

試して覚える Pacemaker入門

AzureでPacemakerを使ってみよう

2024-03-01 OSC2024 Online/Spring
Linux-HA Japan プロジェクト
関口 和哉



自己紹介

- ・関口 和哉(せきぐち かずや)
- ・OSCは今回が初参加です。
- ・普段はNTT OSSセンタという所にいます

本日の流れ

- Pacemakerとは？
- Pacemakerを試したいけど
テキトーな環境が無い
- Microsoft Azureで
Pacemakerを試してみよう！

注意点

- ・「Pacemakerを試す」ことを主にしています。
- ・Azureにおいて効果的な高可用性を保証する構成ではない点にご注意ください。
- ・おおまかに以下のドキュメントに沿います。
 - ・ MS Learn - Azure の Red Hat Enterprise Linux に Pacemaker を設定する
 - ・ <https://learn.microsoft.com/ja-jp/azure/sap/workloads/high-availability-guide-rhel-pacemaker?tabs=msi>

Pacemakerとは？

Pacemakerとは？

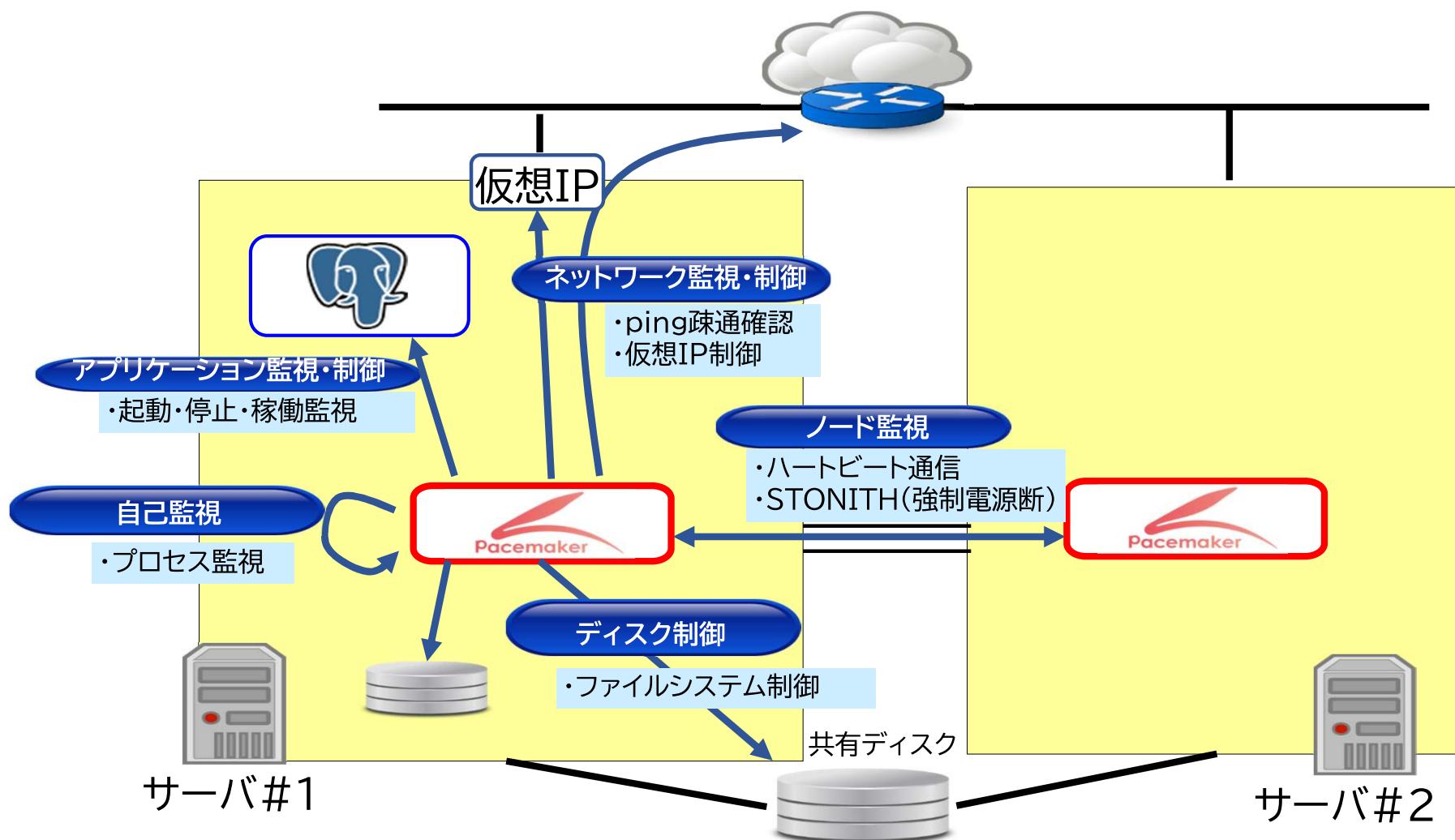
- Pacemakerとは、
オープンソースのHAクラスタソフトウェアです。
- ClusterLabsのコミュニティで管理されています。
 - <https://github.com/ClusterLabs>

High Availability = 高可用性

- システムサービスが停止する時間をできる限り短く
することが目的です。

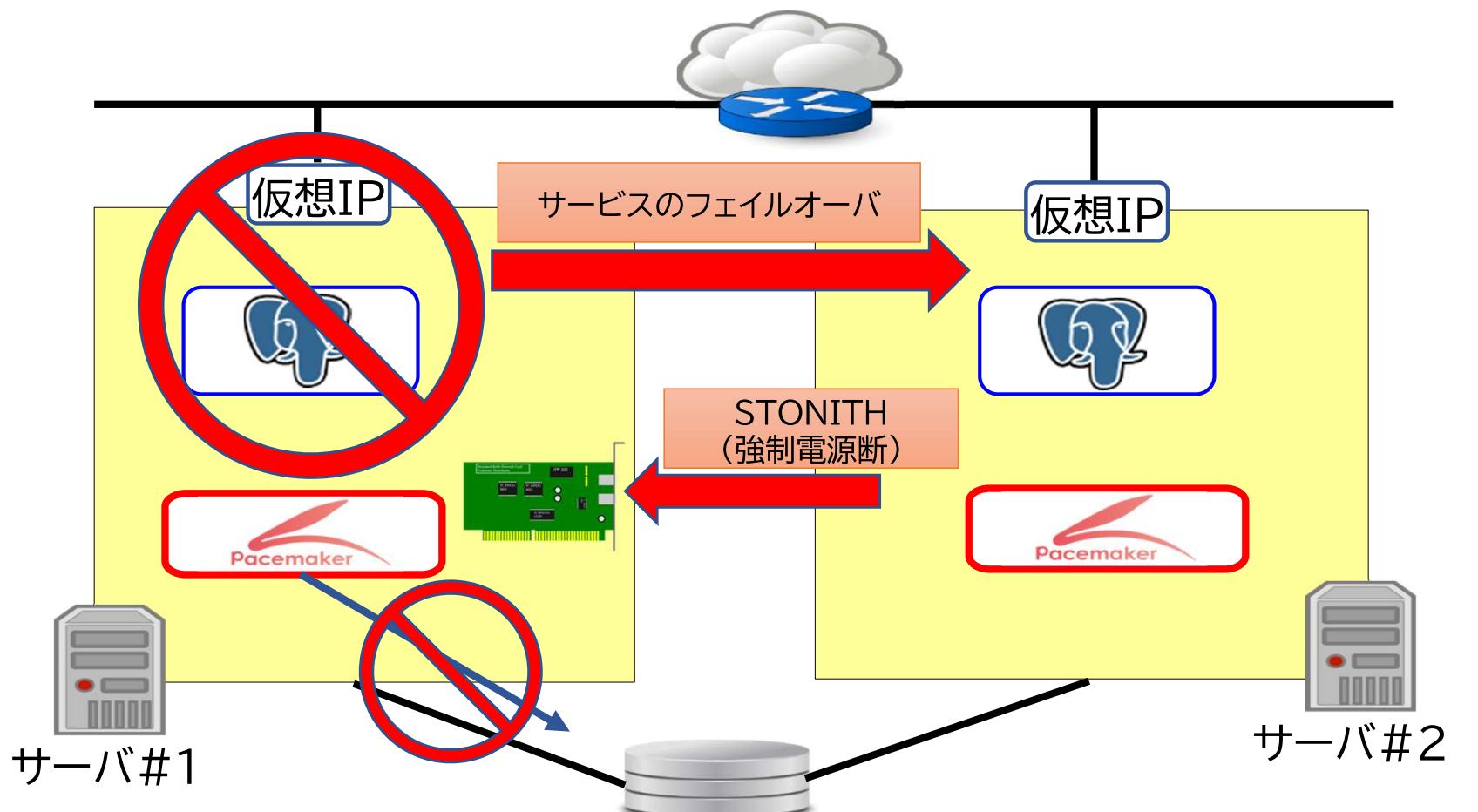
Pacemakerとは？

Pacemakerは様々な対象を起動/停止/監視することが可能



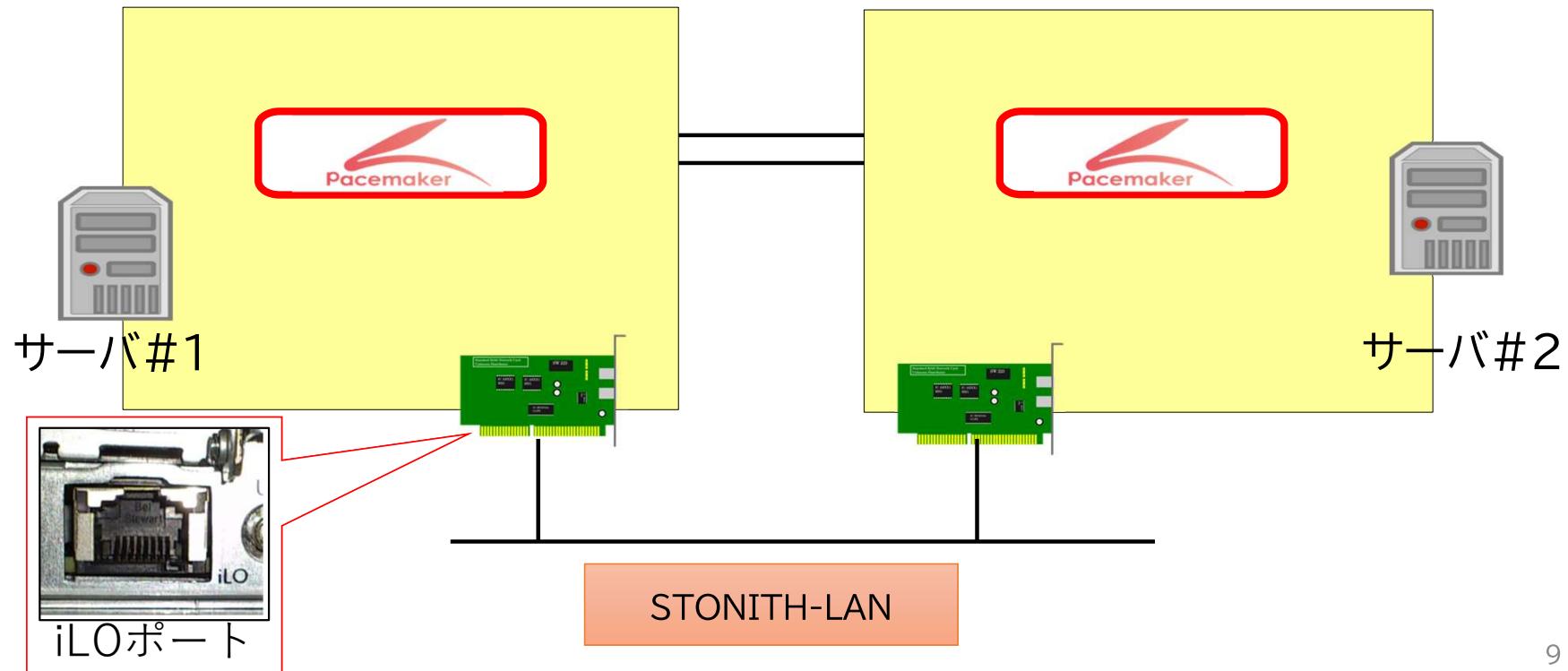
Pacemakerとは？

- ・故障検知時に自動的にフェイルオーバ
- ・ダウントIMEの最小化
- ・STONITHによるデータの安全性確保



スプリットブレインとSTONITH

- STONITH(**Shoot The Other Node In The Head**)
- スプリットブレイン(両系がActive状態)になる前に、対向ノードの強制電源断を実行する機能(排他制御機能)
- 強制電源断することを**フェンシング(fencing)**と呼びます。
- サーバ付属のリモートHW制御ボード(iLOなど)を操作



