APEX

Release Latest

May 09, 2019

Contents

CHAPTER 1

os-nosdn-nofeature-noha overview and description

This document provides scenario level details for Gambia 1.1 of deployment with no SDN controller and no extra features enabled.

1.1 Introduction

This scenario is used primarily to validate and deploy a Queens OpenStack deployment without any NFV features or SDN controller enabled.

1.1.1 Scenario components and composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

1.1.2 Scenario usage overview

Simply deploy this scenario by using the os-nosdn-nofeature-noha.yaml deploy settings file.

1.1.3 Limitations, Issues and Workarounds

None

1.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 2

os-ovn-nofeature-noha overview and description

This document provides scenario level details for Gambia 1.1 of deployment with the OVN SDN controller and no extra features enabled.

2.1 Introduction

This scenario is used primarily to validate and deploy a Pike OpenStack deployment with the OVN SDN controller, and without any NFV features enabled.

2.1.1 Scenario components and composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

2.1.2 Scenario usage overview

Simply deploy this scenario by using the os-ovn-nofeature-ha.yaml deploy settings file.

2.1.3 Limitations, Issues and Workarounds

None

2.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 3

os-nosdn-nofeature-ha overview and description

This document provides scenario level details for Gambia 1.1 of deployment with no SDN controller and no extra features enabled.

3.1 Introduction

This scenario is used primarily to validate and deploy a Queens OpenStack deployment without any NFV features or SDN controller enabled.

3.1.1 Scenario components and composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

All services are in HA, meaning that there are multiple cloned instances of each service, and they are balanced by HA Proxy using a Virtual IP Address per service.

3.1.2 Scenario usage overview

Simply deploy this scenario by using the os-nosdn-nofeature-ha.yaml deploy settings file.

3.1.3 Limitations, Issues and Workarounds

None

3.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 4

os-odl-nofeature-noha overview and description

This document provides scenario level details for Gambia 1.1 of deployment with the OpenDaylight SDN controller and no extra features enabled.

4.1 Introduction

This scenario is used primarily to validate and deploy a Queens OpenStack deployment with OpenDaylight, and without any NFV features enabled.

4.1.1 Scenario components and composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

Only a single controller is deployed in this scenario, which also includes the OpenDaylight service on it.

4.1.2 Scenario usage overview

Simply deploy this scenario by using the os-odl-nofeature-noha.yaml deploy settings file.

4.1.3 Limitations, Issues and Workarounds

- **APEX-268:** VMs with multiple floating IPs can only access via first NIC

4.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 5

os-odl-nofeature-ha overview and description

This document provides scenario level details for Gambia 1.1 of deployment with the OpenDaylight SDN controller and no extra features enabled.

5.1 Introduction

This scenario is used primarily to validate and deploy a Queens OpenStack deployment with OpenDaylight, and without any NFV features enabled.

5.1.1 Scenario components and composition

This scenario is composed of common OpenStack services enabled by default, including Nova, Neutron, Glance, Cinder, Keystone, Horizon. Optionally and by default, Tacker and Congress services are also enabled. Ceph is used as the backend storage to Cinder on all deployed nodes.

All services are in HA, meaning that there are multiple cloned instances of each service, and they are balanced by HA Proxy using a Virtual IP Address per service.

OpenDaylight is also enabled in HA, and forms a cluster. Neutron communicates with a Virtual IP Address for OpenDaylight which is load balanced across the OpenDaylight cluster. Every Open vSwitch node is connected to every OpenDaylight for High Availability.

5.1.2 Scenario usage overview

Simply deploy this scenario by using the os-odl-nofeature-ha.yaml deploy settings file.

5.1.3 Limitations, Issues and Workarounds

- **APEX-268:** VMs with multiple floating IPs can only access via first NIC

5.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 6

k8s-nosdn-nofeature-noha overview and description

This document provides scenario level details for Gambia 1.1 of Kubernetes deployment with no SDN controller, no extra features and no High Availability enabled. Note this scenario is *not* supported for Gambia initial release and will be supported in a later service release of Gambia.

6.1 Introduction

This scenario is used primarily to validate and deploy a Kubernetes deployment without any NFV features or SDN controller enabled.

6.1.1 Scenario components and composition

This scenario deploys a Kubernetes cluster on bare metal or virtual environment with a single master node. TripleO is used to bootstrap all the nodes and set up basic services like SSH. An undercloud VM used similarly to Openstack deployments, however no Openstack services (Nova, Neutron, Keystone, etc) will be deployed to the nodes. After TripleO successfully executes all the bootstrapping tasks, Kubespray is run (using ansible) to deploy Kubernetes cluster on the nodes.

6.1.2 Scenario usage overview

Simply deploy this scenario by using the k8s-nosdn-nofeature-noha.yaml deploy settings file.

6.1.3 Limitations, Issues and Workarounds

None

6.1.4 References

For more information on the OPNFV Gambia release, please visit http://www.opnfv.org/gambia

CHAPTER 7

OPNFV Installation instructions (Apex)

Contents:

7.1 Abstract

This document describes how to install the Gambia release of OPNFV when using Apex as a deployment tool covering it's limitations, dependencies and required system resources.

7.2 License

7.3 Introduction

This document describes the steps to install an OPNFV Gambia reference platform using the Apex installer.

The audience is assumed to have a good background in networking and Linux administration.

7.4 Preface

Apex uses Triple-O from the RDO Project OpenStack distribution as a provisioning tool. The Triple-O image based life cycle installation tool provisions an OPNFV Target System (1 or 3 controllers, 0 or more compute nodes) with OPNFV specific configuration provided by the Apex deployment tool chain.

The Apex deployment artifacts contain the necessary tools to deploy and configure an OPNFV target system using the Apex deployment toolchain. The Apex artifact is a python package capable of automating the installation of TripleO and other OPNFV components.

An OPNFV install requires a "Jump Host" in order to operate. It is required to install CentOS 7 release to the Jump Host for traditional deployment, which includes the required packages needed to run `opnfv-deploy`. If you already have a Jump Host with CentOS 7 installed, you may choose to skip the ISO step and simply install the (`python34-opnfv-apex*.rpm`) RPM.

