Use Case Series: VNF Scale Out

October, 2017

Introduction

The VNF Guidelines and Requirements for ONAP package[1] is the set of documents that defines how a VNF must interact with the ONAP platform. This package serves to define the requirements VNFs need to meet to be compatible with ONAP. This paper is the first in a planned series of papers that are intended to complement the existing Requirements package by providing context around specific VNF Use Cases.

Use Case Definition

The term 'use case' may mean different things to different people. For the purpose of this series of papers, we will use a definition along the lines of how Wikipedia defines a Use Case: "In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role (actor) and a system to achieve a goal."[2] In this series of use cases, the actor will be the VNF and the system will be ONAP. The goal will be different in each use case.

There are three different types of use cases:

1. Target-state: Architects will develop a use case to aid in the system's design. This is typically a greenfield-type use case that starts with a blank piece of paper and determines the end state of how the goal should be achieved.
2. Present-state: On the other end of the spectrum, a use case can define how the goal is achieved today. When developing a new system, it is quite possible that the current view differs greatly from the how things are expected to work in the end-state and may involve many manual operations.
3. Interim-state: If the 'Target-state' and 'Present-state' use cases are very different then there might be a need for an interim use case that shows a point along the roadmap from where the system is today and where it is ultimately heading. In this case, the end-state use case may not be as useful for the actor today because the timeline is too far in the future. The 'Interim-state' use case might be more useful for the actor as a place to aim for immediate development work.

Each use case in this series will ultimately fall into one of these categories, which will depend on the specific lifecycle event being considered. The scale out use case presented here will focus on the interim-state. Many of the steps discussed in this document are still in the planning or development stages.

Safe Harbor:

Information set forth in this series of use cases contains technical statements and other forward-looking statements that are subject to risks and uncertainties, and actual results may differ materially. AT&T

Commented [BS1]: ONAP or Linux Foundation?

[1] The VNF Guidelines and Requirements for ONAP may be found at:
http://onap.readthedocs.io/en/latest/guides/onap-user/vnfprovider.html

[2] https://en.wikipedia.org/wiki/Use_case

disclaims any obligation to update or revise statements contained in these documents based on new information or otherwise.

Commented [BS2]: Need to run this by LF Legal

Point in time document

This paper is intended to provide a point-in-time snapshot of how VNF scale out is expected to work. Unlike the VNF Guidelines and Requirements documents, ONAP does not intend to make regular updates to these use cases. Implementations may end up differing from what is presented in this paper. In some cases, ONAP may choose to update the use case documents to reflect future platform evolutions.

Scale Out

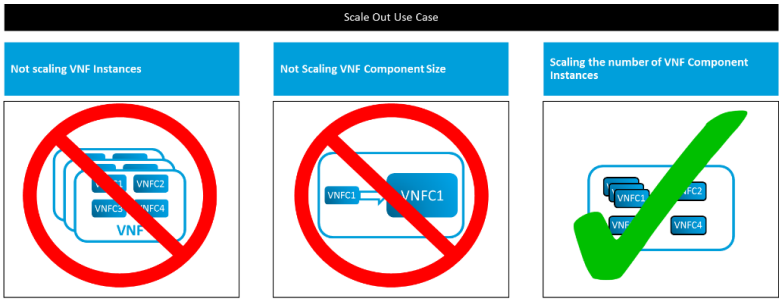
There are many different use cases the ONAP community may find useful. After careful evaluation of several options, *VNF scale out* was chosen as the first one to be documented. VNF scale out is relatively straight forward, and closely related to many other use cases (i.e., VNF instantiation, VNF upgrades). Unlike similar past exercises, the focus of this document is to examine scale out from the point of view of the VNF. In working through this initial use case, several gaps were identified that need addressing for the scale out process to work perfectly. As this is an interim-state use case, there are still components and processes under development as of the time this paper was written. Where gaps may currently exist, they are called out below.

