Red Hat OpenStack Platform 8

High Availability for Compute Instances

Configure High Availability for Compute Instances

Last Updated: 2018-08-24

Red Hat OpenStack Platform 8 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 re-spawn instances on a different Compute node when their host Compute node breaks.

Instance HA automates the evacuation of an instance whenever its host Compute node fails. The evacuation process triggered by Instance HA is similar to what users can do manually, as described in Evacuate Instances. Instance HA works on shared storage and local storage environments, which means that evacuated instances will maintain the same network configuration (static ip, floating ip, and so on) and characteristics inside the new host, even if they are spawned from scratch.

Instance HA is managed by three 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, and runs on one of the Controller nodes
nova-compute-wait	nova-compute-checkevacuate	Restarts Compute services on an instance once it is fully evacuated to a functional Compute host.

This guide focuses on enabling Instance HA on an overcloud through Ansible. To streamline the process, this guide also contains a pre-packaged TAR archive containing the necessary playbooks for doing so.

NOTE

For a brief description of the detection and evacuation process performed by Instance HA, see Appendix A, *Automated Evacuation Through Instance HA*.

CHAPTER 2. ENVIRONMENT REQUIREMENTS AND ASSUMPTIONS

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

- The environment was deployed using Red Hat OpenStack Platform director. See Director Installation and Usage for details.
- Fencing has already manually been enabled on the control plane.
- The following packages are installed on all nodes:
  - **`fence-agents-4.0.11-66.el7_4.3`** (or greater)
  - **`pacemaker-1.1.16-12.el7.x86_64`** (or greater)
  - **`resource-agents-3.9.5-105.el7.x86_64`** (or greater)
- The environment can tolerate a full outage of both Compute and Control planes.
- Shared storage is enabled within the environment for ephemeral and block storage. See Section 2.1, “Exceptions for Shared Storage” for related information.
- The Message Broker (AMQP) recognizes each Compute node’s hostname as valid. To check the hostname of a Compute node:

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

- Each Compute node should be able to reach the endpoint set in **$OS_AUTH_URL**. In addition, this environment variable must be set to either:
  - The overcloud’s authentication service (which requires access to the external network), or
  - The internal authentication URL.

WARNING

When Instance HA is enabled, overcloud upgrade or scale-up operations are not possible. Any attempts to do so will fail. This includes both minor and major upgrades.

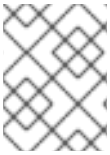
Before upgrading or scaling your overcloud, disable Instance HA first. For instructions, see Chapter 5, *Rollback*.

2.1. EXCEPTIONS FOR SHARED STORAGE

Typically, enabling Instance HA requires shared storage. If you attempt to use the **`no-shared-storage`** option, you are likely to receive an **`InvalidSharedStorage`** error during evacuation, and

instances will not power up on the other node. However, if all your instances are configured to boot up from a Block Storage (**cinder**) volume, then you will not need shared storage to store the disk image of instances; you will be able to 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 can be expected to boot up from the same volume, but on another Compute node. As a result, the evacuated instances can immediately restart their jobs, as the OS image and application data are kept on the Block Storage volume.



NOTE

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

CHAPTER 3. DEPLOYMENT

The following procedure involves the use of *Ansible* to enable Instance HA. For more information about Ansible, see Ansible Documentation.

3.1. CREATING THE NECESSARY ANSIBLE CONFIGURATION FILES

Enabling Instance HA through Ansible requires an *inventory file* and *SSH arguments file*. Both files pass the Ansible variables necessary for implementing Instance HA on your overcloud.

