
VirtualBox と Rocky Linux 8 で始める Pacemaker

～ VirtualBox でも STONITH 機能が試せる! VirtualBMCの活用



2022年10月28日
Linux-HA Japan プロジェクト
<https://linux-ha.osdn.jp/>
森 啓介

- 名前: 森 啓介 (Keisuke MORI)
 - twitter: @ksk_ha
- Linux-HA Japanプロジェクト関連の活動
 - Pacemaker追加ツールのリリース
 - <https://linux-ha.osdn.jp/>
- ClusterLabs プロジェクトのコミット
 - Pacemaker、resource-agents などHAクラスタ関連の開発コミュニティ
 - <https://github.com/ClusterLabs/>
- 本業
 - 普段の業務: NTT OSSセンタ
 - NTTグループ内におけるPacemaker(HA Add-On)の導入支援・サポート
 - バグ報告・パッチ作成などによるNTTから開発コミュニティへのフィードバック・貢献

Pacemaker の STONITH 試験環境を
Windows上の VirtualBox 仮想環境で
作ってみました！

いかがでしたか？

■ Pacemakerとは

- 問題:
仮想環境でPacemaker実験環境を作りたいけど、フェンスエージェントがない!
- 解決案:
STONITHが動作するPacemaker環境を Windows + VirtualBox 上で作ってみた!
- おわりに

