Pishahang: Joint Orchestration of Network Function Chains and Distributed Cloud Applications

Hadi Razzaghi Kouchaksaraei, Tobias Dierich, Holger Karl

Computer Network Group
Paderborn University, Paderborn, Germany
{hadi.razzaghi, tdierich, holger.karl}@uni-paderborn.de

Abstract—Developing cloud applications using a microservice architecture allows their functional blocks to be distributed and deployed on multiple Cloud infrastructures. This enables service providers to mix and match Cloud-based microservices and Virtual Network Functions (VNFs) that are provided by Network Function Virtualization (NFV). Provisioning complex services containing VNFs and Cloud-based microservices across NFV and cloud infrastructures can enhance service quality, reduce latency, and optimise cost. This can be provided by an orchestration system that can handle cross-ecosystem dependencies. To this end, we implemented Pishahang that is a framework for jointly managing and orchestrating virtual network functions and Cloud-based microservices. During the demo, we deploy several complex services to demonstrate features provided by Pishahang to support management and orchestration of complex services.

I. INTRODUCTION

Microservice-based Cloud applications are gaining momentum as a recent study conducted by NGINX [1] states that microservices are either used or investigated by nearly 70% of organisations, with more than 30% currently using microservices in production. In microservice-based application, individual software components of an application are implemented as separate lightweight functional blocks, called microservices. This type of application can also smooth the road to the realisation of Distributed Cloud Computing (DCC) where microservices of a cloud application are deployed on geographically distributed micro data centres instead of on a single data centre. Similar to distributed cloud applications, Network Function Virtualization (NFV) services, which cloudify hardware-based network services, consist of a set of distributed virtualised Network Functions (NFs) that are chained together to deliver a network service (e.g., Set-Top Box).

The distributed structure of NFV services and distributed cloud applications enables to mix and match VNFs and microservices. Such combined services, which we call complex services, are beneficial for all actors (users, service and infrastructure providers) involved in both NFV and cloud ecosystems by optimizing cost, reducing total latency, and improving service performance [2]. An example complex service (Fig. 1) includes a load balancer that spreads the load among application back-end instances and a firewall that filters incoming requests to the application front-end.

The mentioned above advantages of complex services can be achieved if their constituent VNFs and microservices can run on their specific ecosystems. For example, running a firewall in a cloud ecosystem would not be efficient as cloud computing ecosystems are not designed to meet the packet processing performance required by VNFs. On the other hand, cloud services (e.g., a DataBase) do not need to bother with packet processing performance, which makes deployment of microservices in NFV infrastructure inefficient. Therefore, complex services need to be deployed across specialized infrastructures. This can be eased by a centralised management and orchestration system that can handle cross-ecosystem dependencies. To this end, we implemented Pishahang to support management and orchestration of complex services.

II. RELATED WORK

Multiple Cloud and NFV management systems and technologies are used to manage Cloud and NFV services. However, a management system capable of meeting requirements of complex services is missing. In the following, we mention some of these technologies and discuss their capabilities to manage complex services.

OSM [3], an NFV MANO framework, supports separate infrastructure adaptors for OpenVIM, OpenStack, VMware vCloud, and AWS. The infrastructure adaptors use these platforms API to provision and manage services. It uses infrastructure-platform-independent descriptors to describe service requirements. However, OSM does not allow network operators to jointly manage and orchestrate microservices and VNFs due to the lack of support for Cloud-based microservices (no means to describe cloud services, and lack of support for container-based management systems).
Cloudify [4], a native cloud management system, utilizes ARIA [5] to orchestrate VNFs and provides a plugin that allows operators to deploy Cloud services on an existing Kubernetes cluster. It also supports orchestration of VNFs and microservices by allowing service developers to use multiple infrastructure provider plugins per service definition (so-called blueprints). However, it is not possible to chain VNFs and microservices. Moreover, Cloudify’s blueprints depend on the specific infrastructure platform (AWS, Azure, etc.), thus forcing developers to re-write their service blueprints when switching infrastructure platforms.