スプリットブレインとSTONITH

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



スプリットブレインとSTONITH

- 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
- Red Hatでは、STONITH機能の利用は必須であり、機能を無効化したものやフェンシングの設定を行っていないクラスタノードがある場合はサポートしないと明記されています。
 - クラスタのプロパティ stonith-enabled=true
 - デフォルト設定でtrueのため、意図的に無効化しなければ問題ない。
 - Support Policies for RHEL High Availability Clusters - General Requirements for Fencing/STONITH
 - <https://access.redhat.com/articles/2881341>

Pacemakerを試すには

Pacemakerを試すには

- STONITH機能を含めた確認までしないと「試した」と言えません。
- これらを準備するのは、少し手間がかかります。

ベアメタルで試すには

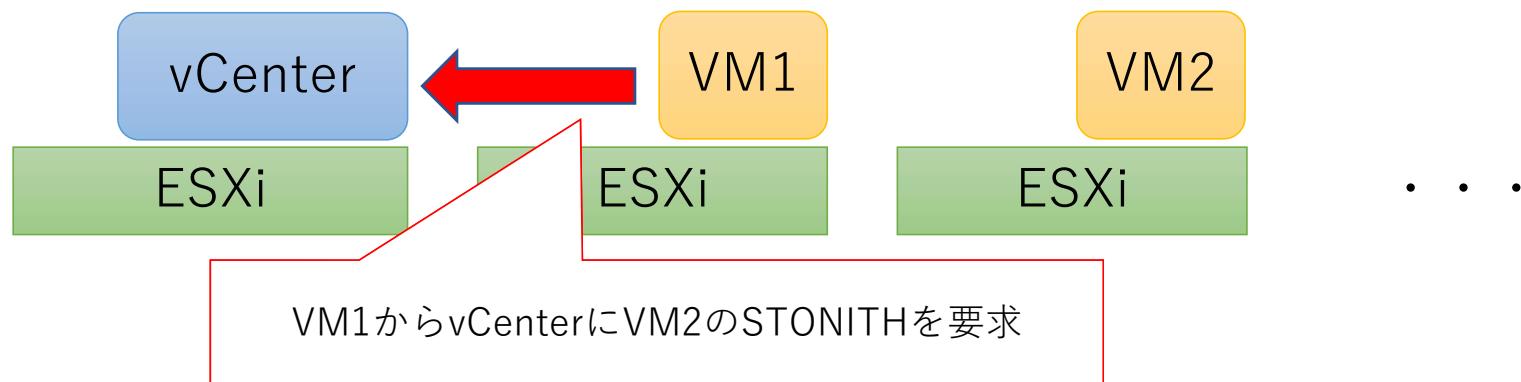
- ・ベアメタルサーバでは、NWやIPMIの設定まで必要。
 - ・IPMI… Intelligent Platform Management Interface
 - ・OSとは独立して機能する、マシンを管理するための規格。
 - ・この規格に沿って、各サーバーメーカーが機能を用意しています。

企業	実装名
Dell	Integrated Dell Remote Access Controller (iDRAC)
HPE	HPE Integrated Lights Out - iLO
富士通	integrated Remote Management Controller (iRMC)

- ・フェンスエージェント fence_ipmilan を使用します
 - ・ https://github.com/ClusterLabs/fence-agents/blob/main/agents/ipmilan/fence_ipmilan.py

vSphere環境で試すには

- VMware vSphere環境では、仮想マシンからvCenterにフェンシングリクエストを送れるよう設定が必要。



- vCenterにSTONITH用の権限を設定したユーザーなどの認証・認可設定が必要になります。
- フェンスエージェントの例:fence_vmware_rest
 - https://github.com/ClusterLabs/fence-agents/blob/main/agents/vmware_rest/fence_vmware_rest.py

Pacemakerを試すには

- ・どちらも手軽に試すには少し面倒。
 - ・各設定のほか、RHELならHigh Availability Add-Onの購入も必要です。
- ・そこでクラウドで試してみましょう！
- ・ここではMicrosoft Azureを使用し、以下の代表的なエンジニアメントを使用した構成を作成します。
 - ・azure-lb
 - ・fence_azure_arm

Microsoft Azure で
Pacemakerを試してみよう

AzureでPacemakerを動作させるには

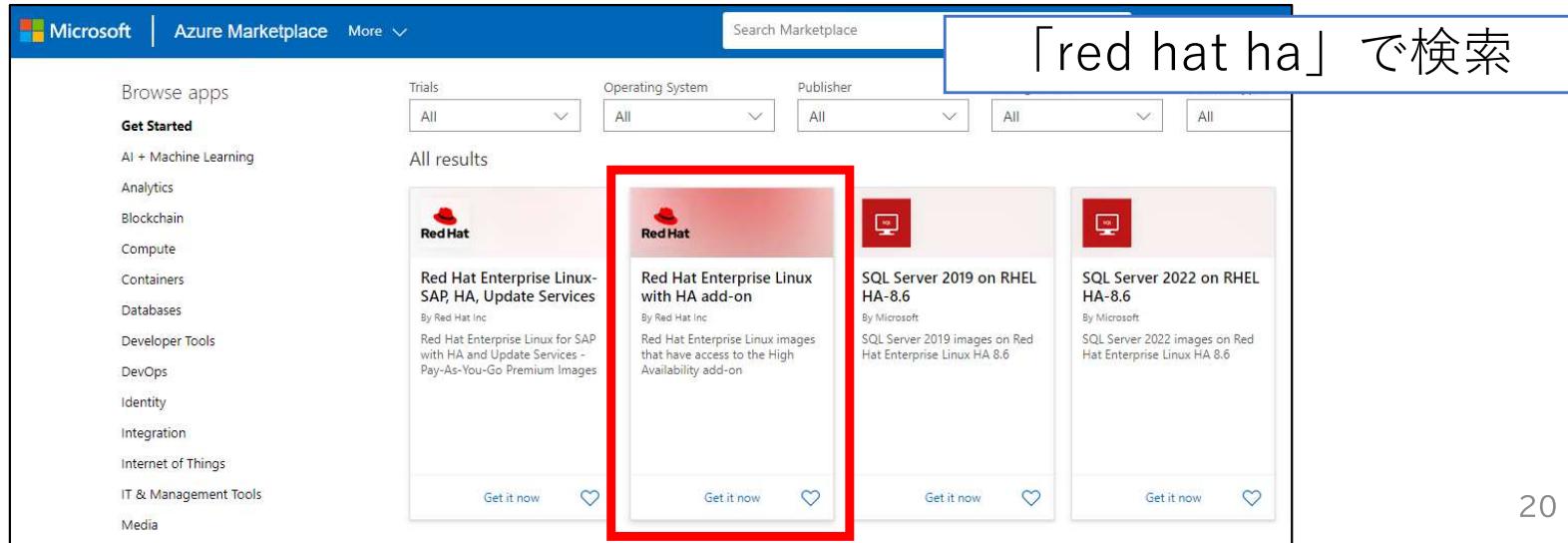
- 以下の流れで説明します
 - 仮想マシン
 - カスタムロール
 - 内部ロードバランサー
 - SNAT用のパブリックロードバランサー
 - Pacemakerの設定
 - クラスタリソースの設定

仮想マシン作成

- ・試験用の仮想マシンを作成します。
- ・基本的な設定はチュートリアルに沿って作成します。
 - ・ クイックスタート: Azure portal で Linux 仮想マシンを作成する
 - ・ <https://learn.microsoft.com/ja-jp/azure/virtual-machines/linux/quick-create-portal?tabs=ubuntu>
- ・ OSはRed Hat Enterprise Linux 9.2を使います。
- ・ 特定の設定が必要なものを次ページ以降に記載します。

仮想マシン作成 イメージの選び方

- ・仮想マシンには従量課金イメージを使用します。
- ・High Availability Add-Onを使う場合、
Azure Marketplaceで「HA add-on」が含まれるもの
を選択します。
 - ・Azure Marketplace
 - ・<https://azuremarketplace.microsoft.com/en-us/home>



