Openstack Installation

Tests

Stato del deliverable

Ver.	Data	Autore della modifica	Note	Validazione
1.0	15-11-2014	Marica Antonacci (INFN-BARI)	Prima Stesura	INFN, ALMAVIVA
1.1	24-12-2014	Marica Antonacci (INFN-BARI)	Test plan ampliato e riorganizzato; individuate 3 categorie di test: basic tests, nominal tests, failover/recovery tests.	INFN, ALMAVIVA

Reference Documents

ID	Title
R1	[PRISMA] Openstack Installation Guide (https://github.com/pon-prisma/OpenStack-Installation-Guide)
R2	[PRISMA] Enterprise IaaS: High-availability Implementation Guide

INDICE DEGLI ARGOMENTI

1. Introduction

This documents provides a plan of tests that can be run to check and verify the correct installation [R1, R2] of your Openstack infrastructure.

Three different categories of tests will be described:

1) **Basic Sanity Tests:** they include some basic health checks that verify the status of the services and, if applicable, the status of the connections towards critical components, like the database and/or to the message broker (AMQP server). Moreover, the correct functioning of the api servers is checked for all the Openstack services (Keystone, Glance, Nova, Cinder, Neutron, etc.) using the openstack CLI clients.
2) **Nominal Tests**: they include a set of functional tests that allow to verify the system behaviour under standard operational conditions.
   All these tests use the basic Openstack services (Nova, Glance, Keystone, Cinder, Neutron, etc) and therefore it is important to check the status of these services before running the tests (that can be done using the series of basic sanity tests).
3) **Failover/Recovery Tests:** these tests ensure that the system can successfully failover and recover from fault conditions (e.g. hardware, software, or network malfunctions). In particular, the failover testing ensures that, for those deployments that implement the High-Availability of the Openstack services, the backup “node” properly “takes over” for the failed one with minimum downtime.

Note that the basic and nominal tests can be conducted on your Openstack installation both in case you have implemented the high-availabilty (HA) as described in [R2] and in case you have not enabled the HA option.

2. Basic Sanity Tests

2.1 Verify the Identity Service (Keystone)

The following steps allow to verify the correct installation of the Openstack Identity Service:

1. if set, clear OS_SERVICE_TOKEN and OS_SERVICE_ENDPOINT environment variables:

```
$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

2. Request an authentication token by using the admin user and the password you chose for that user:

```
$ keystone --os-username=admin --os-password=$ADMIN_PASS --os-auth-url=http://<controller_ip>:35357/v2.0 token-get
```

In response, you should receive a token paired with your **user ID**. This verifies that the Identity Service is running on the expected endpoint and that your user account is established with the expected credentials.

The expected output is shown hereafter:

```
+----------+-------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------------------------------------------------------------------------------
-------+
| Property |
Value
|
+----------+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------
-------+
| expires |
2014-11-11T17:12:03Z
|
| id |
```

MIIC8QYJKoZIhvcNAQcCoIIC4jCCAt4CAQExCTAHBgUrDgMCGjCCAUcGCSqGSIb3DQEHAaCCATgEggE0eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAi
MjAxNC0xMS0xMVQxNjoxMjowMy4zMjE1NTIiLCAiZXhwaXJlcyI6ICIyMDE0LTExLTExVDE3OjEyOjAzWiIsICJpZCI6ICJwbGFjZWhvbGRlciJ9LCAic2VydmljZUNhd
GFsb2ciOiBbXSwgInVzZXIiOiB7InVzZXJuYW1lIjogImFkbWluIiwgInJvbGVzX2xpbmtzIjogW10sICJpZCI6ICI3NjliN2ZhZmQ2YzE0NWNjYjg2ODBlMzk4NzAyZjE0Z
iIsICJyb2xlcyI6IFtdLCAibmFtZSI6ICJhZG1pbiJ9LCAibWV0YWRhdGEiOiB7ImlzX2FkbWluIjogMCwgInJvbGVzIjogW119fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVB
AYTAlVTMQ4wDAYDVQQIDAVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0MRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBw
YFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAFgI+R1fAQQSCqB9qkUt0T4RTNswsI1PnEY9f7zcb50GJJMB8nHXxhkRhtD0KiwkRnLDkeNsFzSgQd+YQ8HS09aaB
L8M5t2SWI+hr4ki2q8buEuRhgpW+ePGlI2LuUPW3F8968W-BVX5OwflHp+AYa3ROfl6T9XxWgesKL7DcMmDem1Uq5w9OR3682m-

```
n4NYAcewkiDPPoj+NRkEmpRZdQVDm9vtitp-RPMcySXW7KSiWzSOD1AbqK7Ug5mh1PSSGAb7LRg8fQ-
w8+ZQMnVpks+uqObNc7MuFqhlIK1PTe1oFLS2YwCp4BtrbWD8xu0dVRchiLpFzcFwBVNnXuC47KA== |
| user_id |
769b7fafd6c145ccb8680e398702f14f
|
+----------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------+
```

3. Verify that authorization behaves as expected. To do so, request authorization on a tenant:

