

# Customer Acceptance Test Specification

Cloud Execution Environment

TEST SPECIFICATION

**Copyright**

© Ericsson AB 2016–2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Scope	1
1.2	Intended Audience	4
1.3	Prerequisites	4
<b>2</b>	<b>Test Cases</b>	<b>6</b>
2.1	Health Check	6
2.2	Reboot vFuel	8
2.3	Reboot vCIC in Single Server Deployment	10
2.4	Reboot Region Switch in Single Server Deployment	12
2.5	Reboot Single Server Host with VM Configured	14
2.6	Backup and Restore Atlas in Single Server Deployment	17
2.7	Reboot Compute Host and Recover vFuel	20
2.8	Reboot Compute Host and Recover vCIC	24
2.9	Reboot vCIC Host with Master RabbitMQ Node	27
2.10	Reboot Region Switch in Multi-Server Deployment	34
2.11	Lock or Unlock Switch Blade in Given BSP Slot	39
2.12	Restart Host Where Atlas Executes	42
2.13	Reboot vCIC in Multi-Server Deployment	45
2.14	Reboot Whole vCIC Cluster	47
2.15	Restart Atlas VM	48
2.16	Reboot VM, Soft	51
2.17	Reboot VM, Hard	52
2.18	VM Stop and VM Start	54
2.19	Backup and Restore Atlas in Multi-Server Deployment	56
2.20	Migrate VM with nova migrate Command	59
2.21	Migrate VM to Unspecified Host Using nova forcemove Command	67
2.22	Verify That Migrated vFuel VMs Are Synchronized after CEE Deployment	73
2.23	Verify That Manual vFuel Handover Is Successful	74
2.24	Verify vFuel Image Synchronization Instructions	76
2.25	Verify Rollback to Previous (Passive) vFuel Version	78
2.26	End-to-End Connectivity with CSS	79



2.27	End-to-End Connectivity with SR-IOV	83
2.28	End-to-End Connectivity with PCI PT	89
2.29	Verify IPv6 Address as Allowed Address Pair	93
2.30	Verify IPv6 Ingress Rules	99
2.31	Verify Software RAID Configuration	104
2.32	Check Scale-In Functionality on Compute Hosting vFuel	105
2.33	Check Scale-In Functionality on Compute Hosting vCIC	106



# 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) functionalities 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
- HDS deployment

Table 1 Test Cases for Supported Deployments

Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell)	HDS
Health Check, see Section 2.1 on page 6	Yes	Yes	
Reboot vFuel, see Section 2.2 on page 8	Yes		
Reboot vCIC in Single Server Deployment, see Section 2.3 on page 10	Yes		
Reboot Region Switch in Single Server Deployment, see Section 2.4 on page 12	Yes		
Reboot Single Server Host with VM Configured, see Section 2.5 on page 14	Yes		
Backup and Restore Atlas in Single Server Deployment <sup>(1)</sup> , see Section 2.6 on page 17	Yes		



Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell)	HDS
Reboot Compute Host and Recover vFuel, see Section 2.7 on page 20		Yes	
Reboot of Compute Host and Recover vCIC, see Section 2.8 on page 24		Yes	
Reboot vCIC Host with Master RabbitMQ Node, see Section 2.9 on page 27		Yes	
Reboot Region Switch in Multi-Server Deployment, see Section 2.10 on page 34		Yes	
Lock or Unlock Switch Blade in Given BSP Slot, see Section 2.11 on page 39		Yes <sup>(2)</sup>	
Restart Host Where Atlas Executes <sup>(1)</sup> , see Section 2.12 on page 42		Yes	
Reboot vCIC in Multi-Server Deployment, see Section 2.13 on page 45		Yes	
Reboot Whole vCIC Cluster, see Section 2.14 on page 47		Yes	
Restart Atlas VM <sup>(1)</sup> , see Section 2.15 on page 48	Yes	Yes	
Reboot VM, Soft, see Section 2.16 on page 50	Yes	Yes	
Reboot VM, Hard, see Section 2.17 on page 52	Yes	Yes	
VM Stop and VM Start, see Section 2.18 on page 54	Yes	Yes	
Backup and Restore Atlas in Multi-Server Deployment <sup>(1)</sup> , see Section 2.19 on page 56		Yes	
Migrate VM with Nova migrate Command, see Section 2.20 on page 59		Yes	



Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell)	HDS
Migrate VM to Unspecified Host Using Nova forcemove Command, see Section 2.21 on page 67		Yes	
Verify That Migrated vFuel VMs Are Synchronized after CEE Deployment, see Section 2.22 on page 73		Yes	
Verify That Manual vFuel Handover Is Successful, see Section 2.23 on page 74		Yes	
Verify vFuel Image Synchronization Instructions, see Section 2.24 on page 76		Yes	
Verify Rollback to Previous (Passive) vFuel Version, see Section 2.25 on page 77		Yes	
End-to-End Connectivity Using CSS see Section 2.26 on page 79	Yes	Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
End-to-End Connectivity with SR-IOV see Section 2.27 on page 83		Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
End-to-End Connectivity Using PCI-PT see Section 2.28 on page 89		Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
Verify IPv6 Address as Allowed Address Pair see Section 2.29 on page 93			Yes
Verify IPv6 Ingress Rules see Section 2.30 on page 99			Yes
Verify Software RAID Configuration see Section 2.31 on page 103		Yes <sup>(3)</sup>	Yes <sup>(3)</sup>



Test Case	Single Server Deployment	Multi-Server Deployment (BSP, Dell)	HDS
Check Scale-In on Compute Hosting vFuel see Section 2.32 on page 104		Yes	Yes
Check Scale-In on Compute Hosting vCIC see Section 2.33 on page 106		Yes	Yes

(1) This test case can only be performed if Atlas is installed and running.

(2) This test is only applicable for BSP deployments.

(3) This test is not applicable for all deployments. See test description.

(4) This test case can only be performed on HDS if extra interfaces are available.

## 1.2 Intended Audience

This document is primarily aimed at skilled professionals from the following groups:

- Installation personnel
- Site operations personnel
- Ericsson personnel installing CEE at customer sites

The user of this document is assumed to have basic understanding of OpenStack and its functionality. For an overview of OpenStack, refer to the following documents:

- OpenStack End User Guide
- OpenStack Administrator Guide
- OpenStack API Complete Reference

## 1.3 Prerequisites

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

### 1.3.1 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.

### 1.3.2 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.
- `sudo` privileges are available.
- RealVNC Viewer is available. The installer can be downloaded from <https://www.realvnc.com/en/connect/download/vnc/>.

### 1.3.3 Documents

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

- Atlas Backup
- Atlas Restore
- Atlas SW Installation
- Fuel Synchronization

### 1.3.4 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.

It is recommended to execute a **system health check** before and after each test case to ensure the state of the DC remains unaltered by the test case executed. A health check is considered successful when each module displays a [PASSED] verdict. (For some modules [SKIPPED] is the expected verdict.) For more information, refer to the [Health Check Procedure](#) document.

**Note:** The execution time of the health check procedure depends on the size of the CEE region.

### 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.

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

— **Check system health:**

Check that the CEE operates in a fault-free state, provides the required functionality, and is available for the users.

Table 2 Health Check

<b>Objective</b>	<b>Health check is run to ensure a DC is operational and ready for service.</b>
<b>Precondition</b>	



Postcondition	
Duration	
Platform	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> <li>• Single server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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:

Execute the following command on vFuel as root:

— `[root@fuel ~]# healthcheck.py`

The procedure checks the system health and displays the verdicts of each module. A detailed log of the health check can be found in the `/var/log/healthcheck/<log_filename>` directory.

For more information, refer to the [Health Check Procedure](#) document.

## 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	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 <vfuel_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 -s /dev/null)"
checking with command "pgrep puppet"
ostf is ready.
checking with command "ps auxx | grep -q 'cobblerd -f' &amp; &amp;pgrep dnsmasq"
1456
checking with command "cobbler profile find --name=ubuntu" | grep -q ubuntu & &pgrep cobbler profile find --name=bootstrap" | grep -q bootstrap"
checking with command "pgrep puppet"
cobbler is ready.
checking with command "curl -f -L -u "nailgy0jhr2U4vLeej32nm0LG4jq6" http://127.0.0.1:15672/api/nodes 1>/dev/null 2>&pgrep 1"
checking with command "curl -f -L -u "mcollective:He100DL5iKhYqUe2X53oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_broadcast'"
checking with command "curl -f -L -u "mcollective:He100DL5iKhYqUe2X53oaJC" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_directed'"
checking with command "pgrep puppet"
rabbitmq is ready.
checking with command "FOFASWORD=qfWQ5YiFaJFwF02HuFFyzf5 /usr/bin/pgrep -h 192.168.0.11 -U "nailgun" "nailgun" -c '%copyright' 2>&pgrep 1 1>/dev/null"
checking with command "pgrep puppet"
postgres is ready.
checking with command "ps auxx | grep -q 'astute'"
checking with command "curl -f -L -u "nailgy0jhr2U4vLeej32nm0LG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'nailgun'"
checking with command "curl -f -L -u "nailgy0jhr2U4vLeej32nm0LG4jq6" -s http://127.0.0.1:15672/api/exchanges | grep -qw 'nailgy_service'"
checking with command "pgrep puppet"
astute is ready.
checking with command "ps auxx | grep -q 'mcollectived'"
checking with command "pgrep puppet"
mcollective is ready.
checking with command "ps auxx | 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 "AgQ43CI3x7g1Cd5x1URp86eq" token-get &pgrep >>/dev/null"
checking with command "pgrep puppet"
keystone is ready.
checking with command "netstat -nl | grep -q 514"
checking with command "pgrep puppet"
zaylog is ready.
checking with command "netstat -ntl | grep -q 873"
checking with command "pgrep puppet"
zaync 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