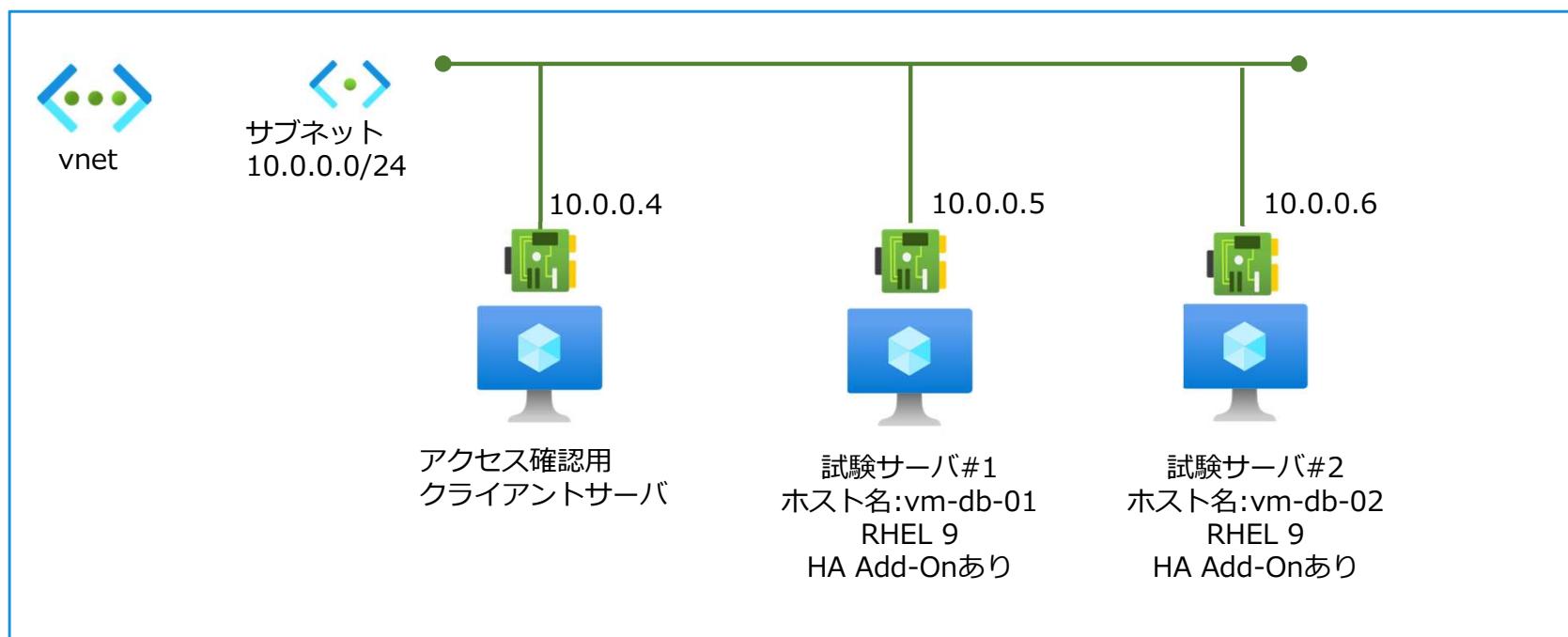
仮想マシン作成 マネージドIDの設定

- ・仮想マシンのシステム割り当てマネージドIDを有効化します。
- ・この設定はフェンスエージェント fence_azure_arm で利用します。



ここまで構成

- 以下のような構成を準備します。
- アクセス確認用クライアントサーバは、Pacemakerによって付与する仮想IPへアクセス確認するために用意します。そのためイメージに制約はありません。



Pacemakerをインストール

Pacemakerに使用するパッケージをインストールします。

インストール ■両サーバで

```
$ sudo dnf install -y pcs pacemaker fence-agents-azure-arm  
nmap-ncat resource-agents-cloud
```

インストールされたフェンスエージェント fence_azure_armの確認 ■両サーバで

```
$ which fence_azure_arm  
/usr/sbin/fence_azure_arm
```

fence用ロールの作成

fence_azure_armがまだ使えない

- この時点では、試験サーバからfence_azure_armコマンドを実施しても、認証が通らずに失敗します。

```
[vm-db-01]$ fence_azure_arm --msi --action=list  
※--msi は認証にマネージドIDを使用するフラグ  
--action=list は操作対象のVMの一覧を表示するアクション
```

```
2024-02-13 08:25:22.229 ERROR: Failed: (AuthorizationFailed)  
The client '2ea...f2' with object id '2ea...f2' does  
not have authorization to perform action  
'Microsoft.Compute/virtualMachines/read' over scope  
'/subscriptions/4a...d3/resourceGroups/rg-ha-test-  
001/providers/Microsoft.Compute' ... 省略 ...
```

カスタムロールを作成する

- これは標準ではAzure リソースへアクセスするための許可権限が、マネージド ID にないためです。
- そこで以下の権限を設定した、fence_azure_arm用のカスタムロールを作成します。

No.	Action	説明
1	Microsoft.Compute/*/read	各種リソースの読み取りを許可する
2	Microsoft.Compute/virtualMachines/powerOff/action	仮想マシンを電源オフする。 仮想マシンの料金は引き続き課金される(ホストから割り当て解除にならない)。
3	Microsoft.Compute/virtualMachines/start/action	仮想マシンを起動する。

Azure リソース プロバイダーの操作 -> Microsoft.Compute

<https://learn.microsoft.com/ja-jp/azure/role-based-access-control/resource-provider-operations#microsoftcompute>

jsonファイルの準備

- 以下の形のjsonファイルを準備します。
- xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の部分は自分のAzureサブスクリプションのIDに書き換えます。

```
{  
    "properties": {  
        "roleName": "Linux Fence Agent Role test",  
        "description": "Allows to power-off and start virtual machines with Pacemaker",  
        "assignableScopes": [  
            "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
        ],  
        "permissions": [  
            {  
                "actions": [  
                    "Microsoft.Compute/*/read",  
                    "Microsoft.Compute/virtualMachines/powerOff/action",  
                    "Microsoft.Compute/virtualMachines/start/action"  
                ],  
                "notActions": [],  
                "dataActions": [],  
                "notDataActions": []  
            }  
        ]  
    }  
}
```

サブスクリプションにカスタムロール追加

- サブスクリプションの管理画面でアクセス制御(IAM)を選択し、「カスタムロールの追加」を選択します。

The screenshot shows the Azure Subscription Management interface. On the left, the navigation pane includes 'サブスクリプション' (Subscription), '追加' (Add), 'ポリシーの管理' (Policy management), and a search bar. Below these are filters for '自分の役割 == すべて' (All roles), '状態 == すべて' (All states), and a 'フィルターの追加' (Add filter) button. The 'サブスクリプション名' (Subscription name) dropdown is set to 'Azure subscription 1'. On the right, the main content area is titled 'Azure subscription 1 | アクセス制御 (IAM)' (Access Control (IAM)). It features a sidebar with '検索' (Search), '概要' (Overview), 'アクティビティ ログ' (Activity log), 'アクセス制御 (IAM)' (Access Control), 'タグ' (Tags), '問題の診断と解決' (Diagnose and solve problems), 'セキュリティ' (Security), 'イベント' (Events), 'コスト管理' (Cost management), 'コスト分析' (Cost analysis), and 'コストのアラート' (Cost alerts). The 'アクセス制御 (IAM)' item is highlighted with a red box. A modal window titled 'マイ アクセス' (My Access) is open, with its '+ 追加' (Add) button also highlighted with a red box. The modal content includes sections for 'ロールの割り当ての追加' (Add role assignment), '共同管理者の追加' (Add shared administrator), and 'カスタム ロールの追加' (Add custom role). A red arrow points from the '+ 追加' button in the main interface to the 'カスタム ロールの追加' section in the modal. The modal also contains a 'マイ アクセスの表示' (Display my access) button.

用意したjsonファイルの読み込み

- 作成画面に変わるので、用意したjsonファイルをアップロードして読み込みます。
- 正常に読み込まれれば、通知と共に入力欄にjsonの値が入力されます。これによりカスタムロールを作成します。

ホーム > サブスクリプション > Azure subscription 1 | アクセス制御 (IAM) >

カスタム ロールを作成する ...

基本 アクセス許可 割り当て可能なスコープ JSON 確認と作成

Azure リソースのカスタム ロールを作成するには、基本情報をいくつか入力します。詳細情報を見る

カスタム ロール名 * ① Linux Fence Agent Role test

説明 Allows to power-off and start virtual machines with Pacemaker

ベースラインのアクセス許可 ① ロールを複製します 最初から始める JSON から開始

ファイル * "Linux-fenceagent-custom-role.json"

通知

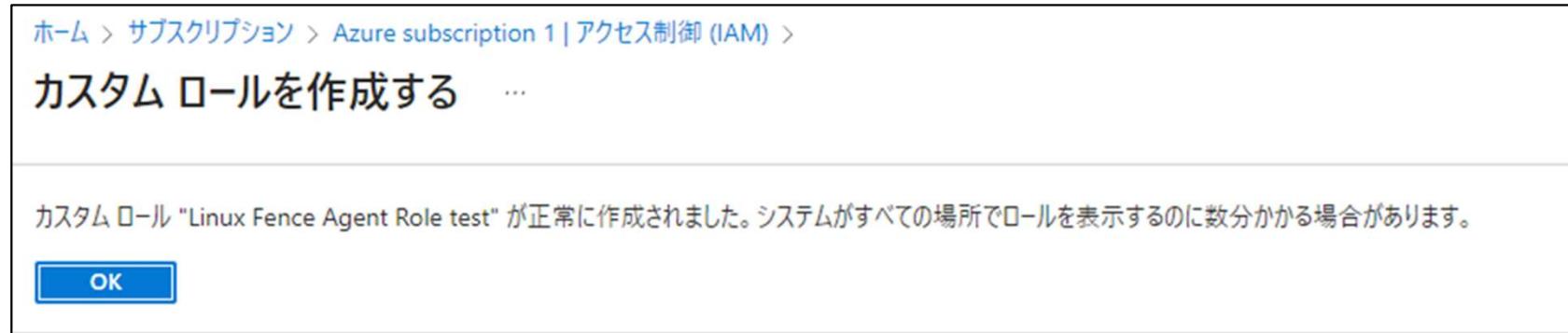
アクティビティ ログのその他のイベント → すべて無視 ▾

ファイルの検証 有効な JSON コンテンツ 数秒前

Linux-fenceagent-custom-role.json のアップロードの完了 682 B | "ストリーミング アップロード" 数秒前

カスタムロール作成の確認

- 正常に作成されると通知が表示されます。
- 通知の通り、各リソースで実際に設定可能になるのは数分の待機が必要です。



ロールをVMに割り当て 1

- VMのアクセス制御(IAM)の役割タブの画面で、作成したカスタムロールが表示されることを確認します。

The screenshot shows the Azure portal interface for managing IAM roles on a virtual machine named 'vm-db-01'. The left sidebar lists other virtual machines: 'vm-db-01' (selected), 'vm-db-02', 'vm-db-03', and 'vm-db-04'. The main content area is titled 'vm-db-01 | アクセス制御 (IAM)' and displays the '役割' (Roles) tab. A search bar at the top right is highlighted with a red box and contains the text 'カスタム ロール名で検索' (Search by custom role name). Below the search bar, a table lists roles, with one row for 'Fence' highlighted by a red box. The 'Fence' role is described as 'Allows to power-off and start virtual machines with Pacemaker'. Navigation tabs include 'アクセスの確認' (Access review), 'ロールの割り当て' (Assign role), '役割' (Roles), '拒否の割り当て' (Deny assignment), and '従来の管理者' (Previous administrators).

ロールをVMに割り当て 2

- VMで、「ロールの割り当てを追加」を実施します。

The screenshot shows the Azure portal interface for managing IAM roles on a virtual machine named 'vm-db-01'. The left sidebar has 'vm-db-01 | アクセス制御 (IAM)' selected. The main content area has a heading 'ロールの割り当ての追加' (Add role assignment). Below it, tabs include '役割' (Role), which is selected, and '拒否の割り当て' (Deny assignment), '従来の管理者' (Previous administrator), and '共同管理者の追加' (Add co-administrator). A search bar at the bottom contains the word 'Fence'. The URL in the address bar is https://portal.azure.com/#blade/Microsoft_Azure_IAM/RoleAssignmentsBlade/resourceId%3D/VM%2Fvm-db-01%2FosDisk%2FmountPoint%2F%2F&roleDefinitionId%3D%2F%2F.

ロールをVMに割り当て 3

- 追加画面に移動するので、次のように設定します。
 - ロールタブでは作成したロールを選択します。
 - メンバータブではマネージドIDを使用し、クラスタに組み込むVMを選択します。

The screenshot shows two Azure portal dialogs and one modal window.

