

やってみよう！ Pacemaker による 高可用クラスタ構築 ～OpenShift Virtualization 編～

2025-10-17 OSC2025 Online/Fall

Linux-HA Japan プロジェクト

山本 拓也



自己紹介

- 名前

- 山本 拓也 (やまもと たくや)

- 経歴

- 2024 年 4 月に NTT OSS センタに配属
- 配属後, Pacemaker と OpenShift に初めて出会いました

- OSC は今回が初参加なのでお手柔らかにお願いします

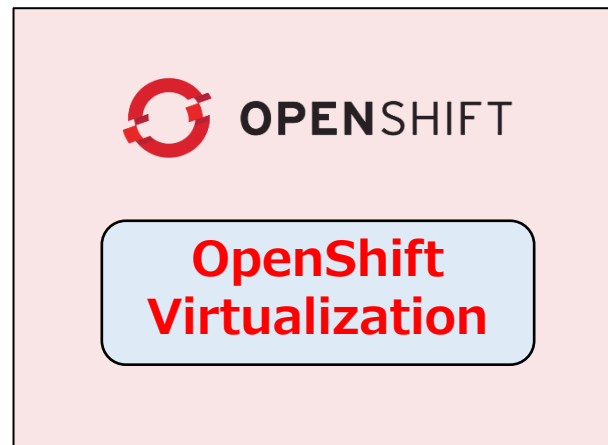
目次

1. OpenShift Virtualization (OCP-V) とは？
2. Pacemaker の役割
3. OCP-V 環境で Pacemaker を使ってみよう
4. おわりに

OpenShift Virtualization (OCP-V) とは？

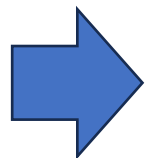
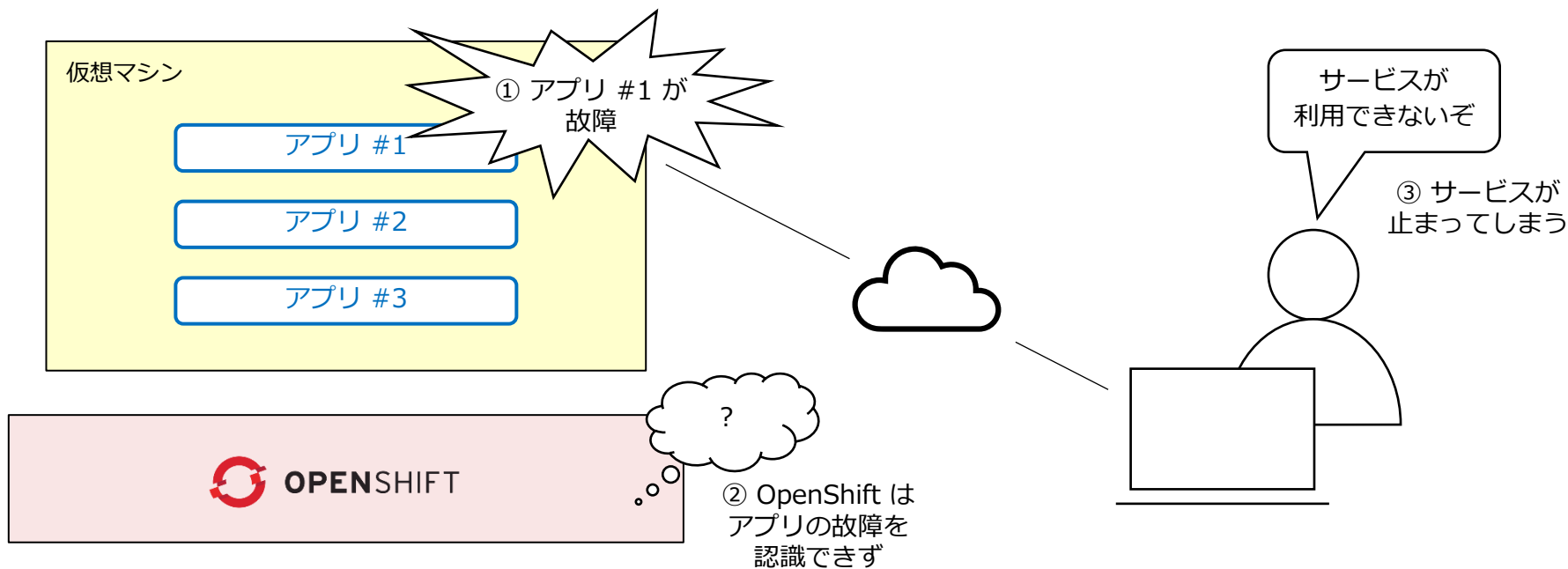
OpenShift Virtualization (OCP-V)

- 仮想マシンを起動・管理する OpenShift の機能の一つ
 - OpenShift : Red Hat 社が提供しているコンテナと仮想マシンを管理可能な統合プラットフォーム
- 仮想マシン本体が故障しても復旧することができます
 - OCP-V 以外の OpenShift の他の機能と組み合わせることで実現
- しかし, 対応できない故障もあります...



OCP-V だけではカバーできない故障

- 仮想マシン内で稼働しているアプリの故障が検知できません
 - アプリは復旧されないため、サービスが停止



Pacemaker と組み合わせて
高可用クラスタを構築しよう！

Pacemaker の役割

まず高可用クラスタって何？

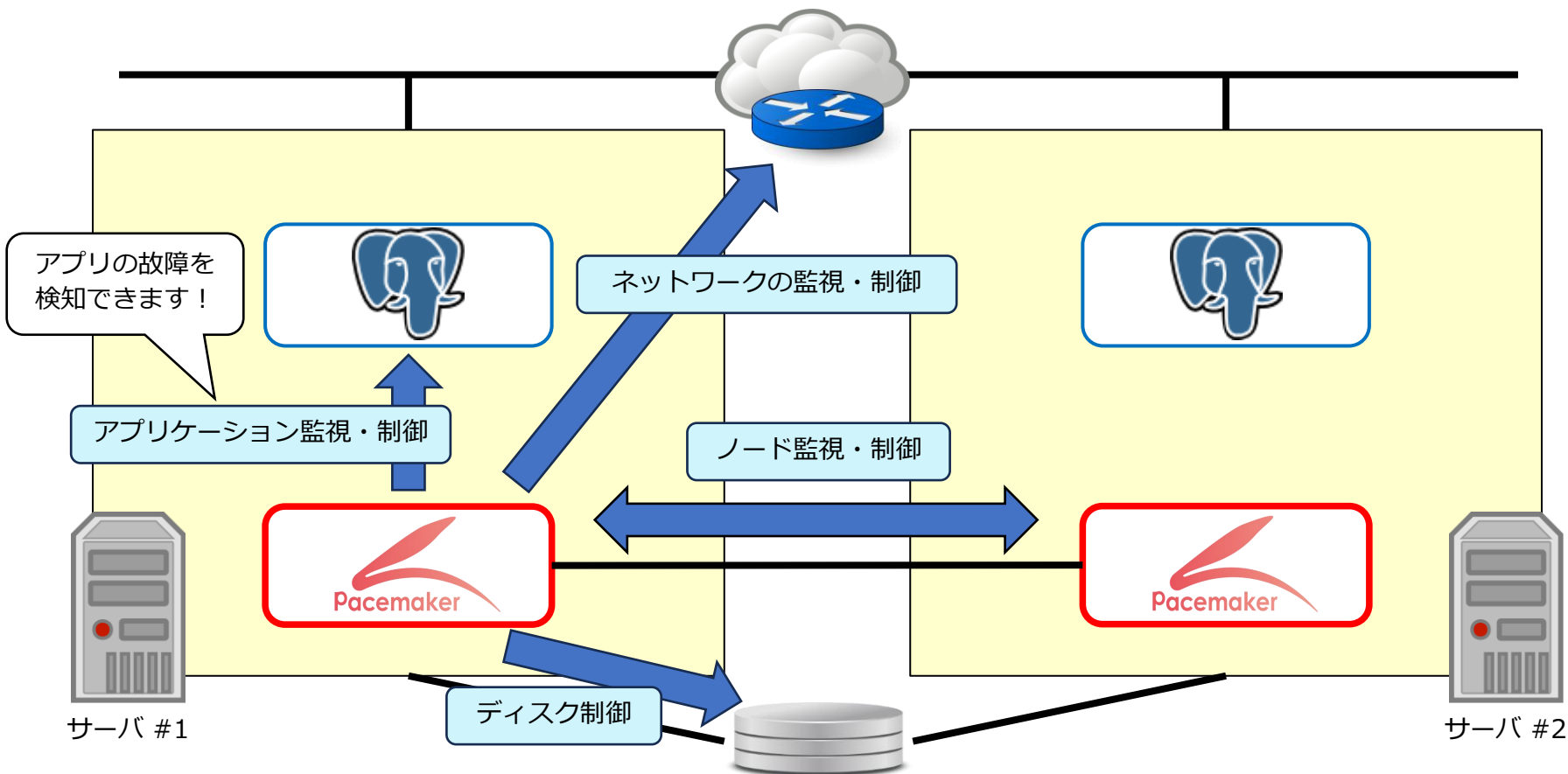
- サービスが停止する時間を可能な限り短くすることを目的としたクラスタです

高可用性 = High Availability

- OCP-V 環境で高可用クラスタを構築するために利用するソフトが HA クラスタソフトとなります
 - HA クラスタソフトのひとつである **Pacemaker** を紹介

