The virtualization of the communications service provider infrastructure presents an opportunity for agile service delivery and short time to market. The path to successfully harnessing those benefits requires adopting an operations model that will overcome the pitfalls hidden within the virtualization technology. This whitepaper describes the existing operational shortcomings in NFV infrastructure and how they can be avoided using methodology and automation.
Contents

Introduction 3
Interoperability with NFVi well-known interfaces 3
Variances in configuration over time 5
Important aspects of an automatic validation tool 5
Verifying functionality and performance after configuration has been validated 7
How to perform verification 8
What to do with verification test results 9
What comes after validation and NFVi verification 9
Role of the verification tool 10
Deployment and operation of a VNF 11
Acronyms 12
Introduction

The concept of network functions virtualization (NFV) has been around for several years. The benefits of virtualizing the communications service provider infrastructure and using proven IT technologies instead of dedicated hardware are obvious and need no further explanation. Service providers, equipment vendors, researchers and independent software vendors have long been experimenting with virtualized network functions (VNFs) running on COTS hardware with virtualization managers such as OpenStack® and VMware. The technologies of software-defined networking (SDN) and software-defined storage (SDS) play key roles. Organizations including ETSI, TM Forum, the MEF, The Linux Foundation and others are defining the architecture for NFV in order to ensure compatibility and interoperability. The use of OpenStack and other community-driven software stacks provides the benefit of a huge community-backed technology that has the right functionality with feature richness and high quality.

However, there is a downside to using this type of leading-edge technology. With the constant changes and ever-growing list of features, getting a VNF instance to run on such a platform proves to be a challenging task. It requires a significant amount of engineering work to get a single VNF to run on a new platform. Often certain tweaks are required, both on the VNF and the NFV platform (or NFV infrastructure [NFVi] in the ETSI NFV architecture). The fact that a VNF was onboarded and validated for functionality and performance on a certain NFVi instance does not guarantee that it will function properly on another NFVi instance. The differences between NFVi instances may be as obvious as different software versions, but may be harder to spot in a different configuration of the NFVi software.

The issues caused by the NFVi affecting the service performance and functionality must be addressed if we are to move NFV from the proof-of-concept and lab phases into production. It is unacceptable to expect that every new service deployment of a VNF (or a modification to an existing one) will involve manual work and lengthy problem solving processes. In order to fulfill the promise of agile service introduction and shorter time-to-market for new services, the issues must be addressed using methodic approaches and automating tools.

Interoperability with well-known NFVi interfaces

Even though an API is well known, the details of the implementation behind the API might cause trouble. Consider, for example, the vCPU core allocation. While there is a standard way in the OpenStack Nova API to specify how many cores a flavor will have, the way the virtual cores are allocated to the guest
virtual machines (VMs) is governed by different configuration parameters of the hypervisor. The actual cores that will be assigned to the guest VM depends on factors such as:

• How many cores are enabled for virtualization in the host?
• How many cores per socket are there?
• Are hyper-threaded sibling cores enabled in the hypervisor BIOS?

Therefore, it is highly likely that using the same Nova flavors on two different NFVi instances will yield different sets of cores assigned to the NFV VM. The NFV software might act differently depending on what core set it is running on, resulting in different performance and sometimes leading to software crashes. Being able to know the core allocation scheme of the hypervisor could prevent situations where the VNF is deployed in a sub-optimal environment.

The information related to core allocation is not available from a single entity as some of it resides in the physical server BIOS, some in the hypervisor configuration, and some in the OpenStack configuration. Before attempting to deploy a VNF, this information should be gathered and compared to the desired allocation scheme of the VNF. While it is possible to manually gather the information, it is obvious that an automatic validation tool would do such a job faster and eliminate the chance of errors. If a VNF is deployed on an NFVi instance that does not provide the right set of virtual cores, it could be extremely difficult to pinpoint the root cause of problems. Often the path from the symptom to the cause may not be a clear and solid one. Running the validation tool prior to the NFV deployment attempt could save hours, sometimes even days, of attempting to find out what went wrong with the deployment.

