HA Deployment Framework Guideline

This document will provide an overall framework for the high availability deployment of NFV. It will also continiously update to include HA deployment guidelines and suggestions for the releases of OPNFV.

1. Overview of High Available Deployment of OPNFV

In this section, we would like to discuss the overall HA deployment of NFV. Different moduels will be included, and HA deployment of each single module will be discussed. However, not all of these HA schemes should be deployed in on system at the same time. For the HA deployment of a single system, we should consider the tradeoff between high availability and the cost and resource to leverage.

1.1 Architecture of HA deployment

This section intends to introduce the different modules we should consider when talking about HA deployment. These moduels include the Hardware (compute, network, storage hardware), the VIM, the hypervisor, VMs and VNFs. HA schemes for these different moduels should all be considered when deploying an NFV system. And the schemes should be coordinated so that the system can make sure to react in its best way when facing failure.

The following picture shows the the architecture of HA deployment based on the framework from ETSI NFV ISG.

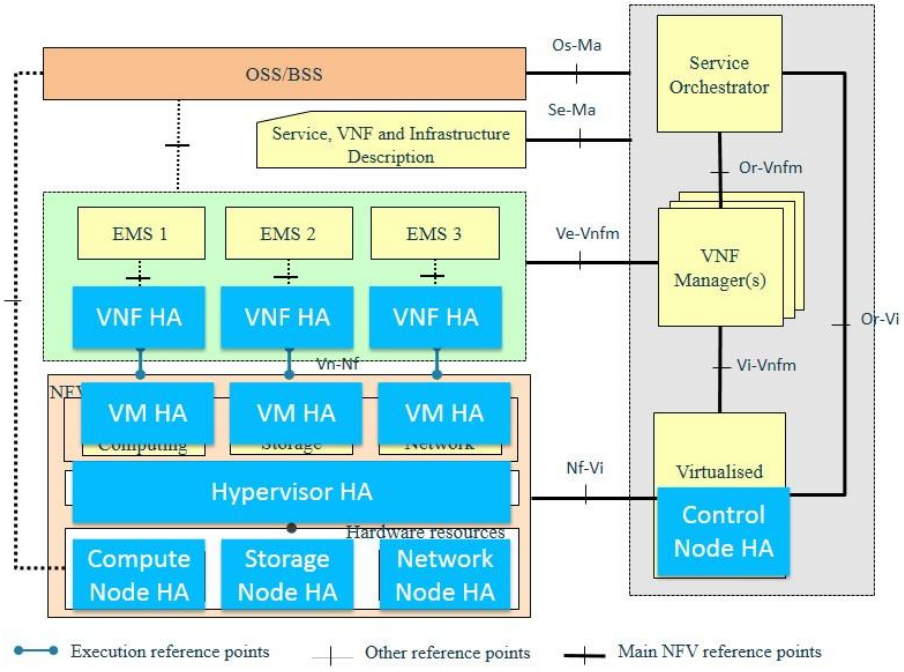


Fig 1. Architecture of HA Deployment based on the Framework of ETSI NFV ISG

1.2 HA deployment topology

This section will introduce the HA deployment topology for an NFV system. The topology is based on typical OPNFV deployment, where openstack is used for VIM. Such topology might be different if alternative VIM is used for the system. However the concept of HA deployment in the hardware level can be the same. The topology explained in this section is to support the software cluster of OPNFV platform, which we will discuss in detail in section1.3.

The topology of a high available deployment OPNFV platform should include at least the cotroller nodes, and the compute nodes. Depend on the request of the users, standalone network nodes or storage nodes can be added into this topology. The simplest HA deployment of OPNFV only include the control nodes and the compute nodes. Figure 2 shows the deployment topology, in which the controller nodes are all in a cluster, and the compute nodes can be in one cluster.