<b>Objective</b>	<b>After the reboot, vCIC functions normally with no negative effects on any part of the system.</b>
<b>Precondition</b>	



Postcondition	
Duration	
Platform	• Single server
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

### Procedure

Do the following:

1. Log on to vFuel:  

```
ssh root@<vfuel_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

<b>Objective</b>	<b>At the end of the reboot, a running VM can be pinged.</b>
<b>Precondition</b>	
<b>Postcondition</b>	<ul style="list-style-type: none"><li>• VMs are up and running on the compute host.</li><li>• VM is reachable.</li></ul>
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"><li>• Single server</li></ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

**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_config.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
```

**Result:**

Switch state is MASTER. The following is an example of the output:

```
DC190_SWA_X670V.1 # show vrrp detail | include cee_ctrl
VLAN: cee_ctrl_sp VRID: 252 VRRP: Enabled State: MASTER
```

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	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 remains on the compute host and has 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:**



```
p_active_mark (lsb:active_marker):    =>
Started cic-1.domain.tld
```

- 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-0-1/,/<\measValue>/' /var/cache/pmreports/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	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 AtlasBackup --p password
```

```
Tue 02 May 2017 08:48:23 INFO: Creating Backup "AtlasBackup" =>  
with ID=1493707703...
```

```
Tue 02 May 2017 08:48:23 INFO: Backing up password =>  
configuration, home directories and database...
```

```
Tue 02 May 2017 08:48:58 INFO: Done.
```

```
Tue 02 May 2017 08:49:05 INFO: Encrypting...
```

```
Tue 02 May 2017 08:49:05 INFO: Done.
```

```
Tue 02 May 2017 08:49:05 INFO: Uploading Backup "AtlasBackup" =>  
of size 654M to Swift...
```

```
Tue 02 May 2017 08:49:15 INFO: Done.
```

```
Tue 02 May 2017 08:49:15 INFO: Backup Complete!
```

```
atlasadm@atlas:~$ sudo atlas backup-list
```



```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID          | Name          | Storage |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Size | Date       | Time       |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1493707703 | AtlasBackup   | LOCAL     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 654M  | 2-May-2017 | 08:49:05 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

**Note:** Record the ID of the backup from the printout as it is required in a later step of the procedure.

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

```
swift list AtlasBackups | grep <ID>
```

where ID corresponds to the ID of the backup recorded in Step 2.

**Result:**

The following is an example of the output:

```
root@cic-1:~# 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
```

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. Log in to the Atlas VM with SSH.

**Result:**

Logged on to the Atlas VM.

- Download the backup files from Swift, as described in [Atlas Restore](#).

**Result:**

The following is an example of the output:

```
root@atlas:~# cd /var/archives/
root@atlas:/var/archives# swift download AtlasBackups =>
-p atlas_backup1465911268
root@atlas:/var/archives# atlas backup-list
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID          | Name          |          |          | Storage | =>
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1465911268  | AtlasBackup   |          |          | LOCAL   | =>
| 654M        | 2-May-2017    | 08:49:05 |          |         | =>
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+

atlasadm@atlas:~$ sudo atlas backup-restore =>
--d 1465911268 --p <backup_password>
Tue 02 May 2017 11:37:48 INFO: Starting Atlas restore...
Tue 02 May 2017 11:37:48 INFO: Restoring database...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring etc-puppet-hieradata=>
-passwords.yaml...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring home-atlasadm...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring root...
Tue 02 May 2017 11:37:56 INFO: Done.
Tue 02 May 2017 11:37:56 INFO: Verifying users...
Tue 02 May 2017 11:38:29 INFO: Applying configuration...
Tue 02 May 2017 11:38:47 INFO: Restarting services...
Tue 02 May 2017 11:38:50 INFO: Done.
Tue 02 May 2017 11:38:50 INFO: Successfully restored from=>
backup:1465911268
```

8. After the successful restore, verify that the created text file has been restored.

```
cat /home/atlasadm/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	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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@<vfuel_address>
```

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

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

**Result:**

Successfully logged on to the Fuel host.

4. Check 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 aux | 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:He100DLsIkNyQfUe2XS3oaJC' -s http://127.0.0.1:15672/api/exchanges | grep -qw 'mcollective_broadcast'"
checking with command "curl -f -L -u 'mcollective:He100DLsIkNyQfUe2XS3oaJC' -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=qfbWQ5Y1FaJpWFO2NuFFYsf5 /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 aux | 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 aux | grep -q mcollectived"
checking with command "! pgrep puppet"
mcollective is ready.
checking with command "ps aux | 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 s>/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:cl: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:cl: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:cl:2d:20 | compute, virt | | 1 | 1
7 | ready | compute-0-3 | 1 | 192.168.0.22 | ec:f4:bb:cl:30:30 | compute, virt | | 1 | 1
6 | ready | compute-0-1 | 1 | 192.168.0.21 | ec:f4:bb:cl:2f:c0 | compute, virt | | 1 | 1
```

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

**nova service-list**

**Result:**

Nova services are available.

- 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.

**Result:**

Logged onto the rebooted compute host.

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/cache/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	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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@cl1-1:~# nova service-list
```

Id	Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
1	nova-consoleauth	cl1-2.domain.tld	internal	enabled	up	2016-07-21T13:40:34.000000	-
2	nova-scheduler	cl1-2.domain.tld	internal	enabled	up	2016-07-21T13:40:37.000000	-
3	nova-conductor	cl1-2.domain.tld	internal	enabled	up	2016-07-21T13:40:41.000000	-
4	nova-cert	cl1-2.domain.tld	internal	enabled	up	2016-07-21T13:40:37.000000	-
7	nova-consoleauth	cl1-1.domain.tld	internal	enabled	up	2016-07-21T13:40:37.000000	-
10	nova-scheduler	cl1-1.domain.tld	internal	enabled	up	2016-07-21T13:40:40.000000	-
13	nova-conductor	cl1-1.domain.tld	internal	enabled	up	2016-07-21T13:40:42.000000	-
19	nova-cert	cl1-1.domain.tld	internal	enabled	up	2016-07-21T13:40:41.000000	-
22	nova-consoleauth	cl1-3.domain.tld	internal	enabled	up	2016-07-21T13:40:33.000000	-
25	nova-scheduler	cl1-3.domain.tld	internal	enabled	up	2016-07-21T13:40:33.000000	-
28	nova-conductor	cl1-3.domain.tld	internal	enabled	up	2016-07-21T13:40:41.000000	-
37	nova-cert	cl1-3.domain.tld	internal	enabled	up	2016-07-21T13:40:38.000000	-

- 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@cl1-1:~# neutron agent-list
```

id	agent_type	host	alive	admin_state_up	binary
06888113-fadb-4210-9a1f-76e2702b74d1	DHCP agent	cl1-1.domain.tld	True	True	neutron-dhcp-agent
162f905e-2c94-4091-adb5-db92264895d0	Open vSwitch agent	compute-0-4.domain.tld	True	True	neutron-openvswitch-agent
4612a4f5-9cbb-42cf-b229-15f0b853ae14	DHCP agent	cl1-3.domain.tld	True	True	neutron-dhcp-agent
5e4693db-794c-4776-b49d-1c14060b1737	Open vSwitch agent	cl1-3.domain.tld	True	True	neutron-openvswitch-agent
6a7d88b7-fc15-4d8c-ac94-6c345e217a	Open vSwitch agent	compute-0-3.domain.tld	True	True	neutron-openvswitch-agent
85fee15-7de3-467a-b7fc-bc87bf23285	Open vSwitch agent	cl1-1.domain.tld	True	True	neutron-openvswitch-agent
aed9143a-ea6a-4aa6-b3a0-33dde1f9beb0	Open vSwitch agent	compute-0-6.domain.tld	True	True	neutron-openvswitch-agent
c6dd9de-2bd5-43c8-b689-a223d6b88053	Open vSwitch agent	compute-0-2.domain.tld	True	True	neutron-openvswitch-agent
d6e8e197-f3db-4a6d-8ebf-77c13e4e671a	Open vSwitch agent	compute-0-5.domain.tld	True	True	neutron-openvswitch-agent
dac68808-df34-4a39-9ac7-9ae7f52bc97b	Open vSwitch agent	compute-0-1.domain.tld	True	True	neutron-openvswitch-agent
dcd451c6-7f44-466b-be21-d1199072c9e4	DHCP agent	cl1-2.domain.tld	True	True	neutron-dhcp-agent
f07083a3-c31c-4e85-970f-7c30203a4af7	Open vSwitch agent	cl1-2.domain.tld	True	True	neutron-openvswitch-agent

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

- Find the vCIC marked active:

```
crm status | grep active_marker
```

- vCIC confirms that the PM reports are still being produced after the reboot:

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

**Note:** Reports are produced every 15 minutes.

- 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

Redundancy exists if there are three vCICs in the system. In case only two vCICs remain, redundancy is lost because two vCICs are required to maintain quorum.

In this test case the host running the master rabbitmq service 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

<b>Objective</b>	<b>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.</b>
<b>Precondition</b>	



Postcondition	
Duration	
Platform	<ul style="list-style-type: none"><li>• Dell multi-server</li><li>• BSP multi-server</li></ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 exist) 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.

