



ONAP Architectures for Network Function Virtualization

Abel Fernández-Nandín, Felipe Gil-Castiñeira^(✉),
and Francisco J. González-Castaño

atlanTTic Research Center for Telecommunication Technologies,
Universidade de Vigo, Campus Universitario, 36310 Vigo, Spain
{abel,xil,javier}@gti.uvigo.es

Abstract. The Network Function Virtualization paradigm is changing the telecommunications industry. Network applications in general purpose telco infrastructures will be instantiated on demand or deployed in the most appropriate location for each use case. Nevertheless, these virtualized scenarios are complex and require tools to manage the different components flexibly and reliably. ONAP is one of the projects that are implementing such tools. It provides a rich set of elements that can be executed in virtual machines or containers, following different architectures. In this paper we present the different possibilities for that and analyze their advantages and disadvantages.

Keywords: Network Function Virtualization · Containers · ONAP

1 Introduction

In the last years, cloud computing has become a key enabler in the IT field. Infrastructure virtualization has been crucial to optimize resource usage [1], and has brought unparalleled management flexibility. The continuous search for improvements has spawned several solutions that continue to shape the cloud computing world.

One of the most relevant technologies in the last years is containers [2], which provide means to encapsulate applications without using virtual machines. Since there is no virtual kernel between the host machine and the containerized application, this technology is lighter, but also less secure: a vulnerable application that compromises the host kernel also compromises the rest of the containers [3]. This is challenging for multi-tenancy scenarios. The most frequent solution is to launch these containers on top of virtual machines [4] that are already running, which does not involve extra boot time.

Containers technology has had such an impact that not only new associated solutions like Kubernetes [5] have appeared, but also traditional cloud computing software like OpenStack has been adapted to support them [6]. There also exist implementations of orchestration engines to automate application deployment, scaling, and management [7].

The telco sector has not been indifferent to this revolution. Initially, telecommunications were just commodities that provided connectivity to the cloud (for the users or to interconnect data centers). Since then many telcos have become cloud computing service providers themselves trying to leverage their intrinsic advantages (such as large network bandwidth) to reach a new market [8]. Nevertheless, the real impact in the telco sector started with network “softwarization”. Typical network functions started to migrate from proprietary hardware to Commercial-Off-The-Shelf (COTS) computers, simplifying tasks such as feature upgrades, fixing bugs, or scaling the service [9]. The next logical step is the virtualization of these network functions, by turning network equipment into virtual entities. This is the driving force behind the Network Function Virtualization (NFV) concept [10]. The new virtual appliances that satisfy the NFV paradigm can be instantiated on demand or deployed in any cloud location with enough available resources, and scaled with the number of requests. Furthermore, the same hardware can perform different functions at a time (by executing different virtual network functions or VNFs in the same hardware) or at different times (e.g. different services over the same physical platform for different customers).

Software Defined Networking (SDN) is another related technology that decouples the control and data planes and centralizes the decisions in a controller [11]. These planes are separated through a well defined API between the switches and the controller. The most popular API nowadays is OpenFlow. SDN allows controlling traffic flows from a centralized location, simplifying the creation of new NFV instances and relocating of existing ones, without interrupting service for end users.

A NFV infrastructure supported by a SDN enabled network is a complex environment that also requires an architecture for management and orchestration. For this purpose, the ETSI NFV group defined the management and orchestration (MANO) framework [12]. It standardizes the functionalities for the provisioning of VNFs, their configuration, and the configuration of the supporting infrastructure. It also includes the orchestration and management of the physical and virtual resources that the VNFs require.

The ONAP (Open Network Automation Platform) project provides a unified operating framework that comprises the orchestration and automation of VNFs, SDNs and the services combining both. Thus, ONAP is a superset of ETSI MANO with extra capabilities. It evolved from the merge of ECOMP (Enhanced Control, Orchestration, Management and Policy) and Open-O, and is under the umbrella of the Linux Foundation. Nowadays it has strong momentum with the participation of vendors that cover over 60% of mobile customers worldwide [13]. This makes ONAP an interesting platform to analyze how containers can help build an NFV infrastructure.

