ERICSSON

Customer Acceptance Test Specification

Cloud Execution Environment

TEST SPECIFICATION

1/192 12-AZE 102 01/5 Uen A

Contents

1 Introduction

This document contains the description of tests that can be carried out on site, as part of the customer acceptance process. The purpose of the tests is to check and verify that Cloud Execution Environment (CEE) funcionalities are fully operational and to provide a basic overview of CEE processes.

The test cases described in this document are also listed in *Customer Acceptance Test Object List*.

Note: This document contains recommended test cases for the CEE R6 release. Alternatively, other test cases can also be used.

1.1 Scope

Table 1 lists the recommended test cases for the below deployments:

- Single server deployment
- BSP multi-server deployment
- Dell multi-server deployment
- HP multi-server deployment

Table 1 Test Cases for Supported Deployments

Test Case	**Single Server Deployment**	**Multi-Server Deployment** **(BSP, Dell, and HP)**
Health Check, see Section 2.1 on page 5	Yes	Yes
Reboot vFuel, see Section 2.2 on page 13	Yes	
Reboot vCIC in Single Server Deployment, see Section 2.3 on page 15	Yes	
Reboot Region Switch in Single Server Deployment, see Section 2.4 on page 17	Yes	
Reboot Single Server Host with VM Configured, see Section 2.5 on page 19	Yes	

Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell, and HP)
Backup and Restore Atlas in Single Server Deployment(1), see Section 2.6 on page 22	Yes	
Reboot Compute Host and Recover vFuel, see Section 2.7 on page 26		Yes
Reboot of Compute Host and Recover vCIC, see Section 2.8 on page 29		Yes
Reboot vCIC Host with Master RabbitMQ Node, see Section 2.9 on page 32		Yes
Reboot Region Switch in multi-Server Deployment, see Section 2.10 on page 39		Yes
Lock or Unlock Blade in a Given Slot, see Section 2.11 on page 44		Yes(2)
Restart Host where Atlas Executes(1), see Section 2.12 on page 47		Yes
Reboot vCIC in multi-Server Deployment, see Section 2.13 on page 49		Yes
Reboot Whole vCIC cluster, see Section 2.14 on page 51		Yes
Restart Atlas VM(1), see Section 2.15 on page 52	Yes	Yes
Reboot VM, Soft, see Section 2.16 on page 54	Yes	Yes
Reboot VM, Hard, see Section 2.17 on page 56	Yes	Yes
VM Stop and VM Start, see Section 2.18 on page 58	Yes	Yes
Backup and Restore Atlas in multi-Server Deployment(1), see Section 2.19 on page 60		Yes

Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell, and HP)
Migrate VM with Nova `migrate` Command, see Section 2.20 on page 64		Yes
Migrate VM to Unspecified Host Using Nova `forcemove` Command, see Section 2.21 on page 71		Yes

(1) This test case can only be performed if Atlas is installed and running.
(2) This test is only applicable for BSP deployments.

1.2 Prerequisites

This section describes the prerequisites that must be fulfilled before the tests can be executed.

1.2.1 Installation and Configuration

Before starting this procedure, ensure that the following conditions are met:

- CEE is installed according to *CEE Installation*.
- The system is hardened according to the *System Hardening Guideline*. To verify, fill in *CEE Hardening Checklist*.

1.2.2 Test Tools

The following tools are needed for test execution:

- A laptop with SSH client application and supported browser is available.
- Connectivity to CEE is available.
- Image file `trusty-server-cloudimg-amd64-disk1.img` is available. The file can be downloaded from http://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-disk1.img.

  Note: This document describes procedures with `trusty-server-cloudimg-amd64-disk1.img`, but a different, locally available image can also be used.

- In case of single server deployments, vFuel is enabled.

1.2.3 Documents

The following documents are needed for the execution of some of the test cases:

- *Atlas Backup*
- *Atlas Restore*
- *Atlas SW Installation*

1.2.4 Alarms

No active alarms can be present prior to the execution of test cases.

2 Test Cases

This section describes the recommended customer acceptance test cases.

2.1 Health Check

This section is applicable for all deployments.

Description

Health check is run to ensure the Data Center (DC) is operational and ready for service. This test case consists of two parts:

- **Verify power supply redundancy to the node:**

  Remove one of the redundant power supplies to the subrack or enclosure and verify that the remaining power supply enables the POD to function normally. Power supply comes from PDUs above the hosts for BSP or EBS. Three switches, turned to `on`, supply power to three cables connected to the right side of the host. Another three switches supply cables connected to the left side of the host. For HP there are two sets of three cables powering each side of the enclosure. Remove one set of three to take away the redundant power supply.

  The infrastructure network solution, including all its components, must be redundant.

- **Check system health:**

  It is recommended to check system health before and after every test case to ensure the state of the DC remains unaltered by the test case executed. For more information refer to *Health Check Procedure*.

Table 2 Health Check

Objective	**Health check is run to ensure a DC is operational and ready for service.**
Precondition	

Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server • Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Verify the power supply redundancy to the node with the following steps:

1. Remove the redundant power supply of the tested subrack or enclosure.

   Result:
   One source of redundant power supply is removed.

2. An alarm is issued informing the operator that the redundant power supply to the host is lost.

   On BSP platforms the alarm comes from BSP, not CEE. It can be accessed from the `dmx-active` SCX using the following commands:

   ```
   plog
   :/persistent/dmx/log
   ls
   ```

   Open `com_alarm.log.0` to see the alarm history. The following alarm indicates loss of power:

   ```
   <Alarm>1;2016-04-19T11:25:59+00:00;ManagedElement=1⇒
   ,Equipment=1,Shelf=1,Slot==
   25,PFM=upper;193;8519688;pfmInterfaceAlarm;78;MAJOR⇒
   ;No power available at power inlet of the indicated ⇒
   PFM.;264;EQUIPMENTALARM;2016-04-19T11:25:59+00:00;MAJOR;NN
   o power available at power inlet of the indicated PFM.⇒
   ;264;0</Alarm>
   ```

Result:
The appropriate alarm is issued.

3. Check that there is no traffic loss due to the loss of redundant power to the node.

   Result:
   There is no traffic loss.

4. Perform a health check.

   Result:
   Health check is successful.

5. Restore the redundant power source to the node.

   Result:
   Redundant power source is restored.

6. The alarm is ceased.

   For BSP or EBS systems the alarm ceasing is issued from BSP.

   Result:
   The appropriate alarm is ceased.

7. Monitor the traffic for three minutes and ensure there are no unexpected events or traffic loss.

   Result:
   There is no traffic loss or unexpected events.

Test system health with the following procedure:

8. Check alarms and alarm history via the GUI interface.

   Result:
   Alarms and history are as expected.

9. Check for presence of crash dumps.

   In case of multi-server deployments, execute the below command on all Virtual Cloud Infrastructure Controller (vCIC) hosts and compute hosts.

   In case of single server deployments, execute the below command on the vCIC host and the compute host.

   ```
   ls -al -R /var/log/crash/
   ```

   Result:
   There are no crash dumps.

10. Verify date and time. Execute the below command from vFuel:

```
for i in `fuel node |egrep '(cic-|compute-)'⇒
|awk '{print $5}'`; do ssh $i hostname 2> ⇒
/dev/null ; ssh $i date 2> /dev/null ; done
```

Result:
Date and time is correct on vCIC (single server deployment) or vCICs (multi-server deployment).

11. In case of multi-server deployments, check Pacemaker for vCIC state and `Cluster Resource` state from a vCIC.

    In case of single server deployments, check Pacemaker for vCIC state and `Cluster Resource` state from the vCIC.

    ```
    crm_mon -1 -rf
    ```

    Specific search for problems:
    ```
    crm_mon -1 -rf | grep FAILED
    crm_mon -1 -rf | grep Stopped
    ```

    Result:
    In case of multi-server deployments, state of the system is `partition with quorum` and all vCICs are online. All services are `started`.

    In case of single-server deployments, the vCIC is online, all services are `started`.

12. In case of multi-server deployments, check CIC maintenance mode from a vCIC.

    In case of single server deployments, check CIC maintenance mode from the vCIC.

    ```
    umm status
    ```

    Result:
    The system is in runlevel `2`:
    `runlevel N 2`

13. Check OpenStack components from any vCIC.

    ```
    neutron device-list
    neutron device-show <device>
    ```

    Note: The above commands do not apply to BSP or single server deployments.

    ```
    nova list --all-tenant
    cinder list --all-tenant
    nova hypervisor-list
    glance image-list
    ```

```
nova service-list
cinder service-list
openstack project list
openstack service list
neutron agent-list
neutron net-list
```

Result:
All neutron devices are `ACTIVE`.

All Virtual Machines (VMs) are `ACTIVE` with power state `Running`.

All `Volumes` are either in-use or available.

All services are up.

All hypervisors are up.

All agents are alive.

At least one tenant, Glance image, and network exists and are listed.

14. Check Ethernet interfaces.

    In case of multi-server deployments, execute the below command on all vCICs and computes.

    In case of single server deployments, execute the below command on the vCIC and the compute node.

    `ovs-appctl bond/show`

    Execute the following checks from vFuel:

    a. Do the following on the vCICs (multi-server deployment) or the vCIC (single server deployment):

```
for nodes in $(fuel node | awk -F "|" '$3 ~ "cic" ⇒
{print $3}')
do
    echo ""
    echo "checking the NIC bonding on host ${node}: "
    echo ""
    ssh -q -oStrictHostKeyChecking=no ⇒
-oUserKnownHostsFile=/dev/null root@${node} ⇒
ovs-appctl bond/show | grep disabled
done
```

b. On compute hosts (multi-server deployment) or compute host (single server deployment) do the following to check for DPDK NIC interfaces:

```
for node in $(fuel node | awk -F "|" '$7 ~ "compute" ⇒
{print $3}')
do
    echo ""
    echo "checking the NIC bonding on host ${node}: "
    echo ""
    ssh -q -oStrictHostKeyChecking=no ⇒
-oUserKnownHostsFile=/dev/null root@${node} ⇒
ovs-appctl bond/show | grep disabled
done
```

Result:

- On vCICs (multi-server deployment) or vCIC (single server deployment): no bonds are disabled
- On compute hosts (multi-server deployment) or compute host (single server deployment): no bonds are disabled

15. In case of multi-server deployments, check disk space on servers hosting vCICs, compute hosts, and the server hosting vFuel. Run the following script on vFuel.

    In case of single server deployments, check disk space on the host. Run the following script on vFuel.

```
for node in $(fuel node | awk -F "|" '$3 ~ "cic" {print $3}');
do
 echo ""
 echo "checking the Disk Space on host ${node}: "
 echo ""
 ssh -q -oStrictHostKeyChecking=no ⇒
-oUserKnownHostsFile=/dev/null root@${node} df -h
done

for node in $(fuel node | awk -F "|" '$7 ~ "compute" ⇒
{print $3}'); do
 echo ""
 echo "checking the Disk Space on host ${node}: "
 echo ""
 ssh -q -oStrictHostKeyChecking=no ⇒
-oUserKnownHostsFile=/dev/null root@${node} df -h
done
```

Result:
Disks are less than 80% occupied.

16. Verify RAM use.

    In case of multi-server deployments, execute the below command on vCICs and compute hosts.

    In case of single server deployments, execute the below command on the host.

    ```
    free -m
    ```

    a. In case of multi-server deployments, do the following to check RAM, on servers hosting the vCICs.

       In case of single server deployments, do the following to check RAM on the host.

       ```
       for node in $(fuel node | awk -F "|" '$3 ~ ⇒
       "cic" {print $3}')
       do
           echo ""
           echo "checking the RAM on host ${node}: "
           echo ""
           ssh -q -oStrictHostKeyChecking=no ⇒
       -oUserKnownHostsFile=/dev/null root@${node} free -m
       done
       ```

    b. On compute hosts (multi-server deployment) or compute host (single server deployment), do the following to check RAM:

       ```
       for node in $(fuel node | awk -F "|" '$7 ~ ⇒
       "compute" {print $3}')
       do
           echo ""
           echo "checking the RAM on host ${node}: "
           echo ""
           ssh -q -oStrictHostKeyChecking=no ⇒
       -oUserKnownHostsFile=/dev/null root@${node} free -m
       done
       ```

    Result:
    The vCICs (multi-server) or vCIC (single server) have at least 20% of RAM free.

    Note: On compute hosts, the use of `ReservedHugePages` for VMs can result in close to 100% RAM usage.

17. In case of multi-server deployments, check Ethernet statistics on all compute nodes.

In case of single server deployments, check Ethernet statistics on the compute node.

```
ovs-appctl dpctl/show -s netdev@ovs-netdev
```

Result:
Few or no errors are seen on any port.

18. In case of multi-server deployments, check RabbitMQ cluster status on all vCICs.

    In case of single server deployments, check RabbitMQ cluster status on the vCIC.