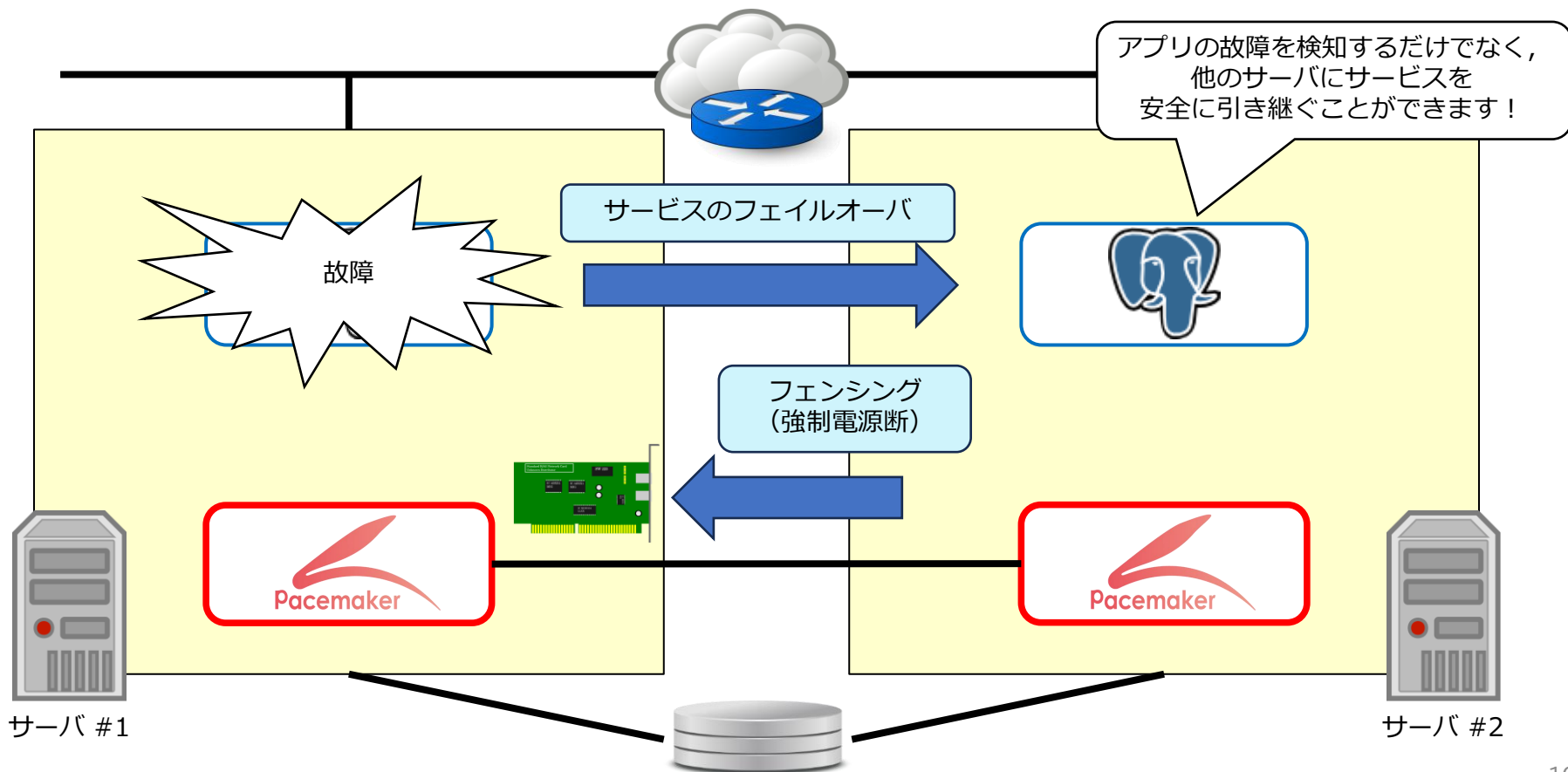
Pacemaker の概要 (1/2)

- オープンソースで開発されている HA クラスタソフト
 - サーバや**アプリケーション**の監視・制御



Pacemaker の概要 (2/2)

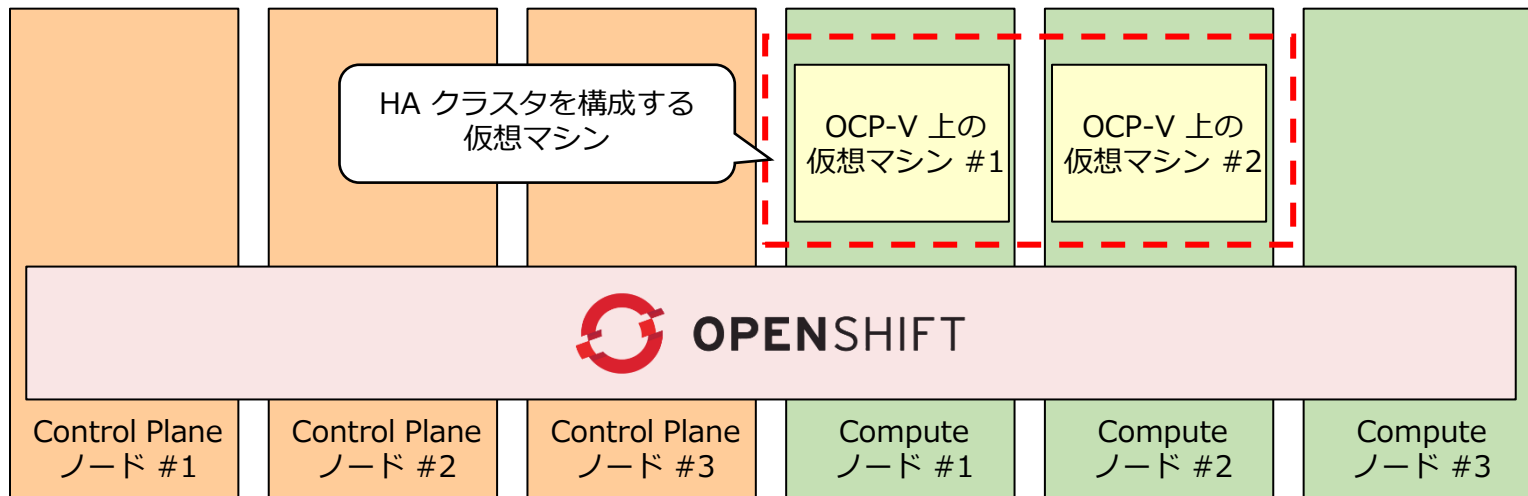
- Pacemaker の役割として、他に以下があります
 - 故障検知時に自動的にフェイルオーバー
 - フェンシング（強制電源断）によるデータの安全性確保



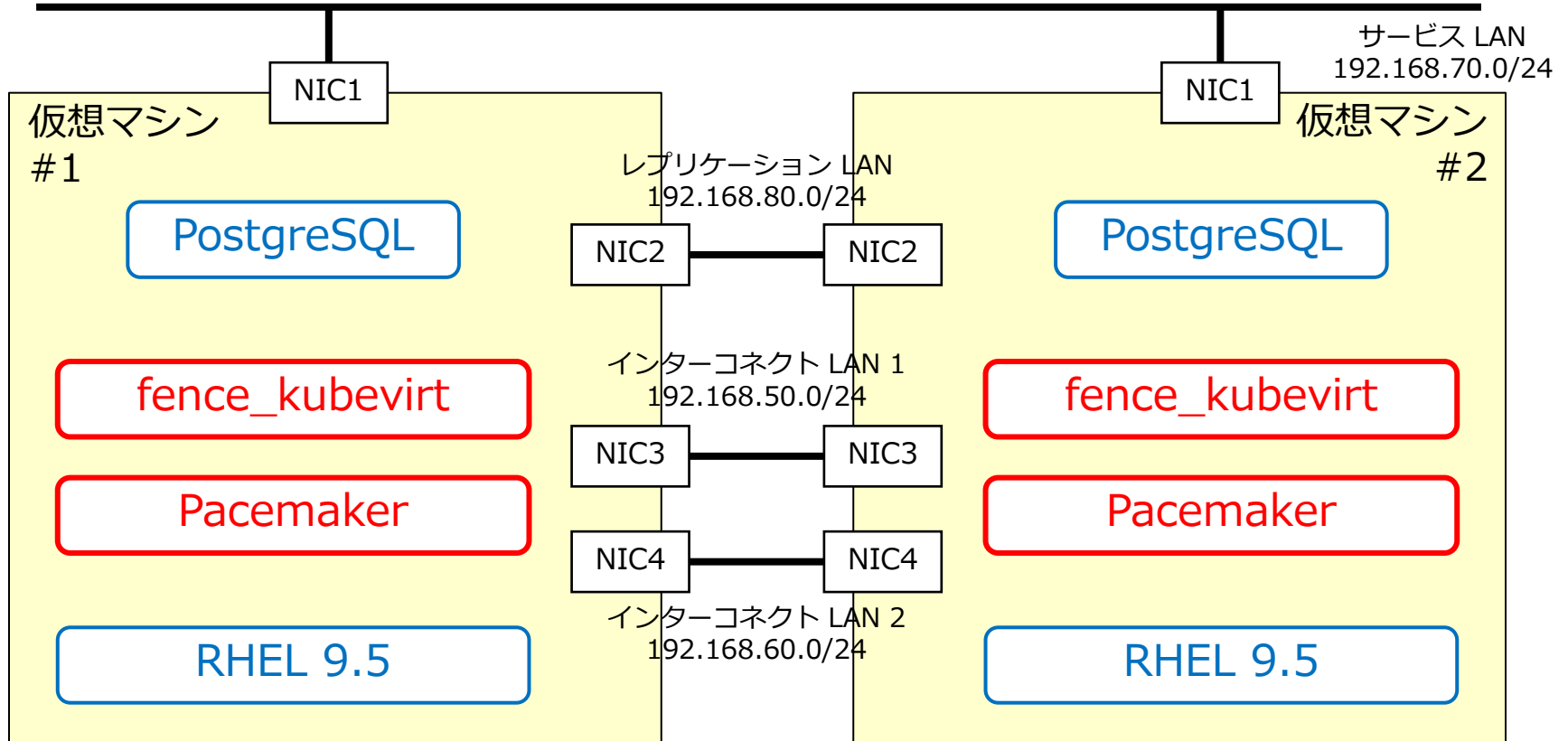
**OCP-V 環境で Pacemaker を
使ってみよう**

Pacemaker を試す OCP-V 環境

- OpenShift : 4.18.11
 - Control Plane ノード : 物理サーバ 3 台
 - OpenShift の制御や監視, 設定などを実施するサーバ
 - Compute ノード : 物理サーバ 3 台
 - ユーザのワークロード (仮想マシンを含むアプリやサービス) を実行するサーバ
- OpenShift Virtualization (OCP-V) : 4.18.13
- 作業用端末 (RHEL 9.5, OCP-V 環境の管理に利用)



Pacemaker を試す HA クラスタ構成



Pacemaker を使用するまでの流れ

1. 仮想マシンの作成
2. OCP-V 環境に NW を追加
3. Pacemaker の設定
4. fence_kubevirt の設定
5. リソースの設定
6. リソース故障のお試し！