This use case is specifically about scale out and does not include the reverse VNF action of scale in. Scale in introduces complexity that is beyond the scope of this first paper, but it is a candidate for a separate use case in the future.

This use case only addresses *horizontal scaling*. Horizontal Scaling is accomplished by adding more instances of a VNF Component (VNFC), each in its own VM, to the VNF or by adding additional instances of the entire VNF. *Vertical scaling* is when more resources (such as vCPUs or memory) are added to existing VNFCs. As stated in the VNF Guidelines for cloud scaling, vertical scaling of VNF components is not supported on the platform and VNFs are recommended to support horizontal scaling.

Horizontal scaling should be achieved by adding more instances of the VNFC that has become the bottleneck rather than additional instances of the full VNF[3]. This leads to a more efficient use of resources since as only resources are added for those components which need them. Scaling in this manner should be a key design consideration for VNF providers as they structure the components (VNFCs) of a given VNF. This scaling method is also important when developing the Heat templates that will be used to compose and scale the VNF.

[3] There are reasons for scaling entire VNFs but this use case does not describe that process.



VNFs may not use Openstack/HEAT Auto-Scaling

Figure 1 Methods of Scale Out

Definitions and Relationships

For clarity, there are a few terms that need to be well understood and clearly defined. By providing specific definitions for each term and how they relate to each other, we hope to avoid potential confusion within this use case. We will use the following terms/definitions throughout this use case:

VNF – Virtual Network Function – Inspired by the work of the ETSI Network Functions Virtualization (NFV) Industry Specification Group (ISG). VNFs include workloads of all types. The VNF is what is acquired from a vendor (VNF Provider) and all interfaces to a VNF should be well defined so that VNFs may be chained together by the Service Provider to form a Service that is consumed by the customer.

VNFC – Virtual Network Function Component – VNFs are composed from one or more VNF Components (VNFCs). VNFCs work together to perform the actions of the VNFs. Interfaces between VNFCs may be proprietary. VNFCs should be designed with the goal of evolving towards microservices. When laying out the components of the VNF, VNF Providers are encouraged to keep in mind the attributes of microservices, especially Single Capability and Independence.

VM – Virtual Machine – A virtualized compute environment that behaves very much like a physical server. A VM has all the ingredients (processor, memory/storage, interfaces/ports) of a physical server and is created by a hypervisor, which partitions the underlying physical resources and allocates them to VMs. Virtual Machines are capable of hosting a virtual network function component (VNFC).

Module – A subset of the resources of a VNF described in a Heat template. Each module is described in a separate Heat template. A module may contain resources other than just a VNFC. An Incremental Module is a growth or scaling unit.

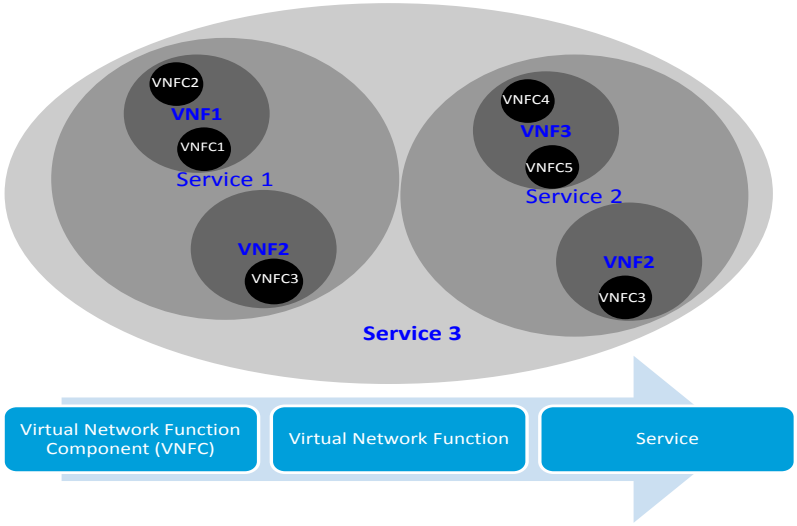


Figure 2: Relationships between VNFs and VNFCs

Relationships

The terms above are all inter-related and these relationships will help in understanding how scaling works on the platform.