The control node cluster here is to provide HA for the controller services, so that the services on the control node can successfully failover when failure happens and the

service can continue. The cluster service should also provide automatic recovery for the control nodes. For OPNFV, the control node cluster should include at least 3 nodes, and should be an odd number if the cluster management system use quorum. This may change if we use different cluster management schemes though.

The compute node clusters is responsible for providing HA for the services running on the compute nodes. These services may include agents for openstack, host os, hypervisors. Such cluster is responsible for the recovery and repair of the services. However, compute node cluster will certainly bring complexity to the whole system, and would increase the cost. There could be multiple solutions for the compute cluster, e.g., senlin from openstack.

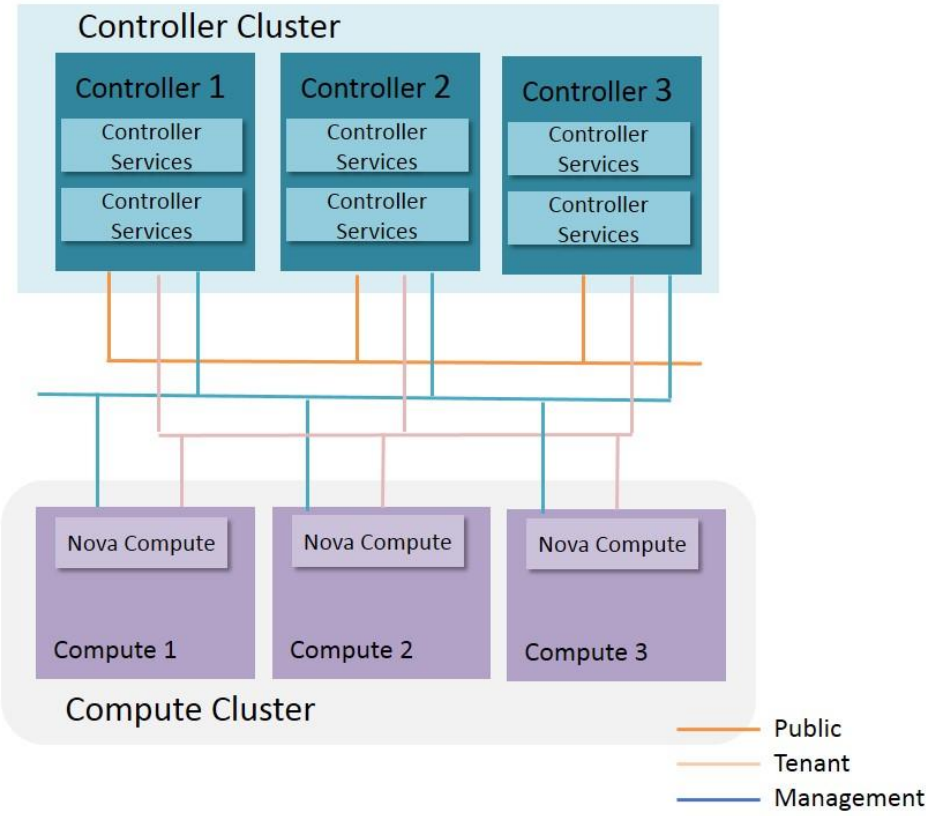


Fig 2. HA Deployment Topology of Control Nodes and Compute Nodes

When the cloud is supporting heavy network traffic, it is necessary to deploy standalone network nodes for openstack. In figure 3, we add network nodes into the topology and shows how to deploy it in a high available way. In this figure, the network nodes are deployed in a cluster. The cluster will provide HA for the services runing on the network nodes.

