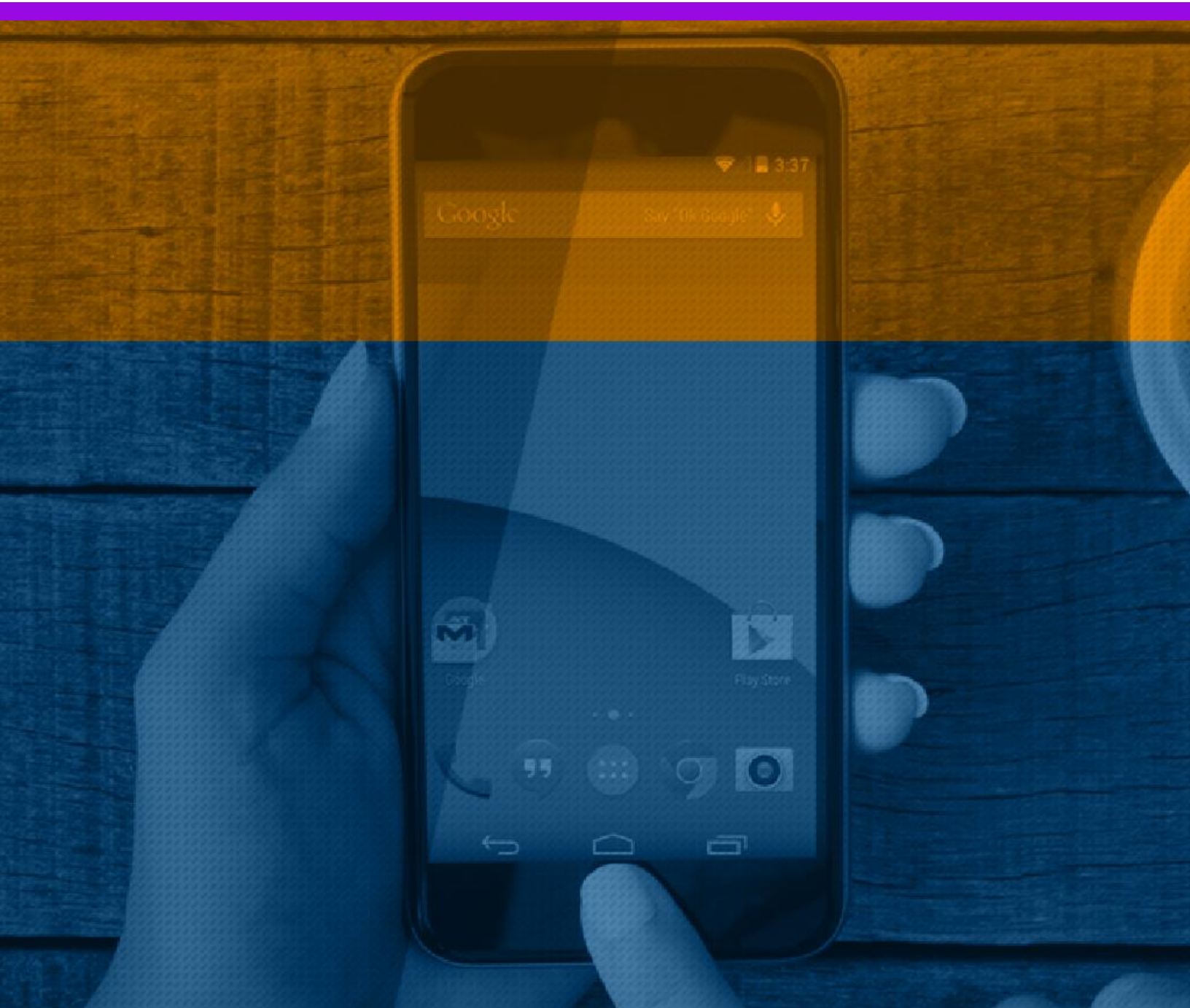

White Paper

Cloud Native VNF Operation Automation

By: Roberto Muggianu, Johanna Nieminen (Telia); Paul Brittain (Metaswitch)



Contents

1	Foreword by Mauro Costa, Director, Network and IT Infrastructure, Telia Company	5
2	Foreword by Martin Taylor, Chief Technology Officer, Metaswitch	7
3	Executive Summary	8
4	Glossary	10
5	Introduction	11
6	Target Scenario	12
7	Desired Lifecycle Behaviour	13
7.1	Deployment	15
7.2	Healing	15
7.3	Scaling/Move	17
7.4	Change/Upgrade	19
7.5	Overload Conditions	20
7.6	Development Cycle	22
8	VNF Design and Environment	23
8.1	Instrumentation	23
8.2	Self-Management and Abstraction	24
8.3	Discovery and Announcement	26
8.4	Scaling and Load Distribution	26
8.5	Efficient Resource Usage	28
8.6	Minimum Resource Footprint	28
8.7	Dynamic Scalability	28
8.8	Resiliency Mechanism	29
8.9	Management vs. Processing Balance	29
8.10	Contention of Resources	30

8.11	State Handling	30
8.12	Rigorous API and Database Versioning	30
9	Additional Considerations	31
9.1	Minimise Special Paths	31
9.2	Licensing	31
9.3	Database Considerations	32
9.4	Security	32
10	Summary of Operational Factors	34
11	Conclusions	35

Tables and Figures

Table 1: Telco cloud scale changes over three years	12
Table 2: Importance of automation aspects in lifecycle operation	34
Figure 1: Lifecycle operations	14
Figure 2: Healing Action Flow	17
Figure 3: Canary Testing	20
Figure 4: Cluster versus instance load	24
Figure 5: Simplifying Operations Automation	25
Figure 6: An example approach to move from 1+1 to N+k	29

1 Foreword by Mauro Costa, Director, Network and IT Infrastructure, Telia Company

Network Functions Virtualisation (NFV) and Software Defined Networking (SDN) are two key concepts that have aggressively spread across the telco community over the last few years. The pursuit of these concepts has created the widely-held expectation that NFV and SDN can make us all a lot more efficient, productive and innovative.

Unfortunately, this is not yet the case.

The goal for telcos is not to virtualise our networks or introduce new software technologies on an ad hoc basis. Rather, it is to make substantial use of such tools to help automate the end-to-end production factory. This will eventually create an opportunity for a more agile service environment and, ultimately, the convergence of networks and IT systems, at least from an infrastructure and process perspective. Resources that are currently locked into manual operational paradigms will eventually be liberated for more useful purposes and we will all find better allocation strategies for our capital expenditure.

To pursue the goal of operational automation, we think a more structured and disciplined way of working is necessary across the industry. Operators will need to learn new ways of designing and sourcing networks and be open to radically changing their traditional approach. Vendors will learn new tools and apply new design principles to their software.

There are clear signs of this new approach in the start-up community, but so far there have been inconsistent efforts to embrace this approach among the traditional, established telecommunications equipment manufacturers.

If we want to make operational automation successful, manufacturers, software companies and telcos must together rise to new levels of leadership in driving this forward.

When we look at the challenge from a network operator's perspective, there are three priorities that need to be addressed quickly and in a structured way:

- Application software
- Platform readiness, performance and efficiency
- Process redesign

Some of the above might have been addressed in the IT space, but networks are different, and without losing sight of our convergence goal, networks deserve some special treatment.

