ONAP
OPEN NETWORK AUTOMATION PLATFORM

OOM

Solution Brief

ONAP Operations Manager Utilizes Kubernetes for ONAP Lifecycle Management

By using Kubernetes to manage ONAP, users can reduce cost, prevent lock-in and increase automation.

INTRODUCING ONAP:

- End-to-end design and orchestration of NFV and SDN services
- Model-driven architecture to enable automation
- Real-time, closed-loop automation to reduce manual repetitive tasks

INTRODUCING KUBERNETES:

- Container orchestration engine manages groups of containers
- Natively features scale-out, self-healing and upgrade features
- Based on 15 years of running production workloads at Google

KUBERNETES ENABLES ONAP LIFECYCLE MANAGEMENT:

- Initial (day-1) deployment of containerized ONAP services
- Ongoing (day-2) lifecycle management and monitoring of ONAP
- Roadmap to enhance for large-scale ONAP deployments

BENEFITS OF USING ONAP WITH KUBERNETES:

- Simple, reliable approach to deploying ONAP
- Improves resource efficiency and speed of deployment
- Reduces lock-in by enabling ONAP deployment on any public or private cloud
- Enhances automation with rich orchestration and multi-tenancy features of Kubernetes

Introducing ONAP

Communications Service Providers (CSPs) are undergoing massive operational transformation by using Software Defined Networking (SDN) and Network Functions Virtualization (NFV) to bring agility to network services and simultaneously reduce costs.

The software stack required to run NFV and SDN consists of several components. The NFV infrastructure (NFVI), Virtualized Infrastructure Manager (VIM) and SDN controller make up the NFV cloud. This NFV cloud runs one or more Network Services (NS) that is comprised of several Virtualized Network Functions (VNFs). The automation of this entire platform is performed by the Management and Orchestration[1] (MANO) software component. The Open Network Automation Platform (ONAP) is an open source project that provides a superset of the MANO functionality.

[1] Strictly speaking MANO includes VIM and SDN Controller. For purposes of this document, we are excluding these components from the definition of MANO and restricting it to NFV orchestration and VNF management.

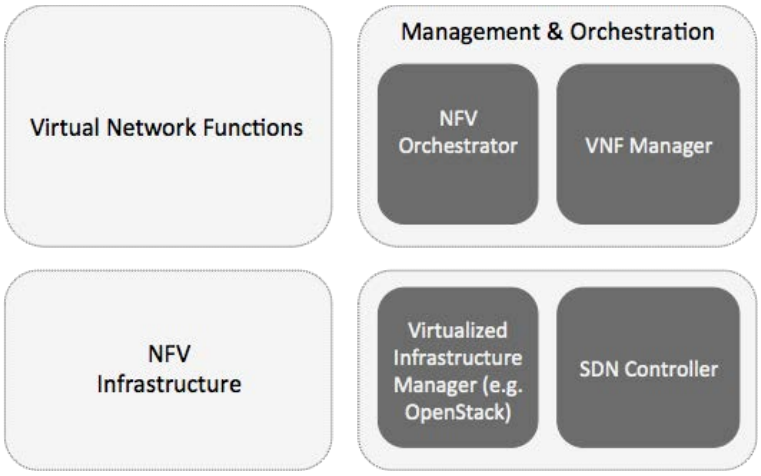

Figure 1: NFV Architecture

ONAP provides a common platform for telecommunications, cable and cloud operators and their solution providers to rapidly design, implement and manage differentiated services. It provides orchestration, automation and end-to-end lifecycle management of network services. It includes all of the MANO layer functionality specified by the ETSI NFV architecture; additionally, it provides tools for network service design and a framework for closed-loop automation.

