Red Hat OpenStack Platform 12

High Availability for Compute Instances

Configure High Availability for Compute Instances

Last Updated: 2018-10-01

Red Hat OpenStack Platform 12 High Availability for Compute Instances

Configure High Availability for Compute Instances

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

A guide for configuring High Availability for Compute Instances (Instance HA) in Red Hat OpenStack Platform. This document focuses on enabling Instance HA through Ansible.

Table of Contents

CHAPTER 1. OVERVIEW

This guide describes how to implement *Instance High Availability (Instance HA)*. Instance HA allows Red Hat OpenStack Platform to automatically evacuate and re-spawn instances on a different Compute node when their host Compute node fails.

The evacuation process that is triggered by Instance HA is similar to what users can do manually, as described in Evacuate Instances.

Instance HA works on shared storage or local storage environments, which means that evacuated instances maintain the same network configuration (static IP, floating IP, and so on) and the same characteristics inside the new host, even if they are spawned from scratch.

Instance HA is managed by the following resource agents:

Agent name	Name inside cluster	Role
`fence_compute`	`fence-nova`	Marks a Compute node for evacuation when the node becomes unavailable.
`NovaEvacuate`	`nova-evacuate`	Evacuates instances from failed nodes. This agent runs on one of the Controller nodes.
`Dummy`	`compute-unfence-trigger`	Releases a fenced node and enables the node to run instances again.

This guide describe how to use the Ansible automation framework to enable Instance HA. For more information about Ansible, see Ansible Documentation.

NOTE

For a high-level description of how Instance HA detects and evacuates instances from failed hosts, see Chapter 2, *How Instance HA Works*.

CHAPTER 2. HOW INSTANCE HA WORKS

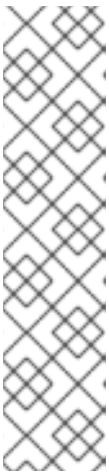
OpenStack uses Instance HA to automate the process of evacuating instances from a Compute node when that node fails. The following procedure describes the sequence of events that are triggered when a Compute node fails.

1. At the time of failure, the **IPMI** agent performs *first-layer fencing* and physically resets the node to ensure that it is powered off. Evacuating instances from online Compute nodes might result in data corruption or in multiple identical instances running on the overcloud. When the node is powered off, it is considered *fenced*.

2. After the physical IPMI fencing, the **fence-nova** agent performs *second-layer fencing* and marks the fenced node with the **"evacuate=yes"** cluster per-node attribute. To do this, the agent runs the following command:

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

   Where *FAILEDHOST* is the hostname of the failed Compute node.

   NOTE

   By default, all instances are to be evacuated, but it is also possible to tag images or flavors for evacuation.

   To tag an image:

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

   To tag a flavor:

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

3. The **nova-evacuate** agent continually runs in the background, periodically checking the cluster for nodes with the **"evacuate=yes"** attribute. When **nova-evacuate** detects that the fenced node contains this attribute, the agent starts evacuating the node using the process described in Evacuate Instances.

4. While the failed node is booting up from the IPMI reset, the **nova-compute** process on that node will start automatically. Because the node was fenced earlier, it will not be able to run any new instance until Pacemaker *unfences* it.

5. When Pacemaker sees that the Compute node is online again, it tries to start the **compute-unfence-trigger** resource on the node, reverting the force-down API call and setting the node as enabled again.

CHAPTER 3. ENVIRONMENT PREREQUISITES AND LIMITATIONS

WARNING

When Instance HA is enabled, you cannot perform overcloud upgrade or scale-up operations. Any attempts to do this will fail. This limitation applies to minor and major upgrades. Before upgrading or scaling your overcloud, you must disable Instance HA first. For instructions, see Chapter 6, *Disabling Instance HA*.