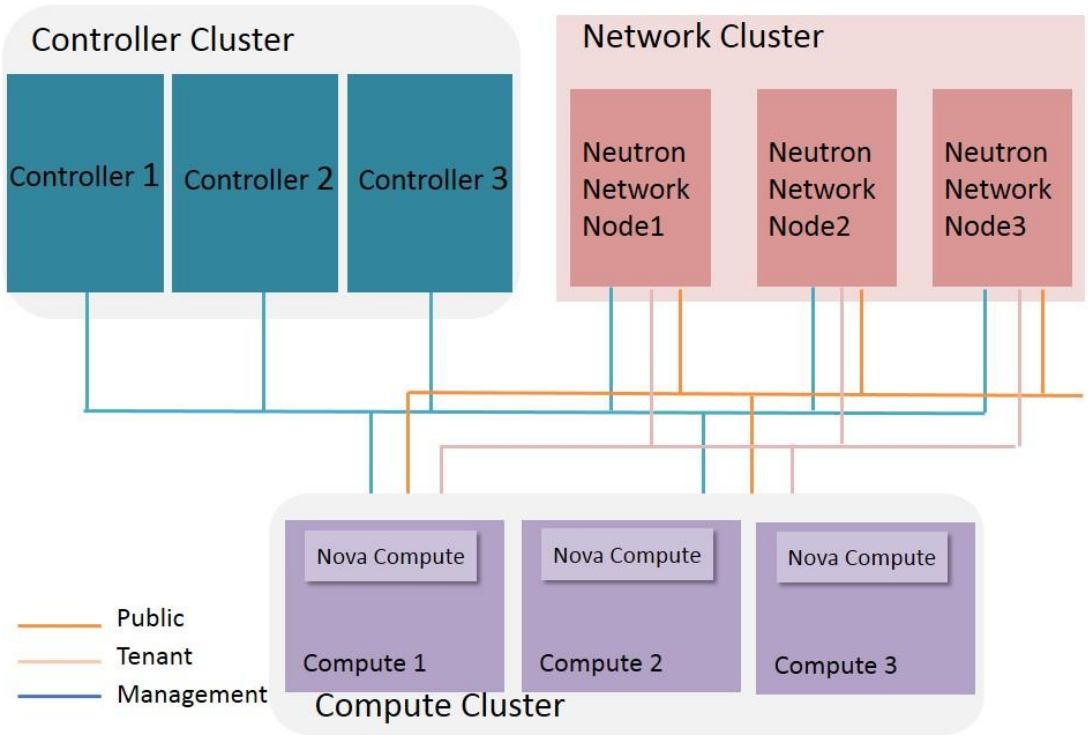


Fig 3. HA Deployment Topology of Control Nodes, Compute Nodes and network Nodes

When the cloud is supporting services that requires large amount of storage, standalone storage nodes are necessary for the deployment. In figure 4, we add storage nodes into the topology. A cluster is also used for the HA pretection of the storage nodes.