```
neutron net-create <net_name>
neutron subnet-create <net_name> <cidr>
nova boot --flavor <flavor_id> --image <image_id> --nic
net-id=<uuid_of_the_previously_created_net> <vm_name>
```

**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_contrackd [p_contrackd]
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 namespace 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-list
```

```
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

<b>Objective</b>	<b>The switch loses its MASTER state to the redundant switch and gets it back at the end of the reboot.</b>
<b>Precondition</b>	
<b>Postcondition</b>	
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"><li>• Dell multi-server</li><li>• BSP multi-server</li></ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

**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> -user-data /tmp/cloud_config.txt --block-device-mapping 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
```

- c. For CAX, access DMX using the following commands:

```
ssh advanced@<switch_ip_address> -p 2024  
> show ManagedElement=1,DmxcnFunction=1,Eqm=1⇒  
,VirtualEquipment=BSP,Blade=25-3  
> show ManagedElement=1, DmxcFunction=1,Eqm=1⇒  
,VirtualEquipment=BSP,Blade=25-6
```

**Result:**

- Extreme: All switches are ACTIVE.
- CMX/CAX: 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⇒
_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
```



- c. For CAX, access DMX using the following commands:

```
ssh advanced@<switch_ip_address> -p 2024
>show ManagedElement=1,DmxcFunction,Eqm=1⇒
,VirtualEquipment=BSP,Blade=25-3
>show ManagedElement=1,DmxcFunction=1,Eqm=1⇒
,VirtualEquipment=BSP.Blade=26-6
```

**Result:**

- Extreme: Switches are in ACTIVE status.
  - CMX/CAX: 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 / CAX-L:

```
eth2 storage all
dpdk0 traffic all computes
```

Switch-B / CMX-R / CAX-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 Switch Blade in Given BSP Slot

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



## Description

This test case ensures that a CMX, SCX, or CAX blade in any given BSP slot can be locked or unlocked without undue disturbance to the CEE system. The lock of a CMX, SCX, or CAX 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 Switch Blade in Given BSP Slot

Objective	The lock of a CMX, SCX, or CAX 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	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

## Procedure

Do the following:

1. Lock the CMX blade (located in slot 26 or 28 in any subrack):

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment==>
BSP,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 6.

**Result:**

The system is running normally.

3. Unlock the CMX blade.

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=>
BSP,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 6

**Result:**

The system is running normally.

5. Lock the SCX blade (located in slot 0 or 25 in any subrack):

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=>
BSP,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 locked:
  - Run the test case described in Section 2.1 on page 6.

**Result:**

The system is running normally.

7. Unlock the SCX blade.

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=>
BSP,Blade=<shelf_id>-<slot_id>,>
administrativeState=UNLOCKED
commit
```

**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 6



**Result:**

The system is running normally.

9. Lock the CAX blade (located in slot 3 and 6 in subrack 25):

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=>
BSP,Blade=25-<slot_id>,>
administrativeState=LOCKED
commit
```

**Result:**

The blade is locked.

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

- Run the test case described in Section 2.1 on page 6.

**Result:**

The system is running normally.

11. Unlock the CAX blade.

```
configure
ManagedElement=1,DmxcFunction=1,Eqm=1,VirtualEquipment=>
BSP,Blade=25-<slot_id>,>
administrativeState=UNLOCKED
commit
```

**Result:**

The blade is unlocked.

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

- Run the test case described in Section 2.1 on page 6.

**Result:**

The system is running normally.

13. Repeat the above steps for the switch blades in the 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

<b>Objective</b>	<b>Check that both the compute host and Atlas VM recovers and that the Atlas GUI is fully functional.</b>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>Atlas VM is up and running.</li> </ul>
<b>Postcondition</b>	<ul style="list-style-type: none"> <li>Atlas VM is up and running.</li> </ul>
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>Dell multi-server</li> <li>BSP multi-server</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

## 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 slot (for BSP) or server (for DELL) the compute host resides in.

The node name is in the following format:



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

```
ssh atlasadm@<atlas_ip>
pwd=<password>
sudo -i
```





```
systemctl status <service_name>
```

where service\_name corresponds to the following values:

- 'rsyslog'
- 'rabbitmq-server'
- 'ovft-engine'
- 'ovft-api'
- 'mysql'
- 'memcached'
- 'heat-engine'
- 'heat-api-clouwatch'
- 'heat-api-cfn'
- 'heat-api'
- 'auditd'
- 'apache2'

**Result:**

All listed services are running.

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	<ul style="list-style-type: none"><li>• Dell multi-server</li><li>• BSP multi-server</li></ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 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 -l -r` or `crm status`.

**Result:**



Cluster status is displayed.

5. Ensure that 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.

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.

Table 15 Reboot Whole vCIC Cluster

Objective	After the reboot, vCICs function normally with no negative impact on the system.
Precondition	
Postcondition	
Duration	
Platform	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 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 and check the recovery of the control functions. Use commands `nova service-list`, `cinder service-list`, `nova list --all-tenant`, and `crm_mon -lf` from one of the controllers.

**Result:**

There are no failed services.

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

**Result:**

Traffic is undisturbed.

6. Check the connectivity for the tenant VMs.

**Result:**

Tenant VMs are reachable.

7. Check the possible impacts on application VMs. Ping the 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	<ul style="list-style-type: none"> <li>Atlas VM is up and running.</li> </ul>
Postcondition	<ul style="list-style-type: none"> <li>Atlas VM is up and running.</li> </ul>
Duration	
Platform	<ul style="list-style-type: none"> <li>Dell multi-server</li> <li>BSP multi-server</li> <li>Single server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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 execute the following command:  
**`sudo systemctl reboot`**
  - 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. Check that all the services are running.

```
ssh atlasadm@<atlas_ip>
pwd = <password>
sudo -i
systemctl status <service_name>
```

where service\_name corresponds to the following values:

- 'rsyslog'
- 'rabbitmq-server'
- 'ovft-engine'
- 'ovft-api'
- 'mysql'
- 'memcached'
- 'heat-engine'
- 'heat-api-clouwatch'
- 'heat-api-cfn'
- 'heat-api'
- 'auditd'
- 'apache2'

**Result:**

All listed services are running.

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

<b>Objective</b>	<b>Do a soft reboot of the started VM. Log in to the VM and verify that the test application is stopped.</b>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>• OpenStack is up and running.</li> <li>• Traffic is up and running.</li> </ul>
<b>Postcondition</b>	<ul style="list-style-type: none"> <li>• OpenStack is up and running.</li> <li>• Traffic is up and running.</li> </ul>
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> <li>• Single server</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

### 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 <username>@<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. Log on 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

<b>Objective</b>	<b>VM is rebooted. Test application is stopped.</b>
<b>Precondition</b>	All expected OpenStack services are running on vCIC and in the host where the vCIC is active (check with command <code>service --status-all</code> ).
<b>Postcondition</b>	All expected OpenStack services are running on vCIC and in the host where the vCIC is active (check with command <code>service --status-all</code> ).
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> <li>• Single server</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

### 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 <username>@<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. Log in 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	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server, single servr</li> <li>• Single server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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> --user-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

<b>Objective</b>	<b>Do a successful Atlas backup and restore.</b>
<b>Precondition</b>	<ul style="list-style-type: none"><li>• Atlas GUI is fully functional.</li></ul>

Postcondition	<ul style="list-style-type: none"> <li>Atlas GUI is fully functional.</li> </ul>
Duration	
Platform	<ul style="list-style-type: none"> <li>Dell multi-server</li> <li>BSP multi-server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

## Procedure

Do the following:

1. Log in 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:

```
atlasadm@atlas:~$ sudo atlas backup-create =>
```

```
--name AtlasBackup --p password
```

```
Tue 02 May 2017 08:48:23 INFO: Creating Backup "AtlasBackup" =>
with ID=1493707703...
```

```
Tue 02 May 2017 08:48:23 INFO: Backing up password =>
configuration, home directories and database...
```

Tue 02 May 2017 08:48:58 INFO: Done.

```
Tue 02 May 2017 08:49:05 INFO: Encrypting...
```

Tue 02 May 2017 08:49:05 INFO: Done.

```
Tue 02 May 2017 08:49:05 INFO: Uploading Backup "AtlasBackup" =>
of size 654M to Swift...
```

Tue 02 May 2017 08:49:15 INFO: Done.

Tue 02 May 2017 08:49:15 INFO: Backup Complete!

```
atlasadm@atlas:~$ sudo atlas backup-list
```

$$+ - - - - - + - - - - - + - - - - - + \Rightarrow$$

-----+-----+-----+

ID	Name	Storage	⇒
----	------	---------	---



Size	Date	Time	
1493707703	AtlasBackup	LOCAL	=>
654M	2-May-2017	08:49:05	

**Note:** Record the ID of the backup from the printout as it is required in a later step of the procedure.

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

```
swift list AtlasBackups | grep <ID>
```

where ID corresponds to the ID of the backup recorded in Step 2.

**Result:**

An example of the printout is:

```
root@cic-1:~# 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
```

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:

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



```

root@atlas:/var/archives# swift download AtlasBackups =>
-p atlas_backup1465911268
root@atlas:/var/archives# atlas backup-list
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID          | Name          | Storage |
| Size       | Date          | Time    |
+-----+-----+-----+-----+-----+-----+
| 1465911268  | AtlasBackup   | LOCAL   |
| 654M       | 2-May-2017    | 08:49:05 |
+-----+-----+-----+-----+-----+-----+

atlasadm@atlas:~$ sudo atlas backup-restore =>
--d 1465911268 --p <backup_password>
Tue 02 May 2017 11:37:48 INFO: Starting Atlas restore...
Tue 02 May 2017 11:37:48 INFO: Restoring database...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring etc-puppet-hieradata=>
-passwords.yaml...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring home-atlasadm...
Tue 02 May 2017 11:37:49 INFO: Done.
Tue 02 May 2017 11:37:49 INFO: Restoring root...
Tue 02 May 2017 11:37:56 INFO: Done.
Tue 02 May 2017 11:37:56 INFO: Verifying users...
Tue 02 May 2017 11:38:29 INFO: Applying configuration...
Tue 02 May 2017 11:38:47 INFO: Restarting services...
Tue 02 May 2017 11:38:50 INFO: Done.
Tue 02 May 2017 11:38:50 INFO: Successfully restored from=>
backup:1465911268

```

8. After the successful restore, verify that the created text file has been restored.

