Demo: Service Function Chaining Across OpenStack and Kubernetes Domains

Hadi Razzaghi Kouchaksaraei
Computer Network Group
Paderborn University
hadi.razzaghi@uni-paderborn.de

Holger Karl
Computer Network Group
Paderborn University
holger.karl@uni-paderborn.de

ABSTRACT

Remarkable advantages of Containers (CNs) over Virtual Machines (VMs) such as lower overhead and faster startup has gained the attention of Communication Service Providers (CSPs) as using CNs for providing Virtual Network Functions (VNFs) can save costs while increasing the service agility. However, as it is not feasible to realise all types of VNFs in CNs, the coexistence of VMs and CNs is proposed. To put VMs and CNs together, an orchestration framework that can chain services across distributed and heterogeneous domains is required. To this end, we implemented a framework by extending and consolidating state-of-the-art tools and technologies originated from Network Function Virtualization (NFV), Software-defined Networking (SDN) and cloud computing environments. This framework chains services provisioned across Kubernetes and OpenStack domains. During the demo, we deploy a service consist of CN- and VM-based VNFs to demonstrate different features provided by our framework.

CCS CONCEPTS

• **Networks → Cloud computing**; **Network management**.

KEYWORDS

Network Function Virtualization, Software-defined Networking, Cloud Computing, service orchestration, OpenStack, Kubernetes

ACM Reference Format:
Hadi Razzaghi Kouchaksaraei and Holger Karl. 2019. Demo: Service Function Chaining Across OpenStack and Kubernetes Domains. In *DEBS '19: The 13th ACM International Conference on Distributed and Event-based Systems (DEBS '19), June 24–28, 2019, Darmstadt, Germany.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3328905.3332505

1 INTRODUCTION

Bringing Containers (CNs) to Network Function Virtualization (NFV) obtains momentum. Containerization is a popular virtualization technique as it provides remarkable advantages over Virtual Machines (VMs) such as lower overhead, faster startup, less maintenance, and easier deployment. These benefits help Communication Service Providers (CSPs) to reduce their costs and provide operational efficiency and service agility. Although the results of some studies show the feasibility of containerising some VNFs [3], it is still not feasible to containerise all VNFs due to the immaturity of CNs on different aspects such as security and fault isolation [13]. In this situation, coexistence of VMs and CNs is proposed [7]; this can combine VMs' and CNs' advantages and provide benefits such as reduced cost and total latency of NFV services compared to the case that only VMs are used to host VNFs.

Despite all the benefits that can be gained by integrating CNs into NFV environments, management and orchestration challenges hinder the use of CN-based VNFs. On the one hand, current NFV orchestrators such as Open Source MANO (OSM) [10] or SONATA [4] do not support orchestration of CN-based VNFs; on the other hand, CN orchestration tools such as Kubernetes (k8)[1] do not support NFV networking requirements such as network isolation and fixed CN IP.

This lacking support of networking requirements jeopardizes a key task in NFV, namely the chaining together of functions into a service. To chain network functions, NFV orchestrator, usually with the help of a Software-Defined Networking (SDN) controller, sets up the network in a way that all incoming flows to a network service go through a specific order of VNFs. Although there are service-chaining solutions for VM-based VNF in OpenStack[2] [11] (i.e., a cloud management system for VMs), CN orchestrators like K8 are still incapable of providing service chaining. One of the challenges here is that IP addresses are dynamically allocated to CNs by K8 and can change multiple times during the lifecycle of CNs. This makes it difficult to chain CNs with other VNFs located outside K8 domains.

This paper contributes to service chaining across heterogeneous, VM- and CN-based VNFs that are managed by OpenStack and K8, respectively. To this end, we present Pishahang: a framework built upon state-of-the-art NFV, SDN, and Cloud computing management and orchestration tools and technologies to provide orchestration for services deployed across multiple technological (VMs/CNs) domains. The remainder of the paper is as follows. In Section 2, we review related work. Pishahang is presented in Section 3 and Section 4 describes the demonstration plan. Finally, in Section 5, we conclude the paper.