This paper reviews different ways to stack technologies to support VNFs with the ONAP platform, which is discussed in depth in Sect. 3. Section 2 presents the related work. Section 4 describes different architectures for ONAP deployment. Finally, Sect. 5 presents our conclusions and plans for future work.

2 Related Work

The NFV paradigm appeared with the maturity of virtualization techniques. Previous work has addressed how to combine different virtualization technologies to deploy NFV in a telco environment [14, 15], including containers [16, 17]. Nevertheless, we are not aware of any other work analyzing the different possibilities to use ONAP for this task.

OpenStack is a platform that allows managing the networking, computing, and storage resources in a cloud. It is very popular in the virtualization field and is one of the main components in ONAP. In [18] the authors studied its suitability for provisioning and deploying NFV services. The paper also discussed future challenges for OpenStack to be adopted to implement an NFV infrastructure. For example, the authors state that OpenStack should not affect service performance, and that its configuration options should cover VNF services from different vendors. Some limitations were identified regarding networking capabilities, traceability, the assignment of VNFs to specific hardware resources and security. Finally, the paper also described the different options to deploy VNFs (on virtual machines, containers, or bare metal) and the interest of such flexibility.

In [19] the authors describe a resource allocation strategy based on ONAP for deploying Virtualized Network Functions (VNFs) on distributed data centers. They also describe the different ONAP components, how they are integrated, and the mechanism for resource allocation (basically, it is the heuristic algorithm implemented in OpenStack's scheduler), but they do not discuss the different options to deploy ONAP.

In [20] the authors describe the existing interest in using containers for NFV and other telecom infrastructures, but they also present several issues that make them not ready for a production state. One of the main concerns is related to the security of the deployments with containers. In [21] the authors discuss the Docker environment's security implications. Docker security relies on three factors: isolation of processes at the userspace level, the enforcement of the isolation by the kernel, and security in the networks operation. If an attacker compromises a container, it should not be possible to affect another. To conclude the paper, the authors state that an orchestrator could solve some of the security issues with mechanisms implemented at higher levels of abstraction, and by using automation to provide an automated way to audit the security and to patch the system. Therefore, if such measures are implemented in ONAP, containers should be a valuable alternative to deploy NFV services (and even the ONAP infrastructure).

Summing up, although there exists significant work on NFV requirements and base technologies (such as OpenStack), we are not aware of any publication on the suitability of different virtualization architectures to deploy ONAP.

3 ONAP and Its Supporting Technologies

ONAP’s architecture is divided into its Design-Time and Run-Time frameworks.

The Design-Time framework features the Service Design and Creation (SDC) and the Policy subsystems, which enables developers to define, simulate, and certify assets and their associated processes and policies.





The Run-Time framework comprises the Active and Available Inventory (AAI), which allows visualizing and managing the assets’ inventory; controllers for managing the state of a resource; the Data Collection, Analytics and Events (DCAE) subsystem, which gathers performance and usage metrics of the resources to find anomalies; the Master Service Orchestrator (MSO), which arranges, sequences, and implements tasks based on rules and policies to coordinate the creation, modification, or removal of resources in the managed environment; and the Security Framework, which aims at providing security by design in a variety of ways.

ONAP relies on OpenStack to allocate resources. Let us mention some key components for ONAP. OpenStack Heat plugin implements an orchestration engine. The infrastructure for a cloud application can be described in a text file, which simplifies version control. The Magnum plugin brings several container orchestration engines into OpenStack. It uses Heat to build virtual machine clusters, on top of which an orchestration engine deploys containerized services. The alternatives for orchestration engines are Docker Swarm, Apache Mesos and Kubernetes. The latter has become the de facto standard for deploying, scaling, and managing containerized applications. Its architecture follows a master/slave hierarchy, in which the master nodes that belong to the control plane manage the worker nodes. Kolla allows deploying Openstack clouds by containerizing its services. This simplifies the configuration and scalability of clouds, since they may run on top of Kubernetes for lifecycle management. Ironi is useful to employ bare metal machines instead of virtual instances, for those cases where a virtualization layer could hinder the performance of the system. By combining this plugin with Magnum container execution reaches peak performance.