    ```
    rabbitmqctl cluster_status
    ```

    Result:
    Expected result for cluster status is:

    `Cluster_name` is identical on all three vCICs

    ```
    Cluster status of node 'rabbit@cic-0-1' ...
    [{nodes,[{disc,['rabbit@cic-0-1','rabbit@cic-0-2',⇒
    'rabbit@cic-0-3']}]},
     {running_nodes,['rabbit@cic-0-3','rabbit@cic-0-2',⇒
    'rabbit@cic-0-1']},
     {cluster_name,<<"rabbit@cic-0-2.domain.tld">>},
     {partitions,[]}]
    ```

19. In case of multi-server deployments, check for zombie processes on all vCICs.

    In case of single server deployments, check for zombie processes on the vCIC.

    ```
    ps -efa | grep defunct
    ```

    Use the following script on vFuel to check for zombie processes:

    ```
    for node in $(fuel node | awk -F "|" '$7 ~ ⇒
    "compute" {print $3}')
    do
        echo ""
        echo "checking the zombie on host ${node}: "
        echo ""
        ssh -q -oStrictHostKeyChecking=no -oUserKnownHostsFile⇒
    =/dev/null root@${node} ps -efa | grep defunct
    done
    ```

Result:
No defunct process exists.

20. Check status from vFuel.

    ```
    ssh root@<vfuel_host_ip_address>
    fuel node.
    ```

    Result:
    List of vCIC and compute hosts is returned.

    Each node shows the status column as `ready`, and the online column as `1`.

21. Check Fuel services in vFuel:

    ```
    fuel-utils check_all | grep ready | cut -d' ' -f1
    fuel-utils check_all
    ```

    Result:
    All Fuel services are `ready`.

22. Check iSCSI multipath connections to VNX. Run the following script on vFuel:

    ```
    for node in $(fuel node | awk -F "|" '$7 ~ ⇒
    "compute" {print $3}')
    do
        echo ""
        echo "checking the iSCSI multipath on host ${node}: "
        echo ""
        ssh -q -oStrictHostKeyChecking=no ⇒
    -oUserKnownHostsFile=/dev/null root@${node}⇒
     multipath -ll | grep failed
    done
    ```

    Result:
    No multipaths are returned as failed. If no block device is attached to a node, an empty result is also acceptable.

2.2 Reboot vFuel

This section is only applicable for single server deployments.

Description

vFuel is rebooted. After the reboot, vFuel functions normally with no negative effects on any part of the system.

Table 3 Reboot vFuel

Objective	**After the reboot, vFuel functions normally with no other negative effects on any part of the system.**
Precondition	
Postcondition	
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to vFuel:

   ```
   ssh <vfuel address>
    fuel node list
   ```

   Result:
   Logged on to vFuel with connections to the vCIC and the host.

2. Reboot vFuel.

   ```
   reboot
   ```

   Result:
   vFuel is rebooted.

3. Wait for the Fuel VM to recover and log on to vFuel again:

   ```
   ssh <fuel address>
   fuel node list
   ```

   Expect recovery in approximately 180 seconds.

   Result:
   Logged on to vFuel.

4. Ensure vFuel is operational again. Use commands `fuel node list` and `fuel release list`.

   Result:
   vFuel is operational.

5. Use the following command to ensure Fuel services in vFuel are up and running after the reboot:
   `[root@fuel ~]# fuel-utils  check_all`

   Result:
   Fuel services are up and running. The following is an example of the output:

```
checking with command "systemctl is-active nailgun"
active
checking with command "! pgrep puppet"
nailgun is ready.
checking with command "egrep -q ^[2-4][0-9]? &lt; &lt;(curl --connect-timeout 1 -s -w '%{http_code}' http://192.168.0.11:8777/ostf/not_found -o /dev/null)"
checking with command "! pgrep puppet"
ostf is ready.
checking with command "ps waux | grep -q 'cobblerd -F' &amp;&amp; pgrep dnsmasq"
1636
checking with command "cobbler profile find --name=ubuntu* | grep -q ubuntu &amp;&amp; cobbler profile find --name=*bootstrap* | grep -q bootstrap"
checking with command "! pgrep puppet"
cobbler is ready.
checking with command "curl -f -L -i  -u "nailyiOjhRZUA6vLsej22nmOLG4jq6" http://127.0.0.1:15672/api/nodes  1>/dev/null 2>&amp;1"
checking with command "curl -f -L -u "mcollective:Hel00DLSikNyQfUe2X53oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_broadcast'"
checking with command "curl -f -L -u "mcollective:Hel00DLSikNyQfUe2X53oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_directed'"
checking with command "! pgrep puppet"
rabbitmq is ready.
checking with command "PGPASSWORD=qfbWQ5Y1FajFwF02NuFFYzf5 /usr/bin/psql -h 192.168.0.11 -U "nailgun" "nailgun" -c '\copyright' 2>&amp;1 1>/dev/null"
checking with command "! pgrep puppet"
postgres is ready.
checking with command "ps waux | grep -q 'astuted'"
checking with command "curl -f -L -u "naily:OjhRZUA6vLsej22nmOLG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'nailgun'"
checking with command "curl -f -L -u "naily:OjhRZUA6vLsej22nmOLG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'naily_service'"
checking with command "! pgrep puppet"
astute is ready.
checking with command "ps waux | grep -q mcollectived"
checking with command "! pgrep puppet"
mcollective is ready.
checking with command "ps waux | grep -q nginx"
checking with command "! pgrep puppet"
nginx is ready.
checking with command "keystone  --os-auth-url "http://192.168.0.11:35357/v2.0" --os-username "nailgun" --os-password "AgQ63CI3x7g1CdSxlUKpH6eQ" token-get &amp;>/dev/null"
checking with command "! pgrep puppet"
keystone is ready.
checking with command "netstat -nl | grep -q 514"
checking with command "! pgrep puppet"
rsyslog is ready.
checking with command "netstat -ntl | grep -q 873"
checking with command "! pgrep puppet"
rsync is ready.
```

6. Ensure traffic is undisturbed by the vFuel reboot.

   Result:
   Traffic is undisturbed.

2.3 Reboot vCIC in Single Server Deployment

This section is only applicable for single server deployments.

Description

vCIC is rebooted. After the reboot, vCIC functions normally with no negative effects on any part of the system.

Table 4 Reboot vCIC

Objective	**After the reboot, vCIC functions normally with no negative effects on any part of the system.**
Precondition	

Postcondition	
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to vFuel:
   ```
   ssh root@<fuel_address>
   fuel node list
   ```

   Result:
   Logged on to vFuel with connections to the vCIC and the host.

2. Reboot the vCIC:
   ```
   ssh cic-1
   reboot
   ```

   Result:
   vCIC is rebooted.

3. Monitor vCIC to ensure that it recovers in 300 seconds.

   Result:
   vCIC recovers.

4. Perform a `crm` process check.

   Use `crm_mon -1 -rf` or `crm status`.

   Result:
   Cluster status is displayed.

5. Check the connectivity for the tenant VMs.

   Result:
   Tenant VMs are reachable.

6. Ensure that the reboot of the vCIC has not interfered with the normal functioning of the node. Use commands `nova service-list`, `cinder service-list`, and `nova list --all-tenant` on the vCIC.

   Result:
   Normal services are available.

2.4 Reboot Region Switch in Single Server Deployment

This section is only applicable for single server deployments.

Description

The region switch, which has VRRP state `MASTER` for VLAN `cee_ctrl`, is rebooted. At the end of the reboot, a running VM can be pinged.

Table 5 Reboot Region Switch

Objective	**At the end of the reboot, a running VM can be pinged.**
Precondition	
Postcondition	• VMs are up and running on the compute host. • VM is reachable.
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

1. Upload image to Glance on vCIC:
   ```
   glance image-create --name Ubuntu_14.04.4_LTS -progress ⇒
   --disk-format qcow2 --container-format bare ⇒
   <trusty-server-cloudimg-amd64-disk1.img
   ```

   Result:
   Image is uploaded.

2. Create a router, network and subnetwork. Add interface to the router and boot two VMs in this network. Execute the following commands:

   ```
   neutron router-create <router-name>
   neutron net-create <net-name>
   neutron subnet-create --name <subnet-name><net-name> ⇒
   <ipaddress/prefix>
   neutron router-interface-add <router-name> <subnet-id>
   ```

   Result:
   Router, network and subnetwork is created.

3. To set the initial password for the Ubuntu user for the `trusty-server-cloudimg-amd64-disk1.img` image, create a `cloud_config.txt` file with the following content:

   ```
   #cloud-config.txt
   password: rootroot chpasswd: { expire: False }
   ssh_pwauth: True
   ```

   Result:
   Text file is created.

4. Boot a VM. Execute the following command:

   ```
   nova boot --flavor <flavor-id> --nic net-id=<net-id>⇒
   -- image <image-id> --user-data /tmp/cloud_c
   onfig.txt<vm-name>
   ```

   Check with **`nova list`** that the VM status is `ACTIVE` and power state is `Running`.

   Result:
   The VM becomes active.

5. Log on to the region switch (for example, `ssh network_admin@192.168.2.2`).

   Result:
   SSH connection to the region switch is established.

6. Check the VRRP state of VLAN `cee_ctrl` in the switch:

   ```
   show vrrp detail | include cee_ctrl
   ```

7. Create a virtual router on the switch and add the interface address.

   Save the configuration.

   Result:
   The interface has been added successfully to the router.

8. Log on to the VM and ping another VM.

   Result:

VM is pingable.

9. Log on to the region switch and reboot the switch with the `reboot` command.

   Result:
   Region switch starts the reboot process.

10. Log on to the VM and ping another VM.

    Result:
    Ping is not successful.

11. Connect again to the region switch.

    Result:
    SSH connection to the region switch is established.

12. Check VRRP state of the switch.

    ```
    show vrrp detail | include cee_ctrl
    ```

    Result:
    Switch state is `MASTER`.

13. Ping the VM again.

    Result:
    The VM is reachable again.

2.5 Reboot Single Server Host with VM Configured

This section is only applicable for single server deployments.

Description

Reboot a compute host and monitor that it recovers.

Table 6 Reboot Single Server Host with VM Configured

Objective	**Reboot Single Server Host with VM Configured**
Precondition	
Postcondition	
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	

Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Create a network and a subnetwork using the following commands:
   neutron net-create *<net-name>*
   neutron subnet-create --name *<subnet-name>* ⇒
   ***<net-name>* 10.0.3.0/24**

   Result:
   Network is created.

2. Upload image to Glance on the vCIC:
   glance image-create --name Ubuntu_14.04.4_LTS -progress ⇒
   --disk-format qcow2 --container-format bare ⇒
   <trusty-server-cloudimg-amd64-disk1.img

   Result:
   Image is uploaded to Glance.

3. To set the initial password for the Ubuntu user for the `trusty-server-cloudimg-amd64-disk1.img` image, create a `cloud_config.txt` file with the following content :
   ```
   #cloud-config.txt
   password: rootroot chpasswd: { expire: False }
   ssh_pwauth: True
   ```

   Result:
   Text file is created.

4. Boot VM1. Execute the following command:
   nova boot --*<flavor-id>* --nic net-id=*<net-id*
   ***>* --user-data ⇒**
   /tmp/cloud_config.txt --image *<image-id>* VM1

5. Reboot the compute host:

   a. Log on to the compute host (the method is optional):

      ssh compute-*<shelf_id>*-*<blade_id>*.

   b. Get node list from vFuel with `fuel node list`.

c. Execute **compute-*<shelf_id>*-*<blade_id>* reboot**.

Result:

- Compute host reboots.
- Access to compute host and vCIC are lost.
- The compute host and vCIC recovers fully.

6. Check that the OpenStack behavior and recovery are according to protocol.

   Result:
   OpenStack behavior and recovery are according to protocol.

7. If the `nova.conf` of the compute contains `resume_guests_state_on_host_boot=true`, VM1 will remain on the compute host and have status `ACTIVE` and power state `Running`.

   Result:
   VM1 has status `ACTIVE` and power state `Running`.

8. From vCIC, verify that `nova-compute` is up again on the rebooted compute host.
   nova service-list
   service --status-all

   Result:
   Nova services are available.

9. Check that Neutron agents are alive:

   ```
   neutron agent-list
   ```

   Result:
   All Neutron agents are alive.

10. Check that a new VM can be booted on the rebooted host after recovery:

    a. **nova boot --flavor *<flavor-id>* --image *<image-ID>* --nic net-id=*<net-ID>* VM2**

       Result:
       VM2 is created in `ACTIVE` status with power state `Running`.

    b. **ip netns exec qdhcp-*<net-ID>* ping *<vm-IP>***

       Result:
       VM2 can be pinged.

11. Check that Performance Management (PM) is still functional after the reboot.

    a. Find the vCIC marked active.

```
crm status | grep active_marker
```

Result:

```
root@cic-0-1:/var/cache/pmreports# crm status ⇒
| grep active
 active_mark    (lsb:active_marker):    Started cic-0-3
```

b. From the active vCIC, confirm that the PM reports are still being produced after the reboot.