2 RELATED WORK

The European Telecommunications Standards Institute (ETSI) NFV Group has proposed a specification of the NFV MANO framework to manage the lifecycle of VNFs and orchestrate network services. Based on these specifications, MANO frameworks such as OSM and SONATA have been developed. Irrespective of all their pros and

[1]https://kubernetes.io/
[2]https://www.openstack.org/

cons, none of them can manage or orchestrate network services that consist of CN-based VNF.

There are also management and orchestration tools and frameworks in the IT/Cloud environment, where CNs are already in use, such as OpenStack, K8, Terraform[3], and Cloudify[4]. OpenStack and K8 are more cloud managers than NFV mangers. They cannot meet the NFV management requirements (e.g., multi-domain orchestration) and they are used as Virtual Infrastructure Manager (VIM) in NFV. Terraform is a tool that sits on top of K8 and OpenStack and provides multi-cloud [12] functionalities for cloud service providers and allows them to deploy cloud services on multiple cloud infrastructures such as AWS and Google Cloud. But, like other cloud orchestrators, Terraform also has been implemented for the cloud environment and does not fit into NFV environments as it cannot provide the requirements of NFV services (e.g., service chaining). Cloudify has been extended to support NFV requirements. It utilises ARIA [1] to orchestrate VNFs and provides a plugin that allows operators to deploy CNs on a K8 cluster. Cloudify supports hybrid VM/CN VNFs where CN-based VNFs are deployed on VMs. However, it does not support chaining of network services where CN- and VM-based VNFs are deployed across CN and VM domains.

As mentioned in Section 1, networking-related issues of CN hinder the integration of CNs into an NFV ecosystem. In this regard, Intel proposed and implemented new services [8] for K8 that can solve a subset of the problem. Examples are the Multus Container Network Interface (CNI) plugin [9] that allows CNs to be connected to more than one network interface (i.e., it is needed for VNFs to provide redundancy of the network and separate data plane from control plane) and CPU Core Manager [2] that manages pools of CPU cores and constrains workloads to specific CPU cores within those pools (a VNF-desirable feature that was missing in K8 [8]). MetalLB[5] is a Google project that provides a network load balancer implementation for bare-metal clusters. Its services (like address allocation) allow providing fixed IP addresses for CNs that can be used for chaining services.

Despite having all these tools and technologies originated from different technological environments (NFV, Cloud), there has not been any orchestration tool that can chain CN- and VM-based VNFs across different technological domains.

3 PISHAHANG

Pishahang is a multi-domain orchestrator that has been initially introduced in [5]. While Pishahang on its initial release could support the joint deployment of VM- and CN-based VNFs on infrastructures managed by OpenStack and K8, it was missing the dynamic service chaining support across CN and VM domains. In this work, we extended Pishahang to support the missing feature. Pishahang is built on top of the SONATA MANO framework. SONATA has been selected as the base MANO framework over its competitors mainly because of two reasons: (i) SONATA follows a microservice-based architecture that allows new functionalities to be added simply by creating and integrating a new microservice (i.e., this makes extending the MANO system much easier) and (ii) it also allows network services to bring their own orchestration code along with other service artefacts by the concept of Service-Specific Management [6] which increases the flexibility of the MANO framework to meet the service requirements that are not pre-supported.