This white paper, written in partnership with Metaswitch, presents our views on the priority of application software. It establishes some key requirements that we will mandate when sourcing virtual network functions (VNFs) and develops a lifecycle management perspective. It captures the (sometimes painful) experience we have had so far in Telia across our automation journey; including some key virtual functions already in operation and serving real customers at scale, and others that we are in the process of launching in the market. Also, it embraces some innovative perspectives that Metaswitch has incorporated in their system design.

All in all, we hope to stimulate a reaction from the industry so that we can come together to improve the prospects for the adoption of network virtualisation.

2 Foreword by Martin Taylor, Chief Technology Officer, Metaswitch

We've been helping network operators to realize the benefits of Network Functions Virtualisation since the very beginnings of the NFV movement, and one message that we hear over and over is that improved operational efficiency is the number one motivation for embarking on the transformation to a virtualised network.

Our NFV journey started before the term "NFV" was coined, back in 2011, when we began the development of an IMS solution designed from the ground up for a cloud environment. Like every other vendor of networking gear, we came from a background where software was packaged as an appliance, and we had a lot to learn about the cloud and about cloud native software design. One lesson we took onboard very early on was that operations automation is a fundamental requirement. If you build a system that is decomposed into several microservices components, which scales out horizontally by instantiating more and more of these components, then you are never going to be able to successfully deploy and manage such a system manually! Clearly, operations automation is something that you must bake into the software from the outset.

Our people are software designers and developers, not telco network operations experts. It's been a huge pleasure collaborating on this white paper with Telia, and we've learned a lot from the Telia team and the perspective they bring on network operations. It's also been very gratifying to have the Telia experts confirm that the approach we've taken to operations automation in the design of our VNFs, learned from best practices in the cloud software world, does indeed provide an excellent basis for the massive improvements in operations efficiency that network operators look for in planning their adoption of NFV.

We hope that this paper, based on the combined perspectives of cloud software developers and network operations experts, will help the industry to better understand how to maximize the most important benefits that NFV transformation can bring: far faster and hugely more efficient network operations.

3 Executive Summary

Network Functions Virtualisation (NFV) radically changes the way communication networks are designed, built and operated with the intention of reducing CAPEX and OPEX costs and increasing service agility. While the early drivers for NFV emphasized the benefits of hardware and CAPEX cost reduction, communication service providers (CSPs) are increasingly prioritizing the potential gains in operational efficiency and rapid service innovation.

Service providers expect NFV to deliver faster development cycles with low operational risk. They also require efficient resource utilization, such as enabling application sharing and the reuse of virtual network function (VNF) components. CSPs also expect a more flexible and agile approach to network and capacity deployment that allows them to start small and scale on demand.

The ability to realize the expected efficiency and agility benefits of NFV is determined by many different components of the NFV system, including the cloud infrastructure, the VNFs themselves and the CSP's processes and organization. However, the biggest obstacle to achieving the full potential of NFV is the lack of maturity in the operational automation of VNFs; automating the lifecycle management of VNFs is a key requirement for NFV.

This white paper focuses on the architecture of a VNF and how the VNF should interact with the cloud environment to simplify the automation of lifecycle management. We provide clear directions on the underlying functional requirements for the VNF to achieve such automation. By embracing the recommendations herein, the entire CSP community can achieve faster, error-free operational management of their cloud-based virtual network infrastructure.

This white paper

- Outlines the target scenario for VNF deployment.
- Examines various aspects of lifecycle management for a typical VNF, identifying desired behaviours that ease operational automation.
- Identifies key requirements on a VNF to achieve target operational simplicity.

Realizing the full benefits of scale enabled by NFV requires adopting these recommendations as soon as possible.

The work builds on the existing European Telecommunications Standards Institute (ETSI) NFV management and orchestration (MANO) framework, but the detailed ETSI NFV MANO interfaces or specific VNF implementation choices are not within the scope of this paper.

Customer-level provisioning, infrastructure and more generic network-related issues are also outside the scope.

This document is primarily aimed at service providers and VNF developers who are involved in the design and operation of NFV networks, and the selection or creation of VNFs running in virtualised networks. It is also useful to strategic decision makers and managers wishing to better understand the operational aspects of NFV.

The authors welcome feedback on this white paper and the evolution towards true cloud native NFV. Contact details are provided at the end.

4 Glossary

Term	Acronym	Definition
Virtual Network Function	VNF	In line with the ETSI definition. It corresponds to a function as defined by the 3GPP standards.
Virtual Network Function Component	VNFC	<p>In line with the ETSI definition. A VNFC represents the lowest granularity for the execution unit of a VNF to describe its expected behaviour when deployed in a cloud environment. Note that</p> <ul style="list-style-type: none">• Nothing is said about the functionalities that build a VNFC — those are implementation details (of no relevance).• A “cluster” is sometimes used as a synonym for VNFC.
Microservice		An execution component with a known and defined interface. It is normally considered to be part of a VNF.
Resources		Features/elements included in the NFVI (CPU, memory, disk and networking).

5 Introduction

The telecom industry is undergoing a fundamental transformation as networks that traditionally relied on vendor-specific appliances are becoming more software-centric and cloud-based with the adoption of NFV. Service providers around the world have started to virtualise network functions and deploy them in cloud environments. As NFV momentum builds, traditional standards bodies such as the European Telecommunications Standards Institute (ETSI) and 3rd Generation Partnership Project (3GPP) have also worked to specify and consolidate NFV requirements, tools and systems.

At the same time, the telecom services market is changing as the 5G era is poised to begin. Data traffic is growing tremendously, in terms of overall volume, per-device throughput and the number of devices connected by the Internet of Things (IoT). Traffic patterns are likely to become less predictable due to new applications and changes in customers behaviour and lifestyles. Deployment topologies for network applications need to address new demands such as very low latency (i.e., less than 5 ms)¹.

Moreover, efficiency (which includes resources used, cost and personnel) and agility are becoming increasingly important. The ability to deploy or reuse network functions within a few minutes and the capability to dynamically change the allocation of resources based on traffic demand or time-of-day factors, are fast becoming must-have criteria for network design.

All industry players need to look at how their systems and processes evolve within this network transformation, and be prepared to take a critical and innovative approach to their NFV plans. This includes the design of the VNFs themselves.

A critical factor in service providers' ability to reap the operational efficiency benefits of NFV will be the extent to which they can automate the lifecycle management of VNFs. One of the attractions of NFV is that network applications can be instantiated rapidly so that new services can be launched in a matter of hours or days, rather than weeks or months (with dedicated appliances). Such speed, and the associated efficiency gains, require automation in the VNF lifecycle management.

But service providers are discovering that not all VNFs are designed to enable operational automation. VNFs need to be architected, or re-architected, based on cloud-native design principles. Simply porting software from a specialized appliance into a cloud environment will not achieve the full benefits of NFV.

¹ Latency requirements will force a deployment model where VNFs run closer to the edge device.