`opnfv-deploy` instantiates a Triple-O Undercloud VM server using libvirt as its provider. This VM is then configured and used to provision the OPNFV target deployment. These nodes can be either virtual or bare metal. This guide contains instructions for installing either method.

7.5 Triple-O Deployment Architecture

Apex is based on the OpenStack Triple-O project as distributed by the RDO Project. It is important to understand the basics of a Triple-O deployment to help make decisions that will assist in successfully deploying OPNFV.

Triple-O stands for OpenStack On OpenStack. This means that OpenStack will be used to install OpenStack. The target OPNFV deployment is an OpenStack cloud with NFV features built-in that will be deployed by a smaller all-in-one deployment of OpenStack. In this deployment methodology there are two OpenStack installations. They are referred to as the undercloud and the overcloud. The undercloud is used to deploy the overcloud.

The undercloud is the all-in-one installation of OpenStack that includes baremetal provisioning capability. The undercloud will be deployed as a virtual machine on a Jump Host.

The overcloud is OPNFV. Configuration will be passed into undercloud and the undercloud will use OpenStack's orchestration component, named Heat, to execute a deployment that will provision the target OPNFV nodes.

7.6 Apex High Availability Architecture

7.6.1 Undercloud

The undercloud is not Highly Available. End users do not depend on the undercloud. It is only for management purposes.

7.6.2 Overcloud

Apex will deploy three control nodes in an HA deployment. Each of these nodes will run the following services:

- Stateless OpenStack services
- MariaDB / Galera
- RabbitMQ
- OpenDaylight
- HA Proxy
- Pacemaker & VIPs
- Ceph Monitors and OSDs

Stateless OpenStack services All running stateless OpenStack services are load balanced by HA Proxy. Pacemaker monitors the services and ensures that they are running.

Stateful OpenStack services All running stateful OpenStack services are load balanced by HA Proxy. They are monitored by pacemaker in an active/passive failover configuration.

MariaDB / Galera The MariaDB database is replicated across the control nodes using Galera. Pacemaker is responsible for a proper start up of the Galera cluster. HA Proxy provides and active/passive failover methodology to connections to the database.

RabbitMQ The message bus is managed by Pacemaker to ensure proper start up and establishment of clustering across cluster members.

OpenDaylight OpenDaylight is currently installed on all three control nodes and started as an HA cluster unless otherwise noted for that scenario. OpenDaylight's database, known as MD-SAL, breaks up pieces of the database into "shards". Each shard will have its own election take place, which will determine which OpenDaylight node is the leader for that shard. The other OpenDaylight nodes in the cluster will be in standby. Every Open vSwitch node connects to every OpenDaylight to enable HA.

HA Proxy HA Proxy is monitored by Pacemaker to ensure it is running across all nodes and available to balance connections.

Pacemaker & VIPs Pacemaker has relationships and restraints setup to ensure proper service start up order and Virtual IPs associated with specific services are running on the proper host.

Ceph Monitors & OSDs The Ceph monitors run on each of the control nodes. Each control node also has a Ceph OSD running on it. By default the OSDs use an autogenerated virtual disk as their target device. A non-autogenerated device can be specified in the deploy file.

VM Migration is configured and VMs can be evacuated as needed or as invoked by tools such as heat as part of a monitored stack deployment in the overcloud.

7.7 OPNFV Scenario Architecture

OPNFV distinguishes different types of SDN controllers, deployment options, and features into "scenarios". These scenarios are universal across all OPNFV installers, although some may or may not be supported by each installer.

The standard naming convention for a scenario is: <VIM platform>-<SDN type>-<feature>-<ha/noha>

The only supported VIM type is "OS" (OpenStack), while SDN types can be any supported SDN controller. "feature" includes things like ovs_dpdk, sfc, etc. "ha" or "noha" determines if the deployment will be highly available. If "ha" is used at least 3 control nodes are required.

7.8 OPNFV Scenarios in Apex

Apex provides pre-built scenario files in /etc/opnfv-apex which a user can select from to deploy the desired scenario. Simply pass the desired file to the installer as a (-d) deploy setting. Read further in the Apex documentation to learn more about invoking the deploy command. Below is quick reference matrix for OPNFV scenarios supported in Apex. Please refer to the respective OPNFV Docs documentation for each scenario in order to see a full scenario description. Also, please refer to release notes for information about known issues per scenario. The following scenarios correspond to a supported <Scenario>.yaml deploy settings file:

Scenario	Owner	Supported
os-nosdn-nofeature-ha	Apex	Yes
os-nosdn-nofeature-noha	Apex	Yes
os-nosdn-bar-ha	Barometer	No
os-nosdn-bar-noha	Barometer	No

Continued on next page

Table 1 – continued from previous page

os-nosdn-calipso-noha	Calipso	Yes
os-nosdn-ovs_dpdk-ha	Apex	No
os-nosdn-ovs_dpdk-noha	Apex	No
os-nosdn-fdio-ha	FDS	No
os-nosdn-fdio-noha	FDS	No
os-nosdn-kvm_ovs_dpdk-ha	KVM for NFV	No
os-nosdn-kvm_ovs_dpdk -noha	KVM for NFV	No
os-nosdn-performance-ha	Apex	No
os-odl-nofeature-ha	Apex	Yes
os-odl-nofeature-noha	Apex	Yes
os-odl-ovs_dpdk-ha	Apex	No
os-odl-ovs_dpdk-noha	Apex	No
os-odl-bgpvpn-ha	SDNVPN	Yes
os-odl-bgpvpn-noha	SDNVPN	Yes
os-odl-sriov-ha	Apex	No
os-odl-sriov-noha	Apex	No
os-odl-l2gw-ha	Apex	No
os-odl-l2gw-noha	Apex	No
os-odl-sfc-ha	SFC	Yes
os-odl-sfc-noha	SFC	Yes
os-odl-gluon-noha	Gluon	No
os-odl-csit-noha	Apex	No
os-odl-fdio-ha	FDS	No
os-odl-fdio-noha	FDS	No
os-odl-fdio_dvr-ha	FDS	No
os-odl-fdio_dvr-noha	FDS	No
os-onos-nofeature-ha	ONOSFW	No
os-onos-sfc-ha	ONOSFW	No
os-ovn-nofeature-noha	Apex	No
os-ovn-nofeature-ha	Apex	Yes

7.9 Setup Requirements

7.9.1 Jump Host Requirements

The Jump Host requirements are outlined below:

1. CentOS 7 (self-installed) or Fedora (with Snapshot deployment).
2. Root access.
3. libvirt virtualization support.
4. minimum 1 networks and maximum 5 networks, multiple NIC and/or VLAN combinations are supported. This is virtualized for a VM deployment.
5. The Gambia Apex RPM and its dependencies.
6. 16 GB of RAM for a bare metal deployment, 64 GB of RAM for a Virtual Deployment.

