



Red Hat OpenStack Platform 10

コンピューティングインスタンスの高可用性

コンピューティングインスタンスの高可用性設定

Red Hat OpenStack Platform 10 コンピュートインスタンスの高可用性

コンピュートインスタンスの高可用性設定

OpenStack Team

rhos-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Red Hat OpenStack Platform でコンピュートインスタンスの高可用性 (インスタンス HA) を構成するためのガイドです。本書では、**Ansible** を使用したインスタンス HA の有効化を中心に記載しています。

目次

第1章 概要	3
第2章 環境の要件および前提条件	4
2.1. 共有ストレージの例外	4
第3章 デプロイメント	6
3.1. 必要な ANSIBLE 設定ファイルの作成	6
3.2. アンダークラウドの準備	7
3.3. インスタンス HA の有効化	8
第4章 テスト	10
第5章 ロールバック	11
付録A インスタンス HA を使用した退避の自動化	12
付録B ANSIBLE PLAYBOOK によって自動化される手動の手順	13

第1章 概要

本ガイドでは、**インスタンスの高可用性 (インスタンス HA)**の実装方法について説明します。インスタンス HA により、Red Hat OpenStack Platform は、インスタンスをホストしているコンピューットノードで障害が発生した場合に、それらのインスタンスを異なるコンピューットノードで自動的に再起動することができます。

インスタンス HA により、ホストのコンピューットノードで障害が発生した場合にはいつでも、インスタンスの退避が自動化されます。インスタンス HA によってトリガーされる退避のプロセスは、「[インスタンスの退避](#)」に記載されている、ユーザーが手動で行うことのできるプロセスと同様です。インスタンス HA は共有ストレージとローカルストレージの環境で機能します。これは、退避されるインスタンスがゼロから起動される場合でも、新しいホスト内で同じネットワーク設定 (静的 IP、Floating IP など) と特性が維持されることを意味します。

インスタンス HA は次の 3 つのリソースエージェントによって管理されます。

エージェント名	クラスター内の名前	ロール
fence_compute	fence-nova	ノードが利用不可になった場合に退避用のコンピューットノードをマークします。
NovaEvacuate	nova-evacuate	障害が発生したノードからインスタンスを退避して、コンピューットノードの中の 1 台で実行します。
nova-compute-wait	nova-compute-checkevacuate	正常に機能するコンピューットホストにインスタンスが退避された後には、そのインスタンスで Compute サービスを再起動します。

本ガイドでは、**Ansible** を使用した、オーバークラウド上でのインスタンス HA の有効化を中心として説明します。本書では、このプロセスを効率化するために、必要な **Playbook** が含まれた、事前パッケージ済みの TAR アーカイブも使用しています。



注記

インスタンス HA によって実行される検出と退避のプロセスについての簡単な説明は、「[付録A インスタンス HA を使用した退避の自動化](#)」を参照してください。

第2章 環境の要件および前提条件

インスタンス HA を有効化するには、Red Hat OpenStack Platform のオーバークラウドは以下の要件を満たす必要があります。

- Red Hat OpenStack Platform director を使用して環境がデプロイされていること。詳しくは、『[director のインストールと使用方法](#)』を参照してください。
- コントロールプレーンでフェンシングが手動で有効化されていること
- 以下のパッケージが全ノードにインストールされていること
 - **fence-agents-4.0.11-66.el7_4.3** (以降)
 - **pacemaker-1.1.16-12.el7.x86_64** (以降)
 - **resource-agents-3.9.5-105.el7.x86_64** (以降)
- コンピュートおよびコントロールプレーンの両方の完全な停止に対応できる環境であること
- 一時ストレージおよびブロックストレージの共有ストレージが環境内で有効になっていること。関連情報は、『[共有ストレージの例外](#)』を参照してください。
- メッセージブローカー (AMQP) が各コンピュートノードのホスト名を有効と認識していること。コンピュートノードのホスト名を確認するには、以下のコマンドを実行します。

```
heat-admin@compute-n $ sudo crudini --get /etc/nova/nova.conf
DEFAULT host
```

- **\$OS_AUTH_URL** で設定されているエンドポイントに各コンピュートノードが到達可能であること。また、この環境変数が、以下のいずれかに設定されていること。
 - オーバークラウドの認証サービス (外部ネットワークへのアクセスが必要)
 - 内部の認証 URL