```
$ keystone --os-username=admin --os-password=$ADMIN_PASS --os-tenant-name=admin --os-auth-url=http://<controller_ip>:35357/v2.0 token-get
```

In response, you should receive a token that includes the ID of the tenant that you specified. This verifies that your user account has an explicitly defined role on the specified tenant and the tenant exists as expected.

The expected output is shown hereafter:

+----------+----------+-------+
| Property | Value |
+----------+----------+-------+
| expires | 2014-11-18T03:04:05Z |
| **id** | MIIrygYJKoZIhvcNAQcCoIIruzCCK7cCAQExCTAHBgUrDgMCGjCCKqMGCSqGSIb3DQEHAaCCKpQEgiqQeyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0xMS0xOFQwMDowNDowNS40MzI1MzciLCAiZXhwaXJlcyI6ICIyMDE0LTExLTE4VDAzOjA0OjA1WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIiIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogImFmYjQ5Nzg3OTZmNTQyMmQ5YWNkZWM2NGRhNmFhZjVmIiwgIm5hbWUiOiAiYWRtaW4ifX0sICJzZXJ2aWNlQ2F0YWxvZyI6IFt7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9jbG91ZDAzLnJvbWEyLmluZm4uaXQ6ODc3NC92Mi9hZ |
+----------+----------+-------+

mI0OTc4Nzk2ZjU0MjJkOWFjZGVjNjRkYTZhYWY1ZiIsICJyZWdpb24iOiAicm0yIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9jbG91ZDAzLnJvbWEyLmluZm4uaXQ6ODc3NC92Mi9hZmI0OTc4Nzk2ZjU0MjJkOWFjZGVjNjRkYTZhYWY1ZiIsICJpZCI6ICI0OTcxNjlmMzAxMGI0ZmI3YTJkODAzYWU5N2JkZjU2MCIsICJwdWJsaWNVUkwiOiAiaHR0cDovL2Nsb3VkMDMucm9tYTIuaW5mbi5pdDo4Nzc0L3YyL2FmYjQ5Nzg3OTZmNTQyMmQ5YWNkZWM2NGRhNmFhZjVmIn0sIHsiYWRtaW5VRUkwiOiAiaHR0cDovL2ljZWN0bC5sbmdzLmluZm4uaXQ6ODc3NC92Mi9hZmI0OTc4Nzk2ZjU0MjJkOWFjZGVjNjRkYTZhYWY1ZiIsICJyZWdpb24iOiAidGVzdHJlZ2lvbiIsICJpbnRlcm5hbFVSTCI6ICJodHRwOi8vaWNlY3RsLmxuZ3MuaW5mbi5pdDo4Nzc0L3YyL2FmYjQ5Nzg3OTZmNTQyMmQ5YWNkZWM2NGRhNmFhZjVmIiwgImlkIjogIjFhZTVhMDYyMDFkNDQ1YTE4YzAzMjMzYmQ2NGJhOTM3IiwgInB1YmxpY1VSTCI6ICJodHRwOi8vaWNlY3RsLmxuZ3MuaW5mbi5pdDo4Nzc0L3YyL2FmYjQ5Nzg3OTZmNTQyMmQ5YWNkZWM2NGRhNmFhZjVmIn0sIHsiYWRtaW5VUkwiOiAiaHR0cDovL2JhcmktcmVnaW9uLWN0bC5iYS5pbmZuLml0Ojg3NzQvdjIvYWZiNDk3ODc5NmY1NDIyZDlhY2RlYzY0ZGE2YWFmNWYiLCAicmVnaW9uIjogImJhcmktcmVnaW9uLWN0bC5iYS5pbmZuLml0Ojg3NzQvdjIvYWZiNDk3ODc5NmY1NDIyZDlhY2RlYzY0ZGE2YWFmNWYiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2JhcmktcmVnaW9uLWN0bC5iYS5pbmZuLml0Ojg3NzQvdjIvYWZiNDk3ODc5NmY1NDIyZDlhY2RlYzY0ZGE2YWFmNWYiLCAiaWQiOiAiOTE2ODg5MGQ3NTMzNGY2OThjZmNhMTgyZTM4ZmFlNWUiLCAicHVibGljVVJMIjogImh0dHA6Ly9iYXJpLXJlZ2lvbi1jdGwuYmEuaW5mbi5pdDo4Nzc0L3YyL2FmYjQ5Nzg3OTZmNTQyMmQ5YWNkZWM2NGRhNmFhZjVmIn0sIHsiY |

| **tenant_id** |

afb4978796f5422d9acdec64da6aaf5f

|

| **user_id** |

13b300f990ec4826a23cd252089e6cad

|

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

-------+

You can also set your --os-* variables in your environment to simplify command-line usage.

For the following tests, create an admin-openrc.sh file in your home directory, with the following content:

```
export OS_USERNAME=admin

export OS_PASSWORD=$ADMIN_PASS

export OS_TENANT_NAME=admin

export OS_AUTH_URL=http://<controller_ip>:35357/v2.0
```

2.2 Verify the Image Service (Glance)

Preparation:

Set your --os-* variables in your environment:

```
$ source admin-openrc.sh
```

The following steps allow to verify the correct installation of the Openstack Image Service:

1. Verify that the Glance API is correctly configured and working:

   ```
   $ glance image-list
   ```

   If you have not uploaded any image yet, the output of this command will show an empty table; otherwise you will get the list of registered images.

2. Verify that the Glance registry process is working correctly by importing the Cirros image available online:

   ```
   $ wget http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img

   $ glance image-create --name "cirros-0.3.2-x86_64" --disk-format qcow2 --container-format bare --is-public True --progress < cirros-0.3.2-x86_64-disk.img
   ```

3. Check the status of the image using the ID returned by the previous command:

   ```
   $ glance image-show <image_id>
   ```

2.3 Verify the Compute Service (Nova)

Preparation:

Set your --os-* variables in your environment:

```
$ source admin-openrc.sh
```

The following steps allow to verify the correct installation of the Openstack Compute Service:

1. Verify that all nova processes are up and running:

   ```
   $ for s in /etc/init.d/nova-*; do status $(basename $s); done
   ```

   The expected output is like the following:

   ```
   nova-api start/running, process 26601

   nova-cert start/running, process 26615

   nova-conductor start/running, process 26625

   nova-consoleauth start/running, process 26659

   nova-novncproxy start/running, process 26669

   nova-scheduler start/running, process 26643
   ```

   If one or more of the above processes are stopped, look at the log files in the folder /var/log/nova in order to find the problem.

2. To verify your configuration, list available images:

```
$ nova image-list
```

The output should be like this:

```
+--------------------------------------+--------------------+--------+--------+
 |                  ID                  |        Name        |        Status        |        Server        |
+--------------------------------------+--------------------+--------+--------+
 |   acafc7c0-40aa-4026-9673-b879898e1fc2    |   cirros-0.3.2-x86_64    |   ACTIVE    |                          |
+--------------------------------------+--------------------+--------+--------+
```

3. Verify the process “nova-cert” is running and connected to the database and messaging server (AMQP):
   a. Verify the connection with the database (replace the DB_PORT value with the correct value depending on your installation. Default is 3306)

```
# export DB_PORT=3306

# netstat -punt | grep -s $DB_PORT | grep -s $(pgrep nova-cert)
```

The expected output shows the ESTABILISHED connection:

```
tcp        0      0 192.168.22.205:39263     192.168.205.99:33306     ESTABLISHED 23311/python
```

b. Verify the connection with the messaging server (replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672)

```
# export AMQP_PORT=5672

# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep nova-cert)
```

The expected output shows the ESTABILISHED connection:

```
tcp        0      0 192.168.22.205:52498     192.168.22.68:5672        ESTABLISHED 23311/python
```

4. Verify the process “nova-scheduler” is running and connected to the database and messaging server (AMQP):
   a. Verify the connection with the database (replace the DB_PORT value with the correct value depending on your installation. Default is 3306)

```
# export DB_PORT=3306

# netstat -punt | grep -s $DB_PORT | grep -s $(pgrep nova-scheduler)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:39303     192.168.205.99:33306     ESTABLISHED 23943/python

tcp        0      0 192.168.22.205:39355     192.168.205.99:33306     ESTABLISHED 23943/python
```

b. Verify the connection with the messaging server (replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672)

```
# export AMQP_PORT=5672

# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep nova-scheduler)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:52528     192.168.22.68:5672      ESTABLISHED 23943/python

tcp        0      0 192.168.22.205:53220     192.168.22.68:5672      ESTABLISHED 23943/python
```

5. Verify the process “nova-consoleauth” is running and connected to the database and messaging server (AMQP):
   a. Verify the connection with the database (replace the DB_PORT value with the correct value depending on your installation. Default is 3306)

```
# export DB_PORT=3306

# netstat -punt | grep -s $DB_PORT | grep -s $(pgrep -f /usr/bin/nova-consoleauth)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:39257     192.168.205.99:33306    ESTABLISHED 23532/python
```

b. Verify the connection with the messaging server (replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672)

```
# export AMQP_PORT=5672

# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep -f /usr/bin/nova-consoleauth)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:43638     192.168.22.68:5672      ESTABLISHED 23532/python

tcp        0      0 192.168.22.205:52524     192.168.22.68:5672      ESTABLISHED 23532/python

tcp        0      0 192.168.22.205:52696     192.168.22.68:5672      ESTABLISHED 23532/python

tcp        0      0 192.168.22.205:53833     192.168.22.68:5672      ESTABLISHED 23532/python
```

6. Verify the process “nova-novnc” is running and listening on its port (default is 6080):

```
# netstat -a | grep -s 6080
```

The expected output is like the following:

