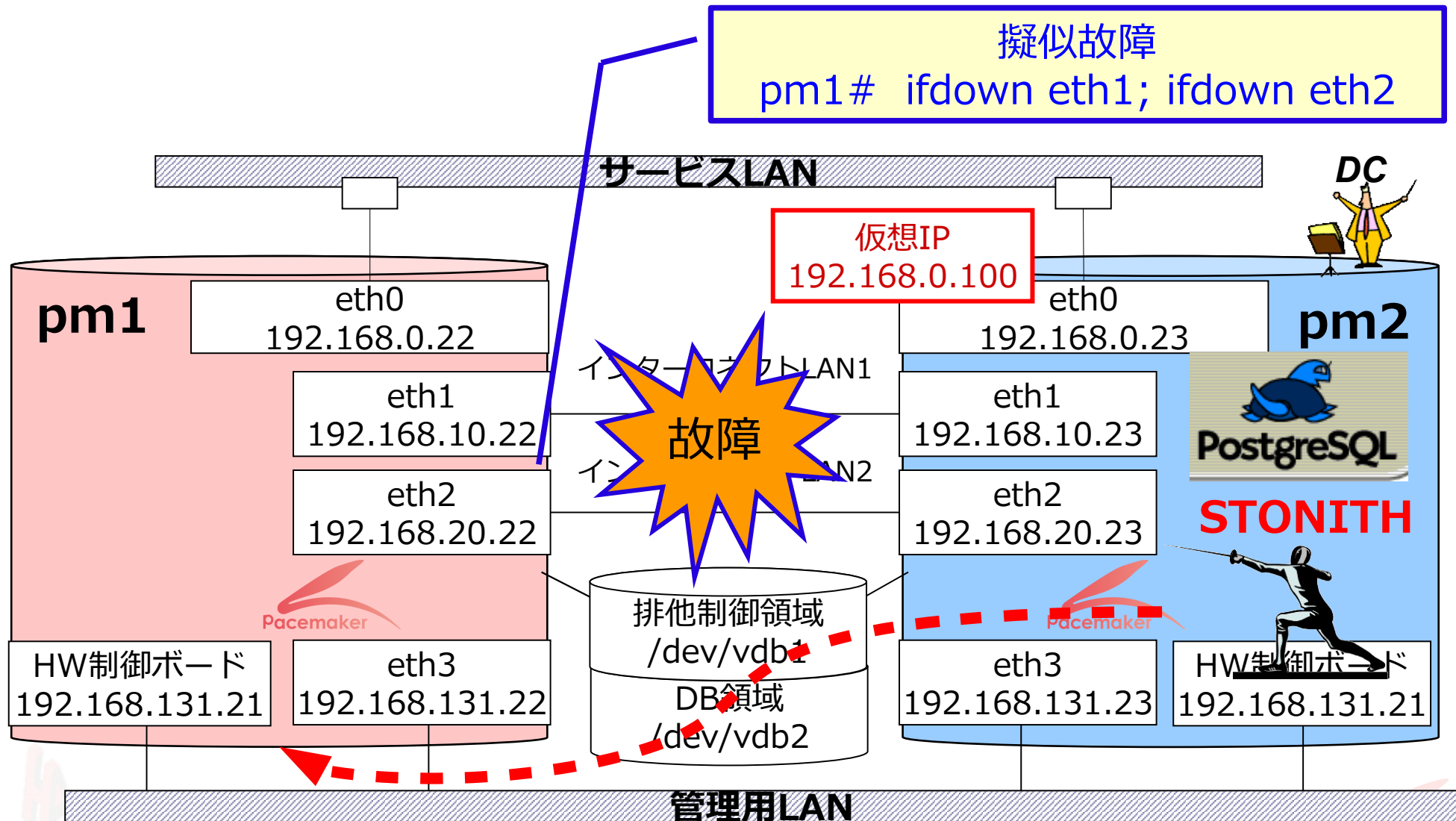
```
pm1 # crm_mon
```

```
Node pm2 (e470e941-7c11-42a4-9180-1256f0c1f22d): UNCLEAN (offline)  
Online: [ pm1 ]
```

```
pm2 # crm_mon
```

```
Node pm1 (73200749-ec8b-4861-9786-f6475d8ae8c8): UNCLEAN (online)  
Online: [ pm2 ]
```

スプリットブレイン...