警告

インスタンス HA が有効化されている場合には、オーバークラウドのアップグレードまたはスケールアップの操作は実行できません。操作を試みても失敗します。これには、マイナーアップグレードとメジャーアップグレードの両方が含まれます。

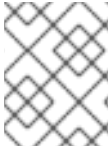
オーバークラウドのアップグレードまたはスケールアップを実行する前には、まずインスタンス HA を無効にしてください。手順については、『[5章 ロールバック](#)』を参照してください。

2.1. 共有ストレージの例外

通常、インスタンス HA を有効化するには、共有ストレージが必要です。**no-shared-storage** オプションの使用を試みると、退避中に **InvalidSharedStorage** エラーが発生する可能性が高くなります。ただし、全インスタンスが **Block Storage (cinder)** ボリュームからブートするように設定されて

いる場合には、インスタンスのディスクイメージを保管する共有ストレージは必要ないので、**no-shared-storage** オプションを使用して全インスタンスを退避することができます。

退避中に、インスタンスが **Block Storage** ボリュームから起動するように設定されている場合には、退避されるインスタンスはいずれも、同じボリュームから起動することが予想されますが、別のコンピュータノード上で起動することになります。そのため、**OS** イメージとアプリケーションデータはその **Block Storage** ボリューム上に保持されているので、退避されたインスタンスは、ジョブを即時に再起動することができます。



注記

本ガイドに記載する **Ansible** ベースのデプロイメント手順では、**no-shared-storage** オプションがサポートされています。

第3章 デプロイメント

以下の手順では、**Ansible** を使用してインスタンス HA を有効にします。**Ansible** に関する詳しい情報は、[Ansible のドキュメント](#)を参照してください。

3.1. 必要な **ANSIBLE** 設定ファイルの作成

Ansible を使用してインスタンス HA を有効にするには、**インベントリーファイル**と **SSH 引数ファイル** が必要です。両ファイルは、オーバークラウドにインスタンス HA を実装するのに必要な **Ansible** の変数を渡します。

インベントリーファイル

インベントリーファイルには、**Ansible Playbook** 用の異なるターゲットホストが記載されます。このファイルは 2 つのセクションに分かれ、最初のセクションには各ノード (名前) と共に、**Ansible** が各 **Playbook** で使用する必要のあるホスト名、ユーザー名、秘密鍵ファイルのパスがリストされます。以下に例を示します。

```
overcloud-controller-0 ansible_host=overcloud-controller-0
ansible_user=heat-admin ansible_private_key_file=/home/stack/.ssh/id_rsa
```

2 番目のセクションには、次の見出し (または **ノードの種別**) の下に各ノードがリストされます:
compute、**undercloud**、**overcloud**、**controller**

/home/stack/hosts という名前のインベントリーファイルを作成します。以下の例には、このファイルに必要な構文を示しています。

```
undercloud ansible_host=undercloud ansible_user=stack
ansible_private_key_file=/home/stack/.ssh/id_rsa
overcloud-compute-1 ansible_host=overcloud-compute-1 ansible_user=heat-
admin ansible_private_key_file=/home/stack/.ssh/id_rsa
overcloud-compute-0 ansible_host=overcloud-compute-0 ansible_user=heat-
admin ansible_private_key_file=/home/stack/.ssh/id_rsa
overcloud-controller-2 ansible_host=overcloud-controller-2
ansible_user=heat-admin ansible_private_key_file=/home/stack/.ssh/id_rsa
overcloud-controller-1 ansible_host=overcloud-controller-1
ansible_user=heat-admin ansible_private_key_file=/home/stack/.ssh/id_rsa
overcloud-controller-0 ansible_host=overcloud-controller-0
ansible_user=heat-admin ansible_private_key_file=/home/stack/.ssh/id_rsa
```

```
[compute]
overcloud-compute-1
overcloud-compute-0
```

```
[undercloud]
undercloud
```

```
[overcloud]
overcloud-compute-1
overcloud-compute-0
overcloud-controller-2
overcloud-controller-1
overcloud-controller-0
```

```
[controller]
```

```
overcloud-controller-2
overcloud-controller-1
overcloud-controller-0
```