```
cat /home/atlasadm/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 the `nova migrate` command for migration.

Table 21 Migration of VMs with `nova migrate` Command

Objective	Migration of VMs with <code>nova migrate</code> command
Precondition	<ul style="list-style-type: none"><li>• All expected OpenStack services are up and running.</li></ul>
Postcondition	<ul style="list-style-type: none"><li>• All expected OpenStack services are up and running.</li></ul>
Duration	
Platform	<ul style="list-style-type: none"><li>• Dell multi-server</li><li>• BSP multi-server</li></ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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. Create server groups for affinity and anti-affinity.
  - a. Create server groups by executing the following commands:
 

```
nova server-group-create server_group_1 affinity
nova server-group-create server_group_2 anti-affinity
nova server-group-create server_group_3 affinity
```
  - b. List the UUIDs of the created server groups for later use by executing the following command:
 

```
nova server-group-list
```

**Result:**

The server groups for affinity/anti-affinity are created.

6. Boot first VM (VM1) from VM\_Volume\_1 on the host selected above, with the server groups policy (hint: group).

Use the following commands:

```
nova boot --hint group=<server_group_1_uuid> --availability-zone nova:<host_id> =>
--block-device-mapping=>
  <vda>=<vm_volume_id>:<type>:<size_in_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.

7. Boot VM2 from VM\_Volume\_2 with the server groups policy (hint: group).



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

Use the command:

```
nova boot --hint group=<server_group_1_uuid>⇒  
--block-device-mapping⇒  
  <vda>=<vm_volume_2_id>:<type>:<size_in_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.

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

**Result:**

The VMs are running on the same host.

9. Boot VM3 from VM\_Volume\_3 with the server groups policy. Use the following commands:

```
nova boot --hint group=<server_group_2_uuid>⇒  
--block-device-mapping ⇒  
<vda>=<vm_volume_3_id>:<type>:<size_in_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.

10. Boot VM4 from VM\_Volume\_4 with the server groups policy. Use the following commands:

```
nova boot --hint group=<server_group_2_uuid>⇒  
--block-device-mapping ⇒  
<vda>=<vm_volume_4_id>:<type>:<size_in_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.

11. Boot VM5 from VM\_Volume\_5 on the host selected above with the server groups policy and the High Availability (HA) policy managed-on-host. Use the following commands:

```
nova boot --hint group=<server_group_3_uuid> --availability-zone nova:<host-id> ⇒
--block-device-mapping ⇒
<vda>=<vm_volume_5_id>:<type>:<size_in_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.

12. 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 group=<server_group_3_uuid> ⇒
--block-device-mapping ⇒
<vda>=<vm_volume_6_id>:<type>:<size_in_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.

13. 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_in_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.

14. 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.

15. 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.

16. Migrate VM1 using the `nova forcemove` command, ignoring server group hints.

- a. Migrate VM1 using the following command:

```
nova forcemove VM1 --ignore-hints
```

- b. Check the result using the following command:

```
nova show VM1
```

- c. Confirm the migration:

```
nova resize-confirm VM1
```

**Result:**

VM1 is migrated to another host.

17. 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.

18. 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.

19. 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.

20. Migrate VM5 using the nova forcemove command, ignoring server group hints.

- a. Migrate VM1 using the following command:

```
nova forcemove VM5 --ignore-hints
```

- b. Check the result using the following command:



```
nova show VM5
```

- c. Confirm the migration:

```
nova resize-confirm VM5
```

**Result:**

VM5 cannot be migrated as HA policy is managed-on-host.

21. 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 cannot be migrated as HA policy is managed-on-host.

22. 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.

23. Migrate VM7.

- a. Attempt migration of VM7 using the command nova forcemove:

```
nova forcemove VM7
```

- b. Check the result using the following command:



```
nova show VM7
```

**Result:**

VM7 cannot be moved as HA policy is not set.

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

**Result:**

The list of migrations is printed.

25. 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.

26. 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	<ul style="list-style-type: none"><li>All expected OpenStack services are up and running.</li></ul>
Postcondition	<ul style="list-style-type: none"><li>All expected OpenStack services are up and running.</li></ul>
Duration	
Platform	<ul style="list-style-type: none"><li>Dell multi-server</li><li>BSP multi-server</li></ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> 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\_name\_x> <volume\_size>**

Issue the following command for verification:

**cinder list**

**Result:**

Cinder volumes are added, status of Cinder volumes is AVAILABLE and <volume\_id\_x> 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. Create server group for affinity.





- a. Create server group by executing the following command:  
`nova server-group-create server_group_1 affinity`
- b. List the UUID of the created server group for later use by executing the following command:  
`nova server-group-list`

**Result:**

The server group is created.

4. 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 --hint group=<server_group_1_uuid> =>  
--flavor <flavor_name> --nic net-id=<net_id> \ =>  
--block-device-mapping <vda>=<volume_id_1>:=>  
<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.

5. 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_id_2>:=>  
<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.

6. 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_id_3>:=>  
<type>:<size_in_GB>:<delete-on-terminate> \  
--hint group=<server_group_1_uuid> --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.

7. 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_id_4>:⇒  
<type>:<size_in_GB>:<delete-on-terminate> ⇒  
--user-data /tmp/cloud_config.txt VM4  
nova show VM4
```

**Result:**

VM4 is created and is ACTIVE.

8. 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.

9. 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.

10. Restart VM1:

```
nova start VM1
```

Do the following for verification:

```
nova list
```

**Result:**

VM1 is ACTIVE again.

11. 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.

12. 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`.

13. 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.

14. 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.

15. Delete VM1 and verify the action:

```
nova delete VM1  
nova list
```

**Result:**

VM1 is deleted.

16. 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, the policy is unmanaged, so 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.

17. Log on to 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.

18. Delete the VMs and verify the result:

```
nova delete VM<x>  
nova list
```

**Result:**

The VMs are deleted.

19. Delete the Cinder volume and verify the result:

```
cinder delete <volume_id>
cinder list
```

**Result:**

The volume is deleted.

## 2.22 Verify That Migrated vFuel VMs Are Synchronized after CEE Deployment

This section is only applicable for multi-server deployments.

**Description**

Verify that vFuel VMs are migrated to the specified nodes in CEE during CEE deployment.

Table 23 Verify That Migrated vFuel VMs Are in Sync after CEE Deployment

<b>Objective</b>	<b>Verify that vFuel VMs are migrated to the specified nodes in CEE during CEE deployment.</b>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>Conditions described in section Prerequisites in Fuel Synchronization are fulfilled.</li> </ul>
<b>Postcondition</b>	
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>Dell multi-server</li> <li>BSP multi-server</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

**Procedure**

Do the following:

1. Collect Fuel VM placement information and store it for later use:
  - a. Log on to vFuel and use the below command:



#### **get\_vfuel\_info**

- b. Check the config.yaml.

**Result:**

Fuel VM placement information is stored.

2. Check vFuel state as described in section Conditions in Fuel Synchronization and store the outputs.

**Result:**

vFuel state information is stored.

3. Shut down the active Fuel VM by following the instructions in section Shut Down the Fuel VM in Fuel Synchronization.

**Result:**

The active Fuel VM is in shut off state on the compute where it is hosted.

4. Log on to the passive compute and start up the passive Fuel VM.

**Result:**

Passive Fuel VM is started.

5. Log on to the freshly started passive Fuel VM.

**Result:**

Log on to the passive Fuel VM is successful.

6. Rerun the checks described in Step 2 on the passive Fuel VM and store the output.

**Result:**

The state information output is stored.

7. Compare the stored outputs. The following conditions must be fulfilled:

- The environment (id, name) is the same.
- The nodes (number, names, roles) are the same.
- The same tasks are completed.

**Result:**

All compared items are matched.

## 2.23 Verify That Manual vFuel Handover Is Successful

This section is only applicable for multi-server deployments.



### Description

Verify that vFuel can be manually handed over to the synchronized passive Fuel VM and CEE is usable after this operation.

Table 24 Verify That Manual vFuel Handover Is Successful

<b>Objective</b>	<b>Verify that vFuel can be manually handed over to the synchronized passive Fuel VM and CEE is usable after this operation.</b>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>Conditions described in section Prerequisites in Fuel Synchronization are fulfilled.</li> </ul>
<b>Postcondition</b>	
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>Dell multi-server</li> <li>BSP multi-server</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

### Procedure

Do the following:

1. Collect Fuel VM placement information and store it for later use:

- a. Log on to vFuel and use the below command:

```
get_vfuel_info
```

- b. Check the config.yaml.

#### Result:

Fuel VM placement information is stored.

2. Check vFuel state as described in section Conditions in Fuel Synchronization and store the outputs.

#### Result:

vFuel state information is stored.



3. Execute the steps of section Failover to the Cold Standby Fuel VM (Recover from Failure) in [Fuel Synchronization](#).

**Result:**

Steps are executed, the passive Fuel VM is started.

4. Check whether the running passive Fuel VM can reach the CICs and compute through the network:

- Fuel is reachable with SSH.
- Fuel can SSH to CICs and compute.
- In the vFuel node command output, the nodes show up as online.

