Model-based orchestration in NFV

Orchestration has become a hot topic in Network Functions Virtualization (NFV). To achieve the goals of NFV, it is critical to understand the role of orchestration. A simple approach to orchestration based on scripting or orchestration languages is insufficient for running an NFV-based network efficiently. For this reason, Alcatel-Lucent favors a model-based approach that is scalable enough to realize the NFV promise of OPEX reduction through automation and coherent management processes. Models of virtual network functions, NFV infrastructure, or other entities are inherently multi-purpose. As a result, once these entities are described in a formal way, the descriptions can be used for installation, assurance, and other lifecycle operations.

About the NFV Insights Series

NFV represents a major shift in the telecommunications and networking industry. NFV applies virtualization and cloud principles to the telecommunications domain. Until recently, this approach appeared to be impossible due to the stringent performance, availability, reliability, and security requirements in communication networks. Many service providers are now keen to implement NFV to help them gain an advantage through automation and responsiveness in order to deliver an enhanced customer experience while reducing operational costs. This series of whitepapers addresses some of the key technical and business challenges on the road to NFV.

Alcatel·Lucent

Table of contents

Many would contend that the most desirable goal of NFV is orchestration and automation because it delivers a new level of efficient network operations and service agility. In the old days of physical boxes, deploying a new network solution required procuring equipment, shipping it to the right locations, finding space in the central office or data center, as well as installing and cabling the equipment—among many other tasks. Each type of equipment required its own spare parts and the system capacities had to be carefully planned ahead. Capital had to be invested in excess capacity to prepare for future growth and software upgrades had to be completed overnight to minimize service impact.

Due to the fact that virtualized network functions (VNF) are pure software, the automation and orchestration of operational processes is more doable with NFV than with physical network elements. As described above, a deployment workflow with VNF can be fully automated. Indeed, a business case study [1] showed that the cost of repair, software maintenance, capacity management and other operational processes can be reduced by up to 85 percent.

Orchestration

The term orchestration is not new in the domain of computing but has become increasingly popular, especially in cloud computing. Of course, orchestration in computing draws an analogy with the musical domain where orchestras are led by conductors. A composer produces a musical score and a conductor uses it to ensure that musicians play together and in harmony.

Choreography is a related concept that has also been imported into the computing domain. Whereas an orchestra is perceived to be under the direct control of a leader, choreography emphasizes the guidance given to a group of semi-autonomous actors. These actors, for example, could be dancers who react to each other rather than obeying a single leader.

Orchestration in computing originates in the IT domain of programming-in-the-large and describes the automated arrangement, coordination, and management of complex computer systems, middleware, and services. It is a software technology that uses a formal language to specify and orchestrate actions to be executed by one or, more typically, by different software systems analogous to musicians. Based on the scripts and models expressed in such languages, an orchestration engine calls software system APIs, which are capable of executing individual actions. An action, for example, might consist of spinning up a virtual machine in a data center or changing the configuration of a VNF. Orchestrated operational processes/workflows can be either fully automated or include human (manual) intervention steps and decisions. For example, human approval may be required before a major upgrade or capacity expansion is executed.

The term orchestration is often associated with specific orchestration languages, such as Web Services Business Process Execution Language (WS-BPEL), an XML-based language, and Business Process Model and Notation (BPMN), a graphical notation for business processes. For NFV, scripting languages are often used, such as Python and operation system shells.

What is NFV orchestration?