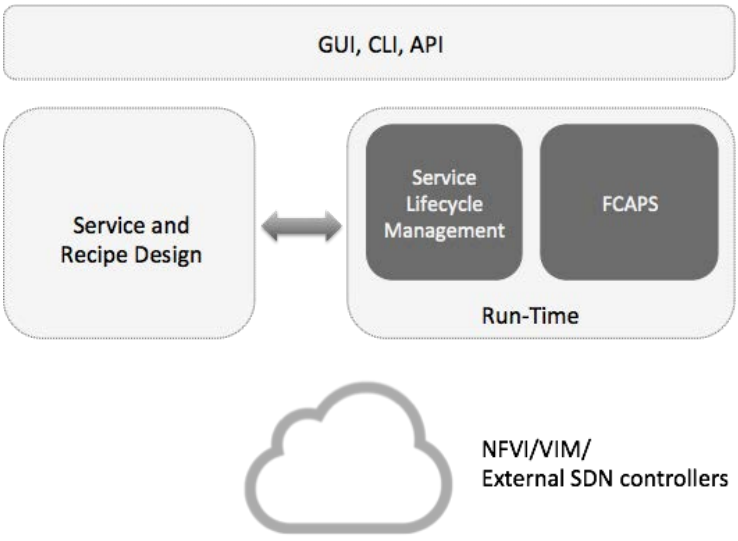

Figure 2: ONAP Functionality

Introducing Kubernetes

Containers are an alternate form of virtualization technology for compute resources.

While virtual machines (VMs) virtualize the entire machine and each VM has its own instance of operating system, a container instead virtualizes the operating system. For this reason, containers are up to 10x denser than VMs and boot up to 10x faster.

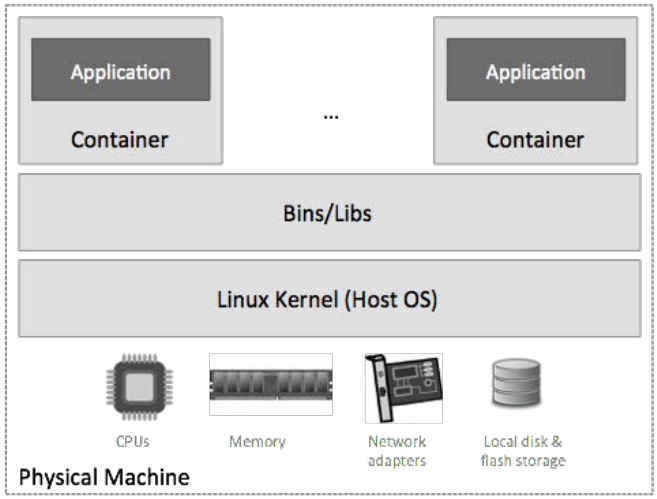

Figure 3: Container Architecture

Containers have proven to be effective in cloud-native architectures where an application is decomposed into microservices, and each microservices does one thing. Cloud-native architectures in turn have dramatically increased software velocity as each microservice can be created and maintained by a small independent team.

Given the sheer number of containers a cloud-native application may consist of, it is impractical to manage each container individually. Instead, containers are managed using a Container Orchestration Engine (COE) such as Kubernetes, an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.

Kubernetes can orchestrate containers on bare metal servers or in virtual machines. These options allow containers to be used with private clouds or public clouds such as AWS, GCP, Azure and others.

Managing ONAP with Kubernetes

ONAP is a cloud-native application that consists of numerous services.

For this reason, ONAP requires sophisticated initial deployment as well as post-deployment management. The ONAP Operations Manager (OOM) project is responsible for end-to-end lifecycle management and monitoring of ONAP.

