Red Hat Enterprise Linux 9.0

Configuring a Red Hat High Availability cluster on Red Hat OpenStack Platform

Installing and configuring HA clusters and cluster resources on RHOSP instances

Last Updated: 2022-11-18

Red Hat Enterprise Linux 9.0 Configuring a Red Hat High Availability cluster on Red Hat OpenStack Platform

Installing and configuring HA clusters and cluster resources on RHOSP instances

Legal Notice

Abstract

This document provides procedures for configuring a Red Hat High Availability (HA) cluster and its fencing and resource agents on Red Hat OpenStack Platform (RHOSP) instances.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the Bugzilla website.
2. Select the correct version from the **Version** menu.
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Submit Bug**.

CHAPTER 1. PREFACE

This document provides procedures for configuring a Red Hat High Availability (HA) cluster and its fencing and resource agents on Red Hat OpenStack Platform (RHOSP) instances.

For general RHOSP documentation, see Product Documentation for Red Hat Openstack Platform .

For Red Hat's policies, requirements, and limitations applicable to the use of RHOSP instances as members of a RHEL High Availability cluster, see Support Policies for RHEL High Availability Clusters - OpenStack Virtual Machines as Cluster Members - Red Hat Customer Portal.

CHAPTER 2. RECOMMENDED RHOSP SERVER GROUP CONFIGURATION

Before creating the RHOSP instances to use as HA cluster nodes, create an instance server group to group the instances by an affinity policy, with one server group per cluster when you configure multiple clusters.

When creating a server group, the affinity policy you set for the server group determines the cluster's resilience if the hypervisor fails.

- With the default affinity policy of **affinity**, all of the cluster nodes could be created on the same RHOSP hypervisor. In this case, if the hypervisor fails the entire cluster fails. For this reason, Red Hat recommends that you set an affinity policy for the server group of **anti-affinity** or **soft-anti-affinity**.

- With an affinity policy of **anti-affinity**, the server group allows only one cluster node per Compute node. Attempting to create more cluster nodes than Compute nodes generates an error. While this configuration provides the highest protection against RHOSP hypervisor failures, it may require more resources to deploy large clusters than you have available.

- With an affinity policy of **soft-anti-affinity**, the server group will distribute cluster nodes as evenly as possible across all Compute nodes. Although this provides less protection against hypervisor failures than a policy of **anti-affinity**, it provides a greater level of high availability than an affinity policy of **affinity**.

Determining the server group affinity policy for your deployment requires balancing your cluster needs against the resources you have available by taking the following cluster components into account:

- The number of nodes in the cluster

- The number of RHOSP Compute nodes available

- The number of nodes required for cluster quorum to retain cluster operations

For information on affinity and creating an instance server group, Compute scheduler filters and the Command Line Interface Reference.

CHAPTER 3. INSTALLING THE HIGH AVAILABILITY AND RHOSP PACKAGES AND AGENTS

On each of the nodes you will use for an HA cluster on OpenStack platform, use the following procedure to install the packages required for configuring a Red Hat High Availability cluster on Red Hat OpenStack Platform (RHOSP).

Prerequisites

- A server group for the RHOSP instances to use as HA cluster nodes, as described in Recommended RHOSP server group configuration
- An RHOSP instance for each HA cluster node
  - The instances are members of a server group
  - The instances are configured as nodes running RHEL 9.1 or later

Procedure

1. Enable the RHEL HA repositories and the RHOSP tools channel.

```
# subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
# subscription-manager repos --enable=openstack-17-tools-for-rhel-9-x86_64-rpms
```

2. Install the Red Hat High Availability Add-On software packages, along with the packages that are required for the RHOSP cluster resource agents and the RHOSP fence agents.