※ OCP-V 環境ならではの設定を中心に紹介します

Pacemaker を試す HA クラスタ構成

1. 仮想マシンの作成

仮想マシン
#1

RHEL 9.5

仮想マシン
#2

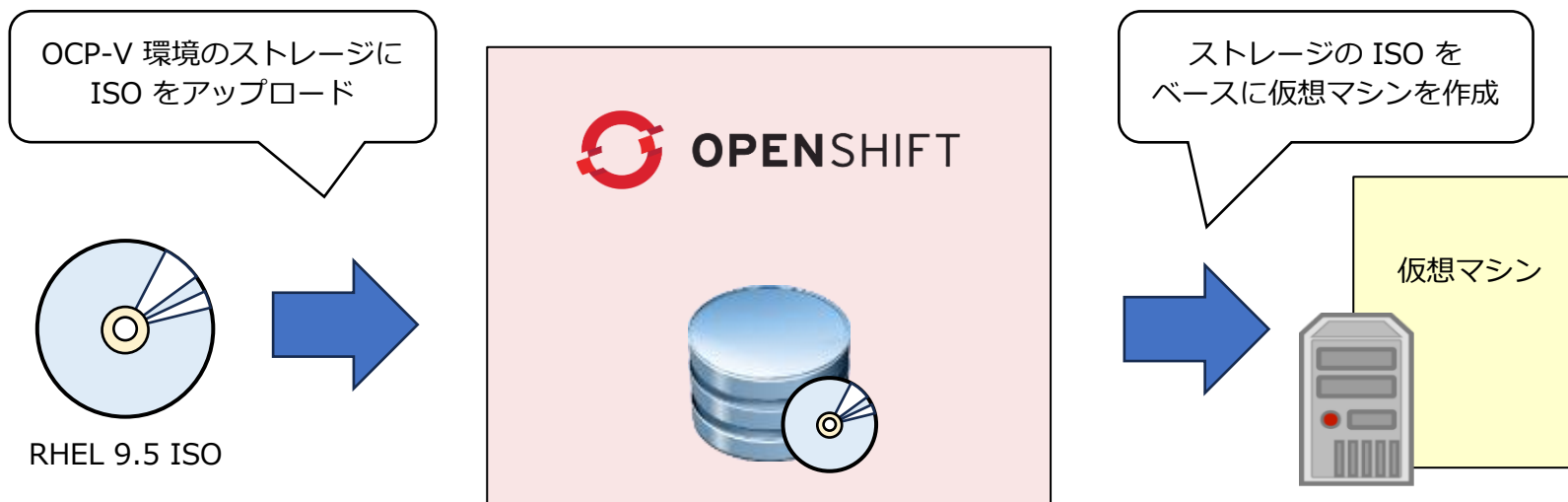
RHEL 9.5



OPENSIFT

1. 仮想マシンの作成 (1/6)

- 事前に RHEL 9.5 の ISO を入手しておきます
- 基本的な作成方法は以下に沿って作成します
 - Red Hat 社のエンジニアが公開しているブログ
[ゼロからはじめるOpenShift Virtualization \(4\) OpenShift Virtualizationのインストールと実行 - 赤帽エンジニアブログ](#)
 - 本発表ではテンプレートから作成する方法を採用



1. 仮想マシンの作成 (2/6)

- クラスタを構築するプロジェクトを選び、事前に用意した ISO をアップロードします
 - oc / virtctl : OpenShift / OCP-V 上の仮想マシンを制御するコマンド
作業端末のコマンドラインから実行

```
$ oc project osc2025
Now using project "osc2025" on server https://api.ocp.home.lab:6443

$ virtctl image-upload dv rhel-9.5-center-iso --size=12G ¥
--image-path=rhel-9.5-x86_64-dvd.iso --insecure
PVC osc2025/rhel-9.5-center-iso not found
DataVolume osc2025/rhel-9.5-center-iso created
~略~
Uploading rhel-9.5-x86_64-dvd.iso completed successfully
```

パラメータ紹介

- rhel-9.5-center-iso: アップロード時のリソース名
- --size: アップロード先の確保サイズ
- --image-path: アップロードデータのパス
- --insecure: アップロード時の検証スキップ

1. 仮想マシンの作成 (3/6)

- アップロードが成功していれば, Web コンソール上で確認することができます

Red Hat OpenShift

管理者向け表示

ホーム

Operator

Workloads

Virtualization

ネットワーク

ストレージ

PersistentVolumes

PersistentVolumeClaims

StorageClasses

プロジェクト: osc2025

クラスタを構築するプロジェクトを選択

PersistentVolumeClaims

名前: rhel-9.5

名前: rhel-9.5 × すべてのフィルターをクリア

名前	ステータス	PersistentVolumes	容量
PVC rhel-9.5-center-iso	Bound	PV pvc-4d51f68f-5a6f-4730-8a5c-2e03aed9dfe2	11.85 GiB

1. 仮想マシンの作成 (4/6)

- Web コンソール上でテンプレートカタログを選択し、
以下のようにパラメータを設定して仮想マシンを作成します

Red Hat OpenShift

プロジェクト: osc2025

クラスタを構築するプロジェクトを選択

Create new VirtualMachine

Select an option to create a VirtualMachine from.

InstanceTypes Template catalog

テンプレートカタログを選択

Template project: All projects

Default templates: 3 items

Name ↑	Workl...	Boot source	CPU Memory
Red Hat Enterprise Linux 7 VM (rhe7-server-small)	Server	PVC	CPU 1 Memory 2 GiB
Red Hat Enterprise Linux 8 VM (rhe8-server-small)	Server	PVC	CPU 1 Memory 2 GiB
Red Hat Enterprise Linux 9 VM (rhe9-server-small)	Server	PVC (auto import)	Source available CPU 1 Memory 2 GiB

RHEL 9 のテンプレートを選択

Web コンソール

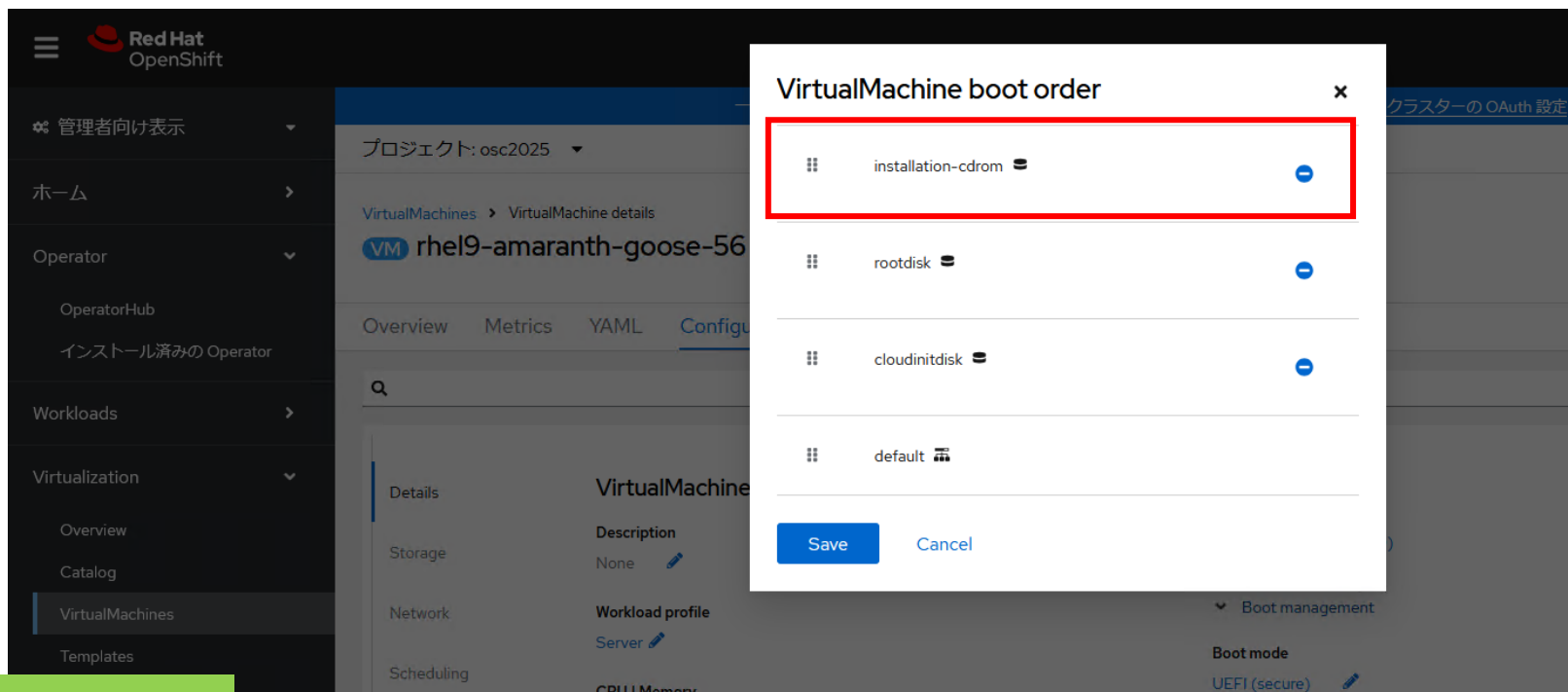
1. 仮想マシンの作成 (5/6)

- Web コンソール上でテンプレートカタログを選択し、
以下のようにパラメータを設定して仮想マシンを作成します

1. 仮想マシンの作成 (6/6)

• 注意点

- RHEL のインストール設定が完了した後、システムの再起動が必要となりますが、そのまま再起動すると再度 ISO 起動となります
- そこで仮想マシンを停止し、「installation-cdrom」の「-」を押して boot order の先頭を「root-disk」に変更してから再起動します



The screenshot displays the Red Hat OpenShift Web Console interface. A modal dialog titled "VirtualMachine boot order" is open, showing a list of boot devices. The "installation-cdrom" entry is highlighted with a red rectangle. Below it are "rootdisk", "cloudinitdisk", and "default". The "Save" button is visible at the bottom of the dialog. The background shows the console navigation menu and the details of a virtual machine named "rhel9-amaranth-oose-56".