Note: Reports are produced every 15 minutes.

```
ls -alFt /var/cache/pmreports/A*.xml
```

Result:
The following is an example of the output:

```
root@cic-1:~# ls -alFt /var/cache/pmreports/A*.xml
-rw-r--r--  1 root root 133874 Jun 17 14:46 A20150617⇒
.1430+0000-0617.1445_DC148.xml
-rw-r--r--  1 root root 133708 Jun 17 15:01 A20150617⇒
.1445+0000-0617.1500_DC148.xml
```

c. Check the latest PM report for data from the compute host:

```
awk '/compute-0-2/,/<\/measValue>/' /var/cache/p
mreports/pm_id.xml
```

Result:
Values for the rebooted host are printed, for example:

```
<measValue measObjLdn="compute-1.domain.tld">
<r p="107">0r p="107">0>
<r p="747">0r p="747">0>
```

12. Delete the VMs created for the test case.

a. `nova delete VM1`

b. `nova delete VM2`

Result:
VMs are deleted.

2.6 Backup and Restore Atlas in Single Server Deployment

This section is only applicable for single server deployments where Atlas is installed and running.

Description

Back up Atlas according to *Atlas Backup*. Delete the Atlas VM. Launch a new Atlas VM. After the Atlas VM becomes `ACTIVE` download the backup files from Swift to the Atlas VM and initiate a restore according to *Atlas Restore*.

Table 7 Backup and Restore of Atlas

Objective	**Atlas VM is restored**
Precondition	• Atlas GUI is fully functional.
Postcondition	• Atlas GUI is fully functional.
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. From Atlas, create a text file in the `/root` directory.
   ```
   ssh atlasadm@<atlas_IP_address>

   sudo -i
   echo 'Test of restore of Atlas environment' >> test_file.txt
   ```

   Result:
   Logged on to Atlas and a text file `test_file.txt` is created.

2. Back up Atlas as described in *Atlas Backup*. Verify that the backup is stored in Swift.

   Result:
   The following is an example of the output:

   ```
   atlasadm@atlas:~$ sudo atlas backup-create --name
   atlas_backup --p password
   Running Atlas backup ...
   Done.
   atlasadm@atlas:~$ sudo atlas backup-list
      ID   Name   Date
   1465911268   atlas_backup   Tue Jun 14 15:34:28 CET 2016
   ```

Uploading Backup to Swift

```
root@atlas:~# cd /var/archives/
root@atlas:/var/archives# export OS_TENANT_NAME='admin'
root@atlas:/var/archives# export OS_USERNAME='admin'
root@atlas:/var/archives# export OS_PASSWORD='admin'

root@atlas:/var/archives# swift upload AtlasBackups ⇒
*1480183339*
atlas_backup1480183339/atlas_backup.1480183339.sha256.enc
atlas_backup1480183339/atlas_backup.1480183339-all-mysql-⇒
databases.sql.bz2.enc
atlas_backup1480183339/atlas_backup.1480183339-etc-puppet-⇒
hieradata-passwords.yaml.master.tar.gz.enc
atlas_backup1480183339/atlas_backup.1480183339-⇒
root.master.tar.gz.enc
atlas_backup1480183339/atlas_backup.1480183339-⇒
home-atlasadm.master.tar.gz.enc
```

3. Check in one of the vCICs that the Atlas backup is uploaded to Swift, using the following command:

   `swift list --lh`

   Result:
   The following is an example of the output:

```
root@cic-1:~#  swift list --lh
    6 1.1M 2015-09-30 06:17:42 AtlasBackups
    0    0 2015-09-29 16:06:39 fuelbackup
    0    0 2015-09-29 16:06:39 fuelbackup_segments
    6 1.1M

root@cic-1:~# swift list --long AtlasBackups
      724815 2016-11-25 14:14:19 application/octet-stream ⇒
AtlasBackup1480083137/AtlasBackup.1480083137-all-mysql-⇒
databases.sql.bz2.enc
         500 2016-11-25 14:14:19 application/octet-stream ⇒
AtlasBackup1480083137/AtlasBackup.1480083137-etc-puppet-⇒
hieradata-passwords.yaml.master.tar.gz.enc
   289296735 2016-11-25 14:14:19 application/octet-stream ⇒
AtlasBackup1480083137/AtlasBackup.1480083137-home-atlasadm.⇒
master.tar.gz.enc
   201311200 2016-11-25 14:14:18 application/octet-stream ⇒
AtlasBackup1480083137/AtlasBackup.1480083137-root.master.⇒
tar.gz.enc
         890 2016-11-25 14:14:18 application/octet-stream ⇒
AtlasBackup1480083137/AtlasBackup.1480083137.sha256.enc
   491334140
```

4. Delete the Atlas VM using the following command:

nova delete ***<atlas_VM_ID>***

Result:
The Atlas VM is deleted.

5. Launch a new Atlas VM, as described in *Atlas SW Installation*.

   Result:
   The VM is `ACTIVE` and reachable.

6. Login to the Atlas VM with SSH.

   Result:
   Logged on to the Atlas VM.

7. Download the backup files from Swift, as described in *Atlas Restore*.

   Result:
   The following is an example of the output:

   Atlas Restore

```
root@atlas:~# atlas backup-list
```

   Downloading Backup File from Swift

```
root@atlas:~# cd /var/archives/
root@atlas:/var/archives# export OS_TENANT_NAME='admin'
root@atlas:/var/archives# export OS_USERNAME='admin'
root@atlas:/var/archives# export OS_PASSWORD='admin'

root@atlas:/var/archives/# swift list AtlasBackups | ⇒
grep 1465911268
atlas_backup1465911268/atlas_backup.1465911268-all-⇒
mysql-databases.sql.bz2.enc
atlas_backup1465911268/atlas_backup.1465911268-etc-⇒
puppet-hieradata-passwords.yaml.master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268-home-⇒
atlasadm.master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268-root.⇒
master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268.sha256.enc

root@atlas:/var/archives# swift download AtlasBackups
-p atlas_backup1465911268

root@atlas:/var/archives# sudo atlas backup-list
  ID              Name                  Date
1465911268     atlas_backup    Wed Sep 30 09:03:00 UTC 2015

root@atlas:/var/archives# atlas backup-⇒
restore --d 1465911268 --p
```

```
<backup_password>
Atlas has been set for restore.
Please reboot.

root@atlas:/var/archives# reboot

Broadcast message from root@atlas
        (/dev/pts/4) at 9:45 ...

The system is going down for reboot NOW!
```

8. When the reboot is finished, log on to the Atlas VM with SSH. Verify that the created text file has been restored.

   ```
   ssh root@<atlas_IP_address>
   ls -l
   cat test_file.txt
   ```

   Result:
   The `test_file.txt` is listed and contains: “Test of restore of Atlas environment”.

2.7 Reboot Compute Host and Recover vFuel

This section is only applicable for multi-server deployments.

Description

Reboot the compute host that servers vFuel and monitor that it recovers. vFuel recovers after rebooting the compute host.

Table 8 Reboot Compute Host and Recover vFuel

Objective	**Reboot the compute host serving vFuel. vFuel recovers after rebooting the compute host.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	

Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Reboot the compute host from vFuel. Get the node list with the command `fuel node list`.

   ```
   ssh compute-<shelf_id>-<blade_id>
   reboot
   ```

   Result:
   The compute host reboots. Full recovery is expected within 120 seconds.

2. Check in Atlas **Alarm & Alert History** that the `Compute Host Restarted` (major ID: `193`, minor ID: `2031703`) alert is issued for the rebooted compute host.

   If Atlas is not installed, execute the following command on vCIC:
   `watchmen-client alarm-history`

   Result:
   Alert `Compute Host Restarted` is issued and visible in the history.

3. Wait for the host to recover and log on to vFuel:

   ```
   ssh root@<fuel address>
   ```

   Expect to be able to login to vFuel in approximately two minutes.

   Note: Fuel services should take 10–15 minutes to recover fully.

   Result:
   Successfully logged on to the Fuel host.

4. Check that Fuel services in vFuel:

   ```
   fuel-utils check_all | grep ready | cut -d' ' -f1
   fuel-utils check_all
   fuel node
   ```

   Result:
   All Fuel services are `ready`. vFuel reads all vCICs and compute hosts.

   The following is an example of the output:

```
[root@fuel ~]# fuel-utils check_all | grep ready | cut -d' ' -f1
nailgun
ostf
cobbler
rabbitmq
postgres
astute
mcollective
nginx
keystone
rsyslog
rsync
[root@fuel ~]#

[root@fuel ~]# fuel-utils check_all
checking with command "systemctl is-active nailgun"
active
checking with command "! pgrep puppet"
nailgun is ready.
checking with command "egrep -q ^[2-4][0-9]? < <(curl --connect-timeout 1 -s -w '%{http_code}' http://192.168.0.11:8777/ostf/not_found -o /dev/null)"
checking with command "! pgrep puppet"
ostf is ready.
checking with command "ps waux | grep -q 'cobblerd -F' && pgrep dnsmasq"
1636
checking with command "cobbler profile find --name=ubuntu* | grep -q ubuntu && cobbler profile find --name=*bootstrap* | grep -q bootstrap"
checking with command "! pgrep puppet"
cobbler is ready.
checking with command "curl -f -L -i  -u "naily:OjhRZUA6vLsej22nmOLG4jq6" http://127.0.0.1:15672/api/nodes  1>/dev/null 2>&1"
checking with command "curl -f -L -u "mcollective:Hel00DLSikNyQfUe2XS3oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_broadcast'"
checking with command "curl -f -L -u "mcollective:Hel00DLSikNyQfUe2XS3oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_directed'"
checking with command "! pgrep puppet"
rabbitmq is ready.
checking with command "PGPASSWORD=qfbWQ5Y1FajPwF02NuFPYzf5 /usr/bin/psql -h 192.168.0.11 -U "nailgun" "nailgun" -c '\copyright' 2>&1 1>/dev/null"
checking with command "! pgrep puppet"
postgres is ready.
checking with command "ps waux | grep -q 'astuted'"
checking with command "curl -f -L -u "naily:OjhRZUA6vLsej22nmOLG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'nailgun'"
checking with command "curl -f -L -u "naily:OjhRZUA6vLsej22nmOLG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'naily_service'"
checking with command "! pgrep puppet"
astute is ready.
checking with command "ps waux | grep -q mcollectived"
checking with command "! pgrep puppet"
mcollective is ready.
checking with command "ps waux | grep -q nginx"
checking with command "! pgrep puppet"
nginx is ready.
checking with command "keystone  --os-auth-url "http://192.168.0.11:35357/v2.0" --os-username "nailgun" --os-password "AgQ63CI3x7glCdSx1UKpH6eQ" token-get &>/dev/null"
checking with command "! pgrep puppet"
keystone is ready.
checking with command "netstat -nl | grep -q 514"
checking with command "! pgrep puppet"
rsyslog is ready.
checking with command "netstat -ntl | grep -q 873"
checking with command "! pgrep puppet"
rsync is ready.
[root@fuel ~]#

[root@fuel ~]# fuel node
id | status | name        | cluster | ip           | mac               | roles             | pending_roles | online | group_id
---+--------+-------------+---------+--------------+-------------------+-------------------+---------------+--------+---------
 9 | ready  | cic-1       |       1 | 192.168.0.29 | f6:18:a9:14:51:44 | controller, mongo |               |      1 |        1
 4 | ready  | compute-0-6 |       1 | 192.168.0.25 | b8:ca:3a:70:e2:2c | compute           |               |      1 |        1
 3 | ready  | compute-0-8 |       1 | 192.168.0.27 | b8:ca:3a:71:5a:28 | compute           |               |      1 |        1
 5 | ready  | compute-0-4 |       1 | 192.168.0.23 | ec:f4:bb:c1:2b:b8 | compute           |               |      1 |        1
 8 | ready  | compute-0-7 |       1 | 192.168.0.26 | b8:ca:3a:70:ee:a4 | compute           |               |      1 |        1
10 | ready  | cic-3       |       1 | 192.168.0.28 | 3e:73:de:c1:b3:49 | controller, mongo |               |      1 |        1
 2 | ready  | compute-0-5 |       1 | 192.168.0.24 | b8:ca:3a:71:54:20 | compute           |               |      1 |        1
11 | ready  | cic-2       |       1 | 192.168.0.30 | 0a:84:56:52:a1:4b | controller, mongo |               |      1 |        1
 1 | ready  | compute-0-2 |       1 | 192.168.0.20 | ec:f4:bb:c1:2d:20 | compute, virt     |               |      1 |        1
 7 | ready  | compute-0-3 |       1 | 192.168.0.22 | ec:f4:bb:c1:30:30 | compute, virt     |               |      1 |        1
 6 | ready  | compute-0-1 |       1 | 192.168.0.21 | ec:f4:bb:c1:2f:c0 | compute, virt     |               |      1 |        1
```

5. From a vCIC, check that `nova-compute` is up again on the rebooted compute host.
   `nova service-list`

   Result:
   Nova services are available.