Left Dialog: ロールの割り当ての追加

- 選択されたロール: Linux Fence Agent Role test
- アクセスの割り当て先: マネージド ID (selected)
- メンバー: + メンバーを選択する
- 検索欄: fence
- 検索結果: 1 - 1 / 1 件 (vm-db-01, vm-db-02, vm1-bastion)

Right Dialog: マネージド ID の選択

- サブスクリプション: Azure サブスクリプション 1
- マネージド ID: 仮想マシン (5)
- 選択: 名前で検索
- リスト: vm-db-03, vm-db-04, vm1-bastion
- 選択したメンバー (赤枠): vm-db-01, vm-db-02

Bottom Buttons: レビューと割り当て, 前へ, 次へ, 選択, 閉じる, フィードバック

ロールをVMに割り当て 4

- VMのロールの割り当てタブで、指定したVMにロールが当たっていればfence_azure_armのリクエストが成功します。

vm-db-01 | アクセス制御 (IAM) ☆ ...

検索

+ 追加 ▾ ロール割り当てのダウンロード ⚡ 列の編集 ⚡ 最新の情報に更新 ✎ 削除 🔍 フィード

概要 アクティビティログ アクセス制御 (IAM) タグ 問題の診断と解決

接続 接続 Bastion

ネットワーク ネットワーク設定 負荷分散 アプリケーションのセキュリティグループ ネットワークマネージャー

設定 ディスク

アクセスの確認 ロールの割り当て 役割 拒否の割り当て 従来の管理者

このサブスクリプションにおけるロール割り当ての数 10 4000 特権あり

割り当てを表示する

すべて 職務権限 (2) 特権あり

仮想マシンを作成して管理する権限など、職務に基づいた Azure リソースへのアクセス権を付与するロールの割り当て

名前または電子メールで検索する 種類: すべて 役割: すべて

2 個のアイテム

名前	種類
Linux Fence Agent Role test (2)	
<input type="checkbox"/> vm-db-01 /subscriptions/[REDACTED] 仮想マシン	
<input type="checkbox"/> vm-db-02 /subscriptions/[REDACTED] 仮想マシン	

◇設定後のコマンド結果

```
[vm-db-01 ~]$ fence_azure_arm --msi --action=list
vm-db-01,
```

ロールをVMに割り当て 5

- ・同じ手順をもう一方のVMに対して実施します。
- ・ロールの割り当てが成功し、以下のようにクラスタを組む対象のVMがすべて表示されれば成功です。

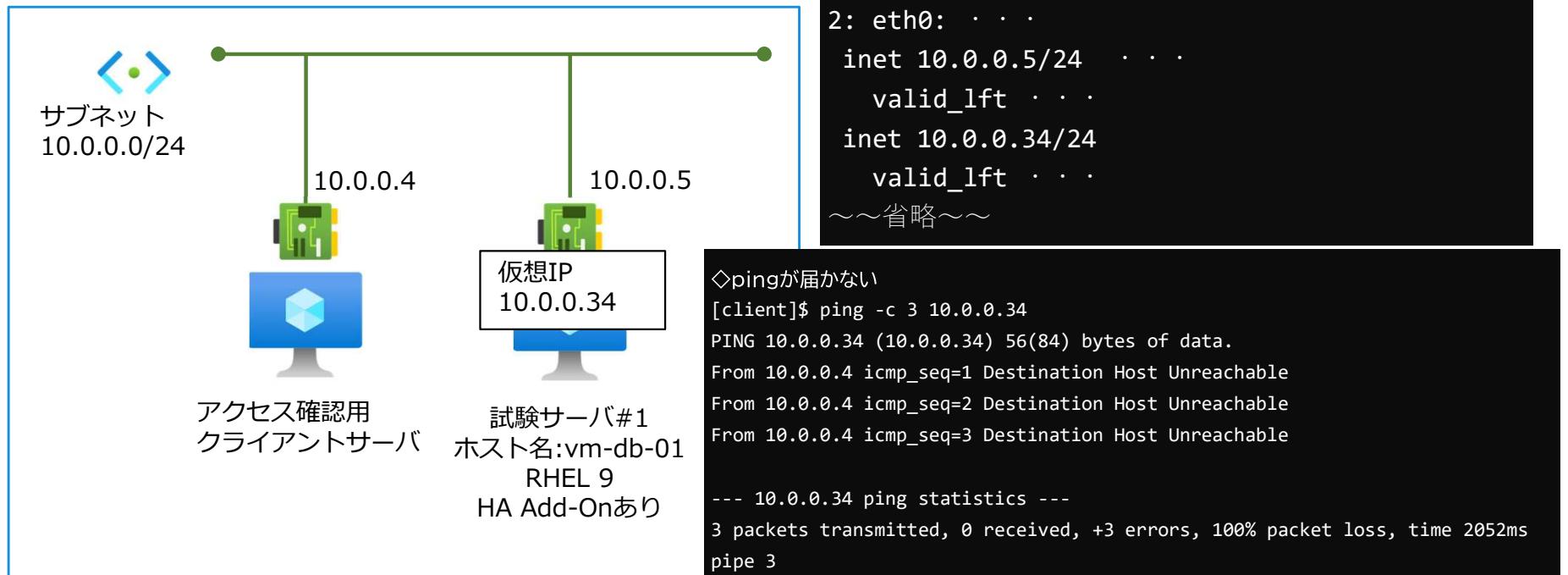
◇すべてのVMにカスタムロールの割り当て設定後のコマンド例

```
[vm-db-01 ~]$ fence_azure_arm --msi --action=list  
vm-db-01,  
vm-db-02,
```

内部ロードバランサーの作成

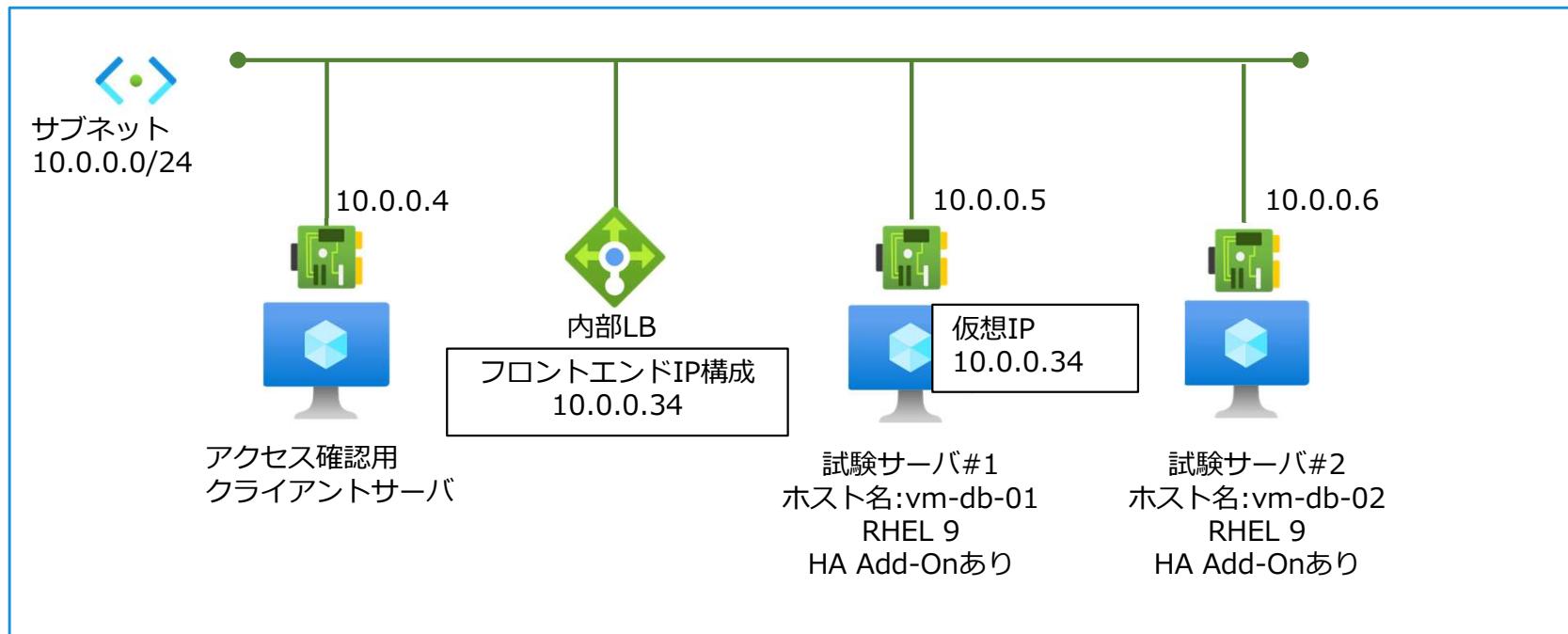
仮想IPに届かない？

- 以下のように仮想マシンで仮想IPアドレスを(セカンダリアドレスとして)設定したとします。
- しかしAzureでは、NICリソースにアドレスが割り振られていないこの状態では、クライアントサーバからのアクセスが仮想IPに届きません。



内部ロードバランサーを用意する

- 以下のように内部ロードバランサーを追加します。
- これにより仮想IPアドレス宛てのアクセスが、ロードバランサーの設定したルールにより試験サーバへ割り振られることで、アクセスできるようになります。



内部ロードバランサーの作成

- ・内部ロードバランサーを作成します。
- ・基本的な設定はチュートリアルに沿って作成します。
 - ・クイック スタート: Azure portal を使用して VM の負荷を分散する内部ロード バランサーを作成する
 - ・<https://learn.microsoft.com/ja-jp/azure/load-balancer/quickstart-load-balancer-standard-internal-portal#create-load-balancer>
- ・特定の設定が必要なものを次ページ以降に記載します。

内部LB フロントエンド構成

- ・どのアドレスへのアクセスを振り分けるか設定する「フロントエンドIP構成」です。
- ・ここではPacemakerによって仮想IPアドレスとして設定予定のアドレスを記載してます。

