Red Hat OpenStack Platform 13

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の設定

Last Updated: 2021-07-03

Red Hat OpenStack Platform 13 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の設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

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、director 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

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

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)** を管理する方法について説明します。インスタンス HA が設定された Red Hat OpenStack Platform では、ホストコンピュートノードに障害が発生した場合に、インスタンスを自動的に退避させ別のコンピュートノードに作成し直すことができます。

インスタンス HA がトリガーとなる退避プロセスは、ユーザーが手動で実施できる操作 (詳細は『インスタンス＆イメージガイド』の「インスタンスの退避」を参照) と類似しています。

インスタンス HAは、共有ストレージまたはローカルストレージ環境で機能します。したがって、インスタンスを新たに作成した場合でも、退避したインスタンスは新しいホスト内で同じネットワーク設定 (静的 IP、Floating IP 等) および同じ特性を維持します。

インスタンス HA は以下のリソースエージェントにより管理されます。

エージェント名	クラスター内での名前	ロール
fence_compute	**fence-nova**	コンピュートノードが使用不能になった場合に、退避のためにそのノードをマーキングする。
NovaEvacuate	**nova-evacuate**	障害が発生したノードからインスタンスを退避させる。このエージェントは、コントローラーノードのいずれかで動作します。
Dummy	**compute-unfence-trigger**	フェンシングされたノードを解放し、再びインスタンスを実行できるようにする。

第2章 インスタンス HA の仕組み

OpenStack では、インスタンス HA を使用して、障害が発生したコンピュートノードからのインスタンス退避プロセスを自動化しています。コンピュートノードの障害がトリガーとなり実施されるイベントのシーケンスを、以下の手順で説明します。

1. 障害が発生すると、**IPMI** エージェントが **第一レイヤーのフェンシング** を実施し、確実に電源オフの状態にするためにノードを物理的にリセットします。オンライン状態のコンピュートノード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複数の同一インスタンスが実行されたりする場合があります。ノードの電源がオフである場合、**フェンシング済み** とみなされます。

2. IPMI による物理的なフェンシングの後に、**fence-nova** エージェントが **第二レイヤーのフェンシング** を実施し、フェンシング済みのノードを **“evacuate=yes”** 属性でマーキングします (この属性は、クラスターのノードごとに設定する必要があります)。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"
   ```

   ここで、**FAILEDHOST** は障害が発生したコンピュートノードのホスト名です。

3. **nova-evacuate** エージェントは常にバックグラウンドで動作し、クラスターに **“evacuate=yes”** 属性のノードがないか定期的に確認します。この属性がフェンシング済みノードに含まれることを **nova-evacuate** が検出すると、エージェントはノードからの全インスタンスの退避を開始します (退避のプロセスについては「インスタンスの退避」を参照)。

4. 障害が発生したノードが IPMI によるリセットから復帰する間、そのノードの **nova-compute** プロセスが自動的に開始されます。ノードは前のステップでフェンシングされているので、Pacemaker が **フェンシングを解除する** まで新しいインスタンスを実行しません。

5. コンピュートノードが再びオンライン状態にあることを Pacemaker が把握すると、ノードの **compute-unfence-trigger** リソースの起動を試み、force-down API の呼び出しを取り消してノードを再び有効な状態に設定します。

2.1. 退避させる特定インスタンスの指定

デフォルトでは、すべてのインスタンスを退避させますが、退避用のイメージまたはフレーバーをタグ付けすることもできます。

イメージをタグ付けするには、以下のコマンドを実行します。

```
$ openstack image set --tag evacuable ID-OF-THE-IMAGE
```

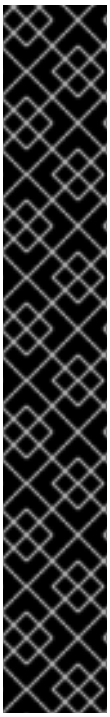
フレーバーをタグ付けするには、以下のコマンドを実行します。