6. From a vCIC, check that Neutron agents are alive:

   `neutron agent-list`

Result:
All Neutron agents are alive.

7. Log on to the rebooted compute host. In case VNX is used, check that the multipath connections are intact after the reboot:

   ```
   root@compute-0-6:~# multipath -ll
   ```

   Result:
   Connections are all `active`, `ready`, and `running`.

8. Check that Performance Management (PM) is still functional after the reboot.

   Find the vCIC marked `active`:

   ```
   crm status | grep active_marker
   ```

   From this vCIC confirm that the PM reports are still being produced after the reboot.

   Note: Reports are produced every 15 minutes.

   ```
   ls -alFt /var/cache/pmreports/A*.xml
   ```

   Check the latest PM report for data from the compute host that was restarted:

   ```
   awk '/compute_host/,/<\/measValue>/' /var/ca
   che/pmreports/pm_id.xml
   ```

   Result:
   Values for the rebooted host are printed. The following is an example of the output:

   ```
   root@cic-0-4:/var/cache/pmreports# crm status | grep active
    active_mark    (lsb:active_marker):    Started cic-0-3

   root@cic-0-3:~# ls -alFt /var/cache/pmreports/A*.xml

   .
   -rw-r--r--  1 root root 133874 Jun 17 14:46 A20150617.⇒
   1430+0000-0617.1445_DC148.xml
   -rw-r--r--  1 root root 133708 Jun 17 15:01 A20150617.⇒
   1445+0000-0617.1500_DC148.xml

   root@cic-0-3:~# awk '/compute-0-2/,/<\/measValue>/' ⇒
   /var/cache/pmreports/A20150423.0845+0000-0423.0900.xml
       <measValue measObjLdn="compute-0-2.domain.tld">
                   <r p="107">0</r>
                   <r p="747">0</r>
   ```

2.8 Reboot Compute Host and Recover vCIC

This section is only applicable for multi-server deployments.

Description

Reboot the compute host that serves a vCIC, and monitor that it recovers. vCIC recovers after rebooting the compute host.

Table 9 Reboot Compute Host and Recover vCIC

Objective	**Reboot the compute host that serves a vCIC. vCIC recovers after rebooting the compute host.**
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to the compute host on vFuel:
   `ssh compute-<shelf_id>-<blade_id>`

   Result:
   Logged on to the compute host.

2. Reboot the compute host.
   `sudo reboot -f`

   Result:
   Compute host reboots. Full recovery is expected within 120 seconds.

3. Check in the Atlas **Alarm & Alert History** that a `Compute Host Restarted` alert (major ID: `193`, minor ID: `2031703`) is issued for the rebooted compute host.

   If Atlas is not installed, execute the following command on vCIC:

watchmen-client alarm-history

Result:
A `Compute Host Restarted` alert is issued and visible in the history.

4. On vCIC, check that `nova-compute` is up again on the rebooted compute host.
   nova service-list

   Result:
   Nova services are available. The following is an example of the output:

```
root@cic-1:~# nova service-list
+----+------------------+-----------------------+----------+---------+-------+----------------------------+-----------------+
| Id | Binary           | Host                  | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+----+------------------+-----------------------+----------+---------+-------+----------------------------+-----------------+
| 1  | nova-consoleauth | cic-2.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:34.000000 | -               |
| 2  | nova-scheduler   | cic-2.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:37.000000 | -               |
| 3  | nova-conductor   | cic-2.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:41.000000 | -               |
| 4  | nova-cert        | cic-2.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:37.000000 | -               |
| 7  | nova-consoleauth | cic-1.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:37.000000 | -               |
| 10 | nova-scheduler   | cic-1.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:40.000000 | -               |
| 13 | nova-conductor   | cic-1.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:42.000000 | -               |
| 19 | nova-cert        | cic-1.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:41.000000 | -               |
| 22 | nova-consoleauth | cic-3.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:33.000000 | -               |
| 25 | nova-scheduler   | cic-3.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:33.000000 | -               |
| 28 | nova-conductor   | cic-3.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:41.000000 | -               |
| 37 | nova-cert        | cic-3.domain.tld      | internal | enabled | up    | 2016-07-21T13:40:38.000000 | -               |
```

5. Check that Neutron agents are alive on vCIC.
   neutron agent-list

   Result:
   All Neutron agents are alive. The following is an example of the output:

```
root@cic-1:~# neutron agent-list
+--------------------------------------+--------------------+-----------------------+-------+----------------+---------------------------+
| id                                   | agent_type         | host                  | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+-----------------------+-------+----------------+---------------------------+
| 06888113-fadb-4210-9a1f-76e2702b74d1 | DHCP agent         | cic-1.domain.tld      | :-)   | True           | neutron-dhcp-agent        |
| 162f905e-2c94-4691-adb5-db92264895d0 | Open vSwitch agent | compute-0-4.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| 4612a4f5-9cbb-42cf-b229-15f0b853ae14 | DHCP agent         | cic-3.domain.tld      | :-)   | True           | neutron-dhcp-agent        |
| 5e4693db-794c-4776-b49d-1c14060b1737 | Open vSwitch agent | cic-3.domain.tld      | :-)   | True           | neutron-openvswitch-agent |
| 6a7d88b7-fc15-4d8c-ac94-6cce345e217a | Open vSwitch agent | compute-0-3.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| 85feea15-7de3-467a-b7fc-bc87bfd23285 | Open vSwitch agent | cic-1.domain.tld      | :-)   | True           | neutron-openvswitch-agent |
| aed9143a-ea6a-4aa6-b3a0-33dde1f9beb0 | Open vSwitch agent | compute-0-6.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| c6ddf9de-2bd5-43c8-b689-a223d6b88053 | Open vSwitch agent | compute-0-2.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| d6e8e197-f3db-4a66-8ebf-77c13ce4671a | Open vSwitch agent | compute-0-5.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| daec6800-df34-4a39-9ac7-9ae7f52bc97b | Open vSwitch agent | compute-0-1.domain.tld | :-)   | True           | neutron-openvswitch-agent |
| dcd451c6-7f44-466b-be21-d1199072c9e4 | DHCP agent         | cic-2.domain.tld      | :-)   | True           | neutron-dhcp-agent        |
| f07083a3-c31c-4e85-970f-7c30203a4af7 | Open vSwitch agent | cic-2.domain.tld      | :-)   | True           | neutron-openvswitch-agent |
+--------------------------------------+--------------------+-----------------------+-------+----------------+---------------------------+
```

6. If VNX is used, check that the multipath connections are intact after the reboot.
   root@compute-*<shelf_id>*-*<blade_id>*:~# multipath -ll

   Result:
   Connections are all `active`, `ready`, and `running`.

7. Check that Performance Management (PM) is still functional after the reboot.

   a. Find the vCIC marked active:
      crm status | grep active_marker

   b. vCIC confirms that the PM reports are still being produced after the reboot:
      ls -alFt /var/cache/pmreports/A*.xml

Note: Reports are produced every 15 minutes.

c. Check the latest PM report for data from the compute host that was restarted.
awk '/compute-*<shelf_id>*-*<blade_id>*/,⇒
/<\/measValue>/' /var/cache/pmreports/
A*<pm_report>*.xml

Result:
An example of the output is the following:

```
root@cic-0-4:/var/cache/pmreports# crm status | grep active
 active_mark    (lsb:active_marker):    Started cic-0-3
root@cic-0-3:~# ls -alFt /var/cache/pmreports/A*.xml
-rw-r--r--  1 root root 133874 Jun 17 14:46 A20150617.1430+0000⇒
-0617.1445_DC148.xml
-rw-r--r--  1 root root 133708 Jun 17 15:01 A20150617.1445+0000⇒
-0617.1500_DC148.xml
root@cic-0-3:~# awk '/compute-0-2/,/<\/measValue>/' ⇒
/var/cache/pmreports/A20150423.0845+0000-0423.0900.xml
```

The following are example values for the rebooted server:

```
<measValue measObjLdn="compute-0-2.domain.tld">
            <r p="107">0</r>
            <r p="747">0</r>
```

2.9 Reboot vCIC Host with Master RabbitMQ Node

This section is only applicable for multi-server deployments.

Note: Test case is not applicable for single server since there is only one vCIC.

Description

The Galera system provides redundancy and synchronization to the MySQL database used on the triple CIC system implemented with OpenStack. At least two vCICs must be working in this system for the database to continue updating correctly. If two vCICs are down, the database in the remaining vCIC cannot sync with the other vCICs and so it ceases to update the database until it regains contact with at least one other vCIC.

In case only two vCICs remain, redundancy is lost as well, because two vCICs are required to sync the database. Redundancy exists only if there are three vCICs in the system.

In this test case the host runnung the master Galera is rebooted, to see how the system recovers. It is expected that a standby master Galera node takes over

as master. Normal functionality remains undisturbed and database updates continue without fault. Traffic in the rest of the node is unaffected. All services remain available.

Table 10 Reboot vCIC Host with Master RabbitMQ Node

Objective	**A standby master Galera node takes over as master. Normal functionality remains undisturbed and database updates continue without fault. Traffic in the rest of the node is unaffected. All services remain available.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. If not already uploaded, upload image to Glance on vCIC:
   `glance image-create --name Ubuntu_14.04.4_LTS --progress ⇒ --disk-format qcow2 --container-format bare ⇒ <trusty-server-cloudimg-amd64-disk1.img`
   Alternatively, an existing image can be used.

   Result:
   Image is uploaded to Glance.

2. To set the initial password for the Ubuntu user for the `trusty-server-cloudimg-amd64-disk1.img` image, create a `cloud_config.txt` file with the following content :
   ```
   #cloud-config.txt
   password: rootroot chpasswd: { expire: False }
   ssh_pwauth: True
   ```

Result:
Text file is created.

3. Start a VM:
   nova boot --flavor *<flavor>* --image *<image_ID>* --nic net-id=*<net_ID>* ⇒
   --user-data /tmp/cloud_config.txt *<VM_name>*

   Do the following on BSP:

   a. Use (or create if does not exists) a flavor with disk configured to `0` Gb.

   b. Create a volume from the flavor:
      cinder create --image-id *<image_id>* --display-name *<volume_name>* *<size>*

   c. Boot the VM:
      nova boot --*<flavor-id>* --nic net-id=*<net_id>* ⇒
      --user-data /tmp/cloud_config.txt ⇒
      --block-device-mapping vda=*<volume_id>*⇒
      :::delete-on-terminate *<vm_name>*

   Result:
   The volume is created. The VM starts.

4. Check that the VM is started:
   nova list

   Result:
   The VM status is `ACTIVE` and power state is `Running`.

5. Ping the VM to verify that it is reachable:

   ip netns exec *<NAMESPACE_ID>* ping *<IP_ADDRESS_VM>*

   To identify the name space, use the following commands:
   neutron net-list
   ip netns

   To identify the IP address of the VM, use the following command:
   nova list

   Result:
   The VM can be pinged.

6. Find the vCIC where the RabbitMQ master node is running. Find the master with the following command from any vCIC:
   crm resource status master_p_rabbitmq-server

   Result:
   The vCIC host where the RabbitMQ master is running is found.

The following is an example of the output:

```
root@cic-0-1:~# crm resource status master_
p_rabbitmq-server
resource master_p_rabbitmq-server is running on: ⇒
cic-0-3.domain.tld Master
resource master_p_rabbitmq-server is running on: ⇒
cic-0-4.domain.tld
resource master_p_rabbitmq-server is running on: ⇒
cic-0-1.domain.tld
```

7. Perform a reboot on the vCIC where the master RabbitMQ is running:
   reboot

   Result:
   The vCIC is rebooting, connection is lost.

8. Check Atlas for the alarm *CIC Failed* (major ID: 193, minor ID: 2031672).

   If Atlas is not installed, execute the following command on vCIC:
   watchmen-client alarm-history

   The following alarms are issued for ports connected to the vCIC host:

   - *Ethernet Port Fault* (major ID: 193, minor ID: 2031681)
   - *Ethernet Port Aggregator Fault*, (major ID:193, minor ID: 2031682)

   Check that the*Service Stopped* (major ID: 193, minor ID: 2031710) alert is received, stating that OpenStack services have been stopped on the vCIC.

   Result:
   Alarms and alerts are received.

9. Wait until vCIC running on rebooted host is OFFLINE in Pacemaker. Run the following command from a vCIC that was not rebooted:
   crm status

   Result:
   The command shows that the rebooted vCIC is OFFLINE.

10. Check that another vCIC takes over as master RabbitMQ node:
    crm status resource master_p_rabbitmq-server

    Result:
    vCIC that is the new master RabbitMQ node is located.

