Red Hat Enterprise Linux 8

Red Hat OpenStack Platform での Red Hat High Availability クラスターの設定

RHOSP インスタンスでの HA クラスターおよびクラスターリソースのインストールと設定

Last Updated: 2024-02-14

Red Hat Enterprise Linux 8 Red Hat OpenStack Platform での Red Hat High Availability クラスターの設定

RHOSP インスタンスでの HA クラスターおよびクラスターリソースのインストールと設定

法律上の通知

概要

Red Hat High Availability Add-On を使用して、Red Hat OpenStack Platform (RHOSP)インスタンスで高可用性 (HA) クラスターを設定できます。このガイドでは、必要なパッケージおよびエージェントをインストールする手順と、基本的なクラスター、フェンスリソース、および HA クラスターリソースを設定する例について説明します。

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、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 はじめに

Red Hat High Availability Add-On を使用して、Red Hat OpenStack Platform (RHOSP) インスタンスで Red Hat High Availability (HA) クラスターを設定できます。これには、必要なパッケージおよびエージェントのインストール、基本的なクラスターの設定、フェンシングリソースの設定、および HA クラスターリソースの設定が必要になります。

RHOSP ドキュメントについては、Red Hat OpenStack Platform の製品ドキュメント を参照してください。

RHEL High Availability クラスターでの RHOSP インスタンスを使用する場合に適用される Red Hat のポリシー、要件、および制限については、Support Policies for RHEL High Availability Clusters - OpenStack Virtual Machines as Cluster Members を参照してください。

第2章 HA インスタンスの RHOSP サーバーグループ設定

RHOSP HA クラスターノードインスタンスを作成する前に、インスタンスサーバーグループを作成します。アフィニティーポリシーでインスタンスをグループ化します。複数のクラスターを設定する場合は、クラスターごとに 1 つのサーバーグループのみがあることを確認してください。

サーバーグループに設定したアフィニティーポリシーは、ハイパーバイザーに障害が発生した場合にクラスターが機能し続けるかどうかを判断できます。

デフォルトのアフィニティーポリシーは **affinity** です。このアフィニティーポリシーを使用すると、すべてのクラスターノードを同じ RHOSP ハイパーバイザーに作成できます。この場合、ハイパーバイザーに障害が発生すると、クラスター全体に障害が発生します。このため、**anti-affinity** または **soft-anti-affinity** のサーバーグループにアフィニティーポリシーを設定します。

- **anti-affinity** のアフィニティーポリシーでは、サーバーグループは、コンピュートノードごとに 1 つのクラスターノードのみを許可します。コンピュートノードよりも多くのクラスターノードを作成しようとすると、エラーが発生します。この設定は、RHOSP ハイパーバイザーの障害に対して最高レベルの保護を提供しますが、大規模なクラスターをデプロイするには、利用可能なリソースよりも多くのリソースが必要になる場合があります。

- **soft-anti-affinity** のアフィニティーポリシーを使用すると、サーバーグループは、すべてのコンピュートノードにできるだけ均等にクラスターノードを分散します。これは、**anti-affinity** ポリシーよりもハイパーバイザーの障害に対する保護が低くなりますが、**affinity** のアフィニティーポリシーよりも高いレベルの高可用性を提供します。

デプロイメントのサーバーグループアフィニティーポリシーを決定する際には、次のクラスターコンポーネントを考慮して、クラスターのニーズと使用可能なリソースのバランスを取る必要があります。

- クラスター内のノード数
- 利用可能な RHOSP コンピュートノード数
- クラスタークォーラムがクラスターオペレーションを維持するために必要なノード数

アフィニティーとインスタンスサーバーグループの作成に関する詳細は、Compute スケジューラーのフィルター と コマンドラインインターフェイスリファレンス を参照してください。

第3章 高可用性および RHOSP パッケージとエージェントのインストール

Red Hat OpenStack Platform (RHOSP) に Red Hat High Availability クラスターの設定に必要なパッケージをインストールします。クラスターメンバーとして使用する各ノードにパッケージをインストールする必要があります。

前提条件

- HA インスタンスの RHOSP サーバーグループ設定 で設定され、説明されているように、HA クラスターノードとして使用する RHOSP インスタンスのサーバーグループがある。
- 各 HA クラスターノードの RHOSP インスタンスがある。
  - インスタンスはサーバーグループのメンバー
  - インスタンスは、RHEL 8.7 以降を実行するノードとして設定