```
$ nova flavor-key ID-OF-THE-FLAVOR set evacuable=true
```

第3章 インスタンス HA のインストールと設定

インスタンス HA は director でデプロイおよび設定されます。ただし、デプロイメントの準備のために、いくつかの追加ステップを実施する必要があります。

本項では、カスタムロールを使用してコンピュートノードのサブセットでインスタンス HA を有効化することを目的に、新しいオーバークラウドで新しいインスタンス HA デプロイメントを設定するために必要なすべての手順について説明します。



重要

- 標準のロールまたはカスタムロールを使用する既存のオーバークラウド等、別の環境でインスタンス HA を有効化する場合には、デプロイメントに関連する手順のみに従い、テンプレートを適切に変更します。
- コンピュートノードのホスト名および Pacemaker リモートリソース名は、W3C 命名規則に従う必要があります。詳細は、W3C ドキュメントの Declaring Namespaces および Names and Tokens を参照してください。
- インスタンス HA では、コンピュートノードがコントローラーノードが使用するのと同じ **internal_api** ネットワークを使用する必要があります。したがって、リーフごとに別個のネットワークが必要であるため、インスタンス HA をスパイン/リーフ環境にデプロイする構成はサポートされません。
- インストール後の director でのインスタンス HA の無効化はサポートされていません。デプロイメントからインスタンス HA コンポーネントを手動で削除する回避策は、「How can I remove Instance HA components from the controller nodes?」の記事を参照してください。.**この回避策は、そのまま提供されます。**実稼働環境で実装する前に、テスト環境で手順を検証する必要があります。

オーバークラウドのデプロイに関する一般的な情報は、『director のインストールと使用方法』を参照してください。カスタムロールの詳細は、「コンポーザブルサービスとカスタムロール」を参照してください。

インスタンス HA ロール、フレーバー、およびプロファイルの設定

1. **ComputeInstanceHA** ロールをロールデータのファイルに追加し、ファイルを再生成します。以下に例を示します。

```
$ openstack overcloud roles generate -o ~/my_roles_data.yaml Controller Compute
ComputeInstanceHA
```

ComputeInstanceHA ロールには、デフォルトの **Compute** ロールの全サービスに加えて、**ComputeInstanceHA** および **PacemakerRemote** サービスが含まれます。カスタムロールおよび **roles-data.yaml** に関する一般的な情報は、**『オーバークラウドの高度なカスタマイズ』**の「ロール」セクションを参照してください。

2. インスタンス HA に指定するコンピュートノードにタグ付けする **compute-instance-ha** フレーバーを作成します。以下に例を示します。

```
$ source ~/stackrc
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 compute-instance-ha
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="compute-instance-ha"
compute-instance-ha
```

3. インスタンス HA に指定する各コンピュートノードに **compute-instance-ha** プロファイルをタグ付けします。

```
$ openstack baremetal node set --property capabilities='profile:compute-instance-ha,boot_option:local' <NODE UUID>
```

4. 以下の内容の環境ファイルを作成して、**ComputeInstanceHA** ロールを **compute-instance-ha** フレーバーにマッピングします。

```
parameter_defaults:
  OvercloudComputeInstanceHAFlavor: compute-instance-ha
```

フェンシングの有効化

1. フェンシング情報を定義した環境ファイルを作成して、オーバークラウドの全コントローラーノードおよびコンピュートノードのフェンシングを有効にします。必ずアクセス可能な場所に環境ファイルを作成してください (**~/templates** 等)。以下に例を示します。

```
parameter_defaults:
  EnableFencing: true
  FencingConfig:
    devices:
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:c7
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6230
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:cb
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6231
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:cf
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6232
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:d3
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6233
        passwd: password
        lanplus: 1
```

```
- agent: fence_ipmilan
  host_mac: 00:ec:ad:cb:3c:d7
  params:
    login: admin
    ipaddr: 192.168.24.1
    ipport: 6234
    passwd: password
    lanplus: 1
```

フェンシングおよび STONITH の設定に関する詳細は、**『高可用性デプロイメントと使用方法』**の「STONITH を使用したコントローラノードのフェンシング」を参照してください。

2. デフォルトでは、インスタンス HA は共有ストレージを使用します。共有ストレージがコンピュートインスタンス用に設定されていない場合には、前の手順で作成した環境ファイルに以下のパラメーターを追加します。