アンダークラウドとオーバークラウドの両方で全ホストの完全なインベントリーを生成するには、以下のコマンドを実行します。

```
stack@director $ tripleo-ansible-inventory --list
```

このコマンドにより、詳細な最新のインベントリーが JSON 形式で生成されます。詳しい情報は、「[Running Ansible Automation](#)」を参照してください。

SSH 引数ファイル

SSH 引数ファイルは、Ansible が各ターゲットホストで Playbook を実行するのに必要な認証情報と認証設定を渡します。

以下のコマンドを (`/home/stack` から) 実行して、SSH 引数ファイルを生成します。

```
stack@director $ cat /home/stack/.ssh/id_rsa.pub >>
/home/stack/.ssh/authorized_keys
stack@director $ echo -e "Host undercloud\n Hostname 127.0.0.1\n
IdentityFile /home/stack/.ssh/id_rsa\n User stack\n StrictHostKeyChecking
no\n UserKnownHostsFile=/dev/null\n" > ssh.config.ansible
/home/stack/stackrc
stack@director $ openstack server list -c Name -c Networks | awk
'/ctlplane/ {print $2, $4}' | sed s/ctlplane=//g | while read node; do
node_name=$(echo $node | cut -f 1 -d " "); node_ip=$(echo $node | cut -f 2
-d " "); echo -e "Host $node_name\n Hostname $node_ip\n IdentityFile
/home/stack/.ssh/id_rsa\n User heat-admin\n StrictHostKeyChecking no\n
UserKnownHostsFile=/dev/null\n"; done >> ssh.config.ansible
```

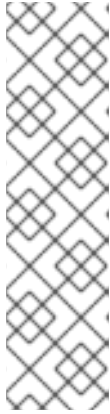
上記のコマンドを実行すると、`/home/stack/ssh.config.ansible` という名前の SSH 引数ファイルが生成されます。このファイルには、各オーバークラウドノードのホスト固有の接続オプションが含まれます。以下に例を示します。

```
Host overcloud-controller-0
  Hostname 192.168.24.11
  IdentityFile /home/stack/.ssh/id_rsa
  User heat-admin
  StrictHostKeyChecking no
  UserKnownHostsFile=/dev/null
```

3.2. アンダークラウドの準備

「[必要な Ansible 設定ファイルの作成](#)」の手順に従ってインベントリーファイルと SSH 引数ファイルを作成した後は、インスタンス HA 用のオーバークラウドを準備することができます。

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. この [TAR アーカイブ](#) を `/home/stack/` にダウンロードします。これには、Playbook、ロール、および Ansible を使用してインスタンス HA を有効化するのに必要なその他のユーティリティーが格納されています。



注記

ここで提供される [TAR アーカイブ](#) は、アップストリームの [GIT リポジトリ](#) をテストおよび修正したバージョンです。このリポジトリをクローンするには、以下のコマンドを実行します。

```
stack@director $ git clone git://github.com/redhat-
openstack/tripleo-quickstart-utils
```

このリポジトリは、通知なしに更新される可能性があるため、この手順で利用可能なアーカイブとは異なる場合があります。

3. TAR アーカイブを展開します。

```
stack@director $ tar -xvf ansible-instanceha.tar
```

4. `/home/stack/ansible.cfg` を作成して、以下の内容を記述します。

```
[defaults]
roles_path = /home/stack/ansible-instanceha/roles
```

5. `ansible.cfg`、ホスト (インベントリーファイル) と `ssh.config.ansible` (SSH 引数ファイル) を以下の環境変数にエクスポートします。

```
stack@director $ export ANSIBLE_CONFIG="/home/stack/ansible.cfg"
stack@director $ export ANSIBLE_INVENTORY="/home/stack/hosts"
stack@director $ export ANSIBLE_SSH_ARGS="-F
/home/stack/ssh.config.ansible"
```

6. オーバークラウドのノード定義テンプレート (デフォルトでは `instackenv.json`) が `/home/stack/` にあることを確認します。ノード定義テンプレートに関する詳しい情報は、「[オーバークラウドへのノードの登録](#)」を参照してください。