Pacemaker を使用するまでの流れ

1. 仮想マシンの作成

2. OCP-V 環境に NW を追加

3. Pacemaker の設定

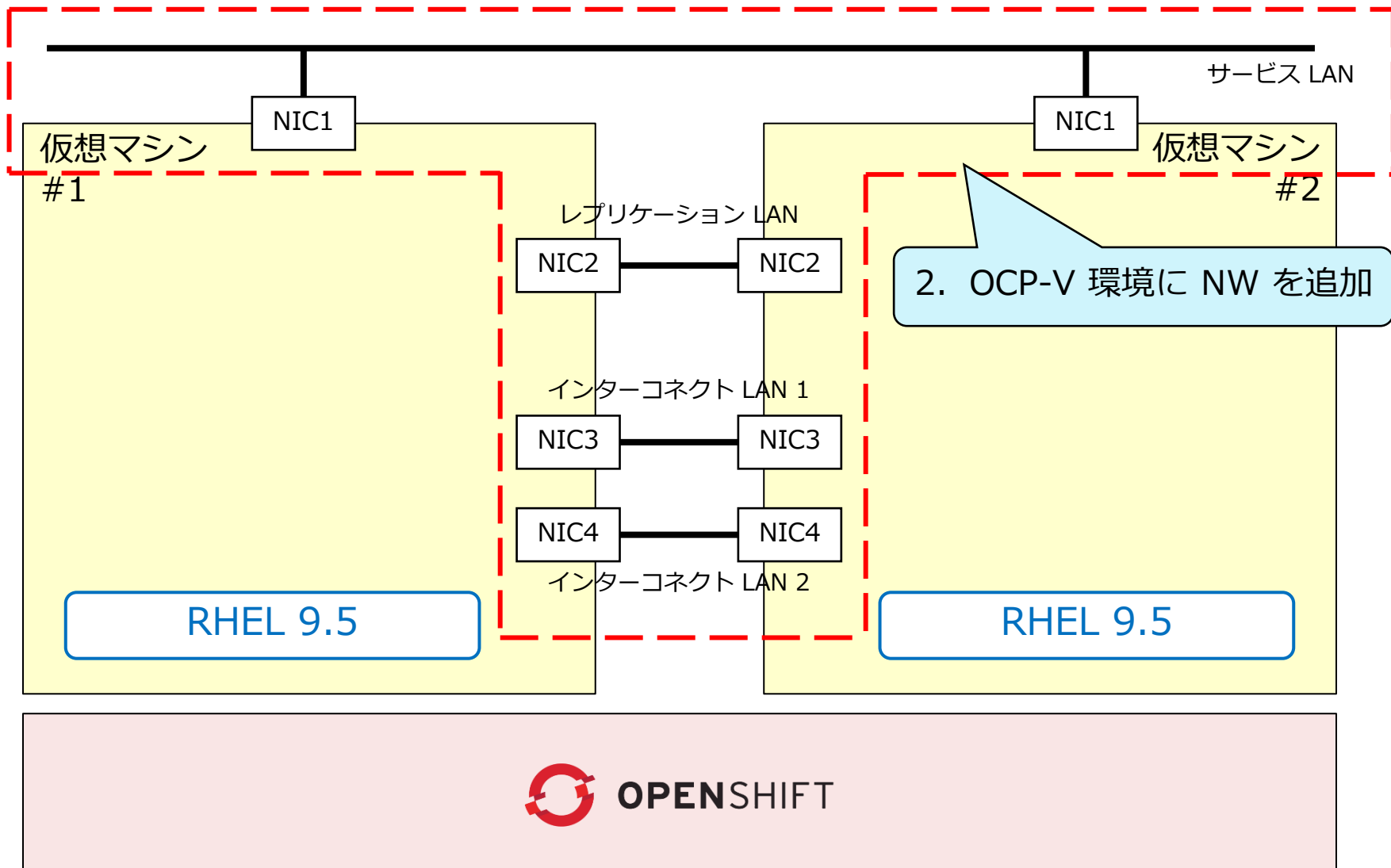
4. fence_kubevirt の設定

5. リソースの設定

6. リソース故障のお試し！

※ OCP-V 環境ならではの設定を中心に紹介します

Pacemaker を試す HA クラスタ構成



2. OCP-V 環境に NW を追加 (1/8)

- クラスタを作成するためには、用途ごとの NW を用意して仮想マシンをその NW に接続する必要があります
- NW の追加と仮想マシンの接続を以下の手順で設定します
 - 2-1. OCP-V 環境に NW 設定を行うオペレータをインストール
 - 2-2. OCP-V 環境に NW 設定を適用するポリシーを作成
 - 2-3. 仮想マシンに NIC を追加するための定義を作成
 - 2-4. 仮想マシンに NIC を追加

2. OCP-V 環境に NW を追加 (2/8)

- Web コンソールで「Kubernetes NMState Operator」をインストールします

The screenshot shows the Red Hat OpenShift OperatorHub interface. The search bar contains the text "nmstate", and the search results display the "Kubernetes NMState Operator" by Red Hat, Inc. The search bar and the operator card are highlighted with red boxes. A callout box points to the search bar with the text "nmstate" で検索するとオペレータが表示 (When you search for "nmstate", the operator is displayed).

OperatorHub

プロジェクト: osc2025

一時的な管理ユーザーとしてログインしています。

nmstate

Kubernetes NMState Operator
Red Hat, Inc. 提供のバージョン 4.18.0-20250910149

インストール

チャンネル: stable

バージョン: 4.18.0-20250910149

機能レベル

- 基本的なインストール
- シームレスなアップグレード
- 完全なライフサイクル
- 詳細な分析
- 自動パイロット

ソース: Red Hat

プロバイダー: Red Hat, Inc.

2. OCP-V 環境に NW を追加 (3/8)

- NNCP (Node Network Configuration Policy) を作成します
 - NNCP : OpenShift 構成ノード の NIC に対して設定を行うポリシー
OCP-V 上に任意の複数の NW を通すことが可能

Red Hat OpenShift

ocpadmin

Create NodeNetworkConfigurationPolicy [Edit YAML](#)

Node network is configured and managed by NM state. Create a node network configuration policy to describe the requested network configuration on your nodes in the cluster. The node network configuration enactment reports the network policies enacted upon each node.

Apply this NodeNetworkConfigurationPolicy only to specific subsets of nodes using the node selector

Policy name *
demo-ovs-br-ens224

Description

Policy Interface(s) ?

+ Add another interface to the policy

Open vSwitch bridge demo-ovs-br

Interface name *
demo-ovs-br

Network state *
Up

任意の NNCP 名を入力

任意のブリッジ名を入力

2. OCP-V 環境に NW を追加 (4/8)

- NNCP (Node Network Configuration Policy) を作成します
 - NNCP : OpenShift 構成ノード の NIC に対して設定を行うポリシー
OCP-V 上に任意の複数の NW を通すことが可能

作成する NW の種類を選択
ここでは "Open vSwitch bridge" を選択

NW の設定を割りあてる
OpenShift 構成ノードの NIC を指定

任意の localnet 名を入力

前スライドで入力した
ブリッジ名を再度入力

2. OCP-V 環境に NW を追加 (5/8)

- NAD (NetworkAttachmentDefinition) を作成します
 - NAD : OCP-V 上の仮想マシンに追加するための NIC を定義

Red Hat OpenShift

プロジェクト: osc2025

クラスタを構築するプロジェクトを選択

Create NetworkAttachmentDefinition

Configure via: Form view YAML view

Name *

任意の NAD 名を入力

Description

Network Type *

NAD の種類を選択
ここでは "OVN Kubernetes secondary localnet network" を選択

Bridge mapping *

NAD をマッピングする localnetを指定
※NNCP で作成した localnet と同じにすること

MTU

VLAN

2. OCP-V 環境に NW を追加 (6/8)

- 仮想マシンに NIC を追加し, 仮想マシンを再起動します
 - 追加した NIC の IP アドレスの設定で OCP-V 固有の対応はなし
 - この手順は仮想マシン #1, 2 の両方で実施

Red Hat OpenShift

プロジェクト: osc2025

VirtualMachines > VirtualMachine details

VM rhel9-amaranth-goose-56 Stopped

Overview Metrics YAML **Configuration** Events Console Snapshots Diagnostics

Network interfaces

Add network interface

フィルター 名前 名前を検索...

Name ↑	Model ↓	Network ↓	Type ↓	MAC address ↓
default	virtio	Pod networking	Masquerade	02:78:c2:00:00:13

“Network” を選択

仮想マシンの “Configuration” を選択

このボタンをクリック

2. OCP-V 環境に NW を追加 (7/8)

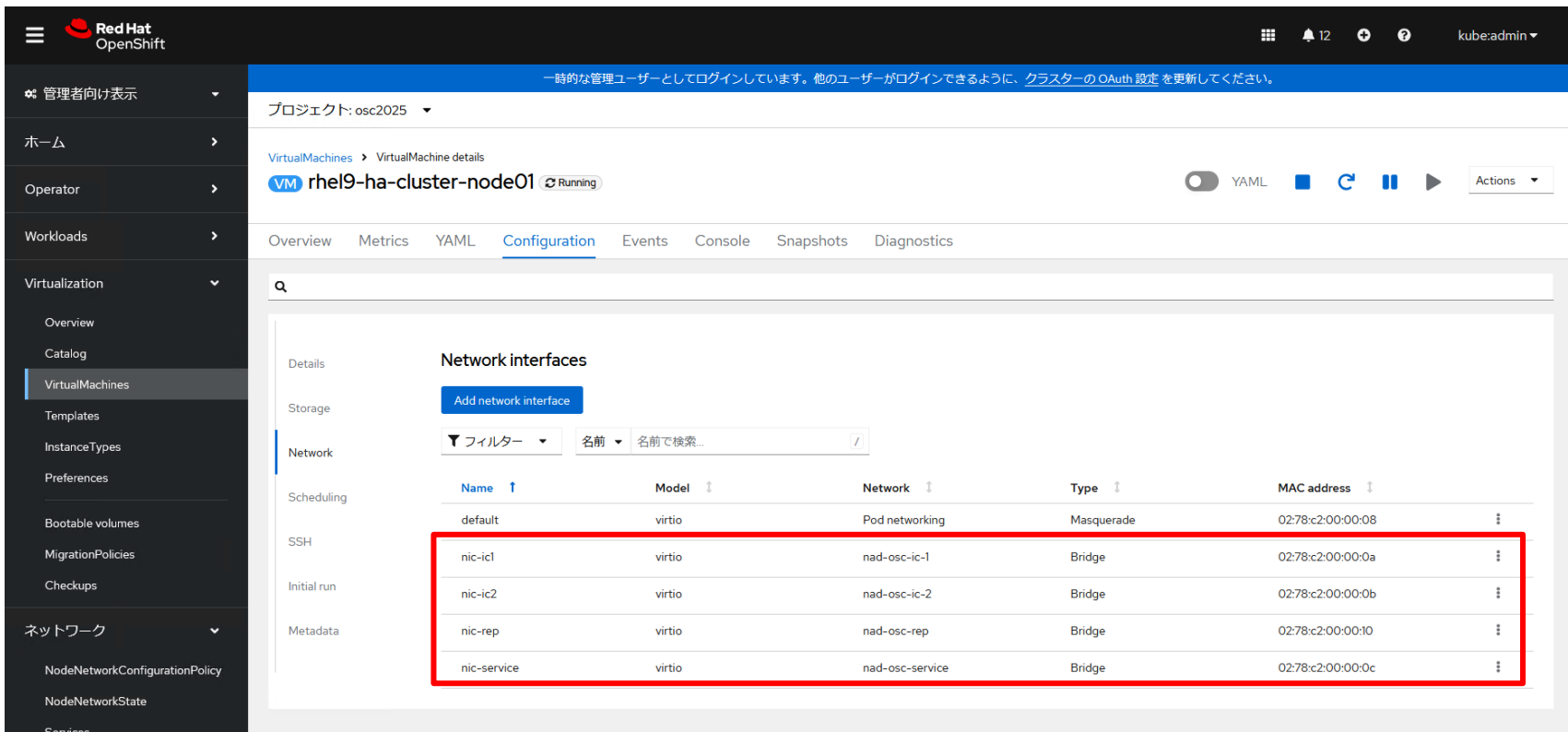
- 仮想マシンに NIC を追加し, 仮想マシンを再起動します
 - 追加した NIC の IP アドレスの設定で OCP-V 固有の対応はなし
 - この手順は仮想マシン #1, 2 の両方で実施

The screenshot shows the 'Add network interface' dialog in the Red Hat OpenShift web console. The dialog has three main fields: 'Name', 'Model', and 'Network'. The 'Name' field contains 'nic-salmon-pike-96', the 'Model' is set to 'virtio', and the 'Network' field contains 'osc2025/nad-osc-ic-1'. There are callout boxes with Japanese text: '任意の NIC 名を入力' (Enter an arbitrary NIC name) pointing to the Name field, and '先ほど作成した NAD を選択' (Select the NAD created earlier) pointing to the Network field. The background shows the OpenShift interface with a table of network interfaces.