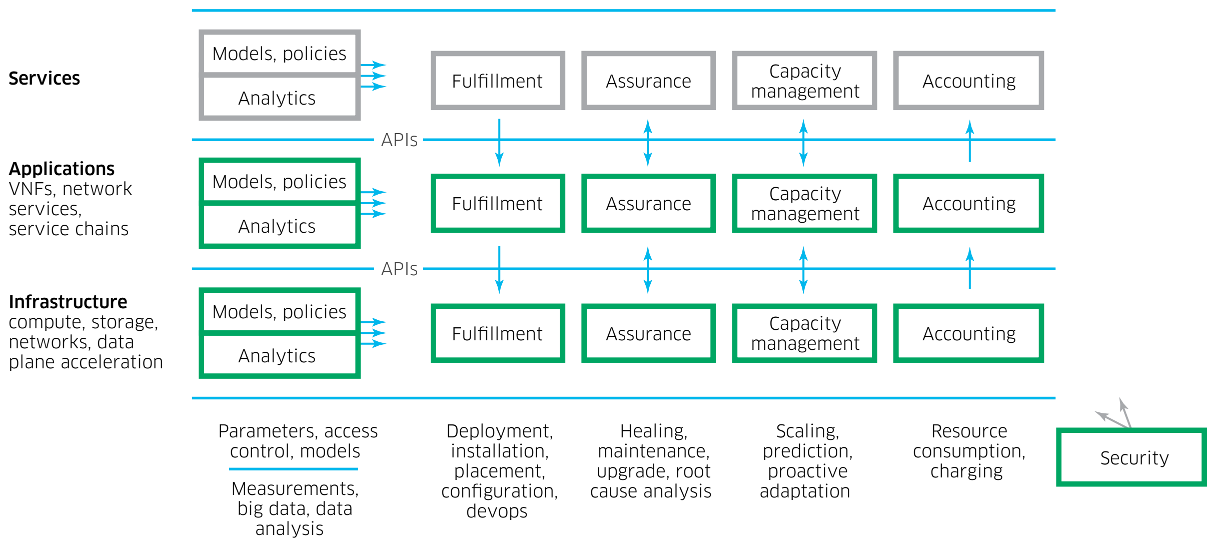
NFV was initiated with the goal to reduce the capital expenditure for network equipment, get out of the rut of traditional network operations, and increase service agility. Due to the fact that network functions are virtualized and implemented in pure software, NFV has the potential to support a new operational approach with significantly fewer manual processes and a higher degree of automation. Traditionally the deployment of a new service required the physical deployment and unique configuration of boxes at various places in the network. With NFV, it is a matter of software deployment and configuration on standard COTS hardware—all of which is operated remotely.

There are different opinions about what NFV orchestration really means. Some think of it only as the automated deployment of the VNFs. However, orchestration is not limited to deployment; the concept of orchestration should be applied to the complete VNF and network service lifecycles including: onboarding, test and validation, scaling, assurance, and software maintenance.

Automated elastic scaling is one of NFV's first orchestration opportunities that has been identified, implemented, and demonstrated. NFV's easy scaling capability minimizes the risk of service unavailability due to capacity shortage. As a result, service providers need not provision large spare capacities for specific types of equipment in the face of uncertainty of service uptake; instead, capacity can be allocated on demand from a pool of general purpose resources. However, automated scalability is not a trivial concept. Firstly, VNFs must be designed to be scalable, using a load balancing approach with stateless application servers. Secondly, without precautions such as specific resource policies, applications can run out of control, "eating up" all NFV resources, thus impacting other services.

In the service assurance domain, orchestration can be applied to the diagnostics and remediation of failures in layered systems of interacting components. Because each layer should comply with performance and availability constraints, alarms should be generated at each layer. In case of a failure, then, it is likely that alarms will be generated by multiple layers of the architecture, including the infrastructure, application, and service layer. To be able to take the right remedial actions, it is important to identify the alarm that represents the root cause of the failure. With this knowledge, an assurance orchestrator, for example, might take remedial action by restarting a network interface card when it detects that the card is the root cause of a number of failures.

Figure 1. Operational processes at different NFV layers


Services
Models, policies
Analytics
Fulfillment
Assurance
Capacity management
Accounting
APIs
Applications
VNFs, network services, service chains
Models, policies
Analytics
Fulfillment
Assurance
Capacity management
Accounting
APIs
Infrastructure
compute, storage, networks, data plane acceleration
Models, policies
Analytics
Fulfillment
Assurance
Capacity management
Accounting
Parameters, access control, models
Measurements, big data, data analysis
Deployment, installation, placement, configuration, devops
Healing, maintenance, upgrade, root cause analysis
Scaling, prediction, proactive adaptation
Resource consumption, charging
Security

The *VNF lifecycle* is not the only candidate for orchestration. The *NFV infrastructure* lifecycle and the service lifecycle are candidates for orchestration, too (Figure 1). Because an NFV infrastructure needs to be much more distributed than a highly consolidated IT cloud, automation of NFV infrastructure operations is a critical part of any efficient NFV solution. Taking a simplified view, we can consider three layers. The lowest layer is the infrastructure layer with the compute, storage, and networking hardware and software. The next layer is the application layer with its VNFs, network services, and service chains. And at the top is the service management layer with the customer-facing services (Figure 1).