6 Target Scenario

To better frame the extent of the challenge faced by service providers and VNF developers as networks evolve towards 5G, we assume that a moderate-scale national network will face the following scale of change in cloud infrastructure requirements over the next 3 years:

	2017	2020+
Data Centers	2-3, centralized	50, central and edge-distributed
vCPU numbers	<10k	>100k
Component/VM count	<1k	<30k

Table 1: Telco cloud scale changes over three years

While a cloud infrastructure has the potential to provide the needed flexibility, and can therefore be the foundation for network evolution, infrastructure evolution alone will be somewhat pointless without both management layer tools for orchestration and appropriately designed network applications.

7 Desired Lifecycle Behaviour

For operators to reach these targets, VNFs need to use resources efficiently and support operations automation. Automation is the key to simplifying the management of cloud network deployments.

If we simply move software from a physical appliance to execute in a virtual machine (VM), that makes little practical difference to the complexity of managing the network. In fact, doing so without sufficient focus on automating lifecycle operations may add further layers and operational process steps, increasing costs despite replacing proprietary hardware with commodity off-the-shelf (COTS) alternatives. As networks get ever larger and more complex, it is increasingly important that the VNFs themselves are rearchitected to enable simple automation of such operations.

It is also worth noting the difference between orchestration and automation as the two terms are often used in the same context and as synonyms, while in practice they are not.

Automation can happen at many different levels. Some of them might not imply orchestration at all (as defined by ETSI²). It is up to proper system analysis to define the level at which the automation will take place³ and to avoid overcomplicating the design by introducing closed loops at many conflicting levels.

Orchestration, as defined by ETSI, is the management of the cloud infrastructure, balancing the needs of multiple VNFs and applications within defined parameters and a given load. This paper focuses on automation of the lifecycle of a VNF, which will form part of orchestration but is not a complete solution to that wider problem. We do not consider the orchestrator itself but instead concentrate on the VNF.

In the following chapters, we consider the lifecycle of a VNF as comprising the following key operational phases

- Deployment
- Healing
- Scaling/Move
- Change/Upgrade

² In many VNFs, overload conditions can be handled by automation and congestion control but does not require orchestration.

³ This analysis is mainly driven by the dynamics of the underlying traffic.

For each phase, we analyse the ideal VNF behaviour and characteristics needed to benefit from cloud infrastructure features while meeting target operational efficiency.

These operational phases group several of the standardised ETSI operations from the NFV MANO framework (see Figure 1: Lifecycle operations). We have concentrated on the process of deploying the application software cluster, the collection of multiple instances of each VNFC that make up the VNF. [Subscriber-level provisioning is outside the scope of this paper.]

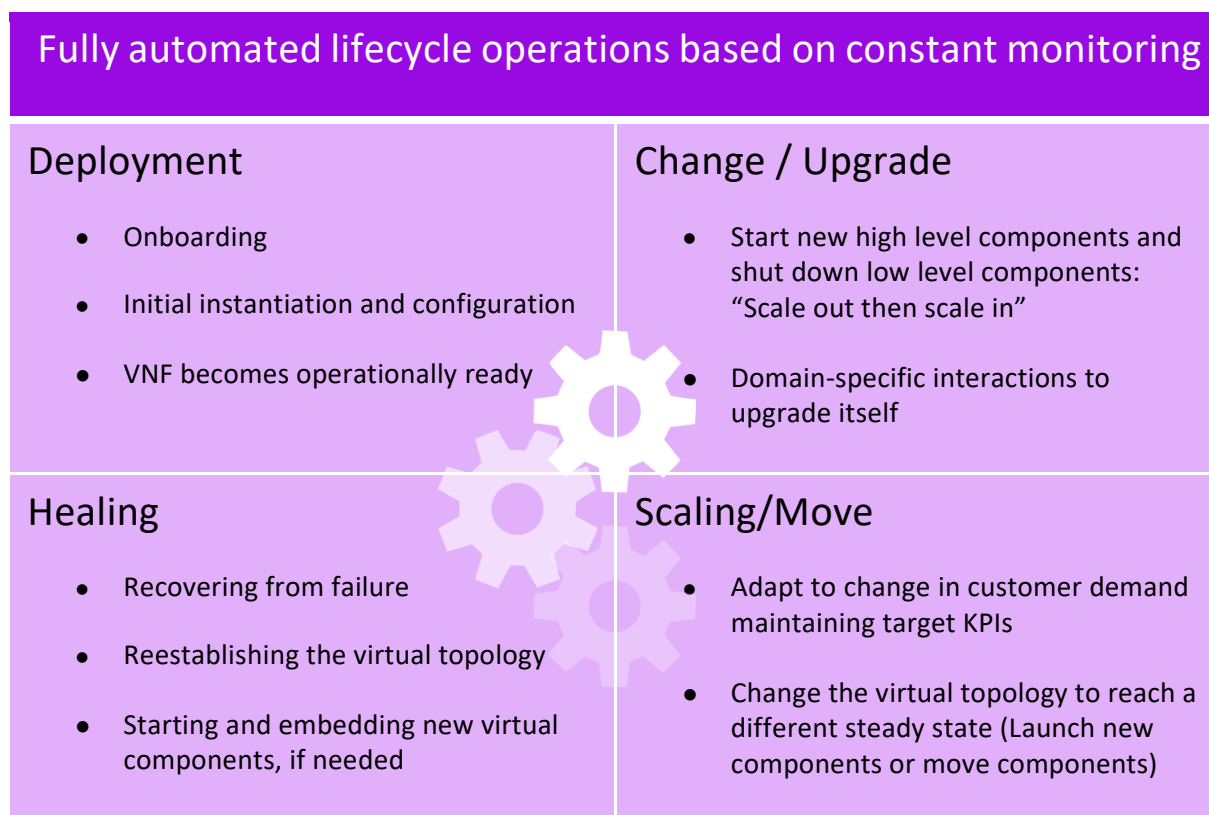


Figure 1: Lifecycle operations

In addition to the ETSI-defined lifecycle operations, successful cloud VNF design must also consider

- Overload conditions
- Development cycles

There are also other cloud operational actions across the entire application/cloud infrastructure/network stack, such as disaster recovery, backup and recovery, or cloud bursting that must be considered, although these are out of the scope of this paper.

7.1 Deployment

This lifecycle operation includes the traditional on-boarding and instantiation, as well as the VNF cluster-level and service-level configuration. Once this phase is complete, the application will be ready to work in its operational environment.

This initial stage of the VNF lifecycle should be completely automated. By encompassing initial service configuration — one of the still largely manual, time-consuming activities for many applications — the service provider is assured that the VNF has been deployed in an error-free manner. It is then ready for subsequent automated scaling or topology changes based on a known configuration model.

This is important today when we have only a few central sites. It becomes even more essential when many more edge sites must be managed in a dynamic way or when more complex operational models are needed — multiple instances of a per-enterprise service hosted by a service provider but managed by the customer, for example

Consider, for example, a URLLC⁴ application running on a service provider's MEC (Multi-access Edge Computing) cloud. Such an application may be dynamically created in any of tens or hundreds of edge sites. If they are to provide the desired low-latency behaviour then they also need careful resource allocation with physical location awareness.

Another example would be a field trial. In this case, speed to market is key and possibly requires deploying in an existing production environment.