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

3. Installing the **pcs** and **pacemaker** packages on each node creates the user **hacluster**, which is the **pcs** administration account. Create a password for user **hacluster** on all cluster nodes. It is recommended that you use the same password for all nodes.

```
# passwd hacluster
```

4. If **firewalld.service** is installed, add the high-availability service to the RHEL firewall.

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

5. Start the **pcs** service and enable it to start on boot.

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

6. Edit the /**etc**/**hosts** file and add RHEL host names and internal IP addresses. See How should the /etc/hosts file be set up on RHEL cluster nodes? for details.

7. Verify that the **pcs** service is running.

```
# 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.
```

Additional resources

- For further information on configuring and managing Red Hat high availability clusters, see Configuring and managing high availability clusters.

CHAPTER 4. AUTHENTICATION METHODS FOR RHOSP

The high availability fence agents and resource agents support three distinct authentication methods for communicating with RHOSP:

- A **clouds.yaml** configuration file
- An OpenRC environment script
- User name and password authentication through Pacemaker

After determining the authentication method to use for the cluster, specify the appropriate authentication parameters when creating a fencing or cluster resource.

4.1. AUTHENTICATION WITH A CLOUDS.YAML FILE

To use a **clouds.yaml** file to authenticate with RHOSP, perform the following steps.

Procedure

1. On each node that will be part of your cluster, set up a **clouds.yaml** file. For information on creating a **clouds.yaml** file, see Users and Identity Management Guide .
   The **clouds.yaml** file for the procedures in this document that use a **clouds.yaml** file for authentication is as follows. Those procedures specify **ha-example** for the **cloud= parameter**, as defined in this file.

```
$ 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. Test whether authentication is successful and you have access to the RHOSP API with the following basic RHOSP command, substituting the name of the cloud you specified in the **clouds.yaml** file you created for **ha-example**. If this command does not successfully display a server list, contact your RHOSP administrator.

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

3. Specify the cloud parameter when creating a cluster resource or a fencing resource.

4.2. AUTHENTICATION WITH AN OPENRC ENVIRONMENT SCRIPT

To use an OpenRC environment script to authenticate with RHOSP, perform the following steps.

Procedure

1. On each node that will be part of your cluster, configure an OpenRC environment script. For information on creating an OpenRC environment script, see Set environment variables using the OpenStack RC file.

2. Test whether authentication is successful and you have access to the RHOSP API with the following basic RHOSP command. If this command does not successfully display a server list, contact your RHOSP administrator.

```
$ openstack server list
```

3. Specify the **openrc** parameter when creating a cluster resource or a fencing resource.

4.3. AUTHENTICATION WITH A USER NAME AND PASSWORD

To authenticate with RHOSP by means of a user name and password, specify the **username**, **password**, and **auth_url** parameters for a cluster resource or a fencing resource when you create the resource. Additional authentication parameters may be required, depending on the RHOSP configuration. The RHOSP administrator provides the authentication parameters to use.

CHAPTER 5. CREATING A BASIC CLUSTER ON RED HAT OPENSTACK PLATFORM

This procedure creates a high availability cluster on an RHOSP platform.

Prerequisites

- An RHOSP instance for each HA cluster node
- The HAcluster node is running RHEL 9.1 or later
- High Availability and RHOSP packages are installed on each node, as described in Installing the high availability and RHOSP packages and agents.

Procedure

1. On one of the cluster nodes, enter the following command to authenticate the **pcs** user **hacluster**. In the command, specify the name of each node in the cluster. In this examples, the nodes for the cluster are **node01**, **node02**, and **node03**.

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

2. Create the cluster. In this example, the cluster is named **newcluster**.

```
[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
```

Verification

1. Enable the cluster.

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

2. Start the cluster.

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

CHAPTER 6. CONFIGURING FENCING FOR AN HA CLUSTER ON RED HAT OPENSTACK PLATFORM

Fencing configuration ensures that a malfunctioning node on your HA cluster is automatically isolated, which prevents the node from consuming the cluster's resources or compromising the cluster's functionality.

Use the **fence_openstack** fence agent to configure a fence device for an HA cluster on RHOSP. Run the following command to view the options for the RHOSP fence agent.