To enable Instance HA, your Red Hat OpenStack Platform overcloud must meet the following requirements:

- The environment is deployed by Red Hat OpenStack Platform director. See Director Installation and Usage for details.
- Fencing is already manually enabled on the control plane.
- The following package versions are installed on all nodes:
  - **`fence-agents-4.0.11-86.el7.x86_64`** (or later)
  - **`pacemaker-1.1.18-11.el7.x86_64`** (or later)
  - **`resource-agents-3.9.5-124.el7.x86_64`** (or later)
- The environment can tolerate a full outage of the Compute and the Control planes.
- Shared storage is enabled in the environment for ephemeral and block storage. See Section 3.1, “Considerations for Shared Storage” for considerations.
- The Message Broker (AMQP) recognizes the hostname of each Compute node as valid. To check the hostname of a Compute node, run the following command:

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

- Each Compute node can reach the endpoint that is set in the **`$OS_AUTH_URL`** environment variable.
- The **`$OS_AUTH_URL`** environment variable must be set to one of the following destinations:
  - The authentication services of the overcloud (which requires access to the external network), or
  - The internal authentication URL

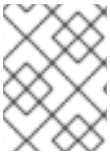
3.1. CONSIDERATIONS FOR SHARED STORAGE

Typically, Instance HA requires that you configure shared storage for disk images of instances.

Therefore, if you attempt to use the **`no-shared-storage`** option, you might receive an **`InvalidSharedStorage`** error during evacuation, and the instances will not boot on another Compute node.

However, if all your instances are configured to boot from an OpenStack Block Storage (**`cinder`**) volume, you do not need to configure shared storage for the disk image of instances, and you can still evacuate all instances using the **`no-shared-storage`** option.

During evacuation, if your instances are configured to boot from a Block Storage volume, any evacuated instances should boot from the same volume on another Compute node. As a result, the evacuated instances immediately restart their jobs because the OS image and the application data are stored on the OpenStack Block Storage volume..



NOTE

The Ansible-based deployment procedure in this guide supports installation with **`no-shared-storage`** option.

CHAPTER 4. DEPLOYING INSTANCE HA

The following sections describe how to use the Ansible automation framework to enable Instance HA. For more information about Ansible, see Ansible Documentation.

4.1. CREATING THE REQUIRED ANSIBLE CONFIGURATION FILES

Before you enable Instance HA through Ansible, you need to create an *inventory file* and an *SSH arguments file*. These files pass the Ansible variables that you will need to prepare the undercloud for Instance HA, as described in Section 4.2, “Preparing the Undercloud”.

Inventory File

The inventory file lists the different target hosts for the ansible playbooks. It is divided into two sections:

- List of all nodes by name, along with the hostname, username, and private key file that Ansible should use for each playbook command. For example:

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

- List of headings that represent node types, where each type lists the nodes that belong to it. The node types are **`compute`**, **`undercloud`**, **`overcloud`**, or **`controller`**.

The following procedure describes how to set up the inventory file.

1. Generate a complete inventory of all nodes in the undercloud and the overcloud with the following command:

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

   This command generates a detailed and updated inventory in JSON format. For more information, see Running Ansible Automation.

2. Create the inventory file with the name ***hosts*** in ***/home/stack/***. The following example shows the required format to use when creating the file:

```
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
```

SSH Arguments File

The SSH arguments file passes the necessary credentials and authentication settings that are needed by Ansible to run the playbooks on each target host.

Create the SSH arguments file from */**home**/**stack*** with the following commands:

```
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
stack@director $ source /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
```

These commands create an SSH arguments file named ***ssh.config.ansible***, which contains host-specific connection options for each overcloud node. For example:

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

4.2. PREPARING THE UNDERCLOUD

After you create the *inventory file* and *SSH arguments file*, as described in Section 4.1, “Creating the Required Ansible Configuration Files”, you can prepare the undercloud for Instance HA.

1. Log in to the undercloud as the **stack** user.

2. Clone the upstream **tripleo-ha-utils** git repository.

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

   This repository contains the playbooks, roles, and other utilities necessary to enable and test Instance HA with Ansible.