手順

1. RHEL HA リポジトリーと RHOSP ツールチャネルを有効にします。

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
# subscription-manager repos --enable=openstack-16-tools-for-rhel-8-x86_64-rpms
```

2. Red Hat High Availability Add-On ソフトウェアパッケージを、RHOSP クラスターリソースエージェントおよび RHOSP フェンスエージェントに必要なパッケージと共にインストールします。

```
# yum install pcs pacemaker python3-openstackclient python3-novaclient fence-agents-openstack
```

3. 各ノードに **pcs** および **pacemaker** パッケージをインストールすると、**pcs** 管理アカウントであるユーザー **hacluster** が作成されます。すべてのクラスターノードの **hacluster** ユーザーのパスワードを作成します。すべてのノードで同じパスワードを使用すると、クラスター管理が簡素化されます。

```
# passwd hacluster
```

4. **firewalld.service** がインストールされている場合は、RHEL ファイアウォールに高可用性サービスを追加します。

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

5. **pcs** サービスを起動し、システムの起動時に開始できるようにします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

6. **pcs** サービスが実行されていることを確認します。

```
# systemctl status pcsd.service
```

```
pcsd.service - PCS GUI and remote configuration interface
Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
Active: active (running) since Thu 2018-03-01 14:53:28 UTC; 28min ago
Docs: man:pcsd(8)
man:pcs(8)
Main PID: 5437 (pcsd)
CGroup: /system.slice/pcsd.service
    └─5437 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
Mar 01 14:53:27 ip-10-0-0-48.ec2.internal systemd[1]: Starting PCS GUI and remote
configuration interface…
Mar 01 14:53:28 ip-10-0-0-48.ec2.internal systemd[1]: Started PCS GUI and remote
configuration interface.
```

7. **/etc/hosts** ファイルを編集して、RHEL ホスト名と内部 IP アドレスを追加します。 **/etc/hosts** の詳細は、Red Hat ナレッジベースのソリューション記事 How should the /etc/hosts file be set up on RHEL cluster nodes? を参照してください。

関連情報

- Red Hat High Availability クラスターの設定と管理の詳細は、高可用性クラスターの設定および管理 を参照してください。

第4章 RHOSP の認証方法のセットアップ

高可用性フェンスエージェントとリソースエージェントは、RHOSP との通信に 3 つの認証方法をサポートしています。

- **clouds.yaml** 設定ファイルを使用した認証
- OpenRC 環境スクリプトによる認証
- Pacemaker による **username** とパスワードを使用した認証

クラスターに使用する認証方法が決定したら、フェンシングまたはクラスターリソースを作成するときに適切な認証パラメーターを指定します。

4.1. CLOUDS.YAML ファイルを使用した RHOSP での認証

認証に **clouds.yaml** ファイルを使用するこのドキュメントの手順は、この手順で示した **clouds.yaml** ファイルを使用します。この手順では、このファイルで定義されているように、**cloud= parameter** に **ha-example** を指定します。

手順

1. クラスターの一部となる各ノードで、以下の例のように **clouds.yaml** ファイルを作成します。**clouds.yaml** ファイルの作成に関する詳細は、ユーザーおよびアイデンティティー管理ガイド を参照してください。

```
$ cat .config/openstack/clouds.yaml
clouds:
  ha-example:
    auth:
      auth_url: https://<ip_address>:13000/
      project_name: rainbow
      username: unicorns
      password: <password>
      user_domain_name: Default
      project_domain_name: Default
  <. . . additional options . . .>
  region_name: regionOne
  verify: False
```

2. 次の基本的な RHOSP コマンドを使用して、認証が成功し、RHOSP API にアクセスできるかどうかをテストします。**ha-example** を、作成した **clouds.yaml** ファイルで指定したクラウドの名前に置き換えます。このコマンドでサーバーの一覧が表示されない場合は、RHOSP 管理者に連絡してください。