```
# pcs stonith describe fence_openstack
```

Prerequisites

- A configured HA cluster running on RHOSP
- Access to the RHOSP APIs, using the RHOSP authentication method you will use for cluster configuration, as described in Authentication methods for RHOSP
- The cluster property **stonith-enabled** set to **true**, which is the default value. Red Hat does not support clusters when fencing is disabled, as it is not suitable for a production environment. Run the following command to display the values of the cluster properties.

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

Procedure

Complete the following steps from any node in the cluster.

1. Determine the UUID for each node in your cluster.
   The following command displays the full list of all of the RHOSP instance names within the **ha-example** project along with the UUID for the cluster node associated with that RHOSP instance. The node host name might not match the RHOSP instance name.

```
# 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. Create the fencing device, using the **pcmk_host_map parameter** to map each node in the cluster to the UUID for that node.

   a. The following command creates a **fence_openstack** fencing device for a 3-node cluster, using a **clouds.yaml** configuration file for authentication. For the **cloud= parameter**, specify the name of the cloud in your clouds.yaml` file.

```
# 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. The following command creates a **fence_openstack** fencing device, using an OpenRC environment script for authentication.

```
# 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. The following command creates a **fence_openstack** fencing device, using a user name and password for authentication. The authentication parameters, including **username**, **password**, **project_name**, and **auth_url**, are provided by the RHOSP administrator.

```
# 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"
```

Verification

1. From one node in the cluster, fence a different node in the cluster and check the status of the cluster to see whether the node you fenced is offline.

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

2. Restart the node that you fenced and check the status to verify that the node started.

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

CHAPTER 7. CONFIGURING HA CLUSTER RESOURCES ON RED HAT OPENSTACK PLATFORM

Use the following resource agents when configuring resources for an HA cluster on RHOSP:

- **openstack-info**: You must configure an **openstack-info** resource as a cloned resource for your cluster in order to run any other RHOSP-specific resource agent other than the **fence_openstack** fence agent.
- **openstack-virtual-ip**: Configures a virtual IP address resource
- **openstack-floating-ip**: Configures a floating IP address resource
- **openstack-cinder-volume**: Configures a block storage resource

When configuring other cluster resources, use the standard Pacemaker resource agents.

7.1. CONFIGURING AN OPENSTACK-INFO RESOURCE (REQUIRED)

An **openstack-info** resource is required in order to run any other RHOSP-specific resource agent other than the **fence_openstack** fence agent.

The following procedure creates an **openstack-info** resource. This procedure uses a **clouds.yaml** file for RHOSP authentication.

Prerequisites

- A configured HA cluster running on RHOSP
- Access to the RHOSP APIs, using the RHOSP authentication method you will use for cluster configuration, as described in Authentication methods for RHOSP

Procedure

Complete the following steps from any node in the cluster.

1. To view the options for the **openstack-info** resource agent, run the following command.

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

2. Create the **openstack-info** resource as a clone resource. In this example, the resource is also named **openstack-info**. This example uses a **clouds.yaml** configuration file and the **cloud=** parameter is set to the name of the cloud in your **clouds.yaml** file.

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

3. Check the cluster status to verify that the resource is running

```
# pcs status

Full List of Resources:

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

7.2. CONFIGURING A VIRTUAL IP ADDRESS IN AN HA CLUSTER ON RED HAT OPENSTACK PLATFORM

The following procedure creates an RHOSP virtual IP address resource for an HA cluster on an RHOSP platform. This procedure uses a **clouds.yaml** file for RHOSP authentication.

The RHOSP virtual IP resource operates in conjunction with an **IPaddr2** cluster resource. When you configure an RHOSP virtual IP address resource, the resource agent ensures that the RHOSP infrastructure associates the virtual IP address with a cluster node on the network. This allows an IPadr2 resource to function on that node.

Prerequisites

- A configured HA cluster running on RHOSP
- An assigned IP address to use as the virtual IP address
- Access to the RHOSP APIs, using the RHOSP authentication method you will use for cluster configuration, as described in Authentication methods for RHOSP

Procedure

Complete the following steps from any node in the cluster.

1. To view the options for the **openstack-virtual-ip** resource agent, run the following command.

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

2. Run the following command to determine the subnet ID for the virtual IP address you are using. In this example, the virtual IP address is 172.16.0.119.

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

3. Create the RHOSP virtual IP address resource.
   The following command creates an RHOSP virtual IP address resource for an IP address of 172.16.0.119, specifying the subnet ID you determined in the previous step.