**Result:**

All checks are successful.

5. Check vFuel application sanity as described in section Conditions in [Fuel Synchronization](#).

**Result:**

Checks are successful.

6. Check the connectivity to the Swift RESTful API in the CEE region.

**Result:**

Connectivity is working.

## 2.24 Verify vFuel Image Synchronization Instructions

This section is only applicable for multi-server deployments.

### Description

Verify that after executing the vFuel image synchronization based on the instructions of [Fuel Synchronization](#), the active and passive Fuel VMs are identical.

Table 25 Verify vFuel Image Synchronization Instructions

<b>Objective</b>	<b>Verify that after executing the vFuel image synchronization the active and passive Fuel VMs are identical.</b>
<b>Precondition</b>	<ul style="list-style-type: none"><li>• Conditions described in section Prerequisites in <a href="#">Fuel Synchronization</a> are fulfilled.</li></ul>





Postcondition	
Duration	
Platform	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• BSP multi-server</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

### Procedure

Do the following:

1. Collect Fuel VM placement information and store it for later use:

- a. Log on to vFuel and use the below command:

```
get_vfuel_info
```

- b. Check the config.yaml.

#### Result:

Fuel VM placement information is stored.

2. Check vFuel state as described in section Conditions in Fuel Synchronization and store the outputs.

#### Result:

vFuel state information is stored.

3. Execute the steps of section Synchronize the Active Fuel VM to the Cold Standby (Backup and Restore) in Fuel Synchronization.

#### Result:

Steps are executed successfully, results are as described in Fuel Synchronization.

4. Execute the checks described in Conditions in Fuel Synchronization. Compare the results with the one stored in Step 2.

#### Result:

Checks before and after the synchronization have the same result.



## 2.25 Verify Rollback to Previous (Passive) vFuel Version

This section is only applicable for multi-server deployments.

### Description

Verify that it is possible to activate the previous (passive) version of vFuel based on the instructions of [Fuel Synchronization](#), for example after a CEE update.

Table 26 Verify Rollback to Previous (Passive) vFuel Version

Objective	Verify that it is possible to activate the previous (passive) version of vFuel.
Precondition	<ul style="list-style-type: none"><li>Conditions described in section Prerequisites in <a href="#">Fuel Synchronization</a> are fulfilled.</li></ul>
Postcondition	
Duration	
Platform	<ul style="list-style-type: none"><li>Dell multi-server</li><li>BSP multi-server</li></ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

### Procedure

Do the following:

1. Collect Fuel VM placement information and store it for later use:
  - a. Log on to vFuel and use the below command:  

```
get_vfuel_info
```
  - b. Check the `config.yaml`.

#### Result:

Fuel VM placement information is stored.

2. Check vFuel state as described in section [Conditions in Fuel Synchronization](#) and store the outputs.

**Result:**

vFuel state information is stored.

3. Modify the configuration on the active Fuel VM: for example add or remove a node, create or delete a file.

**Result:**

Modifications are done.

4. Execute the steps of section Failover to the Cold Standby Fuel VM (Recover from Failure) in [Fuel Synchronization](#).

**Result:**

Steps are executed. The passive Fuel VM starts.

5. Verify that the modification applied in Step 3 cannot be seen on the currently running Fuel.

**Result:**

None of the modifications done in Step 3 are visible.

## 2.26 End-to-End Connectivity with CSS

This section is applicable for single server and specific multi-server deployments.

**Description**

Use this test to check the End-to-End (E2E) connectivity of the VMs with Cloud SDN Switch (CSS). Verify that VMs can communicate with each other when Maximum Transmission Unit (MTU) is set to 9000 bytes.

Table 27 E2E Connectivity with CSS

Objective	Verify Jumbo Frame Support on the Tenant Data Network.
Precondition	All expected OpenStack services are up and running.  In case on Single Server, Fuel must be enabled.
Postcondition	All expected OpenStack services are up and running.
Duration	
Platform	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• Dell single server</li> <li>• HDS (if extra interfaces are available)</li> <li>• BSP</li> </ul>



Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

### Procedure

Do the following:

1. Log on to vFuel:

```
ssh <vfuel_address>
```

**Result:**

Logged on to vFuel.

2. Log on to a vCIC:

```
ssh <vcic_name>
```

**Result:**

Logged on to vCIC.

3. Create Glance image:

```
glance image-create --name <image_name> --progress⇒  
--visibility public --disk-format <disk_format>⇒  
--container-format bare < <image_file_path>
```

**Result:**

Glance image is created.

4. Create Neutron network:

```
neutron net-create <network_name>  
neutron subnet-create <network_id> <subnet_range>
```

**Result:**

All required networks and subnets are created.

5. Create flavor for VMs:

```
nova flavor-create <flavor_name> <flavor_id>⇒  
<flavor_ram> <flavor_disk> <flavor_vcpus>  
nova flavor-key <flavor_name>⇒  
set hw:mem_page_size=<page_size_value>  
nova flavor-key <flavor_name>⇒  
set hw:cpu_policy=dedicated
```

**Result:**

Flavor for <flavor\_id> is created.

6. Register SSH key using the below commands:  
`ssh-keygen -t rsa -N "" -f <ssh_key>`  
`nova keypair-add --pub-key <ssh_key>.pub <key_name>`

**Result:**

SSH key is registered.

7. In case of a system using unmanaged switches, configure the switches.

If using unmanaged Extreme switches, do the following:

- a. Identify the ID of the VLAN:

```
neutron net-show <network_id> | grep "segmentation_id |
awk '{print $4}'
```

- b. Connect to the switch:

```
ssh admin@<switch_ip_address>
```

- c. Create the VLAN on the switch:

```
create vlan <vlan_name>
```

- d. Configure the VLAN on the switch:

```
configure vlan <vlan_name> tag <vlan_id>
```

```
configure vlan <vlan_name> add ports <ports> tagged
```

8. Create VMs using Neutron virtual networks and wait for them to boot up:

```
nova boot --image <image_id> --flavor <flavor_id>⇒
--nic net-id=<ovs_network_id> --key-name <key_name>⇒
--config-drive true --availability-zone⇒
nova:<host>.<domain_name> <vm_name>
```

**Result:**

VM for <vm\_id> created and booted

9. Configure interfaces in the VMs:

```
ip netns exec qdhcp- <ovs_network_id>⇒
ssh -t -o StrictHostKeyChecking=no -I<ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "sudo ifconfig⇒
eth0 mtu 9000
ip netns exec qdhcp- <ovs_network_id>⇒
ssh -t -o StrictHostKeyChecking=no -i⇒
<ssh_key> ubuntu@<vm_dhcp_ip_address> "exit"
```



**Result:**

Interface eth0 in <vm\_dhcp\_ip\_address> is up and configured to Jumbo Frame MTU: 9000.

10. Verify communication with given packet size:

```
ip netns exec qdhcp-<ovs_network_id> ssh -t -o⇒  
StrictHostKeyChecking=no -i <ssh_key>⇒  
ubuntu<vm_dhcp_ip_address> "sudo ping -c<ping_repetitions>⇒  
-M do -s 2140 <destination_ip_address>"
```

**Result:**

Ping test to <destination\_ip\_address> is successful without any packet loss.

11. Delete VMs:

```
nova delete <vm_name>
```

**Result:**

VM is deleted.

12. Delete Nova keypair:

```
nova keypair-delete <key_name>
```

**Result:**

Nova keypair is deleted.

13. In case of a system using unmanaged switches, unconfigure the switches.

If using unmanaged Extreme switches, do the following:

**Note:** All switches must be unconfigured.

- a. Connect to the switch:

```
ssh admin@<switch_ip_address>
```

- b. Delete the VLAN on the switch:

```
delete vlan <vlan_name>
```

**Result:**

VLAN deleted on all switches.

14. Delete Neutron ports:

```
neutron port-delete <port_name>
```

**Result:**

Neutron ports are deleted.



15. Delete Nova flavor:

```
nova flavor-delete <flavor_id>
```

**Result:**

Nova flavor is deleted.

16. Delete Neutron networks:

```
neutron net-delete <network_name>
```

**Result:**

Neutron network <network\_name> is deleted.

17. Delete Glance image:

```
glance image-delete <image_name>
```

**Result:**

Glance image is deleted.

## 2.27 End-to-End Connectivity with SR-IOV

This section is applicable for certain multi-server deployments.

### Description

Use this test to check the End-to-End (E2E) connectivity of the VMs with Single Root I/O Virtualization (SR-IOV). Verify that SR-IOV can communicate with each other when MTU is set to 2140 bytes.

Table 28 E2E Connectivity with SR-IOV

<b>Objective</b>	<b>Verify Jumbo Frame Support on SR-IOV Network.</b>
<b>Precondition</b>	SR-IOV feature must be configured in the configuration file before deployment.
<b>Postcondition</b>	All expected OpenStack services are up and running.
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• HDS (if extra interfaces are available)</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	



Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	

### Procedure

Do the following:

1. Log on to vFuel:

```
ssh <vFuel_address>
```

**Result:**

Logged on to vFuel.

2. Log on to a vCIC:

```
ssh <vCIC_name>
```

**Result:**

Logged on to vCIC.

3. Create Glance image:

```
glance image-create --name <image_name> --progress⇒  
--visibility public --disk-format <disk_format>⇒  
--container-format bare < <image_file_path>
```

**Result:**

Glance image is created.

4. Update network quotas:

```
neutron quota-update --network 20⇒  
--subnet 20 --port 100
```

**Result:**

Network quotas are updated.