11. Check if networks, subnets, and VMs can be successfully created while one vCIC is down.

    Result:
    Networks, subnets, and VMs can be created.

12. Wait until the vCIC running on the rebooted host is ONLINE again in Pacemaker and Pacemaker controlled resources are running. Run the following command from a vCIC that was not rebooted:
    crm status

    The host is expected to recover full functionality within 300 seconds.

    Check the time between when the reboot was issued and the time it takes for RabbitMQ to start and synchronize all queues again.

    Result:
    The rebooted vCIC is back ONLINE.

    Pacemaker controlled services should be in the correct state divided over three vCICs. Check the printout from one vCIC. The order below is not a specific output example, but a structure outline for the distribution of services over the vCICs:

```
root@cic-0-4:~# crm_mon -1 -rf
Last updated: Wed Dec 16 13:08:23 2015
Last change: Fri Dec  4 17:29:56 2015
Stack: corosync
Current DC: cic-0-1.domain.tld (1) - partition with quorum
Version: 1.1.12-561c4cf
3 Nodes configured
50 Resources configured

Online: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]

Full list of resources:

Clone Set: clone_p_vrouter [p_vrouter]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
vip__management (ocf::fuel:ns_IPaddr2): Started cic-0-1⇒
.domain.tld
vip__vrouter_pub (ocf::fuel:ns_IPaddr2): Started cic-0-1⇒
.domain.tld
vip__vrouter (ocf::fuel:ns_IPaddr2): Started cic-0-1⇒
.domain.tld
vip__public (ocf::fuel:ns_IPaddr2): Started cic-0-1⇒
.domain.tld
Clone Set: clone_p_haproxy [p_haproxy]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_dns [p_dns]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_mysql [p_mysql]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
```

```
Master/Slave Set: master_p_rabbitmq-server [p_rabbitmq-server]
Masters: [ cic-0-4.domain.tld ]
Slaves: [ cic-0-1.domain.tld cic-0-3.domain.tld ]
Master/Slave Set: master_p_conntrackd [p_conntrackd]
Masters: [ cic-0-1.domain.tld ]
Slaves: [ cic-0-3.domain.tld cic-0-4.domain.tld ]
p_ceilometer-agent-central (ocf::fuel:ceilometer-agent⇒
-central): Started cic-0-3.domain.tld
p_ceilometer-alarm-evaluator (ocf::fuel:ceilometer-alarm⇒
-evaluator): Started cic-0-4.domain.tld
Clone Set: clone_p_ntp [p_ntp]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_neutron-metadata-agent [p_neutron-metadata⇒
-agent]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_neutron-dhcp-agent [p_neutron-dhcp-agent]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_neutron-plugin-openvswitch-agent ⇒
[p_neutron-plugin-openvswitch-agent]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
p_watchmen-service (lsb:watchmen-service): Started cic-0-3⇒
.domain.tld
Clone Set: clone_p_watchmen-api [p_watchmen-api]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_watchmen-snmpagent [p_watchmen-snmpagent]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
Clone Set: clone_p_watchmen-zabbixendpoint [p_watchmen⇒
-zabbixendpoint]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]
p_active_mark (lsb:active_marker): Started cic-0-4.domain.tld
Clone Set: clone_p_marker_checkd [p_marker_checkd]
Started: [ cic-0-1.domain.tld cic-0-3.domain.tld cic-0-4⇒
.domain.tld ]


Migration summary:

* Node cic-0-1.domain.tld:
* Node cic-0-4.domain.tld:
* Node cic-0-3.domain.tld:
root@cic-0-4:~#
```

13. Check whether the *CIC Failed* and *Ethernet Port Fault* alarms are ceased.

Result:
The alarms are ceased.

14. Check in Atlas **Alarm & Alert History** that the *CIC Restarted* alert (major ID: `193`, minor ID: `2031701`) is issued.

    Result:
    The alert is displayed in Atlas.

15. Check that the VM which was booted before the vCIC reboot is still available in `ACTIVE` state and `Running` power state:
    `nova list`

    Result:
    VM is in `ACTIVE` status and `Running` power state. The VM is unaffected by the disturbance.

16. Verify that the VM created before the reboot can be reached and pinged. Ping a VM:

    `ip netns exec <NAMESPACE_ID> ping <IP_ADDRESS_VM>`

    Identify the name space with the following commands:
    `neutron net-list`
    `ip netns`

    The IP address of the VM can be identified with the following command:
    `nova list`

    Result:
    VM can be pinged.

17. Start another VM.

    `nova boot --<flavor-id> --nic net-id=<net_id> ⇒`
    `--user-data /tmp/cloud_config.txt ⇒`
    `--block-device-mapping vda=<volume_id>⇒`
    `:::delete-on-terminate <vm_name>`

    Result:
    New VM boots up correctly in status `ACTIVE` and power state `Running`.

18. Verify that the new VM is reachable and can be pinged:

    `ip netns exec <NAMESPACE_ID> ping <IP_ADDRESS_VM>`

    The name space can be identified with the following commands:
    `neutron net-lisť`
    `ip netns`

    The IP address of the VM can be identified with the following command:
    `nova list`

When logged on to this VM, ping the first VM address in Step 5.

Result:
The new VM can be pinged and the VMs can ping each other.

19. Delete extra VMs and volumes used for the test case:
```
nova delete <vm_id>
cinder delete <volume_id>
```

Result:
The VMs are deleted.

2.10 Reboot Region Switch in multi-Server Deployment

This section is only applicable for multi-server deployments.

Description

The region switch, which has `MASTER` VRRP state for VLAN `cee_ctrl`, is rebooted. The switch loses the `MASTER` state to the redundant switch and gets back the `MASTER` state at the end of the reboot. A running VM can be pinged during the whole process.

Table 11 Reboot Region Switch

Objective	**The switch loses its MASTER state to the redundant switch and gets it back at the end of the reboot.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Create a network and subnetwork, create and boot a volume, and ping the booted VM.

   a. Create a network and subnetwork.
   neutron net-create *<net-name>*
   neutron subnet-create --name *<subnet-name> <net-name>* ⇒
   10.0.3.0/24

   b. Upload image to Glance on vCIC:
   glance image-create --name Ubuntu_14.04.4_LTS ⇒
   --progress --disk-format qcow2 --container-format bare⇒
   <trusty-server-cloudimg-amd64-disk1.img

   c. Create a volume in the network:
   cinder create --image-id *<image-id>* **--display-name** *<volume-name> <size>*

   The parameter `image-id` is the ID of the Glance image created in Step b.

   The parameter `size` is in GBs, for example `40`.

   The command `cinder list` shows the ID of the created volume.

   d. To set the initial password for the Ubuntu user for the `trusty-server-cloudimg-amd64-disk1.img` image, create a `cloud_config.txt` file with the following content :
   ```
   #cloud-config.txt
   password: rootroot chpasswd: { expire: False }
   ssh_pwauth: True
   ```

   e. Boot a VM from a volume in this network.
   nova boot --*<flavor-id>* **--nic net-id=***<net-id>* **-use**
   r-data /tmp/cloud_config.txt --block-device-m
   apping vda=*<volume-name>***:::delete-on-terminate**
   --availability-zone nova:compute-0-x *<vm-name>*

   f. Ping the VM when it boots up in status `ACTIVE` and power state `Running`.

   `nova list –all-tenant` shows the VM IP address.

   sudo ip netns exec qdhcp- *<net-id>* **ping** *<vm-ip>*

   Result:
   The VM becomes active. Pinging the VM is successful.

2. Check that switches have the correct status before the reboot.

   The command `neutron device-list` shows connected switches.

a. For Extreme switches, get status using the following command:

Note: This step does not apply to BSP deployments.

```
neutron device-show <device-name>
```

b. For CMX, access DMX using the following commands:

```
ssh advanced@<switch_ip_address> -p 2024
> show ManagedElement=1,DmxcFunction=1,Eqm=1⇒
,VirtualEquipment=BSP,Blade=0-26
> show ManagedElement=1,DmxcFunction=1,Eqm=1⇒
,VirtualEquipment=BSP,Blade=0-28
```

Result:

- Extreme: All switches are `ACTIVE`.
- CMX: `operationalState` is `ENABLED`.

3. Check the VRRP state on switch A.

a. If using Extreme switches, check VRRP state of VLAN `cee_ctrl` in switch A:

```
ssh network_admin@<switch_a_ip_address>
'show vrrp detail | include cee_ctrl'
```

b. If using CMX, issue the following commands:

Note: The following is an example. For actual values and addresses, refer to the local *IP and VLAN Plan*

```
>show ManagedElement=1,Transport=1,Router=0-26-om⇒
_sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session=⇒
VRRP_internal_cee_om_sp,vrrpState
vrrpState=MASTER
>show ManagedElement=1,Transport=1,Router=0-28-om⇒
_sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session=⇒
VRRP_internal_cee_om_sp,vrrpState
vrrpState=BACKUP
```

Result:

- Switch A is in `MASTER` state.
- CMX-L is in `Master` state.

4. Reboot the region switch.

a. If using Extreme, reboot SWA with the following commands:

```
ssh network_admin@<switch_a_ip_address>
 reboot
```

b. If using CMX, reboot the CMX with the following commands:

```
ssh advanced@<scx_1_ip_address>
ssh advanced@<cmx_ip_address>
advanced@<cmx_ip_address>:~$ su -
root@<cmx_ip_address>:~# reboot
```

Result:
Region switch A starts the reboot process.

5. Check the VRRP state on switch B.

   a. If using Extreme, check VRRP state of VLAN `cee_ctrl` in switch B using the following commands:

   ```
   ssh network_admin@ <switch_b_ip_address>
   show vrrp detail | include cee_ctrl
   ```

   b. If using CMX, issue the following commands:

   Note: The following is an example. For actual values and addresses, refer to the local *IP and VLAN Plan*.

   ```
   >show ManagedElement=1,Transport=1,Router=0-28-om⇒
   _sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session=⇒
   VRRP_internal_cee_om_sp,vrrpState
   ```

   Result:
   Switch B changes state from `BACKUP` to `MASTER`.

6. Check that pinging the newly created VM still works.

   Result:
   The VM is still reachable.

7. Check the VRRP state of switch B.

   a. If using Extreme, check VRRP state of VLAN `cee_ctrl` in switch B using the following commands:

   ```
   show vrrp detail | include cee_ctrl
   ```

   b. If using CMX, issue the following commands:

   Note: The following is an example. For actual values and addresses, refer to the local *IP and VLAN Plan*.

   ```
   >show ManagedElement=1,Transport=1,Router=0-28⇒
   -om_sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session⇒
   =VRRP_internal_cee_om_sp,vrrpState
   ```

Result:
Switch B changes state from `MASTER` back to `BACKUP`. This may take up to five minutes.

8. Check VRRP state of switch A.

    a. If using Extreme switches, check the VRRP state of VLAN `cee_ctrl` in switch A:

    ```
    ssh network_admin@<switch_a_ip_address>
    'show vrrp detail | include cee_ctrl'
    ```

    b. If using CMX, issue the following commands:

    Note: The following is an example. For actual values and addresses, refer to the local *IP and VLAN Plan*.

    ```
    >show ManagedElement=1,Transport=1,Router=0-26-om⇒
    _sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session⇒
    =VRRP_internal_cee_om_sp,vrrpState
    vrrpState=MASTER
    >show ManagedElement=1,Transport=1,Router=0-28-om⇒
    _sibb_vr,InterfaceIPv4=vlan1.4022,Vrrpv3Ipv4Session⇒
    =VRRP_internal_cee_om_sp,vrrpState
    vrrpState=BACKUP
    ```

    Result:
    Switch A changes state from `BACKUP` back to `MASTER`.

9. Check that pinging the newly created VM still works.

    Result:
    The VM is still reachable.

10. Check that switches have the correct status after the reboot recovery.

    The command `neutron device-list` shows connected switches.

    a. For Extreme switches, get the status using the following command:

    Note: This step does not apply to BSP deployments.

    ```
    neutron device-show <device-name>
    ```

    b. For CMX, access DMX using the following commands:

    ```
    ssh advanced@<switch_ip_address> -p 2024
    > show ManagedElement=1,DmxcFunction=1,Eqm=1⇒
    ,VirtualEquipment=BSP,Blade=0-26
    > show ManagedElement=1,DmxcFunction=1,Eqm=1⇒
    ,VirtualEquipment=BSP,Blade=0-28
    ```

Result:

- Extreme: Switches are in `ACTIVE` status.
- CMX: `operationalState` is `ENABLED`.

11. Check that *Ethernet Port Fault* alarm has been issued and cleared for the relevant ports. In case of BSP, only hosts in the same shelf as the restarted switch can have alarms.

    ```
    watchmen-client alarm-history
    ```

    Switch-A / CMX-L:
    ```
    eth2 storage all
    dpdk0 traffic all computes
    ```

    Switch-B / CMX-R:
    ```
    eth3 storage all
    dpdk1 traffic all computes
    ```

    Result:
    All expected alarms have been issued and cleared.