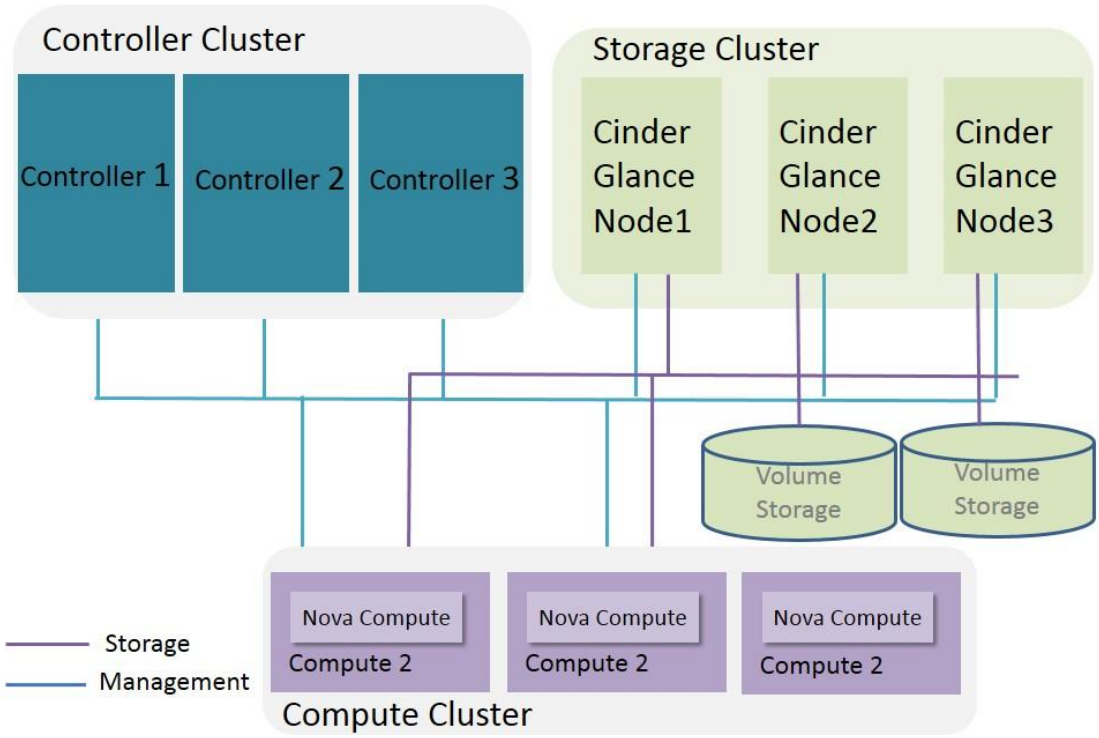


Fig 4. HA Deployment Topology of Control Nodes, Compute Nodes, network Nodes and Storage Nodes

1.3 Software HA Framework

In this section, we would like to introduce more details about the HA schemes for a complete NFV system.

1.3.1 Openstack Controller services (Openstack services)

At its core, a cluster is a distributed finite state machine capable of co-ordinating the startup and recovery of inter-related services across a set of machines. For OpenStack Controller nodes, a cluster management system, such as Pacemaker, is recommended to use to provide the following metrics.

1, Awareness of other applications in the stack

2, Awareness of instances on other machines

3, A shared implementation and calculation of quorum.

4, Data integrity through fencing (a non-responsive process does not imply it is not doing anything)

5, Automated recovery of failed instances

Figure 5 shows the details of HA schemes for Openstack controller nodes with Pacemaker.