- A VNF is made up of one or more VNFCs.
- A VM should contain a single VNFC instance and a VNFC must never span multiple VMs
- An incremental module should contain a single VNFC. An incremental module may contain multiple VNFCs if and only if each of those VNFCs in the module always scale together.

Ideally, a single VNFC should be contained within a single incremental module which should instantiate a single VM for each instance of a VNFC.

This chain of relationships is derived from the fact that a VM is the smallest execution environment within AIC today. If/when ONAP begins to host containerized VNFCs this relationship may change.

Modularity Rules

Heat is the orchestration engine used by OpenStack environments to launch cloud applications. A Heat template describes the infrastructure for a cloud application in a YAML text file that is readable and writable by humans.

Care should be taken to think about how to organize a VNF using Heat Templates as it has a major impact on the ability to properly scale them.

With VNF Modularity, a single VNF should be composed from one or more Heat Orchestration Templates, each of which represents a subset of the overall VNF. These component parts are referred to as “VNF Modules”. During orchestration, these modules are deployed incrementally to create the

complete VNF and additional incremental modules may be deployed at various times to scale portions of the VNF.

All VNFs must have one base VNF module template and that module is the first one deployed: the base template. This base module must include all the shared resources of the VNF including private networks, server groups and security groups. It must also expose all shared resources by their UUID. The base module may include an initial set of VMs and may be operational as a stand-alone minimum configuration of the VNF.

A VNF may also have one or more incremental modules which define additional resources that may be added to an existing VNF. Each module, base or incremental, must be described by a complete Heat template (incremental modules must not be dependent on other incremental modules). These incremental modules should define logical growth units of the VNF.

Ideally each VNFC will be contained in its own incremental module. In this way, scaling can be completed by instantiating the template of the appropriate module.

There are times when a single scaling request might require more than a single additional instance of a VNFC. This can be done by either running the incremental module of a single VNFC multiple times or by creating an incremental module that instantiates multiple instances of the VNFC.

Scale Out Use Case

This section will define the detailed steps of the scale out workflow from a VNF provider perspective. There are many more steps than reflected here in the full scale out use case, but the steps shown here either directly impact the VNF provider or help explain the high level flow of how ONAP accomplishes scaling a VNF.

Overview of Flow

The VNF scale out workflow is broken into 3 areas:

A. Onboarding: This involves everything the VNF Provider needs to provide for Network Cloud Service Provider (NCSP) to run the VNF (Step 1 in **Error! Reference source not found.**). For the sake of simplicity, VVP validation is out of scope for this use case.
B. Deployment: There are many steps required of ONAP between the onboarding area and deployment, but they do not affect the VNF Provider in any significant way, so we will leave those steps out of this use case. (Step 2 in **Error! Reference source not found.**)
C. Runtime: This consists of two passes through orchestration/instantiation as well as implementing the feedback loop to enable automatic scale out. We will consider both automated (Closed-Loop) as well as Manual (Open-Loop) scale out scenarios. (Steps 3-18 in **Error! Reference source not found.**)

   In Figure 3, the green circles mark the steps required for initial instantiation and the yellow circles mark the steps to perform the actual scale out. The initial instantiation steps are included since it is helpful to see the full process from onboarding all the way through scaling. It is also interesting to note the similarities and differences between scaling and initial instantiation.

These three areas are highlighted in the process flow shown in Figure 3. that defines the major steps for scaling.