```
$ openstack --os-cloud=ha-example server list
```

3. クラスターリソースやフェンシングリソースを作成する際にクラウドパラメーターを指定します。

4.2. OPENRC 環境スクリプトを使用した RHOSP での認証

OpenRC 環境スクリプトを使用して RHOSP で認証するには、次の手順を実行します。

手順

1. クラスターの一部となる各ノードで、OpenRC 環境スクリプトを設定します。OpenRC 環境スクリプトの作成に関する詳細は、Set environment variables using the OpenStack RC file を参照してください。

2. 次の基本的な RHOSP コマンドを使用して、認証が成功し、RHOSP API にアクセスできるかどうかをテストします。このコマンドでサーバーの一覧が表示されない場合は、RHOSP 管理者に連絡してください。

```
$ openstack server list
```

3. クラスターリソースまたはフェンシングリソースを作成する際に、**openrc** パラメーターを指定します。

4.3. USERNAME とパスワードを使用した RHOSP での認証

username とパスワードを使用して RHOSP で認証するには、リソースを作成するときにクラスターリソースまたはフェンシングリソースの **username**、**password**、および **auth_url** パラメーターを指定します。RHOSP 設定によっては、追加の認証パラメーターが必要になる場合があります。使用する認証パラメーターは、RHOSP 管理者によって提供されます。

第5章 RED HAT OPENSTACK PLATFORM での基本クラスターの作成

この手順では、フェンシングやリソースが設定されていない RHOSP プラットフォーム上に高可用性クラスターを作成します。

前提条件

- RHOSP インスタンスが各 HA クラスターノードに設定されている。
- HA クラスターノードが RHEL 8.7 以降を実行している
- 高可用性および RHOSP パッケージとエージェントのインストール の説明のように、各ノードにインストールされた高可用性および RHOSP パッケージがある。

手順

1. クラスターノードのいずれかで以下のコマンドを実行し、**pcs** ユーザー **hacluster** を認証します。クラスターの各ノードの名前を指定します。この例では、クラスターのノードは **node01**、**node02** および **node03** です。

```
[root@node01 ~]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. クラスターを作成します。この例では、クラスターの名前は **newcluster** です。

```
[root@node01 ~]# pcs cluster setup newcluster node01 node02 node03

...

Synchronizing pcsd certificates on nodes node01, node02, node03…
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates…
node02: Success
node03: Success
node01: Success
```

検証

1. クラスターを有効にします。

```
[root@node01 ~]# pcs cluster enable --all
node01: Cluster Enabled
node02: Cluster Enabled
node03: Cluster Enabled
```

2. クラスターを起動します。コマンドの出力は、クラスターが各ノードで起動したかどうかを示します。

```
[root@node01 ~]# pcs cluster start --all
node02: Starting Cluster…
node03: Starting Cluster…
node01: Starting Cluster...
```

第6章 RED HAT OPENSTACK PLATFORM における HA クラスターのフェンシングの設定

フェンシング設定により、HA クラスターで誤動作しているノードが自動的に分離されます。これにより、ノードがクラスターのリソースを消費したり、クラスターの機能が損なわれたりすることを防ぎます。

fence_openstack フェンスエージェントを使用して、RHOSP 上の HA クラスターのフェンスデバイスを設定します。RHOSP フェンスエージェントのオプションを表示するには、次のコマンドを使用します。

```
# pcs stonith describe fence_openstack
```

前提条件

- RHOSP で実行されている設定済みの HA クラスターがある。
- RHOSP の認証方法のセットアップ で説明されているように、クラスター設定に使用する RHOSP 認証方法を使用して、RHOSP API へアクセスできる。
- クラスタープロパティー **stonith-enabled** は、デフォルト値の **true** に設定されている。実稼働環境でフェンシングを無効にすることは適していないため、フェンシングが無効になっている場合は、Red Hat ではクラスターがサポートされないことに注意してください。以下のコマンドを実行して、フェンシングが開始されていることを確認します。

```
# pcs property config --all
Cluster Properties:
. . .
stonith-enabled: true
```

