OpenStack Services HA test specification

maxdepth:

Scope

The HA test area evaluates the ability of the System Under Test to support service continuity and recovery from component failures on part of OpenStack controller services(“nova-api”, “neutron-server”, “keystone”, “glance-api”, “cinder-api”) and on “load balancer” service.

The tests in this test area will emulate component failures by killing the processes of above target services, stressing the CPU load or blocking disk I/O on the selected controller node, and then check if the impacted services are still available and the killed processes are recovered on the selected controller node within a given time interval.

References

This test area references the following specifications:

- ETSI GS NFV-REL 001
  - http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_nfv-rel001v010101p.pdf
- OpenStack High Availability Guide
  - https://docs.openstack.org/ha-guide/

Definitions and abbreviations

The following terms and abbreviations are used in conjunction with this test area

- SUT - system under test
- Monitor - tools used to measure the service outage time and the process outage time
- Service outage time - the outage time (seconds) of the specific OpenStack service
- Process outage time - the outage time (seconds) from the specific processes being killed to recovered

System Under Test (SUT)

The system under test is assumed to be the NFVi and VIM in operation on a Pharos compliant infrastructure.

SUT is assumed to be in high availability configuration, which typically means more than one controller nodes are in the System Under Test.

Test Area Structure

The HA test area is structured with the following test cases in a sequential manner.

Each test case is able to run independently. Preceding test case’s failure will not affect the subsequent test cases.

Preconditions of each test case will be described in the following test descriptions.

Test Descriptions

Test Case 1 - Controller node OpenStack service down - nova-api

Short name

opnfv.ha.tc001.nova-api_service_down

Use case specification

This test case verifies the service continuity capability in the face of the software process failure. It kills the processes of OpenStack “nova-api” service on the selected controller node, then checks whether the “nova-api” service is still available during the failure, by creating a VM then deleting the VM, and checks whether the killed processes are recovered within a given time interval.

Test preconditions

There is more than one controller node, which is providing the “nova-api” service for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for verifying service continuity and recovery

The service continuity and process recovery capabilities of “nova-api” service is evaluated by monitoring service outage time, process outage time, and results of nova operations.

Service outage time is measured by continuously executing “openstack server list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the “nova-api” service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is measured by checking the status of “nova-api” processes on the selected controller node. The time of “nova-api” processes being killed to the time of the “nova-api” processes being recovered is the process outage time. Process recovery is verified by checking the existence of “nova-api” processes.

All nova operations are carried out correctly within a given time interval which suggests that the “nova-api” service is continuously available.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that “nova-api” processes are running on Node1
- Test action 2: Create a image with “openstack image create test-cirros –file cirros-0.3.5-x86_64-disk.img –disk-format qcow2 –container-format bare”
- Test action 3: Execute”openstack flavor create m1.test –id auto –ram 512 –disk 1 –vcpus 1” to create flavor “m1.test”.
- Test action 4: Start two monitors: one for “nova-api” processes and the other for “openstack server list” command. Each monitor will run as an independent process
- Test action 5: Connect to Node1 through SSH, and then kill the “nova-api” processes
- Test action 6: When “openstack server list” returns with no error, calculate the service outage time, and execute command “openstack server create –flavor m1.test –image test-cirros test-instance”
- Test action 7: Continuously Execute “openstack server show test-instance” to check if the status of VM “test-instance” is “Active”
- Test action 8: If VM “test-instance” is “Active”, execute “openstack server delete test-instance”, then execute “openstack server list” to check if the VM is not in the list

- Test action 9: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

The nova operations are carried out in above order and no errors occur.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the process of “nova-api” if they are not running. Delete image with “openstack image delete test-cirros” Delete flavor with “openstack flavor delete m1.test”

Test Case 2 - Controller node OpenStack service down - neutron-server

Short name

opnfv.ha.tc002.neutron-server_service_down

Use case specification

This test verifies the high availability of the “neutron-server” service provided by OpenStack controller nodes. It kills the processes of OpenStack “neutron-server” service on the selected controller node, then checks whether the “neutron-server” service is still available, by creating a network and deleting the network, and checks whether the killed processes are recovered.

Test preconditions

There is more than one controller node, which is providing the “neutron-server” service for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of “neutron-server” service is evaluated by monitoring service outage time, process outage time, and results of neutron operations.

Service outage time is tested by continuously executing “openstack router list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the “neutron-server” service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is tested by checking the status of “neutron-server” processes on the selected controller node. The time of “neutron-server” processes being killed to the time of the “neutron-server” processes being recovered is the process outage time. Process recovery is verified by checking the existence of “neutron-server” processes.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that “neutron-server” processes are running on Node1
- Test action 2: Start two monitors: one for “neutron-server” process and the other for “openstack router list” command. Each monitor will run as an independent process.
- Test action 3: Connect to Node1 through SSH, and then kill the “neutron-server” processes
- Test action 4: When “openstack router list” returns with no error, calculate the service outage time, and execute “openstack network create test-network”
- Test action 5: Continuously executing “openstack network show test-network”, check if the status of “test-network” is “Active”
- Test action 6: If “test-network” is “Active”, execute “openstack network delete test-network”, then execute “openstack network list” to check if the “test-network” is not in the list
- Test action 7: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

The neutron operations are carried out in above order and no errors occur.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the processes of “neutron-server” if they are not running.