12. Repeat above steps but this time reboot switch B.

    Note: If using a large node with four or more switches, repeat the test case for all switches in the data center.

    Nodes with 48 hosts or more can have two traffic switches and two storage switches. Reboot all switches one by one, making all necessary checks in between to ensure the integrity of the node.

    Result:
    Test repeated successfully for all switches.

2.11 Lock or Unlock Blade in a Given Slot

This section is only applicable for BSP multi-server deployments.

Description

This test case ensures that a blade in any given BSP slot can be locked or unlocked without undue disturbance to the CEE system. The lock of a blade is a graceful take down, so ongoing traffic is getting transferred to a backup blade before shutdown. At unlock the blade is powered up again and resumes its place in the cluster. The CEE system maintains normal function throughout.

Table 12 Lock or Unlock Blade in a Given Slot

Objective	**The lock of a blade is a graceful take down, so ongoing traffic is getting transferred to a backup blade before shutdown. At unlock the blade is powered up again and resumes its place in the cluster. The system maintains normal function throughout.**
Precondition	
Postcondition	
Duration	
Platform	• BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Lock a compute blade carrying traffic:
   ```
   ssh advanced@dmx_ip_addres -p 2024
   configure
   ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=⇒
   <tenant_id>,
   Blade=<shelf_id>,<slot_id>,administrativeState=LOCKED
   commit
   ```

   Result:
   The blade is locked.

2. Ensure that the system is running normally after the blade is locked:

   - Run the test case described in Section 2.1 on page 5.

   Result:
   The system is running normally.

3. Unlock the locked blade:
   ```
   configure
   ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=⇒
   <tenant_id>,
   ```

```
Blade=<shelf_id>,<slot_id>,administrativeState=UNLOCKED
commit
```

Result:
The blade is unlocked.

4. Ensure that the system is running normally after the blade is unlocked:

   - Run the test case described in Section 2.1 on page 5.

   Result:
   The system is running normally.

5. Lock a slot where CMX is located (slot 26 or 28 in any subrack):

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=⇒
<tenant_id>,Blade=<shelf_id>,<slot_id>,⇒
administrativeState=LOCKED
commit
```

   Result:
   The blade is locked.

6. Ensure that the system is running normally after the blade is unlocked:

   - Run the test case described in Section 2.1 on page 5.

   Result:
   The system is running normally.

7. Unlock the CMX blade.

   Result:
   The blade is unlocked.

8. Ensure that the system is running normally after the blade is unlocked:

   - Run the test case described in Section 2.1 on page 5

   Result:
   The system is running normally.

9. Lock a slot where SCX is located (slot 0 or 25 in any subrack):

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=⇒
<tenant_id>,Blade=<shelf_id>,<slot_id>,⇒
administrativeState=LOCKED
commit
```

   Result:
   The blade is locked.

10. Ensure that the system is running normally after the blade is unlocked:

- Run the test case described in Section 2.1 on page 5.

Result:
The system is running normally.

11. Unlock the SCX blade.

    Result:
    The blade is unlocked.

12. Repeat the above steps for the blades in your system.

    Result:
    Results are as described above.

2.12 Restart Host where Atlas Executes

This section is only applicable for multi-server deployments where Atlas is installed and running.

Description

Restart the compute host where Atlas VM is running. Check that both the compute host and Atlas VM recovers and that the Atlas GUI is fully functional.

Table 13 Restart Host where Atlas Executes

Objective	**Check that both the compute host and Atlas VM recovers and that the Atlas GUI is fully functional.**
Precondition	• Atlas VM is up and running.
Postcondition	• Atlas VM is up and running.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to Atlas GUI with a browser. Check the status of instances: Select **System Panel - Instances**.

   Result:
   The Atlas VM instance has `Active` status and `Running` power state.

2. Check that the scheduler hints are not preventing the Atlas VM to evacuate. Use the following command to locate the server hosting Atlas:
   `nova list --fields name,status,task_state,host,metadata`

   Alternatively, check this in the Atlas GUI column **Instances>Hosts**.

   Check if vFuel and Atlas are hosted on the same server. If vFuel and Atlas are running on the same compute host, down time is expected to be longer.

   Result:
   The column `Host` in the printout shows the compute host name.

3. Check which bay (for HP), slot (for BSP) or server (for DELL) the compute host resides in.

   The node name is in the following format:

   `HP: compute-<subrack-number>-<bay>`
   `BSP: compute-<subrack-number>-<slot>`
   `Dell: compute-<subrack-number>-<server>`

   Check MAC address number from the `compute-<x>-<x>` host with `ifconfig`. Match this MAC address with the MAC address in iLo or iDRAC.

   Result:
   The bay, slot or server for the Atlas compute host is found.

4. Reset the server.

   a. For Dell platform, use iDRAC. Execute the following commands:

      `http://<atlas_host>`
      `racadm serveraction hardreset`

   b. For HP platform, use iLo to log on to the servers.

      `https://10.0.3.100`

      In the browser window, log on, then select **Device Bays**. Click on the server containing Atlas and select **Virtual device**. Use the **reset** button to reset the server.

c. For BSP platform, log on to the NB DMX, lock and unlock the blade. Execute the following commands:

```
ssh advanced@10.0.10.2 -p 2024
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment⇒
=CEE,Blade=1-3,administrativeState=LOCKED
commit -s

ManagedElement=1,DmxcFunction=1,Eqm=1,⇒
VirtualEquipment=CEE,Blade=1-3,⇒
administrativeState=UNLOCKED
commit
```

Result:
Atlas VM is evacuated to another compute host.

5. Verify that Atlas VM is evacuated according to the policies on the VM by executing the following command:

```
nova list --fields name,status,task_state,host,metadata
```

Result:
Atlas VM is evacuated to another compute host.

6. Log on to Atlas GUI. Check that it is fully functional.

Result:
The Atlas GUI is working normally.

7. Login to Atlas VM. Print status information for monitored services. Check that all the services are running.

```
ssh atlasadm@<atlas_IP>
pwd=<password>
sudo -i
monit summary
monit status
```

Result:
All monitored services statuses are listed.

8. Wait for some minutes, then verify that the restarted compute host has recovered by executing the following command:

```
nova service-list
```

Result:
Status is `enabled` and state is `up`.

2.13 Reboot vCIC in multi-Server Deployment

This section is only applicable for multi-server deployments.

Description

vCIC is rebooted. After the reboot, vCIC functions normally with no negative impact on the system.

Table 14 Reboot vCIC

Objective	**After the reboot, vCIC functions normally with no negative impact on the system.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to the vFuel:

   ```
   ssh <fuel_address>
   fuel node list
   ```

   Result:
   Logged on to vFuel with connections to relevant controllers and hosts.

2. Reboot the vCIC.

   ```
   ssh cic-0-1
   reboot
   ```

   Result:
   vCIC is rebooted.

3. Monitor vCIC to ensure it recovers in 300 seconds.

   Result:

vCIC recovers.

4. Perform a `crm` process check.

   Use `crm_mon -1 -rf` or `crm status`.

   Result:
   Cluster status is displayed.

5. Ensure reboot of vCIC has no impact on normal functioning of the node. Use commands `nova service-list`, `cinder service-list`, and `nova list --all-tenant` from one of the controllers.

   Result:
   Normal services are available.

6. Ensure that traffic is undisturbed by the vCIC reboot.

   Result:
   Traffic is undisturbed.

2.14 Reboot Whole vCIC Cluster

This section is only applicable for multi-server deployments.

Description

vCIC cluster is rebooted. After the reboot, vCICs functions normally with no negative impact on the system.

Table 15 Reboot Whole vCIC Cluster

Objective	**After the reboot, vCICs function normally with no negative impact on the system.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	

Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to the vFuel:

   ```
   ssh <fuel_address>
   fuel node list
   ```

   Result:
   Logged on to vFuel with connections to relevant controllers and hosts.

2. Reboot the vCICs.

   Result:
   vCICs are rebooted.

3. Monitor the vCICs to ensure they recover in 300 seconds.

   Result:
   vCICs recover.

4. Ensure that the reboot of the vCICs has no impact on normal functioning of the node. Use commands `nova service-list`, `cinder service-list`, and `nova list --all-tenant` from one of the controllers.

   Result:
   Normal services are available.

5. Ensure that traffic is undisturbed by the vCIC reboot.

   Result:
   Traffic is undisturbed.

6. Check the recovery of the control functions.

   Result:
   Control functions are recovered.

7. Check the possible impacts on application VMs.

   Result:
   Application VMs are reachable.

2.15 Restart Atlas VM

This section is applicable for all deployments where Atlas is installed and running.

Description

Restart the Atlas VM and check that it recovers and that the GUI is fully functional. No alarm or alert is expected for this kind of interruption.

Table 16 Restart Atlas VM

Objective	**No alarm or alert is expected for this kind of interruption.**
Precondition	• Atlas VM is up and running.
Postcondition	• Atlas VM is up and running.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server • Single server
Executor	
Date and Time ofEExecution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Log on to Atlas GUI with a supported browser. For more information on the supported browser, refer to the GUI description section in *Atlas Dashboard End User Guide*.

   Check the status of instances: Select **System Panel - Instances**.

   Result:
   The Atlas VM instance has `Active` status and `Running` power state.

2. Perform a reboot of the Atlas VM in one of the following ways:

a. Log on to the compute host which hosts the Atlas VM and run the `libvirt` command:

   Note: The *`instance-ID`* to be used is displayed if the command `nova show <atlas_vm_id>` is executed on the vCIC (or any vCIC, in case of multi-server deployment). The command output contains `OS-EXT-SRV-ATTR:instance_name`, where `instance_name` is the *`Instance-ID`*.

b. Or log on to Atlas VM and run one of the following commands:

   - **`reboot`**
   - **`shutdown -r 0`**

c. Or from a vCIC, run the command **`nova reboot --hard`** ***`<VM-ID>`***

d. Or from Atlas GUI **Instances**, select **Actions > More > Hard Reboot Instance** for the Atlas VM.

Result:
The Atlas VM is rebooting.

3. Log on to Atlas GUI. Check that it is fully operational.

   Result:
   The Atlas GUI is fully operational.

4. Log on to the Atlas VM. Print status information for monitored services. Check that all the services are running.

```
ssh atlasadm@<atlas_IP>
pwd = <password>
sudo -i
monit summary
monit status
```

   Result:
   All monitored services status are listed.

5. Perform all the different ways to reboot the Atlas VM as listed in Step 2.

   Result:
   All different rebooting methods has been performed and recovery proved to be working.

2.16 Reboot VM, Soft

This section is applicable for all deployments.

Description

Reboot VMs. Launch a VM for a compute node from an image. Start a test application; in this test case `vi` is used. Do a soft reboot of the VM. Log on to the VM and verify that the test application is stopped. Delete the VM.

Table 17 Reboot VM, Soft

Objective	**Do a soft reboot of the started VM. Login to the VM and verify that the test application is stopped.**
Precondition	• OpenStack is up and running. • Traffic is up and running.
Postcondition	• OpenStack is up and running. • Traffic is up and running.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server • Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Check for compute nodes, zones and images.

   Result:
   List of available items is printed.

2. Launch one VM on a compute node.

   Result:
   VM is started.

3. Do remote SSH on the started VM and start a test application. In this test case `vi` is used.

```
sudo ip netns exec qdhcp-0fa14a9e-65fc-488b-962f-⇒
534353a4e82d ssh <user_name>@<ip_address>
password: <password>
sudo -i
vi
```

Result:
Application is started.

4. Check if application process is running with the command `ps -ef`.

   Result:
   The process is running.

5. Do a soft reboot of the VM.

   ```
   nova reboot --soft <vm-id>
   ```

   Result:
   VM is rebooting. All application processes are killed.

6. Login to the VM and verify that the application is not running any more.

   Result:
   The application is not running anymore.

7. Remove the started VM.

   Result:
   VM is deleted.

2.17 Reboot VM, Hard

This section is applicable for all deployments.

Description

Reboot VMs. Launch a VM for a compute node from an image. Start a test application; in this test case `vi` is used. Do a hard reboot of the VM. Log on to the VM and verify that the test application is stopped. Delete the VM.

Table 18 Reboot VM, Hard

Objective	**VM is rebooted. Test application is stopped.**
Precondition	All expected OpenStack services are running on vCIC and in the host where the vCIC is active (check with command `service --status-all`).

Postcondition	All expected OpenStack services are running on vCIC and in the host where the vCIC is active (check with command `service --status-all`).
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server • Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

1. Check for compute nodes, zones and images.

   ```
   nova list --fields name,status,task_state,host,⇒
   tenant_id,Networks --all-tenants
   glance image-list
   ```

   Result:
   List of available items is printed.

2. Launch one VM on a compute node.

   Result:
   VM is started.