4 Possible ONAP Architectures

By combining previous technologies it is possible to build different architectures for ONAP, which may have advantages and disadvantages in particular scenarios.

The following sections discuss different alternatives for the deployment of ONAP and VNFs. The diagrams in this section represent the different elements of the architectures with the following color code:

-  Operating System
-  Bare-metal machine
-  Virtual Machine
-  Container

Finally, Table 1 summarizes the main features of the alternative architectures. It describes the cloud computing software used to deploy ONAP’s services, the VIM (Virtual Infrastructure Manager) used by said services to launch the VNFs, the ease of integration between the proposed architecture and an existing OpenStack cloud, the ease of managing the system according to how independent the different technologies are from each other, the qualitative estimation of the resources it uses, and the degree of security it provides according to the existence of a virtualization layer between the host machine and both the VNFs and ONAP’s services.

4.1 ONAP on OpenStack

Figure 1 shows an architecture where a vanilla OpenStack installation runs bare-metal on top of Ubuntu (or any other Linux distribution). In this scenario, ONAP services are deployed as several virtual machines orchestrated by OpenStack Heat.

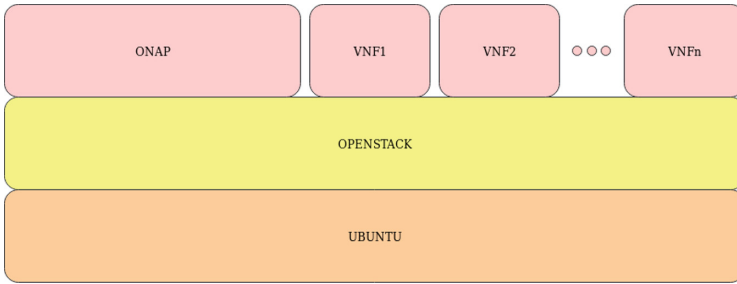


Fig. 1. ONAP on OpenStack

The OpenStack infrastructure that hosts ONAP services also acts as its own VIM. Therefore, VNFs are launched as virtual machines on top of the same OpenStack infrastructure.

The high hardware costs make this choice unattractive in principle. However, it may be useful in a scenario where a tenant with an OpenStack cloud wants to adopt ONAP without any changes in its architecture.

4.2 ONAP on Kubernetes on OpenStack

The architecture depicted in Fig. 2 shows a bare-metal OpenStack installation on top of which a virtualized Kubernetes cluster runs.

ONAP services now run as containerized applications managed by Kubernetes, which simplifies management, scaling, and auto-healing. Kubernetes may be installed manually inside virtual machines or by using the Magnum plugin.

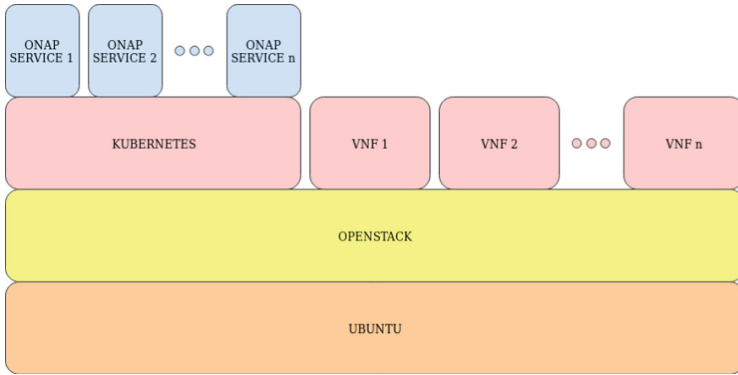


Fig. 2. ONAP on Kubernetes on OpenStack

Since containers, unlike virtual machines, do not need to secure resources before they run, their resource usage is more efficient. Moreover, the fact that ONAP services still run on top of OpenStack makes integration almost as easy. However, if a tight integration between Kubernetes and OpenStack is desired, it would be necessary to install the Magnum plugin. This would presumably require some cloud architecture changes and, therefore, adoption is not as straightforward as in Sect. 4.1.