3.3. インスタンス HA の有効化

アンダークラウドが完全に準備できたら、「[アンダークラウドの準備](#)」のステップでダウンロードおよび展開した、事前に記述済みの **Playbook** を実行することができます。これらの **Playbook** により、**STONITH** 設定ありまたはなしで、コントローラーおよびコンピューターノードにインスタンス HA を有効化することができます。「[コントローラーノードのフェンシング](#)」を参照してください。

コントローラーノードとコンピューターノードの両方でインスタンス HA を有効化して **STONITH** を設定するには、以下のコマンドを実行します。

```
stack@director $ ansible-playbook /home/stack/ansible-
instanceha/playbooks/overcloud-instance-ha.yml \
-e release="RELEASE"
```

デフォルトでは、**Playbook** により、`instance-ha` ソリューションは共有ストレージが有効化された状態でインストールされます。お使いのオーバークラウドで共有ストレージを使用しない場合は、`instance_ha_shared_storage=false` オプションを使用してください。

```
stack@director $ ansible-playbook /home/stack/ansible-
instanceha/playbooks/overcloud-instance-ha.yml \
```

```
-e release="RELEASE" -e instance_ha_shared_storage=false
```



注記

インスタンス HA における共有ストレージについての詳しい情報は、「[共有ストレージの例外](#)」を参照してください。

コントローラーノードとコンピューターノードの両方で STONITH を設定せずに インスタンス HA を有効化するには、以下のコマンドを実行します。

```
stack@director $ ansible-playbook /home/stack/ansible-  
instanceha/playbooks/overcloud-instance-ha.yml \  
-e release="RELEASE" -e stonith_devices="none"
```

インスタンス HA を有効化して STONITH をコンピューターノードのみで設定するには (例: STONITH がコントローラーノードではすでに設定済みの場合など)、以下のコマンドを実行します。

```
stack@director $ ansible-playbook /home/stack/ansible-  
instanceha/playbooks/overcloud-instance-ha.yml \  
-e release="RELEASE" -e stonith_devices="computes"
```

第4章 テスト



警告

以下の手順では、コンピュートノードを意図的にクラッシュさせる必要があります。これにより、インスタンス HA を使用したインスタンスの自動退避が強制されます。

1. オーバークラウド上でインスタンスを1つまたは複数起動し、それらのインスタンスをホストしているコンピュートノードをクラッシュさせます。

```
stack@director $ . overcloudrc
stack@director $ nova boot --image cirros --flavor 2 test-failover
stack@director $ nova list --fields name,status,host
```

2. 上記のステップで起動したインスタンスをホストしているコンピュートノード (**compute-n**) にログインします。

```
stack@director $ . stackrc
stack@director $ ssh -lheat-admin compute-n
heat-admin@compute-n $
```

3. ノードをクラッシュさせます。

```
heat-admin@compute-n $ echo c > /proc/sysrq-trigger
```

4. 数分後に、これらのインスタンスがオンラインのコンピュートノードで再起動されたことを確認します。以下のコマンドでチェックしてください。

```
stack@director $ nova list --fields name,status,host
stack@director $ nova service-list
```

第5章 ロールバック

インスタンス HA が有効化されている場合には、アップグレードまたはスケールアップの操作は実行できません。操作を試みても失敗します。これには、マイナーアップグレードとメジャーアップグレードの両方が含まれます。オーバークラウドのアップグレードまたはスケーリングを実行する前には、最初にインスタンス HA を無効にしてください。

インスタンス HA を無効にするには、アンダークラウドで **stack** ユーザーとして以下のコマンドを実行します。

```
stack@director $ ansible-playbook /home/stack/ansible-  
instanceha/playbooks/overcloud-instance-ha.yml \  
-e release="RELEASE" -e instance_ha_action="uninstall"
```

インスタンス HA を有効化する際に **stonith_devices** オプションを使用した場合には、ロールバック中に同じオプションを指定する必要があります。たとえば、インスタンス HA の設定で **STONITH** を無効にしていた場合には(「[インスタンス HA の有効化](#)」)、以下のコマンドを使用します。

```
stack@director $ ansible-playbook /home/stack/ansible-  
instanceha/playbooks/overcloud-instance-ha.yml \  
-e release="RELEASE" -e instance_ha_action="uninstall" -e  
stonith_devices="none"
```