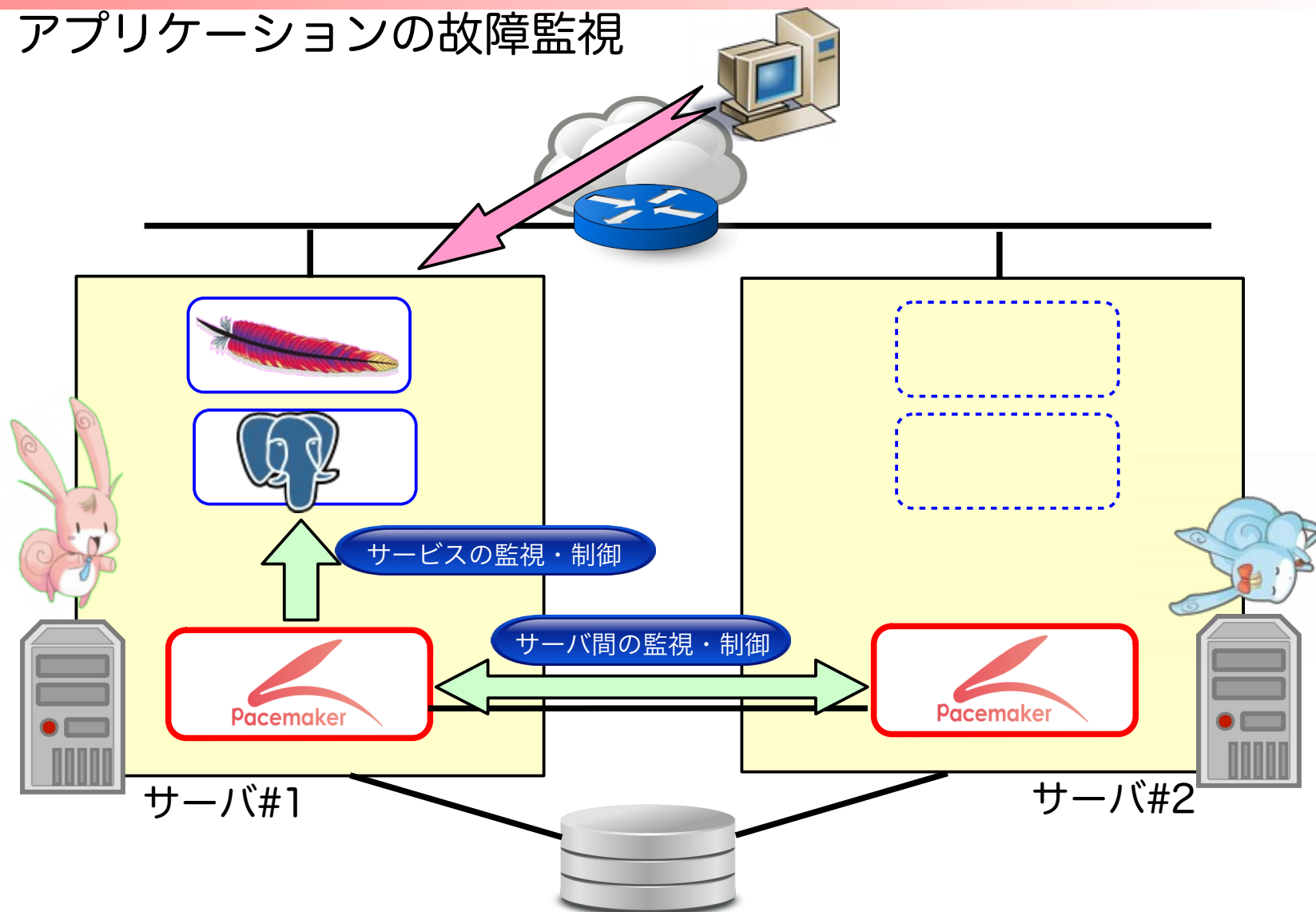
- Pacemakerとは、オープンソースのHAクラスタソフトウェアです。

High **A**vailability = 高可用性

サービスが停止する時間を
できる限り短くする！
こと

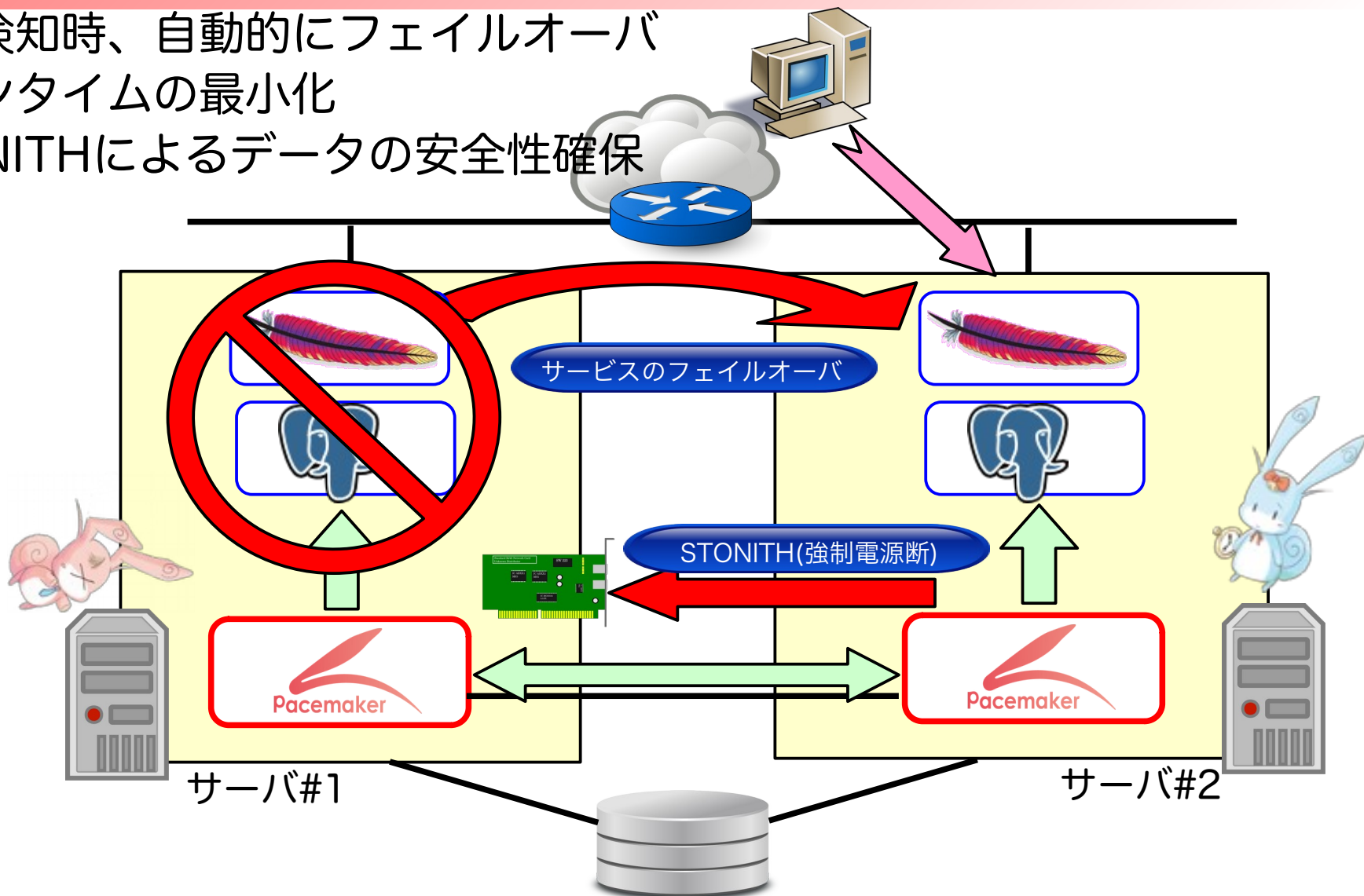
Pacemakerの概要

■ サーバ・アプリケーションの故障監視



Pacemakerの概要

- 故障検知時、自動的にフェイルオーバー
- ダウンタイムの最小化
- STONITHによるデータの安全性確保



STONITH機能(フェンシング)とは？

■ フェンスを立てて隔離すること



■ STONITH機能の目的

- スプリットブレイン対策(排他制御) ⇒ データ破損の防止
- 制御不能な故障ノードの強制停止 ⇒ サービス継続性の向上

■ RHEL 8 HA Add-On では**STONITH機能の利用は必須**です

- stonith-enabled=true (デフォルト設定)



フェンスエージェントの種類(主なもの)

- フェンスエージェント(STONITH機能用のモジュール)は環境に合わせたものを利用します。

環境	フェンスエージェント	説明
物理環境	fence_ipmilan	物理環境用。IPMIによる強制電源断
仮想環境	fence_vmware_rest	VMware仮想環境用。vCenter を経由した仮想マシンの電源断
クラウド環境	fence_aws	AWS環境用。APIを経由した仮想マシンの電源断
	fence_azure	Azure環境用。APIを経由した仮想マシンの電源断
	fence_gcp	GCP環境用。APIを経由した仮想マシンの電源断
共有ディスク環境 (物理/仮想/クラウド)	fence_sbd	共有ディスク領域とwatchdogを利用した電源断 ハードウェアwatchdog が必須
	fence_scsi	SCSIリザーベーション機能を利用したI/Oフェンシング ハードウェアwatchdogが利用不可の環境でも利用可

■ Pacemakerとは

■ 問題:
仮想環境でPacemaker実験環境を作りたいけど、フェンスエージェントがない!

■ 解決案:
STONITHが動作するPacemaker環境を Windows + VirtualBox 上で作ってみた!

■ おわりに

Pacemakerを実際に触ってみたい！けど…？

- 要望: 試しに軽く使ってみたいので、
Windows + VirtualBox で実験用の仮想環境を作りたい！
 - 本番環境は物理マシンでも、いまどき自分の実験に使える物理マシンなんて気軽に用意できるとは限らない
 - 個人ですぐ用意できる環境として VirtualBox はとてもお手軽
- しかし…
- 問題: フェンスエージェントがない！
 - 「STONITH機能は必須」「フェンスエージェントは環境に合わせる」
 - …と言われても、 Windows + VirtualBox 上ではどうすればいいの!?

» ※VMware環境や対応クラウド環境が使えるのであれば、その環境に合わせたフェンスエージェントを使うことができます。しかし設定方法やクラウド固有のAPI仕様など環境ごとの違いに注意する必要があります。

IPMI(ハードウェア制御ボード)は仮想化できないの？

■ 実はそういうものがあります

■ VirtualBMC

- 仮想BMC(Baseboard Management Controller)ソフトウェア
 - IPMI経由で仮想マシンを制御することが可能。
- OpenStack の一プロジェクト
 - <https://github.com/openstack/virtualbmc>
- 動作環境
 - ホストOS: Linux
 - 仮想マシンの制御: libvirt/KVM

■ 今回やったこと: これを Windows 上で動かしてみよう！

- ホストOS: Windows 10 + WSL2 (Ubuntu)
 - (Ubuntuなんだからサクッと動くんじゃね?)
- 仮想マシンの制御: VirtualBox
 - (libvirtのAPIの代わりに VBoxManage コマンドを使うように修正すればいいじゃん)