In all cases, the common requirement is to minimise operational risk during the service lifetime. Only by successful automation of the initial deployment can a service provider attain the expected flexibility while at the same time reduce the complexity of later operational phases, and therefore the overall operational risk.

7.2 Healing

Per ETSI GS NFV-IFA 010 V2.1.1 (2016-04), VNF Healing is “a procedure that includes all virtualisation-related corrective actions to repair a faulty VNF, and/or its VNFC instances and internal VNF Virtual Link(s).”

ETSI addresses healing from a management layer perspective. Here, we look at it from the VNF perspective.

⁴ Ultra-reliable low-latency communications application

The assumption here is that the VNF is running in a 'steady state' situation and undefined external or internal conditions change the VNF's state. The expected healing behaviour is to re-establish the previous topology as quickly as possible.

The external conditions that force healing to happen might be unforeseen (e.g. a failure) or planned (e.g. driven by the need to deploy a configuration change). Healing relies on a model of the network and VNF, and the ability to continuously perform closed-loop actions to re-establish the desired steady-state.

Healing may be achieved by immediately fixing a failed instance, or more generally, creating a new instance of the failed component. The failure can be detected at different levels and it is the responsibility of proper instrumentation (see 8.1) that these events are signalled in a timely manner to the related components.

Healing is different from resiliency. Resiliency is a largely 'static' concept, which enables a certain function to achieve a defined service level. Traditional resiliency schemas are typically 1+1, N+1 or N+k.

Healing refers to a more dynamic feature where the VNF and infrastructure can undertake operations to re-establish an existing topology, such as recovering to full capacity, by adding a new VNFC to a cluster, replacing that which was removed due to a host failure. The new component may receive a different configuration (e.g. a new IP address) from the cloud infrastructure. To handle this, the VNF should not rely on neither the physical characteristics of the infrastructure nor the expected behaviour of the NFV Infrastructure (e.g. how addresses are allocated). Such design is implicit to the cloud environment.

The VNF shall be able to bring new instances of VNF components into the deployment topology, as needed to re-establish the target state. This implies the ability to properly configure them and integrate with surrounding systems (and if needed with the networking context), irrespective of whether these instances are created at the start of day (like in traditional resiliency models) or more dynamically as fits the cloud. Figure 2 depicts how the healing action flow works in practice.

It is highly recommended that the interaction(s) needed with the management layer during healing is kept as light as possible since this operation does not require any change in the resources allocation and it is 'simply' re-establishing a target topology.

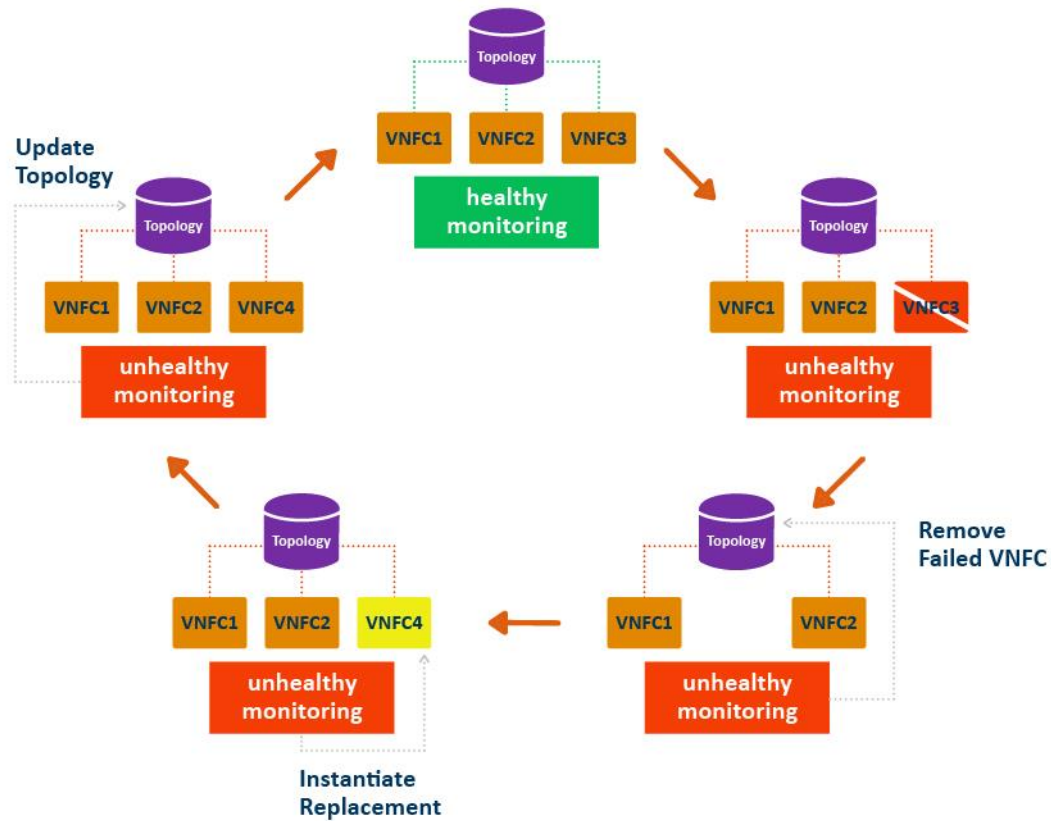


Figure 2: Healing Action Flow

A final word about the concept of healing versus resiliency: under the right circumstances⁵, the healing process might completely replace the traditional static resiliency mechanisms.

7.3 Scaling/Move

This lifecycle operation describes the ability of a VNF to adapt to a change in the demand of the customers being served. This adaptation is achieved by looking to keep a defined target, such as a set of pre-defined KPIs, within an acceptable range or value. The desired behaviour is to do that within the most efficient way possible, from a resources perspective.

The most obvious scenario is when the resource being considered is the CPU. The desired behaviour for scale is to keep the application occupying the lowest number of CPUs. In this case the VNF will act to change its topology to reach a different steady state (as opposed to trying to preserve when healing). This change typically implies a change in the number of components of the VNF.

⁵ Strictly related to the speed of the healing process and application design

If the scale scenario requires the creation of new VNF components, then the same considerations given to the healing operation should apply; with new components being recognized and incorporated into the VNF cluster automatically.

In general, scaling should be a completely automated operation. Efficient scaling with automation requires that

- KPIs must accurately reflect resource use, and be updated in a timely manner.
- The execution time of the scaling actions shall be in line with the dynamic of the characteristic to be controlled, or efficiency may be lost.
- “Scale-in” action is essential for efficiency, but may be more complex than scale-out if the VNF is stateful.
- If the VNF has several distinct components, with different scaling logic based on different aspects of the traffic load (e.g. concurrent registrations vs transactions-per-second), it shall be possible to change them independently.⁶
- The change overhead (on the management of the infrastructure) shall be kept to a minimum to reduce complexity and aid timely reaction to changes in traffic load.

The ideal condition and desired behaviour is where homogeneous components are evenly loaded⁷.

The scaling mechanism should also implement algorithm and policies to avoid instability, such as flip-flopping or overconsumption of resources.