4.3 ONAP on Kubernetes and Openstack

The architecture in Fig. 3 has two independent components, Kubernetes and OpenStack, which run bare-metal.

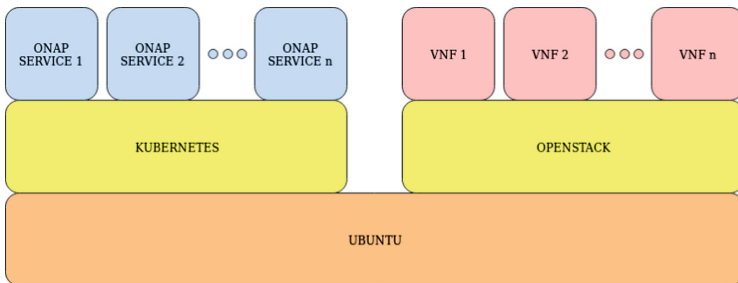


Fig. 3. ONAP on Kubernetes and Openstack

OpenStack no longer serves as the infrastructure for ONAP. It just acts as a virtual infrastructure manager to launch the VNFs. Kubernetes, however, allows managing the containerized ONAP services. This scenario behaves as that in Sect. 4.2, but ONAP services are more efficient because there is no virtualization layer between them and the host machine.

In principle, there is a trade-off between this scenario and that in Sect. 4.2 related to ONAP services infrastructure. Since Kubernetes is now bare-metal, the containers it manages can achieve peak performance. However, OpenStack no longer controls the lifecycle of the Kubernetes cluster, so another approach is necessary for infrastructure deployment.

Fortunately, there exists an alternative that offers the best of both worlds by making use of the Magnum and Ironic plugins: by combining them, OpenStack can handle bare-metal machines as if they were virtual ones, and provision a Kubernetes cluster inside them.

4.4 ONAP and OpenStack on Kubernetes

The architecture in Fig. 4 corresponds to a scenario in which a bare-metal Kubernetes containerizes and manages services from both ONAP and OpenStack.

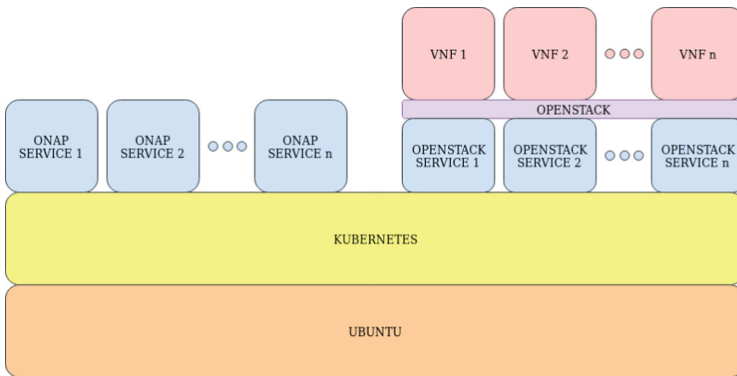


Fig. 4. ONAP and OpenStack on Kubernetes

Here Kolla is the key enabler, as it allows OpenStack’s services to run inside Docker containers, and the Kolla-Kubernetes project permits a Kubernetes cluster to manage them.

This approach containerizes the whole underlying infrastructure of the platform, enabling the management, scaling, and auto-healing of all the services both from ONAP and OpenStack. It would also be the most resource-efficient due to the lack of virtualization. However, it no longer takes advantage of an existing OpenStack deployment, so it would not integrate well with a cloud where ONAP and an existing platform must coexist.