■ ⇒ これでPacemakerを物理環境と同じ設定で動かせる！

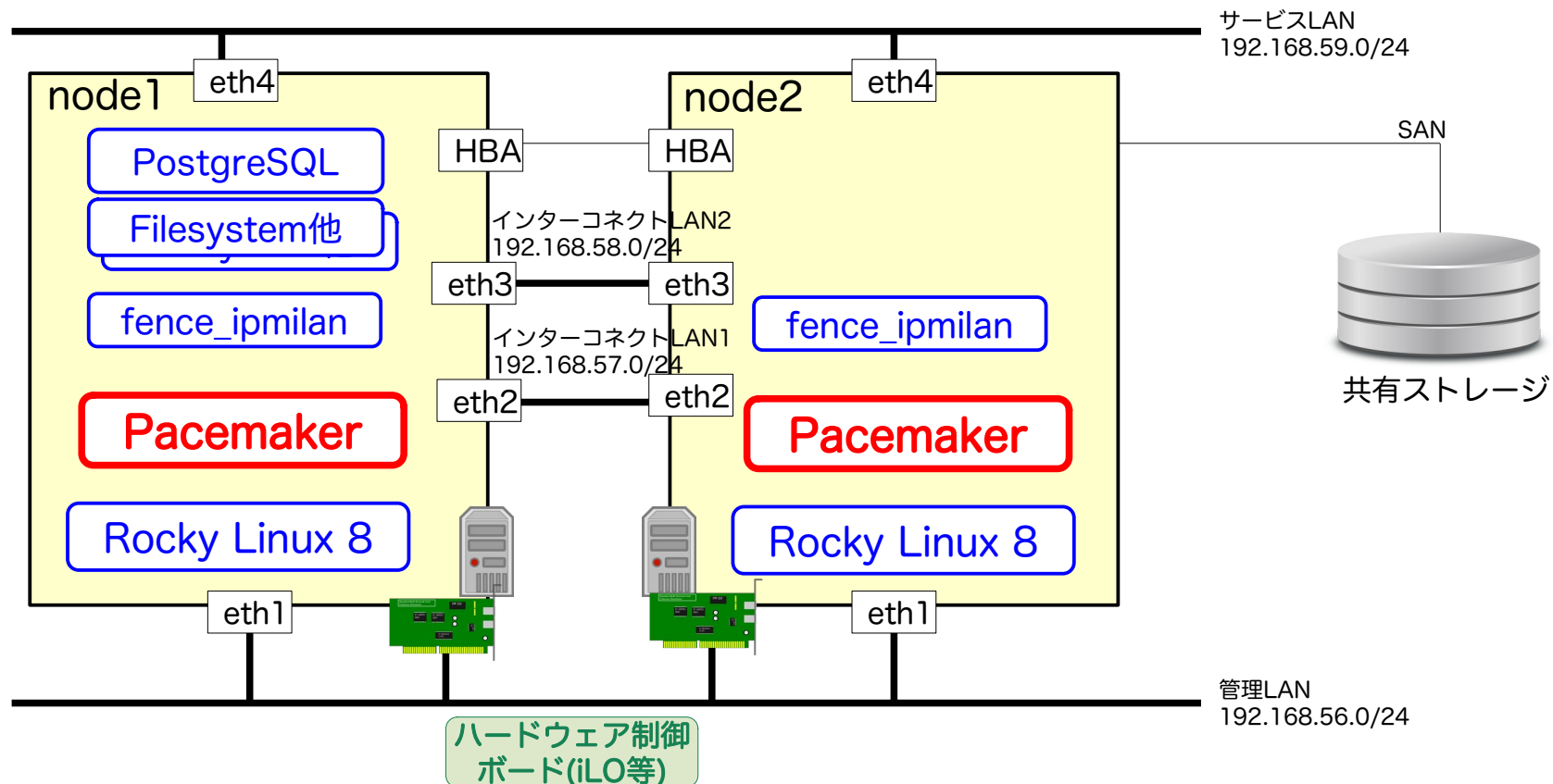
- Pacemakerとは

- 問題:
仮想環境でPacemaker実験環境を作りたいけど、フェンスエージェントがない!

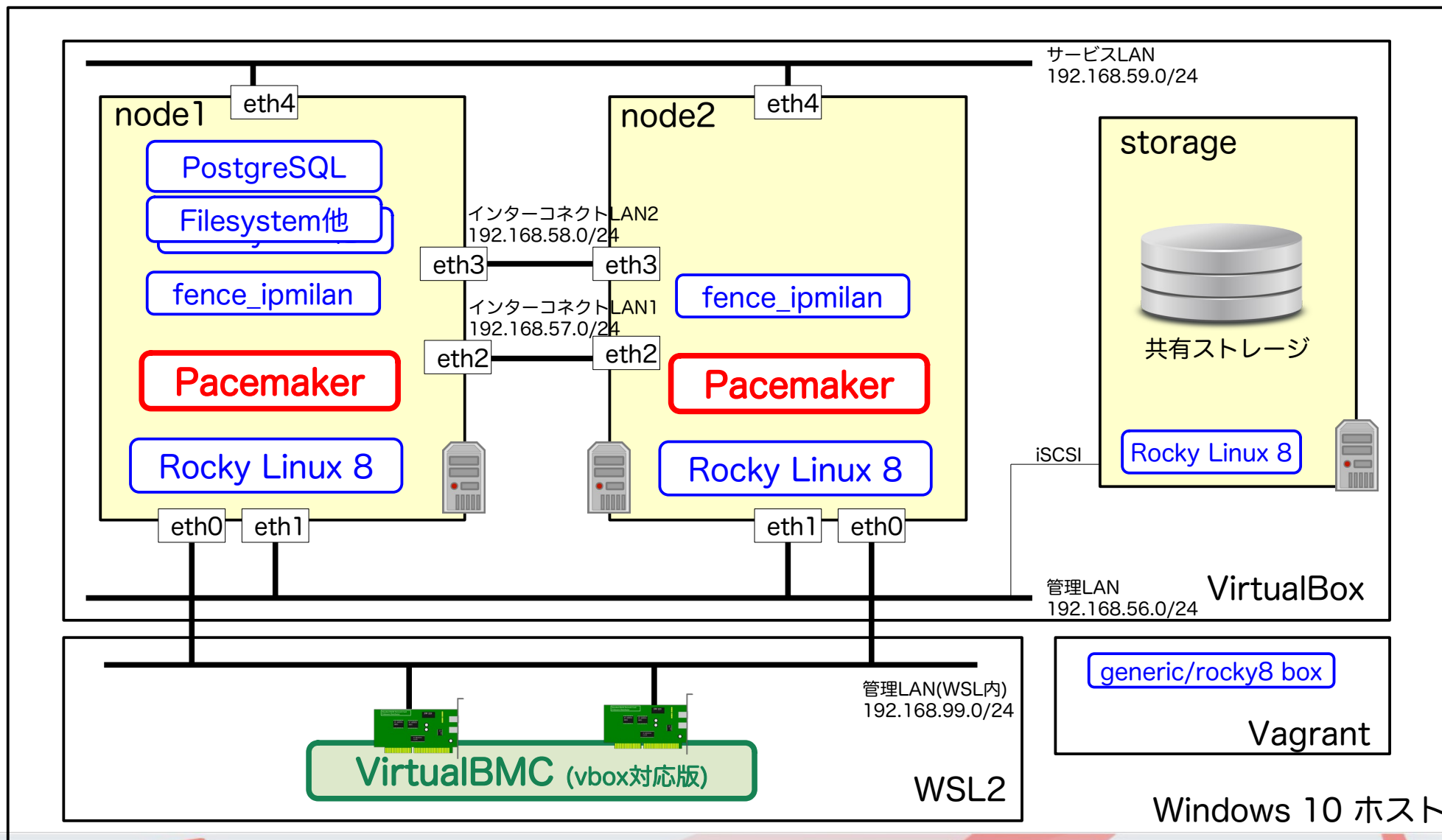
- 解決案:
STONITHが動作するPacemaker環境を Windows + VirtualBox 上で作ってみた!