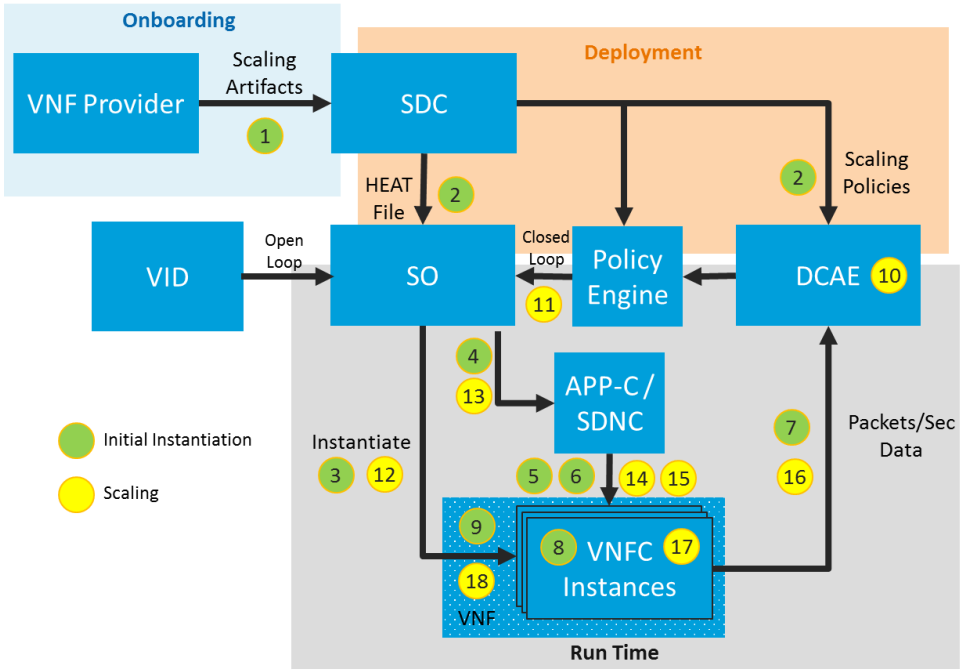


Figure 3: High level overview of scale-out workflow

Detail view of the VNF scale out flow

Step 1 Onboarding

After a VNF Provider has been selected for a VNF implementation the VNF Provider must send the VNF and all relevant information to the Network Cloud Service Provider (NCSP). This is done in the "Onboarding Package". Additional work to further define and evolve this package is underway in ONAP via the VNF SDK project (specifically the VNF Package Model sub-project[4]). The VNF SDK Project will have tools for building, validating and extracting the onboarding package.

The onboarding package is described in section 7b of the ONAP "VNF Provider Guide".

The onboarding package consists of all the data elements needed by the Service Provider to run and operate the VNF. The elements relevant to scaling are:

- **The VNFC images**

[4] Details for the VNF Package Model sub-project may be found at:
https://wiki.onap.org/pages/viewpage.action?pageId=8226059

The onboarding package should contain an image for each VNFC that is used within the VNF.

Image requirements are described in requirement R-26881

- **Engineering Rules (which are later translated to scaling policies)**

  The VNF Provider is required to send the Service Provider all the engineering rules associated with the VNF. These rules will then be translated by the Service Provider into a list of policies that can be used by ONAP. Many of these engineering rules may have to do with scaling (e.g., if Traffic on the VM gets above ### packets/second then another instance of the VNFC should be added to the pool).

  In the future, the NCSP may ask the VNF provider to provide the actual scaling policies, but as of now only the engineering rules are required.

  Scaling Policies (and Engineering rules) are covered in requirements R-69565, R-22888, R-01478, R-56815, R-96634, R-13613 and R-27511.

- **Configuration Files (Yang models, Ansible Playbooks and CHEF Recipes also)**

  Device Yang models are required by VNFs that utilize NETCONF for configuration by the Application Controller. If the VNF is using either Chef or Ansible for configuration, then the VNF Package must include Cookbooks or Playbooks along with JSON files for each supported action.

  Configuration requirements are covered in Section 7c of the ONAP "VNF Provider Guide".