Test Case 3 - Controller node OpenStack service down - keystone

Short name

opnfv.ha.tc003.keystone_service_down

Use case specification

This test verifies the high availability of the “keystone” service provided by OpenStack controller nodes. It kills the processes of OpenStack “keystone” service on the selected controller node, then checks whether the “keystone” service is still available by executing command “openstack user list” and whether the killed processes are recovered.

Test preconditions

There is more than one controller node, which is providing the “keystone” service for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of “keystone” service is evaluated by monitoring service outage time and process outage time

Service outage time is tested by continuously executing “openstack user list” command in loop and checking if the response of the command request is reutrned with no failure. When the response fails, the “keystone” service is

considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is tested by checking the status of “keystone” processes on the selected controller node. The time of “keystone” processes being killed to the time of the “keystone” processes being recovered is the process outage time. Process recovery is verified by checking the existence of “keystone” processes.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that “keystone” processes are running on Node1
- Test action 2: Start two monitors: one for “keystone” process and the other for “openstack user list” command. Each monitor will run as an independent process.
- Test action 3: Connect to Node1 through SSH, and then kill the “keystone” processes
- Test action 4: Calculate the service outage time and process outage time
- Test action 5: The test passes if process outage time is less than 20s and service outage time is less than 5s
- Test action 6: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the processes of “keystone” if they are not running.

Test Case 4 - Controller node OpenStack service down - glance-api

Short name

opnfv.ha.tc004.glance-api_service_down

Use case specification

This test verifies the high availability of the “glance-api” service provided by OpenStack controller nodes. It kills the processes of OpenStack “glance-api” service on the selected controller node, then checks whether the “glance-api” service is still available, by creating image and deleting image, and checks whether the killed processes are recovered.

Test preconditions

There is more than one controller node, which is providing the “glance-api” service for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of “glance-api” service is evaluated by monitoring service outage time, process outage time, and results of glance operations.

Service outage time is tested by continuously executing “openstack image list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the “glance-api” service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is tested by checking the status of “glance-api” processes on the selected controller node. The time of “glance-api” processes being killed to the time of the “glance-api” processes being recovered is the process outage time. Process recovery is verified by checking the existence of “glance-api” processes.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that “glance-api” processes are running on Node1
- Test action 2: Start two monitors: one for “glance-api” process and the other for “openstack image list” command. Each monitor will run as an independent process.
- Test action 3: Connect to Node1 through SSH, and then kill the “glance-api” processes
- Test action 4: When “openstack image list” returns with no error, calculate the service outage time, and execute “openstack image create test-image –file cirros-0.3.5-x86_64-disk.img –disk-format qcow2 –container-format bare”
- Test action 5: Continuously execute “openstack image show test-image”, check if status of “test-image” is “active”
- Test action 6: If “test-image” is “active”, execute “openstack image delete test-image”. Then execute “openstack image list” to check if “test-image” is not in the list
- Test action 7: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

The glance operations are carried out in above order and no errors occur.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the processes of “glance-api” if they are not running.

Delete image with “openstack image delete test-image”.

Test Case 5 - Controller node OpenStack service down - cinder-api

Short name

opnfv.ha.tc005.cinder-api_service_down

Use case specification

This test verifies the high availability of the “cinder-api” service provided by OpenStack controller nodes. It kills the processes of OpenStack “cinder-api” service on the selected controller node, then checks whether the “cinder-api” service is still available by executing command “openstack volume list” and whether the killed processes are recovered.

Test preconditions

There is more than one controller node, which is providing the “cinder-api” service for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of “cinder-api” service is evaluated by monitoring service outage time and process outage time

Service outage time is tested by continuously executing “openstack volume list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the “cinder-api” service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is tested by checking the status of “cinder-api” processes on the selected controller node. The time of “cinder-api” processes being killed to the time of the “cinder-api” processes being recovered is the process outage time. Process recovery is verified by checking the existence of “cinder-api” processes.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that “cinder-api” processes are running on Node1
- Test action 2: Start two monitors: one for “cinder-api” process and the other for “openstack volume list” command. Each monitor will run as an independent process.
- Test action 3: Connect to Node1 through SSH, and then execute kill the “cinder-api” processes
- Test action 4: Continuously measure service outage time from the monitor until the service outage time is more than 5s
- Test action 5: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

The cinder operations are carried out in above order and no errors occur.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the processes of “cinder-api” if they are not running.

Test Case 6 - Controller Node CPU Overload High Availability

Short name

opnfv.ha.tc006.cpu_overload

Use case specification

This test verifies the availability of services when one of the controller node suffers from heavy CPU overload. When the CPU usage of the specified controller node is up to 100%, which breaks down the OpenStack services on this node, the Openstack services should continue to be available. This test case stresses the CPU usage of a specific

controller node to 100%, then checks whether all services provided by the SUT are still available with the monitor tools.