- おわりに

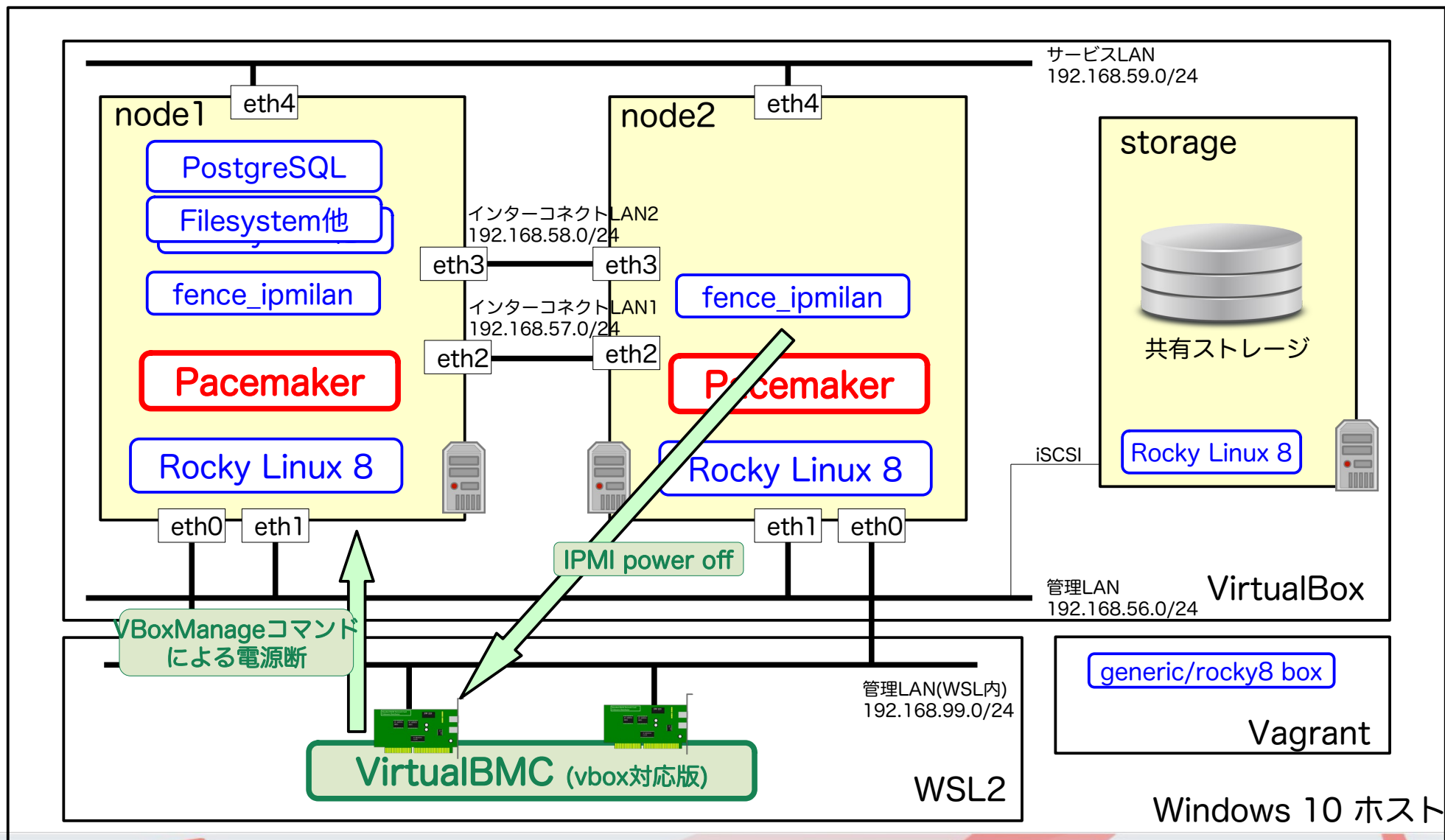
本番環境構成例 (物理環境)



検証用環境構成 (VirtualBox 仮想環境)



検証用環境構成 (VirtualBox 仮想環境)



	物理環境での構築	VirtualBox仮想環境での構築
(0)	-	Windows 10 ホスト上での事前準備
(1)	マシンセットアップ、ネットワーク接続	仮想マシン作成、仮想ネットワーク作成 (vagrant)
(2)	OSインストール	OSインストール (vagrant)
(3)	ハードウェア制御ボード設定(iLO等)	VirtualBMC インストール・設定
(4)	FC共有ディスク設定	iSCSI共有ディスク設定
(5)	PostgreSQL インストール	PostgreSQLインストール
(6)	Pacemaker インストール	Pacemaker インストール
(7)	Pacemaker リソース設定 (fence_ipmilan)	Pacemaker リソース設定 (fence_ipmilan)

(0)Windows 10 ホスト上での事前準備

■ VirtualBox のインストール

- <https://www.virtualbox.org/wiki/Downloads>

■ Vagrant のインストール

- <https://www.vagrantup.com/downloads>

■ WSL2 のインストール

- <https://learn.microsoft.com/ja-jp/windows/wsl/install>

- ディストリビューション: Ubuntu (Ubuntu-20.04 LTS)

```
> wsl --install -d Ubuntu
```

■ ※Symantec Endpoint Protection (SEP) を利用している場合

- WSL2から外部への通信を行うための設定が必要

 - » 設定の変更 → ネットワークとホストのエクспロイト緩和機能 → オプションの設定
→ 不一致トラフィックの設定 → IPトラフィックを許可する

- 参考情報 (他の対処方法など)

 - » <https://kemasoft.net/?vm/wsl2%A4%C8SEP%A4%C8stone#w98afb3e>

 - » <https://computational-sediment-hyd.hatenablog.jp/entry/2022/05/02/231428>

(0)-2 Windows 10 ホスト上での事前準備

■ WSL2 の設定 (必要に応じて設定)

□ ansible-playbook (後述)を利用する場合の設定

```
$ sudo vi /etc/wsl.conf  
[automount]  
options = "metadata"
```

Windowsファイルシステム上
(/mnt/c/配下)でも、playbookや
ssh鍵のパーミッションが
正しく設定可能となる。

□ proxy環境での設定

```
$ sudo vi /etc/environment  
  
http_proxy=http://PROXY:8080/  
https_proxy=http://PROXY:8080/
```

※ proxy環境変数は両方で
設定しておいた方が無難

```
$ vi $HOME/.bashrc  
  
export http_proxy=http://PROXY:8080/  
export https_proxy=http://PROXY:8080/  
#export VAGRANT_CWD=/mnt/c/Users/xxx/vagrant  
export WSLENV=VAGRANT_CWD/p:http_proxy:https_proxy
```

※ vagrantを使う場合は好みで
設定しておくとう便利

□ WSL2の再起動 (設定の反映)

```
$ exit  
> wsl --shutdown  
> wsl  
$
```

(0)-3 Windows 10 ホスト上での事前準備

■ WSL2 上のansibleのインストール (必要に応じて設定)