7.9.2 Network Requirements

Network requirements include:

1. No DHCP or TFTP server running on networks used by OPNFV.
2. 1-5 separate networks with connectivity between Jump Host and nodes.
    - Control Plane (Provisioning)
    - Private Tenant-Networking Network*
    - External Network*
    - Storage Network*
    - Internal API Network* (required for IPv6 **)
3. Lights out OOB network access from Jump Host with IPMI node enabled (bare metal deployment only).
4. External network is a routable network from outside the cloud, deployment. The External network is where public internet access would reside if available.

*These networks can be combined with each other or all combined on the Control Plane network.

**Internal API network, by default, is collapsed with provisioning in IPv4 deployments, this is not possible with the current lack of PXE boot support and therefore the API network is required to be its own network in an IPv6 deployment.

7.9.3 Bare Metal Node Requirements

Bare metal nodes require:

1. IPMI enabled on OOB interface for power control.
2. BIOS boot priority should be PXE first then local hard disk.
3. BIOS PXE interface should include Control Plane network mentioned above.

7.9.4 Execution Requirements (Bare Metal Only)

In order to execute a deployment, one must gather the following information:

1. IPMI IP addresses for the nodes.
2. IPMI login information for the nodes (user/pass).
3. MAC address of Control Plane / Provisioning interfaces of the overcloud nodes.

7.10 Installation High-Level Overview - Bare Metal Deployment

The setup presumes that you have 6 or more bare metal servers already setup with network connectivity on at least 1 or more network interfaces for all servers via a TOR switch or other network implementation.

The physical TOR switches are **not** automatically configured from the OPNFV reference platform. All the networks involved in the OPNFV infrastructure as well as the provider networks and the private tenant VLANs needs to be manually configured.

The Jump Host can be installed using the bootable ISO or by using the (`opnfv-apex*.rpm`) RPMs and their dependencies. The Jump Host should then be configured with an IP gateway on its admin or public interface and

configured with a working DNS server. The Jump Host should also have routable access to the lights out network for the overcloud nodes.

`opnfv-deploy` is then executed in order to deploy the undercloud VM and to provision the overcloud nodes. `opnfv-deploy` uses three configuration files in order to know how to install and provision the OPNFV target system. The information gathered under section Execution Requirements (Bare Metal Only) is put into the YAML file `/etc/opnfv-apex/inventory.yaml` configuration file. Deployment options are put into the YAML file `/etc/opnfv-apex/deploy_settings.yaml`. Alternatively there are pre-baked deploy_settings files available in `/etc/opnfv-apex/`. These files are named with the naming convention os-sdn_controller-enabled_feature-[no]ha.yaml. These files can be used in place of the `/etc/opnfv-apex/deploy_settings.yaml` file if one suites your deployment needs. Networking definitions gathered under section Network Requirements are put into the YAML file `/etc/opnfv-apex/network_settings.yaml`. `opnfv-deploy` will boot the undercloud VM and load the target deployment configuration into the provisioning toolchain. This information includes MAC address, IPMI, Networking Environment and OPNFV deployment options.

Once configuration is loaded and the undercloud is configured it will then reboot the overcloud nodes via IPMI. The nodes should already be set to PXE boot first off the admin interface. The nodes will first PXE off of the undercloud PXE server and go through a discovery/introspection process.

Introspection boots off of custom introspection PXE images. These images are designed to look at the properties of the hardware that is being booted and report the properties of it back to the undercloud node.

After introspection the undercloud will execute a Heat Stack Deployment to continue node provisioning and configuration. The nodes will reboot and PXE from the undercloud PXE server again to provision each node using Glance disk images provided by the undercloud. These disk images include all the necessary packages and configuration for an OPNFV deployment to execute. Once the disk images have been written to node's disks the nodes will boot locally and execute cloud-init which will execute the final node configuration. At this point in the deployment, the Heat Stack will complete, and Mistral will takeover the configuration of the nodes. Mistral handles calling Ansible which will connect to each node, and begin configuration. This configuration includes launching the desired OPNFV services as containers, and generating their configuration files. These configuration is largely completed by executing a puppet apply on each container to generate the config files, which are then stored on the overcloud host and mounted into the service container at runtime.

7.11 Installation Guide - Bare Metal Deployment

This section goes step-by-step on how to correctly install and provision the OPNFV target system to bare metal nodes.

7.11.1 Install Bare Metal Jump Host

1a. If your Jump Host does not have CentOS 7 already on it, or you would like to do a fresh install, then download the CentOS 7 DVD and perform a "Virtualization Host" install. If you perform a "Minimal Install" or install type other than "Virtualization Host" simply run `sudo yum -y groupinstall "Virtualization Host" chkconfig libvirtd on && reboot` to install virtualization support and enable libvirt on boot. If you use the CentOS 7 DVD proceed to step 1b once the CentOS 7 with "Virtualization Host" support is completed.