  Requirements R-89571, R-30278, R-13390, R-18525, R-75608, R-16777, R-46567, R-16065 describe the actual artifacts that must be provided as part of the VNF Package.

- **Heat Files**

  Heat templates are used by the Service Orchestrator (SO) for orchestrating the building of the VNF. The VNF Heat Template Requirements outline a method for organizing these files as a series of modules that can be efficiently utilized to perform the initial build of the VNF as well as adding new VMs to host new instances of VNFCs as they are needed. Each of these Heat Templates may consist of a series of nested files.

  As ONAP continues to mature, the Onboarding Package may evolve to support alternatives to Heat.

  A Heat template is required per requirement R-97102 and requirements for building Heat templates are documented in Section 5b of the ONAP "VNF Provider Guide".

Step 2 Deployment

There are many steps ONAP needs to accomplish between the time it receives the VNF and the time it is deployed to operational AIC sites, but these are all internal steps for verifying and testing the VNF. They do not directly impact the VNF Provider (unless the NCSP needs help validating the VNF) and therefore they are omitted from this scale out use case.

After the NCSP has received the VNF Onboarding Package and all the needed data elements have been created within SDC (Service Design and Creation), it is time to deploy those elements to the various components of ONAP.

Step 2 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 3 SO Orchestration

The SO (Service Orchestrator) uses the Heat files to orchestrate the assignment and creation of the VNF within AIC. For the initial instantiation, the base Heat module and any needed incremental modules are executed. This should instantiate everything needed for the VNF to run properly.

Step 3 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 4 APPC/SDNC Call

After instantiation, SO requests APPC/SDNC to configure the VNF. For this use case, from the VNF point of view, the Application Controller (APPC) and the Software Defined Network Controller (SDNC) do nearly identical functions. APPC will configure and control some VNFs (primarily L4-L7 VNFs) while SDNC will configure and control other VNFs (Primarily L2-L3 VNFs).

Step 4 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 5 VNF Configuration

After receiving the signal from SO, either APPC or SDNC will then begin configuring the entire VNF. If the VNF is using Netconf then APPC (or SDNC) will use the supplied Device YANG model to configure all the VNFCs within the VNF. If the VNF is using Ansible or CHEF then APPC/SDNC will trigger configuration through the Ansible or CHEF servers using the supplied Playbooks or Recipes.

Configuration requirements are covered in Section 7c of the ONAP "VNF Provider Guide".

Step 6 VNF Validation

Once the VNF is fully configured it is time to test it to ensure that it is functional. The mechanism ONAP has for doing validation is "Healthcheck". Currently, Healthcheck encompasses the status of the entire VNF, which is sufficient to perform a validation after initial instantiation. APPC/SDNC will send a HTTP GET to the URL/Port provided by the VNF or, in case the VNF supports Healthcheck through Ansible or Chef, it will send a request to the Ansible or Chef server to run the VNF Healthcheck Playbook or Cookbook.

The HealthCheck RPC executes a vendor-defined VNF Healthcheck over the scope of the entire VNF (e.g., if there are multiple VNFCs, then run a health check, as appropriate, for all VNFCs). It returns a 200 OK if the test completes. A JSON object is returned indicating state (healthy, unhealthy), scope identifier, time-stamp and one or more blocks containing info and fault information

Healthcheck is described in Section 7c of the ONAP "VNF Provider Guide".

Step 7 VNF Monitoring

Once configuration is complete, DCAE (Data Collection Analytics and Events) begins collecting performance data and other metrics the VNF is sending to ONAP. Each VNFC must send all its data to DCAE, it may also send a subset of that data to other VNFCs within the VNF.

Monitoring is described in Section 7d of the ONAP "VNF Provider Guide".

Step 8 "In Service"

If the VNF is not already serving live traffic, then now is the time to put it into service and start accepting traffic. This could be done by an action from APPC/SDNC on the VNF for cases where the VNF itself controls whether it receives traffic, or as an action on a different VNF (e.g., an external load-balancer) when traffic distribution is not within the VNF's control.