Another example may be network interface maximum transfer unit (MTU). It has been observed that some VNFs may require a large MTU in order to enable tunneling and encapsulation. Some other VNFs have shown malfunctions in cases where the MTU of the infrastructure was inconsistent with the configured MTU of the guest VMs. In such cases, the virtual network MTU must be equal to that of the guest VM.

The OpenStack Neutron API has a well-known API for creating network ports, but it does not provide a standard way to control the MTU. The actual MTU of the virtual network port depends on a configuration of the hypervisor. Attempting to deploy a VNF on a hypervisor with an MTU that does not meet the one desired by the VNF can result in malfunctions that are sometimes hard to trace (e.g., the problem may only present itself for certain network packets but not for others). Running automated validation before deploying the VNF takes the guesswork out of the process and increases the certainty that the VNF will function properly.
Variance in configuration over time

Even if at some certain point in time the NFVi was configured correctly, when another VNF is deployed, it is likely that it will have different requirements. This might cause the administrator of the NFVi to make configuration changes that will no longer provide an optimal environment for the original VNF.

The NFVi may be patched or updated to a newer version, causing some configurations to be overwritten. Newer NFVi versions may read their configuration from a different source (e.g., networking parameters read from a Neutron configuration file rather than from the hypervisor).

Performing the validation process once does not guarantee the results will remain the same over time. It is recommended that the validation process be carried out:

- Before deploying a VNF
- At periodic intervals to ensure consistent functionality
- Before the VNF or any of its components are migrated or scaled out.
  This is required in order to ensure that the new instances are running on infrastructure that provides the same functionality as the original instances.
- Whenever there is a suspected malfunction or performance downgrade of the VNF. Since the validation is fully automated, it requires very minimal effort in order to run. It should become an essential and early phase in any VNF troubleshooting process.

Important aspects of an automatic validation tool

There are no secrets as to the parameters that need to be validated. After several attempts to deploy, every VNF developer can identify the NFVi parameters that are affecting his VNF. Most of the time the information is in a configuration database or file, and only basic information such as the name of the parameter or configuration element is needed. That information is usually available through a database query or a shell script. The trick is to sift through all the available information and find the right parameters. Once the desired information is obtained, it must be compared to some reference data set in order to determine a go/no-go for the VNF deployment.

In order to make the validation process effortless we would recommend the use of an automated validation toolset. From our experience, we have determined that any tool deployed should have the following characteristics:

- Simplifies access to the relevant source data. The end user should provide some basic information such as the IP address and credentials of the OpenStack controller and the tool should do the rest.
• Provides a set of reference data or desired filtered data set. This data can be matched with the data retrieved from the NFVi in order to determine the compatibility between VNF needs and NFVi capabilities.

• Provides a clear summary of validation results with a go/no-go indication.

• Provides the ability to drill down and get further data in case of a mismatch. It may be important to discover that the MTU on some hosts is different from the desired one, but to assist resolution the user of the tool may also want to know the actual configured value and the MTU for other available hosts in the infrastructure.

• Be a compact and easily deployable portable application that will enable validation of any NFVi. The need to run validation may arise at any time and in various locations. It is important to minimize the effort required to prepare the tool for action. If the tool is not easy to deploy and use, the professionals dealing with VNF deployments will be reluctant to use it and prefer some inefficient manual validation techniques.

• Be extensible. Our experience indicates that the set of parameters and configuration elements that need to be validated is an ever-growing list. As new capabilities emerge in the NFVi, VNFs become sensitive to new configuration elements. If the validation tool is not able to keep up with new requirements, it will be abandoned and the users will resort to using the cumbersome manual validation techniques, or skip validation altogether and face extended deployment times.

• Presents comparison between current and archived previous test results, to detect changes that might affect the VNF.

Figure 1. A Validation tool UI – providing an interface to specify desired values and presenting a clear go/no-go result
Verifying functionality and performance after configuration has been validated

The validation process can guarantee that all the parameters are set properly but that does not guarantee the expected performance from the NFVi or even the expected behavior. Some aspects of the NFVi behavior cannot be predicted based on the configuration data alone. This may be the result of:

- External entities that are affecting the NFVi - external networking equipment, a power distribution system, or even improper cooling
- Differences between the current deployed hardware and the set used for VNF validation
- Newly introduced functionality in the infrastructure that might have a negative effect on the VNF.