3. Do remote SSH to the started VM and start a test application; in this test case `vi` is used.

   For example, logging into the VM:

   ```
   sudo ip netns exec qdhcp-0fa14a9e-65fc-488b-962f-⇒
   534353a4e82d ssh <user_name>@<ip_address>
   password: <password>
   sudo -i
   vi
   ```

Result:
The application is started.

4. Check if the application process is running using the command `ps -ef`.

   Result:
   The process is running.

5. Execute a hard reboot of the VM.

   ```
   nova reboot --hard <vm-id>
   ```

   Result:
   VM is rebooting. All application processes are killed.

6. Login to the VM and verify that the application is not running anymore.

   Result:
   The application is not running anymore.

7. Remove the started VM.

   Result:
   The VM is deleted.

2.18 VM Stop and VM Start

This section is applicable for all deployments.

Description

Create a VM, stop it, start it again and finally remove it.

Table 19 VM Stop and VM Start

Objective	**Create a VM, stop it, start it again and finally remove it.**
Precondition	
Postcondition	
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server, single servr • Single server
Executor	
Date and Time of Execution	

Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Upload image to Glance on vCIC, or use the existing image:
   ```
   glance image-create --name Ubuntu_14.04.4_LTS --progress ⇒
   --disk-format qcow2 --container-format bare ⇒
   <trusty-server-cloudimg-amd64-disk1.img
   ```

   Result:
   The image is uploaded to Glance.

2. Create a `cloud_config.txt` file with the following content:

   ```
   #cloud-config.txt
   password: rootroot
   chpasswd: { expire: False }
   ssh_pwauth: True
   ```

   Result:
   The text file is created.

3. Create a network and subnetwork:
   ```
   neutron net-create <net-name>
   neutron subnet-create --name <subnet-name> <net-name> ⇒
   10.0.3.0/24
   ```

   Result:
   The network and subnetwork are created.

4. Start a new VM in one of the compute hosts.

   ```
   nova boot <VM_name> --image <image-id> --use
   r-data /tmp/⇒
   cloud_config.txt --<flavor-id> --nic net-id=<net-id>
   ```

   Result:
   A new VM is started.

5. Check that the VM is created successfully.

   Result:
   VM is in `active` status.

6. Ping the VM.

   Result:
   The VM is reachable.

7. Stop the VM.

   Result:
   The VM is stopped, and is in status `SHUTOFF`.

8. Start the VM again.

   Result:
   The VM is back in `active` status.

9. Ping the VM.

   Result:
   The VM is reachable.

10. Do a cleanup of the system, delete the VM.

    Result:
    VM is deleted.

2.19 Backup and Restore Atlas in multi-Server Deployment

This section is only applicable for multi-server deployments where Atlas is installed and running.

Description

Atlas backup is created according to *Atlas Backup*. Atlas VM is deleted. A new Atlas VM is launched. After Atlas VM is active, the backup files are downloaded from Swift to Atlas VM and a restore is initiated according to *Atlas Backup*.

Table 20 Backup and Restore Atlas

Objective	**Do a successful Atlas backup and restore.**
Precondition	• Atlas GUI is fully functional.
Postcondition	• Atlas GUI is fully functional.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	

Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Login to Atlas VM with SSH. Create a text file in the `/root` directory.

   ```
   ssh root<atlas_IP_address>

   sudo -i
   echo 'Test of restore of Atlas environment' >> test_file.txt
   ```

   Result:
   Logged on to Atlas and the text file `test_file.txt` is created.

2. Back up Atlas, as described in *Atlas Backup*.

   Result:
   The following is an example of the output:

   Atlas Backup

   ```
   atlasadm@atlas:~$ sudo atlas backup-create --name
   atlas_backup --p password
   Running Atlas backup ...
   Done.
   atlasadm@atlas:~$ sudo atlas backup-list
      ID   Name   Date
   1465911268   atlas_backup   Tue Jun 14 15:34:28 CET 2016
   ```

   Uploading Backup to Swift

   ```
   root@atlas:/var/archives# swift upload AtlasBackups ⇒
   *1480183339*
   atlas_backup1480183339/atlas_backup.1480183339.sha256.enc
   atlas_backup1480183339/atlas_backup.1480183339-all-mysql-⇒
   databases.sql.bz2.enc
   atlas_backup1480183339/atlas_backup.1480183339-etc-puppet-⇒
   hieradata-passwords.yaml.master.tar.gz.enc
   atlas_backup1480183339/atlas_backup.1480183339-root.⇒
   master.tar.gz.enc
   atlas_backup1480183339/atlas_backup.1480183339-home-⇒
   atlasadm.master.tar.gz.enc
   ```

3. Check in one of the vCICs that the Atlas backup is uploaded to Swift, using the following commands:

   ```
   swift list --lh
   swift list --long AtlasBackups
   ```

   Result:
   An example of the printout is:

   ```
   root@cic-1:~# swift list --lh
       6 897K 2016-01-14 07:51:32 AtlasBackups
       6 897K
   root@cic-1:~# swift list --long AtlasBackups
         724815 2016-11-25 14:14:19 application/octet-stream ⇒
   AtlasBackup1480083137/AtlasBackup.1480083137-all-mysql-⇒
   databases.sql.bz2.enc
            500 2016-11-25 14:14:19 application/octet-stream ⇒
   AtlasBackup1480083137/AtlasBackup.1480083137-etc-puppet-⇒
   hieradata-passwords.yaml.master.tar.gz.enc
      289296735 2016-11-25 14:14:19 application/octet-stream ⇒
   AtlasBackup1480083137/AtlasBackup.1480083137-home-atlasadm.⇒
   master.tar.gz.enc
      201311200 2016-11-25 14:14:18 application/octet-stream ⇒
   AtlasBackup1480083137/AtlasBackup.1480083137-root.⇒
   master.tar.gz.enc
            890 2016-11-25 14:14:18 application/octet-stream ⇒
   AtlasBackup1480083137/AtlasBackup.1480083137.sha256.enc
      491334140
   ```

4. Delete the Atlas VM using the following command:

   ```
   nova delete <atlas_VM_ID>
   ```

   Result:
   The Atlas VM is deleted.

5. Launch a new Atlas VM as described in*Atlas SW Installation*.

   Result:
   The Atlas VM is `ACTIVE` and reachable.

6. Log on to the Atlas VM with SSH.

   Result:
   Logged on to the Atlas VM.

7. Download the backup files from Swift according to *Atlas Restore*.

   Result:
   The following is an example of the output:

   Atlas Restore

```
root@atlas:~# atlas backup-list
```

Downloading Backup Files from Swift

```
root@atlas:~# cd /var/archives/

root@atlas:/var/archives/# swift list AtlasBackups | ⇒
grep 1465911268
atlas_backup1465911268/atlas_backup.1465911268-all-⇒
mysql-databases.sql.bz2.enc
atlas_backup1465911268/atlas_backup.1465911268-etc-⇒
puppet-hieradata-passwords.yaml.master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268-home-⇒
atlasadm.master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268-⇒
root.master.tar.gz.enc
atlas_backup1465911268/atlas_backup.1465911268.⇒
sha256.enc

root@atlas:/var/archives# swift download AtlasBackups
-p atlas_backup1465911268

root@atlas:/var/archives# sudo atlas backup-list
  ID              Name                  Date
1465911268     atlas_backup    Wed Sep 30 09:03:00 UTC 2015

root@atlas:/var/archives# atlas backup-restore ⇒
--d 1465911268 --p
<backup_password>
Atlas has been set for restore.
Please reboot.

root@atlas:/var/archives# reboot

Broadcast message from root@atlas
        (/dev/pts/4) at 9:45 ...

The system is going down for reboot NOW!
```

8. When the reboot is finished, log on to the Atlas VM. Verify that the created text file has been restored.

   ```
   ssh root@<atlas_IP_address>
   ls -l
   cat test_file.txt
   ```

   Result:
   Logged on to Atlas, the `test_file.txt` is listed and contains: “Test of restore of Atlas environment”

9. From the GUI verify that all panels are functional.

   Result:
   The web GUI is fully functional.

2.20 Migrate VM with nova migrate Command

This section is only applicable for multi-server deployments.

Description

Migration allows moving a VM from one compute host to another when compute hosts are up. Use `nova migrate` or `nova forcemove` commands for migration.

Table 21 Migration of VMs with nova migrate Command

Objective	**Migration of VMs with `nova migrate` command**
Precondition	• All expected OpenStack services are up and running.
Postcondition	• All expected OpenStack services are up and running.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

1. Select a compute host that will be used.

   ```
   nova service-list
   ```

   Result:
   Compute host is selected and noted. Its state is `up`.

2. If not already uploaded, upload image to Glance on vCIC:
   `glance image-create --name Ubuntu_14.04.4_LTS --progress ⇒`

--disk-format qcow2 --container-format bare ⇒
<trusty-server-cloudimg-amd64-disk1.img
Alternatively, an existing image can be used.

Result:
The image is uploaded to Glance.

3. Create seven Cinder volumes from an image and wait until they reach AVAILABLE status:
cinder create --image-id *<image_id>* --display-name VM_Volume_*<X>* *<volume_size>*,

   where *<X>* is replaced with 1 to 7 matching the name of the VM.

   Use the command `cinder list` to verify.

   Result:
   Cinder volumes are added, status of Cinder volumes AVAILABLE and Volume name are noted.

4. To set the initial password for the Ubuntu user for the trusty-server-cloudimg-amd64-disk1.img image, create a cloud_config.txt file with the following content :

```
#cloud-config.txt
password: rootroot chpasswd: { expire: False }
ssh_pwauth: True
```

   Result:
   Text file is created.

5. Boot first VM (VM1) from VM_Volume_1 on the host selected above. Use the following commands:

nova boot --availability-zone nova:*<host-id>* ⇒
--block-device-mapping⇒
<vda>=*<VM_Volume_id>*:*<type>*:*<size(GB)>*:⇒
***<delete-on-terminate>* flavor *<flavor>* --nic net-id=⇒**
***<network>* --user-data /tmp/cloud_config.txt ⇒**
--meta ha-policy=ha-offline VM1

   Use the command **nova show VM1**.
   Use 1 for parameter <delete-on-terminate> so that the volume is deleted afterwards.

   Result:
   VM1 is started successfully.

6. Boot VM2 from VM_Volume_2 with the server groups policy (hint: server_group).

   Note: Scheduler hints same_host and different_host are also supported, but deprecated.

Use the command:

```
nova boot --hint server_group=<VM1 uuid>⇒
--block-device-mapping⇒
 <vda>=<VM_Volume_2_id>:<type>:<size(GB)>:
<delete-on-terminate> --flavor <flavor> --nic net-⇒
id=<network> --user-data /tmp/cloud_config.txt ⇒
--meta ha-policy=ha-offline VM2
```

Use *nova list* to verify that VM2 is on same host as VM1. Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

Result:
VM2 is started successfully.

7. Check that the VMs are running on the selected host. Use command `nova show VMx`.

   Result:
   The VMs are running on the same host.

8. Boot VM3 from `VM_Volume_3`. Use the following commands:

   ```
   nova boot
   --block-device-mapping
   <vda>=<VM_Volume_3_id>:<type>:<size(GB)>:
   <delete-on-terminate> --flavor=<flavor> --nic net-⇒
   id=<network> --user-data /tmp/cloud_config.txt ⇒
   --meta ha-policy=ha-offline VM3
   ```

   Use `nova show VM3` Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

   Result:
   VM3 is started successfully.

9. Boot VM4 from `VM_Volume_4`. Use the following commands:

   ```
   nova boot --hint different_host=<VM3 uuid>
   --block-device-mapping
   <vda>=<VM_Volume_4_id>:<type>:<size(GB)>:⇒
   <delete-on-terminate> --flavor=<flavor> --nic net-id=⇒
   <network> --user-data /tmp/cloud_config.txt ⇒
   --meta ha-policy=ha-offline VM4
   ```

   Use `nova show VM4` to verify that VM4 is on a different host from VM3. Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

   Result:

VM4 is started successfully.

10. Boot VM5 from `VM_Volume_5` on the host selected above with High Availability (HA) policy `managed-on-host`. Use the following commands:

    ```
    nova boot --availability-zone nova:<host-id>
    --block-device-mapping
    <vda>=<VM_Volume_5_id>:<type>:<size(GB)>:<delete-⇒
    on-terminate> --flavor=<flavor> --nic net-⇒
    id=<network> --user-data /tmp/cloud_config.txt ⇒
    --meta ha-policy=managed-on-host VM5
    ```

    Use `nova show VM5`. Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

    Result:
    VM5 is started successfully.

11. Boot VM6 from `VM_Volume_6` with the server groups policy and with HA policy `managed-on-host`. Use the following command:

    ```
    nova boot --hint server_group=<VM5 uuid>
    --block-device-mapping
    <vda>=<VM_Volume_6_id>:<type>:<size(GB)>:⇒
    <delete-on-terminate> --flavor=<flavor>⇒
     --nic net-id=<network> --user-data /tmp/cloud_config.txt ⇒
    --meta ha-policy=managed-on-host VM6
    ```

    Use **`nova list`** to verify that VM6 is on same host as VM5. Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

    Result:
    VM6 is started successfully.