3. Create the */home/stack/ansible.cfg* file with the following content:

```
[defaults]
roles_path = /home/stack/tripleo-ha-utils/roles
```

4. Export the *ansible.cfg* file, the inventory file, and the SSH arguments file to the following environment variables:

```
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"
```

5. Make sure that the node definition template of the overcloud is located in */home/stack/*. By default, the template file is named *instackenv.json*. For more information about the node definition template, see Registering Nodes for the Overcloud.

4.3. ENABLING INSTANCE HA

After you prepare the undercloud you can run the Ansible playbooks that you downloaded and extracted, as described in Section 4.2, “Preparing the Undercloud”. These playbooks allow you to enable Instance HA with or without configuring STONITH for the Controller and Compute nodes. For more information about STONITH, see Fencing the Controller Nodes.

Enabling Instance HA with STONITH

To enable Instance HA and configure STONITH for both Controller and Compute nodes, run the following command:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12"
```

By default, the playbook installs the **instance-ha** solution with shared storage enabled. If your overcloud does not use shared storage, run the command with the **instance_ha_shared_storage=false** option:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12" -e instance_ha_shared_storage=false
```

NOTE

See Section 3.1, “Considerations for Shared Storage” for more information about shared storage in Instance HA.

Enabling Instance HA without STONITH

To enable Instance HA without configuring STONITH for the Controller and Compute nodes, run the following command:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12" -e stonith_devices="none"
```

Enabling Instance HA with STONITH only on the Compute nodes

To enable Instance HA and configure STONITH only on the Compute nodes (for example, if STONITH is already configured on the Controller nodes), run the following command:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12" -e stonith_devices="computes"
```

CHAPTER 5. TESTING EVACUATION WITH INSTANCE HA

WARNING

The following procedure involves deliberately crashing a Compute node. Doing this forces the automated evacuation of instances through Instance HA.

1. Boot one or more instances on the overcloud before crashing the Compute node that hosts the instances to test.

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

2. Log in to the Compute node that hosts the instances, using the **compute-n** format.

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

3. Crash the Compute node.

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

4. Wait a few minutes and then verify that these instances re-spawned on another Compute nodes.

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

CHAPTER 6. DISABLING INSTANCE HA

When Instance HA is enabled, upgrade or scale-up operations are not possible. Any attempts to do this will fail. This limitation applies to minor and major upgrades. Before upgrading or scaling your overcloud, you must disable Instance HA first.

To disable Instance HA, run the following command as the **stack** user on the undercloud:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12" -e instance_ha_action="uninstall"
```

If you used the **stonith_devices** option when you enabled Instance HA, you need to specify this option when you disable Instance HA. For example, if your Instance HA configuration excludes STONITH devices, use the following command syntax:

```
stack@director $ ansible-playbook /home/stack/tripleo-ha-
utils/playbooks/overcloud-instance-ha.yml \
-e release="rhos-12" -e instance_ha_action="uninstall" -e
stonith_devices="none"
```

APPENDIX A. STEP-BY-STEP ANSIBLE PLAYBOOK FOR CONFIGURING INSTANCE HA

The Ansible-based solution that is described in this guide provides an automated and supported way to configure Instance HA. For reference, this appendix describes the steps that are automated by the playbook.

1. Create an authentication key on the director node to use with the **pacemaker-remote** service.

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

2. Copy the authentication key to the Compute and Controller nodes:

```
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
```

3. On each Compute node, enable the **pacemaker-remote** service and configure the firewall.

```
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
```

4. Confirm that the required versions of the **pacemaker** (**1.1.18-11.el7.x86_64**) and **resource-agents** (**3.9.5-124.el7.x86_64**) packages are installed on the Controller and Compute nodes.

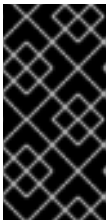
```
heat-admin@controller-n $ sudo rpm -qa | egrep '(pacemaker|resource-
agents)'
```

5. Create a **NovaEvacuate** active/passive resource with the ***overcloudrc.v3*** file, which provides the **auth_url**, **username**, **tenant** and **password** values.