There is still no substitute for running real cloud resource instances and testing their performance and behavior before depending on them in the live network. A best practice would be to run the validation tool, and only if it provides positive results, continue to actual resource testing. The actual verification of performance may require some time to run, and there is no point in investing this time if the NFVi is not configured properly.

The verification can detect problems that are not evident in the configuration. Consider the case where not enough CPU cores were allocated to the virtual switch on the hypervisor. Only by running VMs that push a high rate of traffic through the vswitch can this problem be detected.

Another possible issue may be some networking policy configuration that blocks certain types of traffic in a switch or router in the data center. While the examination of the OpenStack and hypervisor configuration parameters by the validation tool may not have indicated a configuration issue, attempting to deploy a VNF in this environment could still lead to a malfunction. The root cause in this case may be harder to pinpoint. Without pre-verification the time spent on deployment, configuration and troubleshooting might be proportionally longer.

Being able to verify the functionality and performance of the NFVi can substantially reduce the time to deploy a VNF on a new NFVi. Performance issues in particular are hard to troubleshoot because the actual symptom observed can be very different from the true cause. Our experience shows that a VNF component might report network connectivity issues with another component, where in fact the real problem is lack of sufficient CPU cycles to handle connectivity keep-alive messages.
How to perform verification

Verifying the infrastructure involves creating and running virtual resources that behave in a manner similar to the actual VNF components. Such resources may include:

- VMs using the same flavors as the actual VNF component VMs
- Virtual networks using the same configurations as the VNF (VLAN, tunnels, etc.)
- Storage partitions similar to the ones used by the VNF
- VMs and networks connected in the same topology as the VNF or associated network service.

Once the resources are created, they should be driven to behave in a manner similar to the VNF. That may include:

- Stressing the virtual CPU cores
- Running high network throughput in various topologies. The NFVi might behave differently for network traffic on the same host, between hosts, or in different availability zones. All the relevant topologies must be tested in order to avoid surprises after the VNF has been deployed.
- Using the communications protocols used by the VNF. It is not enough to perform just basic User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) tests. VNFs tend to use a unique set of protocols such as Stream Control Transmission Protocol (SCTP). The NFVi may allow standard UDP/TCP traffic but behave differently for SCTP.

The verification tool should perform the above tasks and have the following characteristics:

- Simplicity of operation. While the actual tests to run might get complicated, the end user should be presented with a simple and clear UI that would make running the tests a simple task.
- Full automation and scheduling capabilities. Running verification tests can be a lengthy process that involves creating virtual resources, configuring them, running tests over time and collecting the results. The end user should have a “fire and forget” type of interface that will initiate the tests with one click and then run them automatically without the need for human intervention.
- Clarity of results. The tests might get complicated but the information presented to the network operator should be clear. It should indicate if the NFVi is ready to run the VNF.
- Ability to drill down and get all relevant information in case of a detected issue.
What to do with verification test results

Running verification tests on an NFVi could yield an enormous amount of data. While it has advantages, it presents a challenge when trying to answer the simple question, “Will my VNF perform as expected on this NFVi?”

The key to making the results of the verification actionable is to compare them to some reference data. As an example, running the tests might indicate that the VMs can make up to 200 input/output operations per second (IOPS), but what does it mean for my VNF? In some cases, there is a clear requirement from the VNF to get a minimal IOPS value. But what happens if my VNF does not have any known requirements for IOPS? The solution in this case could be in comparing the results of the tested NFVi to some known NFVi performance. If I know that my VNF functioned properly on an NFVi instance that provides 150 IOPS, I can predict with a high level of confidence that it will function properly on the NFVi with 200 IOPS (assuming the other performance criteria are either met or exceeded).

Figure 2. A verification tool test result report card

What comes after validation and NFVi verification

While the steps described in the previous sections can provide a high level of confidence that the VNF will run on the NFVi with the expected performance, there is still no guarantee. The behavior of the VNF itself might be slightly different from that of the resources used for verification. In some other
cases, the results of the verification might not be conclusive. The tested NFVi may over-perform in some aspects but might under-perform in others. Some functionality of the VNF may not be easily simulated and can only be verified when the VNF itself is running. Of course running the NFVi verification prior to testing the VNF can save time and effort and so we advise you to deploy and test the VNF only after you have performed the previous steps. It could be argued that you could start immediately with testing the VNF; however if issues arise, pinpointing the root cause might not be a straightforward process.