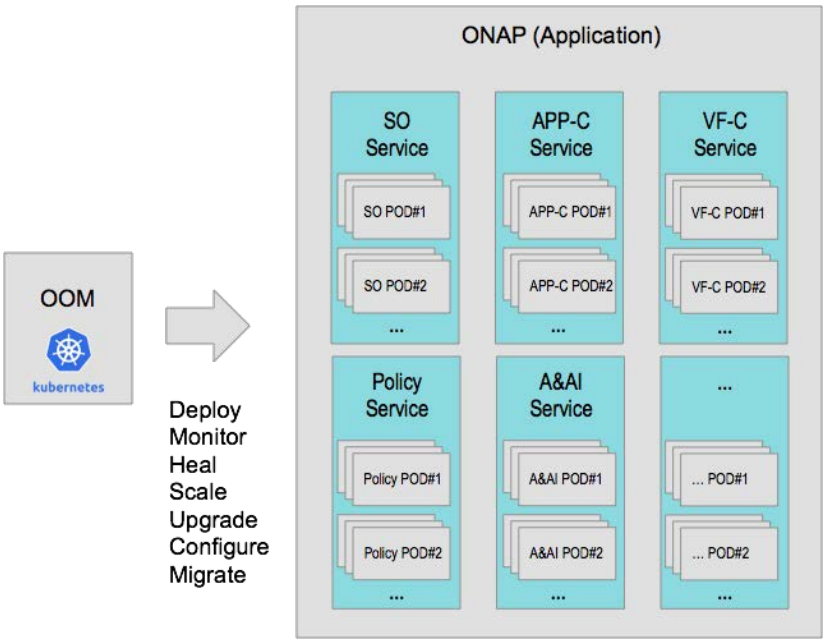

Figure 4: OOM Functionality

Readers should note that OOM does not provide support for containerization or orchestration of network services or VNFs that are managed by ONAP. Instead OOM orchestrates the lifecycle of the ONAP platform components.

A Kubernetes deployment consists of the following:

- **Nodes:** Physical machines or VMs that host multiple containers; this is also called a worker machine. If ONAP containers are hosted in VMs, the VMs may be created manually or using automation such as a OpenStack Heat template, Cloudify TOSCA template, AWS CloudFormation and so on.
  Each node requires an operating system — either traditional or a lighter weight, container-native operating system.

- **Services:** An application, such as ONAP, is decomposed into a number of Kubernetes Services. Note: a Kubernetes Services is not the same as a Network Service.
- **Pods:** ONAP Kubernetes Services are further decomposed into a set of Pods. A Pod is made up of one or more containers that provide a specific functionality, though at this time, OOM associates a Pod with one container. Pods are connected to each other using an overlay network. The Service–›Pod structure allows a particular Service to perform lifecycle management operations on its Pods (e.g. scale-out) without impacting other Services.
- **Persistent volumes:** These are used to store non-ephemeral ONAP state and configuration data.

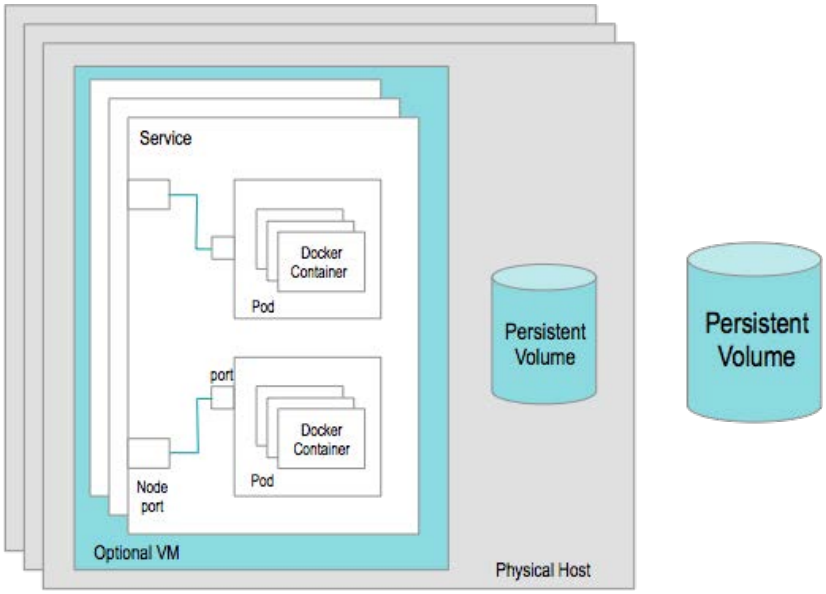

Figure 5: ONAP usage of Kubernetes

In total, OOM uses roughly 100 containers. Each of the containers is monitored to make scaling and healing decisions. Additionally, OOM healing operations are aware of dependencies, thus avoiding situations such as continuously restarting a container in a crash loop.