12. Boot VM7 from `VM_Volume_7` without any HA_policy. Use the following command:

    ```
    nova boot --block-device-mapping <vda>=<VM_
    Volume_7_id>:⇒
    <type>:<size(GB)>:<delete-on-terminate>⇒
     --flavor=<flavor> --nic net-id=<network>⇒
     --user-data /tmp/cloud_config.txt VM7
    ```

    Use **`nova list`** to verify that VM7 is created with any HA policy. Use `1` for parameter `<delete-on-terminate>` so that the volume is deleted afterwards.

    Result:

VM7 is started successfully.

13. Ping the previously created VMs.

    Use, for example, command `ip netns exec qdhcp-<net-id> ping -c 2 <server-address_VMx>`.

    Result:
    It is possible to ping the VMs.

14. Log on to the previously created VMs. Create test files in the Cinder volumes of the VMs Use, for example, the following commands:

```
ip netns exec qdhcp-<net-id> ssh <user>@<server-address_VMx>
mkdir testDir
cd testDir
echo "this text is written prior to migration" >
testfile.txt
cat testfile.txt
```

    Result:
    Test files are created in all VMs.

15. Migrate VM1.

    a. Use the following command with `poll` parameter to check the migration process:

       nova migrate VM1 --poll Check afterwards using command nova show VM1

    b. Confirm the migration:

       nova resize-confirm VM1

    Result:
    VM1 is migrated to another host.

16. Migrate VM2.

    a. Use the command with `poll` parameter to check the migration process:

       nova migrate VM2 --poll

    b. Check the result using the following command:

       nova show VM2

    c. Confirm the migration:

```
nova resize-confirm VM2
```

Result:
VM2 is also migrated to the same host as VM1.

17. Migrate VM3.

    a. Use command with poll parameter to check the migration process:

    ```
    nova migrate VM3 --poll
    ```

    b. Check the result using the following command:

    ```
    nova show VM3
    ```

    c. Confirm the migration:

    ```
    nova resize-confirm VM3
    ```

    VM3 is migrated to another host.

Result:
VM4 is migrated to a different host than VM3.

18. Migrate VM4.

    a. Use the following command with `poll` parameter to check the migration process:

    ```
    nova migrate VM4 --poll
    ```

    b. Check the result using the following command:

    ```
    nova show VM4
    ```

    c. Confirm the migration:

    ```
    nova resize-confirm VM4
    ```

Result:
VM4 is migrated to a different host than VM3.

19. Migrate VM5.

    a. Use the following command with `poll` parameter to check the migration process:

    ```
    nova migrate VM5 --poll
    ```

    b. Check the result using the following command:

    ```
    nova show VM5
    ```

c. Confirm the migration:

```
nova resize-confirm VM5
```

Result:
VM5 is migrated to another host independently of HA policy `managed-on-host` defined during creation of VM5.

20. Migrate VM6.

    a. Use the following command with `poll` parameter to check the migration process:

    ```
    nova migrate VM6 --poll
    ```

    b. Check the result using the following command:

    ```
    nova show VM6
    ```

    c. Confirm the migration:

    ```
    nova resize-confirm VM6
    ```

    Result:
    VM is migrated.

21. Migrate VM7.

    a. Use the following command with `poll` parameter to check the migration process:

    ```
    nova migrate VM7 --poll
    ```

    b. Check the result using the following command:

    ```
    nova show VM7
    ```

    c. Confirm the migration:

    ```
    nova resize-confirm VM7
    ```

    Result:
    VM7 is migrated as HA policy is not considered for `nova migrate`.

22. Migrate VM7.

    a. This time use the command `nova forcemove` to see the difference in the behavior of the two commands. This command will be tested in deep in `ecsciv-309`.

    ```
    nova forcemove VM7
    ```

    b. Check the result using the following command:

nova show VM7

Result:
VM7 is not migrated as `nova forcemove` commands takes into consideration the HA policy defined during VM creation.

23. Print the list of migrations. Use the command `nova migration-list`.

    Result:
    The list of migrations is printed.

24. Log onto the VMs and check that the test files on the VMs are still readable Use, for example, the following commands:

    ```
    ip netns exec qdhcp-<net-id>
    ssh <user>@<server-address_VMx>
    cat testDir/testfile.txt
    ```

    Result:
    String `this text is written prior to migration` is read from the text files.

25. Delete the VMs. Execute the following commands:

    nova delete VMx and
    nova list to verify
    cinder list

    Result:
    VMs and boot volumes are deleted.

2.21 Migrate VM to Unspecified Host Using nova forcemove Command

This section is only applicable for multi-server deployments.

Description

Four VMs are booted from volume:

- VM1 with HA policy set to `managed-on-host`
- VM2 with HA policy set to `ha-offline`
- VM3 on the same host than VM1 and with HA policy set to `ha-offline`
- VM4 with no HA policies set

It is checked on which hosts the VMs are created and text files are created in the Cinder volumes of the VMs. Then the `nova forcemove` command is

applied to the VMs. It is checked if the VMs are migrated to another host and if the text files of the VMs are still readable after the migration.

Table 22 Migrate VM to Unspecified Host Using nova forcemove Command

Objective	**VMs are migrated to another host and the text files of the VMs are readable after the migration.**
Precondition	• All expected OpenStack services are up and running.
Postcondition	• All expected OpenStack services are up and running.
Duration	
Platform	• Dell multi-server • HP multi-server • BSP multi-server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	☐ OK ☐ Not OK ☐ N/A
Comment	

Procedure

Do the following:

1. Create four bootable Cinder volumes from an image and wait until they reach `AVAILABLE` status:
   `cinder create --image-id <image_id> --display-name <volume_nameX> <volume_size>`

   Issue the following command for verification:
   `cinder list`

   Result:
   Cinder volumes are added, status of Cinder volumes is `AVAILABLE` and *`<volume_idX>`* is noted.

2. To set the initial password for the Ubuntu user for the `trusty-server-cloudimg-amd64-disk1.img` image, create a `cloud_config.txt` file with the following content :
   ```
   #cloud-config.txt
   password: rootroot chpasswd: { expire: False }
   ssh_pwauth: True
   ```

Result:
Text file is created.

3. Boot VM1 from the first volume with HA policy set to `managed-on-host`, wait until it reaches `ACTIVE` status and check on which host VM1 was created:
```
nova boot --flavor <flavor_name> --nic net-id=<net_id> \
  --block-device-mapping <vda>=<volume_id1>:⇒
<type>:<size_in_GB>:<delete_on_terminate> \
 --user-data /tmp/cloud_config.txt  ⇒
--meta st16a-309 ha-policy=managed-on-host VM1
nova show VM1
```

   Result:
   VM1 with `managed-on-host` policy is created and is `ACTIVE`, the host for VM1 and `VM1_ID` noted.

4. Boot VM2 from the second volume with ha-policy set to `ha-offline`, wait until it reaches `ACTIVE` status and check on which host VM2 was created:
```
nova boot --flavor <flavor_name> --nic net-id=<net_id> \
  --block-device-mapping <vda>=<volume_id2>:⇒
<type>:<size_in_GB>:<delete_on_terminate> \
 --user-data /tmp/cloud_config.txt ⇒
--meta ha-policy=ha-offline VM2
nova show VM2
```

   Result:
   VM2 is created and is `ACTIVE`, the host for VM2 is noted.

5. Boot VM3 from the third volume on the same host as VM1 and with HA policy set to `ha-offline`. Wait until it reaches `ACTIVE` status and verify that VM3 was created on the same host as VM1:
```
nova boot --flavor <flavor_name> --nic net-id=<net_id> \
  --block-device-mapping <vda>=<volume_id3>:⇒
<type>:<size_in_GB>:<delete_on_terminate> \
   --hint server_group=<VM1_id> --user-data ⇒
/tmp/cloud_config.txt --meta ha-policy=⇒
ha-offline VM3
nova show VM3
```

   Result:
   VM3 is created on the same host as VM1 and is `ACTIVE`, the host for VM3 is noted.

6. Boot VM4 from the fourth volume and wait until it reaches `ACTIVE` state:
```
nova boot --flavor <flavor_name> --nic net-id=<net_id> \
  --block-device-mapping <vda>=<volume_id4>:⇒
<type>:<size_in_GB>:<delete_on_terminate> ⇒
--user-data /tmp/cloud_config.txt VM4
nova show VM4
```

   Result:

VM4 is created and is ACTIVE.

7. Log onto the previously created VMs and create test files in the Cinder volumes of the VMs:
   ip netns exec qdhcp-*<net_id>* ssh *<user>*@*<server_address_VMx>*
   mkdir testDir
   cd testDir
   echo "this text is written prior to migration" > testfilex.txt

   where *<VMx>* is replaced with the name of the VM.

   Result:
   Test files are created.

8. Try to migrate VM1 using the `nova forcemove` command and check that VM1 is not migrated to another host because the `managed-on-host` policy was set:
   nova forcemove VM1
   nova show VM1

   Result:
   Check that the results are the following:

   - The printout of `nova forcemove` command shows the following:
     `Move accepted = True`
     `Needs start = True`
   - VM1 is not moved to another host.
   - The status of VM1 is SHUTOFF.

   As the `managed-on-host` policy was set for this VM, the VM was shut down, but was not moved anywhere. Therefore, the VM needs to be started again.

9. Restart VM1:
   nova start VM1

   Do the following for verification:
   nova list

   Result:
   VM1 is ACTIVE again.

10. Log onto VM1 and check that the test file is still readable:
    ip netns exec qdhcp-*<net_id>*
    ssh *<user>*@*<server_address_VM1>*
    cat testDir/testfile1.txt

    Result:
    String ‘‘this text is written prior to migration’’ is read from text file.

11. Migrate VM2 using the `nova forcemove` command and check that VM2 was migrated to another host:
    `nova forcemove VM2`
    `nova show VM2`

    Do the following when the status of VM2 is `VERIFY_RESIZE`:
    `nova resize-confirm VM2`

    Because the `ha-offline` policy was set for this VM, the VM undergoes a "cold migration", so the status changes from `ACTIVE` to `VERIFY_RESIZE` during the migration.

    Note: The `nova host-maintenance` feature confirms the resize after `forcemove` automatically.

    Result:
    Check that the results are the following:

    - The result printout of `nova forcemove` command shows the following:
      ```
      Move accepted = True
      Needs start = False
      ```
    - VM2 is moved to another host.
    - The status of VM2 is `ACTIVE`.

12. Try to migrate VM3 using the `nova forcemove` command and check that VM3 cannot be migrated to another host because it has to be on the same host as VM1:
    `nova forcemove VM3`
    `nova show VM3`

    Result:
    Check that the results are the following:

    - The result printout of `nova forcemove` command shows the following:
      ```
      Move accepted = False
      Error message = No valid host was found
      ```
    - VM3 is not moved to another host.

13. Migrate VM3 using the `nova forcemove` command with the option `--ignore-hints` and check that VM3 was migrated to another host:
    `nova forcemove --ignore-hints VM3`
    `nova show VM3`

    Do the following when status of VM3 is `VERIFY_RESIZE`:
    `nova resize-confirm VM3`

    Result:
    Check that the results are the following:

    - The result printout of `nova forcemove` command shows the following:

```
Move accepted = True
Needs start = False
```

- VM3 is moved to another host.
- The status of VM3 is `ACTIVE`.
- VM3 still shows the dependency to VM1.

14. Delete VM1 and verify the action:
    `nova delete VM1`
    `nova list`

    Result:
    VM1 is deleted.

15. Try to migrate VM4 using the `nova forcemove` command and verify that VM4 is not migrated:
    `nova forcemove VM4`
    `nova show VM4`

    As no policy was provided by metadata during boot, the default policy is applied during the `forcemove`. In this case, `nova forcemove` initiates live migration, but as live migration is not supported in R6, the move is not accepted.

    Result:
    Check that the results are the following:

    - The result printout of `nova forcemove` command shows the following:
      ```
      Move accepted = False
      Needs start = False
      ```
    - VM4 is not moved to another host.
    - The status of VM4 is `ACTIVE`.

16. Log onto the remaining VMs and check that the test files on the VMs are still readable:
    `ip netns exec qdhcp-`*`<net_id>`*
    `ssh `*`<user>@<server_address_VMx>`*
    `cat testDir/testfilex.txt`

    where *`<VMx>`* is replaced with the name of the VM.

    Result:
    String “this text is written prior to migration” is read from text files.

17. Delete the VMs and verify the result:
    `nova delete VM`*`<x>`***
    `nova list`

    Result:

The VMs are deleted.

18. Delete the Cinder volume and verify the result:
    cinder delete *<volume_id>*
    cinder list

    Result:
    The volume is deleted.