It is worth mentioning that some scale operations may require that components of a VNF are instantiated differently across a geographically distributed execution environment. We refer this as “move.” Move operations might occur to keep a latency target on a set of customers served by the function, or for other reasons.

Beside the specific trigger event, all the considerations about scaling and healing apply also to “move.” Note that the moved component might take a different flavour and scale per different policies⁸ enforced at its new location.

Specific attention should be placed on the rebalancing of customers if they need to be moved as well. This move can happen to newly created or existing components.

⁶ Respecting any cross-dependency if needed

⁷ That is the optimal configuration assuming homogenous components are equally sized.

⁸ The sizing of the moved component, for example, could be completely different

7.4 Change/Upgrade

The upgrade of a VNF is a relevant part of its lifecycle and one that most often needs careful planning because of its complexity and the operational risk implied. It is very important to reduce, as much as possible, the complexity of upgrade to simplify and lower the risk of more frequent development cycles for the function or application⁹.

In addition to software version upgrades, there are occasions when major service-level configuration changes may be required that are similar in scope and impact to an upgrade. Examples include permitted codec combinations, national dial plan changes or fundamental service set alterations.

Traditional approaches to such upgrades/changes relied on the network being made up of multiple independent systems like switches and routers. The change is initially made in a first office application and a progressive roll-out executed after a stable period. Each such upgrade is manual however and the coordinated upgrade of a complete network may take many weeks across multiple maintenance windows.

Cloud deployment offers an alternative, faster and lower risk path to perform these operations.

At the foundation is the capability of a VNF to update its components independently, which can be achieved by replacing the current version of one component with the new one.

This upgrade may be performed in-place, using the same VM and other resources already allocated, or via a scale-out/in operation to add an up-level node to the cluster and then removing a back-level node.

The choice of upgrade style depends on the capabilities and service needs of the specific VNF and its components to be upgraded. In either case, the VNF must be able to operate correctly while at different revision levels.

Where a VNF uses a distributed database for state storage, upgrading the version of a database itself requires more careful coordination to ensure that the system remains quorate in each site, and to ensure consistency in the data during the upgrade procedure.

The cloud also offers the possibility of deploying very small instances of a function quickly and easily. Web-scale players¹⁰ use this to allow them to upgrade a small portion of their user base to a new level initially, monitor that community intensively to validate the upgrade is good, and then apply it more widely. Service providers can apply the same approach by temporarily segmenting their subscriber base during upgrade, and can achieve

⁹ A quicker development time tracks more closely to market demand.

¹⁰ Facebook, for example, use this approach as part of their DevOps process.

this with little or no cost in terms of resources¹¹, reducing the risk of a flash-cut to the new system for the entire user base. This is sometimes referred to as Canary¹² testing (as shown in Figure 3).

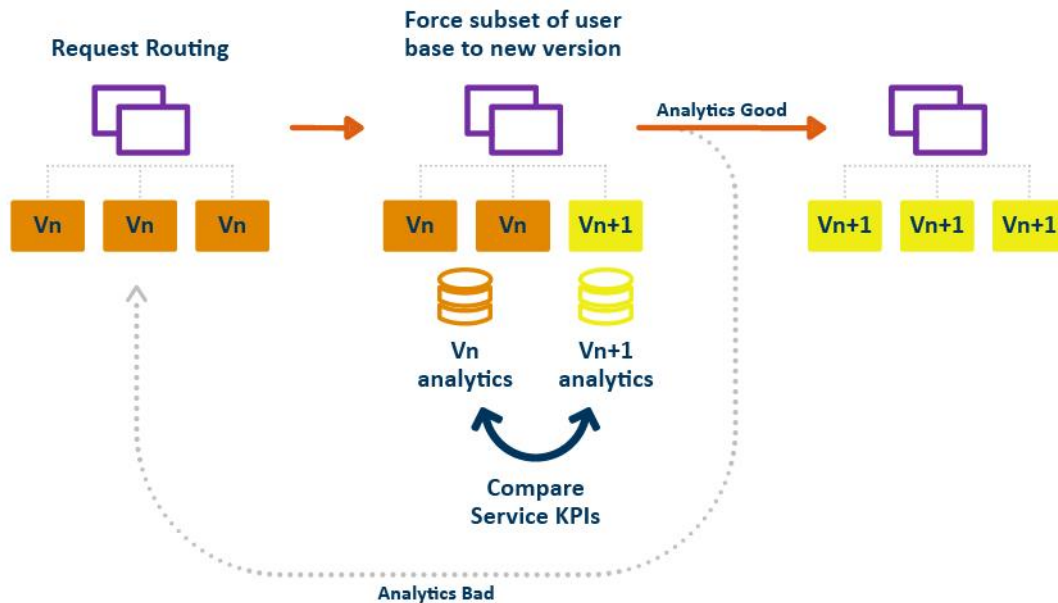


Figure 3: Canary Testing

Developing or replicating an automated upgrade procedure as a part of the on-boarding process to a NFVI/MANO stack for each service provider adds time, cost and risk to the on-boarding process. Ideally, the function should self-manage the complex application domain-specific interactions for upgrade, exposing only easy-to-use application programming interfaces (APIs) to request the upgrade or change.

7.5 Overload Conditions

In an overload condition, the function may need to manage sudden and unusually high demand. In traditional system analysis terminology, this is equivalent to experiencing a step function in traffic load.

This situation typically occurs because of a major network failure where an extremely large number of customers attempts to use the service (i.e. a signalling storm), or in a situation like a DDoS attack.

¹¹ One function is replaced by two whose capacity sums to the original one.

¹² See <http://whatis.techtarget.com/definition/canary-canary-testing>

Network functions should deal with this abnormal situation gracefully, avoiding instability flapping, and recovering in a finite and predictable amount of time.

To achieve these goals:

- Functions must keep working in saturation conditions. For example, using smart tactical overload protections mechanisms to reject some traffic early.
- Functions should be able to accept and use additional resources that the cloud infrastructure makes available to them during overload, but should not pre-reserve dedicated resources for this.

The art of not reserving dedicated resources deserves a closer look. In legacy scenarios there is normally a portion of each node capacity dedicated to absorbing overload and engineering logic is to limit the resource consumptions to a certain threshold¹³. Capacity is engineered for each function separately.

In a cloud environment, the VM resources are available beyond the obvious hardware constraints of a traditional physical appliance. The capacity engineering in such overload situations will be more efficient if the spare resources are treated as a shared pool. In general, one function's action to limit overload will protect some others down the chain¹⁴, though which function is first to experience overload will depend on the exact nature of the load pattern. This phenomenon means that overload capacity can be shared, and potentially even reallocated away from less mission-critical functions (via scale-in if needed). However, each function should expect that resources for overload are contended, and must remain stable in a state of saturation

Automated operational processes form an essential part of overload handling in the cloud — via scale-out/in, for example. In addition, for each VNF and the cloud to remain stable during and after overload, a VNF must

- Intelligently discard load¹⁵, such that it can continue to process some load even under saturation.
- Request additional resources to handle overload only when required.
- Remain stable even if additional resources are not granted immediately.

¹³ For example, CPU load in those conditions limited to a certain level.

¹⁴ For example, an EPC function can protect the elements on the data chain.