At each layer, there are similar types of operational processes, including fulfillment, assurance, capacity management, and accounting processes. All of these processes may be governed by policies and may be supported through data analytics. Also, the processes at the different layers are interdependent. For example, a fulfillment process at the service layer can trigger a fulfillment process at the application layer (deployment of a VNF) and eventually the deployment of more NFV infrastructure.

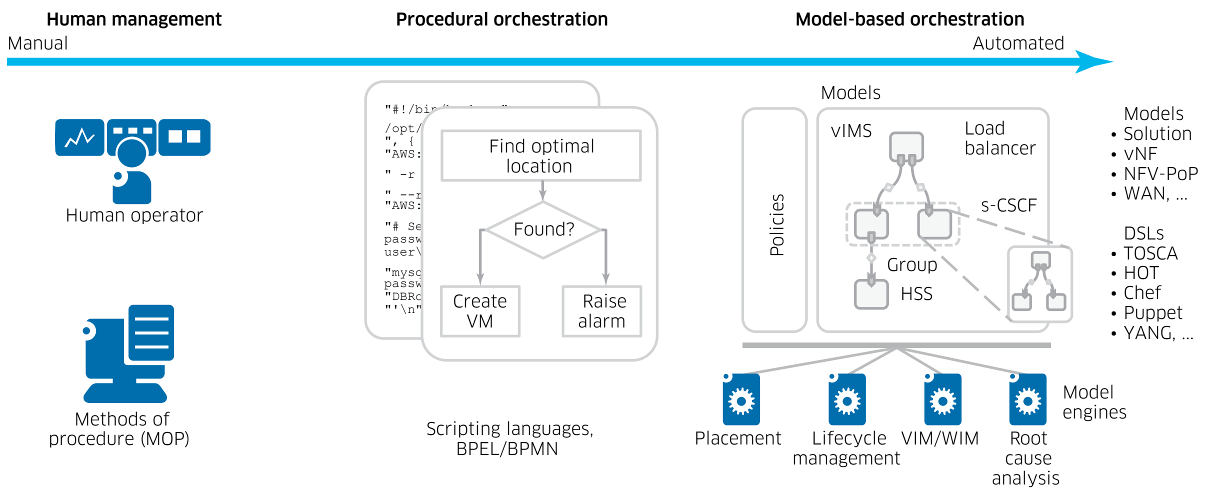
For example, a business service may comprise virtual private networks, security appliances as well as storage and voice services. To deploy the business service, an orchestrator directs several specialized components to allocate and configure resources for the service:

- An NFV virtual infrastructure manager allocates storage volumes and create the virtual security appliances and voice servers.
- An SDN controller creates virtual networks and the service chains to steer data traffic through the network elements in the right order.
- An element management system configures the network elements to work together properly according to customer policies.

From methods of procedure to model-based orchestration

More fundamentally, it is important to ask: What is the right approach to orchestration? (Figure 2)

Figure 2. From human management to model-based orchestration



Human management: In many networks, operators have developed massive methods of procedures (MOPs) that document and describe operational processes. Often, these MOPs have been executed manually by operations staff, which can be an error-prone process.

Procedural orchestration: Following human management, operators began developing scripts in order to automate procedures. But it soon became evident that simple scripting was neither scalable nor maintainable as more and more network elements and procedures were being automated. Consider, for example, an application that changes over time and requires the addition of a new VM. Or take the case of a service provider that needs to change its policies for higher availability or security. In both examples, all scripts would have to be changed in order to include the new VM and the new policies. With different scripts for each application, it would also be difficult to achieve coherent monitoring and lifecycle management policies, as well as error handling.

Model-based orchestration: Due to the challenges of human management and procedural orchestration, Alcatel-Lucent is promoting a model- and policy-based approach to orchestration. In this approach, VNFs and NFV infrastructure are described through domain-specific languages and different engines automate a variety of operational processes based on these models. A major advantage of the model-based approach is the possibility for different engines to use the same model for their respective purposes:

- Deployment/placement
- Capacity management, scaling
- Healing and root-cause analysis
- Maintenance/software upgrades
- Agile development and operations (devops)

For example, a deployment engine knows how to take a model of an NFV application with VNFs and service chains (represented in a VNF descriptor) and deploy that application on an NFV infrastructure while satisfying a set of policies and parameters. A capacity management engine monitors application usage and scales out or scales in the application, as described in the model. To give an orchestration example at the infrastructure layer, another deployment engine takes a blueprint of a cloud node and installs the node software on a bare metal cloud node delivered to a new NFV point of presence.