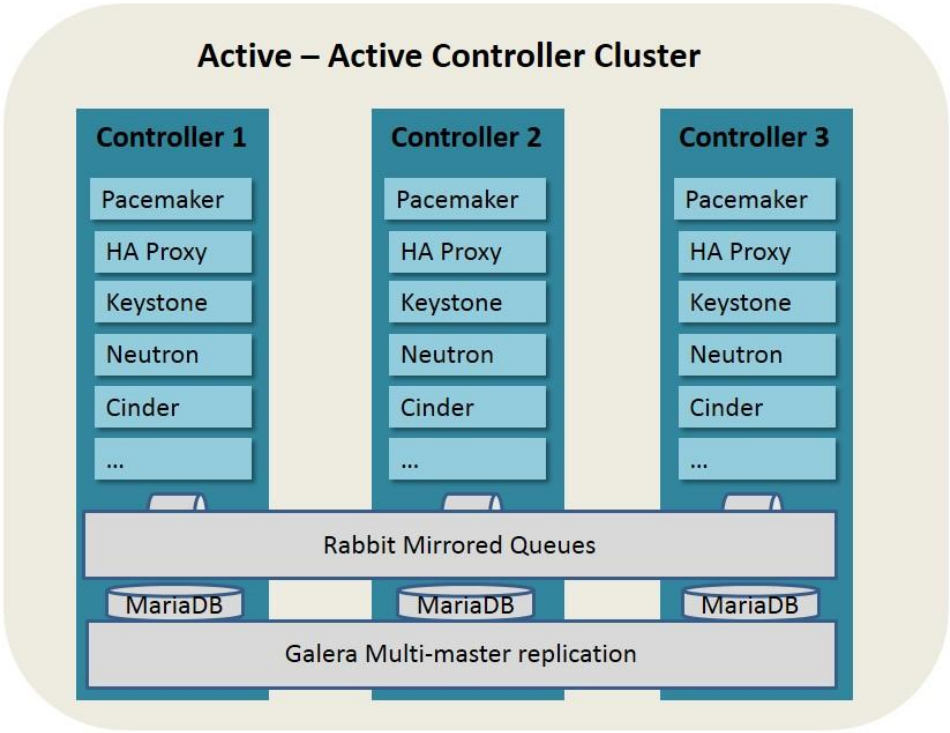


Fig 5. HA Deployment of Openstack Control Nodes based on Pacemaker

High availability of all stateless services are provided by pacemaker and HAProxy.

Pacemaker cluster stack is the state-of-the-art high availability and load balancing stack for the Linux platform. Pacemaker is useful to make OpenStack infrastructure highly available. Also, it is storage and application-agnostic, and in no way specific to OpenStack.

Pacemaker relies on the Corosync messaging layer for reliable cluster communications. Corosync implements the Totem single-ring ordering and membership protocol. It also provides UDP and InfiniBand based messaging, quorum, and cluster membership to Pacemaker.

Pacemaker does not inherently (need or want to) understand the applications it manages. Instead, it relies on resource agents (RAs), scripts that encapsulate the knowledge of how to start, stop, and check the health of each application managed by the cluster.These agents must conform to one of the OCF, SysV Init, Upstart, or Systemd standards.Pacemaker ships with a large set of OCF agents (such as those managing MySQL databases, virtual IP addresses, and RabbitMQ), but can also use any agents already installed on your system and can be extended with your own (see the developer guide).

After deployment of Pacemaker, HAProxy is used to provide VIP for all the OpenStack services and act as load balancer. HAProxy provides a fast and reliable HTTP reverse proxy and load balancer for TCP or HTTP applications. It is particularly suited for web crawling under very high loads while needing persistence or Layer 7 processing. It realistically supports tens of thousands of connections with recent hardware.

Each instance of HAProxy configures its front end to accept connections only from the virtual IP (VIP) address and to terminate them as a list of all instances of the corresponding service under load balancing, such as any OpenStack API service. This makes the instances of HAProxy act independently and fail over transparently together with the network endpoints (VIP addresses) failover and, therefore, shares the same SLA.

We can alternatively use a commercial load balancer, which is a hardware or software. A hardware load balancer generally has good performance.

Galera Cluster, or other database cluster service, should also be deployed to provide data replication and synchronization between data base. Galera Cluster is a synchronous multi-master database cluster, based on MySQL and the InnoDB storage engine. It is a high-availability service that provides high system uptime, no data loss, and scalability for growth. The selection of DB also will have potential influence on the behaviour on the application code. For instance using Galera Clusterl may give you higher concurrent write perfomance but may require a more complex conflict resolution.

We can also achieve high availability for the OpenStack database in many different ways, depending on the type of database that we are using. There are three implementations of Galera Cluster available:

1, Galera Cluster for MySQL The MySQL reference implementation from Codership;

2, MariaDB Galera Cluster The MariaDB implementation of Galera Cluster, which is commonly supported in environments based on Red Hat distributions;

3, Percona XtraDB Cluster The XtraDB implementation of Galera Cluster from Percona.

In addition to Galera Cluster, we can also achieve high availability through other database options, such as PostgreSQL, which has its own replication system.

To make the RabbitMQ high available, Rabbit HA queue should be configued, and all openstack services should be configurd to use the Rabbit HA queue.

In the meantime, specific schemes should also be provided to avoid single point of failure of Pacemaker. And services failed should be automaticly repaired.

Note that the schemes we discribed above is just one possible scheme for the HA deployment of the controller nodes. Other schemes can also be used to provide cluster management and monitoring.