```
tcp        0      0 preprod-01.ba.infn:6080 *:*                     LISTEN
```

2.4 Verify the Block Storage Service (Cinder)

Preparation:

Set your --os-* variables in your environment:

```
$ source admin-openrc.sh
```

The following steps allow to verify the correct installation of the Openstack Block Storage Service:

1. Verify that all cinder processes are up and running:

   ```
   $ for s in /etc/init.d/cinder-*; do status $(basename $s); done
   ```

   The expected output is like the following:

   ```
   cinder-api start/running, process 30050

   cinder-scheduler start/running, process 30087

   cinder-volume start/running, process 30127
   ```

   If one or more of the above processes are stopped, look at the log files in the folder /var/log/cinder in order to find the problem. Note: depending on your installation, the cinder-volume process may not be running on the controller node (if you have decided to deploy it on a dedicated host, you should check its status there: ssh <cinder-host> service cinder-volume status).

7. Verify the process “cinder-scheduler” is running and connected to the database and messaging server (AMQP):

a. Verify the connection with the database (replace the DB_PORT value with the correct value depending on your installation. Default is 3306)

```
# export DB_PORT=3306
# netstat -punt | grep -s $DB_PORT | grep -s $(pgrep -f /usr/bin/cinder-scheduler)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:45718     192.168.205.99:33306    ESTABLISHED 23943/python
tcp        0      0 192.168.22.205:45812     192.168.205.99:33306    ESTABLISHED 23943/python
```

b. Verify the connection with the messaging server (replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672)

```
# export AMQP_PORT=5672
# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep -f /usr/bin/cinder-scheduler)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.205:52528     192.168.22.68:5672       ESTABLISHED 23943/python
tcp        0      0 192.168.22.205:53220     192.168.22.68:5672       ESTABLISHED 23943/python
```

8. Finally, to verify that cinder is configured properly, create a new volume:

```
$ cinder create --display-name test 1
```

The expected output is like the following:

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-06-22T01:14:02.705154      |
| display_description |                 None                 |
|     display_name    |                 test                 |
|      encrypted      |                False                 |
|          id         | ad2f9004-3939-4b1c-a234-8ab26b8fe961 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
```

```
|     snapshot_id      |                  None                  |
|     source_volid     |                  None                  |
|        status        |                creating                |
|     volume_type      |                  None                  |
+----------------------+----------------------------------------+
```

9. Check the volume status using the command “cinder list”. The status should pass from “creating” to “available”:

```
$ cinder list
```

The expected output is like the following:

```
+--------------------------------------+-----------+--------------+------+-------------+----------+-----------
--+
|                  ID                  |   Status  | Display Name | Size | Volume Type | Bootable | Attached
to |
+--------------------------------------+-----------+--------------+------+-------------+----------+-----------
--+
| ad2f9004-3939-4b1c-a234-8ab26b8fe961 | available |      test      |   1    |       None      |   false    |
|
| cfe55712-5933-42fe-b9a2-aacaa8620cd6 |   creating |      test      |   1    |       None      |   false    |
|
```

```
+--------------------------------------+-----------+--------------+------+-------------+----------+-----------
--+
```

If the status value is not *available*, the volume creation failed. Check the log files in the /var/log/cinder/ directory on the controller and volume nodes to get information about the failure.

2.5 Verify the Networking Service (Neutron)

Preparation:

Set your --os-* variables in your environment:

```
$ source admin-openrc.sh
```

The following steps allow to verify the correct installation of the Openstack Networking Service. In this guide we assume that the networking services have been deployed onto a dedicated node (network node); therefore the following commands should be issued on the network node.

1. Verify that all neutron processes are up and running:

```
$ for s in /etc/init.d/neutron-*; do status $(basename $s); done
```

The expected output is like the following:

```
neutron-dhcp-agent start/running, process 7515

neutron-l3-agent start/running, process 7529

neutron-metadata-agent start/running, process 7537

neutron-ovs-cleanup start/running

neutron-plugin-openvswitch-agent start/running, process 7812

neutron-server start/running, process 7820
```

If one or more of the above processes are stopped, look at the log files in the folder /var/log/neutron in order to find the problem.

2. Query the neutron API to get the list of networks:

```
$ neutron net-list
```

The expected output shows the list of the available networks (if any).

3. Verify the process “neutron-dhcp-agent” is running and connected to the messaging server (AMQP):
   a. replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672

```
# export AMQP_PORT=5672

# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep -f /usr/bin/neutron-dhcp-agent)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.218:47911    192.168.22.68:5672      ESTABLISHED 14058/python
tcp        0      0 192.168.22.218:47912    192.168.22.68:5672      ESTABLISHED 14058/python
tcp        0      0 192.168.22.218:47910    192.168.22.68:5672      ESTABLISHED 14058/python
```

4. Verify the process “neutron-l3-agent” is running and connected to the messaging server (AMQP):
   b. replace the AMQP_PORT value with the correct value depending on your installation. Default is 5672

```
# export AMQP_PORT=5672
```

```
# netstat -punt | grep -s $AMQP_PORT | grep -s $(pgrep -f /usr/bin/neutron-l3-agent)
```

The expected output shows the ESTABILISHED connections:

```
tcp        0      0 192.168.22.218:46892    192.168.22.69:5672      ESTABLISHED 9986/python
tcp        0      0 192.168.22.218:47843    192.168.22.68:5672      ESTABLISHED 9986/python
tcp        0      0 192.168.22.218:46891    192.168.22.69:5672      ESTABLISHED 9986/python
```

2.6 Verify the services on the Compute Nodes

Check that the compute and networking agents are up and running and able to communicate with the controller. Use the commands “nova service-list” and “neutron agent-list” (they can be issued from the controller node).