The way to ensure proper functionality and performance of the VNF is to perform verification by means of VNF-specific tests. There is no other way to know for sure if your virtual router is routing properly or if your IP Multimedia Subsystem (IMS) core is handling subscribers until you run real network traffic against them. The verification process should be automated so that it can be repeated every time the VNF is deployed, migrated or scaled, or in case of debugging whenever a problem is suspected.

The VNF verification tool should provide the following capabilities:

- Run VNF-specific tests, using the relevant VNF test tools. For a virtual router, that may be a packet generator; for an IMS service, it may be a SIP call generator.
- Perform a predetermined set of tests in an automated manner without requiring human intervention
- Be extensible to allow future coverage for testing new types of VNFs or network services
- Provide the end user with clear results that will indicate pass/fail.

Role of the verification tool

The verification tool plays a slightly different role than standard VNF test tools and assurance mechanisms. It is not aimed at replacing existing test tools or existing service assurance elements in the OSS. The main purpose of the tool is to indicate that the VNF deployment process completed successfully, and to help in troubleshooting when the VNF fails. Other testing tools and verification mechanisms will continue to co-exist. The verification tool may be integrated with the VNF Manager in the future or even with the Network Service Orchestrator. It has a clear role in the deployment and life-cycle management process of a VNF and that of an end-to-end network service.

The verification tool may provide integration of existing VNF-specific test tools, in order to automate the end-to-end process and present the user with a clear view of the test results. As an example, for a Voice over Wi-Fi® service, it may be necessary to run Evolved Packet Core tests, as well as IMS tests. The results should be combined to provide an indication of the service readiness.
Deployment and operation of a VNF

Using the right building blocks for NFV - such as NFVi, Virtualized Infrastructure Manager (VIM), VNF Manager, NFV Orchestrator and of course the VNF - is crucial for a successful service. However, as this paper shows, that is not enough. Only by using the right process with the right set of tools can a VNF become part of a production service. The difference between success and failure begins with realizing that there are differences between infrastructure instances. There may even be differences in the functionality of the same infrastructure at two different points in time. While careful design and strict rules for operation can minimize those differences, they will not eliminate them. Therefore, there will always be a need for validation and verification processes, which should be appropriately automated.

Figure 3 shows a typical cycle of deploying and operating a VNF.

Figure 3. VNF deployment and operation cycle

![](image)

The process has a cyclic nature because the task does not end with a successful deployment. The condition of the infrastructure might change over time due to expansion, replacement, or simply aging. The VNF itself is dynamic and changes over time through events such as scaling out, migration and software updates. All these constant changes require the validation and verification processes to be repeated on a regular basis and hence the need for automation.

Only through automation can the process become repeatable and provide meaningful results. Performing the process using automatic tools removes uncertainties by eliminating the human factor. It ensures no aspect is neglected and makes the task effortless, meaning it is more likely to be repeated whenever required without undue burden.

The techniques and tools described in this paper can make the VNF deployment and life-cycle management much more predictable, resulting in quicker time-to-market and less wasted effort. Through our deep historical understanding of VNFs and the work of onboarding a wide variety of very different VNFs, we have developed both the expertise and the required tools to successfully repeat this process.

For more information on how Nokia may help you with the process, please visit our NFV page.
Acronyms

COTS    commercial off-the-shelf
ETSI    European Telecommunications Standards Institute
IMS     IP Multimedia Subsystem
IOPS    input/output operations per second
MTU     maximum transfer unit
NFV     network functions virtualization
NFVi    NFV infrastructure
OSS     operations support system
SCTP    Stream Control Transmission Protocol
SDS     software-defined storage
SDN     software-defined networking
TCP     Transmission Control Protocol
UDP     User Datagram Protocol
vCPU    virtual CPU
VIM     Virtualized Infrastructure Manager
VLAN    virtual LAN
VM      virtual machine
VNF     virtualized network function