フロントエンド IP 構成の追加 ×

名前 *

 ✓

仮想ネットワーク
vm1-bastion-vnet

サブネット *

 ▾

割り当て

動的 静的

IP アドレス *

 ✓

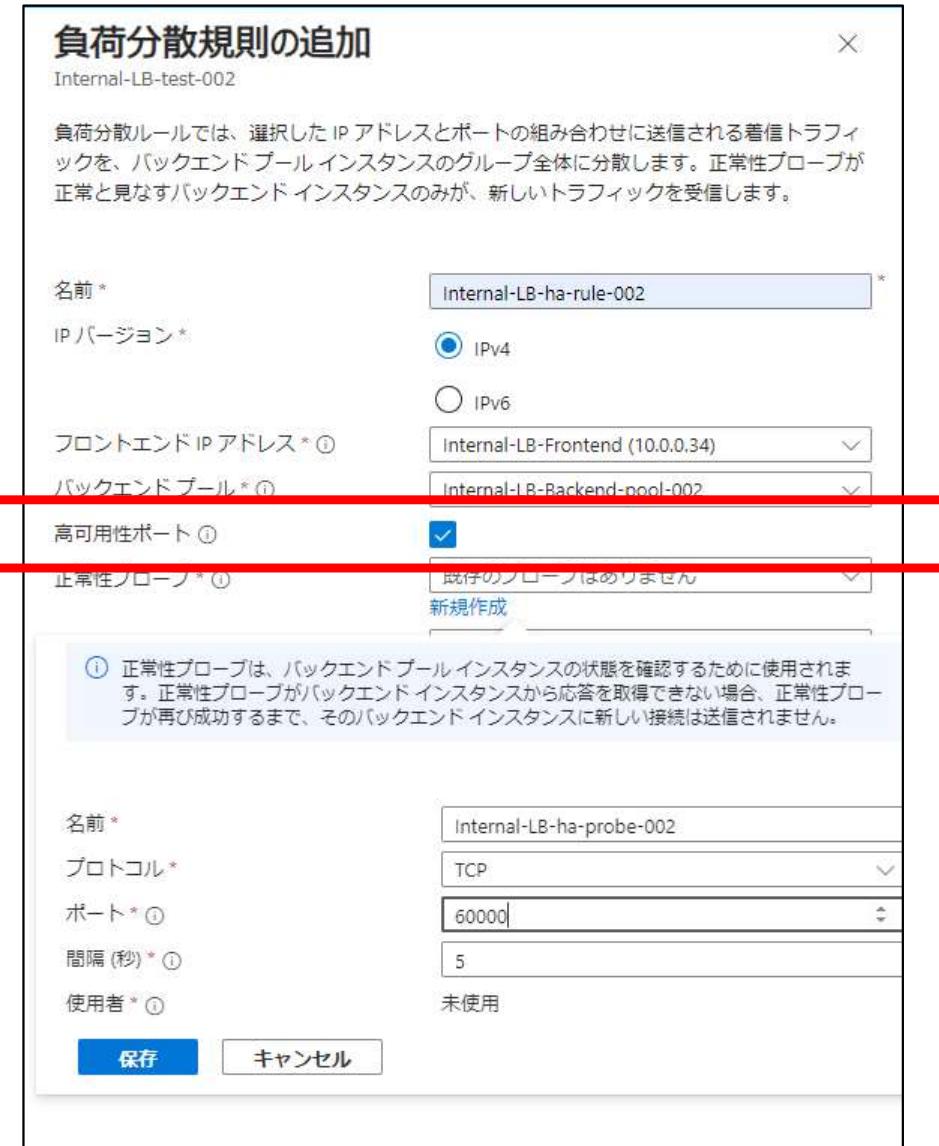
可用性ゾーン * ⓘ

 ▾

追加

内部LB 負荷分散規則 1

- ・アクセスの振り分けルールの設定です。
- ・高可用性ポートのチェックボックスにチェックを入れます。
- ・これにより、TCP プロトコルと UDP プロトコル用のすべてのポートで負荷分散が有効になります。



内部LB 負荷分散規則 2

- 正常性プローブの設定で、port 60000にプローブを送るよう設定します。
- この値はリソースエンジント azure-lb で使用します。

負荷分散規則の追加
Internal-LB-test-002

負荷分散ルールでは、選択した IP アドレスとポートの組み合わせに送信される着信トラフィックを、バックエンド プールインスタンスのグループ全体に分散します。正常性プローブが正常と見なすバックエンド インスタンスのみが、新しいトラフィックを受信します。

名前 * Internal-LB-ha-rule-002 *

IP バージョン * IPv4

フロントエンド IP アドレス * ① Internal-LB-Frontend (10.0.0.34)

バックエンド プール * ① Internal-LB-Backend-pool-002

高可用性ポート ①

正常性プローブ * ① 既存のプローブはありません 新規作成

① 正常性プローブは、バックエンド プールインスタンスの状態を確認するために使用されます。正常性プローブがバックエンド インスタンスから応答を取得できない場合、正常性プローブが再び成功するまで、そのバックエンド インスタンスに新しい接続は送信されません。

名前 * Internal-LB-ha-probe-002

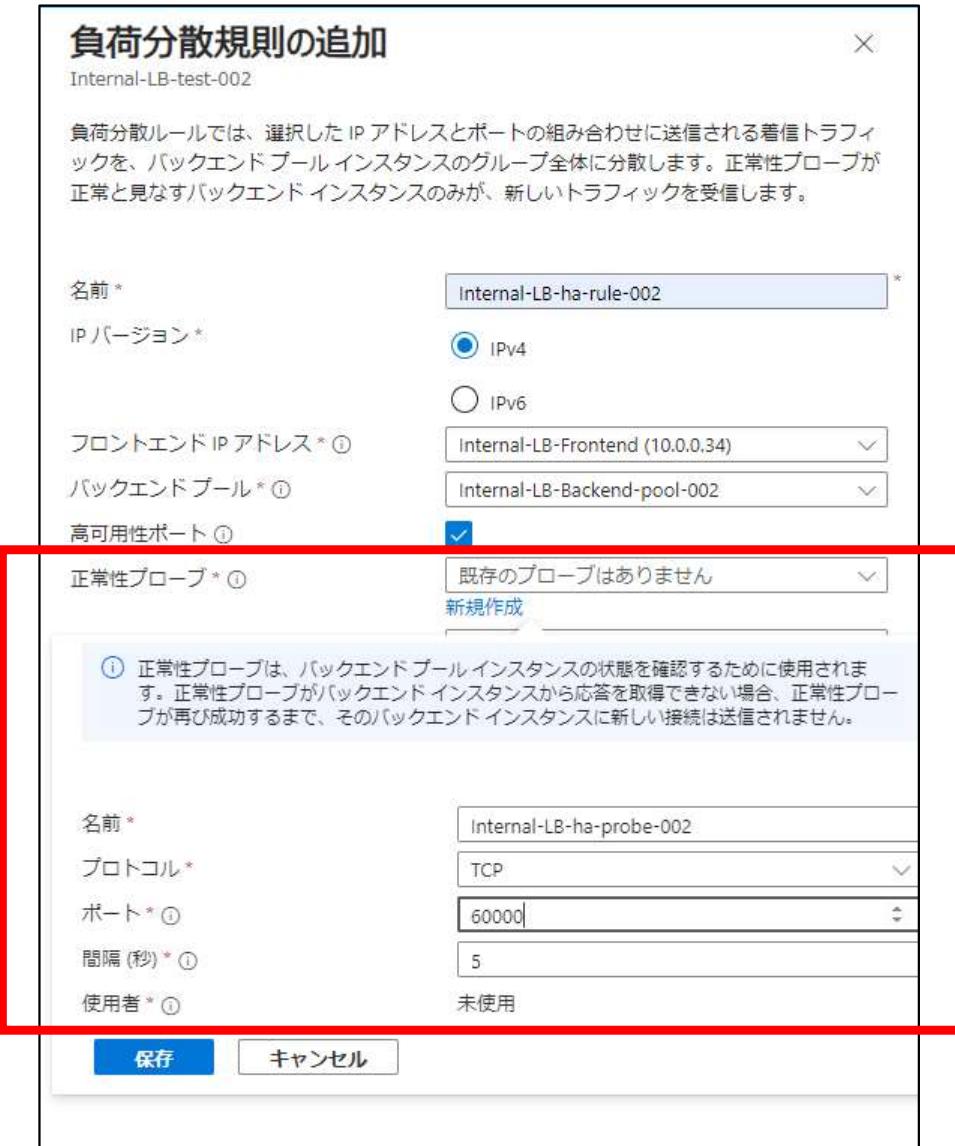
プロトコル * TCP

ポート * ① 60000

間隔 (秒) * ① 5

使用者 * ① 未使用

保存 キャンセル



内部LB 負荷分散規則 3

- ・フローティングIPを有効します。
- ・これで内部LBを作成します。



- ・Azure Load Balancer のフローティング IP の構成
 - ・<https://learn.microsoft.com/ja-jp/azure/load-balancer/load-balancer-floating-ip>

firewalldの設定

- VM側のfirewalldが起動している場合、LBからの通信を許可する必要があります。LBのIPアドレスを許可する設定を行います。

LBからの通信をすべて許可 ■両サーバで

```
$ firewall-cmd --get-active-zones
public
  interfaces: eth0

$ sudo firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source address
="168.63.129.16/32" accept"

$ sudo firewall-cmd --reload
```

- IP アドレス 168.63.129.16 とは

- <https://learn.microsoft.com/ja-jp/azure/virtual-network/what-is-ip-address-168-63-129-16>
- VM が「ロード バランサー バックエンド」プールの一部である場合、正常性プローブ通信を、168.63.129.16 から発信できるようにする必要があります。既定のネットワーク セキュリティ グループの構成には、この通信を許可する規則があります。

パブリックロードバランサー の作成

fence_azure_armができる?

- ここまで設定を行うと、fence_azure_armの通信が失敗するようになります。

◇内部LBを設定後、通信ができなくなる例。verboseオプションを付けている。

```
[vm-db-01]# fence_azure_arm --verbose --msi --action=list
```

~~省略~~

```
2024-02-19 05:55:40,782 INFO: Request URL:  
'https://management.azure.com/subscriptions/xxxxxxxxx-xxxx-xxxx-xxxx-  
xxxxxxxxxxxx/resourceGroups/rg-ha-test-  
001/providers/Microsoft.Compute/virtualMachines?api-version=REDACTED'  
2024-02-19 05:55:40,782 INFO: Request method: 'GET'
```

~~省略~~

```
2024-02-19 05:55:40,784 DEBUG: Starting new HTTPS connection (1):  
management.azure.com:443
```