Name	Model	Network	Type	MAC address
default	virtio	Pod networking	Masquerade	02:78:c2:00:00:13

2. OCP-V 環境に NW を追加 (8/8)

- 新たに NIC が追加されていることを確認します



The screenshot shows the Red Hat OpenShift web console interface. The left sidebar contains navigation menus for '管理者向け表示', 'ホーム', 'Operator', 'Workloads', 'Virtualization', and 'ネットワーク'. The main content area displays the configuration for a virtual machine named 'rhel9-ha-cluster-node01'. The 'Network interfaces' section is active, showing a table of network interfaces. The table has columns for Name, Model, Network, Type, and MAC address. The following table is shown:

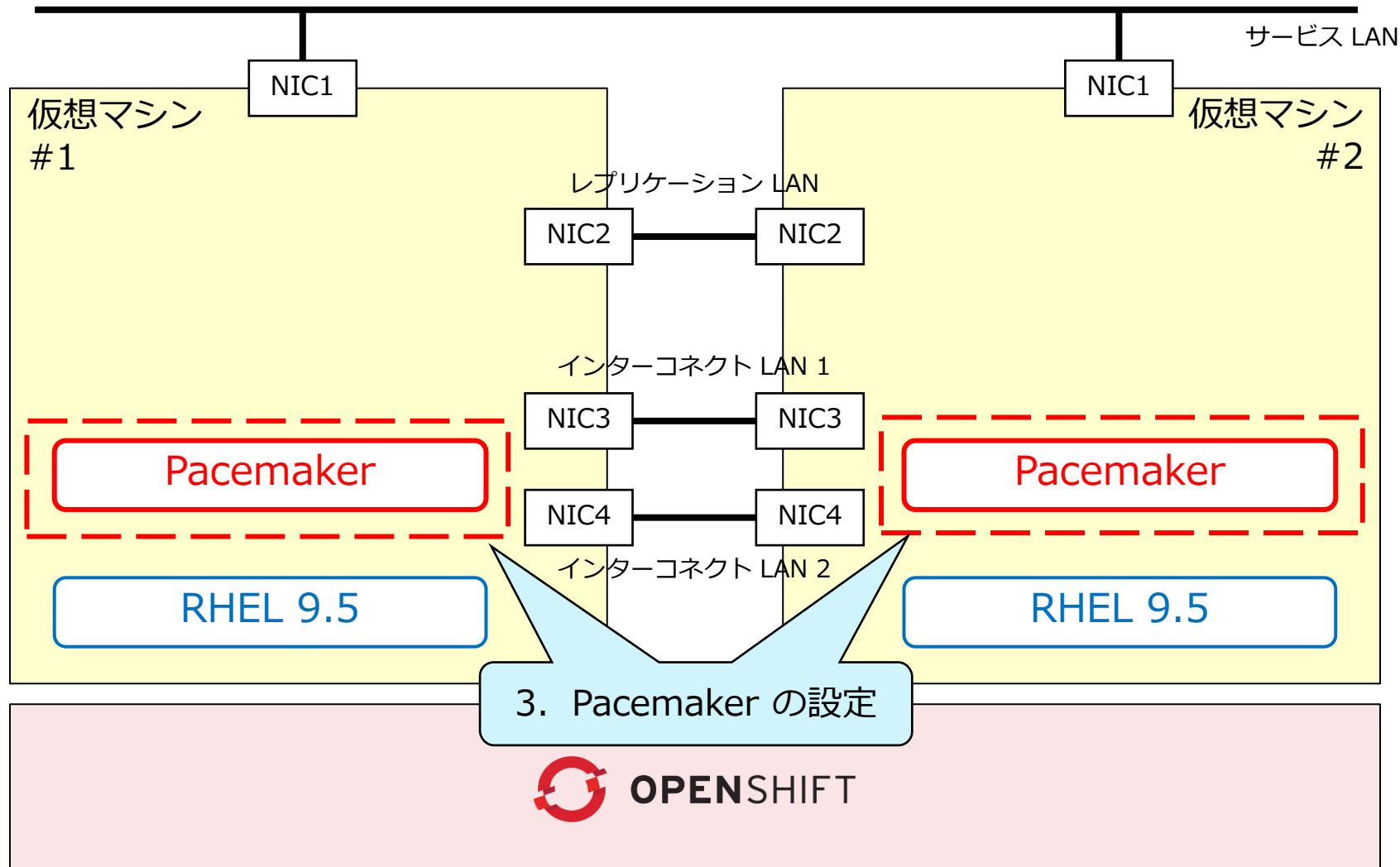
Name	Model	Network	Type	MAC address
default	virtio	Pod networking	Masquerade	02:78:c2:00:00:08
nic-ic1	virtio	nad-osc-ic-1	Bridge	02:78:c2:00:00:0a
nic-ic2	virtio	nad-osc-ic-2	Bridge	02:78:c2:00:00:0b
nic-rep	virtio	nad-osc-rep	Bridge	02:78:c2:00:00:10
nic-service	virtio	nad-osc-service	Bridge	02:78:c2:00:00:0c

Pacemaker を使用するまでの流れ

1. 仮想マシンの作成
2. OCP-V 環境に NW を追加
3. Pacemaker の設定
4. fence_kubevirt の設定
5. リソースの設定
6. リソース故障のお試し！

※ OCP-V 環境ならではの設定を中心に紹介します

Pacemaker を試す HA クラスタ構成



3. Pacemaker の設定 (1/3)

- Pacemaker に使用するパッケージをインストールします
 - この手順は仮想マシン #1, 2 の両方で実施

```
# dnf -y install pcs pacemaker fence-agents-all  
# dnf -y install fence-agents-kubevirt
```

- クラスタが利用するポートの設定を行います
 - この手順は仮想マシン #1, 2 の両方で実施

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

3. Pacemaker の設定 (2/3)

- pcsd.service を起動させます
 - この手順は仮想マシン #1, 2 の両方で実施
 - pcsd : Pacemaker/Corosync Configuration System daemon
Pacemaker と Corosync を管理するツール

```
# systemctl enable pcsd.service --now
```

- hacluster ユーザのパスワードを設定します
 - この手順は仮想マシン #1, 2 の両方で実施

```
# passwd hacluster  
ユーザー hacluster のパスワードを変更。  
新しいパスワード:  
新しいパスワードを再入力してください:  
passwd: すべての認証トークンが正しく更新できました。
```

3. Pacemaker の設定 (3/3)

- クラスタノードの認証設定を行います
 - この手順は仮想マシン #1, 2 の**どちらか**で実施

```
# pcs host auth rhel9-ha-cluster-node01 addr=192.168.70.10 ¥
rhel9-ha-cluster-node02 addr=192.168.70.20
Username: hacluster
Password:
rhel9-ha-cluster-node02: Authorized
rhel9-ha-cluster-node01: Authorized
```

設定したパスワードを入力

- 故障検知時のクラスタノードの挙動を設定します
 - この手順は仮想マシン #1, 2 の両方で実施
 - Linux-HA Japan の推奨設定

```
# vi /etc/sysconfig/pacemaker
~略~
PCMK_fail_fast=yes
PCMK_panic_action=sync-reboot
```

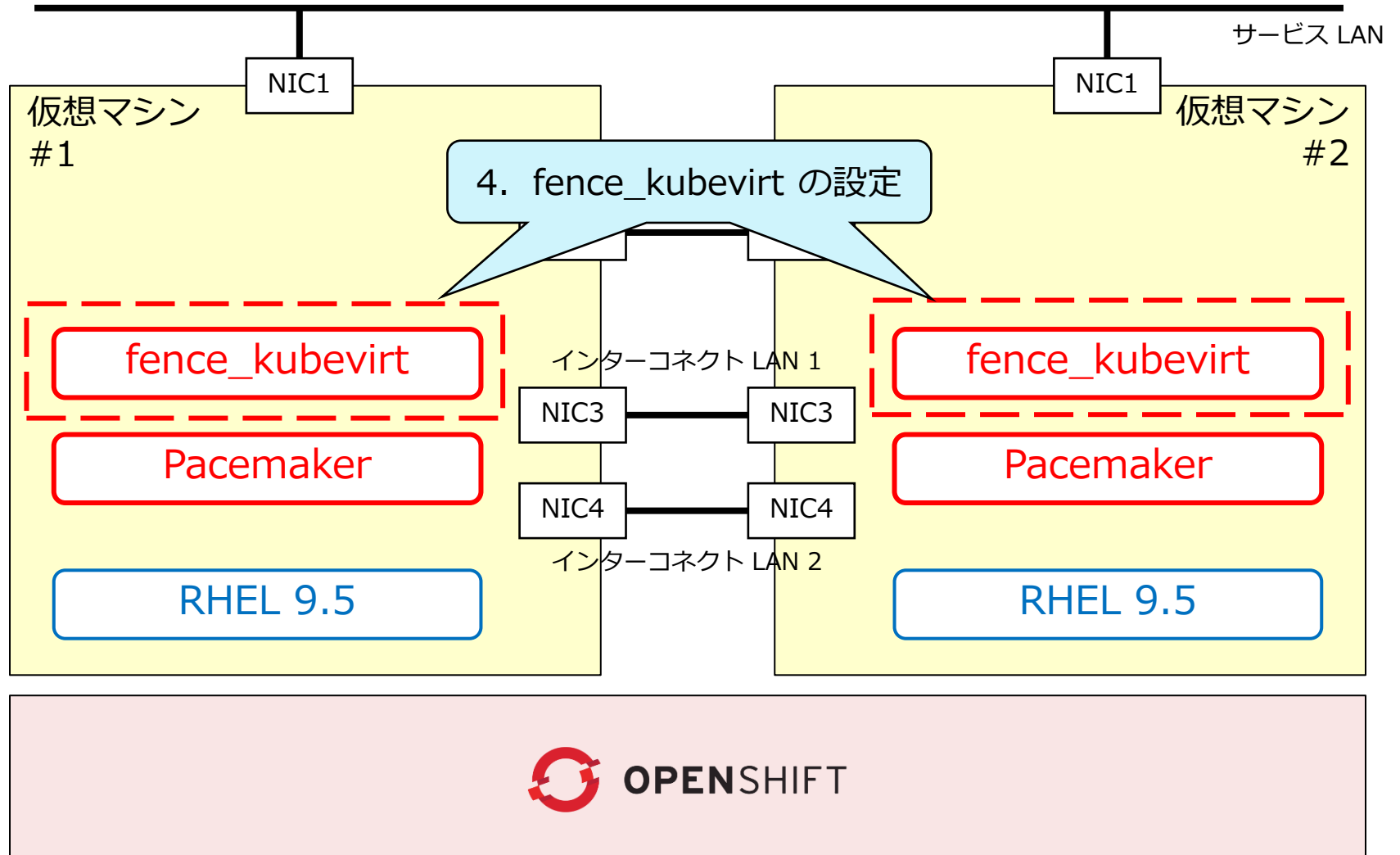
- PCMK_fail_fast=yes:
Pacemaker に異常が起きた際に再起動
- PCMK_panic_action=sync-reboot
メモリ上にある変更内容をディスクに書き込み後、再起動