¹⁵ Simply discarding 50% of IP packets, for example, is likely to result in no traffic being successfully processed. Protocol-level retries from adjacent elements may then prolong the overload to the point where the entire system is very slow to recover back to steady-state resource use.

- Ensure these additional resources are released promptly as load subsides.

7.6 Development Cycle

These automated lifecycle operations are necessary but not sufficient to achieve radical gains in operational agility. Service providers may also need to adopt more end-to-end process changes like continuous integration. These changes are complex and necessarily tailored to each organization and relevant stakeholders, so we concentrate here on high-level observations relating only to the VNF architecture.

All service providers likely need to review how parts of the complete development cycle are performed, and how they can shorten the path from an a new function request to actual service deployment¹⁶. This effort will give benefits in terms of

- The ability to respond to customers' requirements in the most efficient and timely manner.
- Lowering the overall complexity for a new function by leveraging existing functions instead of building complete new systems from scratch (yet another "box").

This effort will impose certain requirement on the VNFs, processes and related tools.

With respect to the VNFs, in addition to focusing on lifecycle automation the VNF architecture should separate each component with clearly defined interfaces to adjacent components. This puts focus on allowing the reuse of individual components with minimal reworking of those affected by new requirements. This is commonly termed a microservice¹⁷ architecture.

¹⁶ DevOps is the common terminology summarizing this requirement.

¹⁷ See <https://en.wikipedia.org/wiki/Microservices> for a deeper explanation of microservices.

8 VNF Design and Environment

This section describes the key features a VNF should include to meet the target lifecycle behaviors. The aim is to describe those key features without specific reference to either technologies or development patterns.

8.1 Instrumentation

It is critical that all VNFs maintain and make available a complete set of information related to its deployment status and to the service it provides. The dynamic and timely nature of this information is essential for closed-loop automation to be effective.

Examples of relevant information include

- A topology of the functions — components, version(s), IP addresses, relevant identifiers
- Health status
- Load indications as measured or calculated at VNF component level.
- Traffic/performance measurement at application and component levels e.g. messages processed and bandwidth used.
- Service specific information suitable for detailed Quality of Experience (QoE) analyses e.g. measured latency or round-trip time (RTT).
- Key load distribution metrics.

VNFs should make this information available

- Via well-documented, open (royalty free) APIs and, possibly, integrated with most common monitoring and analytical tools¹⁸
- With a sampling time appropriate to the application domain (which can be less than seconds).
- In a reliable way for all the foreseen service conditions, including overload.

It is preferable to have all the information available via a single VNF-level entry point (in addition to or instead of VNFC-level entry point) — e.g. a specific KPI function component

¹⁸ Such as Grafana, Splunk, Elastic stack

— but its failure shall not compromise the VNF’s main functional behaviour¹⁹. This avoids having to re-implement complex logic to combine KPIs in the correct manner as part of on-boarding to each NFVI and MANO.

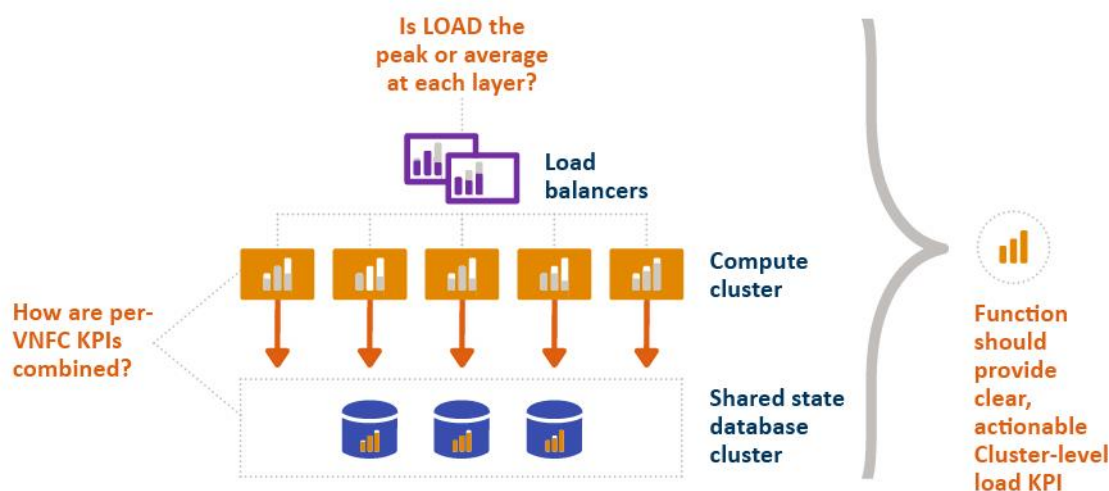


Figure 4: Cluster versus instance load

High quality built-in instrumentation assists all other lifecycle operations and can also replace hardware-based probes. This is particularly true of traffic tracing, which in the cloud context means always-on recording of internal events as well as protocol flows.

With tracing built into a VNF, problem diagnostics are not reliant on having probes in the right place, at the right time. This results in faster resolution without needing to reproduce the problem. It also avoids having to hairpin traffic in/out of each VNF to costly high-performance line trace probes, which may be external to the cloud virtual switch domain, wasting bandwidth and other resources.

Good instrumentation and built-in tracing are expected, if properly implemented, to offer a valuable opportunity to reduce operational complexity and costs. This is a fundamental requirement for any VNF to achieve the desired levels of automation and operational savings.

8.2 Self-Management and Abstraction

VNFs shall not assume a higher level of management (e.g. orchestration in the ETSI NFV MANO framework) or support any level of complexity or granularity of control to manage

¹⁹ This reference point and related functions can be regarded as the control part of the application.

lifecycle operations for the application. Where application domain-specific knowledge is required to complete an action correctly, the VNF itself should handle that complexity.

This level of abstraction is reached with an API is as simple as possible and informative enough to enable a more complicated flow of operation at the orchestration level.

Examples of operations that require more complicated orchestrated flows include

- upgrade, check and roll-out.
- Scaling overflow (when a new VNF(C) instance is required).
- Healing, after one or more host failures.

This approach is similar and extends the ETSI MANO concept whereby the VNF manager (VNFM) provides abstraction of the function towards the upper layer management systems. The exact labelling and split of responsibility between VNFM and VNF is not important so long as the package delivered to the service provider handles any VNF domain-specifics, such as VNFC start-up phasing and clustering, without requiring complex per-deployment scripting.

- ✓ **VNF components pick up basic initial config automatically**
Via Nova metadata/user_data or cloud-init, not via CLI
- ✓ **Components talk to a (built-in) distributed config store**
Massively simplifies the management of initial configuration
- ✓ **Components of the VNF can be brought up in any order**
Components must be happy to wait until their dependencies are met
- ✓ **Interfaces between components are versioned**
Components must be happy to wait until their dependencies are met



Figure 5: Simplifying Operations Automation

8.3 Discovery and Announcement

Efficient cloud deployment models result in a very dynamic network status, with each node and service potentially scaling or being upgraded independently, in addition to network topology changes due to site or cloud failures.

Relying on manual or scripted mechanisms to keep pace with such change is not tenable therefore VNFs should provide mechanisms to manage those dynamics.