Inventory File

The inventory file lists the different target hosts for the ansible playbooks. It is divided into two sections: the first section lists each node (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
```

The second section lists each node under the following headings (or *node types*): **`compute`**, **`undercloud`**, **`overcloud`**, or **`controller`**.

Create an inventory file named **`/home/stack/hosts`**. The following sample demonstrates the syntax required for this:

```
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
```

To generate a complete inventory of all hosts in both undercloud and overcloud, run the following command:

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

This command will generate a detailed and updated inventory in JSON format. See Running Ansible Automation for more details.

SSH Arguments File

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

Create an SSH arguments file using the following commands (from **/home/stack**):

```
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 will result in the creation of an SSH arguments file named **/home/stack/ssh.config.ansible**, which will contain 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
```

3.2. PREPARING THE UNDERCLOUD

After creating the *inventory file* and *SSH arguments file* (from Section 3.1, “Creating the Necessary Ansible Configuration Files”), you can now prepare the overcloud for Instance HA:

1. Log in to the undercloud as the **stack** user.
2. Download this TAR archive to **/home/stack/**. It contains the playbooks, roles, and other utilities necessary for enabling Instance HA through Ansible.

NOTE

The TAR archive provided here is a tested and modified version of an upstream GIT repository. To clone this repository, run:

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

This repository may be updated without notice, and therefore may be different from the archive available in this step.

3. Extract the TAR archive:

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

4. Create **/home/stack/ansible.cfg** with the following contents:

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

5. Export the **ansible.cfg**, hosts (the inventory file), and **ssh.config.ansible** (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"
```

6. Ensure that the node definition template of the overcloud (by default, **instackenv.json**) is located in **/home/stack/**. For more information about the node definition template, see Registering Nodes for the Overcloud.

3.3. ENABLING INSTANCE HA

Once the undercloud is fully prepared, you can now run the prescribed playbooks you downloaded and extracted in Section 3.2, “Preparing the Undercloud”. These playbooks allow you to enable Instance HA with or without configuring STONITH for Controller and Compute nodes. For more information about STONITH, see Fencing the Controller Nodes.

In each of the following commands, replace *RELEASE* with the corresponding code for your version of Red Hat OpenStack Platform — namely, **rhos-8**.

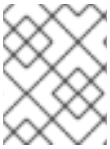
To enable Instance HA and configure STONITH for both Controller and Compute nodes:

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

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

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

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



NOTE

See Section 2.1, “Exceptions for Shared Storage” for more information about shared storage in Instance HA.

To enable Instance HA *without* configuring STONITH for both Controller and Compute nodes:

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

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

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

CHAPTER 4. TESTING

WARNING

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

1. Boot one or more instances on the overcloud before crashing the Compute node hosting the instances in question:

```
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 hosting the instances you just booted (as in, ***compute-n***):

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

3. Crash the node:

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

4. After several minutes, verify that these instances restarted on online Compute nodes. To check:

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

CHAPTER 5. ROLLBACK

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

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

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

Replace *RELEASE* with the corresponding code for your version of Red Hat OpenStack Platform — namely, **rhos-8**.

If you used the **stonith_devices** option when you enabled Instance HA , you need to specify the same option during rollback. For example, if your Instance HA configuration has STONITH disabled (as in Section 3.3, “Enabling Instance HA”), use:

```
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”
```

APPENDIX A. AUTOMATED EVACUATION THROUGH INSTANCE HA

With Instance HA, OpenStack automates the process of evacuating instances from a Compute node when that node fails. The following process describes the sequence of events triggered in the event of a Compute node failure.

1. When a Compute node fails, the **IPMI** agent performs *first-level fencing* and physically resets the node to ensure that it is powered off. Evacuating instances from online Compute nodes could result in data corruption or multiple identical instances running on the overcloud. Once the node is powered off, it is considered *fenced*.

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

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

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

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

4. Meanwhile, while the failed node is booting up from the IPMI reset, the **nova-compute-checkevacuate** agent will wait (by default, for 120 seconds) before checking whether **nova-evacuate** is finished with evacuation. If not, it will check again after the same time interval.

5. Once **nova-compute-checkevacuate** verifies that the instances are fully evacuated, it triggers another process to make the fenced node available again for hosting instances.

APPENDIX B. MANUAL PROCEDURE AUTOMATED BY ANSIBLE PLAYBOOKS

The Ansible-based solution provided by this document is designed to automate a manual procedure for configuring Instance HA in a supported manner. For reference, this appendix provides the steps automated by the solution.

1. Begin by disabling *libvirtd* and all OpenStack services on the Compute nodes:

```
heat-admin@compute-n # sudo openstack-service stop
heat-admin@compute-n # sudo openstack-service disable
heat-admin@compute-n # sudo systemctl stop libvirtd
heat-admin@compute-n # sudo systemctl disable libvirtd
```

2. Create an authentication key for use with *pacemaker-remote*.
Perform this step on one of the Compute nodes:

```
heat-admin@compute-1 # sudo mkdir -p /etc/pacemaker/
heat-admin@compute-1 # sudo dd if=/dev/urandom of=/etc/pacemaker/authkey
bs=4096 count=1
heat-admin@compute-1 # sudo cp /etc/pacemaker/authkey ./
heat-admin@compute-1 # sudo chown heat-admin:heat-admin authkey
```

3. Copy this key to the director node, and then to the remaining Compute and Controller nodes:

```
stack@director # scp authkey 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 chown root:haclient /etc/pacemaker/authkey
```

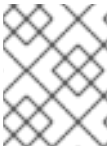
4. Enable *pacemaker-remote* on all Compute nodes:

```
heat-admin@compute-n # sudo systemctl enable pacemaker_remote
heat-admin@compute-n # sudo systemctl start pacemaker_remote
```

5. Confirm that the required versions of the pacemaker (**`1.1.12-22.el7_1.4.x86_64`**) and resource-agents (**`3.9.5-40.el7_1.5.x86_64`**) packages are installed on the controller and Compute nodes:

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

6. Apply the following constraint workarounds required for BZ#1257414.



NOTE

This issue has been addressed in RHSA-2015:1862, and might not be required for your environment.

```
heat-admin@controller-1 # sudo pcs constraint order start openstack-nova-
novncproxy-clone then openstack-nova-api-clone
heat-admin@controller-1 # sudo pcs constraint order start rabbitmq-clone
then openstack-keystone-clone
```

```
heat-admin@controller-1 # sudo pcs constraint order promote galera-master
then openstack-keystone-clone
heat-admin@controller-1 # sudo pcs constraint order start haproxy-clone
then openstack-keystone-clone
heat-admin@controller-1 # sudo pcs constraint order start memcached-clone
then openstack-keystone-clone
heat-admin@controller-1 # sudo pcs constraint order promote redis-master
then start openstack-ceilometer-central-clone require-all=false
heat-admin@controller-1 # sudo pcs resource defaults resource-
stickiness=INFINITY
```

7. Create a *NovaEvacuate* active/passive resource using the *overcloudrc* file to provide the **auth_url**, **username**, **tenant** and **password** values:

```
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 \ 1
```

1 If you are not using shared storage, include the **no_shared_storage=1** option. See Section 2.1, "Exceptions for Shared Storage" for more information.

8. Confirm that *nova-evacuate* is started after the floating IP resources, and the Image Service (glance), OpenStack Networking (neutron), 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
heat-admin@controller-1 # for i in openstack-glance-api-clone neutron-
metadata-agent-clone openstack-nova-conductor-clone; do sudo pcs
constraint order start $i then nova-evacuate require-all=false ; done
```

9. Disable all OpenStack resources across the control plane:

```
heat-admin@controller-1 # sudo pcs resource disable openstack-keystone --
wait=540
```

The timeout used here (*--wait=540*) is only used as an example. Depending on the time needed to stop the Identity Service (and on the power of your hardware), you can consider increasing the timeout period.

