OPNFV Platform Overview Document

Release brahmaputra.3.0 (605ebda)

OPNFV

April 27, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION

OPNFV is an integration effort that takes outputs from several open source communities to build a NFV platform. This task of integration leads to providing different kinds of output to its users.

The primary goal of the OPNFV project is the target software platform, which is a integrated solution of a set of components/building blocks of the ETSI ISG NFV reference architecture. In the Brahmaputra release, this is limited to the NFVI and VIM blocks. OPNFV users will be able to deploy their VNFs there using some MANO solution. The target software platform is integrated from a set of other open source components, of which the biggest ones are OpenStack and SDN controllers. There are multiple combinations possible and a subset is provided and tested by the Brahmaputra release. These subsets are called here scenarios.

Besides the target software platform, OPNFV provides a set of tools that helps the user deploy this target software platform on a set of servers. These tools are installers. Brahmaputra provides multiple options here. Naturally the different installers have different capabilities, that is they support deployment of different scenarios.

The installers allow users to deploy OPNFV target software platform on a bare metal environment or a set of virtual machines. In both cases, some hosts (bare metal or virtual) will act as controller nodes, while other hosts will be the compute nodes hosting the VNFs. The installers use a separate server to control the deployment process. This server is called "jump server" and is installed with the installer's software at the beginning of a deployment. The jump server also can be bare metal or virtual.

This configuration - jump servers and a set of typically 5 nodes to run the target software platform - is also described as part of an OPNFV release. This allows the users to build their own labs accordingly and deploy OPNFV easily. A lab compliant to this description sometimes is called "Pharos-compliant" after the OPNFV project providing the lab description.

Another major part of the OPNFV release is a testing framework and test cases. This test framework allows users to verify their deployment of the OPNFV target software platform. It will execute and test major functions of the platform relevant to NFV applications (VNFs) so the user can be confident that VNFs can successfully run.

OPNFV releases come with the necessary documentation describing target software platform, deployment tools, test cases, etc. in their architecture, configuration and usage. The most important documents here are configuration guides and user guides that help to set up a OPNFV deployment and use it.

The OPNFV project takes major effort to provide lab environments to the community. The OPNFV community labs of course need to be Pharos-compliant. They are used for OPNFV development tasks and release creation, but should also provide users with the opportunity to run their own OPNFV tests. OPNFV community labs are not part of a OPNFV release. Please find more information on the labs in the Pharos project documentation.

We should also mention that OPNFV works on requirements of open source projects used in OPNFV to make these projects better suitable for NFV telco carrier use cases. These requirements are described in requirement documents and also forwarded to the "upstream" projects in the format required by these projects. These requirement documents are not bound to OPNFV releases.

OPNFV bundles the target software, installers, documentation, test cases and lab description to releases.

This overview document introduces these components and scenarios on a high level and points you to more detailed documentation.

CHAPTER

TWO

OPNFV PLATFORM ARCHITECTURE

The OPNFV project addresses a number of aspects in the development of a consistent virtualization platform including common hardware requirements, software architecture and installed state. The platform architecture as the OPNFV project approaches it is dicussed in the following sections.

2.1 OPNFV Lab Infrastructure

The Pharos Project provides a lab infrastructure that is geographically and technically diverse. Labs instantiate bare-metal and virtual environments that are accessed remotely by the community and used for OPNFV platform and feature development, build, deploy and testing. This helps in developing a highly robust and stable OPNFV platform with well understood performance characteristics.

Community labs are hosted by OPNFV member companies on a voluntary basis. The Linux Foundation also hosts an OPNFV lab that provides centralised CI and other production resources which are linked to community labs. Future lab capabilities will include the ability easily automate deploy and test of any OPNFV install scenario in any lab environemnt as well as on a virtual infrastructure.

2.2 Software architecture

This section will provide information which upstream projects, versions and components are integrated in the release to implement OPNFV requirement. You can find the list of common requirements for deployment tools here: http://artifacts.opnfv.org/genesisreq/brahmaputra/requirements/requirements.pdf.

2.2.1 OpenStack

OPNFV integrates OpenStack as cloud management system where Brahmaputra uses the OpenStack Liberty Release. The set of sub-projects deployed in a brahmaputra platform varies slightly depending on the installer and scenario.

The following table shows which components are deployed.

services	type	Apex	Compass	Fuel	Joid
aodh	alarming	Available	—	—	—
ceilometer	metering	Available	Available	Available	Available
cinder	volume	Available	Available	Available	Available
cloud	cloudformation	—	Available	Available	Available
glance	image	Available	Available	Available	Available
heat	orchestration	Available	Available	Available	Available
keystone	identity	Available	Available	Available	Available
neutron	network	Available	Available	Available	Available
nova	compute	Available	Available	Available	Available
swift	object-store	Available	—	Available	Available

Note that additional components of OpenStack are used as part of deployment tools and test frameworks (Fuel, Tempest, Rally).