付録A インスタンス HA を使用した退避の自動化

インスタンス HA を使用すると、OpenStack はコンピュートノードで障害が発生した場合に、そのノードからインスタンスを退避させるプロセスを自動化します。以下のプロセスは、コンピュートノードの障害発生時にトリガーされるイベントのシーケンスを説明しています。

1. コンピュートノードで障害が発生すると、**IPMI** エージェントは **第1レベルのフェンシング** を実行してノードを物理的にリセットし、電源オフの状態にします。オンラインのコンピュートからインスタンスを退避させると、データが破損したり、オーバークラウド上で全く同じインスタンスが複数実行されてしまう可能性があります。ノードの電源が切断されると、そのノードは **フェンシングされた状態** と見なされます。
2. 物理的な IPMI フェンシングの後には、**fence-nova** エージェントが **第2レベルのフェンシング** を実行して、フェンシングされたノードに **"evacuate=yes"** というクラスターのノードごとの属性をマークします。そのために、エージェントは以下のコマンドを実行します。

```
$ attrd_updater -n evacuate -A name="evacuate" host="FAILEDHOST" value="yes"
```

FAILEDHOST は、障害の発生したコンピュートノードのホスト名です。

3. **nova-evacuate** エージェントはバックグラウンドで常時実行され、クラスターに **"evacuate=yes"** 属性のノードがあるかどうかをチェックします。**nova-evacuate** が、フェンシングされたノードにこの属性が付いていることを確認すると、エージェントは、[「インスタンスの退避」](#)に記載されている方法と同じプロセスを使用してノードの退避を開始します。
4. その一方で、障害が発生したノードが IPMI リセットによって起動している間には、**nova-compute-checkevacuate** エージェントは待ち時間 (デフォルトでは 120 秒) が経過してから、**nova-evacuate** が退避を終了したかどうかをチェックします。終了していない場合には、同じ時間間隔で再チェックします。
5. インスタンスが完全に退避されたことを **nova-compute-checkevacuate** エージェントが確認した後は、フェンシングされたノードが再びインスタンスをホストできるようにする別のプロセスがトリガーされます。

付録B ANSIBLE PLAYBOOK によって自動化される手動の手順

本書に記載している Ansible ベースのソリューションは、サポートされている方法で、インスタンス HA の手動設定手順を自動化するように設計されています。以下の付録には、参考情報として、このソリューションにより自動化されるステップを記載します。

1. 各コンピュータノード上で **Compute** サービスを停止し、無効化します。

```
heat-admin@compute-n $ sudo systemctl stop openstack-nova-compute
heat-admin@compute-n $ sudo systemctl disable openstack-nova-compute
```

2. **pacemaker-remote** に使用する認証キーを作成します。**director** ノードでこのステップを実行します。

```
stack@director # dd if=/dev/urandom of=~/.authkey bs=4096 count=1
```

3. このキーをコンピュータノードとコントローラーノードにコピーします。

```
stack@director # scp authkey heat-admin@node-n:~/
stack@director # ssh heat-admin@node-n:~/
heat-admin@node-n $ sudo mkdir -p --mode=0750 /etc/pacemaker
heat-admin@node-n $ sudo chgrp haclient /etc/pacemaker
heat-admin@node-n $ sudo mv authkey /etc/pacemaker/
heat-admin@node-n $ sudo chown root:haclient /etc/pacemaker/authkey
```

4. コンピュータノードで **pacemaker-remote** サービスを有効化し、それに応じてファイアウォールを設定します。

```
heat-admin@compute-n $ sudo systemctl enable pacemaker_remote
heat-admin@compute-n $ sudo systemctl start pacemaker_remote
heat-admin@compute-n $ sudo iptables -I INPUT 11 -p tcp --dport 3121 -j ACCEPT ; /sbin/service iptables save
```

5. コントローラーノードとコンピュータノードに **pacemaker (1.1.12-22.el7_1.4.x86_64)** および **resource-agents (3.9.5-40.el7_1.5.x86_64)** パッケージの必要なバージョンがインストールされていることを確認します。

```
heat-admin@controller-n $ sudo rpm -qa | egrep
\'(pacemaker|resource-agents)\'
```