1.3.2 SDN controller services

SDN controller software is data intensive application. All static and dynamic data has one or more duplicates distributed to other physical nodes in cluster. Build-in HA schema always be concordant with data distribution and build-in mechanism will select or re-select master nodes in cluster. In deployment stage software of SDN controller should be deployed to at least two or more physical nodes regardless whether the software is deployed inside VM or containner. Dural management network plane should be provided for SDN controller cluster to support build-in HA schema.

1.3.3 Host OS and Hypervisor

The Host OS and Hypervisor should be supervised and monitored for failure, and should be repaired when failure happens. Such supervision can based on a cluster scheme, or can just simply use controller to constantly monitor the computer host. Figure 6 shows a simplified framework for hypervisor cluster.

When host/hypervisor failure happens, VMs on that host should be evacuated. However, such scheme should coordinate with the VM HA scheme, so that when both the host and the VM detect the failure, they should know who should take responsibility for the evacuation.

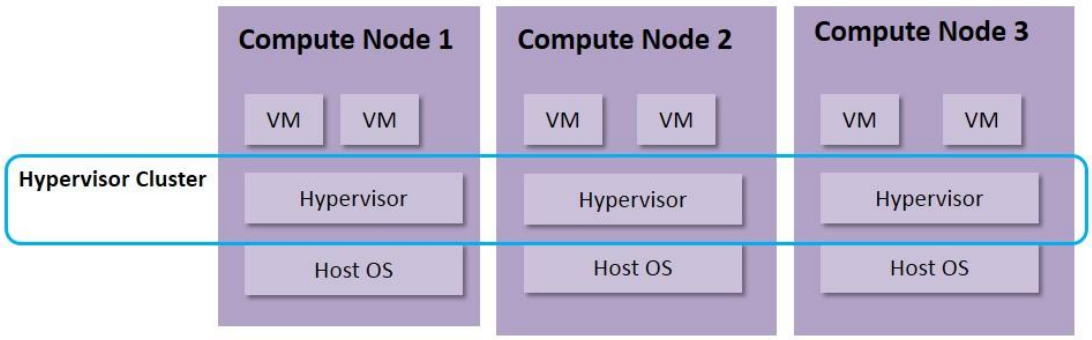


Fig 6. HA Deployment of Host OS and Hypervisor

1.3.4 Virtual Machine (VM)

VM should be supervised and monitored for failure, and should be repaired when failure happens. We can rely on the hypervisor to monitor the VM failure. Another scheme can be used is a cluster for the VM, in which failure of VMs in one cluster can be supervised and will be repaired by the cluster manager. Pacemaker and other cluster management schemes can be considered for the VM cluster.

In case when VNFs do not have HA schemes, extra HA scheme for VM should be taken into consideration. Such approach is kind of best effort for the NFV platform to provide HA for the VNF service, and may lead to failure copy between VMs when VNF fails. Since the NFVI can hardly know of the service runing in the VNF, it is imporssible for the NFVI level to provide overall HA solution for the VNF services. Therefore, even though we mention this scheme here, we strongly suggest the VNF should have its own HA schemes.

Figure 7 gives an example for the VM active/standby deployment. In this case, both the active VM and the standby VM are deployed with the same VNF image. When failure

happens to the active VM, the standby VM should take the traffic and replace the active VM. Such scheme is the best effort of the NFVI when VNFs do not have HA schemes and would only rely on VMs to provide redundancy. However, for stateful VNFs, there should be data copy between the active VM and standby VM. In this case, fault for the active VM can also be copied to the standby VM, leading to failure of the new active VM.