1b. Boot the ISO off of a USB or other installation media and walk through installing OPNFV CentOS 7. The ISO comes prepared to be written directly to a USB drive with dd as such:

```
dd if=opnfv-apex.iso of=/dev/sdX bs=4M
```

Replace /dev/sdX with the device assigned to your usb drive. Then select the USB device as the boot media on your Jump Host

2a. Install these repos:

```
sudo yum install https://repos.fedorapeople.org/repos/openstack/
openstack-queens/rdo-release-queens-1.noarch.rpm      sudo yum install
epel-release     sudo curl -o /etc/yum.repos.d/opnfv-apex.repo http://
artifacts.opnfv.org/apex/gambia/opnfv-apex.repo
```

The RDO Project release repository is needed to install OpenVSwitch, which is a dependency of opnfv-apex. If you do not have external connectivity to use this repository you need to download the OpenVSwitch RPM from the RDO Project repositories and install it with the opnfv-apex RPM. The opnfv-apex repo hosts all of the Apex dependencies which will automatically be installed when installing RPMs, but will be pre-installed with the ISO.

2b. Download the first Apex RPMs from the OPNFV downloads page, under the TripleO RPMs `https://www.opnfv.org/software/downloads`. The dependent RPMs will be automatically installed from the opnfv-apex repo in the previous step. The following RPMs are available for installation:

- python34-opnfv-apex - (reqed) OPNFV Apex Python package
- python34-markupsafe - (reqed) Dependency of python34-opnfv-apex **
- python34-jinja2 - (reqed) Dependency of python34-opnfv-apex **
- python3-ipmi - (reqed) Dependency of python34-opnfv-apex **
- python34-pbr - (reqed) Dependency of python34-opnfv-apex **
- python34-virtualbmc - (reqed) Dependency of python34-opnfv-apex **
- python34-iptables - (reqed) Dependency of python34-opnfv-apex **
- python34-cryptography - (reqed) Dependency of python34-opnfv-apex **
- python34-libvirt - (reqed) Dependency of python34-opnfv-apex **

** These RPMs are not yet distributed by CentOS or EPEL. Apex has built these for distribution with Apex while CentOS and EPEL do not distribute them. Once they are carried in an upstream channel Apex will no longer carry them and they will not need special handling for installation. You do not need to explicitly install these as they will be automatically installed by installing python34-opnfv-apex when the opnfv-apex.repo has been previously downloaded to `/etc/yum.repos.d/`.

Install the required RPM (replace <rpm> with the actual downloaded artifact): `yum -y install <python34-opnfv-apex>`

3. After the operating system and the opnfv-apex RPMs are installed, login to your Jump Host as root.
4. Configure IP addresses on the interfaces that you have selected as your networks.
5. Configure the IP gateway to the Internet either, preferably on the public interface.
6. Configure your `/etc/resolv.conf` to point to a DNS server (8.8.8.8 is provided by Google).

7.11.2 Creating a Node Inventory File

IPMI configuration information gathered in section Execution Requirements (Bare Metal Only) needs to be added to the `inventory.yaml` file.

1. Copy `/usr/share/doc/opnfv/inventory.yaml.example` as your inventory file template to `/etc/opnfv-apex/inventory.yaml`.
2. The nodes dictionary contains a definition block for each baremetal host that will be deployed. 0 or more compute nodes and 1 or 3 controller nodes are required (the example file contains blocks for each of these already). It is optional at this point to add more compute nodes into the node list. By specifying 0 compute nodes in the inventory file, the deployment will automatically deploy "all-in-one" nodes which means the compute

will run along side the controller in a single overcloud node. Specifying 3 control nodes will result in a highly-available service model.

3. Edit the following values for each node:

   - `mac_address`: MAC of the interface that will PXE boot from undercloud
   - `ipmi_ip`: IPMI IP Address
   - `ipmi_user`: IPMI username
   - `ipmi_password`: IPMI password
   - **`pm_type`: Power Management driver to use for the node** values: pxe_ipmitool (tested) or pxe_wol (untested) or pxe_amt (untested)
   - `cpus`: (Introspected*) CPU cores available
   - `memory`: (Introspected*) Memory available in Mib
   - `disk`: (Introspected*) Disk space available in Gb
   - `disk_device`: (Opt***) Root disk device to use for installation
   - `arch`: (Introspected*) System architecture
   - **`capabilities`: (Opt**) Node's role in deployment** values: profile:control or profile:compute

   * Introspection looks up the overcloud node's resources and overrides these value. You can leave default values and Apex will get the correct values when it runs introspection on the nodes.

   ** If capabilities profile is not specified then Apex will select node's roles in the OPNFV cluster in a non-deterministic fashion.

   *** disk_device declares which hard disk to use as the root device for installation. The format is a comma delimited list of devices, such as "sda,sdb,sdc". The disk chosen will be the first device in the list which is found by introspection to exist on the system. Currently, only a single definition is allowed for all nodes. Therefore if multiple disk_device definitions occur within the inventory, only the last definition on a node will be used for all nodes.

7.11.3 Creating the Settings Files

Edit the 2 settings files in /etc/opnfv-apex/. These files have comments to help you customize them.

1. deploy_settings.yaml This file includes basic configuration options deployment, and also documents all available options. Alternatively, there are pre-built deploy_settings files available in (`/etc/opnfv-apex/`). These files are named with the naming convention os-sdn_controller-enabled_feature-[no]ha.yaml. These files can be used in place of the (`/etc/opnfv-apex/deploy_settings.yaml`) file if one suites your deployment needs. If a pre-built deploy_settings file is chosen there is no need to customize (`/etc/opnfv-apex/deploy_settings.yaml`). The pre-built file can be used in place of the (`/etc/opnfv-apex/deploy_settings.yaml`) file.
2. network_settings.yaml This file provides Apex with the networking information that satisfies the prerequisite Network Requirements. These are specific to your environment.

7.11.4 Running `opnfv-deploy`

You are now ready to deploy OPNFV using Apex! `opnfv-deploy` will use the inventory and settings files to deploy OPNFV.