[デモ5]


STONITH最後の砦
meatwareの
デモします！



meatware 起動時は、 こんなログ表示になります。

/var/log/pm_logconv.out

```
Aug 1 18:45:14 pm2 info: Try to STONITH (RESET) the Node pm1 to  
prmSt1-1 (external/stonith-helper) (pid=9400)  
Aug 1 18:45:18 pm2 ERROR: Failed to STONITH the Node pm1 with one  
local device (exitcode=5). Will try to use the next local device.  
Aug 1 18:45:18 pm2 info: Try to STONITH (RESET) the Node pm1 to  
prmSt1-2 (external/libvirt) (pid=9636)  
Aug 1 18:45:20 pm2 ERROR: Failed to STONITH the Node pm1 with one  
local device (exitcode=5). Will try to use the next local device.  
Aug 1 18:45:20 pm2 info: Try to STONITH (RESET) the Node pm1 to  
prmSt1-3 (meatware) (pid=9669)
```

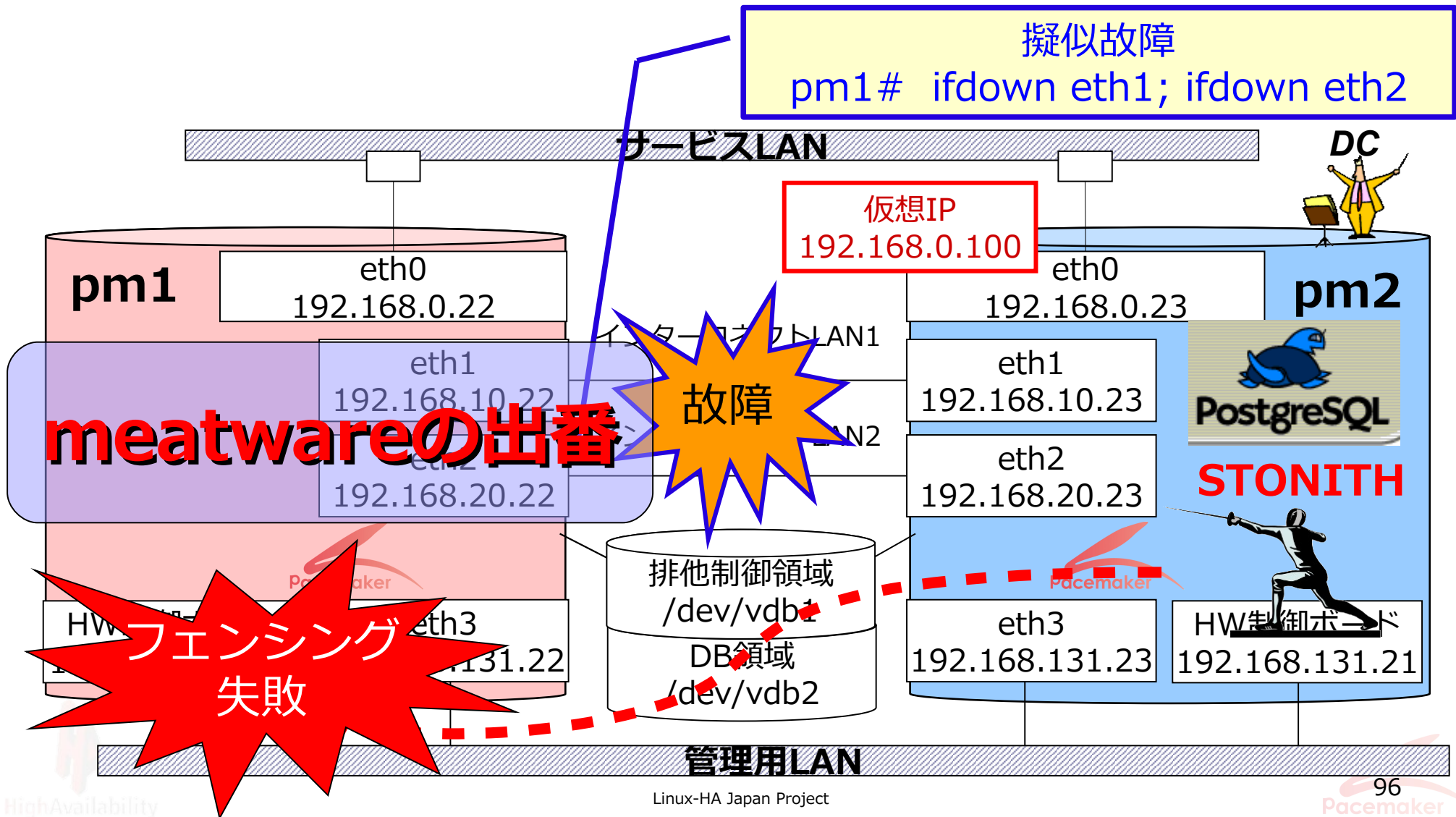


meatclient 実行時は、 こんなログ表示になります。

/var/log/pm_logconv.out

```
Aug 1 18:50:49 pm2 info: Succeeded to STONITH (RESET)  
the Node pm1 by Node pm2.
```

スプリットブレイン & STONITH失敗





⑤

Linux-HA Japanについて

Linux-HA Japan URL

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

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

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。



LINUX-HA JAPAN
High-Availability Clustering on Linux

HOME | メーリングリスト | ダウンロード&インストール | マニュアル | デスクトップテーマ・壁紙等 | コミュニティ概要 | 関連リンク

その他 | ニュース | イベント情報 | 読み物 | WEBラジオ

Linux-HA Japan プロジェクト

Linux上で高可用(HA)クラスシステムを構築するための 部品として、オープンソースの、クラスターリソースマネージャ、クラス通信レイヤ、ブロックデバイス複製、その他、さまざまなアプリケーションに対応するための数多くのリソースエージェント等を、日本国内向けに維持管理、支援等を行っているプロジェクトです。

今は主に Pacemaker, Heartbeat, Corosync, DRBD等を扱っています。