1. Load the admin credentials:

```
$ source admin-openrc.sh
```

2. verify that all the compute nodes are up:

```
$ nova service-list | grep nova-compute
```

```
| nova-compute     | preprod-05 | nova     | enabled | up   | 2014-11-10T12:30:30.000000 | None            |
| nova-compute     | preprod-03 | nova     | enabled | up   | 2014-11-17T15:49:45.000000 | None            |
| nova-compute     | preprod-04 | nova     | enabled | up    | 2014-11-17T22:48:51.000000 | None
```

3. verify that the neutron open-vswitch agent is running on the compute nodes:

```
$ neutron agent-list | grep vSwitch
```

The output shows “:-)” if the service is working fine or “xxx” if there are problems

```
| 156c39d5-f685-450a-8fef-1bf9ca3c1e0f | Open vSwitch agent | preprod-05 | :-)   | True           |
| 30facdbc-7ae5-4f84-b11c-b8908544c7af | Open vSwitch agent | preprod-04 | :-)   | True           |
```

```
| df7f0820-24d1-41e6-82db-fec1e3296af5 | Open vSwitch agent | preprod-03 | :-)   | True           |
| ed2a74b3-f246-44dd-a3f4-d1170e30b3c0 | Open vSwitch agent | preprod-02 | :-)   | True           |
```

3. End-to-End (Nominal) Tests

3.1 Verify the User creation (only as admin) and authentication

The following bash script can be used to check that the installed Openstack infrastructure is able to provide basic user management functions.

To execute the script you must fill the variables at the beginning of the file with proper values depending on your installation.

```
#!/bin/bash

########################################

# BEFORE RUNNING THIS SCRIPT FILL THE

# FOLLOWING VARIABLES WITH PROPER VALUES

########################################

export OS_USERNAME=admin #do not change this. Only admin can create users

export OS_PASSWORD=<password>

export OS_TENANT_NAME=admin

export OS_AUTH_URL=http://<controller_ip>:35357/v2.0

########################################


######################
```

```
###  MAIN

######################


if [ $# -ne 0 ]; then

      echo "Usage: ./`basename $0`"

      echo -e "\nThis probe tries to create a user and then tries to get a token using the user credentials.\nExit code: 0 - probe successfully run; 1 - probe failed"

      exit 1

fi


TENANT_ID=$(keystone tenant-create --name testenant | sed -n  's/^|\ \+id\ \+|\ \+\([^\ ].*\)\ \+|/\1/p')

keystone tenant-list | grep -q -s $TENANT_ID


if [ $? -ne 0 ]; then

   echo "Some error occurred while creating tenant"

   exit 1

else

   echo "Tenant testenant ($TENANT_ID) successfully created"
```

```
fi


USER_ID=$(keystone user-create --name tesuser --tenant-id $TENANT_ID --pass 12qwas | sed -n  's/^|\ \+id\ \+|\ \+\([^\
].*\)\ \+|/\1/p')
keystone user-list | grep -q -s $USER_ID


if [ $? -ne 0 ]; then
   echo "Some error occurred while creating user"
   exit 1
else
   echo "User tesuser ($USER_ID) successfully created"
fi


keystone --os-tenant-id $TENANT_ID --os-username tesuser --os-password 12qwas token-get | grep -q -s user_id


if [ $? -ne 0 ]; then
   echo "Some error occurred during authentication of user tesuser ($USER_ID) in tenant testenant ($TENANT_ID)"
   exit 1
```

```
else
   echo "User tesuser ($USER_ID) successfully authenticated in tenant testenant ($TENANT_ID)"
fi


echo Cleaning env
keystone user-delete $USER_ID && echo "user deleted successfully" || (echo "Error deleting user tesuser" && exit 1)
keystone tenant-delete $TENANT_ID && echo "tenant deleted successfully" || (echo "Error deleting tenant testenant" && exit 1)
```

3.2 Verify the VM instantiation

The following bash script can be used to check that the installed Openstack infrastructure is able to provide running virtual machines.

To execute the script you must fill the variables at the beginning of the file with proper values depending on your installation.

IMAGE_ID

To set the IMAGE_ID variable you can use the command “glance image-list” to list the available image ids.

FLAVOR_ID

To set the FLAVOR variable you can use the command “nova flavor-list” to list the available flavors (both the flavor name and id can be used).