Follow the steps below to execute:

1. Execute opnfv-deploy `sudo opnfv-deploy -n network_settings.yaml -i inventory.yaml -d deploy_settings.yaml` If you need more information about the options that can be passed to opnfv-deploy use `opnfv-deploy --help`. -n network_settings.yaml allows you to customize your networking topology. Note it can also be useful to run the command with the `--debug` argument which will enable a root login on the overcloud nodes with password: 'opnfvapex'. It is also useful in some cases to surround the deploy command with `nohup`. For example: `nohup <deploy command> &`, will allow a deployment to continue even if ssh access to the Jump Host is lost during deployment.
2. Wait while deployment is executed. If something goes wrong during this part of the process, start by reviewing your network or the information in your configuration files. It's not uncommon for something small to be overlooked or mis-typed. You will also notice outputs in your shell as the deployment progresses.
3. When the deployment is complete the undercloud IP and overcloud dashboard url will be printed. OPNFV has now been deployed using Apex.

7.12 Installation High-Level Overview - Virtual Deployment

Deploying virtually is an alternative deployment method to bare metal, where only a single bare metal Jump Host server is required to execute deployment. This deployment type is useful when physical resources are constrained, or there is a desire to deploy a temporary sandbox environment.

With virtual deployments, two deployment options are offered. The first is a standard deployment where the Jump Host server will host the undercloud VM along with any number of OPNFV overcloud control/compute nodes. This follows the same deployment workflow as baremetal, and can take between 1 to 2 hours to complete.

The second option is to use snapshot deployments. Snapshots are saved disk images of previously deployed OPNFV upstream. These snapshots are promoted daily and contain and already deployed OPNFV environment that has passed a series of tests. The advantage of the snapshot is that it deploys in less than 10 minutes. Another major advantage is that the snapshots work on both CentOS and Fedora OS. Note: Fedora support is only tested via PIP installation at this time and not via RPM.

7.12.1 Standard Deployment Overview

The virtual deployment operates almost the same way as the bare metal deployment with a few differences mainly related to power management. `opnfv-deploy` still deploys an undercloud VM. In addition to the undercloud VM a collection of VMs (3 control nodes + 2 compute for an HA deployment or 1 control node and 0 or more compute nodes for a Non-HA Deployment) will be defined for the target OPNFV deployment. All overcloud VMs are registered with a Virtual BMC emulator which will service power management (IPMI) commands. The overcloud VMs are still provisioned with the same disk images and configuration that baremetal would use. Using 0 nodes for a virtual deployment will automatically deploy "all-in-one" nodes which means the compute will run along side the controller in a single overcloud node. Specifying 3 control nodes will result in a highly-available service model.

To Triple-O these nodes look like they have just built and registered the same way as bare metal nodes, the main difference is the use of a libvirt driver for the power management. Finally, the default network settings file will deploy without modification. Customizations are welcome but not needed if a generic set of network settings are acceptable.

7.12.2 Snapshot Deployment Overview

Snapshot deployments use the same `opnfv-deploy` CLI as standard deployments. The snapshot deployment will use a cache in order to store snapshots that are downloaded from the internet at deploy time. This caching avoids re-downloading the same artifact between deployments. The snapshot deployment recreates the same network and libvirt setup as would have been provisioned by the Standard deployment, with the exception that there is no undercloud VM.

The snapshot deployment will give the location of the RC file to use in order to interact with the Overcloud directly from the jump host.

Snapshots come in different topology flavors. One is able to deploy either HA (3 Control, 2 Computes, no-HA (1 Control, 2 Computes), or all-in-one (1 Control/Compute. The snapshot deployment itself is always done with the os-odl-nofeature-* scenario.

7.13 Installation Guide - Virtual Deployment

This section goes step-by-step on how to correctly install and provision the OPNFV target system to VM nodes.

7.13.1 Special Requirements for Virtual Deployments

In scenarios where advanced performance options or features are used, such as using huge pages with nova instances, DPDK, or iommu; it is required to enabled nested KVM support. This allows hardware extensions to be passed to the overcloud VMs, which will allow the overcloud compute nodes to bring up KVM guest nova instances, rather than QEMU. This also provides a great performance increase even in non-required scenarios and is recommended to be enabled.

During deployment the Apex installer will detect if nested KVM is enabled, and if not, it will attempt to enable it; while printing a warning message if it cannot. Check to make sure before deployment that Nested Virtualization is enabled in BIOS, and that the output of `cat /sys/module/kvm_intel/parameters/nested` returns "Y". Also verify using `lsmod` that the kvm_intel module is loaded for x86_64 machines, and kvm_amd is loaded for AMD64 machines.

7.13.2 Install Jump Host

Follow the instructions in the Install Bare Metal Jump Host section.

7.13.3 Running `opnfv-deploy` for Standard Deployment

You are now ready to deploy OPNFV! `opnfv-deploy` has virtual deployment capability that includes all of the configuration necessary to deploy OPNFV with no modifications.

If no modifications are made to the included configurations the target environment will deploy with the following architecture:

- 1 undercloud VM
- The option of 3 control and 2 or more compute VMs (HA Deploy / default) or 1 control and 0 or more compute VMs (Non-HA deploy)
- 1-5 networks: provisioning, private tenant networking, external, storage and internal API. The API, storage and tenant networking networks can be collapsed onto the provisioning network.

Follow the steps below to execute:

1. `sudo opnfv-deploy -v [ --virtual-computes n ] [ --virtual-cpus n ] [ --virtual-ram n ] -n network_settings.yaml -d deploy_settings.yaml` Note it can also be useful to run the command with the `--debug` argument which will enable a root login on the overcloud nodes with password: 'opnfvapex'. It is also useful in some cases to surround the deploy command with `nohup`. For example: `nohup <deploy command> &`, will allow a deployment to continue even if ssh access to the Jump Host is lost during deployment. By specifying `--virtual-computes 0`, the deployment will proceed as all-in-one.