- ansible-playbook (後述)を利用する場合にインストールする
- Ansible インストール手順

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

□ 参考

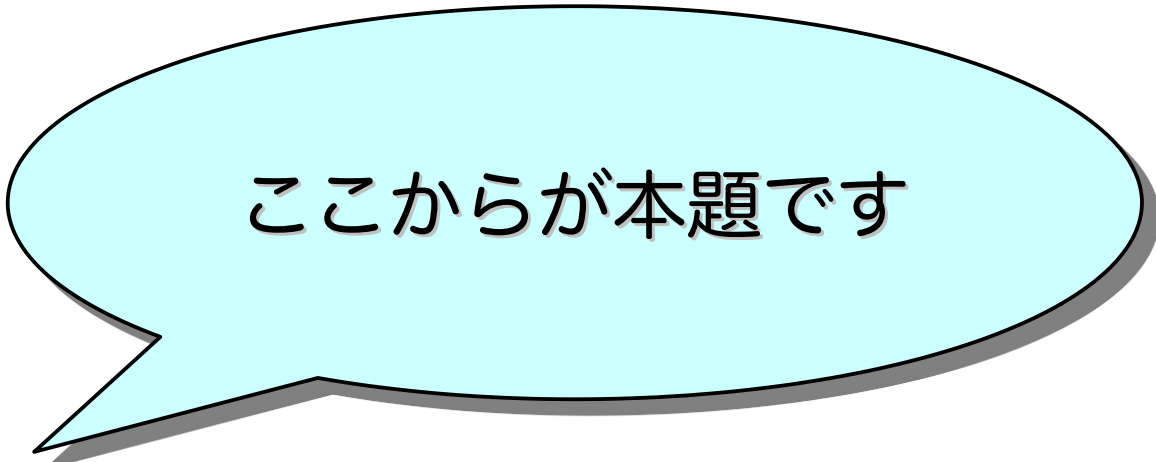
■ 公式のインストール手順

- https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu

■ proxy環境でエラーが出る場合は以下の設定も追加

```
$ sudo vi /etc/apt/apt.conf.d/99proxy
```

```
Acquire::http::proxy "http://PROXY:8080/";
Acquire::https::proxy "http://PROXY:8080/";
Acquire::http::Timeout "300";
```



ここからが本題です

(3)-1 VirtualBMC インストール手順(WSL2内で実行)

■ VirtualBMC (vbox対応版)のチェックアウト

```
$ git clone https://github.com/kskmori/virtualbmc-vbox
$ cd virtualbmc-vbox
$ git checkout devel-vbox-2.0
$ cd ..
```

vbox 対応版の
ブランチをチェックアウト

■ Python 仮想環境の作成とインストール

```
$ python3 -m venv --without-pip ./venv-vbmc
$ . ./venv-vbmc/bin/activate
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
$ python get-pip.py
$ pip install --upgrade pip
$ pip install -e ./virtualbmc-vbox/
$ deactivate
```

WSL2では pip を個別に
インストールする必要あり

■ VirtualBMC 設定ファイルの作成

```
$ sudo mkdir /root/.vbmc/
$ sudo vi /root/.vbmc/virtualbmc.conf
```

```
[log]
logfile: /var/log/virtualbmc.log
#debug: true

[vbox]
vbox_user: USER
```

vbox対応版固有の設定項目
WSL2のユーザ名(\$USER)
を設定

(3)-2 VirtualBMC 起動手順(WSL2内で実行)

■ 管理LAN(WSL内)の設定(要管理者権限)

```
$ powershell.exe Start-Process -Verb RunAs powershell.exe \  
-ArgumentList "'-Command New-NetIPAddress -IPAddress 192.168.99.1 \  
-PrefixLength 24 -InterfaceAlias 'vEthernet (WSL)'"'
```

■ VirtualBMCデーモン(vbmcd)の起動

```
$ sudo ./venv-vbmc/bin/vbmcd
```

WSL2内でIPアドレスを固定するために
個別のサブネット用アドレスを設定

■ IPMI受付用IPアドレスの設定 (2ノード分)

```
$ sudo ip addr add 192.168.99.91 dev eth0  
$ sudo ip addr add 192.168.99.92 dev eth0
```

■ VirtualBMC ノード追加設定 (2ノード分)

```
$ sudo ./venv-vbmc/bin/vbmc add --username pacemaker \  
--password pacemakerpass1 --address 192.168.99.91 node1  
$ sudo ./venv-vbmc/bin/vbmc add --username pacemaker \  
--password pacemakerpass1 --address 192.168.99.92 node2
```

ハードウェア制御ボード
(iL0等)の設定に相当する

■ VirtualBMC 実行開始 (2ノード分)

```
$ sudo ./venv-vbmc/bin/vbmc start node1  
$ sudo ./venv-vbmc/bin/vbmc start node2
```

(3)-3 VirtualBMC 状態確認(WSL2内で実行)

■ VirtualBMC 状態確認 実行例

```
$ sudo ./venv-vbmc/bin/vbmc list
```

Domain name	Status	Address	Port
node1	running	192.168.99.91	623
node2	running	192.168.99.92	623

	物理環境での構築	VirtualBox仮想環境での構築	
(0)	-	Windows 10 ホスト上での事前準備	済
(1)	マシンセットアップ、ネットワーク接続	仮想マシン作成、仮想ネットワーク作成 (vagrant)	省略
(2)	OSインストール	OSインストール (vagrant)	省略
(3)	ハードウェア制御ボード設定(iLO等)	VirtualBMC インストール・設定	済
(4)	FC共有ディスク設定	iSCSI共有ディスク設定	省略
(5)	PostgreSQL インストール	PostgreSQLインストール	省略
(6)	Pacemaker インストール	Pacemaker インストール	省略
(7)	Pacemaker リソース設定 (fence_ipmilan)	Pacemaker リソース設定 (fence_ipmilan)	

(6)-1 Pacemaker インストール手順(ゲスト上で実行)

■ Pacemaker のインストール (両ノードで実行)

```
# dnf install pcs pacemaker fence-agents-all --enablerepo=ha
# passwd hacluster
New password:
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

Rocky Linux 8 では
ha リポジトリを有効化します

■ pm extra tools (Linux-HA Japan追加ツール)のインストール(両ノードで実行)

```
# dnf install pm_extra_tools-1.4-1.el8.noarch.rpm
```

■ ダウンロードURL: <https://linux-ha.osdn.jp/wp/dl>

■ firewall設定 (両ノードで実行)

```
# firewall-cmd --add-service=high-availability
# firewall-cmd --permanent --add-service=high-availability
```

■ クラスタの作成 (いずれか一つのノードで実行)

```
# pcs host auth node1 addr=192.168.56.11 node2 addr=192.168.56.12 \
-u hacluster
Password:
# pcs cluster setup cluster_name \
node1 addr=192.168.57.11 addr=192.168.58.11 \
node2 addr=192.168.57.12 addr=192.168.58.12
```

※ proxy環境での注意: pcs コマンド実行時は proxy 設定を無効化しておく必要があります。
unset https_proxy HTTPS_PROXY もしくは
export no_proxy=192.168.56.11,192.168.56.12 を設定しておくなど

(6)-2 Pacemaker 追加設定 (ゲスト上で実行)

- /etc/sysconfig/pacemaker 設定 (両ノードで設定)
 - 故障検知時のfail_fast動作 (Linux-HA Japan での推奨設定)

```
# vi /etc/sysconfig/pacemaker
```

```
(...)  
PCMK_fail_fast=yes  
PCMK_panic_action=sync-reboot
```

- fence_ipmilan 用設定 (両ノードで設定)
 - ACPI soft-off 動作の抑止