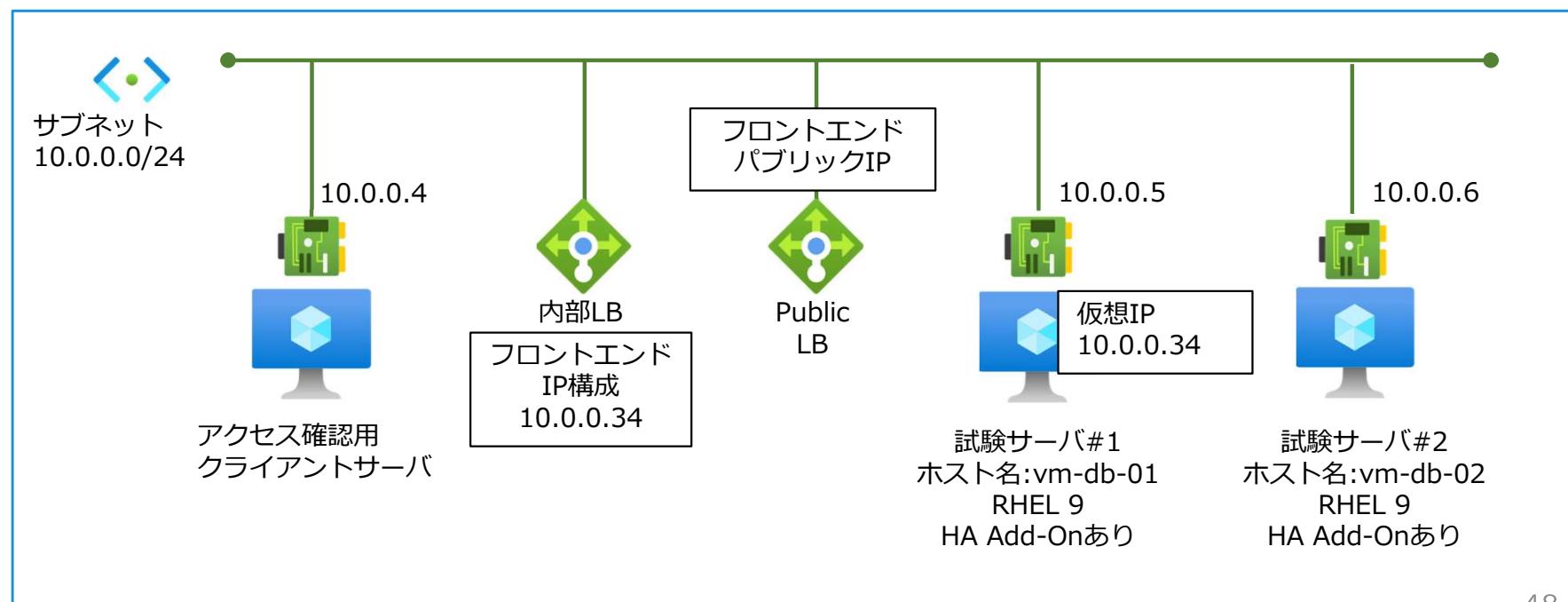
※このログを最後に、タイムアウトまで何も出力されない。

外部通信ができない？

- これは「VMが内部LBのバックエンドプールに追加されると、外部と通信できなくなる」仕様のためです。
- ログの通り、management.azure.com へアクセスする必要がありますがこの通信が失敗します。
- Azure VM の送信接続 (SNAT) オプションまとめ
 - <https://jpaztech.github.io/blog/network/snata-options-for-azure-vm/>
- Azure での既定の送信アクセス
 - <https://learn.microsoft.com/ja-jp/azure/virtual-network/ip-services/default-outbound-access>
- 送信接続での送信元ネットワーク アドレス変換 (SNAT) を使用する
 - <https://learn.microsoft.com/ja-jp/azure/load-balancer/load-balancer-outbound-connections>

パブリックロードバランサーを用意する

- 対策としていろいろな方法がありますが、ここではパブリックロードバランサー(PublicLB)を追加します。
- PublicLBのフロントエンド IP アドレスから外部通信が行われます。



PublicLBの作成

- PublicLBを作成します。
- 基本的な設定はチュートリアルに沿って作成します。
 - クイック スタート: Azure portal を使用して、VM の負荷分散を行うパブリック ロード バランサーを作成する
 - <https://learn.microsoft.com/ja-jp/azure/load-balancer/quickstart-load-balancer-standard-public-portal#create-load-balancer>
- 特定の設定が必要なものを次ページ以降に記載します。

PublicLB フロントエンド構成

- ・今回の利用方法では外部通信の出口としてのみ利用します。
- ・そのためパブリックIPアドレスの設定のみ行えば問題ありません。



PublicLB インバウンド規則は設定しない

- インバウンド規則は何も設定しません。

The screenshot shows the 'Inbound Rules' tab selected in a navigation bar. Below it, two sections are displayed: 'Load Balancing Rule' and 'Inbound NAT Rule'. Both sections have a 'Create Rule' button and a table with columns for Name, Frontend IP Configuration, and Backend Pool.

名前 ↑↓	フロントエンド IP 構成 ↑↓	バックエンド プール ↑↓
開始するには規則を追加してください		

名前 ↑↓	フロントエンド IP 構成 ↑↓	サービス ↑↓
開始するには規則を追加してください		

PublicLB アウトバウンド規則

- アウトバウンド規則では前のタブで作成予定のアドレスなどを選択して組み合わせ、LBをデプロイします。

アウトバウンド規則の追加

名前 *
PublicLB-Outbound-rule-001

IP バージョン *
 IPv4
 IPv6

フロントエンド IP アドレス *
1 個が選択済み

プロトコル
 All
 TCP
 UDP

アイドル タイムアウト (分) *
30 最大: 100

TCP リセット *
 有効
 無効

バックエンド プール *
PublicLB-Backend-Pool-002 (2 個のインスタンス)

ポートの割り当て

Azure では、フロントエンド IP アドレスとバックエンド プール インスタンスの数に基づいて、送信元ネットワーク アドレス変換 (SNAT) に使用される送信ポートの数が自動的に割り当てられます。送信接続の詳細 ▾

ポートの割り当て ⓘ
送信ポートの数を手動で選択する

送信ポート
選択基準 *

インスタンスごとのポート

インスタンスごとのポート * ⓘ
10000

使用可能なフロントエンド ポート
64000

バックエンド インスタンスの最大数 ⓘ
6

追加

PublicLB 設置後

- PublicLBのデプロイ後、VMから外部通信が可能になります。
- VMからfence_azure_armを実施して、VMの一覧が出ることを確認します。

◇PublicLBのデプロイ後

```
[vm-db-01 ~]$ fence_azure_arm --msi --action=list
vm-db-01,
vm-db-02,
```

Pacemakerの構築

Pacemakerを構築 1

クラスタが利用するポートの許可 □両サーバで

```
$ sudo firewall-cmd --permanent --add-service=high-availability  
$ sudo firewall-cmd --add-service=high-availability
```

pcsdサービスの起動 □両サーバで

```
$ sudo systemctl enable pcسد.service --now
```

haclusterユーザのパスワード設定 ■両サーバで

```
$ sudo passwd hacluster  
Changing password for user hacluster.  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.
```

Pacemakerを構築 2

クラスタノードの認証設定 ■いずれか片方のノードで。例では#1で実施

```
[vm-db-01 ]$ sudo pcs host auth vm-db-01 addr=10.0.0.5 vm-db-02 addr=10.0.0.6  
Username: hacluster  
Password: (先ほど設定したパスワードを入力)  
vm-db-01 : Authorized  
vm-db-02 : Authorized
```

クラスタの作成 ■上の認証設定を行ったノードで

```
[vm-db-01 ]$ sudo pcs cluster setup DB_Cluster vm-db-01 addr=10.0.0.5 vm-db-02 addr=10.0.0.6
```

Pacemakerの起動 ■上の認証設定を行ったノードで

```
[vm-db-01 ]$ sudo pcs cluster start --all
```

リソース設定の流れ

リソース設定をpcsだけで行う場合

- Red Hatのお作法に従いpcsで構築する際には、これらのpcsコマンドを一から作成しなければなりません。

```
# pcs resource defaults resource-stickiness=200
# pcs resource defaults migration-threshold=1
# pcs resource create filesystem1 ocf:heartbeat:Filesystem device="/dev/mapper/mpatha2" directory="/dbfp/pgdata" fstype="xfs"
force_unmount="safe" op start timeout=60s on-fail=restart monitor timeout=60s interval=10s on-fail=restart stop timeout=60s on-fail=fence
# pcs resource create filesystem2 ocf:heartbeat:Filesystem device="/dev/mapper/mpatha3" directory="/dbfp/pgwal" fstype="xfs"
force_unmount="safe" op start timeout=60s on-fail=restart monitor timeout=60s interval=10s on-fail=restart stop timeout=60s on-fail=fence
# pcs resource create filesystem3 ocf:heartbeat:Filesystem device="/dev/mapper/mpatha4" directory="/dbfp/pgarch" fstype="xfs"
force_unmount="safe" op start timeout=60s on-fail=restart monitor timeout=60s interval=10s on-fail=restart stop timeout=60s on-fail=fence
# pcs resource create ipaddr ocf:heartbeat:IPaddr2 ip="192.168.1.87" nic="ens4" cidr_netmask="24" op start timeout=60s on-fail=restart monitor
timeout=60s interval=10s on-fail=restart stop timeout=60s on-fail=fence
# pcs resource create pgsql ocf:linuxhajp:pgsql pgctl="/usr/pgsql-11/bin/pg_ctl" psql="/usr/pgsql-11/bin/psql" pgdata="/dbfp/pgdata/data"
pgdba="postgres" pgport="5432" pgdb="template1" op start timeout=300s on-fail=restart monitor timeout=60s interval=10s on-fail=restart stop
timeout=300s on-fail=fence
# pcs resource group add pgsql-group filesystem1 filesystem2 filesystem3 ipaddr pgsql
# pcs resource create ping ocf:pacemaker:ping name="ping-status" host_list="192.168.1.1" attempts="2" timeout="2" debug="true" op start
timeout=60s on-fail=restart monitor timeout=60s interval=10s on-fail=restart stop timeout=60s on-fail=fence
# pcs resource clone ping
# pcs stonith create fence1-ipmilan fence_ipmilan delay="10" pcmk_host_list="r81-1" ip="192.168.2.85" username="root" password="XXXXXX"
lanplus="1" ipmitool_path="/usr/bin/ipmitool_stub" op start timeout=60s on-fail=restart monitor timeout=60s interval=3600s on-fail=restart stop
timeout=60s on-fail=ignore
# pcs stonith create fence2-ipmilan fence_ipmilan delay="0" pcmk_host_list="r81-2" ip="192.168.2.86" username="root" password="XXXXXX"
lanplus="1" ipmitool_path="/usr/bin/ipmitool_stub" op start timeout=60s on-fail=restart monitor timeout=60s interval=3600s on-fail=restart stop
timeout=60s on-fail=ignore
# pcs constraint location pgsql-group prefers r81-1=200
# pcs constraint location pgsql-group prefers r81-2=100
# pcs constraint location fence1-ipmilan avoids r81-1
# pcs constraint location fence2-ipmilan avoids r81-2
# pcs constraint location pgsql-group rule score=-INFINITY ping-status lt 1 or not_defined ping-status
# pcs constraint colocation add pgsql-group with ping-clone score=INFINITY
# pcs constraint order ping-clone then pgsql-group symmetrical=false
```

ツール pm_pcsgen 1

- Linux-HA Japan では、Excel形式の環境定義書からpcsコマンドを生成する pm_pcsgen というツールを用意しています。
 - Linux-HA Japan の pm_extra_tools リリースノート
 - [https://github.com/linux-ha-japan/pm extra tools/releases](https://github.com/linux-ha-japan/pm_extra_tools/releases)

```
# dnf install pm_extra_tools-1.5-1.el9.noarch.rpm
```

/usr/share/pm_extra_tools/pm_pcsgen_sample.xlsx にサンプルファイルがあります。

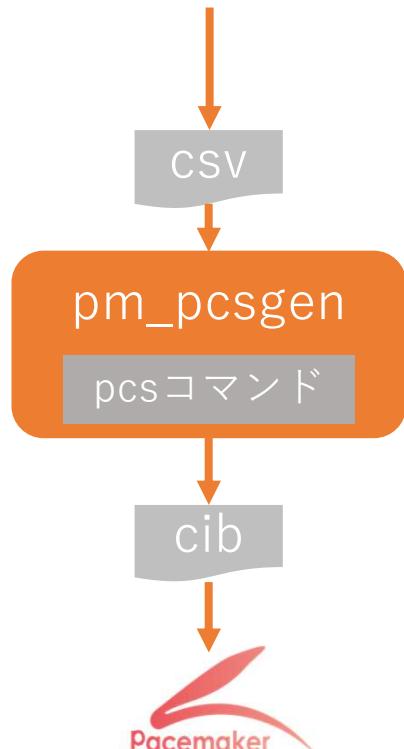
PRIMITIVE					
P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	azure-lb-primary	ocf	heartbeat	azure-lb	Azureロードバランサ応答
A	type	name	value		
#	パラメータ種別	項目	設定内容		概要
	options	port	60000		正常性プローブ要求に応答するポート番号。 内部ロードバランサ作成時に設定した正常性プローブ要求のポート番号と同一のも
O	type	timeout	interval	on-fail	role start-delay
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	
	start	20s		restart	
	monitor	20s	10s	restart	
	stop	20s		fence	