2. It will take approximately 45 minutes to an hour to stand up undercloud, define the target virtual machines, configure the deployment and execute the deployment. You will notice different outputs in your shell.
3. When the deployment is complete the IP for the undercloud and a url for the OpenStack dashboard will be displayed

7.13.4 Running `opnfv-deploy` for Snapshot Deployment

Deploying snapshots requires enough disk space to cache snapshot archives, as well as store VM disk images per deployment. The snapshot cache directory can be configured at deploy time. Ensure a directory is used on a partition with enough space for about 20GB. Additionally, Apex will attempt to detect the default libvirt storage pool on the jump host. This is typically '/var/lib/libvirt/images'. On default CentOS installations, this path will resolve to the /root partition, which is only around 50GB. Therefore, ensure that the path for the default storage pool has enough space to hold the VM backing storage (approx 4GB per VM). Note, each Overcloud VM disk size is set to 40GB, however Libvirt grows these disks dynamically. Due to this only 4GB will show up at initial deployment, but the disk may grow from there up to 40GB.

The new arguments to deploy snapshots include:

- *–snapshot*: Enables snapshot deployments
- *–snap-cache*: Indicates the directory to use for caching artifacts

An example deployment command is:

In the above example, several of the Standard Deployment arguments are still used to deploy snapshots:

- *-d*: Deploy settings are used to determine OpenStack version of snapshots to use as well as the topology
- *–virtual-computes* - When set to 0, it indicates to Apex to use an all-in-one snapshot
- *–no-fetch* - Can be used to disable fetching latest snapshot artifact from upstream and use the latest found in *–snap-cache*

7.13.5 Verifying the Setup - VMs

To verify the set you can follow the instructions in the Verifying the Setup section.

7.14 Deploying Directly from Upstream

When installing the Undercloud and Overcloud, the disk images are now downloaded directly from upstream artifacts. Essentially this deployment pulls the latest RDO overcloud and undercloud artifacts at deploy time. This option is useful for being able to pull the latest Queens and other OPNFV components which have been promoted via a TripleO pipeline and deemed to be stable.

7.14.1 Upstream Deployment Key Features

In addition to being able to install newer versions of OpenStack, the upstream deployment option allows the use of a newer version of TripleO, which provides overcloud container support. Therefore when deploying from upstream with an OpenStack version newer than Pike, every OpenStack service (also OpenDaylight) will be running as a docker container. Furthermore, deploying upstream gives the user the flexibility of including any upstream OpenStack patches he/she may need by simply adding them into the deploy settings file. The patches will be applied live during deployment.

7.15 Installation Guide - Upstream Deployment

This section goes step-by-step on how to correctly install and provision the OPNFV target system using a direct upstream deployment.

7.15.1 Special Requirements for Upstream Deployments

With upstream deployments it is required to have internet access. In addition, the upstream artifacts will be cached under the root partition of the jump host. It is required to at least have 10GB free space in the root partition in order to download and prepare the cached artifacts.

7.15.2 Scenarios and Deploy Settings for Upstream Deployments

The deploy settings and scenarios included with the Gambia release of Apex will be supported as deemed by the OPNFV Scenarios in Apex section of this guide. Each of these scenarios has been tested by Apex over the Gambia release, and using those deploy settings will control which upstream artifacts are pulled at deploy time. By specifying different versions of OpenStack, ODL, or other components in the deploy settings, different upstream artifacts may be downloaded and are not considered to be supported. For deploying newer versions of components it is advised to use the master branch of OPNFV Apex as part of our continuous integration effort to support those components.

7.15.3 Including Upstream Patches with Deployment

With upstream deployments it is possible to include any pending patch in OpenStack gerrit with the deployment. These patches are applicable to either the undercloud or the overcloud. This feature is useful in the case where a developer or user desires to pull in an unmerged patch for testing with a deployment. In order to use this feature, include the following in the deploy settings file, under "global_params" section:

```
patches:
  undercloud:
    - change-id: <gerrit change id>
      project: openstack/<project name>
      branch:  <branch where commit is proposed>
  overcloud:
    - change-id: <gerrit change id>
      project: openstack/<project name>
      branch:  <branch where commit is proposed>
```

You may include as many patches as needed. If the patch is already merged or abandoned, then it will not be included in the deployment.

7.15.4 Running `opnfv-deploy`

Deploying is similar to the typical method used for baremetal and virtual deployments with the addition of a few new arguments to the `opnfv-deploy` command. The artifacts for each upstream deployment are only downloaded when a newer version is detected upstream. In order to explicitly disable downloading new artifacts from upstream if previous artifacts are already cached, please use the `--no-fetch` argument.

7.15.5 Interacting with Containerized Overcloud

Upstream deployments will use a containerized overcloud. These containers are Docker images built by the Kolla project. The Containers themselves are run and controlled through Docker as root user. In order to access logs for each service, examine the '/var/log/containers' directory or use the *docker logs* <*container name*>. To see a list of services running on the node, use the `docker ps` command. Each container uses host networking, which means that the networking of the overcloud node will act the same exact way as a traditional deployment. In order to attach to a container, use this command: `docker exec -it <container name/id> bin/bash`. This will login to the container with a bash shell. Note the containers do not use systemd, unlike the traditional deployment model and are instead started as the first process in the container. To restart a service, use the `docker restart <container>` command.

7.16 Verifying the Setup

Once the deployment has finished, the OPNFV deployment can be accessed via the undercloud node. From the Jump Host ssh to the undercloud host and become the stack user. Alternatively ssh keys have been setup such that the root user on the Jump Host can ssh to undercloud directly as the stack user. For convenience a utility script has been provided to look up the undercloud's ip address and ssh to the undercloud all in one command. An optional user name can be passed to indicate whether to connect as the stack or root user. The stack user is default if a username is not specified.

```
opnfv-util undercloud root
su - stack
```

Once connected to undercloud as the stack user look for two keystone files that can be used to interact with the undercloud and the overcloud. Source the appropriate RC file to interact with the respective OpenStack deployment.

`source stackrc` (undercloud)
`source overcloudrc` (overcloud / OPNFV)

The contents of these files include the credentials for the administrative user for undercloud and OPNFV respectively. At this point both undercloud and OPNFV can be interacted with just as any OpenStack installation can be. Start by listing the nodes in the undercloud that were used to deploy the overcloud.

```
source stackrc
openstack server list
```

The control and compute nodes will be listed in the output of this server list command. The IP addresses that are listed are the control plane addresses that were used to provision the nodes. Use these IP addresses to connect to these nodes. Initial authentication requires using the user heat-admin.

```
ssh heat-admin@192.0.2.7
```

To begin creating users, images, networks, servers, etc in OPNFV source the overcloudrc file or retrieve the admin user's credentials from the overcloudrc file and connect to the web Dashboard.