For more information about the OpenStack features and usage refer to the official OpenStack documentation page. Please ensure that you have the Liberty release selected at the `More Releases & Languages` drop down menu.

2.2.2 Operating System

OPNFV uses Linux on all target machines. Depending on the installers, different distributions are supported.

Ubuntu 14 supported by Fuel, Compass and Joid installers CentOS 7 supported by Apex and Compass

2.2.3 SDN Controllers

OPNFV Brahmaputra release supports different SDN controllers. Some scenarios don't use an SDN controller but rely just on Neutron networking capabilities.

Depending on the SDN controller you are using, the featureset available will vary. More information on feature support and scenarios can be found in OPNFV Configuration Guide and OPNFV User Guide. Brahmaputra also provides scenarios without an SDN controller, just using OpenStack Neutron.

OpenDaylight is an SDN controller aiming to accelerate adoption of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) with a transparent approach that fosters new innovation. OpenDaylight runs within a JVM and is installed in OPNFV within a container and integrated with OpenStack using the ODL device driver. The Brahmaputra release of OPNFV integrates the Beryllium release. You can find more information about OpenDaylight among the release artifacts at the Downloads page. Please ensure you are using the Beryllium documentation.

ONOS is an SDN controller written in Java with a distributed architecture with special focus to support scalability, fault tolerance and hardware and software upgrades without interrupting network traffic. More information on the internal design of ONOS may be found in User's Guide and Architecture+Guide on the wiki of the ONOS project. ONOS is integrated to OPNFV using a framework ONOSFW and Neutron plugins. Details can be found in the ONOS specific OPNFV documents, Design guide, User guide and Configuration guide.

2.2.4 Data Plane

OPNFV extends Linux's virtual networking capabilies by using virtual switch and router components including improving those components by requirements specific to telco use cases.

For instance some scenarios use OpenVSwitch to replace Linux bridges as it offers advantages in terms of mobility, hardware integration and use by network controllers. OPNFV leverages these by upgrade to a specific installation using user-space datapath. This includes changes to other dataplane components, including libvirt, qemu, DPDK etc. Please find more information in OVSNFV User's Guide.

2.2.5 Other Components

KVM

NFV infrastructure has special requirements on hypervisors with respect of interrupt latency (timing correctness and packet latency in data plane) and live migration.

Besides additional software changes, this requires some adjustments to system configuration information, like hardware, BIOS, OS, etc.

Please find more information at KVM4NFV project documentation.

2.3 Deployment Architecture

OPNFV starts with a typical configuration with 3 controller nodes running OpenStack, SDN, etc. and a minimum of 2 compute nodes for deployment of VNFs. A detailed description of this 5 node configuration can be found in pharos documentation.

The 3 controller nodes allow to provide an HA configuration. The number of compute nodes can be increased dynamically after the initial deployment.

OPNFV can be deployed on bare metal or in a virtual environment, where each of the hosts is a virtual machine and provides the virtual resources using nested virtualization.

The initial deployment is done using a so-called "jumphost". This server (either bare metal or virtual) is first installed with the installer program that then installs OpenStack and other components on the controller nodes and compute nodes. See the OPNFV User Guide for more details.

In Brahmaputra, different scenarios can be deployed to provide the different feature sets, e.g. HA, IPV6, BGPVPN, KVM, or select the different implementations, e.g. SDN controllers.

The following scenarios are supported, some of them can be deployed using different installers.

- nosdn-nofeature
- odl_l2-ha
- odl_l3-ha
- odl_l2-bgpvpn-noha
- onos-ha
- nosdn-ovs-ha
- nosdn-kvm-ha
- nosdn-ovs_kvm-ha

Please find more information at: http://artifacts.opnfv.org/opnfvdocs/brahmaputra/configguide/configoptions.html#opnfv-scenario-s.

CHAPTER

THREE

DEPLOYMENT TOOLS

Brahmaputra provides 4 different installers.

The installers will deploy the target platform onto a set of virtual or bare metal servers according to the configuration files. After the deployment, it doesn't matter which of the installers had been used to deploy the target scenario.

Apex is an OPNFV Installation tool based on RDO Manager that deploys OPNFV using the RDO Project OpenStack Distribution. RDO manager is a Triple-O based installation tool. Triple-O is an image based life cycle deployment tool that is a member of the OpenStack Big Tent Governance. Apex uses Centos on all target platforms and can deploy all SDN controllers. Find more information at OpenStack Wiki for Triple-O and OPNFV Configuration Guide.

Compass is an installer project based on open source project Compass, which provides automated deployment and management of OpenStack and other distributed systems. It can be considered as what the LiveCD to a single box for a pool of servers – bootstrapping the server pool. Compass is based on Ansible. It can deploy Ubuntu or Centos as target operating system and ODL and ONOS as SDN controllers. Find more information at OpenStack Wiki for Compass and OPNFV Configuration Guide.