手順

クラスター内の任意のノードから次の手順を実行します。

1. クラスター内の各ノードの UUID を決定します。
   次のコマンドは、**ha-example** プロジェクト内のすべての RHOSP インスタンス名の完全なリストと、ヘデッィング **ID** のもと、その RHOSP インスタンスに関連付けられたクラスターノードの UUID を表示します。ノードのホスト名は RHOSP インスタンス名と一致しない場合があります。

```
# openstack --os-cloud="ha-example" server list
…
| ID                                   | Name            |...
| 6d86fa7d-b31f-4f8a-895e-b3558df9decb|testnode-node03-vm|...
| 43ed5fe8-6cc7-4af0-8acd-a4fea293bc62|testnode-node02-vm|...
| 4df08e9d-2fa6-4c04-9e66-36a6f002250e|testnode-node01-vm|...
```

2. **pcmk_host_map parameter** を使用してクラスター内の各ノードをそのノードの UUID にマップし、フェンシングデバイスを作成します。以下のフェンスデバイス作成コマンドの例では、それぞれ異なる認証方法を使用しています。

   a. 次のコマンドは、認証用の **clouds.yaml** 設定ファイルを使用して、3 ノードクラスター用の **fence_openstack** フェンシングデバイスを作成します。**cloud= parameter** には、clouds.yaml` ファイルのクラウド名を指定します。

```
# pcs stonith create fenceopenstack fence_openstack
pcmk_host_map="node01:4df08e9d-2fa6-4c04-9e66-
36a6f002250e;node02:43ed5fe8-6cc7-4af0-8acd-a4fea293bc62;node03:6d86fa7d-
b31f-4f8a-895e-b3558df9decb" power_timeout="240" pcmk_reboot_timeout="480"
pcmk_reboot_retries="4" cloud="ha-example"
```

b. 次のコマンドは、認証用の OpenRC 環境スクリプトを使用して、**fence_openstack** フェンシングデバイスを作成します。

```
# pcs stonith create fenceopenstack fence_openstack
pcmk_host_map="node01:4df08e9d-2fa6-4c04-9e66-
36a6f002250e;node02:43ed5fe8-6cc7-4af0-8acd-a4fea293bc62;node03:6d86fa7d-
b31f-4f8a-895e-b3558df9decb" power_timeout="240" pcmk_reboot_timeout="480"
pcmk_reboot_retries="4" openrc="/root/openrc"
```

c. 次のコマンドは、認証にユーザー名とパスワードを使用して、**fence_openstack** フェンシングデバイスを作成します。**username**、**password**、**project_name**、および **auth_url** などの認証パラメーターは、RHOSP 管理者によって提供されます。

```
# pcs stonith create fenceopenstack fence_openstack
pcmk_host_map="node01:4df08e9d-2fa6-4c04-9e66-
36a6f002250e;node02:43ed5fe8-6cc7-4af0-8acd-a4fea293bc62;node03:6d86fa7d-
b31f-4f8a-895e-b3558df9decb" power_timeout="240" pcmk_reboot_timeout="480"
pcmk_reboot_retries="4" username="XXX" password="XXX"
project_name="rhelha" auth_url="XXX" user_domain_name="Default"
```

検証

1. クラスター内のいずれかのノードから、クラスター内の別のノードをフェンスし、クラスターのステータスを確認します。フェンスされたノードがオフラインの場合、フェンシング操作は成功しました。

```
[root@node01 ~] # pcs stonith fence node02
[root@node01 ~] # pcs status
```

2. フェンシングしたノードを再起動し、ステータスをチェックして、ノードが起動したことを確認します。

```
[root@node01 ~] # pcs cluster start node02
[root@node01 ~] # pcs status
```

第7章 RED HAT OPENSTACK PLATFORM での HA クラスターリソースの設定

次の表には、RHOSP 上の HA クラスターのリソースを設定するために使用する RHOSP 固有のリソースエージェントを記載しています。

openstack-info (必須)	RHOSP 固有のリソースエージェントのサポートを提供します。**fence_openstack** フェンスエージェント以外の RHOSP 固有のリソースエージェントを実行するには、**openstack-info** リソースをクラスターのクローンリソースとして設定する必要があります。openstack-info リソースの設定に関する詳細は、Red Hat OpenStack Platform 上の HA クラスターで **openstack-info** リソースを設定する を参照してください。
openstack-virtual-ip	仮想 IP アドレスリソースを設定します。**openstack-virtual-ip** リソースの設定に関する詳細は、Red Hat Openstack Platform 上の HA クラスターにおける仮想 IP アドレスの設定 を参照してください。
openstack-floating-ip	Floating IP アドレスリソースを設定します。**openstack-floating-ip** リソースの設定に関する詳細は、Red Hat OpenStack Platform 上の HA クラスターにおけるフローティング IP アドレスの設定 を参照してください。
openstack-cinder-volume	ブロックストレージリソースを設定します。**openstack-cinder-volume** リソースの設定に関する詳細は、Red Hat OpenStack Platform 上の HA クラスターにおけるブロックストレージリソースの設定 を参照してください。