You are now able to follow the *OpenStack Verification* section.

7.17 OpenStack Verification

Once connected to the OPNFV Dashboard make sure the OPNFV target system is working correctly:

1. In the left pane, click Compute -> Images, click Create Image.
2. Insert a name "cirros", Insert an Image Location `http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img`.
3. Select format "QCOW2", select Public, then click Create Image.
4. Now click Project -> Network -> Networks, click Create Network.
5. Enter a name "internal", click Next.
6. Enter a subnet name "internal_subnet", and enter Network Address `172.16.1.0/24`, click Next.
7. Now go to Project -> Compute -> Instances, click Launch Instance.
8. Enter Instance Name "first_instance", select Instance Boot Source "Boot from image", and then select Image Name "cirros".
9. Click Launch, status will cycle though a couple states before becoming "Active".
10. Steps 7 though 9 can be repeated to launch more instances.
11. Once an instance becomes "Active" their IP addresses will display on the Instances page.
12. Click the name of an instance, then the "Console" tab and login as "cirros"/"cubswin:)"
13. To verify storage is working, click Project -> Compute -> Volumes, Create Volume
14. Give the volume a name and a size of 1 GB
15. Once the volume becomes "Available" click the dropdown arrow and attach it to an instance.

Congratulations you have successfully installed OPNFV!

7.18 Developer Guide and Troubleshooting

This section aims to explain in more detail the steps that Apex follows to make a deployment. It also tries to explain possible issues you might find in the process of building or deploying an environment.

After installing the Apex RPMs in the Jump Host, some files will be located around the system.

1. /etc/opnfv-apex: this directory contains a bunch of scenarios to be deployed with different characteristics such HA (High Availability), SDN controller integration (OpenDaylight/ONOS), BGPVPN, FDIO, etc. Having a look at any of these files will give you an idea of how to make a customized scenario setting up different flags.
2. /usr/bin/: it contains the binaries for the commands opnfv-deploy, opnfv-clean and opnfv-util.
3. /usr/share/opnfv/: contains Ansible playbooks and other non-python based configuration and libraries.
4. /var/opt/opnfv/: contains disk images for Undercloud and Overcloud

7.18.1 Utilization of Images

As mentioned earlier in this guide, the Undercloud VM will be in charge of deploying OPNFV (Overcloud VMs). Since the Undercloud is an all-in-one OpenStack deployment, it will use Glance to manage the images that will be deployed as the Overcloud.

So whatever customization that is done to the images located in the jumpserver (/var/opt/opnfv/images) will be uploaded to the undercloud and consequently, to the overcloud.

Make sure, the customization is performed on the right image. For example, if I virt-customize the following image overcloud-full-opendaylight.qcow2, but then I deploy OPNFV with the following command:

```
sudo opnfv-deploy -n network_settings.yaml -d /etc/opnfv-apex/
os-onos-nofeature-ha.yaml
```

It will not have any effect over the deployment, since the customized image is the opendaylight one, and the scenario indicates that the image to be deployed is the overcloud-full-onos.qcow2.

7.18.2 Post-deployment Configuration

Post-deployment scripts will perform some configuration tasks such ssh-key injection, network configuration, NATing, OpenVswitch creation. It will take care of some OpenStack tasks such creation of endpoints, external networks, users, projects, etc.

If any of these steps fail, the execution will be interrupted. In some cases, the interruption occurs at very early stages, so a new deployment must be executed. However, some other cases it could be worth it to try to debug it.

1. There is not external connectivity from the overcloud nodes:

   Post-deployment scripts will configure the routing, nameservers and a bunch of other things between the overcloud and the undercloud. If local connectivity, like pinging between the different nodes, is working fine, script must have failed when configuring the NAT via iptables. The main rules to enable external connectivity would look like these:

   ```
   iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE       iptables
   -t nat -A POSTROUTING -s ${external_cidr} -o eth0 -j MASQUERADE
   iptables -A FORWARD -i eth2 -j ACCEPT        iptables -A FORWARD -s
   ${external_cidr} -m state --state ESTABLISHED,RELATED -j ACCEPT
   service iptables save
   ```

   These rules must be executed as root (or sudo) in the undercloud machine.

7.18.3 OpenDaylight Integration

When a user deploys a scenario that starts with os-odl*:

OpenDaylight (ODL) SDN controller will be deployed and integrated with OpenStack. ODL will run as a systemd service, and can be managed as as a regular service:

```
systemctl start/restart/stop opendaylight.service
```

This command must be executed as root in the controller node of the overcloud, where OpenDaylight is running. ODL files are located in /opt/opendaylight. ODL uses karaf as a Java container management system that allows the users to install new features, check logs and configure a lot of things. In order to connect to Karaf's console, use the following command:

```
opnfv-util opendaylight
```

This command is very easy to use, but in case it is not connecting to Karaf, this is the command that is executing underneath:

```
ssh -p 8101 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
karaf@localhost
```

Of course, localhost when the command is executed in the overcloud controller, but you use its public IP to connect from elsewhere.

7.18.4 Debugging Failures

This section will try to gather different type of failures, the root cause and some possible solutions or workarounds to get the process continued.

1. I can see in the output log a post-deployment error messages:

   Heat resources will apply puppet manifests during this phase. If one of these processes fail, you could try to see the error and after that, re-run puppet to apply that manifest. Log into the controller (see verification section for that) and check as root /var/log/messages. Search for the error you have encountered and see if you can fix it. In order to re-run the puppet manifest, search for "puppet apply" in that same log. You will have to run the last "puppet apply" before the error. And It should look like this:

   ```
   FACTER_heat_outputs_path="/var/run/heat-config/
   heat-config-puppet/5b4c7a01-0d63-4a71-81e9-d5ee6f0a1f2f"
   FACTER_fqdn="overcloud-controller-0.localdomain.com" \
   FACTER_deploy_config_name="ControllerOvercloudServicesDeployment_Step4"
   puppet apply --detailed-exitcodes -l syslog -l
   console \ /var/lib/heat-config/heat-config-puppet/
   5b4c7a01-0d63-4a71-81e9-d5ee6f0a1f2f.pp
   ```

   As a comment, Heat will trigger the puppet run via os-apply-config and it will pass a different value for step each time. There is a total of five steps. Some of these steps will not be executed depending on the type of scenario that is being deployed.