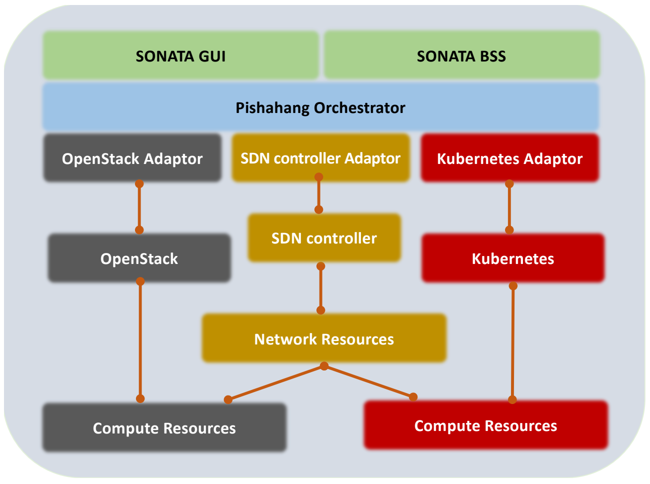
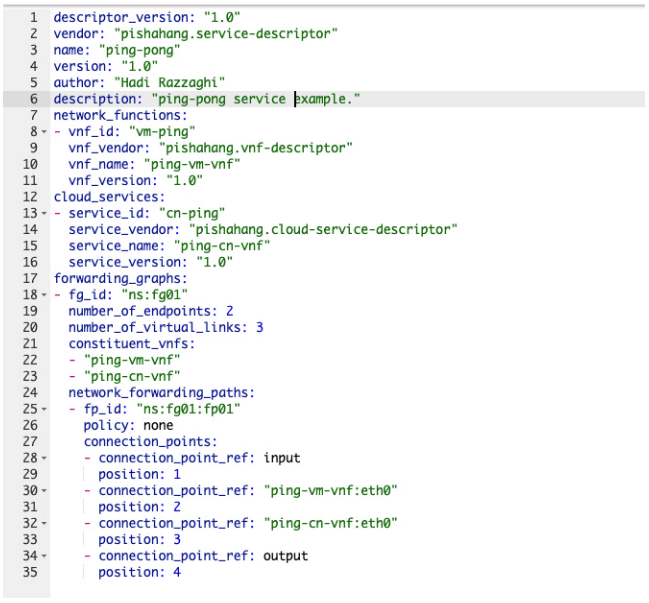


Figure 1: Pishahang high-level architecture

Fig. 1 shows the high-level architecture of Pishahang. To support service function chaining, first, we extended the descriptors schema that allows us to describe chaining-related requirements. For example, the schema, on the service level, allows defining a service graph to be used for chaining across heterogeneous domains. Fig. 2 illustrates an example of a service descriptor. In the MANO framework, we extended the K8 adaptor to provide CN's networking metadata to the orchestrator. This Terraform-based adaptor enables Pishahang to instantiate, start, update, and delete CN-based VNFs. For VM-based network functions, the SONATA adaptor is used to access OpenStack APIs. To chain services, the third adaptor has been developed; it is responsible for connecting the Pishahang orchestrator to an SDN controller. The SDN adaptor translates the service graphs included in the descriptors to a chain of MAC and IP addresses to be sent to the SDN controller. The SDN controller then converts the chain to forwarding rules and installs them on switches' forwarding tables. On top of all adaptors, the Pishahang orchestrator carries out intra-domain management tasks and orchestrates the service as a whole. On the K8 side, we employed MetalLB which allows K8 to allocate fixed IP addresses to CNs on bare-metal clusters. These IP addresses can be accessed externally and enable the Pishahang orchestrator to connect VM-based VNFs to CN-based VNFs and chain services across VM and CN domains.

To avoid reinventing the wheel, we used SONATA's Graphical User Interface (GUI) and Business Support System (BSS) to on-board descriptors and instantiate services, respectively. These components have been slightly extended to make them compatible with Pishahang needs.