他のクラスターリソースを設定する場合は、標準の Pacemaker リソースエージェントを使用します。

7.1. RED HAT OPENSTACK PLATFORM 上の HA クラスターでの OPENSTACK-INFO リソースの設定 (必須)

fence_openstack フェンスエージェント以外の RHOSP 固有のリソースエージェントを実行するには、**openstack-info** リソースを設定する必要があります。

openstack-info リソースを作成するこの手順は、RHOSP 認証に **clouds.yaml** ファイルを使用します。

前提条件

- RHOSP で実行されている設定済みの HA クラスターがある。
- RHOSP の認証方法のセットアップ で説明されているように、クラスター設定に使用する RHOSP 認証方法を使用して、RHOSP API へアクセスできる。

手順

クラスター内の任意のノードから次の手順を実行します。

1. **openstack-info** リソースエージェントのオプションを表示するには、次のコマンドを実行します。

```
# pcs resource describe openstack-info
```

2. **openstack-info** リソースをクローンリソースとして作成します。この例では、このリソースにも **openstack-info** という名前が付けられています。この例では、**clouds.yaml** 設定ファイルを使用しており、**cloud=** パラメーターは **clouds.yaml** ファイル内のクラウドの名前に設定されています。

```
# pcs resource create openstack-info openstack-info cloud="ha-example" clone
```

3. クラスターのステータスをチェックして、リソースが実行中であることを確認します。

```
# pcs status

Full List of Resources:

    * Clone Set: openstack-info-clone [openstack-info]:
      * Started: [ node01 node02 node03 ]
```

7.2. RED HAT OPENSTACK PLATFORM 上の HA クラスターにおける仮想 IP アドレスの設定

RHOSP プラットフォーム上の HA クラスターの RHOSP 仮想 IP アドレスリソースを作成するこの手順は、RHOSP 認証に **clouds.yaml** ファイルを使用します。

RHOSP 仮想 IP リソースは、**IPaddr2** クラスターリソースと連動して動作します。RHOSP 仮想 IP アドレスリソースを設定すると、リソースエージェントは、RHOSP インフラストラクチャーが仮想 IP アドレスをネットワーク上のクラスターノードに関連付けるようにします。これにより、**IPaddr2** リソースがそのノードで機能できるようになります。

前提条件

- RHOSP で実行されている設定済みの HA クラスターがある。
- 仮想 IP アドレスとして使用するために割り当てられた IP アドレスがある。
- RHOSP の認証方法のセットアップ で説明されているように、クラスター設定に使用する RHOSP 認証方法を使用して、RHOSP API へアクセスできる。

手順

クラスター内の任意のノードから次の手順を実行します。

1. **openstack-virtual-ip** リソースエージェントのオプションを表示するには、次のコマンドを実行します。

```
# pcs resource describe openstack-virtual-ip
```

2. 次のコマンドを実行して、使用している仮想 IP アドレスのサブネット ID を確認します。この例では、仮想 IP アドレスは 172.16.0.119 です。

```
# openstack --os-cloud=ha-example subnet list
+--------------------------------------+ ...  +----------------+
| ID                                   | ...  | Subnet         |
```

```
+--------------------------------------+ ...  +----------------+
| 723c5a77-156d-4c3b-b53c-ee73a4f75185 | ...  | 172.16.0.0/24  |
+--------------------------------------+ ...  +----------------+
```

3. RHOSP 仮想 IP アドレスリソースを作成します。
次のコマンドは、前の手順で決定したサブネット ID を指定して、IP アドレスが 172.16.0.119 の RHOSP 仮想 IP アドレスリソースを作成します。

```
# pcs resource create ClusterIP-osp ocf:heartbeat:openstack-virtual-ip cloud=ha-example ip=172.16.0.119 subnet_id=723c5a77-156d-4c3b-b53c-ee73a4f75185
```