7.18.5 Reporting a Bug

Please report bugs via the OPNFV Apex JIRA page. You may now use the log collecting utility provided by Apex in order to gather all of the logs from the overcloud after a deployment failure. To do this please use the `opnfv-pyutil --fetch-logs` command. The log file location will be displayed at the end of executing the script. Please attach this log to the JIRA Bug.

7.19 Frequently Asked Questions

7.20 License

All Apex and "common" entities are protected by the Apache 2.0 License.

7.21 References

7.21.1 OPNFV

OPNFV Home Page

OPNFV Apex project page

OPNFV Apex Release Notes

7.21.2 OpenStack

OpenStack Queens Release artifacts

OpenStack documentation

7.21.3 OpenDaylight

Upstream OpenDaylight provides a number of packaging and deployment options meant for consumption by downstream projects like OPNFV.

Currently, OPNFV Apex uses OpenDaylight's Puppet module, which in turn depends on OpenDaylight's RPM.

7.21.4 RDO Project

RDO Project website

Authors Tim Rozet (trozet@redhat.com)

Authors Dan Radez (dradez@redhat.com)

Version 7.1

7.22 Indices and tables

- search

CHAPTER 8

OPNFV Apex Release Notes

Contents:

8.1 OPNFV Release Notes for the Gambia release of OPNFV Apex deployment tool

8.1.1 Abstract

This document provides the release notes for Gambia release with the Apex deployment toolchain.

8.1.2 License

8.1.3 Important Notes

This is the OPNFV Gambia release that implements the deploy stage of the OPNFV CI pipeline via Apex.

Apex is based on RDO's Triple-O installation tool chain. More information at http://rdoproject.org

Carefully follow the installation-instructions which guide a user on how to deploy OPNFV using Apex installer.

8.1.4 Summary

Gambia release with the Apex deployment toolchain will establish an OPNFV target system on a Pharos compliant lab infrastructure. The current definition of an OPNFV target system is OpenStack Pike combined with an SDN controller, such as OpenDaylight. The system is deployed with OpenStack High Availability (HA) for most OpenStack services. SDN controllers are deployed on every controller unless deploying with one the HA FD.IO scenarios. Ceph storage is used as Cinder backend, and is the only supported storage for Gambia. Ceph is setup as 3 OSDs and 3 Monitors,

one OSD+Mon per Controller node in an HA setup. Apex also supports non-HA deployments, which deploys a single controller and n number of compute nodes. Furthermore, Apex is capable of deploying scenarios in a bare metal or virtual fashion. Virtual deployments use multiple VMs on the Jump Host and internal networking to simulate the a bare metal deployment.

- Documentation is built by Jenkins
- .iso image is built by Jenkins
- .rpm packages are built by Jenkins
- Jenkins deploys a Gambia release with the Apex deployment toolchain bare metal, which includes 3 control+network nodes, and 2 compute nodes.

8.1.5 Release Data

Project	apex
Repo/tag	opnfv-7.1.0
Release designation	7.1.0
Release date	2018-12-14
Purpose of the delivery	OPNFV Gambia release

Version change

Module version changes

This is the first tracked version of the Gambia release with the Apex deployment toolchain. It is based on following upstream versions:

- OpenStack (Queens release)
- OpenDaylight (Oxygen releases)
- CentOS 7

Document Version Changes

This is the first tracked version of Gambia release with the Apex deployment toolchain. The following documentation is provided with this release:

- OPNFV Installation instructions for the Gambia release with the Apex deployment toolchain - ver. 1.0.0
- OPNFV Release Notes for the Gambia release with the Apex deployment toolchain - ver. 1.0.0 (this document)

Deliverables

Software Deliverables

- Apex .rpm (python34-opnfv-apex)
- build.py - Builds the above artifact
- opnfv-deploy - Automatically deploys Target OPNFV System
- opnfv-clean - Automatically resets a Target OPNFV Deployment

- opnfv-util - Utility to connect to or debug Overcloud nodes + OpenDaylight

Documentation Deliverables

- OPNFV Installation instructions for the Gambia release with the Apex deployment toolchain - ver. 7.1
- OPNFV Release Notes for the Gambia release with the Apex deployment toolchain - ver. 7.1 (this document)

8.1.6 Known Limitations, Issues and Workarounds

System Limitations

Max number of blades: 1 Apex undercloud, 3 Controllers, 20 Compute blades

Min number of blades: 1 Apex undercloud, 1 Controller, 1 Compute blade

Storage: Ceph is the only supported storage configuration.

Min master requirements: At least 16GB of RAM for baremetal Jump Host, 24GB for virtual deployments (noHA).

Known Issues

JIRA TICKETS:

JIRA REFERENCE	SLOGAN
JIRA: APEX-280	Deleted network not cleaned up on controller
JIRA: APEX-295	Missing support for VLAN tenant networks
JIRA: APEX-368	Ceilometer stores samples and events forever
JIRA: APEX-371	Ceph partitions need to be prepared on deployment when using 2nd disk
JIRA: APEX-375	Default glance storage points to http,swift when ceph disabled
JIRA: APEX-389	Compute kernel parameters are used for all nodes
JIRA: APEX-412	Install failures with UEFI

Workarounds

-

8.1.7 Test Result

Please reference Functest project documentation for test results with the Apex installer.

8.1.8 References

For more information on the OPNFV Gambia release, please see:

http://wiki.opnfv.org/releases/Gambia

Authors Tim Rozet (trozet@redhat.com)

Authors Dan Radez (dradez@redhat.com)

Version 7.1