このようなExcelを書いて、リソースの設定ができます。

ツール pm_pcsgen 2

- 作成したExcelファイルをcsvで保存し、pm_pcsgenを利用してすることで、pcsコマンドからクラスター設定ファイル(cib)を作成するところも自動で実施してくれます。Red Hatのお作法に従いながら簡単に構築ができます。

PRIMITIVE				
#	id	class	provider	type
	リソースID	class	provider	type
#	azure-lb-primary	ocf	heartbeat	azure-lb
A	type	name	value	備考
	ハートバート種別	選択	設定内容	備考
	options	port	60010	正常性プローブ要求に応答するポート番号。 内部ハートバントサ作成時に設定した正常性プローブ要求のポート番号と同一のものを持たせます。
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
	start	20s		restart
	monitor	20s	10s	restart
	stop	20s		force



Excelの「名前をつけて保存」で、リソースを定義しているシートのみをcsvとして保存

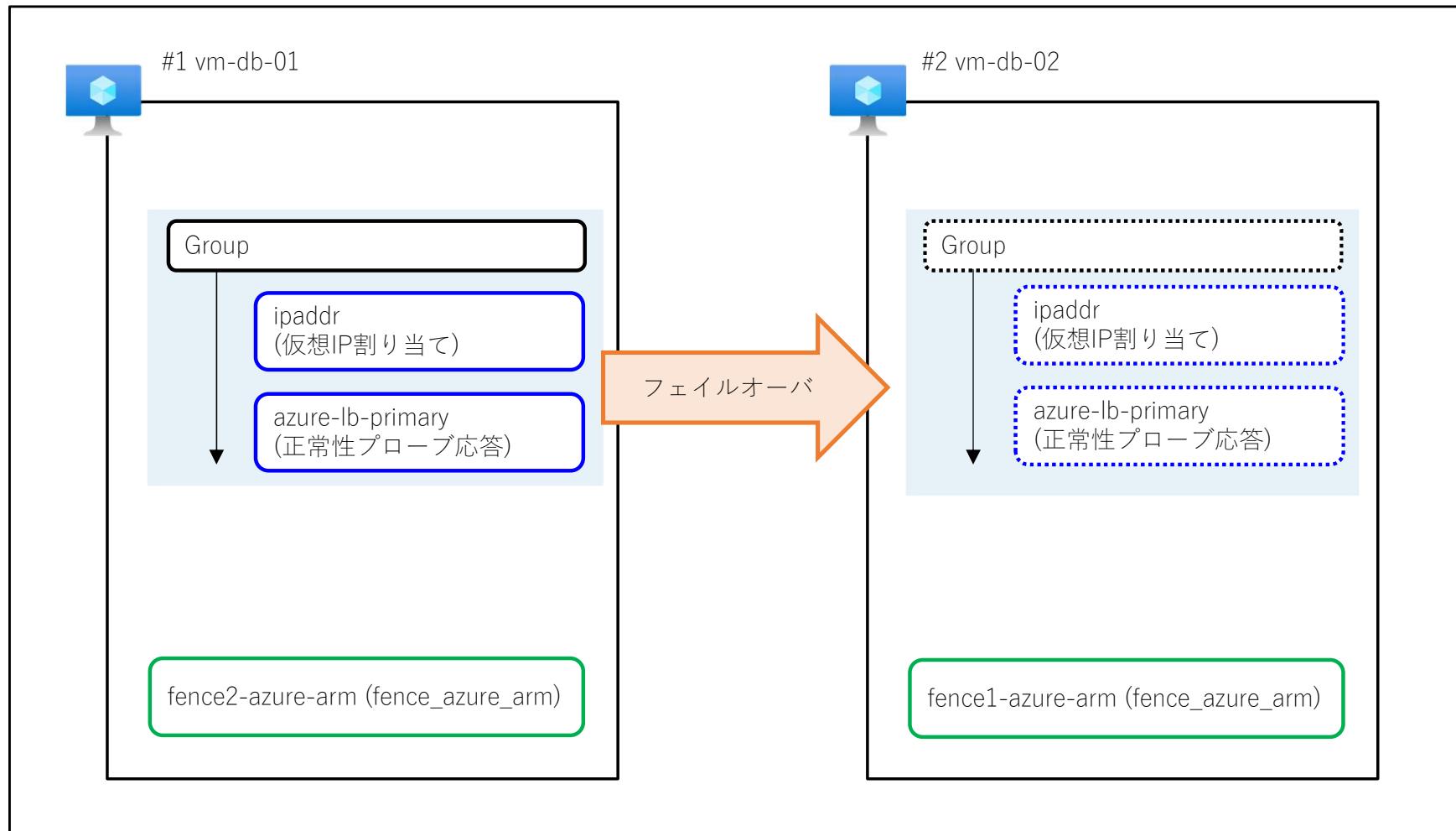
```
$ pm_pcsgen resource-def.csv
resource-def.xml(CIB), resource-def.sh(PCS)を出力しました。
$ ls
resource-def.csv resource-def.xml resource-def.sh
```

(cluster start後に)

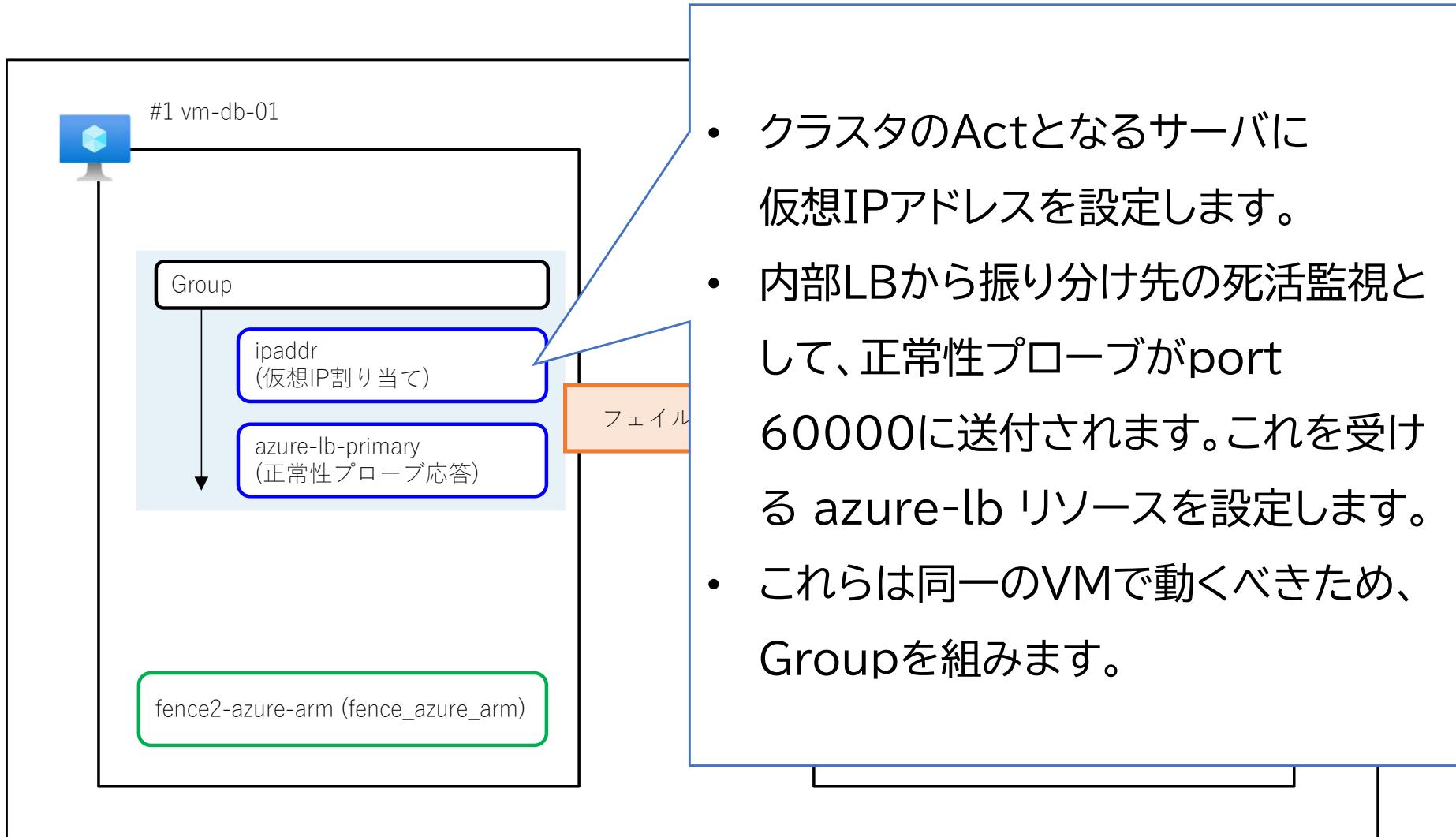
```
$ sudo pcs cluster cib-push resource-def.xml --config
```

試しのリソース設定

- ・今回は以下のようなリソースの構成例です。



仮想IPアドレスとazure-lbリソース



サンプル設定 仮想IPアドレス

PRIMITIVE								
P	id	class	provider	type				
#	リソースID ipaddr	class ocf	provider heartbeat	type IPAddr2	概要 仮想IP割当			
A	type	name	value					
#	パラメータ種別 options	項目 ip nic cidr_netmask	設定内容 10.0.0.34 eth0 24	概要 仮想IPアドレス アドレス付与デバイス名 ネットマスク				
O	type	timeout	interval	on-fail	role	start-delay		
#	オペレーション start monitor stop	タイムアウト値 60s 60s 60s	監視間隔 10s	障害時の動作 restart restart fence	役割	起動前待機時間		
						備考		

- リソースID

- このリソースIDはクラスタ内で一意にします。
- 同じリソースエージェントを使用して複数リソースを稼働できるため、このIDにより識別します。

- class:provider:type

- 使用するリソースエージェントを指定します。以下のページで定義されています。
 - <https://www.clusterlabs.org/pacemaker/doc/2.1/Pacemaker Explained/singlehtml/#resource-properties>
- ocf:heartbeat:IPAddr2 ならば、/usr/lib/ocf/resource.d/heartbeat/IPAddr2 にあります。

サンプル設定 azure-lb

PRIMITIVE								
P	id	class	provider	type	概要			
#	リソースID azure-lb-primary	class ocf	provider heartbeat	type azure-lb	Azureロードバランサ応答			
A	type	name	value					
#	パラメータ種別 options	項目 port	設定内容 60000	概要 正常性プローブ要求に応答するポート番号。 内部ロードバランサ作成時に設定した正常性プローブ要求のポート番号と同一のも				
O	type	timeout	interval	on-fail	role	start-delay		
#	オペレーション start	タイムアウト値 20s	監視間隔	障害時の動作 restart	役割	起動前待機時間		
	monitor	20s	10s	restart				
	stop	20s		fence				