KEY_NAME

To set the KEY_NAME variable you can use the command “nova keypair-list” to list the available keypairs

NET_ID

To set the NET_ID variable you can use the command “neutron net-list” to list the available networks.

```
#!/bin/bash

########################################

# BEFORE RUNNING THIS SCRIPT FILL THE

# FOLLOWING VARIABLES WITH PROPER VALUES

########################################

export OS_USERNAME=admin

export OS_PASSWORD=<password>

export OS_TENANT_NAME=admin

export OS_AUTH_URL=http://<controller_ip>:35357/v2.0

export IMAGE_ID=<image id>

export FLAVOR=<flavor id or name>

export KEY_NAME=<key name>
```

```
export NET_ID=<network id>

########################################

LOOP_THRESH=5

WAIT_TIMEOUT=30


wait_vm_active()

{

    typeset vmid=$1

    let i=0

    status=

    while [ $i -lt $LOOP_THRESH -a "$status" != active ]

    do

        let i++

        status=$(nova show $vmid | awk '/OS-EXT-STS:vm_state/{print $4}')

        echo "VM status is <$status>"

        [ "$status" = active ] && continue

        sleep 30

    done
```

```
    [ "$status" = active ]
}


######################
###  MAIN
######################
if [ $# -ne 0 ]; then
      echo "Usage: ./`basename $0`"
      echo -e "\nThis probe tries to create a new VM.\nExit code: 0 - probe successfully run (vm is active); 1 - probe failed"
      exit 1
fi


#create the test VM
VM_ID=$(nova boot --image $IMAGE_ID --key-name $KEY_NAME --flavor $FLAVOR --nic net-id=$NET_ID test-vm | sed -n  's/^|\ \+id\ \+|\ \+\([^\ ].*\)\ \+|/\1/p')
echo "VM id is <$VM_ID>"
# wait for vm to become active
wait_vm_active $VM_ID
```

```
# check status
if [ $? -ne 0 ]; then
      echo "Error: instance not running after $LOOP_THRESH x $WAIT_TIMEOUT [sec]"
      exit_code=1
else
      echo "OK. VM creation was successful"
      exit_code=0
fi

# terminate instance
nova delete $VM_ID

#return 0 if test is ok, 1 otherwise
exit $exit_code
```

3.3 Verify the Volume creation and attachment

The following bash script can be used to check that the installed Openstack infrastructure is able to provide on-demand block devices to be attached to the running VMs. The component in charge of providing block storage functionalities is called “Cinder”.

To execute the script you must fill the variables at the beginning of the file with proper values depending on your installation.

```
#!/bin/bash

########################################

# BEFORE RUNNING THIS SCRIPT FILL THE

# FOLLOWING VARIABLES WITH PROPER VALUES

########################################

export OS_USERNAME=admin

export OS_PASSWORD=<password>

export OS_TENANT_NAME=admin

export OS_AUTH_URL=http://<controller_ip>:5000/v2.0


LOOP_THRESH=10

WAIT_TIMEOUT=30

```

```
wait_volume_available()

{

    typeset volid=$1


    let i=0

    status=

    while [ $i -lt $LOOP_THRESH -a "$status" != available ]

    do

        let i++


        status=$(cinder show $volid | awk '/ status /{print $4}')

        echo "Volume status is <$status>"

        [ "$status" = available ] && continue

        sleep 3

    done


    [ "$status" = available ]

}
```

```
wait_volume_in_use()
{
    typeset volid=$1

    let i=0
    status=
    while [ $i -lt $LOOP_THRESH -a "$status" != in-use ]
    do
        let i++

        status=$(cinder show $volid | awk '/ status /{print $4}')
        echo "Volume status is <$status>"
        [ "$status" = in-use ] && continue
        sleep 3
    done

    [ "$status" = in-use ]
```

```
}


######################

###  MAIN

######################


if [ $# -gt 1 ]; then

      echo "Usage: ./`basename $0` [<instance-id>]"

      echo -e "\nThis probe tries to create a volume and (optional) attach it to the input instance, if provided.\nExit code: 0 - probe successfully run; 1 - probe failed"

      exit 1

fi



VOL_ID=$(cinder create 1 | sed -n  's/^|\ \+id\ \+|\ \+\([^\ ].*\)\ \+|/\1/p')


wait_volume_available $VOL_ID
```

```
# check status
if [ $? -ne 0 ]; then
      echo "Error: volume not available yet"
      exit 1
else
      echo "OK. Volume creation was successful (ID=$VOL_ID)"
fi


if [ ! -z $1 ]; then
      echo Trying to attach volume $VOL_ID to instance $1
      nova volume-attach $1 $VOL_ID | grep -q -s device && echo "OK" || (echo ERROR && exit 1)


      wait_volume_in_use $VOL_ID


      if [ $? -ne 0 ]; then
            echo ERROR while attaching volume
      else
            echo Volume successfully attached
```

```
      fi


      echo Trying to detach volume $VOL_ID from instance $1

      nova volume-detach $1 $VOL_ID && echo "OK" || (echo ERROR && exit 1)


fi


echo Trying to delete volume

cinder delete $VOL_ID && echo "OK" || (echo ERROR && exit 1)
```

4. Failover/Recovery Tests

Assumption: the Openstack infrastructure has been set-up implementing the high-availability as described in [R2].