5. Create Neutron network for DHCP and for SR-IOV interfaces:

- a. Create DHCP network:

```
neutron net-create <dhcp_network_name>
```

- b. Do one of the following:

Create flat network:

```
neutron net-create<network_name>⇒  
--provider:physical_network <physical_network>⇒  
--provider:network_type flat
```





```
neutron subnet-create <sriov_dhcp_network_id>⇒
<dhcp_subnet_range>
```

or

Create VLAN network:

```
neutron net-create<network_name> --provider:
physical_network ⇒
<physical_network> --provider:network_type vlan⇒
--provider:segmentation_id <vlan_id>
```

**Result:**

All required networks and subnets are created.

6. Create Neutron ports for SR-IOV network:

```
neutron port-create <network_id> --name⇒
<port_name> --vnic-type direct
```

**Result:**

Neutron port created

7. In case of a system using unmanaged switches, configure the SR-IOV network on the switches.

If using unmanaged Extreme switches, do the following:

**Note:** Use the ports that are used for the SR-IOV VLANs.

All switches must be configured.

- a. Identify the ID of the VLAN:

```
neutron net-show <network_id> | grep⇒
"segmentation_id" | awk '{print $4}'
```

- b. Connect to the switch:

```
ssh admin@<switch_ip_address>
```

- c. Create VLAN on the switch:

```
create vlan <vlan_name>
```

- d. Configure the VLAN on the switch:

```
configure vlan <vlan_name> tag <vlan_id>
```

```
configure vlan <vlan_name>⇒
add ports <sriov_traffic_ports> tagged
```

**Result:**



Extreme Switches are configured.

8. Create Nova flavor for VMs:

```
nova flavor-create <flavor_name> <flavor_id>⇒  
  <flavor_ram> <flavor_disk> <flavor_vcpus>
```

```
nova flavor-key <flavor_name>⇒  
  set hw:mem_page_size=<page_size_value>
```

```
nova flavor-key <flavor_name>⇒  
  set hw:cpu_policy=dedicated
```

**Result:**

Flavor for <flavor\_id> is created.

9. Register SSH key:

```
ssh-keygen -t rsa -N "" -f <ssh_key>
```

```
nova keypair-add --pub-key <ssh_key>.pub <key_name>
```

**Result:**

SSH key is registered.

10. Create VMs on SR-IOV interfaces and wait for them to boot up:

```
nova boot --image <image_id>⇒  
  --flavor <flavor_id> --key-name <key_name>⇒  
  --config-drive true --nic net-id=<dhcp_network_id>⇒  
  --nic port-id=<port_id> --availability-zone nova:⇒  
  <host_name>.domain.tld<vm_name>
```

**Result:**

VMs on SR-IOV are created and booted.

11. Configure the interfaces in the VMs:

For flat network configuration:

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒  
  StrictHostKeyChecking=no -i <ssh_key>⇒  
  ubuntu@<vm_dhcp_ip_address> "sudo ifconfig ⇒  
  <sriov_interface> up"
```

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒  
  StrictHostKeyChecking=no -i <ssh_key>⇒  
  ubuntu@<vm_dhcp_ip_address>⇒  
  "sudo ip link add link <sriov_interface>⇒  
  <sriov_interface>.<vlan_id> type vlan⇒  
  proto 802.1Q id <vlan_id>"
```

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
```



```

StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>⇒
"sudo ifconfig <sriov_interface>.<vlan_id>⇒
<ip_sriov>/<ip_mask> up"

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "sudo ifconfig⇒
<sriov_interface> mtu 9000"

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "sudo ifconfig⇒
<sriov_interface>.<vlan_id> mtu 9000"

```

For VLAN network configuration:

```

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>⇒
"sudo ifconfig <sriov_interface>.<vlan_id>⇒
<ip_sriov>/<ip_mask> up"

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "sudo ifconfig⇒
<sriov_interface> mtu 9000"

```

#### Result:

VM interfaces are configured.

## 12. Positive verification with given packet size:

For VLAN network configuration:

```

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>⇒
"ping -I<sriov_interface>⇒
-c<ping_repetitions> -M do -s ⇒
2140 <destination_ip_address>"

```

For flat network verification:

```

ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>⇒
"ping -I<sriov_interface>.<vlan_id>⇒
-c<ping_repetitions> -M do -s ⇒
2140 <destination_ip_address>"

```

#### Result:



Ping test to <destination\_ip\_address> is successful without any packet loss.

13. Delete VM:

```
nova delete <vm_name>
```

**Result:**  
VM is deleted.

14. Delete Nova keypair:

```
nova keypair-delete <key_name>
```

**Result:**  
Nova keypair is deleted.

15. Delete Neutron ports:

```
neutron port-delete <port_name>
```

**Result:**  
Neutron ports are deleted.

16. Delete Nova flavor:

```
nova flavor-delete <flavor_id>
```

**Result:**  
Nova flavor is deleted.

17. Delete Neutron networks:

```
neutron net-delete <network_name>
```

**Result:**  
Neutron network for <network\_name> is deleted.

18. Delete Glance image:

```
glance image-delete <image_id>
```

**Result:**  
Glance image is deleted.

19. In case of a system using unmanaged switches, unconfigure the SR-IOV network on the switches.

If using unmanaged Extreme switches, do the following:

**Note:** All switches must be unconfigured.

- a. Connect to the switch:



```
ssh admin@<switch_ip_address>
```

- b. Delete VLAN on the switch:

```
delete vlan <vlan_name>
```

**Result:**

VLAN is deleted on all switches.

20. Reset network quotas to default:

```
neutron quota-delete
```

**Result:**

Network quotas are reset.

## 2.28 End-to-End Connectivity with PCI PT

This section is applicable for certain multi-server deployments.

### Description

Use this test to check the End-to-End (E2E) connectivity of the VMs with Peripheral Component Interconnect passthrough (PCI PT). Verify that PCI PT VMs can communicate with each other when MTU is set to 2140 bytes.

Table 29 E2E Connectivity with PCI PT

Objective	Verify E2E connectivity with PCI PT.
Precondition	PCI passthrough feature must be configured in the configuration file before deployment.
Postcondition	All expected OpenStack services are up and running.
Duration	
Platform	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• HDS (if extra interfaces are available)</li> </ul>
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	



## Procedure

Do the following:

1. Log on to vFuel:

```
ssh <vfuel_address>
```

**Result:**

Logged on to vFuel.

2. Log on to a vCIC:

```
ssh <vcic_name>
```

**Result:**

Logged on to vCIC.

3. Create Glance image:

```
glance image-create --name <image_name> --progress⇒  
--visibility public --disk-format <disk_format>⇒  
--container-format bare < <image_file_path>
```

**Result:**

Glance image is created.

4. In case of a system using unmanaged switches, configure the PCI passthrough network on the switches.

If using unmanaged Extreme switches, do the following:

**Note:** All switches must be configured.

- a. Connect to the switch:

```
ssh admin@<switch_ip_address>
```

- b. Create the VLAN on the switch:

```
create vlan <vlan_name>
```

- c. Configure the VLAN on the switch:

```
configure vlan <vlan_name> tag <vlan_tag>
```

```
configure vlan <vlan_name>⇒  
add ports <sriov_traffic_ports> tagged
```

**Result:**

Extreme Switches are configured.

5. Create Nova flavor for VMs:



```
nova flavor-create <flavor_name> <flavor_id>⇒
<flavor_ram> <flavor_disk> <flavor_vcpus>
```

```
nova flavor-key <flavor_name>⇒
set hw:mem_page_size=<page_size_value>
```

```
nova flavor-key <flavor_name>⇒
set hw:cpu_policy=dedicated
```

```
nova flavor-key <flavor_name>⇒
set pci_passthrough:alias=<pci_alias>:⇒
<number_of_pci_devices_to_be_assigned>
```

**Result:**

Flavor for <flavor\_name> is created.

6. Register SSH key:

```
ssh-keygen -t rsa -N "" -f <ssh_key>
```

```
nova keypair-add --pub-key <ssh_key>.pub <key_name>
```

**Result:**

SSH key is registered.

7. Create Neutron network for DHCP:

```
neutron net-create --provider:network_type vlan⇒
--provider:physical_network <physical_network>⇒
--provider:segmentation_id <dhcp_vlan_id> ⇒
<dhcp_vlan_id>
```

```
neutron subnet-create <dhcp_network_id> ⇒
<dhcp_subnet_range>
```

**Result:**

All required networks and subnets are created.

8. Create VMs on PCI PT interfaces and wait for the boot up:

```
nova boot --image <image_name> --flavor <flavor_id>⇒
--key-name <key_name> --config-drive true⇒
--nic net-id=<dhcp_network_id> --availability-zone nova:⇒
<host>.domain.tld <vm_name>_<host>
```

**Result:**

VMs are created and booted.

9. Check the connectivity of the VM:

```
ip netns exec qdhcp-<pcipt_dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "exit"
```

**Result:**

Connectivity test to <vm\_dhcp\_ip\_address> is True.

## 10. Set up PCI PT interfaces on VMs:

```
ip netns exec qdhcp-<pcipt_dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>"sudo modprobe ixgbe"
```

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>"sudo ifconfig ⇒
<pcipt_interface> up"
```

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address> "sudo ip link add link⇒
<pcipt_interface> <pcipt_interface>.<pcipt_vlan_id>⇒
type vlan proto 802.1Q id <pcipt_vlan_id>"
```

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>"sudo ifconfig⇒
<pcipt_interface>.<pcipt_vlan_id>⇒
<pcipt_ip>/<ip_mask> up"
```

**Result:**

PCI PT interface (<pcipt\_interface>.<pcipt\_vlan\_id> with IP <pcipt\_ip> on <vm\_dhcp\_ip\_address> is up and configured.