[3]https://www.terraform.io/
[4]https://cloudify.co/
[5]https://metallb.universe.tf/



```
descriptor_version: "1.0"
vendor: "pishahang.service-descriptor"
name: "ping-pong"
version: "1.0"
author: "Hadi Razzaghi"
description: "ping-pong service example."
network_functions:
- vnf_id: "vm-ping"
  vnf_vendor: "pishahang.vnf-descriptor"
  vnf_name: "ping-vm-vnf"
  vnf_version: "1.0"
cloud_services:
- service_id: "cn-ping"
  service_vendor: "pishahang.cloud-service-descriptor"
  service_name: "ping-cn-vnf"
  service_version: "1.0"
forwarding_graphs:
- fg_id: "ns:fg01"
  number_of_endpoints: 2
  number_of_virtual_links: 3
  constituent_vnfs:
  - "ping-vm-vnf"
  - "ping-cn-vnf"
  network_forwarding_paths:
  - fp_id: "ns:fg01:fp01"
    policy: none
    connection_points:
    - connection_point_ref: input
      position: 1
    - connection_point_ref: "ping-vm-vnf:eth0"
      position: 2
    - connection_point_ref: "ping-cn-vnf:eth0"
      position: 3
    - connection_point_ref: output
      position: 4
```

Figure 2: An example of network service descriptor

4 DEMONSTRATION

To showcase service chaining across OpenStack and K8, we consider a scenario in which a network service consisting of CN- and VM-based VNFs will be deployed using Pishahang. In the following sections, we explain the implemented network service, the demo scenario, and the demonstration steps in detail.

4.1 Demo network service

Fig. 3 shows the demo network service and its constituent VNFs. The service consists of two VNFs, one VM-based VNF, which forwards ICMP ping requests to the next hop in the service chain, and one CN-based VNF, which does the same. As shown in Fig. 3, the VM-based forwarder will be connected to a source (SRC) host and the CN-based forwarder will be connected to a destination (DST) host to show Pishahang’s capability of end-to-end service chaining. SRC and DST are physical hosts located in different domains. Although this is a simple network service, it allows us to fully validate the capability of Pishahang to chain services across K8 and OpenStack domains. Using a more complicated VNF such as Load balancer or Deep Packet Inspection (DPI) would validate the chaining of VNFs across different domains no better. Using this service, the VNF chaining validation will be performed by the demo scenario explained below.

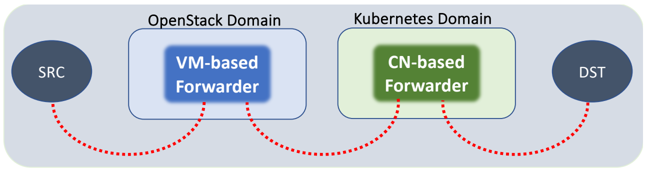


Figure 3: Implemented network service

4.2 Demo Scenario

To validate Pishahang’s capability to chain services across heterogeneous domains, first, we deploy the demo service and show that all ICMP ping requests sent from SRC to DST will be redirected to the VM-based forwarder, from there to the CN-based forwarder, and finally, the packets will be forwarded to DST. Next, we stop one of the VNFs in the chain and show that the packets do not reach DST anymore.

4.3 Demonstration Steps

The following steps will be taken during the demonstration.

(1) Demonstration of the demo setup.
(2) Demonstration of the demo service descriptors (NS and VNF descriptors).
(3) On-boarding the demo network service.
(4) Deployment and management of the demo network service across OpenStack and K8 domains.
(5) Demonstration of packet steering across VNFs.
(6) Stopping one of the VNFs.
(7) Demonstration of halt in packet steering across VNFs.

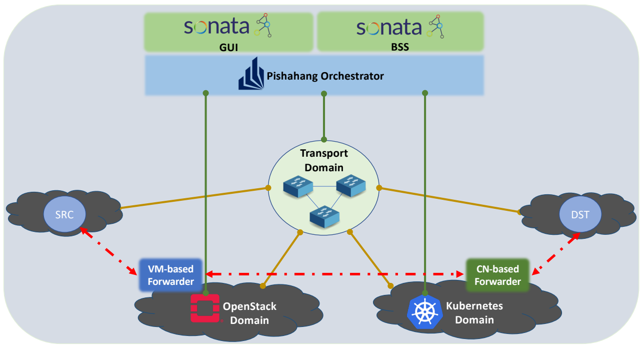


Figure 4: Demo setup overview

Fig. 4 shows an overview of the demo setup. There is also a YouTube video available that shows the planned demo[6]. Our implementation source code is also available at GitHub[7].