OOM takes advantage of several Kubernetes innate features to provide ONAP with functionality such as:

- Initial deployment with in-built dependency rules
- Unified configuration across all ONAP components
- Ongoing real-time health monitoring of ONAP components
- Healing, where failed ONAP components are restarted automatically
- Scaling, where ONAP components can be clustered and seamlessly scaled
- Upgrade with little to no impact on the overall ONAP availability
- Delete to cleanup the entire ONAP installation

Benefits of OOM and Kubernetes

The combination of Kubernetes, containers and ONAP provides numerous benefits to users:

- **Efficiency—**Containers are inherently more compact than VMs. With OOM, ONAP can be deployed with a minimum resource requirement of 156-220GB memory and 54 vCPUs[2]. In contrast, an equivalent VM-based deployment of ONAP requires[3] 336GB memory and 148 vCPUs. The delta directly contributes to a lower CapEx.
- **Cloud independence—**Kubernetes applications can run on any cloud. For this reason, OOM can deploy ONAP on bare metal servers, private clouds or public clouds. ONAP can also be ported or migrated across clouds with this same feature.
- **GuestOS license fee reduction—**In situations where a user is paying for GuestOS licenses (e.g. a public cloud), an OOM approach is more cost-effective than VMs given that containers require only one shared operating system.
- **Multiple ONAP instances—**One Kubernetes cluster can support multiple ONAP instances using separate Namespaces. This can be very useful for creating separate dev, test, staging and production environments on the same Kubernetes cluster thus saving on CapEx.
- **Richer automation—**Kubernetes provides richer automation than VM-based orchestration tools. For example, in Kubernetes, services communicate with each other using fully qualified Namespaces rather than IP addresses, which avoids hard-coding IP addresses in deployment templates. Furthermore, Kubernetes has a very rich concept of inter-Pod relationships (e.g. Replica Sets, Daemon Sets, StatefulSets etc.) allowing a higher degree of flexibility in scale-out, scale-in and self-healing.

[2] wiki.onap.org/display/DW/ONAP+on+Kubernetes+on+OpenStack?focusedCommentId=30900980#comment-30900980
[3] onap.readthedocs.io/en/latest/guides/onap-developer/settingup/onap_heat.html#footprint

Next Steps

There are two types of enhancements being considered to OOM. The first one is to further shrink the footprint of containerized ONAP services so that ONAP can be deployed on a laptop.

This step is important for a number of reasons — ONAP open source contributors will be able to have a fully contained environment on their laptop. Operators wanting to "try out" ONAP will be able to do so without a server. Additionally, ONAP developers building analytics or portal applications, or using ONAP interfaces to connect to OSS/BSS applications, will be able to do so on a laptop as well.

The second enhancement is to enhance large-scale ONAP deployments by taking full advantage of Kubernetes capabilities. Some items on the roadmap are:

- High availability, reliability, repeatability
- Backup and restore of persistent volumes
- Geo-redundant deployments

Conclusion

In conclusion, the ONAP open source project provides network automation for SDN and NFV services. Moreover, ONAP itself is a cloud native application that can be containerized and managed using the Kubernetes container orchestration engine. The ONAP OOM project uses this approach to provide numerous benefits to CSPs deploying ONAP, such as efficiency, speed, cloud independence, the ability to run multiple ONAP environments and richer automation.

Resources

ONAP

Kubernetes

OOM

Rancher OS

OOM Users' Guide