Test preconditions

There is more than one controller node, which is providing the “cinder-api”, “neutron-server”, “glance-api” and “keystone” services for API end-point. Denoted a controller node as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of related OpenStack service is evaluated by monitoring service outage time

Service outage time is tested by continuously executing “openstack router list”, “openstack stack list”, “openstack volume list”, “openstack image list” commands in loop and checking if the response of the command request is returned with no failure. When the response fails, the related service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Methodology for stressing CPU usage

To evaluate the high availability of target OpenStack service under heavy CPU load, the test case will first get the number of logical CPU cores on the target controller node by shell command, then use the number to execute ‘dd’ command to continuously copy from /dev/zero and output to /dev/null in loop. The ‘dd’ operation only uses CPU, no I/O operation, which is ideal for stressing the CPU usage.

Since the ‘dd’ command is continuously executed and the CPU usage rate is stressed to 100%, the scheduler will schedule each ‘dd’ command to be processed on a different logical CPU core. Eventually to achieve all logical CPU cores usage rate to 100%.

Test execution

- Test action 1: Start four monitors: one for “openstack image list” command, one for “openstack router list” command, one for “openstack stack list” command and the last one for “openstack volume list” command. Each monitor will run as an independent process.
- Test action 2: Connect to Node1 through SSH, and then stress all logical CPU cores usage rate to 100%
- Test action 3: Continuously measure all the service outage times until they are more than 5s
- Test action 4: Kill the process that stresses the CPU usage

Pass / fail criteria

All the service outage times are less than 5s.

A negative result will be generated if the above is not met in completion.

Post conditions

No impact on the SUT.

Test Case 7 - Controller Node Disk I/O Overload High Availability

Short name

opnfv.ha.tc007.disk_I/O_overload

Use case specification

This test verifies the high availability of control node. When the disk I/O of the specific disk is overload, which breaks down the OpenStack services on this node, the read and write services should continue to be available. This test case blocks the disk I/O of the specific controller node, then checks whether the services that need to read or write the disk of the controller node are available with some monitor tools.

Test preconditions

There is more than one controller node. Denoted a controller node as Node1 in the following configuration. The controller node has at least 20GB free disk space.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of nova service is evaluated by monitoring service outage time

Service availability is tested by continuously executing “openstack flavor list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the related service is considered in outage.

Methodology for stressing disk I/O

To evaluate the high availability of target OpenStack service under heavy I/O load, the test case will execute shell command on the selected controller node to continuously writing 8kb blocks to /test.dbf

Test execution

- Test action 1: Connect to Node1 through SSH, and then stress disk I/O by continuously writing 8kb blocks to /test.dbf
- Test action 2: Start a monitor: for “openstack flavor list” command
- Test action 3: Create a flavor called “test-001”
- Test action 4: Check whether the flavor “test-001” is created
- Test action 5: Continuously measure service outage time from the monitor until the service outage time is more than 5s
- Test action 6: Stop writing to /test.dbf and delete file /test.dbf

Pass / fail criteria

The service outage time is less than 5s.

The nova operations are carried out in above order and no errors occur.

A negative result will be generated if the above is not met in completion.

Post conditions

Delete flavor with “openstack flavor delete test-001”.

Test Case 8 - Controller Load Balance as a Service High Availability

Short name

opnfv.ha.tc008.load_balance_service_down

Use case specification

This test verifies the high availability of “load balancer” service. When the “load balancer” service of a specified controller node is killed, whether “load balancer” service on other controller nodes will work, and whether the controller node will restart the “load balancer” service are checked. This test case kills the processes of “load balancer” service on the selected controller node, then checks whether the request of the related OpenStack command is processed with no failure and whether the killed processes are recovered.

Test preconditions

There is more than one controller node, which is providing the “load balancer” service for rest-api. Denoted as Node1 in the following configuration.

Basic test flow execution description and pass/fail criteria

Methodology for monitoring high availability

The high availability of “load balancer” service is evaluated by monitoring service outage time and process outage time

Service outage time is tested by continuously executing “openstack image list” command in loop and checking if the response of the command request is returned with no failure. When the response fails, the “load balancer” service is considered in outage. The time between the first response failure and the last response failure is considered as service outage time.

Process outage time is tested by checking the status of processes of “load balancer” service on the selected controller node. The time of those processes being killed to the time of those processes being recovered is the process outage time. Process recovery is verified by checking the existence of processes of “load balancer” service.

Test execution

- Test action 1: Connect to Node1 through SSH, and check that processes of “load balancer” service are running on Node1
- Test action 2: Start two monitors: one for processes of “load balancer” service and the other for “openstack image list” command. Each monitor will run as an independent process
- Test action 3: Connect to Node1 through SSH, and then kill the processes of “load balancer” service
- Test action 4: Continuously measure service outage time from the monitor until the service outage time is more than 5s
- Test action 5: Continuously measure process outage time from the monitor until the process outage time is more than 30s

Pass / fail criteria

The process outage time is less than 30s.

The service outage time is less than 5s.

A negative result will be generated if the above is not met in completion.

Post conditions

Restart the processes of “load balancer” if they are not running.