Fuel is an installer project based on the Fuel OpenStack open source project providing automated deployment and management of OpenStack and other distributed systems. Fuel is based on puppet and deploys the Ubuntu Linux operating system; the OpenStack virtual Infra-structure manager, and OpenDaylight or ONOS SDN controllers. Find more information at OpenStack Wiki for Fuel and OPNFV Configuration Guide.

Joid is an installer utilizes the technology developed in Juju and MAAS. Juju allows you to deploy, configure, manage, maintain, and scale cloud services quickly and efficiently on public clouds, as well as on physical servers, OpenStack, and containers. Together with MAAS hardware usage can be optimized. For more info on Juju and MAAS, please visit https://jujucharms.com/, http://maas.ubuntu.com and the Joid User Guide.

CHAPTER

FOUR

TESTING ECOSYSTEM

Testing is a key area and also a challenge for OPNFV in order to be able to verify and validate the platform we are creating.

This is a complex task as we have to test the integration of the components, the functionality we are creating and using and we have to verify that the platform is suitable for telecom applications. We do testing in an automated fashion by using several test tools and frameworks in our CI/CD pipeline.

This chapter gives an overview about our testing tools and activities.

4.1 Release verification

OPNFV releases the target platform together with the deployment tools in a set of scenarios that provides choices regarding the deployed components and the available features. The scenarios and the provided functionality are tested automatically in the CI/CD pipeline mentioned above and they are considered to be ready for release after at least 4 successful consecutive iterations.

The functional testing and the platform validation are divided between two projects called Functest and Yardstick.

4.1.1 Functest

Functest provides a functional testing framework along with a set of test suites and test cases that test and verify OPNFV platform functionality. The scope of Functest and relevant test cases can be found in its user guide.

In Brahmaputra, Functest is focusing on OpenStack and SDN controllers deployment testing. Its testing framework combines a number of testing tools to verify the key components of the OPNFV platform are running successfully. For example, Rally and Tempest are integrated for OpenStack basic functional testing and benchmark, Robot is used for ODL testing, and Teston is integrated for ONOS testing. Besides these, Functest also includes tests by deploying candidate VNFs such as vPing and vIMS, and testing their basic functionality.

4.1.2 Yardstick

Yardstick is a testing project for verifying the infrastructure compliance when running VNF applications. Yardstick can benchmark a number of characteristics/performance vectors about the infrastructure, that makes it become a useful pre-deployment NFVI testing tools.

Yardstick is also a flexible testing framework supporting OPNFV feature testing by the various projects in OPNFV. Projects can plug in their test cases for specific features easily.

The detail of Yardstick project can be found in the yardstick user guide.

There are two types of test cases in Yardstick: Yardstick generic test cases and OPNFV feature test cases. Yardstick generic test cases include basic characteristics benchmarking in compute/storage/network area. OPNFV feature test cases include basic telecom feature testing from OPNFV projects, for example nfv-kvm, sfc, ipv6, Parser, Availability and SDN VPN.

All of the Yardstick test cases are listed on http://artifacts.opnfv.org/yardstick/brahmaputra/docs/configguide_yardstick_testcases/03-list-of-tcs.html.

4.2 Additional Testing

Besides the test suites and cases for release verification, there are some additional testing for specific feature or characteristics on OPNFV platform. These testing framework and test cases may include some specific needs, such as extended measurements, or additional testing stimuli, or tests which cause disturbances on the environment. These additional testing can provide a more complete evaluation about OPNFV platform deployment.

4.2.1 Qtip

Qtip is a performance benchmark testing project by using a "Bottom-Up" approach in characterizing and benchmarking OPNFV platform. Qtip aims to benchmark the performance of components for a quantitative analysis and doesn't deal with validation of the platform.

In Brahmaputra, Qtip develops a flexible framework, adds a number of test cases covering compute/storage/network area, and compares these benchmarks on a bare metal machine vs a VM. These contrastive result can be used to evaluate the performance of these components on the OPNFV platform basically.

4.2.2 VSPERF

VSPERF will develop a generic and architecture agnostic vSwitch testing framework and associated tests, that will serve as a basis for validating the suitability of different vSwitch implementations in a Telco NFV deployment environment. The output of this project will be utilized as part of OPNFV Platform and VNF level testing and validation.

4.2.3 Bottlenecks

Bottlenecks will provide a framework to find system bottlenecks by testing and verifying OPNFV infrastructure in a staging environment before committing it to a production environment. The Bottlenecks framework can not only find specific system limitations and bottlenecks, but also the root cause of these bottlenecks.

In Brahmaputra, Bottlenecks includes two test cases: rubbos and vstf. These test cases are executed on OPNFV community pods, and test result report are visible on the community testing dashboard.