```
# vi /etc/systemd/logind.conf
```

```
(...)  
HandlePowerKey=ignore
```

```
# systemctl restart systemd-logind.service
```

※ 詳細は Red Hat社ナレッジ参照

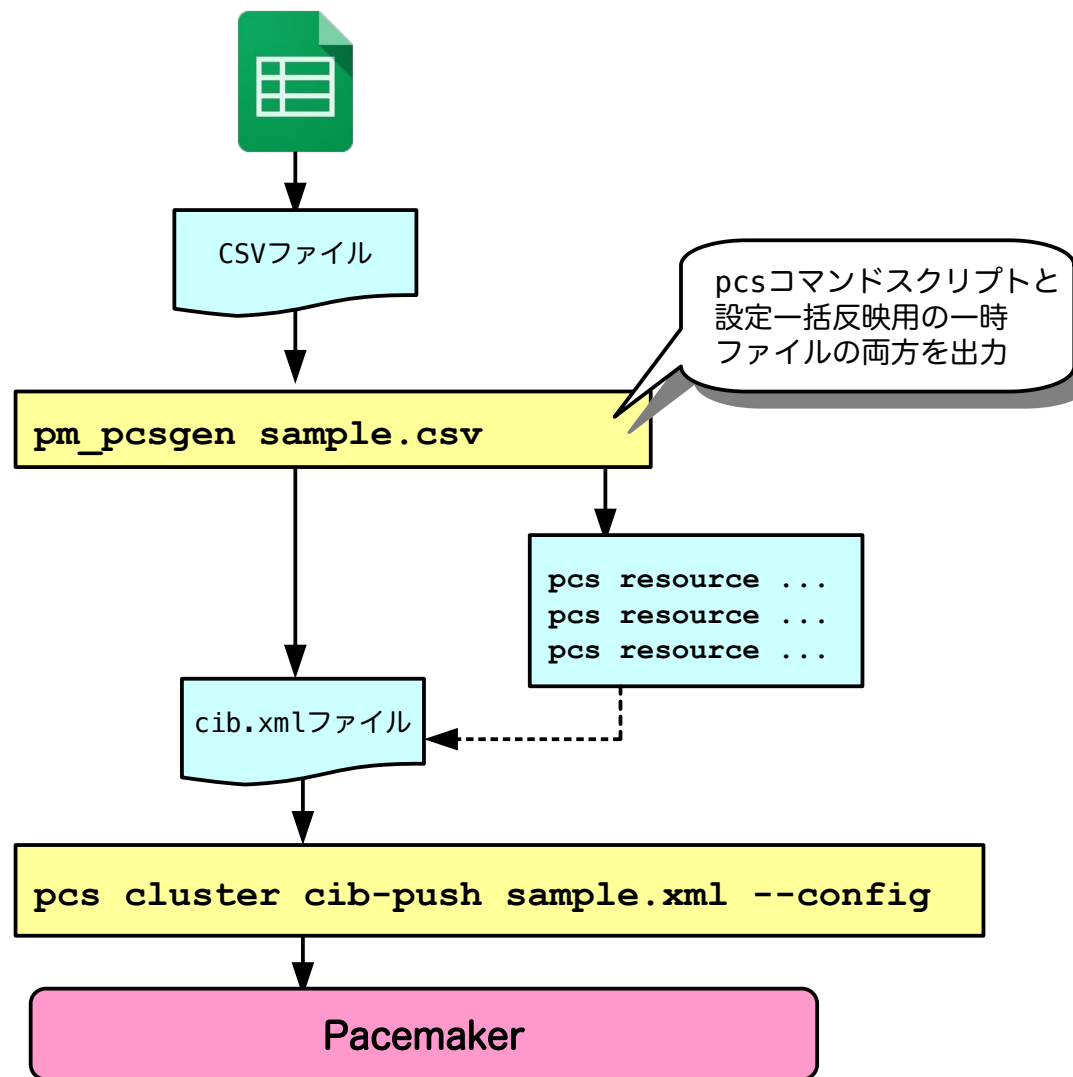
(7) pm_pcsgen によるPacemakerリソース設定

パラメタシートの作成(xlsx, ods)

CSVでエクスポート

設定ファイルの変換

設定の一括反映



(7)pm_pcsngen 設定例(fence_ipmilan 抜粋)

#表 8-1 クラスタ設定 ... STONITHリソース

STONITH

P	id	type			
#	STONITHリソースID	type	概要		
	fence1-ipmilan	fence_ipmilan			
A	type	name	value		
#	パラメータ種別	項目	設定内容	概要	
	options	pcmk_host_list	node1		
		ip	192.168.99.91	IPMIポートのIPアドレス	
		username	pacemaker		
		password	pacemakerpass1		
		lanplus	1		
O	type	timeout	interval	on-fail	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	備考
	start	60s		restart	
	monitor	60s	3600s	restart	
	stop	60s		ignore	

STONITH					
P	id	type			
#	STONITHリソースID	type	概要		
	fence2-ipmilan	fence_ipmilan			
A	type	name	value		
#	パラメータ種別	項目	設定内容	概要	
	options	pcmk_host_list	node2		
		ip	192.168.99.92	IPMIポートのIPアドレス	
		username	pacemaker		
		password	pacemakerpass1		
		lanplus	1		
O	type	timeout	interval	on-fail	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	備考
	start	60s		restart	
	monitor	60s	3600s	restart	
	stop	60s		ignore	

https://github.com/kskmori/osc2022fall-demo/blob/main/pm_pcsngen-config/pm_pcsngen_osc2022fall.xlsx

Pacemaker 起動画面(構築完了)

```
# pcs status
Cluster name: cluster_name
Cluster Summary:
  * Stack: corosync
  * Current DC: node1 (version 2.1.2-4.el8_6.2-ada5c3b36e2) - partition with quorum
  * Last updated: Fri Oct 21 09:55:38 2022
  * Last change: Thu Oct 20 13:24:37 2022 by root via crm_resource on node1
  * 2 nodes configured
  * 9 resource instances configured

Node List:
  * Online: [ node1 node2 ]