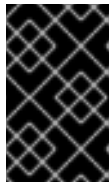
6. **overcloudrc** ファイルを使用して **NovaEvacuate Active/Passive** リソースを作成し、**auth_url**、**username**、**tenant**、**password** の値を指定します。

```
stack@director # scp overcloudrc heat-admin@controller-1:~/
heat-admin@controller-1 $ . ~/overcloudrc
heat-admin@controller-1 $ sudo pcs resource create nova-evacuate
ocf:openstack:NovaEvacuate auth_url=$OS_AUTH_URL
username=$OS_USERNAME password=$OS_PASSWORD
tenant_name=$OS_TENANT_NAME
```



注記

共有ストレージを使用していない場合には、**no_shared_storage=1** オプションを追加してください。詳しい説明は、「[共有ストレージの例外](#)」を参照してください。



重要

「[2章 環境の要件および前提条件](#)」で前述したように、**\$OS_AUTH_URL** は各コンピュータノードから到達可能である必要があります。この環境変数には、オーバークラウドの認証サービスか、内部の認証 URL を指定する必要があります。

7. **nova-evacuate** が Floating IP リソース、Image サービス (glance)、OpenStack Networking サービス (neutron)、Compute サービス (nova) の後に起動されることを確認します。

```
heat-admin@controller-1 $ for i in $(sudo pcs status | grep IP | awk
\{'{ print $1 }'\}); do sudo pcs constraint order start $i then nova-
evacuate ; done
```

8. **cibadmin** データを使用して現在のコントローラー一覧を作成します。

```
heat-admin@controller-1 $ controllers=$(sudo cibadmin -Q -o nodes |
grep uname | sed s/.*uname../ | awk -F" \" '{print $1}')
```

```
heat-admin@controller-1 $ echo $controllers
```

9. この一覧を使用して、それらのノードを **osprole=controller** プロパティでコントローラーとしてタグ付けします。

```
heat-admin@controller-1 $ for controller in ${controllers}; do sudo
pcs property set --node ${controller} osprole=controller ; done
```

```
heat-admin@controller-1 $ sudo pcs property
```

新たに割り当てられたロールは、**Node attributes** のセクション下に表示されるはずです。

10. 環境内にすでに存在する **stonith** デバイスの一覧を作成します。

```
heat-admin@controller-1 $ stonithdevs=$(sudo pcs stonith | awk
\{'{print $1}')
```

```
heat-admin@controller-1 $ echo $stonithdevs
```

11. コントロールプレーンサービスをタグ付けし、一覧内の **stonith** デバイスをスキップして、上記で特定したコントローラーのみで実行されるようにします。

```
heat-admin@controller-1 $ for i in $(sudo cibadmin -Q --xpath
//primitive --node-path | tr ' ' '\n' | awk -F "id=\"" '{print $2}'
| awk -F "\"" '{print $1}' | uniq); do
    found=0
    if [ -n "$stonithdevs" ]; then
        for x in $stonithdevs; do
            if [ $x = $i ]; then
                found=1
            fi
        done
    fi
done
```

```

    if [ $found = 0 ]; then
        sudo pcs constraint location $i rule resource-
discovery=exclusive score=0 osprole eq controller
    fi
done

```

12. nova-compute リソースを **pacemaker** 内に設定します。

```

heat-admin@controller-1 $ . /home/heat-admin/overcloudrc
heat-admin@controller-1 $ sudo pcs resource create nova-compute-
checkevacuate ocf:openstack:nova-compute-wait auth_url=$OS_AUTH_URL
username=$OS_USERNAME password=$OS_PASSWORD
tenant_name=$OS_TENANT_NAME domain=localdomain op start timeout=300
--clone interleave=true --disabled --force

```



注記

このコマンドは、デフォルトのクラウドドメイン名 **localdomain** を使用することを前提としています。カスタムのクラウドドメイン名を使用する場合には、**domain=** パラメーターの値として設定してください。