Multi-cloud tools such as Terraform [6] and LibCloud [7] allow service providers to deploy services on multiple cloud infrastructures (e.g., AWS, Google Cloud). These tools can handle cross-cloud dependencies, however, they lack support for NFV requirements (e.g., service chaining).

III. PISHAHANG

Pishahang deploys, manages, and orchestrates complex services by consolidating and extending current Cloud and NFV tools and technologies. As it is depicted in Fig. 2, the main tools used in Pishahang includes SONATA [8] (a network service development and orchestration framework), Terraform [6] (a multi-cloud tool), and cloud management systems (e.g., OpenStack, Kubernetes).

![Fig. 2. Pishahang high-level architecture](image)

In Pishahang framework, SONATA carries out the high-level orchestration tasks of complex services including VNF/microservice chaining, monitoring, scaling and so on. We opted to use SONATA as an orchestrator as it allows per-service management and orchestration, which enables service developers to customize the lifecycle of services and also enhance the capability of the MANO framework to support specific service requirements that are not supported by pre-implemented and generic modules/components of the MANO framework. This is a valuable functionality for complex services that have management requirements other than conventional network/cloud services/applications.

To make SONATA compatible with our needs, we extended some of its components. These components include Business Support System (BSS), Graphical User Interface (GUI), gatekeeper, lifecycle management, monitoring, and placement. We also designed descriptors that describe complex services on two levels. A high-level descriptor, (i.e., called complex services descriptor) to describe general information and dependencies of complex services (e.g., constituent VNFs/microservices, service chaining) and low-level descriptors including VNF and microservice descriptors, which are used to describe specific requirements of microservices and VNFs (e.g., resource requirements).

Terraform and SONATA infrastructure adaptor are used to connect the orchestrator to Cloud and NFV infrastructures. Terraform provides a high-level syntax for describing how cloud resources and services should be created, combined, and provisioned on multiple cloud provider resources. It supports a wide range of Cloud providers (e.g., AWS, Google Cloud) and eliminates the need for implementing specific adaptor for each Cloud provider to deploy complex services. To make Terraform serving SONATA orchestrator, we implemented a module that translates orchestration instructions to configuration scripts to be injected into Terraform.

SONATA infrastructure adaptor is used along with Terraform as it can provide specific inter-infrastructure NFV management requirements (e.g., VNF chaining) that are not supported by Terraform. An Open vSwitch (OVS) [9] adaptor is also implemented which joins Cloud and NFV infrastructures and provide intra-infrastructure service chaining for complex services. Combining Terraform with SONATA infrastructure and OVS adaptors provides a unified NFV and cloud infrastructure that can be used by the orchestrator to deploy complex services.

IV. DEMONSTRATION

To showcase supporting complex services, we will deploy one using Pishahang on a Kubernetes infrastructure and an OpenStack NFV infrastructure. The implemented complex service and demonstration steps are explained in the following subsections.

A. Used Complex Service

The complex service that we have implemented for the demo is based on a social network scenario. The service consists of multiple microservices and VNFs. The main microservice is a social network (similar to Facebook) which uses a MySQL database as its storage backend and a Redis server for caching (used to cache settings and database queries). These microservices are containerised and will be deployed using Kubernetes. For the NFV side, a VNF cache is used to cache static files such as profile images. Furthermore, Snort is used to build an intrusion detection VNF. These VM-based VNFs will be deployed using OpenStack. The components and chaining between them are visualized in Fig. 4.

B. Demonstration Steps

The followings are steps that will be taken during the demonstration.

1) Specification and setup of the demo infrastructure.
Fig. 3. Screenshots from SONATA, OpenStack, and Kubernetes dashboards showing the deployment of the complex service and monitoring data

V. CONCLUSION

Our demonstrated framework provides a joint orchestration and lifecycle management for distributed services that are deployed across NFV and Cloud infrastructures. This is provided by consolidating and extending state of the art cloud and NFV tools and technologies, which offers advantages such as reusability improvement and reducing maintenance overhead. Pishahang’s implementation code is open-source and freely available.

ACKNOWLEDGMENT

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