Specifically, VNF components should

- Expose and announce changes for their main integration points and adjacent internal components.
- Monitor for topology announcement and act accordingly
- Expose and announce changes to the physical or external components they integrate with²⁰.
- Document failure detection and recovery mechanisms that can be automated via standard actions in adjacent components – for example, retry to the next DNS SRV record for the IMS core.
- Incorporate location awareness and selection and local feedback loops, where needed for URLLC applications; to avoid long feedback cycles if resource/location selection always route to a central site, for example.

8.4 Scaling and Load Distribution

VNFs typically meet demand by horizontal scaling (i.e. by adding or taking down instances of the required components). For this to function well in a cloud with efficient resource usage, VNFs need to

- Spread load across the multiple VNFC instances.
- Evenly distribute the load on homogeneous components under normal traffic patterns.
- Reallocate load during scaling operations.
- Avoid focused overload (such as DDoS) impacting the service for all users

²⁰ For example, an external DNS.

Some VNF deployments may also need to be able to operate in heterogeneous environments, where different-sized components are needed on different cloud platforms deployed in edge and core sites.

It is therefore very important that each VNF implements efficient and intelligent load distribution policies. To achieve this, several aspects of traffic and platform performance need to be considered into the load balancing mechanisms.

The load balancing mechanisms must be purely software based. This is to avoid both bottlenecks and the constraints that come from deploying physical functions (as opposed to using pure software operations).

VNFs should support load distribution across all available nodes within a cluster. This balancing may be pure “equal weight” between instances, or incorporate location awareness, preferring local nodes in an edge site with fall back to central nodes. This should be achieved without need for special processing paths to achieve good load balancing, as this reduces complexity. The techniques used to do this may vary by VNF but can include sharding of the user population statically or dynamically (via short lifetime association of per message/transaction²¹/registration), often coupled with single or multi-horizon routing or anycast (as discovery mechanisms), and the retry mechanisms built into many standard protocol flows.

Regardless of load balancing implementation (for a specific VNF), it must satisfy the following criteria

- Dependencies on adjacent functions, including any internal components using the same mechanisms²², should be standards-based and clearly documented²³.
- Any external systems that form part of this load balancing, such as DNS, should be updated as part of the automated operational processes used to manage this VNF.
- It must be possible to tailor the load balancing mechanism to ensure efficient routing across multiple sites in a distributed system²⁴

²¹ For example, first interaction in a transaction routes to any available node, then subsequent parts of same transaction are routed back to same node.

²² See as well Discovery and Announcement

²³ DNS with retry to alternate SRV records, or anycast, for example.

²⁴ Such as ‘hair-pinning’ media at the edge

- The breakdown of the VNF into sub-components should not introduce undue inefficiencies in the routing of requests between sites or between hosts/VMs within a site²⁵.

8.5 Efficient Resource Usage

It should be a clear target and major performance indicator for any VNF to make efficient use of the cloud resources. In this context, **all the resources** required by all the VNFs are considered. This search for efficiency depends on multiple criteria and, of course, on the efficiency of any single component.

Note that this requirement for efficiency has always been relevant and only becomes more significant in cloud deployments.

8.6 Minimum Resource Footprint

When considering the resource required to effectively run any VNF, it is important to ensure that

- VNFs tightly fit the demand, without waste, at any point in time.
- VNFs scale down elegantly; very small demand should be covered by a very small footprint.
- Component instances are as small as possible, while remaining efficient, for optimal cloud utilization.
- Only needed resources are requested²⁶.
- Granted resources are used efficiently²⁷.

8.7 Dynamic Scalability

The process for adding or removing VNF components should be as simple and as fast as possible. This allows resource usage to closely follow any demand curve.

VNF components should scale per their own needs to meet this target.

²⁵ Particular care is required for user-plan functions to avoid multiple “east-west” routing between subcomponents

²⁶ Assuming no over-commitment is used, which typically is how network functions are deployed

²⁷ If a VNF is poorly implemented, putting it in the cloud will not make it scale well – it will just use more cloud resource!

Note that dynamic VNF scaling over short time periods may require the introduction of technologies that enable very rapid spinning up and down (e.g. containers).

8.8 Resiliency Mechanism

The ideal VNF design for cloud uses N+k active-active redundancy model where both N and k are configurable. If suited to the service delivered by the VNF, the N+k cluster should be able to be spread across sites for maximum hardware efficiency²⁸.

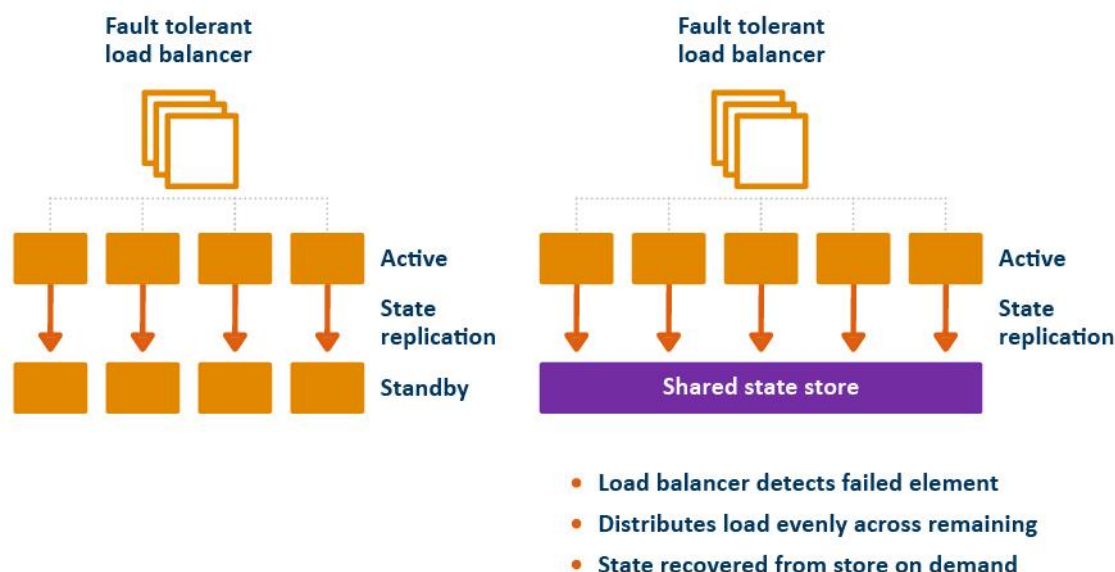


Figure 6: An example approach to move from 1+1 to N+k

8.9 Management vs. Processing Balance

VNFs should maintain a fair balance between resources dedicated to its management and to those processing the demand. In a scale-out scenario, the needs of demand processing should dominate.

²⁸ For example, a 6-site distributed N+k model, with 20% local redundancy as well, uses 1.44N, compared with 4N for traditional 1+1 resiliency both locally and across 2 sites; a saving of nearly 3x.

8.10 Contention of Resources

In general, network functions tend to peak at the same time, preventing use of over-committed resources. With the advent of new cloud and service paradigms (such as network slicing), a service provider may possibly opt to use contention in the following ways

- All VNFs contend for spare resources from a common pool to handle overload²⁹.
- Light overprovisioning for some resource types³⁰ between network slices (especially for higher-layers of the service stack).