4.1 Verify the Controller node failover

To verify that controller node HA is working, simply shut down the node hosting the controller services: you can power-off the host or you can simulate the host failure, as described below.

You should see the controller services group start on the other node.

1. Check the cluster status using the command "`crm status`":

```
root@controller-01:~# crm status
============
Last updated: Tue Dec 23 21:53:07 2014
Last change: Tue Dec 16 16:16:14 2014 via crmd on controller-02
Stack: openais
Current DC: controller-01 - partition with quorum
Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c
3 Nodes configured, 3 expected votes
23 Resources configured.
```

```
============


Online: [ controller-01 controller-02 controller-03 ]


 Resource Group: g_controller
     controller_p_vip   (ocf::heartbeat:IPaddr2):     Started controller-01
     controller_pub_vip (ocf::heartbeat:IPaddr2):     Started controller-01
     p_apache     (ocf::heartbeat:apache):      Started controller-01
     p_glance-registry  (ocf::openstack:glance-registry):   Started controller-01
     p_glance-api (ocf::openstack:glance-api):  Started controller-01
     p_nova_api   (ocf::openstack:nova-api):    Started controller-01
     p_nova_cert  (ocf::openstack:nova-cert):   Started controller-01
     p_nova_consoleauth (ocf::openstack:nova-consoleauth):  Started controller-01
     p_nova_novnc (ocf::openstack:nova-novnc):  Started controller-01
     p_nova_scheduler   (ocf::openstack:nova-scheduler):    Started controller-01
     p_cinder-api (ocf::openstack:cinder-api):  Started controller-01
     p_cinder-scheduler (ocf::openstack:cinder-schedule):   Started controller-01
 Resource Group: g_network
```

```
    network_p_vip      (ocf::heartbeat:IPaddr2):     Started controller-03
    network_pub_vip    (ocf::heartbeat:IPaddr2):     Started controller-03
    p_neutron-server   (ocf::openstack:neutron-server):    Started controller-03
    p_neutron-agent-l3 (ocf::openstack:neutron-agent-l3): Started controller-03
    p_neutron-metadata-agent (ocf::openstack:neutron-metadata-agent): Started controller-03
    p_neutron-dhcp-agent     (ocf::openstack:neutron-agent-dhcp):      Started controller-03
```

2. Shutdown the current controller node (controller-01 in our case). You can use the command "`crm node standby`" to put a node in standby mode. Note that any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Therefore, this can be used to simulate a node shutdown.

```
root@controller-01:~# crm node standby controller-01
```

3. You will see the controller services migrating from controller-01 to controller-02:

```
root@controller-01:~# crm status


============

Last updated: Tue Dec 23 22:14:13 2014

Last change: Tue Dec 23 22:14:11 2014 via crm_attribute on controller-01
```

```
Stack: openais

Current DC: controller-02 - partition with quorum

Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c

3 Nodes configured, 3 expected votes

23 Resources configured.

============


Node controller-01: standby

Online: [ controller-02 controller-03 ]


 Resource Group: g_controller

     controller_p_vip   (ocf::heartbeat:IPaddr2):     Started controller-02

     controller_pub_vip (ocf::heartbeat:IPaddr2):     Started controller-02

     p_apache     (ocf::heartbeat:apache):      Started controller-02

     p_glance-registry  (ocf::openstack:glance-registry):   Started controller-02

     p_glance-api (ocf::openstack:glance-api):  Started controller-02

     p_nova_api   (ocf::openstack:nova-api):    Started controller-02

     p_nova_cert  (ocf::openstack:nova-cert):   Started controller-02
```

```
    p_nova_consoleauth (ocf::openstack:nova-consoleauth):  Started controller-02
    p_nova_novnc (ocf::openstack:nova-novnc):  Started controller-02
    p_nova_scheduler   (ocf::openstack:nova-scheduler):    Started controller-02
    p_cinder-api (ocf::openstack:cinder-api):  Started controller-02
    p_cinder-scheduler (ocf::openstack:cinder-schedule):   Started controller-02
Resource Group: g_network
    network_p_vip      (ocf::heartbeat:IPaddr2):     Started controller-03
    network_pub_vip    (ocf::heartbeat:IPaddr2):     Started controller-03
    p_neutron-server   (ocf::openstack:neutron-server):    Started controller-03
    p_neutron-agent-l3 (ocf::openstack:neutron-agent-l3):  Started controller-03
    p_neutron-metadata-agent (ocf::openstack:neutron-metadata-agent):  Started controller-03
    p_neutron-dhcp-agent     (ocf::openstack:neutron-agent-dhcp):      Started controller-03
```

4. Verify that all the services are active and running properly. Re-run the sanity checks described in section 2 and the functional tests in section 3.
5. Restore the node by removing the standby attribute from it: the node will become a fully active member of the cluster again:

```
root@controller-01:~# crm node online controller-01
```

4.2 Verify the Network node failover

To verify that network node HA is working, simply shut down the node hosting the network node services: you can power-off the host or you can simulate the node shutdown .