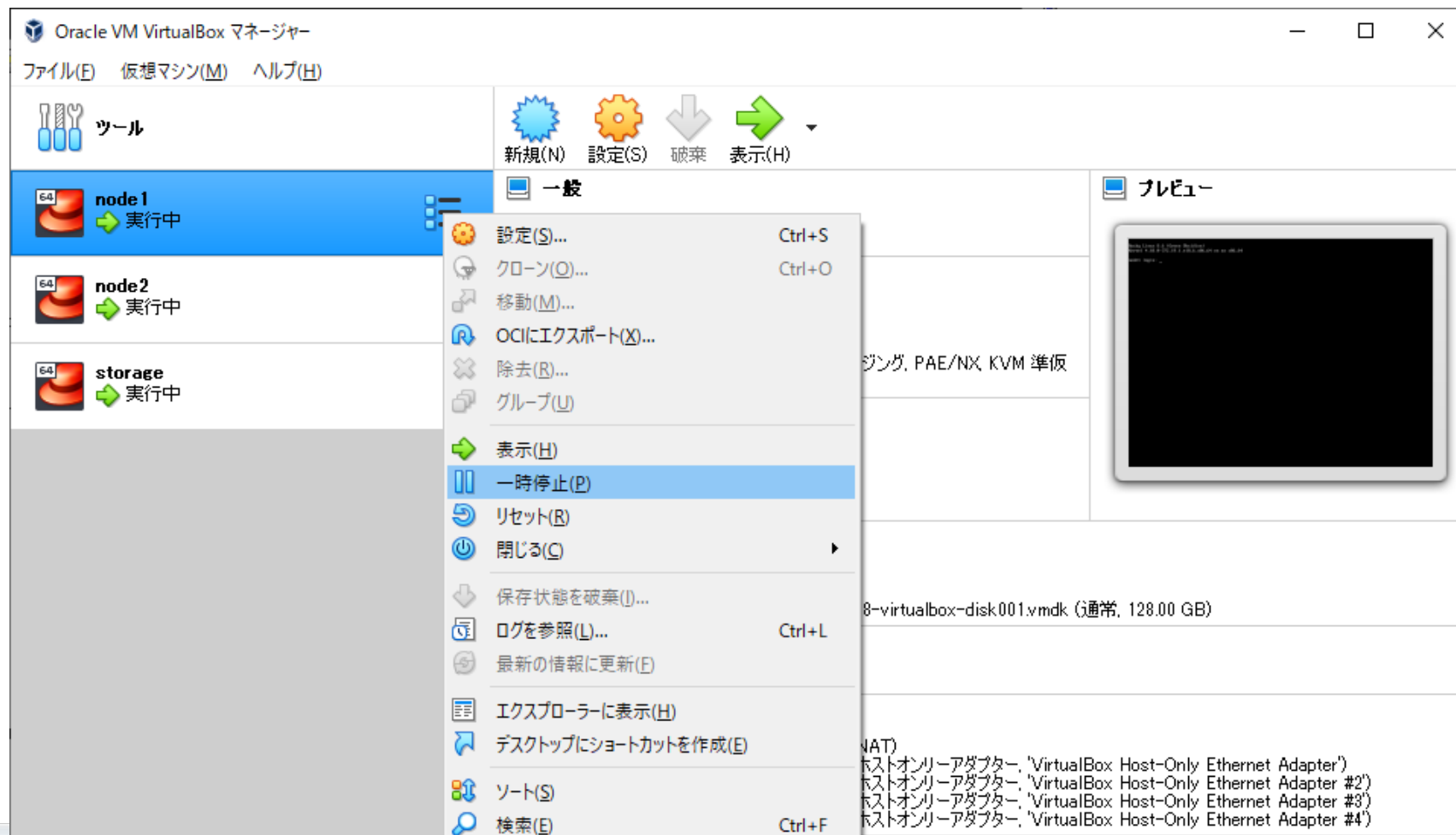
Full List of Resources:
  * Resource Group: pgsql-group:
    * filesystem1 (ocf::heartbeat:Filesystem): Started node1
    * filesystem2 (ocf::heartbeat:Filesystem): Started node1
    * filesystem3 (ocf::heartbeat:Filesystem): Started node1
    * ipaddr (ocf::heartbeat:IPaddr2): Started node1
    * pgsql (ocf::linuxha:pgsql): Started node1
  * Clone Set: ping-clone [ping]:
    * Started: [ node1 node2 ]
  * fence1-ipmilan (stonith:fence_ipmilan): Started node2
  * fence2-ipmilan (stonith:fence_ipmilan): Started node1

Daemon Status:
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled
```

fence_ipmilan を利用
したSTONITH構成

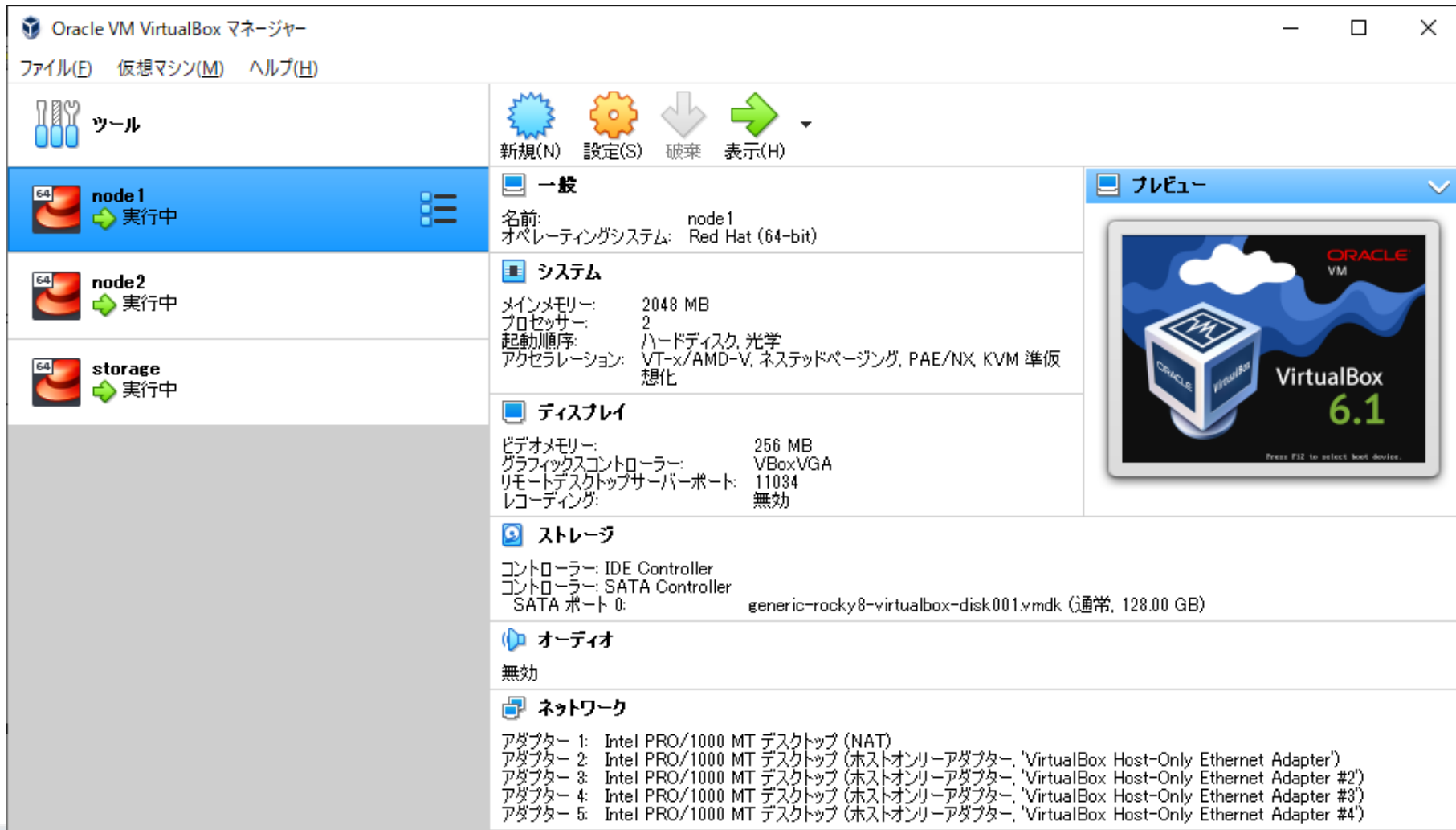
■ 故障事例:

□ 一時的なカーネルハング(VirtualBox 一時停止でエミュレーション)



デモ:PacemakerのSTONITH実行例(結果)

■ STONITH実行による node1 再起動発生



The screenshot displays the Oracle VM VirtualBox Manager interface. On the left, a list of VMs shows 'node1' as '実行中' (Running). The main pane shows the settings for 'node1', which is a Red Hat (64-bit) VM. The settings are categorized into several sections:

- 一般 (General):** Name: node1, Operating System: Red Hat (64-bit).
- システム (System):** Main Memory: 2048 MB, Processor: 2, Boot Order: Hard Disk, Optical, Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM 準仮想化 (Enabled).
- ディスプレイ (Display):** Video Memory: 256 MB, Graphics Controller: VBoxVGA, Remote Desktop Server Port: 11034, Recording: Disabled.
- ストレージ (Storage):** Controller: IDE Controller, Controller: SATA Controller, SATA Port 0: generic-rocky8-virtualbox-disk.001.vmdk (通常, 128.00 GB).
- オーディオ (Audio):** Disabled.
- ネットワーク (Network):** Adapter 1: Intel PRO/1000 MT デスクトップ (NAT), Adapter 2: Intel PRO/1000 MT デスクトップ (ホストオンリーアダプター, 'VirtualBox Host-Only Ethernet Adapter'), Adapter 3: Intel PRO/1000 MT デスクトップ (ホストオンリーアダプター, 'VirtualBox Host-Only Ethernet Adapter #2'), Adapter 4: Intel PRO/1000 MT デスクトップ (ホストオンリーアダプター, 'VirtualBox Host-Only Ethernet Adapter #3'), Adapter 5: Intel PRO/1000 MT デスクトップ (ホストオンリーアダプター, 'VirtualBox Host-Only Ethernet Adapter #4').

On the right, a 'プレビュー' (Preview) window shows the VirtualBox 6.1 logo and the text 'Press F12 to select boot device.'

デモ:STONITH実行後の状態