In reality, a purely descriptive model may not always be sufficient. Some procedural process descriptions may still be necessary to capture the specifics of certain applications, leading to a descriptive/procedural orchestration.

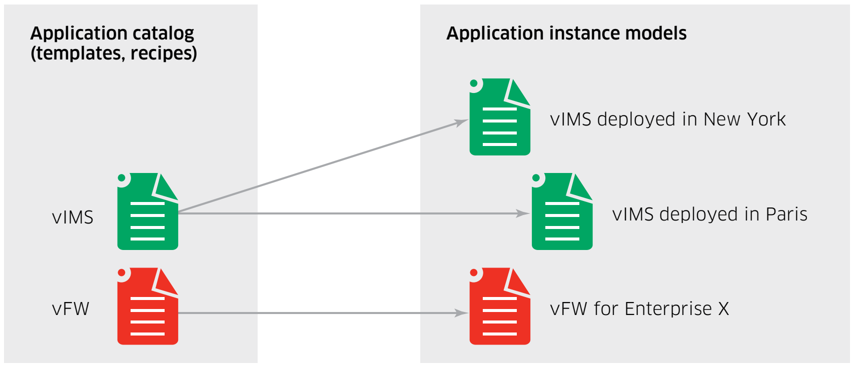
The role of analytics in orchestration

Orchestration is often more complex than simple condition action rules. Advanced orchestration relies on extensive data about the current and historical state of the NFV solution. Using data analytics, an orchestrator can detect application usage trends and trigger the deployment of additional service capacity or remove such capacity when it is likely no longer needed. Data analytics also plays a role in the execution of operational processes. For example, analytics can be used to determine the optimal place for a VNF, or it can be used to identify critical network states that need to be remedied. In the security domain, analytics play an important role in identifying suspicious attack patterns.

Templates versus instance models

In a model-based NFV orchestrator, two types of models exist. The first type of model – called templates in this paper (also known as recipes or blueprints)—describes types of applications, infrastructures, or other resources that can be instantiated/deployed any number of times in an NFV environment. An ETSI VNF descriptor is such a type of model. Typically templates are parameterized allowing operators to specify different values for these parameters, such as size or location. The process of adding new templates is called onboarding. The set of onboarded application templates forms an application catalog from which operators can choose applications to deploy.

Figure 3. Application catalog with templates and application instance models



Once a template has been onboarded onto the NFV Management and Orchestration (MANO) component, the application that it describes can be instantiated and deployed many times in different places and with different parameters. As well, the NFV platform creates a second type of mode—an instance model for each deployment. At the time of deployment, these models are initialized from the templates but also contain runtime values that change over time. For example, the location of a virtual resource changes when it is migrated from one server to another. Resources may be scaled by assigning more or less server and storage resources. Resources may also be in different states, such as running, stopped, or failed. The set of these latter models represents the resource inventory of the NFV network.

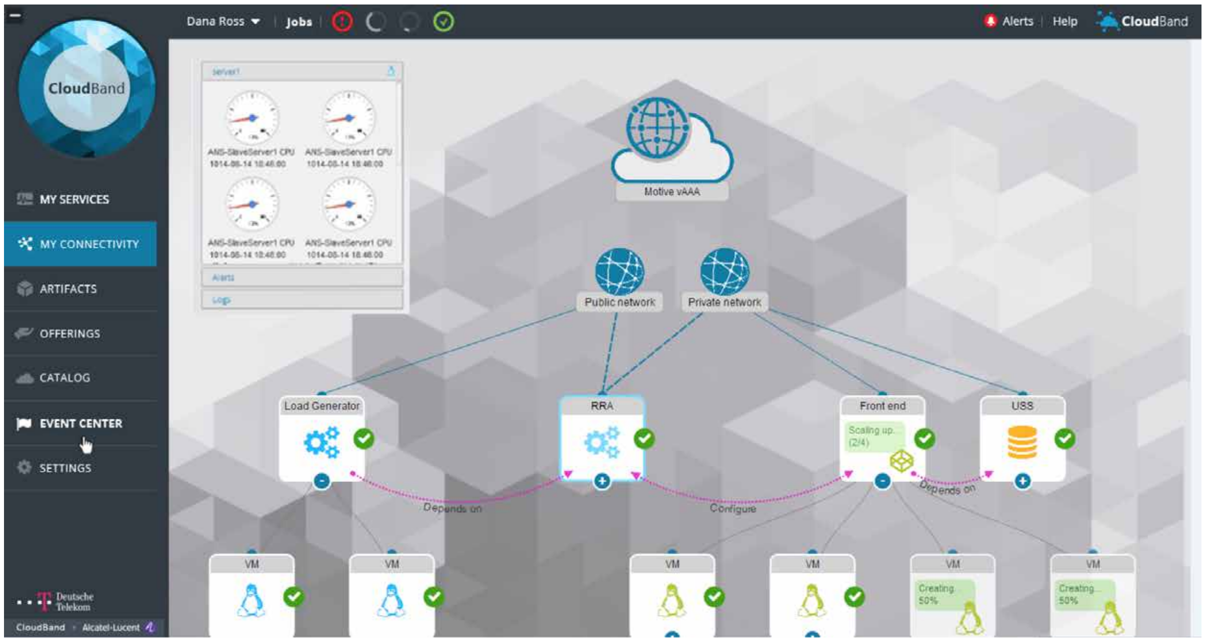
Domain-specific languages for cloud orchestration