## 11. Positive verification with given packet size:

```
ip netns exec qdhcp-<dhcp_network_id> ssh -t -o⇒
StrictHostKeyChecking=no -i <ssh_key>⇒
ubuntu@<vm_dhcp_ip_address>⇒
"ping -c<ping_repetitions> -M do -s⇒
2140 <destination_ip_address>"
```

**Result:**

Ping test to <destination\_ip\_address> is successful without any packet loss.

## 12. Delete VM:

```
nova delete <vm_name>
```

**Result:**

VM is deleted.

## 13. Delete Nova keypair:

```
nova keypair-delete <key_name>
```



**Result:**

Nova keypair is deleted.

## 14. Delete Nova flavor:

```
nova flavor-delete <flavor_id>
```

**Result:**

Nova flavor is deleted.

## 15. Delete Neutron networks:

```
neutron net-delete <network_name>
```

**Result:**

Neutron network for <network\_name> is deleted.

## 16. Delete Glance image:

```
glance image-delete <image_id>
```

**Result:**

Glance image is deleted.

## 17. In case of a system using unmanaged switches, unconfigure the PCI passthrough network on the switches.

If using unmanaged Extreme switches, do the following:

**Note:** All switches must be unconfigured.

- a. Connect to the switch:

```
ssh admin@<switch_ip_address>
```

- b. Delete the VLAN on the switch:

```
delete vlan <vlan_name>
```

**Result:**

VLAN is deleted on all switches.

## 2.29 Verify IPv6 Address as Allowed Address Pair

This section is applicable for HDS deployment.

**Description**

Use this test to verify that VMs can communicate with each other using the allowed address pair IP and manually assigned IPv6 address.



Table 30 Verifying IPv6 Address as Allowed Address Pair

<b>Objective</b>	<b>Verify IPv6 IP address as allowed address pair.</b>
<b>Precondition</b>	All expected OpenStack and SDN services are up and running.  SDN health check is done.
<b>Postcondition</b>	All expected OpenStack and SDN services are up and running.  SDN health check is done.
<b>Duration</b>	
<b>Platform</b>	HDS
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

### Procedure

Do the following:

1. Log on to vFuel:

```
ssh <vfuel_address>
```

**Result:**

Logged on to vFuel.

2. Log on to a vCIC:

```
ssh <vcic_name>
```

**Result:**

Logged on to vCIC.

3. Create Neutron network for floating IP:

- a. Create external network:

```
neutron net-create <external_network_name> =>  
--router:external=True --provider:network_type gre  
  
neutron subnet-create <external_network_name> =>  
--name <subnet_name> --disable-dhcp =>
```



```
<public_range_subnet_cidr>
```

```
neutron net-external-list
```

- b. Create router and set gateway:

```
neutron router-create <router_name>
```

```
neutron router-gateway-set <router_id> ⇒  
<external_network_id>
```

- c. Create L3VPN and associate it to external network:

```
neutron bgpvpn-create --export-targets ⇒  
<EXPORT-TARGETS> --import-targets <IMPORT-TARGETS> ⇒  
--route-distinguishers <ROUTE-DISTINGUISHERS> --name ⇒  
<vpn_name> --type 13
```

```
neutron bgpvpn-net-assoc-create <vpn_name> ⇒  
--network <external_network_name>
```

#### Result:

Neutron network for floating IP is created.

4. Create security group rules for IPv4 and IPv6:

- a. Create security group:

```
neutron security-group-create <sec_group_name>
```

- b. Create rules for IPv4 to access VM using floating IP:

```
neutron security-group-rule-create --direction ingress⇒  
--ethertype IPv4 --protocol tcp --port-range-min 22⇒  
--port-range-max 22 --remote-ip-prefix⇒  
0.0.0.0/0 <sec_group_name>
```

```
neutron security-group-rule-create --direction egress⇒  
--ethertype IPv4 --protocol tcp --port-range-min 22⇒  
--port-range-max 22 --remote-ip-prefix⇒  
0.0.0.0/0 <sec_group_name>
```

```
neutron security-group-rule-create --direction ingress ⇒  
--ethertype IPv4 --protocol icmp --remote-ip-prefix⇒  
0.0.0.0/0⇒ <sec_group_name>
```

```
neutron security-group-rule-create --direction egress ⇒  
--ethertype IPv4 --protocol icmp --remote-ip-prefix⇒  
0.0.0.0/0 <sec_group_name>
```

- c. Create rules for IPv6 to verify communication:

```
neutron security-group-rule-create --direction ingress⇒
```



```

--ethertype IPv6 --protocol tcp --port-range-min 22⇒
--port-range-max 22 --remote-ip-prefix⇒
0::0/0 <sec_group_name>

neutron security-group-rule-create --direction egress⇒
--ethertype IPv6 --protocol tcp --port-range-min 22⇒
--port-range-max 22 --remote-ip-prefix⇒
0::0/0 <sec_group_name>

neutron security-group-rule-create --direction ingress⇒
--ethertype IPv6 --protocol icmpv6 --remote-ip-prefix⇒
0::0/0 <sec_group_name>

neutron security-group-rule-create --direction egress⇒
--ethertype IPv6 --protocol icmpv6 --remote-ip-prefix⇒
0::0/0 <sec_group_name>

```

**Result:**

Security group rules for IPv4 and IPv6 are created.

5. Create Neutron network with dual stack:

```

neutron net-create <network_name>

neutron subnet-create --name <subnet_v4_name> ⇒
<network_name> <subnet_range>

neutron subnet-create --name <subnet_v6_name> ⇒
<network_name> --ip-version 6 --enable_dhcp=false ⇒
<subnet_range>

```

**Result:**

All required networks and subnets are created.

6. Add private network IPv4 and IPv6 subnets to the router:

```
neutron router-interface-add <router_id> <subnet_id>
```

**Result:**

Subnets are added to the router.

7. Create Neutron port with IPv6 as allowed address pair:

```

neutron port-create --name <port-name> ⇒
<network_name> --security-group <security_group_name>⇒
--allowed-address-pairs type=dict list=true ip_address=⇒
<ipv6_address_in_given_ipv6_subnet> --fixed-ip ip_address=⇒
<fixed_ipv4_address> --fixed-ip ip_address=<fixed_ipv6_address>

```

**Result:**

Port is created.

8. Create Neutron port with fixed IP address:



```
neutron port-create <network_name> --name <port_name>⇒
--security-group <security_group_name> --fixed-ip ip_address⇒
<fixed_ipv4_address> --fixed-ip ip_address⇒
<fixed_ipv6_address>
```

**Result:**

Port is created.

## 9. Boot two VMs:

```
nova boot --image <image_name> --flavor⇒
<flavor_id> --nic port-id=<port_id>⇒
--availability-zone=zone:<host_name> <vm_name>⇒
```

```
nova list
```

**Result:**

VM instances are created.

## 10. Associate floating IPs to VMs:

## a. Create two floating IPs for two VMs:

```
neutron floatingip-create <external_network_name>
```

## b. Associate floating IPs to VMs:

```
neutron floatingip-associate <floatingip_id>
<vm_port_id>
```

**Result:**

Floating IPs are associated to the VMs.

## 11. Log in to VM1 (for which aap needs to be assigned) using floating IPs:

```
ssh <vm_username>@<floatingip_1>
```

## a. Add fixed IPv6 address to the VM interface:

```
sudo ip addr add <fixed_ipv6_address/prefix>⇒
dev <vm_interface_name>
```

## b. Add allowed address pair IPv6 address as secondary IP to the VM interface:

```
sudo ip addr add <allowed_ipv6_address/prefix>⇒
dev <vm_interface_name>
```

**Result:**

VM is logged in and secondary address is assigned.

## 12. Log in to VM2 using floating IP:



```
ssh <vm_username>@<floatingip_2>
```

- a. Add fixed IPv6 address to the VM interface:

```
sudo ip addr add <fixed_ipv6_address/prefix>⇒  
dev <vm_interface_name>
```

- b. Verify connectivity:

```
ping6 <allowed_address_pair_ip>
```

```
ssh <vm_username>@<allowed_address_pair_ip>
```

**Result:**

Connectivity to VM1 is successful.

13. Delete VM instances:

```
nova delete <vm_id>
```

**Result:**

VM instances are deleted.

14. Delete floating IPs:

```
neutron floating-ip-delete <floatingip_id>
```

**Result:**

Floating IP deleted

15. Delete router:

- a. Delete private net sub-interface:

```
neutron router-interface-delete <router_name>  
<subnet_id>
```

- b. Clear gateway:

```
neutron router-gateway-clear <router_name>
```

- c. Delete router:

```
neutron router-delete <router_id>
```

**Result:**

Router is deleted.

16. Delete ports:

```
neutron port-delete <port_id>
```

**Result:**



Ports are deleted.

#### 17. Delete networks:

```
neutron net-delete <network_id>
```

**Result:**

Network is deleted.

#### 18. Delete security group:

```
neutron security-group-delete <security_group_name>
```

**Result:**

Security group is deleted.

## 2.30 Verify IPv6 Ingress Rules

This section is applicable for HDS deployment.

### Description

Use this test to verify that VMs can communicate with each other based on ingress rules and IPv6 address assigned through Neutron router in SLAAC addressing mode.

Table 31 Verifying IPv6 Ingress Rules

Objective	Verify IPv6 ingress rules.
Precondition	All expected OpenStack and SDN services are up and running. SDN health check is done.
Postcondition	All expected OpenStack and SDN services are up and running. SDN health check is done.
Duration	
Platform	HDS
Executor	
Date and Time of Execution	
Log and Trace Name	
Result	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
Comment	



## Procedure

Do the following:

1. Log on to vFuel:

```
ssh <vfuel_address>
```

**Result:**

Logged on to vFuel.