```
# pcs status --full
Cluster name: cluster_name
Cluster Summary:
  * Stack: corosync
  * Current DC: node2 (2) (version 2.1.2-4.el8_6.2-ada5c3b36e2) - partition with quorum
  * Last updated: Fri Oct 21 10:22:33 2022
  * Last change: Thu Oct 20 13:24:37 2022 by root via crm_resource on node1
  * 2 nodes configured
  * 9 resource instances configured
```

Node List:

```
  * Online: [ node2 (2) ]
  * OFFLINE: [ node1 (1) ]
```

Full List of Resources:

```
  * Resource Group: pgsql-group:
    * filesystem1 (ocf::heartbeat:Filesystem): Started node2
    * filesystem2 (ocf::heartbeat:Filesystem): Started node2
    * filesystem3 (ocf::heartbeat:Filesystem): Started node2
    * ipaddr      (ocf::heartbeat:IPAddr2):    Started node2
    * pgsql       (ocf::linuxhajp:pgsql):      Started node2
  * Clone Set: ping-clone [ping]:
    * ping        (ocf::pacemaker:ping):       Started node2
    * ping        (ocf::pacemaker:ping):       Stopped
    * fence1-ipmilan (stonith:fence_ipmilan):   Started node2
    * fence2-ipmilan (stonith:fence_ipmilan):   Stopped
```

(一部略)

Fencing History:

```
  * reboot of node1 successful: delegate=node2, client=pacemaker-controld.8732,
    origin=node2, completed='2022-10-21 10:21:07Z'
```

node1 はOFFLINE

サービスは node2 へ
フェイルオーバー

STONITHの実行履歴

■ その他のSTONITHが発動する故障事例:

□ インターコネクトLAN全断

■ 再現手順例

- VirtualBox による疑似切断: 設定→ネットワーク→アダプター(1~4)→高度→ケーブル接続

■ 想定動作

- node2 (スタンバイノード)の電源断 (アクティブノード優先設定のため)

□ アプリケーションプロセスハング(停止不可)

■ 再現手順例

- kill -STOP (PostgreSQLプロセスPID)

■ 想定動作

- pgsql RA monitor タイムアウト(60s)、stop タイムアウト(300s)発生後、node1 電源断、フェイルオーバーが発生

ansible-playbookも用意しています

■ VirtualBMC インストール用 playbook リポジトリ

```
$ git clone https://github.com/kskmori/ansible-virtualbmc
```

■ インベントリの設定

□ 必要に応じて編集

```
$ cd ansible-virtualbmc  
$ cp ansible.cfg.sample ansible.cfg  
$ cp inventories/all.yml.sample-wsl2 inventories/all.yml
```

■ インストール

□ 「(3)-1 VirtualBMC インストール手順(WSL2内で実行)」を自動実行

```
$ ansible-playbook 10-vbmc-install.yml -K
```

■ 起動

□ 「(3)-2 VirtualBMC 起動手順(WSL2内で実行)」を自動実行

■ Windows ホストを再起動した後はここから実行する

```
$ ansible-playbook 20-vbmc-start.yml -K
```

■ 今回のデモ環境全体を構築する playbook です

□ 以下の(1)～(7)の環境構築を実行します

```
$ git clone --recursive https://github.com/kskmori/osc2022fall-demo
```

	VirtualBox仮想環境での構築	playbook / 個別playbookリポジトリ
(0)	Windows 10 ホスト上での事前準備	(なし)
(1)	仮想マシン作成、仮想ネットワーク作成 (vagrant)	10-vagrant.yml
(2)	OSインストール (vagrant)	
(3)	VirtualBMC インストール・設定	20-virtualbmc.yml https://github.com/kskmori/ansible-virtualbmc
(4)	iSCSI共有ディスク設定	30-postgresql-shared.yml https://github.com/kskmori/ansible-postgresql-shared
(5)	PostgreSQLインストール	
(6)	Pacemaker インストール	40-pacemaker.yml https://github.com/kskmori/ansible-pacemaker-rocky8
(7)	Pacemaker リソース設定 (fence_ipmilan)	

■ Pacemakerとは

■ 問題:
仮想環境でPacemaker実験環境を作りたいけど、フェンスエージェントがない!

■ 解決案:
STONITHが動作するPacemaker環境を Windows + VirtualBox 上で作ってみた!

■ おわりに

- Windows + VirtualBox 仮想環境上で Pacemaker の STONITH動作環境ができました!
- 想定用途
 - Pacemakerの学習・習熟用
 - 本番環境(物理環境)構築前のPacemaker設定事前確認用
- 注意点
 - 商用環境では利用できません
 - 同一物理ノード上でクラスタを構築しても冗長性はありません。
 - VirtualBMCが新たな単一故障点(SPOF)になりえます。

- 疑問:
「そもそもクラウド全盛のこの時代に、物理ノードのHAクラスタなんているの？ 令和やぞ？」
 - それはそう
 - でも適用領域が減っているとはいえ、必要とされる場面はある

- クラウド・コンテナ時代のHAクラスタの使われ方
 - クラウド上でのサービス監視
 - クラウド基盤だけでは対応できない個別のアプリケーション監視
 - クラウド・コンテナ基盤の冗長化
 - OpenStack コントローラノードの冗長化など
 - オンプレミスとクラウドのハイブリッド
 - 全てのサービスがクラウド化に適しているとは限らない

Pacemakerをさらに詳しく知りたかったら…

■ Linux-HA Japan ウェブサイトURL

<https://linux-ha.osdn.jp/>

□ メーリングリストへのご参加もお待ちしております。



Pacemaker 応援キャラクター



■ Linux-HA Japan をこれからもよろしくお願いします!