Pacemaker を使用するまでの流れ

1. 仮想マシンの作成
2. OCP-V 環境に NW を追加
3. Pacemaker の設定
4. fence_kubevirt の設定
5. リソースの設定
6. リソース故障のお試し！

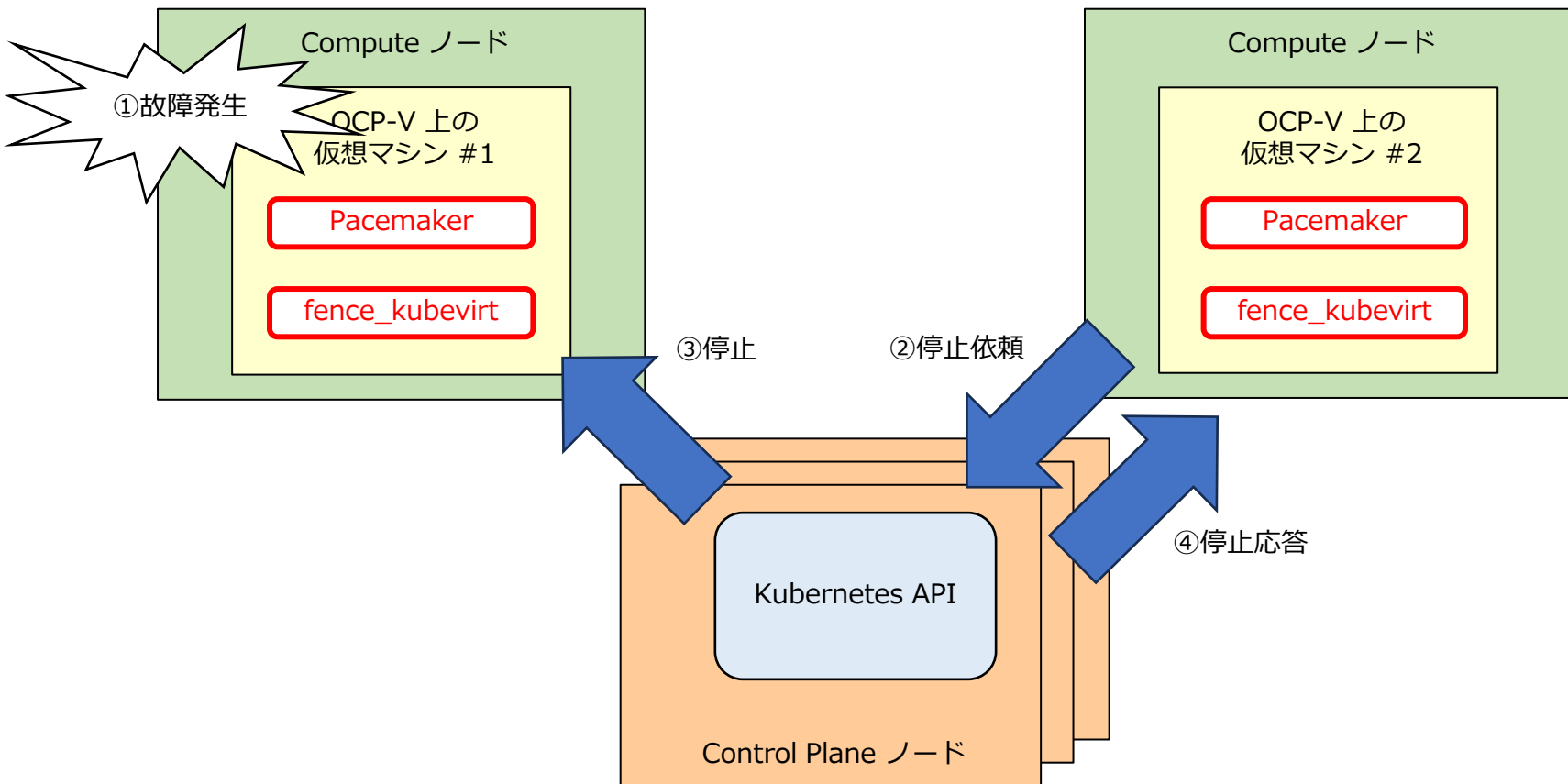
※ OCP-V 環境ならではの設定を中心に紹介します

Pacemaker を試す HA クラスタ構成



4. fence_kubevirt の設定 (1/11)

- fence_kubevirt は Control Plane の Kubernetes API に
 依頼して故障したノードをフェンシングします



4. fence_kubevirt の設定 (2/11)

- 仮想マシンから Kubernetes API へのアクセスを許可するために以下の手順で設定します
 - 4-1. サービスアカウントの作成
 - 4-2. 作成したサービスアカウントに管理者権限を付与
 - 4-3. 仮想マシンにサービスアカウントの情報を追加
 - 4-4. fence_kubevirt が正常に動作するか確認

4. fence_kubevirt の設定 (3/11)

- サービスアカウントを作成します

Red Hat OpenShift

Virtualization > ネットワーク > ストレージ > Builds > モニタリング > コンピュート > ユーザー管理 > 管理 >

プロジェクト: osc2025

クラスタを構築するプロジェクトを選択

ServiceAccount の作成

YAML または JSON 定義を手動で入力するか、またはファイルをエディターにドラッグアンドドロップして作成します。

Alt + F1 アクセシビリティヘルプ | ショートカットの表示 | ツールチップを表示 | サイドバーの表示

```
1 apiVersion: v1
2 kind: ServiceAccount
3 metadata:
4   name: fence-user
5   namespace: osc2025
6
```

任意のユーザ名を入力

クラスタを構築するプロジェクト名を入力

作成 キャンセル ダウンロード

4. fence_kubevirt の設定 (4/11)

- サービスアカウントに管理者権限を付与するための RoleBinding を作成します

Red Hat OpenShift

一時的な管理ユーザーとしてログインしています。他のユーザーがログインできるように、[クラスターの OAuth 設定](#) を更新してください。

RoleBinding の作成

選択したロールにユーザー/グループを関連付けて、許可するアクセス内容およびリソースタイプを定義します。

バインディングタイプ

- namespace のロールバインディング (RoleBinding)
選択した namespace 内のユーザーまたはユーザーのセットにパーミッションを付与します。
- クラスター全体のロールバインディング (ClusterRoleBinding)
パーミッションをクラスターレベルおよびすべての namespace のユーザーまたはユーザーのセットに付与します。

RoleBinding

名前*
osc2025-fencing

Namespace*
PR osc2025

特定のサービスアカウントに権限を付与するタイプを選択

任意の RoleBinding 名を入力

サービスアカウントが所属するプロジェクトと同じプロジェクトを選択

4. fence_kubevirt の設定 (5/11)

- サービスアカウントに管理者権限を付与するための RoleBinding を作成します

Red Hat OpenShift

Virtualization >

ネットワーク >

ストレージ >

Builds >

モニタリング >

コンピュート >

ユーザー管理 >

User

Group

ServiceAccounts

Roles

RoleBindings

管理 >

一時的な管理ユーザーとしてログインしています。他のユーザーがログインできるように、[クラスターの OAuth 設定](#) を更新してください。

Role

Role 名 *

CR admin

付与したい権限を選択

サブジェクト

User

グループ

ServiceAccount

権限を付与する対象を選択

namespace の選択 *

PR osc2025

サービスアカウントが所属するプロジェクトを選択

サブジェクトの名前 *

osc2025-fencing-subject

任意のサブジェクト名を入力

作成 キャンセル

4. fence_kubevirt の設定 (6/11)

- サービスアカウントのシークレットを作成した後に、管理者権限をサービスアカウントに付与します
 - シークレット：証明書やトークンなどの機密情報を管理するデータ

```
$ vi service-ac-osc2025.yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-sa-osc2025
  annotations:
    kubernetes.io/service-account.name: "fence-user"
type: kubernetes.io/service-account-token

$ oc apply -f service-ac-osc2025.yaml

$ oc adm policy add-role-to-user admin -z fence-user -n osc2025
clusterrole.rbac.authorization.k8s.io/admin added: "fence-user"
```

4. fence_kubevirt の設定 (7/11)

- サービスアカウントのシークレットを確認します

Red Hat OpenShift

管理者向け表示

ホーム

Operator

Workloads

- Pods
- Deployments
- DeploymentConfigs
- StatefulSets
- シークレット
- ConfigMaps
- CronJobs
- Jobs
- DaemonSets
- ReplicaSet
- ReplicationControllers
- HorizontalPodAutoscalers
- PodDisruptionBudgets

Virtualization

プロジェクト: osc2025

シークレット

作成

フィルター

名前

名前を検索...

名前	タイプ	サイズ	作成済み
secret-sa-osc2025	kubernetes.io/service-account-token	4	2025年9月12日 18:13
yama-sshkey	Opaque	1	2025年9月10日 15:45

作成したシークレットを選択

4. fence_kubevirt の設定 (8/11)

- サービスアカウントの証明書とトークンを確認します

Red Hat OpenShift

管理者向け表示

ホーム

Operator

Workloads

Pods

Deployments

DeploymentConfigs

StatefulSets

シークレット

ConfigMaps

CronJobs

Jobs

DaemonSets

ReplicaSet

ReplicationControllers

HorizontalPodAutoscalers

PodDisruptionBudgets

Virtualization

プロジェクト: osc2025

作成日時
2025年9月12日 18:13

オーナー
オーナーなし

データ

証明書を確認
※次の手順に必要

値を表示する

ca.crt

namespace

service-ca.crt

トークンを確認
※次の手順に必要

token

4. fence_kubevirt の設定 (9/11)

- 証明書とトークンを仮想マシンに追加します
 - この手順は仮想マシン #1, 2 の両方で実施

```
# mkdir /root/.kube
# vi /root/.kube/config
apiVersion: v1
clusters:
- cluster:
  #insecure-skip-tls-verify: true
  certificate-authority: /root/rootCA.pem
  server: https://api.ocp.home.lab:6443
  name: api.ocp.home.lab:6443
contexts:
- context:
  cluster: api.ocp.home.lab:6443
  user: fence-user/api.ocp.home.lab:6443
  name: osc2025/api.ocp.home.lab:6443/fence-user
(続く)
```

下線部の情報は環境合わせて読み替えが必要

- サービスアカウント名
- プロジェクト名
- OpenShift サーバドメイン名
- 証明書のパス

4. fence_kubevirt の設定 (10/11)