2. Log on to a vCIC:

```
ssh <vcic_name>
```

**Result:**

Logged on to vCIC.

3. Create network for floating IP:

- a. Create external network:

```
neutron net-create <external_network_name> ⇒  
--router:external=True --provider:network_type gre
```

```
neutron subnet-create <external_network_name> ⇒  
--name <subnet_name> --disable-dhcp ⇒  
<public_range_subnet_cidr>
```

```
neutron net-external-list
```

- b. Create router and set gateway:

```
neutron router-create <router_name>
```

```
neutron router-gateway-set <router_id> ⇒  
<external_network_id>
```

- c. Create L3VPN and associate it to external network:

```
neutron bgpvpn-create --export-targets ⇒  
<EXPORT-TARGETS> --import-targets <IMPORT-TARGETS> ⇒  
--route-distinguishers <ROUTE-DISTINGUISHERS> --name ⇒  
<vpn_name> --type 13
```

```
neutron bgpvpn-net-assoc-create <vpn_name> ⇒  
--network <external_network_name>
```

**Result:**

Network for floating IP is created.

4. Create security group rules for IPv4 and ICMP rule only for IPv6:





- a. Create security group:

```
neutron security-group-create <sec_group_name>
```

- b. Create rules for IPv4 to access VM using floating IP:

```
neutron security-group-rule-create --direction ingress⇒
--ethertype IPv4 --protocol tcp --port-range-min 22⇒
--port-range-max 22 --remote-ip-prefix⇒
0.0.0.0/0 <sec_group_name>
```

```
neutron security-group-rule-create --direction egress⇒
--ethertype IPv4 --protocol tcp --port-range-min 22⇒
--port-range-max 22 --remote-ip-prefix⇒
0.0.0.0/0 <sec_group_name>
```

```
neutron security-group-rule-create --direction ingress ⇒
--ethertype IPv4 --protocol icmp --remote-ip-prefix⇒
0.0.0.0/0 <sec_group_name>
```

```
neutron security-group-rule-create --direction egress ⇒
--ethertype IPv4 --protocol icmp --remote-ip-prefix ⇒
0.0.0.0/0 <sec_group_name>
```

- c. Create IPv6 rules to verify IPv6 communication:

```
neutron security-group-rule-create --direction ingress⇒
--ethertype IPv6 --protocol icmpv6 --remote-ip-prefix ⇒
0::0/0 <sec_group_name>
neutron security-group-rule-create --direction egress⇒
--ethertype IPv6 --protocol icmpv6 --remote-ip-prefix ⇒
0::0/0 <security_group_name>
```

**Result:**

Security group with IPv4 and IPv6 rules is created.

5. Create Neutron network with dual stack:

```
neutron net-create <network_name>
```

```
neutron subnet-create --name <subnet_v4_name> ⇒
<network_name> <subnet_range>
```

```
neutron subnet-create --name <subnet_v6_name> ⇒
<network_name> --ip-version 6 --ipv6_address_mode=slaac ⇒
--ipv6_ra_mode=slaac <subnet_range>
```

**Result:**

All required networks and subnets are created.

6. Add private IPv4 and IPv6 subnets to router:

```
neutron router-interface-add <router_id> ⇒
```



```
<subnet_id>
```

**Result:**

Subnets are added to the router.

7. Create two Neutron ports with security groups:

```
neutron port-create <network_name> --security-group=>
<security_group_name> --name <port_name>
```

**Result:**

Neutron ports are created.

8. Boot two VMs using port ID:

```
nova boot --image <image_name> --flavor<flavor_id>=>
--nic port-id=<port_id> --availability-zone=zone:>
<host_name> <vm_name>
```

```
nova list
```

**Result:**

VMs are created.

9. Associate floating IPs to VMs:

- a. Create two floating IPs for two VMs:

```
neutron floatingip-create <external_network_name>
```

- b. Associate floating IPs to VMs:

```
neutron floatingip-associate <floatingip_id>
<vm_port_id>
```

**Result:**

Floating IPs are associated to the VMs.

10. Log in to VM1 using floating IP:

```
ssh <vm_username>@<floatingip_1>
```

- a. Verify connectivity to VM2:

```
ping6 <vm2_ipv6_address>
```

```
ssh <vm_username>@<vm2_ipv6_address>
```

**Result:**

Connectivity using ping to VM2 is successful. Connectivity using SSH to VM2 fails, because there is no security group rule.

11. Delete VM instances:



```
nova delete <vm_id>
```

**Result:**

VM instances are deleted.

## 12. Delete floating IPs:

```
neutron floating-ip-delete <floatingip_id>
```

**Result:**

Floating IP is deleted.

## 13. Delete router:

## a. Delete private net sub-interface:

```
neutron router-interface-delete <router_name>  
<subnet_id>
```

## b. Clear gateway:

```
neutron router-gateway-clear <router_name>
```

## c. Delete router:

```
neutron router-delete <router_id>
```

**Result:**

Router is deleted.

## 14. Delete ports:

```
neutron port-delete <port_id>
```

**Result:**

Ports are deleted.

## 15. Delete network:

```
neutron net-delete <network_id>
```

**Result:**

Network is deleted.

## 16. Delete security group:

```
neutron security-group-delete <security_group_name>
```

**Result:**

Security group is deleted.



## 2.31 Verify Software RAID Configuration

This section is applicable for all deployments.

**Note:** Perform the test described in this section if verifying Software Raid functionality is requested.

### Description

Use this test to verify the software RAID configuration on compute blades.

### Procedure

Do the following:

1. Modify the `config.yaml` to enable `swraid` on any compute blade as described in the Software Raid Configuration section in the [Configuration File Guide](#).
2. Deploy the node to enable `swraid` on the compute blades:

- a. Deploy the node using the `installcee.sh` script if it is a fresh installation.
- b. Run the following command if it is a deployed node:

```
removeceenode --name <compute-name> --force  
expandcee --repair
```

3. Log on to the compute blade that has `swraid` configured.
4. Check the contents of `/proc/mdstat`. If correctly configured, active `raid1` is in the result.

### Result:

An example of the printout is:

```
root@compute-0-5:~# cat /proc/mdstat  
Personalities: [linear] [multipath] [raid0] [raid1] [raid6]⇒  
[raid5] [raid4] [raid10]  
md124 : active raid1 sdb6[1] sda6[0]  
295055168 blocks super 1.2 [2/2] [UU]  
  
md125 : active raid1 sdb3[1] sda3[0]  
204608 blocks super 1.2 [2/2] [UU]  
  
md126 : active raid1 sdb5[1] sda5[0]  
41910144 blocks super 1.2 [2/2] [UU]  
  
md127 : active raid1 sdb4[1] sda4[0]  
52395904 blocks super 1.2 [2/2] [UU]  
  
unused devices: <none>
```



## 2.32 Check Scale-In Functionality on Compute Hosting vFuel

This section is not applicable for single server deployments.

In the current release of CEE, only the removal of compute hosts not hosting vFuel or vCIC is possible. An error message is the expected result of this test case.

### Description

Use this test to check the scale-in functionality of the CEE Region.

Table 32 Check Scale-In Functionality on Compute Hosting vFuel

<b>Objective</b>	<b>Verify that a compute hosting vFuel cannot be removed from the CEE Region.</b>
<b>Precondition</b>	All expected OpenStack services are up and running.
<b>Postcondition</b>	All expected OpenStack services are up and running.
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"> <li>• Dell multi-server</li> <li>• HDS</li> <li>• BSP</li> </ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

### Procedure

Do the following:

1. Log on to vFuel using SSH.
2. Fetch the ID of the compute node hosting vFuel:  

```
get_vfuel_info --name --primary
```
3. Run the below command to attempt to remove the compute host from the CEE Region:



```
ceescalein --names <compute_host>
```

**Result:**

The following error message is displayed:

```
"Error: Cannot remove node <compute_node> as it is hosting  
primary/secondary vFuel"
```

## 2.33 Check Scale-In Functionality on Compute Hosting vCIC

This section is not applicable for single server deployments.

In the current release of CEE only the removal of compute hosts not hosting vFuel or vCIC is possible. An error message is the expected result of this test case.

**Description**

Use this test to check the scale-in functionality of the CEE Region.

Table 33 Check Scale-In Functionality on Compute Hosting vCIC

<b>Objective</b>	<b>Verify that a compute hosting vCIC cannot be removed from the CEE Region.</b>
<b>Precondition</b>	All expected OpenStack services are up and running.
<b>Postcondition</b>	All expected OpenStack services are up and running.
<b>Duration</b>	
<b>Platform</b>	<ul style="list-style-type: none"><li>• Dell multi-server</li><li>• HDS</li><li>• BSP</li></ul>
<b>Executor</b>	
<b>Date and Time of Execution</b>	
<b>Log and Trace Name</b>	
<b>Result</b>	<input type="checkbox"/> OK <input type="checkbox"/> Not OK <input type="checkbox"/> N/A
<b>Comment</b>	

**Procedure**

Do the following:

1. Log on to vFuel using SSH.



2. Fetch the ID of the compute node hosting the vCIC:

```
fuel node |grep virt |cut -d "|" -f3 |head -1
```

3. Run the below command to attempt to remove the compute host from the CEE Region:

```
ceescalein --names <compute_host>
```

**Result:**

The following error message is displayed:

```
"Scale-in is supported only for pure computes. <node_role> is selected"
```