You should see the network node services group start on the other node.

1. Check the cluster status using the command “`crm status`”:

```
root@controller-01:~# crm status
============
Last updated: Tue Dec 23 22:44:07 2014
Stack: openais
Current DC: controller-01 - partition with quorum
Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c
3 Nodes configured, 3 expected votes
23 Resources configured.
============


Online: [ controller-01 controller-02 controller-03 ]
```

```
Resource Group: g_controller
    controller_p_vip   (ocf::heartbeat:IPaddr2):     Started controller-01
    controller_pub_vip (ocf::heartbeat:IPaddr2):     Started controller-01
    p_apache     (ocf::heartbeat:apache):      Started controller-01
    p_glance-registry  (ocf::openstack:glance-registry):   Started controller-01
    p_glance-api (ocf::openstack:glance-api):  Started controller-01
    p_nova_api   (ocf::openstack:nova-api):    Started controller-01
    p_nova_cert  (ocf::openstack:nova-cert):   Started controller-01
    p_nova_consoleauth (ocf::openstack:nova-consoleauth):  Started controller-01
    p_nova_novnc (ocf::openstack:nova-novnc):  Started controller-01
    p_nova_scheduler   (ocf::openstack:nova-scheduler):    Started controller-01
    p_cinder-api (ocf::openstack:cinder-api):  Started controller-01
    p_cinder-scheduler (ocf::openstack:cinder-schedule):   Started controller-01
Resource Group: g_network
    network_p_vip      (ocf::heartbeat:IPaddr2):     Started controller-03
    network_pub_vip    (ocf::heartbeat:IPaddr2):     Started controller-03
    p_neutron-server   (ocf::openstack:neutron-server):    Started controller-03
    p_neutron-agent-l3 (ocf::openstack:neutron-agent-l3):  Started controller-03
```

```
    p_neutron-metadata-agent (ocf::openstack:neutron-metadata-agent):  Started controller-03
    p_neutron-dhcp-agent     (ocf::openstack:neutron-agent-dhcp):      Started controller-03
```

6. Shutdown the current controller node (controller-03 in our case). You can use the command "`crm node standby`" to put a node in standby mode. Note that any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Therefore, this can be used to simulate the node failure as described below.

```
root@controller-01:~# crm node standby controller-03
```

7. You will see the network node services migrating from controller-03 to controller-02:

```
root@controller-01:~# crm status


============
Last updated: Tue Dec 23 22:45:13 2014
Stack: openais
Current DC: controller-02 - partition with quorum
Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c
3 Nodes configured, 3 expected votes
23 Resources configured.
```

```
============

Node controller-03: standby
Online: [ controller-02 controller-01 ]

 Resource Group: g_controller
     controller_p_vip   (ocf::heartbeat:IPaddr2):     Started controller-01
     controller_pub_vip (ocf::heartbeat:IPaddr2):     Started controller-01
     p_apache     (ocf::heartbeat:apache):      Started controller-01
     p_glance-registry  (ocf::openstack:glance-registry):   Started controller-01
     p_glance-api (ocf::openstack:glance-api):  Started controller-01
     p_nova_api   (ocf::openstack:nova-api):    Started controller-01
     p_nova_cert  (ocf::openstack:nova-cert):   Started controller-01
     p_nova_consoleauth (ocf::openstack:nova-consoleauth):  Started controller-01
     p_nova_novnc (ocf::openstack:nova-novnc):  Started controller-01
     p_nova_scheduler   (ocf::openstack:nova-scheduler):    Started controller-01
     p_cinder-api (ocf::openstack:cinder-api):  Started controller-01
     p_cinder-scheduler (ocf::openstack:cinder-schedule):   Started controller-01
```

```
Resource Group: g_network
    network_p_vip      (ocf::heartbeat:IPaddr2):     Started controller-02
    network_pub_vip    (ocf::heartbeat:IPaddr2):     Started controller-02
    p_neutron-server   (ocf::openstack:neutron-server):    Started controller-02
    p_neutron-agent-l3 (ocf::openstack:neutron-agent-l3):  Started controller-02
    p_neutron-metadata-agent (ocf::openstack:neutron-metadata-agent):  Started controller-02
    p_neutron-dhcp-agent     (ocf::openstack:neutron-agent-dhcp):      Started controller-02
```

8. Verify that all the services are active and running properly. Re-run the sanity checks described in section 2 and the functional tests in section 3.

9. Restore the node by removing the standby attribute from it: the node will become a fully active member of the cluster again:

```
root@controller-01:~# crm node online controller-03
```