Linux-HA Japan 成果物ダウンロード

RHEL/CentOS向けPacemaker RPMパッケージ(yumのリポジトリ形式)や設定ファイル(crm)作成支援ツール、ディスク監視機能などをダウンロードできます。とりあえずRHELもしくはCentOS等のRHEL互換OSにインストールしてみたい場合は[こちら](#)。インストール後にとりあえず何か動かしてみたい場合は[こちら](#)を参考にしてみてください。

マニュアル

本家コミュニティ提供の公式マニュアルやLinux-HA Japan提供の翻訳マニュアル。マニュアル読んでもよくわからない場合は、過去の[カンファレンス](#)や[勉強会](#)等の発表資料も参考に。

メーリングリスト

インストール方法や設定方法等の質問はMLまで。
※投稿するにはメールアドレスの登録が必要です。

イベント情報

カンファレンスへの出席や講演、勉強会開催情報、講演時のスライド公開など。

開発者向けサイト

Linux-HA Japan開発者向けサイトです。Linux-HA Japan独自開発機能のソースコードやバイナリのダウンロード等。

Twitter 公式ハッシュタグ [#linux_ha_jp](#)

本サイトに関するお問い合わせは、Linux-HA Japan メーリングリスト管理者
(linux-ha-japan-owner@lists.sourceforge.jp) までお願いいたします。

最近の投稿

- Nightlyビルド公開します
- Fedora 19 上で PostgreSQL ストリーミングレプリケーションのクラスターリング
- Linux-HA Japanのリポジトリをgithubに移行しました
- Pacemakerリポジトリパッケージ1.0.13-1.1リリース
- ベースメーカーからベースメーカーへ変更になりました
- OSC 2013 Tokyo/Spring 講演資料公開
- Pacemakerリポジトリパッケージ 1.0.12-1.3 をリリース

よく使われるタグ

April Fool
cluster-glue
corosync DRBD
DRBD-MC
Heartbeat OSC
pacemaker
PostgreSQL あん

Nightlyビルド公開中!

Linux-HA Japanでは、いち早く最新のパッケージを使ってみたいという方のために、RHEL6(x86_64)および互換OS向けのNightlyビルドrpmを公開中です。

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

ビルド対象のパッケージは、Pacemaker 1.0系(安定版)、1.1系(開発版)、および周辺コンポーネントです。

Nightlyビルドの情報については、以下URLを参照してください。

<http://linux-ha.sourceforge.jp/wp/archives/3710>

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、Heartbeat 3、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

ご清聴ありがとうございました。



Linux-HA Japan

検索

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