The VNFs, however, are not free of a virtualization layer, since they run on virtual machines on top of OpenStack. The transition from VNFs to cVNFs (containerized VNFs) will be discussed in Sect. 5.

Table 1. Comparison of architectures

| Architecture | ONAP infrastructure | VIM | Integration | Management | Resource usage | Security environment |
|------------------|------------------------|-------------------------|-------------------|-------------------|----------------|-----------------------------------|
| Architecture 4.1 | OpenStack | OpenStack | Easy | Easy | High | Virtualized infrastructure + VNFs |
| Architecture 4.2 | Virtualized Kubernetes | OpenStack | Easy | Medium difficulty | Medium | Virtualized infrastructure + VNFs |
| Architecture 4.3 | Kubernetes | OpenStack | Medium difficulty | Hard | Medium | VNFs |
| Architecture 4.3 | Kubernetes | Containerized OpenStack | Hard | Medium difficulty | Low | VNFs |

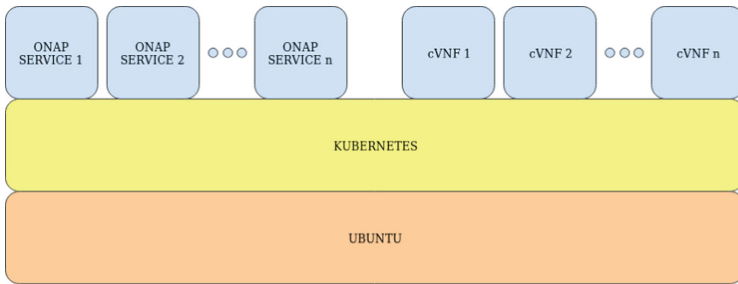


Fig. 5. ONAP on Kubernetes

5 Conclusions and Future Work

From the discussion of the different architectures, one might infer a development trend of cloud computing platforms for NFV towards containerization. However, the support for containerized VNFs is not fully extended or developed. For this reason, ONAP has been taking steps into integrating Kubernetes as a virtual infrastructure manager, which would enable containerized VNFs to reside adjacently to containerized ONAP services. This would result in a flat architecture as shown in Fig. 5. One important benefit derived from this flat architecture would be that the way of enforcing security and load balancing policies would be shared between the ONAP services and the cVNFs, since Kubernetes provides those capabilities. Because the current solutions make use of both Kubernetes and OpenStack, the configuration of these policies is split among the two platforms and, therefore, more cumbersome to implement.

It is apparent that containerizing platforms as much as possible would greatly benefit the infrastructure, but there is some concern that this approach would not be valid for multi-tenancy scenarios. As a consequence, there is a trade-off between virtual machines and containers. The former are secure but slow, and the latter are fast but insecure. This is caused by sharing the kernel between the host machine and the guest applications. There are some solutions that bring the best of both approaches.

For example, Hyper (<https://hypercontainer.io/>) aims at providing a suitable environment for multi-tenancy by implementing hardware-enforced isolation, which is achieved by containing applications within separate kernel spaces. Since the kernel is really streamlined, it does not affect the performance of the container heavily, so sub-second boot times are still feasible.

Clear Containers has a similar goal (<https://clearlinux.org/containers>), since it implements lightweight virtual machines by placing an optimized kernel between the host machine and the guest application.

Kata Containers (<https://katacontainers.io/>) is an upcoming project by the OpenStack Foundation. It will combine underlying technologies of the previous projects, Hyper's runV and Clear Containers' runtime. The goal is an architecture-agnostic system to be run on multiple hypervisors, supporting the OCI specification. It will also support the CRI standard, so it will be compatible with Kubernetes' container runtime.

Frakti (<https://github.com/kubernetes/frakti>) is a hypervisor-based container runtime for Kubernetes. It seeks to leverage Kata Containers to provide strong isolation by running containers and Kubernetes' pods directly inside hypervisors. The containers of each pod will share the kernel, reducing burden. However, this will not compromise the infrastructure of other tenants, since each pod will have its own kernel.