VNFs should provide predictable behaviours when working with a contention of resources.

8.11 State Handling

The ideal internal design for a cloud native VNF will usually be to separate state into a distributed database, with stateless compute nodes processing traffic based on, and updating, that stored state. This architecture eases load balancing and separates the scaling of the compute cluster and database processing per the dynamic needs of the traffic model.

This ideal model for stateless processing nodes may not be possible for all VNFs. However, state should always be clearly identified and sand-boxed to a limited set of sub-components within a VNF. This keeps all the expensive operations related to the protection, replica and recovery of the state to a minimum, and eases the automation of operational tasks as far as possible.

8.12 Rigorous API and Database Versioning

Cloud network functions are typically split into multiple subcomponents, or microservices. For this modular approach to work well with automated operations such as scaling, healing and upgrade across multiple sites, all interfaces to the function and between subcomponents must be

- Clearly defined and documented.
- Allow different versions (at least N and N+1) to coexist with defined behaviour to enable an in-place upgrade.

In this context, the database schema used to store state is effectively an internal interface that must also be versioned. This is exposed via APIs to the VNF components that store

²⁹ See overload section

³⁰ Contention may be foreseen for CPU not considered for memory or storage; if the minimum VM size for some components of a slice is larger than required for the total traffic envisaged, for example.

state, rather than directly exposing the internal schemas or native APIs of the databases in these components. By decoupling APIs from schemas and making them technology neutral, it becomes possible to safely evolve database schemas or change database technologies, hiding these changes behind versioned APIs.

Up-level capabilities should only be used across an interface if both sides have agreed to their use. Many 3GPP and other standards incorporate such capabilities and similar logic should be applied to internal interfaces too.

9 Additional Considerations

In addition to the VNF features discussed in the previous section, there are several other considerations that will affect operations automation.

9.1 Minimise Special Paths

Complex but rarely used processes and code paths – such as for disaster recovery – are prone to failure, with potentially serious effect.

Cloud VNFs should be designed with minimal special code paths; recovery from failure should be architected into the mainline paths.

This requires a different mindset to many traditional physical systems, though the principles outlined above will help. For example, segregating state into a distributed, resilient database should eliminate the complex “sync” operations typically used between appliances with local state stores, and all the risks of failure or any need for “resync.”

A cloud VNF should be aware of the declarative model of desired state for its operation, and work towards bringing its internal state to that desired state as the mainline code path. Subscriber data not in cache, for example, is fetched if it is not found – whether that is on the first transaction from the subscriber or after a cache failure. And recovery from failure of a compute process can be via a simple transaction retry.

In short, parts of the cloud will fail and a cloud VNF shall handle such events as part of its normal processing.

9.2 Licensing

Cloud environments shall be dynamic, with resources allocated on demand and processes potentially scaled or moved frequently. Traditional appliance-based licensing systems do not cope well with such environments. Typical issues include trying to limit process instances, or locking licenses to physical hosts or VM IDs.

VNF licensing should not impose limits, instead nurturing a fully automated cloud deployment model. Cloud native licensing models should also lend themselves to various business models, such as pay-as-you-grow or license pooling.

Note that good instrumentation of the VNF certainly eases implementation of a flexible and cloud-friendly licensing model.

9.3 Database Considerations

There are many good distributed databases available for VNFs to use to store state and configuration in a cloud architectures. But databases are not equal, particularly with regards to their consistency and availability, or partition tolerance (see the CAP theorem³¹), or the balance of read/write performance requirements.

Different VNFs, or components of one VNF, may require different databases to fit with the resiliency characteristics required for the service. At the same time, service providers may be concerned about the proliferation of database technologies causing increased load to understand, manage and secure each variant.

An ideal cloud VNF will have clearly defined APIs to the underlying database it requires, with clear definition of the CAP and other characteristics. The service provider then has the option to plumb the VNF into the preferred database technology.

9.4 Security

Security is a very wide topic which we cannot cover in depth in this paper. However, a few high-level implications on VNF design for the cloud are worth pointing out:

- Each VNF should document and utilize an appropriate security framework and policy which can be applied automatically and without error across all lifecycle operations
- The inherent distributed nature of a cloud VNF calls for proper protection (typically encryption) of all communication among the components
- Denial of Service attacks may be mitigated in a cloud environment via dynamic scale-out under appropriate policy controls, more quickly and easily than has been possible for physical appliances.

















































There is no “obviously right” answer for security. Each service provider must assess their own needs and constraints. The only certainty is that any pretence of security via physical access control or obscure interfaces – both of dubious protection in any case as shown in

³¹ https://en.wikipedia.org/wiki/CAP_theorem

widely-reported SS7 network attacks – certainly do not apply to an all IP-based publicly accessible cloud.

10 Summary of Operational Factors

Table 2 summarizes the relative importance of the various operational automation aspects for each lifecycle operation.

	Deploy	Healing	Scale/ Move	Change/ Upgrade	Overload	Devops Cycle
Instrumentation						
Self-management						
Discovery						
Load-balancing						
Resource use						
Minimize Special Paths						
State handling						
Versioning						



Critical



Important



Less Important

Table 2: Importance of automation aspects in lifecycle operation

11 Conclusions

It is very clear that the telco industry is undergoing a deep transformation, both technological and cultural, which will require a radical mind shift in all the players, current and future. From a technological perspective, the shift moves us into a world that is software centric with extensive automation of operations. Faster development cycles with low operational risk, efficient resource utilization and scaling on-demand are the key benefits being sought. For this shift to be successful, the automated operations must understand the domain-specific characteristics of each VNF. Failure to do so is likely to lead to sub-optimal results. This is most efficiently achieved by embedding application domain-specific logic within the VNF itself (as far as possible), so that it can be reused across different cloud environments and different service providers with minimal effort or risk of error.

In future, VNFs may need to evolve to operate not just in VM-based Infrastructure-as-a-Service (IaaS) clouds but also containerized or even serverless models. Cloud-bursting and the sharing of resources between IT and NFV payloads promise further optimization of the resource requirements for service providers.

Culturally, the shift to cloud requires that engineering, planning and operations teams and processes (not only internally but across all the stakeholders involved) come closer together, applying scarce people resources to high-value operations and decision making rather than more menial tasks. Automation enables this, but wider organizational buy-in is required to move to a true DevOps model. In this paper, we have laid out the application-layer groundwork necessary for this to occur, but each service provider must define in further detail how they will make their processes and organization more agile, with specific influences like market size, market position and strategic targets also in mind.

While it is unrealistic to expect that the transition to more efficient cloud VNFs will happen in one easy step, all the players in the telco arena should have a clear focus on the benefits they are targeting. If executed correctly, cloud migration will be a win-win situation for everyone involved.

For further questions and feedback please contact:

Roberto Muggianu, Senior Network Architect, Telia Company
roberto.muggianu@teliacompany.com

Johanna Nieminen, Senior Network Architect, Telia Company
johanna.x.nieminen@teliacompany.com

Paul Brittain, VP Product Strategy, Metaswitch Networks
Paul.Brittain@metaswitch.com

Response times may vary.