10. Create a list of the current controllers using **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
```

11. Use this list 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
```

12. Build a list of *stonith* devices already present in the environment:

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

13. Tag the control plane services to make sure they only run on the controllers identified above, skipping any stonith devices listed:

```
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
```

14. Begin to populate the Compute node resources within *pacemaker*, starting with *neutron-openvswitch-agent*:

```
heat-admin@controller-1 # sudo pcs resource create neutron-openvswitch-
agent-compute systemd:neutron-openvswitch-agent op start timeout 200s stop
timeout 200s --clone interleave=true --disabled --force
heat-admin@controller-1 # sudo pcs constraint location neutron-
openvswitch-agent-compute-clone rule resource-discovery=exclusive score=0
osprole eq compute
heat-admin@controller-1 # sudo pcs constraint order start neutron-server-
clone then neutron-openvswitch-agent-compute-clone require-all=false
```

Then the compute *libvirtd* resource:

```
heat-admin@controller-1 # sudo pcs resource create libvirtd-compute
systemd:libvirtd op start timeout 200s stop timeout 200s --clone
interleave=true --disabled --force
heat-admin@controller-1 # sudo pcs constraint location libvirtd-compute-
clone rule resource-discovery=exclusive score=0 osprole eq compute
heat-admin@controller-1 # sudo pcs constraint order start neutron-
openvswitch-agent-compute-clone then libvirtd-compute-clone
heat-admin@controller-1 # sudo pcs constraint colocation add libvirtd-
compute-clone with neutron-openvswitch-agent-compute-clone
```

Then the *openstack-ceilometer-compute* resource:

```
heat-admin@controller-1 # sudo pcs resource create ceilometer-compute
systemd:openstack-ceilometer-compute op start timeout 200s stop timeout
```

```
200s --clone interleave=true --disabled --force
heat-admin@controller-1 # sudo pcs constraint location ceilometer-compute-
clone rule resource-discovery=exclusive score=0 osprole eq compute
heat-admin@controller-1 # sudo pcs constraint order start openstack-
ceilometer-notification-clone then ceilometer-compute-clone require-
all=false
heat-admin@controller-1 # sudo pcs constraint order start libvirtd-
compute-clone then ceilometer-compute-clone
heat-admin@controller-1 # sudo pcs constraint colocation add ceilometer-
compute-clone with libvirtd-compute-clone
```

Then the *nova-compute* resource:

```
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
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 constraint order start openstack-nova-
conductor-clone then nova-compute-checkevacuate-clone require-all=false
heat-admin@controller-1 # sudo pcs resource create nova-compute
systemd:openstack-nova-compute --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
heat-admin@controller-1 # sudo pcs constraint order start libvirtd-
compute-clone then nova-compute-clone
heat-admin@controller-1 # sudo pcs constraint colocation add nova-compute-
clone with libvirtd-compute-clone
```

15. Add stonith devices for the Compute nodes. Run the following command for each Compute node:

```
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;
```

Where:

- *N* is the identifying number of each compute node (for example, **ipmilan-overcloud-compute-1**, **ipmilan-overcloud-compute-2**, and so on).
- *IPMILANUSER* and *IPMILANPW* are the username and password to the IPMI device.

16. Create a seperate *fence-nova* stonith device:

```
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
```

17. Make certain the Compute nodes are able to recover after fencing:

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

18. Create Compute node resources and set the stonith *level 1* to include both the nodes's physical fence device and *fence-nova*. Run the following commands for 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 nodes with the stonith devices created earlier (for example, **overcloud-compute-1** and **ipmilan-overcloud-compute-1**).

19. Enable the control and Compute plane services:

```
heat-admin@controller-1 # sudo pcs resource enable openstack-keystone
heat-admin@controller-1 # sudo pcs resource enable neutron-openvswitch-
agent-compute
heat-admin@controller-1 # sudo pcs resource enable libvirtd-compute
heat-admin@controller-1 # sudo pcs resource enable ceilometer-compute
heat-admin@controller-1 # sudo pcs resource enable nova-compute-
checkevacuate
heat-admin@controller-1 # sudo pcs resource enable nova-compute
```

20. Allow some time for the environment to settle before cleaning up 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
```