OPNFV Platform Overview document

Release arno.2015.1.0 (72e3bbe)

OPNFV

February 22, 2016

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.

First of all there is of course 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 called **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.

Of course, the OPNFV releases come with the necessary **documentation**, describing target software platform, deployment tools, tests, etc. in their architecture 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.

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** and provides documentation describing the scope and features provided.

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

Also community labs are independent of releases. Only the lab description is included in the release and describes the requirements of a lab to successfully run Brahmaputra deployments.

CHAPTER

TWO

TARGET SOFTWARE PLATFORM

2.1 Software architecture

This section will provide information which upstream projects, versions and components are integrated in the Brahmaputra release

2.1.1 OpenStack

OPNFV uses OpenStack as cloud management system. Brahmaputra is based on OpenStack Liberty Release. It comprises the following sub-projects and modules:

- Nova (Compute)
- Neutron (Network)
- Cinder (Block Storage)
- Swift (Object Storage)
- Ceilometer (Telemetry)
- Keystone (Identity)
- Glance (Image Service)
- Heat (Orchestration)
- etc.

Some of the sub-projects are not deployed in all scenarios. Besides target software, also deployment and test framework use OpenStack components (Fuel, Tempest, Rally)

2.1.2 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

2.1.3 SDN Controllers

OPNFV Brahmaputra release supports three different SDN controllers:

- OpenDaylight (ODL, Beryllium release)
- ONOS (Emu release)

- OpenContrail (?)

Depending on the SDN controller you are using, the featureset will vary. Brahmaputra also provides scenarios without an SDN controller, just using OpenStack Neutron.

OpenDaylight

Editor's note: We need a high level paragraph here and a description of how we use ODL.

ONOS

ONOS stands for **O** pen **N** etwork **O** perating **S** ystem. ONOS provides the control plane for a software-defined network (SDN), managing network components, such as switches and links, and running software programs or modules to provide communication services to end hosts and neighboring networks.

ONOS provides a platform for SDN applications and use cases for routing, management, or monitoring services for software-defined networks.

ONOS can run as a distributed system across multiple servers, allowing it to use the CPU and memory resources of multiple servers while providing fault tolerance in the face of server failure and potentially supporting live/rolling upgrades of hardware and software without interrupting network traffic.

The ONOS kernel and core services, as well as ONOS applications, are written in Java as bundles that are loaded into the Karaf OSGi container. OSGi is a component system for Java that allows modules to be installed and run dynamically in a single JVM.

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:

OpenContrail

Editors note: We need a high level paragraph here and a description of how we use OpenContrail, including its vRouter capabilities.

2.1.4 Data Plane

2.1.5 Other Components

2.2 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 installer documentation for more details.

Editors note: In a second level of detail, describe how software is distributed over the 3 controller nodes, compute nodes and other hardware.

In Brahmaputra, the following scenarios are supported:

2.3 Dynamic View

Editors note: we might skip this section completely for Brahmaputra.

Or we provide rather short statements. In later versions, we have to describe which software is involved in which way during:

- VNF Life Cycle (onboarding, instantiate, scaling): we can reference to other documents
- Hardware Life Cycle (mainly how to add compute nodes, but also other cases)
- ...

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.

3.1 Apex

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.

3.2 Compass

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.

3.3 Fuel

Fuel is an installer project based on the Fuel OpenStack open source project providing automated deployment and management of OpenStack and other distributed systems. It is based on puppet and deploys the Ubuntu Linux operating system; the OpenStack virtual Infra-structure manager, and OpenDaylight or ONOS SDN controllers.

3.4 Joid

Editors note: Just a high level intro and link to the main joid documents.

CHAPTER

FOUR

TESTCASES AND FRAMEWORK

Testing is a very important area and a key challenge for OPNFV platform. Testing an OPNFV deployment from multiple scopes and different perspectives can verify the real status of OPNFV platform. In OPNFV community, several specific testing projects have been validated and already deal with developing frameworks, tools and test cases. Some of them aim to functional verification, some of them focus on performance measurement and reliability testing. Finally, OPNFV will build an open source automated testing ecosystem for telecom cloud testing.

4.1 Release verification

Before a new OPNFV platform version is released, some suites of test cases are executed to verify major functions and capability of the platform. These test cases are automatically executed as part of the CI/CD pipeline, and passing these test cases at least 4 successful consecutive iterations in CI demonstrates the platform is ready for being released. In Brahmaputra, these test cases for verification are provided by two testing projects: Functest and Yardstick.

4.1.1 Functest

Functest tries to provide a functional testing framework along with a set of test suites and test cases to test and verify OPNFV platform functionality that covers the VIM and NFVI components. The scope of Functest and relevant test cases can be found at: https://wiki.opnfv.org/opnfv_functional_testing.

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 OPNFV platform 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 candidate VNF functional testing, such as vPing and vIMS.

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 for a number of OPNFV feature testing. The detail of Yardstick project can be found at: https://wiki.opnfv.org/yardstick.

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 form other OPNFV projects, just like nfv-kvm, sfc, ipv6, Parser, Availability and SDN VPN. All of the Yardstick test cases are listed on: https://wiki.opnfv.org/candidates_for_test_cases.

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.