The requirements for putting a VNF(C) in service have not yet been defined.

Step 9 VNF Validation

Once the VNF is "In Service" and live traffic is flowing through it, a final Healthcheck will be requested by ONAP to ensure that all processes, VNFCs, and other resources are operating as expected. If there are any problems reported by Healthcheck, then corrective actions should be taken by operations. There is a section later in the document that goes into more detail on Healthcheck.

Healthcheck is described in Section 7c of the ONAP "VNF Provider Guide".

Step 10: DCAE Analytics

DCAE begins analyzing the traffic being sent from the VNF/VNFC instances. This analysis can be very simple analysis or can involve complex algorithms that use multiple metrics from multiple sources. Today, the NCSP develops the analytics used by DCAE; currently there is no method for VNF Providers to provide code to DCAE for analytics.

For simplicity in this use case, we assume that DCAE does not perform any complex analysis, rather it is just passing along the packets per second metric.

Step 10 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 11: Events and Policy

In this step, the Policy engine is looking to see if any of its thresholds (developed from the engineering rules provided with the VNF in the onboarding package) have been triggered. For this use case, the trigger is assumed to be simply based on a packets per second threshold. If the packets per second threshold is exceeded for a VNFC then a scaling event will be triggered for that component.

Step 11 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 12: SO Orchestration

In response to the policy engine detecting that a threshold limit has been exceeded, additional VMs need to be instantiated to host additional VNFC instances. This process is very similar to the initial

instantiation except that the entire VNF does not need to be instantiated. Only the incremental module associated with the scaling event needs to be instantiated

When the SO receives the scaling event, it will execute the incremental Heat module associated with the scaling event. (This step may also be triggered manually via the VID. See the section on Open Loop Scale Out below.) Ideally this will identify the one VNFC that is becoming a bottleneck and trigger additional instances to be instantiated via the incremental module. The SO will use the Heat incremental module to orchestrate the assignment and creation of that VNFC.

Step 12 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements. However, Heat modules are described in section 5b of the ONAP "VNF Provider Guide".

Step 13: APPC/SDNC Call

Once the SO has instantiated the VNFC it will then call upon the Application Controller (APPC) or Software Defined Network Controller (SDNC) to configure the new instance of the VNFC.

Step 13 does not directly impact the VNF Provider and therefore there are no requirements associated with this step in the VNF Requirements.

Step 14 VNFC Configuration

At this point, APPC or SDNC will then begin configuring the VNFC running on the newly instantiated VM. If the VNFC is using Netconf for configuration, then APPC (or SDNC) will use the supplied Device YANG model. If the VNFC is using Ansible or CHEF then APPC/SDNC will trigger configuration through the Ansible or CHEF servers using the supplied Playbooks or Recipes.

Configuration requirements are covered in Section 7c of the ONAP "VNF Provider Guide".

Step 15 VNF Validation

Once the VNFC has been completely instantiated and configured then it is time to ensure it is running properly. Healthcheck is still a VNF level process but if the VNF Healthcheck receives a "Healthy" response after the new instance has been added to the VNF then we can assume that the new instance is healthy and ready for traffic.

Once the VNF returns a response indicating it is "Healthy" then ONAP can add the new VNFC instance to the service.

Healthcheck is described in Section 7c of the ONAP "VNF Provider Guide".

Step 16 VNFC Monitoring

Once configuration is complete, DCAE (Data Collection Analytics and Events) begins collecting performance data and other metrics that the VNFC is sending to ONAP. The VNFC must send all its data to DCAE, it may also send a subset of that data to other VNFCs within the VNF.

Monitoring is described in Section 7d of the ONAP "VNF Provider Guide".

Step 17 "In Service"

The new instance of the VNFC should now be put into service. This may just be an additional configuration step on the VNFC or it could be a configuration step on a different VNFC. When adding new instances of a VNFC to a VNF, no existing flows should be affected or experience any disruptions.