Part of the NFV vision is the establishment of an open, efficient VNF marketplace where service providers can pick and choose the best network functions from any vendor to serve their purposes. This kind of marketplace will only work if VNFs can be easily onboarded onto service provider NFV platforms. A critical element here is the VNF descriptor—a model of the VNF's components, requirements, parameters, and lifecycle operations. To make sure that these models can be understood by different NFV platforms, they need to be expressed in a commonly accepted language that represents the concepts and artifacts of NFV applications and NFV infrastructure. The ETSI NFV specification group has done a great job in its Phase 1 effort to define an architectural and conceptual framework. But it is not yet enough. Clear guidance on standardized VNF descriptors is still missing.

To address this challenge, one possibility would be a fully generic language such as XML. However, to achieve interoperability among different NFV implementations, common NFV concepts should be represented consistently for all vendors. That is why domain-specific languages, such as HOT (OpenStack Heat Orchestration Templates) and OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications) are being defined for this purpose. These languages already contain terminology for NFV and cloud concepts such as hosts, VMs, storage, networks and software applications. To make the models usable for a variety of purposes, they are mostly descriptive with only a few procedural elements, if required.

Figure 4 shows a specific NFV application: Motive Authentication, Authorization and Accounting. The application is modeled as a service template with four types of server resources - Load Generator, Radius Routing Agent (RRA), Front End, Universal Session Store (USS) - and two network nodes - Public Network, Private Network. The figure illustrates how multiple instances of a node type can be instantiated on multiple VMs. For example, the front end type is running on four VMs. The figure also shows how node types are connected to networks. These relationships enable the automation of lifecycle operations, as will be shown below.

Figure 4. A visual representation of a model for a AAA NFV application



Applying models

As noted, one of the major advantages of a model-based approach to orchestration is that the same model can be used for multiple purposes.

Using models for application deployment

Deployment is probably the first use of application models that comes to mind. An orchestration engine—for example a VNF manager—processes a VNF descriptor to instantiate components at runtime, and it uses the relationships between components to derive the order of component instantiation. For example, during the instantiation of a two-tier application that includes a web application that depends on a database, an orchestration engine would first invoke the ‘create’ operation on the database component to install and configure the database. It would then invoke the ‘create’ operation of the web application to install and configure the application, (which includes configuration of the database connection).

Using models for visualizing application state

In a virtualized environment, operators cannot directly see the state and location of the network functions. This is only possible with visualization tools. Figure 4 shows an automatically generated visualization of the Motive Authentication, Authorization and Accounting application based on the associated VNF descriptor. Operational state and resource usage can be highlighted. For example, the visualization shows that the front end node is in the process of being scaled up.