- 証明書とトークンを仮想マシンに追加します
 - この手順は仮想マシン #1, 2 の両方で実施

(続き)

```
current-context: ocp-virt/api.ocp.home.lab:6443/fence-user
kind: Config
preferences: {}
users:
- name: fence-user/api.ocp.home.lab:6443
  user:
    token: XXXX
# vi /root/rootCA.pem
YYYY
```

トークンを入力

証明書を入力

4. fence_kubevirt の設定 (11/11)

- 仮想マシンが fence_kubevirt を通して, Kubernetes API にアクセスできることを確認します
 - この手順は仮想マシン #1, 2 の両方で実施

```
# fence_kubevirt --namespace osc2025 -o status -n rhel9-ha-cluster-node02
```

```
Status: ON
```

仮想マシン #2 の起動状態を確認

```
# fence_kubevirt --namespace osc2025 -o off -n rhel9-ha-cluster-node02  
--disable-timeout=1
```

```
Success: Powered OFF
```

仮想マシン #2 を停止

```
# fence_kubevirt --namespace osc2025 -o status -n rhel9-ha-cluster-node02
```

```
Status: OFF
```

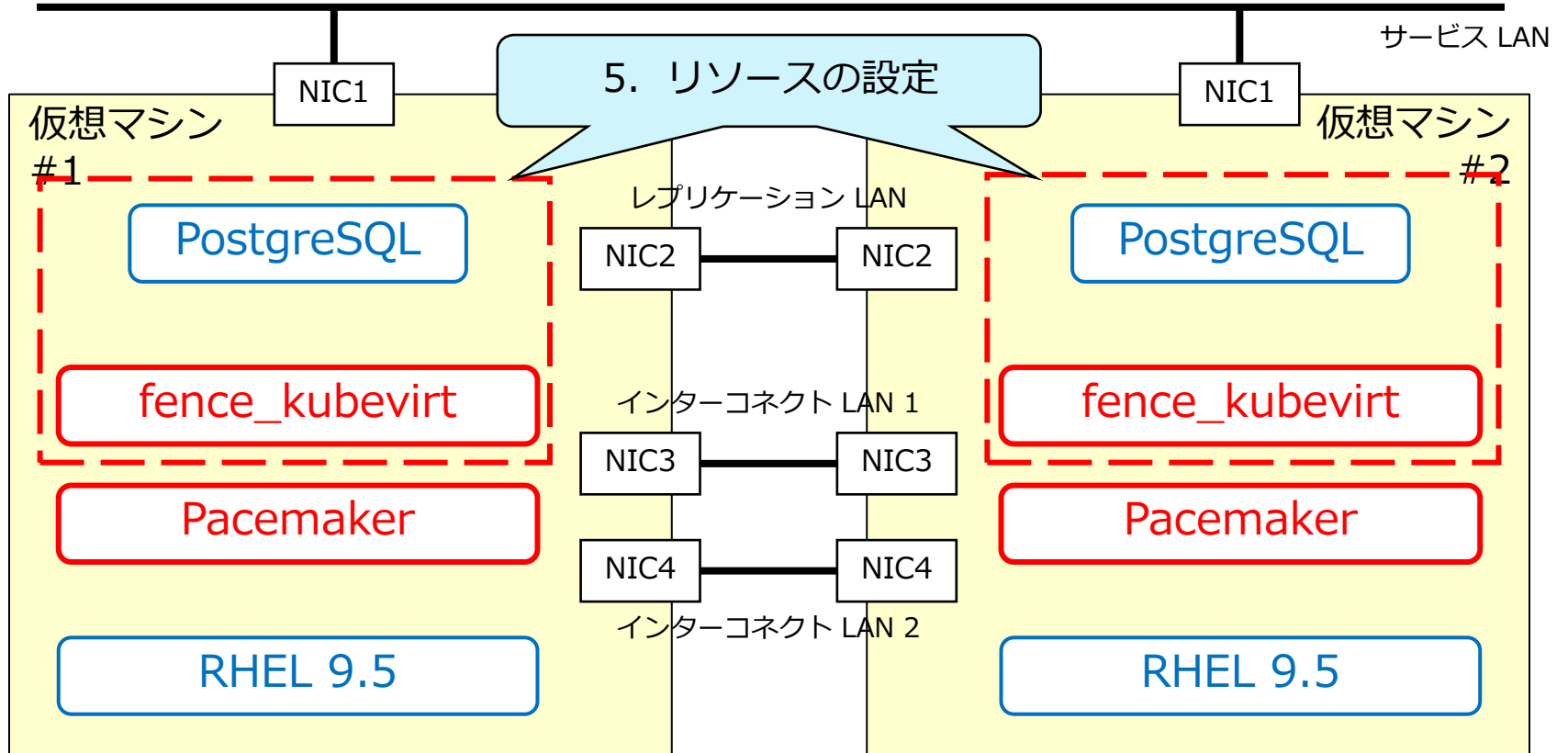
仮想マシン #2 の停止状態を確認

Pacemaker を使用するまでの流れ

1. 仮想マシンの作成
2. OCP-V 環境に NW を追加
3. Pacemaker の設定
4. fence_kubevirt の設定
5. リソースの設定
6. リソース故障のお試し！

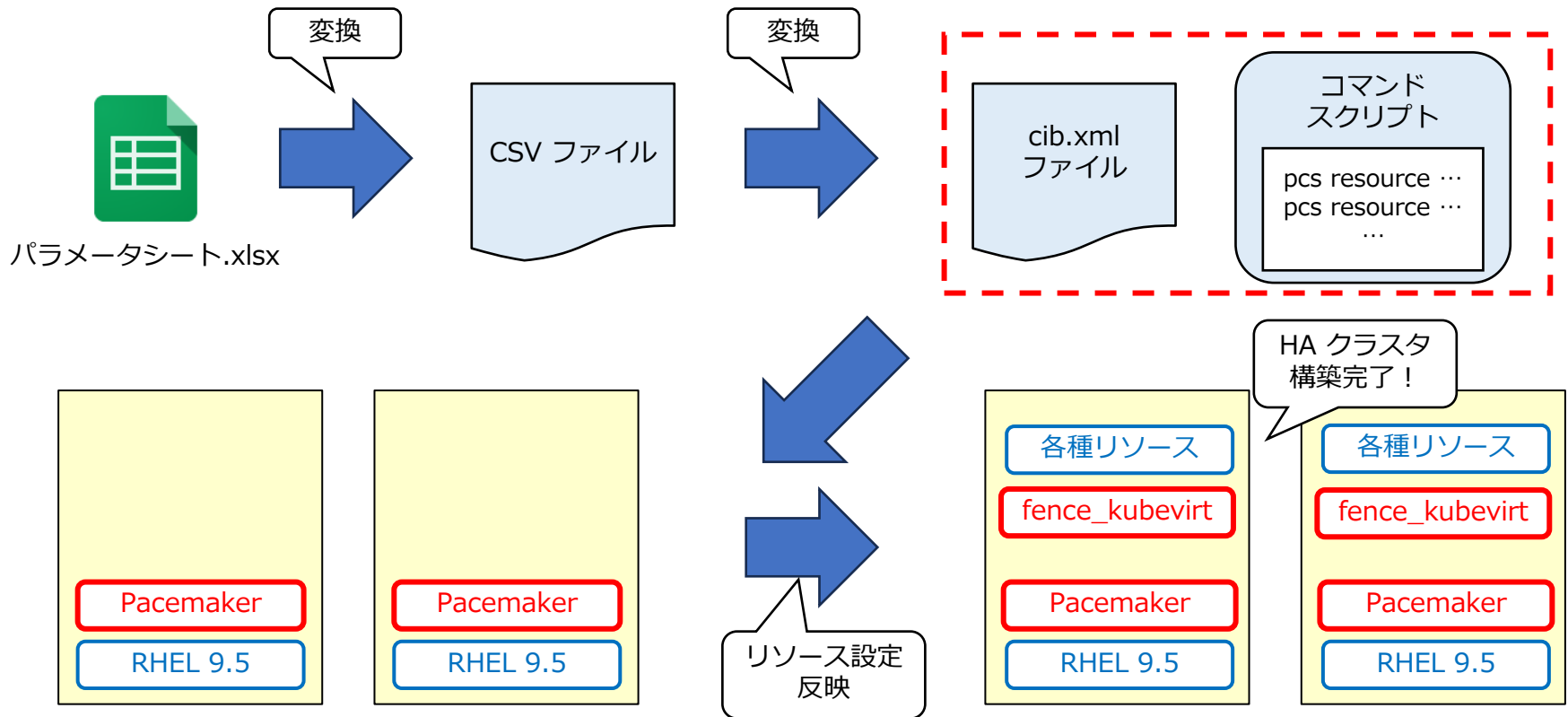
※ OCP-V 環境ならではの設定を中心に紹介します

Pacemaker を試す HA クラスタ構成



5. リソースの設定 (1/6)

- pm_extra_tools を利用して, Excel 形式で記載したパラメータシートをクラスタに入力する形式に変換します
 - pm_extra_tools : Linux-HA Japan が提供している Pacemaker の構築・運用を補助するツール



5. リソースの設定 (2/6)

- pm_extra_tools をインストールします
 - この手順は仮想マシン #1, 2 の**どちらか**で実施
 - pm_extra_tools は以下のサイトから入手可能
[Releases · linux-ha-japan/pm_extra_tools](#)

```
# dnf install pm_extra_tools-1.6-1.el9.noarch.rpm
```

```
# ls /usr/share/pm_extra_tools/
```

```
pm_pcsген.conf  pm_pcsген.py  pm_pcsген_sample.xlsx
```

パラメータシートのサンプル

5. リソースの設定 (3/6)

- パラメータシートにリソースの各種パラメータを設定します
- 抜粋して fence_kubevirt のリソース設定を例を紹介

#表 2-1 クラスタ設定 ... クラスタ・プロパティ

PROPERTY			
#	項目	設定内容	説明
	priority-fencing-delay	240s	fencing実行遅延時間
	node-health-strategy	only-green	ノード正常性属性
	pe-input-series-max	0	正常状態遷移ファイルの履歴数
	pe-warn-series-max	0	エラー発生状態遷移ファイルの履歴数
	pe-error-series-max	0	警告発生状態遷移ファイルの履歴数
	stonith-max-attempts	600	STONITHが失敗した際のトライ回数

priority-fencing-delay は **240s** 以上に設定してください

#表 8-1-1 クラスタ設定 ... STONITHリソース (id=fence1-kube_virt)

STONITH				
#	P id	type	概要	
	STONITHリソースID	type		
	fence1-kubevirt	fence_kubevirt	fence_kubevirtプラグイン(STONITHプラグイン)*A	
#	A type	name	value	概要
	パラメータ種別	項目	設定内容	
	options	pcmk_host_map	rhel9-ha-cluster-node01:rhel9-ha-cluster-node01	STONITH対象Masterノードのゲスト上のホスト名とOCP管理上の仮想マシン名 *A
		pcmk_reboot_action	off	状態不明ノードをフェンシングする際の動作*A
		namespace	osc2025	
#	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