- パラメータ options の port

- 内部LBで正常性プローブを送付するポートとして指定した値を、ここで設定します。
- 指定したポートで nc(netcat) が起動し、正常性プローブを待ち受けるようになります。

サンプル設定 Group

#表 4-1 クラスタ設定 ... リソース構成

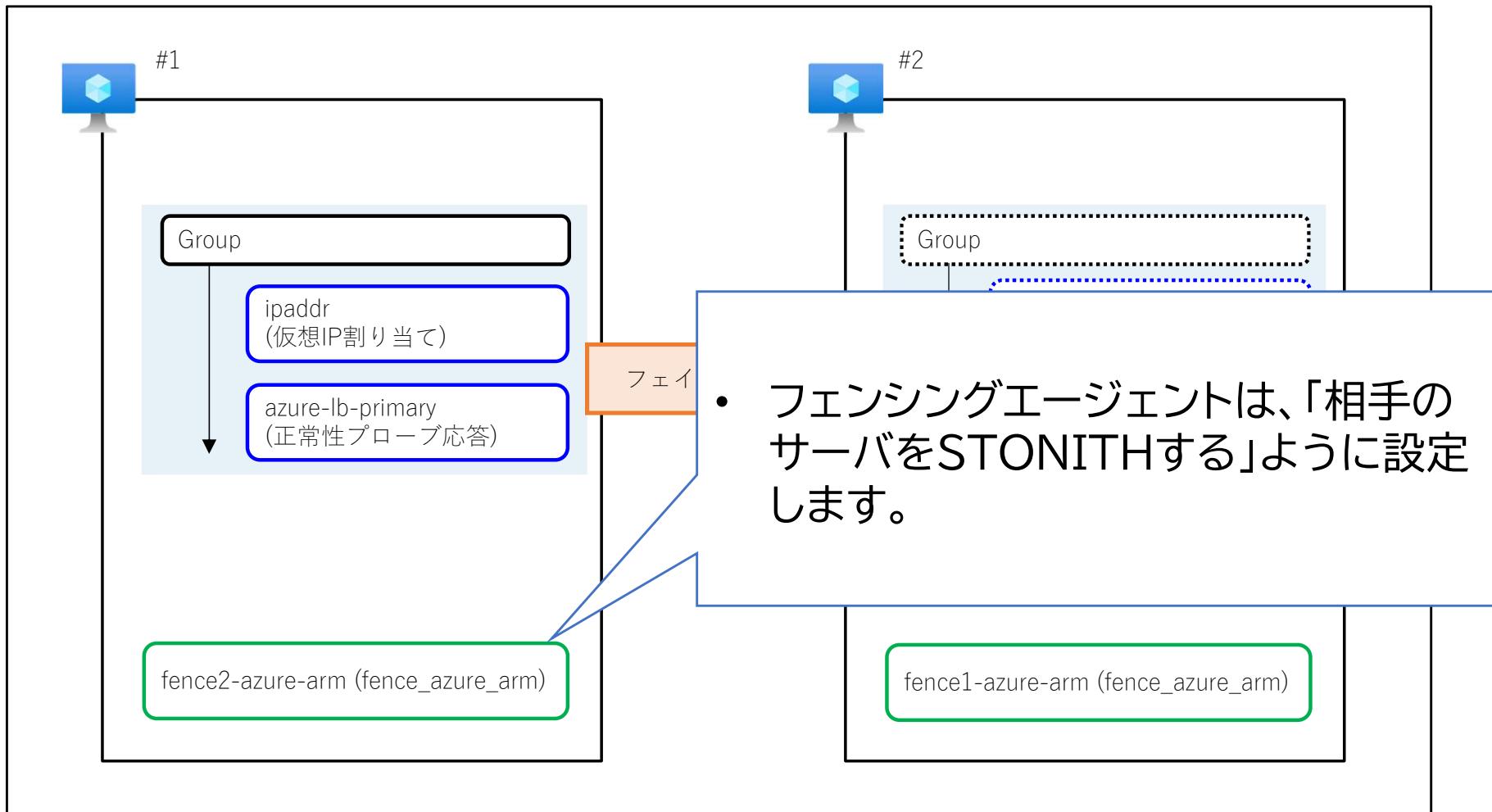
RESOURCES			
	resourceItem	resourceItem	id
#	リソース構成要素	リソースID	概要
	Group	primary-group	グループ定義
	Primitive	ipaddr	仮想IP割当
	Primitive	azure-lb-primary	Azureロードバランサ応答

RSC_ATTRIBUTES			
	id	name	value
#	リソースID	項目	設定内容
	primary-group	priority	1

LOCATION_NODE				
	rsc	prefers/avoids	node	score
#	リソースID	prefers/avoids	ノード	スコア
	primary-group	prefers	vm-db-01	200
			vm-db-02	100

- ・ 指定のリソースでGroupを組みます。
- ・ GroupのリソースIDを使用し、Actノードで稼働しているべきリソースにpriority値を設定します。「『priority値の設定されているリソースが稼働しているノード』がAct」と判断されます。
- ・ 優先的に起動してほしいノードのスコアを高くします。
- ・ これで、1号機 vm-db-01 がActとしてリソースが起動します。

fence_azure_armリソース



サンプル設定 fence_azure_arm

STONITH				
P	id	type		
#	STONITHリソースID	type	概要	
#	fence1-azure-arm	fence_azure_arm	fence_azure_armプラグイン (STONITHプラグイン)	
A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
#	options	msi	true	マネージドIDを使用する場合はtrueを指定(デフォルトはfalse)
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
#	start	60s		restart
#	monitor	60s	3600s	restart
#	stop	60s		ignore

STONITH				
P	id	type		
#	STONITHリソースID	type	概要	
#	fence2-azure-arm	fence_azure_arm	fence_azure_armプラグイン (STONITHプラグイン)	
A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
#	options	msi	true	マネージドIDを使用する場合はtrueを指定(デフォルトはfalse)
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
#	start	60s		restart
#	monitor	60s	3600s	restart
#	stop	60s		ignore

- ・リソースIDを変えて、2つSTONITHリソースを用意します。
- ・パラメータ options の msi
 - ・フェンシングリクエスト元の認証にシステム割り当てマネージドIDを利用するため、trueを設定します。

サンプル設定 fence_azure_armの配置

LOCATION_NODE				
#	rsc	prefers/avoids	node	score
	リソースID	prefers/avoids	ノード	スコア
	primary-group	prefers	vm-db-01	200
			vm-db-02	100
	fence1-azure-arm	avoids	vm-db-01	
	fence2-azure-arm	avoids	vm-db-02	

- ・「自分自身をSTONITHするような配置はしない」ように設定します。
 - vm-db-01をSTONITHするリソースは、「vm-db-01を避けて(avoids)配置する」という設定を行うことで、vm-db-02に配置されます。

構成、リソース設定がうまくいった場合

- これらのサンプル設定を実施してうまくいった場合は以下のようになります。

```
[vm-db-01]# pcs status --full
~~省略~~

Node List:
* Node vm-db-01 (1): online, feature set 3.16.2
* Node vm-db-02 (2): online, feature set 3.16.2

Full List of Resources:
* Resource Group: primary-group:
  * ipaddr          (ocf::heartbeat:IPAddr2): Started vm-db-01
  * azure-lb-primary (ocf::heartbeat:azure-lb): Started vm-db-01
  * fence1-azure-arm (stonith:fence_azure_arm):           Started vm-db-02
  * fence2-azure-arm (stonith:fence_azure_arm):           Started vm-db-01

~~省略~~
```

STONITHを試そう

リソース故障が発生した場合 1

- ・意図的に故障を起こしてみます。

Actで動いている azure-lb にいたずらしてみましょう。

プロセスの確認

```
[vm-db-01]# systemctl status pacemaker --no-pager
pacemaker.service - Pacemaker High Availability Cluster Manager
~~省略~~

CGroup: /system.slice/pacemaker.service
      |- 省略
      |└- 3766 /usr/bin/nc -l -k 60000
~~省略~~
```

リソース故障が発生した場合 2

- azure-lb のリソースエージェント
/usr/lib/ocf/resource.d/heartbeat/azure-lb
のシェルスクリプトを編集します。
- 停止処理時に行われる lb_stop() の1行目に
"return \$OCF_ERR_GENERIC" を追加して、
停止処理がうまくいかないようにします。

```
[vm-db-01]# vi /usr/lib/ocf/resource.d/heartbeat/azure-lb
~~省略~~
lb_stop(){
    return $OCF_ERR_GENERIC
    stop_rc=$OCF_SUCCESS
~~省略~~
```

リソース故障が発生した場合 3

- ・この状態でncプロセスをkillしてみましょう。
- ・正常時には停止処理をしつつ、フェイルオーバしますが…

プロセスをkillする

```
[vm-db-01]# pkill -KILL -f /usr/bin/nc
```

リソース故障が発生した場合 4

- 停止処理に異常が発生したため、vm-db-01がSTONITHされます。

Sby側だった2号機で確認

```
[vm-db-02]# pcs stonith history
reboot of vm-db-01 successful: delegate=vm-db-02, client=pacemaker-
controld.19554, origin=vm-db-02, completed='日付時刻'
1 event found
```

リソース故障が発生した場合 5

- Sby側へフェイルオーバし、仮想IPアドレスが2号機側に設定されます。

```
[vm-db-02]# pcs status --full
~~省略~~

Node List:
* Node vm-db-01 (1): OFFLINE
* Node vm-db-02 (2): online, feature set 3.16.2

Full List of Resources:
* Resource Group: primary-group:
  * ipaddr          (ocf::heartbeat:IPAddr2): Started vm-db-02
  * azure-lb-primary (ocf::heartbeat:azure-lb): Started vm-db-02
  * fence1-azure-arm (stonith:fence_azure_arm):           Started vm-db-02
  * fence2-azure-arm (stonith:fence_azure_arm):           Stopped

~~省略~~
```

リソース故障が発生した場合 6

- 内部LBによる仮想IPアドレスへの振り分けも、2号機側へ接続されます。

```
クライアントから仮想IPアドレスへのping
```

```
[client]# ping 10.0.0.34  
~~省略~~
```

```
64bytes from 10.0.0.34: icmp_seq=4 ttl=64 time=1.45 ms  
64bytes from 10.0.0.34: icmp_seq=5 ttl=64 time=1.83 ms  
64bytes from 10.0.0.34: icmp_seq=6 ttl=64 time=1.40 ms  
64bytes from 10.0.0.34: icmp_seq=7 ttl=64 time=1.74 ms  
64bytes from 10.0.0.34: icmp_seq=8 ttl=64 time=33.8 ms  
64bytes from 10.0.0.34: icmp_seq=23 ttl=64 time=2.62 ms  
64bytes from 10.0.0.34: icmp_seq=24 ttl=64 time=2.02 ms  
~~省略~~
```

切り替え時に当たって、
時間がかかっている。

切り替え時に流れたものはロスしている。

このような形で、AzureでPacemakerを試せます！

- ・手間はすこしかかります…
- ・それでもライセンスの確保や、認証の管理のアレコレをAzureに任せることができるので、まだマシ。
- ・似たような構成をどんどん立てれば複数台クラスタも検証できるので試してみましょう！

もっと知りたい方はこちらまで

<https://linux-ha-japan.github.io/>

OSDNから移転しました



今後もPacemakerを
よろしくお願いします