5 CONCLUSION

The framework demonstrated in this paper allows bringing CNs to NFV by managing, orchestrating and chaining network services consisting of VNFs realised by CN and VM virtualization techniques on infrastructures managed by K8 and OpenStack, respectively. This is enabled by combining and extending state-of-the-art Cloud Computing, SDN, and NFV tools and technologies offering advantages such as reusability improvement and reducing maintenance overhead. The implementation code of Pishahang is open-source and freely available.

[6]https://youtu.be/ceebA-BaoyM
[7]https://github.com/CN-UPB/Pishahang

ACKNOWLEDGMENTS

This work has been partially supported by the 5G-PICTURE project, funded by the European Commission under Grant number 762057 through the Horizon 2020 and 5G-PPP programs and the German Research Foundation in the Collaborative Research Centre On-The-Fly Computing (SFB 901).

REFERENCES

[1] ARIA [n. d.]. Apache ARIA TOSCA Orchestration Engine. URL: http://dpdk.org/ [retrieved: January 2018].
[2] CPU Manager [n. d.]. CPU Core Manager for Kubernetes. URL: https://github.com/intel/CPU-Manager-for-Kubernetes [retrieved: February 2019].
[3] R. Cziva and D. P. Pezaros. 2017. Container Network Functions: Bringing NFV to the Network Edge. *IEEE Communications Magazine* 55, 6 (June 2017), 24–31. https://doi.org/10.1109/MCOM.2017.1601039
[4] Sevil Dräxler et al. 2017. SONATA: Service Programming and Orchestration for Virtualized Software Networks. In *IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 973–978.
[5] H. R. Kouchaksaraei, T. Dierich, and H. Karl. 2018. Pishahang: Joint Orchestration of Network Function Chains and Distributed Cloud Applications. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. 344–346. https://doi.org/10.1109/NETSOFT.2018.8460134
[6] H. R. Kouchaksaraei, S. DrÃďxler, M. Peuster, and H. Karl. 2018. Programmable and Flexible Management and Orchestration of Virtualized Network Functions. In *2018 European Conference on Networks and Communications (EuCNC)*. 1–9. https://doi.org/10.1109/EuCNC.2018.8442528
[7] H. R. Kouchaksaraei and H. Karl. 2018. Joint Orchestration of Cloud-Based Microservices and Virtual Network Functions. In *Cloud Computing 2018 : The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization*. 153–154.
[8] M Siddiqui and T Radi and L Obuchowicz, P Rutkowski. 2017. *Enabling New Features with Kubernetes for NFV*. White Paper. Intel.
[9] Multus [n. d.]. Multus Container Network Interface (CNI). URL: https://github.com/intel/multus-cni [retrieved: February 2019].
[10] OSM. 2018. *OSM Release Four a Technical Overview*. Group Specification. ETSI.
[11] OSSC [n. d.]. Service Function Chaining Extension for OpenStack Networking. URL: https://docs.openstack.org/networking-sfc/latest/ [retrieved: February 2019].
[12] Dana Petcu. 2013. Multi-Cloud: Expectations and Current Approaches. In *Proceedings of the international workshop on Multi-cloud applications and federated clouds*. ACM, 1–6.
[13] Csaba Rotter, Lóránt Farkas, Gábor Nyíri, Gergely Csatári, László Jánosi, and Róbert Springer. 2016. Using Linux containers in telecom applications. In *19th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. 234–241.

A APPENDIX

A.1 Demonstration Requirements

The demo will be executed remotely on servers located at Paderborn University. It requires a power outlet for a laptop, proper Internet connection, and two large screens to show the dashboards and VNF terminals. A wall or stand to mount a poster will help to describe the demo to the audience.