```
# 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. Configure ordering and location constraints to ensure that the **openstack-info** resource starts before the virtual IP address resource and that the Virtual IP address resource runs on the same node as the **openstack-info** resource.

```
# 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. Create an **IPaddr2** resource for the virtual IP address.

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

6. Configure ordering and location constraints to ensure that the **openstack-virtual-ip** resource starts before the **IPaddr2** resource and that the **IPaddr2** resource runs on the same node as the **openstack-virtual-ip** resource.

```
# 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
```

Verification

1. Verify the resource constraint configuration.

```
# 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. Check the cluster status to verify that the resources are running.

```
# 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. CONFIGURING A FLOATING IP ADDRESS IN AN HA CLUSTER ON RED HAT OPENSTACK PLATFORM

The following procedure creates a floating IP address resource for an HA cluster on RHOSP. This procedure uses a **clouds.yaml** file for RHOSP authentication.

Prerequisites

- A configured HA cluster running on RHOSP
- An IP address on the public network to use as the floating IP address, assigned by the RHOSP administrator
- Access to the RHOSP APIs, using the RHOSP authentication method you will use for cluster configuration, as described in Authentication methods for RHOSP

Procedure

Complete the following steps from any node in the cluster.

1. To view the options for the **openstack-floating-ip** resource agent, run the following command.

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

2. Find the subnet ID for the address on the public network that you will use to create the floating IP address resource.

   a. The public network is usually the network with the default gateway. Run the following command to display the default gateway address.

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

   b. Run the following command to find the subnet ID for the public network. This command generates a table with ID and Subnet headings.

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

3. Create the floating IP address resource, specifying the public IP address for the resource and the subnet ID for that address. When you configure the floating IP address resource, the resource agent configures a virtual IP address on the public network and associates it with a cluster node.

```
# 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. Configure an ordering constraint to ensure that the **openstack-info** resource starts before the floating IP address resource.

```
# 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. Configure a location constraint to ensure that the floating IP address resource runs on the same node as the **openstack-info** resource.

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

Verification

1. Verify the resource constraint configuration.

```
# 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. Check the cluster status to verify that the resources are running.

```
# 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. CONFIGURING A BLOCK STORAGE RESOURCE IN AN HA CLUSTER ON RED HAT OPENSTACK PLATFORM

The following procedure creates a block storage resource for an HA cluster on RHOSP. This procedure uses a **clouds.yaml** file for RHOSP authentication.

Prerequisites

- A configured HA cluster running on RHOSP
- A block storage volume created by the RHOSP administrator
- Access to the RHOSP APIs, using the RHOSP authentication method you will use for cluster configuration, as described in Authentication methods for RHOSP

Procedure

Complete the following steps from any node in the cluster.

1. To view the options for the **openstack-cinder-volume** resource agent, run the following command.

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

2. Determine the volume ID of the block storage volume you are configuring as a cluster resource. Run the following command to display a table of available volumes that includes the UUID and name of each volume.

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

If you already know the volume name, you can run the following command, specifying the volume you are configuring. This displays a table with an ID field.

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

3. Create the block storage resource, specifying the ID for the volume.

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

4. Configure an ordering constraint to ensure that the **openstack-info** resource starts before the block storage resource.

```
# 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. Configure a location constraint to ensure that the block storage resource runs on the same node as the **openstack-info** resource.

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

Verification

1. Verify the resource constraint configuration.

```
# 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. Check the cluster status to verify that the resource is running.

```
# 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
```