4. 順序および場所の制約を設定します。
   - **openstack-info** リソースが、仮想 IP アドレスリソースの前に起動することを確認します。
   - 仮想 IP アドレスリソースが **openstack-info** リソースと同じノードで実行されていることを確認します。

```
# pcs constraint order start openstack-info-clone then ClusterIP-osp
Adding openstack-info-clone ClusterIP-osp (kind: Mandatory) (Options: first-action=start then-action=start)
# pcs constraint colocation add ClusterIP-osp with openstack-info-clone score=INFINITY
```

5. 仮想 IP アドレスの **IPaddr2** リソースを作成します。

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=172.16.0.119
```

6. **openstack-virtual-ip** リソースが **IPaddr2** リソースの前に起動し、**IPaddr2** リソースが **openstack-virtual-ip** リソースと同じノードで実行されるように、順序と場所の制約を設定します。

```
# pcs constraint order start ClusterIP-osp then ClusterIP
Adding ClusterIP-osp ClusterIP (kind: Mandatory) (Options: first-action=start then-action=start)
# pcs constraint colocation add ClusterIP with ClusterIP-osp
```

検証

1. リソース制約の設定を確認します。

```
# pcs constraint config
Location Constraints:
Ordering Constraints:
  start ClusterIP-osp then start ClusterIP (kind:Mandatory)
  start openstack-info-clone then start ClusterIP-osp (kind:Mandatory)
Colocation Constraints:
  ClusterIP with ClusterIP-osp (score:INFINITY)
  ClusterIP-osp with openstack-info-clone (score:INFINITY)
```

2. クラスターのステータスをチェックして、リソースが実行されていることを確認します。

```
# pcs status
```

```
. . .

Full List of Resources:
  * fenceopenstack      (stonith:fence_openstack):       Started node01
  * Clone Set: openstack-info-clone [openstack-info]:
    * Started: [ node01 node02 node03 ]
  * ClusterIP-osp       (ocf::heartbeat:openstack-virtual-ip):   Started node03
  * ClusterIP   (ocf::heartbeat:IPaddr2):        Started node03
```

7.3. RED HAT OPENSTACK PLATFORM 上の HA クラスターにおけるフローティング IP アドレスの設定

次の手順では、RHOSP で HA クラスターのフローティング IP アドレスリソースを作成します。この手順では、RHOSP 認証に **clouds.yaml** ファイルを使用します。

前提条件

- RHOSP で実行されている設定済みの HA クラスターがある。
- RHOSP 管理者によって割り当てられた、フローティング IP アドレスとして使用するパブリックネットワーク上の IP アドレスがある。
- RHOSP の認証方法のセットアップ で説明されているように、クラスター設定に使用する RHOSP 認証方法を使用して、RHOSP API へアクセスできる。

手順

クラスター内の任意のノードから次の手順を実行します。

1. **openstack-floating-ip** リソースエージェントのオプションを表示するには、次のコマンドを実行します。

```
# pcs resource describe openstack-floating-ip
```

2. フローティング IP アドレスリソースの作成に使用するパブリックネットワーク上のアドレスのサブネット ID を見つけます。
   a. パブリックネットワークは通常、デフォルトゲートウェイのあるネットワークです。次のコマンドを実行して、デフォルトゲートウェイアドレスを表示します。

```
# route -n | grep ^0.0.0.0 | awk '{print $2}'
172.16.0.1
```

   b. 次のコマンドを実行して、パブリックネットワークのサブネット ID を見つけます。このコマンドは、ID とサブネットの見出しを含むテーブルを生成します。

```
# openstack --os-cloud=ha-example subnet list
+--------------------------------------+---+---------------+
| ID                                   |   | Subnet
+--------------------------------------+---+---------------+
| 723c5a77-156d-4c3b-b53c-ee73a4f75185 |   | 172.16.0.0/24 |
+--------------------------------------+-----------------+
```

3. リソースのパブリック IP アドレスとそのアドレスのサブネット ID を指定して、フローティン

グ IP アドレスリソースを作成します。フローティング IP アドレスリソースを設定すると、リソースエージェントはパブリックネットワーク上に仮想 IP アドレスを設定し、それをクラスターノードに関連付けます。