仮想マシン #1 の fence_kubevirt 設定
※仮想マシン #2 も同様に設定

5. リソースの設定 (4/6)

- 作成したパラメータシートを CSV 形式に変換し、仮想マシンに転送します
- pm_extra_tools を用いて、CSV 形式のパラメータシートから pcs コマンドスクリプトとリソース定義ファイルを生成します
 - この手順は pm_extra_tools をインストールした仮想マシンで実施

```
# pm_pcsgen pm_pcsgen_env.csv  
pm_pcsgen_env.xml (CIB), pm_pcsgen_env.sh (PCS) を出力しました。
```

リソース定義ファイル

pcs コマンドスクリプト

5. リソースの設定 (5/6)

- クラスタを作成し, リソース設定を反映させます
 - 先ほどまでの手順を行った仮想マシンで実施

```
# pcs cluster setup test_cluster rhel9-ha-cluster-node01 ¥  
  addr=192.168.50.10 addr=192.168.60.10 rhel9-ha-cluster-node02 ¥  
  addr=192.168.50.20 addr=192.168.60.20
```

～略～

Cluster has been successfully set up.

```
# pcs cluster start  
Starting Cluster...
```

Pacemaker を起動し,
Active ノードとしてクラスタに参加

```
# pcs cluster cib-push pm_pcsgen_env.xml  
CIB updated
```

クラスタにリソース設定を反映

- もう一方の仮想マシンでも Pacemaker を起動します

```
# pcs cluster start  
Starting Cluster...
```


5. リソースの設定 (6/6)

- 設定が正常に完了した場合は以下のような状態になります
 - この手順は仮想マシン #1, 2 の両方で確認可能

```
# pcs status --full
~略~
Node List:
 * Node rhel9-ha-cluster-node01 (1): online, feature set 3.19.5
 * Node rhel9-ha-cluster-node02 (2): online, feature set 3.19.5

Full List of Resources:
 * Clone Set: pgsql-clone [pgsql] (promotable):
   * pgsql (ocf:linuxhajp:pgsql): Promoted rhel9-ha-cluster-node01
   * pgsql (ocf:linuxhajp:pgsql): Starting rhel9-ha-cluster-node02
~略~
```

設定したノード数やリソースの内容によって、
クラスタのステータス表示は変化

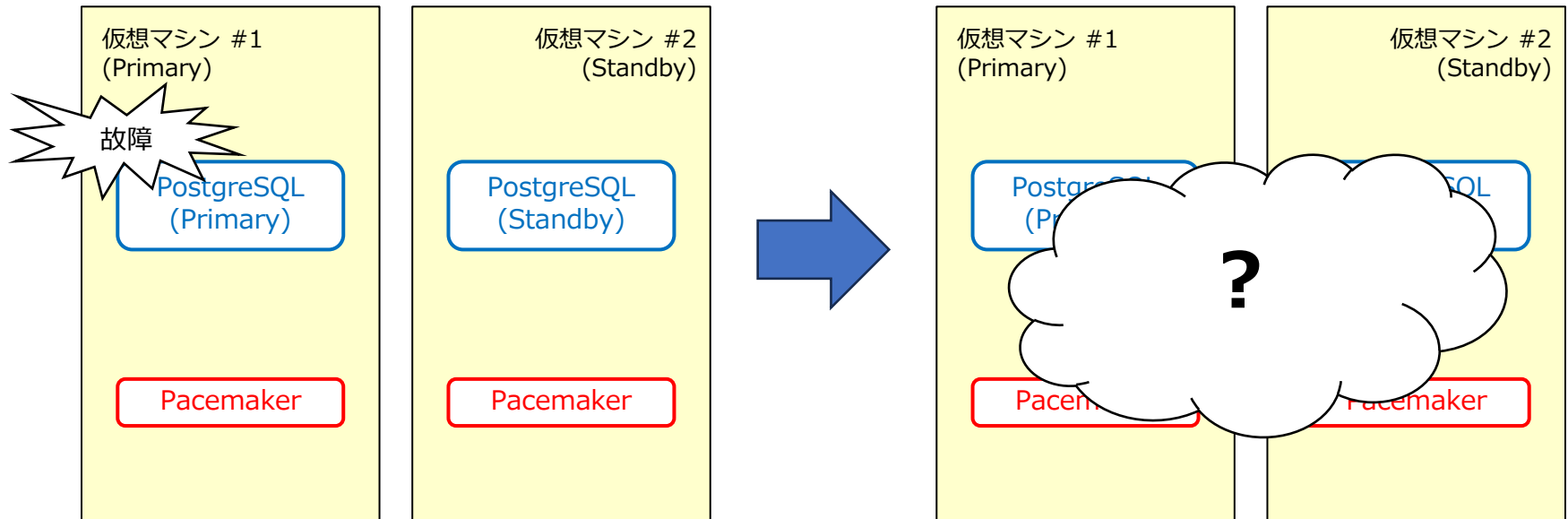
Pacemaker を使用するまでの流れ

1. 仮想マシンの作成
2. OCP-V 環境に NW を追加
3. Pacemaker の設定
4. fence_kubevirt の設定
5. リソースの設定
6. リソース故障のお試し！

※ OCP-V 環境ならではの設定を中心に紹介します

6. リソース故障のお試し！ (1/3)

- それでは実際にリソース故障時の動きを確認してみます
 - Pacemaker が未稼働/稼働の状態を比較
 - 期待動作は 仮想マシン #1 の PostgreSQL が故障した際に 仮想マシン #2 の PostgreSQL が Primary に昇格



6. リソース故障のお試し！ (2/3)

- Pacemaker なしの場合は自動でサービスの継続はできません
 - 仮想マシン #2 の PostgreSQL を Primary に昇格させる必要あり

```
postgres@rhel9-ha-cluster-node01 ~]$ pg_ctl start
サーバーの起動完了を待っています...2025-09-29 14:46:10.560 JST [2518168] LOG: redirecting log output to logging collector process
2025-09-29 14:46:10.560 JST [2518168] HINT: Future log output will appear in directory "log".
完了
サーバー起動完了
postgres@rhel9-ha-cluster-node01 ~]$ psql -c "SELECT pg_is_in_recovery();"
pg_is_in_recovery
-----
f
(1 行)

postgres@rhel9-ha-cluster-node01 ~]$
postgres@rhel9-ha-cluster-node01 ~]$ kill -9 postgres
postgres@rhel9-ha-cluster-node01 ~]$
postgres@rhel9-ha-cluster-node01 ~]$ psql -c "SELECT pg_is_in_recovery();"
psql: エラー: ソケット"/run/postgresql/.s.PGSQL.5432"のサーバーへの接続に失敗しました: 接続を拒否されました
サーバーはローカルで稼働していてそのソケットで接続を受け付けていますか?
postgres@rhel9-ha-cluster-node01 ~]$

postgres@rhel9-ha-cluster-node02 ~]$ pg_ctl start
サーバーの起動完了を待っています...2025-09-29 14:46:11.487 JST [599555] LOG: redirecting log output to logging collector process
2025-09-29 14:46:11.487 JST [599555] HINT: Future log output will appear in directory "log".
完了
サーバー起動完了
postgres@rhel9-ha-cluster-node02 ~]$ psql -c "SELECT pg_is_in_recovery();"
pg_is_in_recovery
-----
t
(1 行)

postgres@rhel9-ha-cluster-node02 ~]$
postgres@rhel9-ha-cluster-node02 ~]$ psql -c "SELECT pg_is_in_recovery();"
pg_is_in_recovery
-----
t
(1 行)

postgres@rhel9-ha-cluster-node02 ~]$
```

pg_is_in_recovery() の返り値

- f (false): Primary で稼働中
- t (ture): Standbyで稼働中

仮想マシン #1 の PostgreSQL を故障させても
仮想マシン #2 の PostgreSQL は

- t (ture): Standbyで稼働中

となっているので、Primary に昇格しておらず

6. リソース故障のお試し！ (3/3)

- Pacemaker ありの場合は自動でサービスの継続ができます
 - 仮想マシン #2 の PostgreSQL が自動で Primary に昇格

```
root@rhel9-ha-cluster-node01 ~# pcs resource
* Clone Set: postgresql-clone [postgresql] (promotable):
  * Promoted: [ rhel9-ha-cluster-node01 ]
  * Unpromoted: [ rhel9-ha-cluster-node02 ]
* Resource Group: primary-group:
  * ipaddr-primary (ocf:heartbeat:IPaddr2): Started rhel9-ha-cluster-node01
  * ipaddr-replication (ocf:heartbeat:IPaddr2): Started rhel9-ha-cluster-node01
* Clone Set: ping-clone [ping]:
  * Started: [ rhel9-ha-cluster-node01 rhel9-ha-cluster-node02 ]
* Clone Set: storage-mon-clone [storage-mon]:
  * Started: [ rhel9-ha-cluster-node01 rhel9-ha-cluster-node02 ]
root@rhel9-ha-cluster-node01 ~#
root@rhel9-ha-cluster-node01 ~# pkill -9 postgres
root@rhel9-ha-cluster-node01 ~#
root@rhel9-ha-cluster-node01 ~# pcs resource
* Clone Set: postgresql-clone [postgresql] (promotable):
  * postgresql (ocf:linuxha:postgresql): Promoting rhel9-ha-cluster-node02
  * Stopped: [ rhel9-ha-cluster-node01 ]
* Resource Group: primary-group:
  * ipaddr-primary (ocf:heartbeat:IPaddr2): Stopped
  * ipaddr-replication (ocf:heartbeat:IPaddr2): Stopped
* Clone Set: ping-clone [ping]:
  * Started: [ rhel9-ha-cluster-node01 rhel9-ha-cluster-node02 ]
* Clone Set: storage-mon-clone [storage-mon]:
  * Started: [ rhel9-ha-cluster-node01 rhel9-ha-cluster-node02 ]
root@rhel9-ha-cluster-node01 ~#
```

pcs resource コマンドのパラメータ

- Promoted: Primary で稼働中
- Unpromoted: Standbyで稼働中

仮想マシン #1 の PostgreSQL を故障させると
仮想マシン #2 の PostgreSQL は

- Promoted: Primary で稼働中

となっているので、Primary に昇格している！

おわりに

おわりに

- OCP-V に Pacemaker を組み合わせることで、より可用性の高いクラスタを構築する方法を紹介しました
- Pacemaker を利用するには、OCP-V 周りの設定が非常に多くなっています
- ただ、上記の設定さえ完了すれば物理サーバ上と同じように Pacemaker を利用できるのです、ぜひ活用してみてください！

より Pacemaker について知りたい方はこちらに
アクセスしてみてください

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

従来のメーリングリストに代わり、
9/18 より新たな情報交換の場もオープンしています！

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



本発表で詳しく紹介できなかったリソース
の設定についても紹介されているので、
気になる方はぜひ見てみてください



今後も Linux-HA Japan を
よろしくお願いします