The requirements for putting a VNF(C) in service have not yet been defined.

Step 18 VNF Validation

Once the instance(s) of the VNFC is up and running, a Healthcheck is done on the entire VNF. This ensures that the new instance(s) is not negatively impacting the overall VNF.

Healthcheck is described in Section 7c of the ONAP "VNF Provider Guide".

Healthcheck

Healthcheck is a new feature, recently introduced and documented in Section 7c of the ONAP *"VNF Provider Guide"*. Its purpose is to provide a single command by which ONAP can validate the overall health and status of a VNF. Healthcheck is simply an API (Restful, Chef or Ansible) call to a function on the VNF. This function is created by the VNF Provider and needs to test to ensure the VNF is 100% healthy. It needs to test all resources, including all VNFC instances and VMs that comprise the VNF. All processes within a VNFC should be tested along with network connectivity to/from each VNFC. It should only return a result of "Healthy" if everything is running as it should. If there is any issue with the VNF then Healthcheck should return "Unhealthy".

Healthcheck may be called at any time during the VNF's lifecycle. Therefore, it should be non-disruptive to live traffic that is being served by the VNF.

Today, Healthcheck only operates at the VNF level. This is a useful measurement and if the VNF is running in a healthy manner before a scaling operation and becomes unhealthy after the scaling operation it might indicate that something is wrong with the new VNFC instance(s) that were added to the VNF. ONAP developers are currently studying the Healthcheck API to determine if it should operate at a more granular level or be extended to provide a richer response payload.

As it is currently implemented, the NCSP well expect that a Chef or Ansible Healthcheck success shall be returned (return code 0) by a Playbook or Cookbook only when the VNF is 100% healthy (or "Healthy" in the JSON object if using the REST API). This means all the following criteria need to be met:

- All VNF application processes are running and ready to take service requests.
- Critical and non-critical resources are ready.
- There are no open MINOR, MAJOR or CRITICAL traps/alarms.
- There are no issues with the VNF that need attention even if they do not impact services provided by the VNF.

If any of these items are not met then the playbook should return an "UnHealthy" in the JSON object and a non zero return code for Playbooks.

NOTE: A VNF that reports as healthy should either be already serving traffic successfully, or ready to serve traffic when put into service (if such a state is supported).

Closed Loop vs Open Loop Scale Out

As ONAP progresses towards supporting fully automated VNF scale out that is triggered by control loops and policies, there will likely be several interim phases where some manual intervention is still required. Additionally, even in the Target-state, there will be times when an operations team wants to scale components for reasons outside of what's been defined in the scaling policies. For instance, if an NCSP knows that a certain VNF always needs more capacity at the end of the month it may proactively, manually or via a new policy, trigger the scaling process to prepare the infrastructure for the additional traffic.

These manual scaling events are accomplished through VID (Virtual Infrastructure Delivery) which is the operations front end to the ONAP Platform. Using VID, operations can either insert themselves in the middle of the closed loop and react manually to scaling events so that they can ensure that scaling is being done properly (we refer to this as Open Loop scaling). They can also use VID to trigger scaling events for reasons other than those provided by the existing scaling policies.

Conclusion

This VNF scale out use case is the first of what will be a series of use cases to better communicate to VNF Providers how their VNFs should behave during certain lifecycle events. This paper provided an interim-state view of the onboarding, deployment, and runtime steps relevant to VNF scale out and explained the specific VNF Requirements that are related. Many of the steps discussed in this document are still in the planning or development stages.

While this paper focused only on one use case, VNF scale out is highly related to several other VNF lifecycle events. In particular, VNF scale out shares many of the same steps (and requirements) as VNF instantiation, VNF scale in, VNF software upgrade, and several VNF resiliency operations. For instance, a VNFC software upgrade may involve instantiating a new instance of the component and taking down the old instance of a component. This involves many of the same steps as a VNFC scale out operation followed by a VNFC scale in operation.