重要

「[2章 環境の要件および前提条件](#)」で前述したように、**\$OS_AUTH_URL** は各コンピュートノードから到達可能である必要があります。この環境変数には、オーバークラウドの認証サービスか、内部の認証 URL を指定する必要があります。

```

heat-admin@controller-1 $ sudo pcs constraint location nova-compute-
checkevacuate-clone rule resource-discovery=exclusive score=0
osprole eq compute
heat-admin@controller-1 $ sudo pcs resource create nova-compute
systemd:openstack-nova-compute op start timeout=60s --clone
interleave=true --disabled --force
heat-admin@controller-1 $ sudo pcs constraint location nova-compute-
clone rule resource-discovery=exclusive score=0 osprole eq compute
heat-admin@controller-1 $ sudo pcs constraint order start nova-
compute-checkevacuate-clone then nova-compute-clone require-all=true
heat-admin@controller-1 $ sudo pcs constraint order start nova-
compute-clone then nova-evacuate require-all=false

```

13. コンピュートノード用に **stonith** デバイスを追加します。各コンピュートノードで以下のコマンドを実行します。

```

heat-admin@controller-1 $ sudo pcs stonith create ipmilan-overcloud-
compute-N fence_ipmilan pcmk_host_list=overcloud-compute-0
ipaddr=10.35.160.78 login=IPMILANUSER passwd=IPMILANPW lanplus=1
cipher=1 op monitor interval=60s;

```

ここで、

- **N** は各コンピュートノードを識別するための番号に置き換えます (例:**ipmilan-overcloud-compute-1**、**ipmilan-overcloud-compute-2** など)。

- **IPMILANUSER** および **IPMILANPW** は、IPMI デバイスのユーザー名とパスワードに置き換えます。

14. 別の **fence-nova stonith** デバイスを作成します。

```
heat-admin@controller-1 $ . overcloudrc
heat-admin@controller-1 $ sudo pcs stonith create fence-nova
fence_compute \
                                auth-url=$OS_AUTH_URL \
                                login=$OS_USERNAME \
                                passwd=$OS_PASSWORD \
                                tenant-name=$OS_TENANT_NAME \
                                record-only=1 --force \
                                domain=localdomain
```



注記

このコマンドは、デフォルトのクラウドドメイン名 **localdomain** を使用することを前提としています。カスタムのクラウドドメイン名を使用する場合には、**domain=** パラメーターの値として設定してください。

共有ストレージを使用していない場合には、**no_shared_storage=1** オプションを追加してください。詳しい説明は、「[共有ストレージの例外](#)」を参照してください。

15. **fence-nova** に必要な制約を設定します。

```
heat-admin@controller-1 $ sudo pcs constraint location fence-nova
rule resource-discovery=never score=0 osprole eq controller
heat-admin@controller-1 $ sudo pcs constraint order promote galera-
master then fence-nova require-all=false
heat-admin@controller-1 $ sudo pcs constraint order start fence-nova
then nova-compute-clone
```

16. コンピュートノードがフェンシング後に回復できるようにします。

```
heat-admin@controller-1 $ sudo pcs property set cluster-recheck-
interval=1min
```

17. コンピュートノードのリソースを作成して、**stonith level 1** にノードの物理フェンスデバイスと **fence-nova** の両方が含まれるように設定します。そのためには、各コンピュートノードに対して以下のコマンドを実行します。

```
heat-admin@controller-1 $ sudo pcs resource create overcloud-
compute-N ocf:pacemaker:remote reconnect_interval=60 op monitor
interval=20
heat-admin@controller-1 $ sudo pcs property set --node overcloud-
compute-N osprole=compute
heat-admin@controller-1 $ sudo pcs stonith level add 1 overcloud-
compute-N ipmilan-overcloud-compute-N,fence-nova
heat-admin@controller-1 $ sudo pcs stonith
```

N は、各コンピュートノードを識別するための番号に置き換えます (例:**overcloud-compute-1**、**overcloud-compute-2** など)。これらの識別番号を使用して、前のステップで

作成した stonith デバイスに各コンピュータノードを照合します (例: **overcloud-compute-1** および **ipmilan-overcloud-compute-1**)。

環境が安定するまでしばらく待ってから、失敗したリソースをクリーンアップします。

```
heat-admin@controller-1 $ sleep 60
heat-admin@controller-1 $ sudo pcs resource cleanup
heat-admin@controller-1 $ sudo pcs status
heat-admin@controller-1 $ sudo pcs property set stonith-enabled=true
```