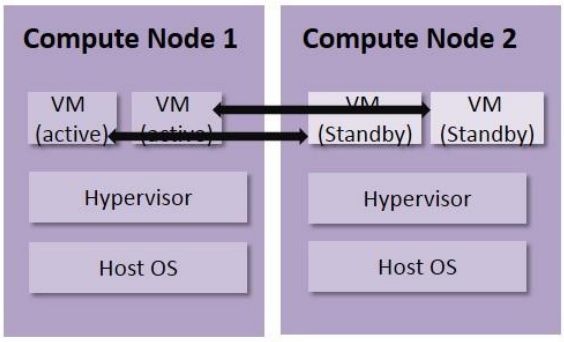


Fig 7. VM Active/Standby Deployment

1.3.5 Virtual Network Functions (VNF)

For telecom services, it is suggested that VNFs should have its own HA schemes to provide high available service to the customers. HA schemes for the VNFs can based on cluster. In this case, OpenSAF, pacemaker and other cluster management services can be used.

HA schemes for the VNFs should be coordinate with the lower layer. For example, it should be clear which level will take responsibility for VM restart. A suggested schemes could be, the VNF layer should be responsible for the redundancy and failover of the VNFs when failure happens. Such failover should take place in quite a short time (less

then seconds). The repairing procedure will then take place from upper layer to lower layer, that is, the VNF layer will first check if the failure is at its layer, and should try to repair itself. If it fails to repaire the failure, the failure should escalate to lower layers and let the NFVI layer to do the repair work. There could also be cases that the NFVI layer has detected the failure and will repair it before the escalation.

In the meantime, the VNFs can take advantage of API the hypervisor can provide to them to enhance HA. Such API may include constant health check from the hypervisor, affinity/inaffinity deployment support. example about watchdog

Figure 8 gives an example for the VNF HA scheme.

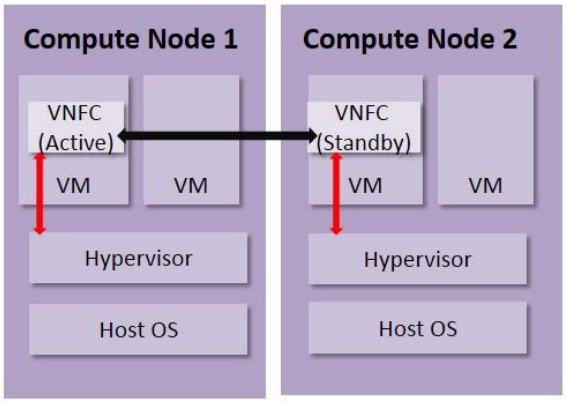


Fig 8. HA Deployment of VNFs

2 HA deployment guideline for OPNFV releases

In this section, we will continiously update the HA deployment guideline for the releases of OPNFV.

2.1 HA deployment guideline for Arno

2.1.1 Deployment Framework

Figure 9 shows a overall architecture for the HA deployment of ARNO.

For OPNFV Arno release, HA deployment of Openstack Control Node (Openstack Juno) and ODL controller (ODL Helium) is supported. Both deployment tools (fuel and forman)support such HA deployment.

For such HA deployment, the following components' failure is protected

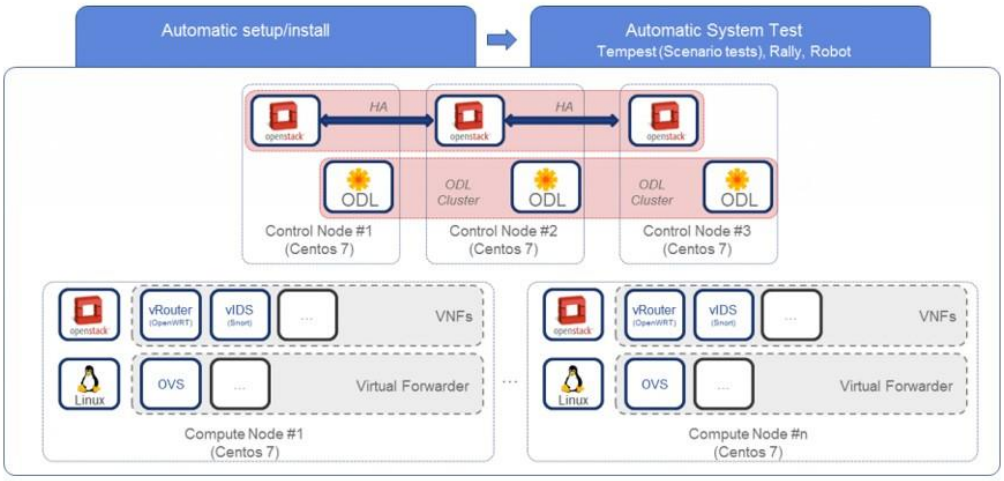
Software:

- Nova scheduler
- Nova conductor
- Cinder scheduler
- Neutron server
- Heat engine

Controller hardware:

- dead server
- dead switch
- dead port
- dead disk
- full disk

Figure 8 gives an example for the VNF HA scheme.



9

Fig 8. HA Deployment of OPNFV ARNO release

2.1.2 HA test result for ARNO

Two specific High Availability testcases are done on the ARNO release. These test cases are collaboratively developed by the High Availability project and the Yardstick project.

Both cases are excuted in the China Mobile's Lab, where ARNO SR1 release is deployed with Fuel.

The two testcases respectively test the following two aspects:

1, Controll Node Service HA

In this test, HA of "nova-api" is tested. According to the result, the service can successfully failover to the other controller nodes within 2.36s, once failure happens at the active node. However, the service can't repair itself automatically. more explaination about the repair, other services are not tested yet.

2, Control Node Hardware HA

In this test, HA of the controller node hardware is tested. One of the hardware is abnormally shutdown, and the service of "nova-api" is monitored. According to the test results, the service can failover to the other controller node within 10.71 secondes. However, the failed hardware can't automatically repair itself.

See more details about these test cases in the Yardstick doc of "Test Results for yardstick-opnfv-ha".

From these basic test cases we can see that OPNFV ARNO has integrated with some HA schemes in its controller nodes. However, its capability of self repair should be enhanced.

2.2 HA deployment guideline for Brahmaputra

In the Brahmaputra release, 4 installers are provided. We will discuss about the HA deployment of each installer.

2.2.1 Apex

For the installer of Apex, all of the OpenStack services are in HA on all 3 controllers. The services are monitored by pacemaker and load balanced by HA Proxy with VIPs. The SDN controllers usually only run as a singleinstance on the first controller with no HA scheme.

Database is clustered with galera in an active passive failover via pacemaker and the message bus is rabbitHA and the services are managed by pacemaker.

Storage is using ceph, clustered across the control nodes.

In the future, more work is on the way to provide HA for the SDN controller. The Apex team has already finished a demo that runs ODL on each controller, load balanced to neutron via a VIP + HA Proxy, but is not using pacemaker. Meanwhile, they are also working to include ceph storage HA for compute nodes as well.

2.2.2 Compass

2.2.3 Fuel

At moment Fuel installer support the following HA schemes.

1)Openstackcontrollers: N-way redundant (1,3,5, etc) 2)OpenDaylight:No redundancy 3)Cephstorage OSD: N-way redundant (1,3,5, etc) 4)Networkingattachment redundancy: LAG 5)NTPredundancy: N-way relays, up to 3 upstream sources 6)DNSredundancy: N-way relays, up to 3 upstream sources 7)DHCP:1+1

2.2.4 JOID

JOID provides HA based on openstack services. Individual service charms have been deployed in a container within a host, and each charms are distributed in a way each service which meant for HA will go into container on individual nodes. For example keystone service, there are three containers on each control node and VIP has been assigned to use by the front end API to use keystone. So in case any of the container fails VIP will keep responding to via the other two services. As HA can be maintainer with odd units at least one service container is required to response.

Reference

- https://www.rdoproject.org/ha/ha-architecture/