Using models for cloud node deployment and management

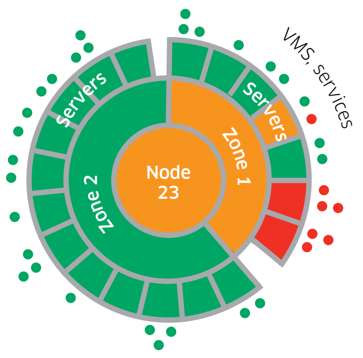
Service providers will deploy cloud nodes differing in hardware vendor and model, CPU capacity, storage and input/output capacity, according to their preferences and virtual application mix. Some servers will provide magnetic rotating disc media; others will provide solid-state discs, and still others will use hardware accelerators. Due to the distributed NFV infrastructure, cloud nodes will be frequently added in different locations. A model-based approach to manage cloud nodes is best able to handle these dynamics. A generic infrastructure manager can use such models to deploy and commission new nodes of different types in an automated way. The models can also be used in capacity management and for optimal placement of NFV applications.

Using models for root-cause analysis

In an NFV environment, applications run on a shared and geographically distributed infrastructure. Any failure may be due to a problem in the infrastructure (a failed server, an overloaded network) or due to a problem in the application software or its configuration. Often different teams are responsible for application and infrastructure. This makes it important to quickly find the root cause of the many alarms generated from the different components and levels of the NFV architecture.

The process of root-cause analysis can benefit from a variety of models. A resource dependency graph (Figure 5) can identify the applications affected by a server or network failure. This graph shows a cloud node with its resources grouped into two zones. Each zone includes a number of servers, and each server is virtualized to offer a set of VMs and software components running within them. The graph combines information from infrastructure models and VNF models and helps to identify applications affected by a resource failure.

Figure 5. Resource dependency graph (shown in a sunburst diagram)



More advanced alarm correlation rules model fine-grained dependencies between physical or virtual resources. For example, the failure of an OpenStack Ceilometer agent on a server might be the root cause of a failure to get metrics for a set of VMs.

When likely root causes have been identified, NFV platforms orchestrate the service restoration process. For example, a network interface card can be restarted or redundancy can be restored by creating a new application server instance on a different server.

Who creates the models?

For any large-scale NFV implementation, it is critical to minimize the work needed to create the orchestration models. Generally, the VNF provider or infrastructure vendors deliver the models along with their software and hardware products. Service providers can then adapt these models to fit their specific environments specifying parameters or defining subtypes of the model entities and supplying network templates.

Advanced model-based orchestration with CloudBand

CloudBand is an NFV platform composed of two parts. The CloudBand Management System fulfils the ETSI MANO role, including a generic VNF manager, and the CloudBand Node provides and manages a set of compute, storage, and networking resources that can be rapidly deployed at any location in the network. CloudBand's model-based architecture is essential to achieve the desired openness and multi-vendor support at all levels. CloudBand uses models to simplify orchestration of VNFs and the NFV infrastructure. CloudBand Insights records a wide variety of system data in a centralized big data store and provides the corresponding data analytics. For example, this data is used in model-based, root-cause analysis and provides a real-time view of the system and application state. Whenever possible CloudBand uses standardized modeling languages to ensure that NFV applications from members of the CloudBand Ecosystem—but not limited to it—can be quickly onboarded and put into service their network functions.

Conclusion

Orchestration is a capability critical for achieving the goals of NFV. Simple script-based orchestration is insufficient to run a NFV-based network efficiently. As a result, Alcatel-Lucent CloudBand has implemented a model-based approach that is scalable enough for realizing the NFV promise of OPEX reduction through automation and coherent management processes. Models of VNFs, NFV infrastructure, or other entities are inherently multi-purpose; once these entities are described in a formal way, the descriptions can be used for installation, assurance, and other lifecycle operations.

References

1. NFV Insights Series: Business case for moving DNS to the cloud, http://resources.alcatel-lucent.com/?cid=178476
2. TOSCA Simple Profile in YAML Version 1.0 Working Draft 05

Additional reading

NFV Insights Series (http://resources.alcatel-lucent.com/cwf/cwf544)

- The right SDN is right for NFV
- CloudBand with OpenStack as NFV Platform, a joint Red Hat/Alcatel-Lucent white paper
- Why distribution matters in NFV, a joint Telefonica/Alcatel-Lucent white paper
- Providing security in NFV - Challenges and opportunities

Glossary

BPMN	Business Process Model and Notation
Devops	A methodology for integrated development and operations
ETSI	European Telecommunications Standards Institute
HOT	Heat Orchestration Templates
MANO	Management and Orchestration
NFV	Network Functions Virtualization
TOSCA	Topology and Orchestration Specification for Cloud Applications
VM	Virtual machine
VNF	Virtualized network function
WS-BPEL	Web Services Business Process Execution Language