As future work we will contribute to the integration of one of these approaches with the future Kubernetes virtual infrastructure manager, for ONAP to be not only fully containerized but also multi-tenant.

References

1. Armbrust, M., et al.: Above the clouds: a Berkeley view of cloud computing. Technical report UCB/EECS-2009-28, vol. 4, pp. 506–522. EECS Department, University of California, Berkeley (2009)
2. Vaughan-Nichols, S.J.: New approach to virtualization is a lightweight. *Computer* **39**(11), 12–14 (2006)
3. Gao, X., Gu, Z., Kayaalp, M., Pendarakis, D., Wang, H.: ContainerLeaks: emerging security threats of information leakages in container clouds. In: 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 237–248. IEEE (2017)
4. Manco, F., et al.: My VM is lighter (and safer) than your container. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 218–233. ACM (2017)
5. Brewer, E.A.: Kubernetes and the path to cloud native. In: Proceedings of the Sixth ACM Symposium on Cloud Computing, pp. 167–167. ACM (2015)
6. Cacciatore, K., et al.: Exploring Opportunities: Containers and OpenStack. OpenStack White Paper, 19 (2015)
7. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. *ACM SIGCOMM Comput. Commun. Rev.* **41**(1), 45–52 (2011)
8. Lei, X., Zhe, X., Shaowu, M., Xiongyan, T.: Cloud computing and services platform construction of telecom operator. In: 2nd IEEE International Conference on Broadband Network & Multimedia Technology, IC-BNMT 2009, pp. 864–867. IEEE (2009)

9. Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., Flinck, H.: Network slicing & softwareization: a survey on principles, enabling technologies & solutions. *IEEE Commun. Surv. Tutor.* (2018)
10. Chiosi, M., et al.: Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action. In: *SDN and OpenFlow World Congress*, pp. 22–24 (2012)
11. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. *Proc. IEEE* **103**(1), 14–76 (2015)
12. ISG, N.: Network Functions Virtualisation (NFV): Management and Orchestration. European Telecommunications Standards Institute, Technical report (2014)
13. Parker-Johnson, P., Doiron, T.: Succeeding on an open field: the impact of open source technologies on the communication service provider ecosystem. *ACG Research Report* (2018)
14. Duan, Q., Ansari, N., Toy, M.: Software-defined network virtualization: an architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw.* **30**(5), 10–16 (2016)
15. Kourtis, M.A., et al.: T-NOVA: an open-source MANO stack for NFV infrastructures. *IEEE Trans. Netw. Serv. Manag.* **14**(3), 586–602 (2017)
16. Cziva, R., Jouet, S., White, K.J., Pezaros, D.P.: Container-based network function virtualization for software-defined networks. In: *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 415–420. *IEEE* (2015)
17. Cziva, R., Pezaros, D.P.: Container network functions: bringing NFV to the network edge. *IEEE Commun. Mag.* **55**(6), 24–31 (2017)
18. Kavanagh, A.: OpenStack as the API framework for NFV: the benefits, and the extensions needed. *Ericsson Rev.* **2** (2015)
19. Slim, F., Guillemin, F., Gravey, A., Hadjadj-Aoul, Y.: Towards a dynamic adaptive placement of virtual network functions under ONAP. In: *Third International NFV-SDN 2017-O4SDI-Workshop on Orchestration for Software-Defined Infrastructures* (2017)
20. Rotter, C., Farkas, L., Nyíri, G., Csatári, G., Jánosi, L., Springer, R.: Using Linux containers in telecom applications. In: *19th International ICIN Conference - Innovations in Clouds, Internet and Networks*, pp. 234–241 (2016)
21. Combe, T., Martin, A., Di Pietro, R.: To Docker or not to Docker: a security perspective. *IEEE Cloud Comput.* **3**(5), 54–62 (2016)