```
# pcs resource create float-ip openstack-floating-ip cloud="ha-example"
ip_id="10.19.227.211" subnet_id="723c5a77-156d-4c3b-b53c-ee73a4f75185"
```

4. **openstack-info** リソースがフローティング IP アドレスリソースの前に起動するように、順序の制約を設定します。

```
# pcs constraint order start openstack-info-clone then float-ip
Adding openstack-info-clone float-ip (kind: Mandatory) (Options: first-action=start then-action=start
```

5. フローティング IP アドレスリソースが **openstack-info** リソースと同じノードで実行されるように、場所の制約を設定します。

```
# pcs constraint colocation add float-ip with openstack-info-clone score=INFINITY
```

検証

1. リソース制約の設定を確認します。

```
# pcs constraint config
Location Constraints:
Ordering Constraints:
  start openstack-info-clone then start float-ip (kind:Mandatory)
Colocation Constraints:
  float-ip with openstack-info-clone (score:INFINITY)
```

2. クラスターのステータスをチェックして、リソースが実行されていることを確認します。

```
# pcs status
. . .
Full List of Resources:
  * fenceopenstack     (stonith:fence_openstack):      Started node01
  * Clone Set: openstack-info-clone [openstack-info]:
    * Started: [ node01 node02 node03 ]
  * float-ip    (ocf::heartbeat:openstack-floating-ip):  Started node02
```

7.4. RED HAT OPENSTACK PLATFORM 上の HA クラスターにおけるブロックストレージリソースの設定

次の手順では、RHOSP で HA クラスターのブロックストレージリソースを作成します。この手順では、RHOSP 認証に **clouds.yaml** ファイルを使用します。

前提条件

- RHOSP で実行されている設定済みの HA クラスターがある。
- RHOSP 管理者によって作成されたブロックストレージボリュームがある。

- RHOSP の認証方法のセットアップ で説明されているように、クラスター設定に使用する RHOSP 認証方法を使用して、RHOSP API へアクセスできる。

手順

クラスター内の任意のノードから次の手順を実行します。

1. **openstack-cinder-volume** リソースエージェントのオプションを表示するには、次のコマンドを実行します。

```
# pcs resource describe openstack-cinder-volume
```

2. クラスターリソースとして設定するブロックストレージボリュームのボリューム ID を決定します。
次のコマンドを実行して、各ボリュームの UUID と名前を含む、使用可能なボリュームのテーブルを表示します。

```
# openstack --os-cloud=ha-example volume list
| ID                                   | Name                    |
| 23f67c9f-b530-4d44-8ce5-ad5d056ba926| testvolume-cinder-data-disk |
```

ボリューム名がすでにわかっている場合は、設定するボリュームを指定して、次のコマンドを実行できます。これにより、ID フィールドを含むテーブルが表示されます。

```
# openstack --os-cloud=ha-example volume show testvolume-cinder-data-disk
```

3. ボリュームの ID を指定して、ブロックストレージリソースを作成します。

```
# pcs resource create cinder-vol openstack-cinder-volume volume_id="23f67c9f-b530-4d44-8ce5-ad5d056ba926" cloud="ha-example"
```

4. **openstack-info** リソースがブロックストレージリソースの前に起動するように、順序の制約を設定します。

```
# pcs constraint order start openstack-info-clone then cinder-vol
Adding openstack-info-clone cinder-vol (kind: Mandatory) (Options: first-action=start then-action=start
```

5. ブロックストレージリソースが **openstack-info** リソースと同じノードで実行されるように、場所の制約を設定します。

```
# pcs constraint colocation add cinder-vol with openstack-info-clone score=INFINITY
```

検証

1. リソース制約の設定を確認します。

```
# pcs constraint config
Location Constraints:
Ordering Constraints:
  start openstack-info-clone then start cinder-vol (kind:Mandatory)
Colocation Constraints:
  cinder-vol with openstack-info-clone (score:INFINITY)
```

2. クラスターのステータスをチェックして、リソースが実行中であることを確認します。

```
# pcs status
. . .
Full List of Resources:
  * Clone Set: openstack-info-clone [openstack-info]:
    * Started: [ node01 node02 node03 ]
  * cinder-vol  (ocf::heartbeat:openstack-cinder-volume):        Started node03
  * fenceopenstack      (stonith:fence_openstack):        Started node01
```