```
stack@director # scp overcloudrc.v3 heat-admin@controller-1:~/
heat-admin@controller-1 $ . ~/overcloudrc.v3
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 project_domain=$OS_PROJECT_DOMAIN_NAME
user_domain=$OS_USER_DOMAIN_NAME
```

NOTE

If you are not using shared storage, include the **no_shared_storage=1** option. See Section 3.1, “Considerations for Shared Storage” for more information.



IMPORTANT

As mentioned in Chapter 3, *Environment Prerequisites and Limitations*, the **$OS_AUTH_URL** destination must be the accessible to each Compute node. This environment variable should be set to either the overcloud's authentication service or the internal authentication URL.

6. Make sure that **nova-evacuate** runs only on non-Compute nodes.

```
heat-admin@controller-1 $ pcs constraint location nova-evacuate rule
resource-discovery=never score=-INFINITY osprole eq compute
```

7. Confirm that **nova-evacuate** is started after the floating IP resources, OpenStack Image Service (glance), OpenStack Networking (neutron), and Compute (nova) services.

```
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. Create a list of the current controllers from the **cibadmin** data.

```
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. Use the list you created in the previous step to tag these nodes as controllers with the **osprole=controller** property.

```
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
```

The newly assigned roles should appear in the **Node attributes** section.

10. Create a list of STONITH devices that are already present in the environment.

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

11. Tag the control plane services to make sure they only run on listed Controller nodes and skip any listed STONITH devices.

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

12. Set the **requires** property to **fencing** as the default for all resources. This operation is required to enable the **unfence** mechanism.

```
heat-admin@controller-1 $ sudo pcs resource defaults
requires=fencing
```

13. Create a separate **fence-nova** STONITH device.

```
heat-admin@controller-1 $ . overcloudrc.v3
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 \
                                domain=localdomain record-only=1 \
                                meta provides=unfencing \
                                --force
```

NOTE

- This command assumes that you are using the default cloud domain name **localdomain**. If you are using a custom cloud domain name, set it as the value of the **domain=** parameter.
- If you are not using shared storage, include the **no_shared_storage=1** option. See Section 3.1, “Considerations for Shared Storage” for more information.

14. Create a pacemaker constraint for **fence-nova** to restrict the service to Controller nodes and set **resource-discovery** to **never**.

```
heat-admin@controller-1 $ pcs constraint location fence-nova rule
resource-discovery=never score=0 osprole eq controller
```

15. Add a STONITH device on each Compute nodes. This command should be run separately on each Compute node.

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

Where:

- **N**. Identifying number of each compute node. For example, **ipmilan-overcloud-compute-1**, **ipmilan-overcloud-compute-2**, and so on.
- **IPADDR**. IP address of the IPMI interface.
- **IPMILANUSER**. User name of the IPMI device.
- **IPMILANPW**. Password of the IPMI device.

16. Make sure that the Compute nodes can recover after fencing.

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

17. Create a pacemaker remote resource for each Compute node and set **osprole** to **compute**.

```
heat-admin@controller-1 $ sudo pcs resource create overcloud-
compute-n ocf:pacemaker:remote reconnect_interval=240 op monitor
interval=20"
heat-admin@controller-1 $ sudo pcs property set --node overcloud-
compute-n osprole=compute"
```

18. Create Compute node resources and set the STONITH **level 1** to include the physical fence device of the nodes and the **fence-nova** service. Run this command separately on each Compute node.

```
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
```

Replace **N** with the identifying number of each Compute node, for example, **overcloud-compute-1**, **overcloud-compute-2**, and so on. Use these identifying numbers to match each Compute node with the STONITH devices that you created earlier, for example, match **overcloud-compute-1** with **ipmilan-overcloud-compute-1**.

19. After you complete this procedure, allow some time for the environment to settle and then cleanup any failed resources.

```
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
```