```
parameter_defaults:
  ExtraConfig:
    tripleo::instanceha::no_shared_storage: true
```

インスタンスのディスクイメージ用に共有ストレージを設定するのではなく、OpenStack Block Storage (cinder) ボリュームから起動する方法は、「共有ストレージに関する注意事項」を参照してください。

オーバークラウドのデプロイ

作成したすべての環境ファイルのほか、**compute-instanceha.yaml** 環境ファイル用に、**-e** オプションを指定して **openstack overcloud deploy** コマンドを実行します。以下に例を示します。

```
$ openstack overcloud deploy --templates \
  -e <FLAVOR_ENV_FILE> \
  -e <FENCING_ENV_FILE> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/compute-instanceha.yaml
```

注記

- **compute-instanceha.yaml** 環境ファイルは変更しないでください。
- オーバークラウドデプロイメントに追加する各環境ファイルへのパスを含めるようにしてください。
- アンダークラウンドの作成後は、いつでもオーバークラウドにインスタンス HA を設定することができます。オーバークラウドをすでにデプロイしている場合は、新しいインスタンス HA ファイルで **overcloud deploy** コマンドを再実行する必要があります。

デプロイメントが完了すると、それぞれのコンピュートノードには **STONITH** デバイスおよび **GuestNode** サービスが含まれます。

3.1. 共有ストレージに関する注意事項

一般的に、インスタンス HA ではインスタンスのディスクイメージ用に共有ストレージを設定する必要があります。したがって、**no-shared-storage** オプションの使用を試みると、退避中に **InvalidSharedStorage** エラーが表示され、インスタンスが別のコンピュートノードで起動しなくなり

ます。

ただし、すべてのインスタンスが OpenStack Block Storage (**cinder**) 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、インスタンスのディスクイメージ用に共有ストレージを設定する必要はないので、**no-shared-storage** オプションを使用してすべてのインスタンスを退避させることができます。

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、退避したインスタンスは別のコンピュートノード上の同じボリュームから起動するはずです。したがって、OS イメージおよびアプリケーションデータが OpenStack Block Storage ボリュームに保管されているので、退避したインスタンスは直ちにジョブを再開します。

第4章 インスタンス HA による退避のテスト

警告

以下の手順では、コンピュートノードを故意にクラッシュさせます。これにより、意図的にインスタンス HA によるインスタンスの自動退避を実行させます。

1. テスト用インスタンスをホストするコンピュートノードをクラッシュさせる前に、オーバークラウド上でインスタンスを 1 つ以上起動します。

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

2. インスタンスをホストするコンピュートノードにログインし、**root** ユーザーに変更します。**compute-n** をコンピュートノードの名前に置き換えます。

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

3. コンピュートノードをクラッシュさせます。

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

4. 数分待ってから、これらのインスタンスが別のコンピュートノード上で作成し直されていることを確認します。

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

第5章 以前のバージョンからのインスタンス HA の無効化

以前のバージョンから Red Hat OpenStack Platform 13 にアップグレードする場合には、アップグレードの前に手動でインスタンス HA を無効にする必要があります。これには、メジャーおよびマイナーアップグレードならびに Fast Forward Upgrade が含まれます。

注記

Red Hat OpenStack Platform 13 以降では、director を使用してインスタンス HA のインストールおよびアップグレードを行います。バージョン 13 からそれ以降のバージョンにアップグレードする場合は、手動によるロールバックは必要ありません。

STONITH デバイスを使用しないインスタンス HA の無効化

STONITH デバイスなしでデプロイされたインスタンス HA を無効化するには、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="[rhos-NN]" -e instance_ha_action="uninstall"
```

"[rhos-NN]" フィールドの値を OpenStack Platform の実際のバージョンに置き換えます。例: **"rhos-10"**

STONITH デバイスを使用したインスタンス HA の無効化

インスタンス HA をデプロイする際に **stonith_devices** オプションを使用している場合には、インスタンス HA を無効にする際にこの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 設定で STONITH デバイスを除外している場合には、以下のコマンド構文を使用します。

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

"[rhos-NN]" フィールドの値を OpenStack Platform の実際のバージョンに置き換えます。例: **"rhos-10"**