NexComm 2021 Congress
The Twentieth International Conference on Networks
Advances in Software Defined Networking and Network Functions
Virtualization SOFTNETWORKING 2021

Containerization using Docker technology

Alexandru Eftimie, Eugen Borcoci

University POLITEHNICA of Bucharest

alexandru.eftimie@gmail.com

eugen.borcoci@elcom.pub.ro

Objectives

- This paper aims to provide a clearer view of container technology, such as its advantages and disadvantages, how it can cooperate with Openstack;
- We will follow the comparison of household equipment that can play an active role in computing at the periphery of networks to highlight what types of applications or calculations can be performed on them
- Describe the components of Docker technology and its benefits;

Contents

1. Introduction

- Software Defined Networking (SDN)
  - ✓ separates network control and data forwarding functions leading to centralized and programmable network control.
  - ✓ SDN architecture has several main components such as: data plane, control plane;

- Network function virtualization (NFV)
  - aims to implement by software many functions, that traditionally have been implemented as expensive hardware-software combinations.
  - Recent standards define the NFV architecture and also how to implement different virtualized network functions (NFV) in a virtual environment – to replace the traditional dedicated boxes which performed individual functions.

The SDN and NFV are complementary technologies, usable independently or in cooperation. While NFV replaces hardware network elements, SDN deals with replacement of network protocols, bringing centralized control.

2. Containers and microservices

Containerization offers significant benefits:

- **Portability**: Agility: The open source Docker engine for running containers has started the industry standard for containers with simple tools for developers and a universal presentation approach that works on both Linux and Windows operating systems. The container ecosystem has shifted to engines managed by the Open Container (OCI) initiative.
- **Speed**: Containers are often referred to as “lightweight,” which means they share the core of the machine’s operating system (OS). Not only does this lead to higher server efficiency, but it also reduces server and licensing costs, while speeding up startup times because there is no operating system to boot.
- **Defect isolation**: Each containerized application is isolated and operates independently of the others. The failure of one container does not affect the continuous operation of other containers. Development teams can identify and correct any technical issues in one container without having time to stop in other containers.
- **Efficiency**: Software running in containerized environments shares the machine's operating core and application layers in a container can be shared across containers. Thus, the containers are inherently smaller than a VM and require less start-up time, allowing many more containers to run on the same computing power as a single VM.
- **Easy to manage**: A container orchestration platform automates the installation, scaling and management of containerized tasks and services. Container orchestration platforms can make management tasks easier, such as scaling containerized applications, running newer versions of applications, and providing monitoring, recording, and debugging, among other functions.

3. Docker technology

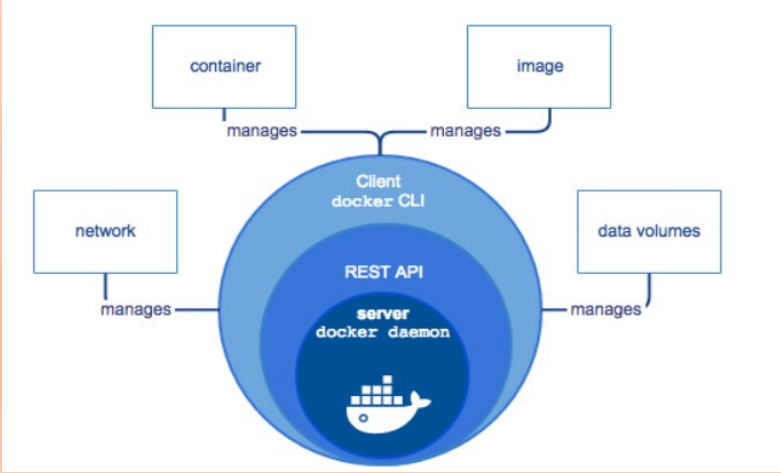
Docker is an open platform for developing, transporting and running applications. Docker provides tools and a platform to manage container lifecycle:

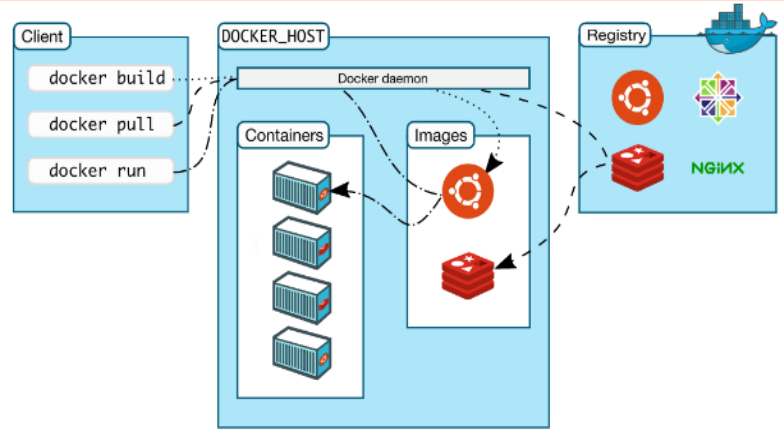
- The application and its support components can be developed using containers.
- The container becomes the unit for distributing and testing the application.
- Implementing the application in the production environment, as a container or an orchestrated service. This works the same whether the production environment is a local data center, a cloud provider, or a hybrid of the two

3. Docker technology

Docker Engine is a client-server application with the following major components:

- ✓ A server that is a type of long-term program called a daemon process.
- ✓ REST API that specifies the interfaces that programs can use to talk to the daemon and instruct it on what to do.
- ✓ A command line interface client (CLI).





4. Containers in NFV

To cope with the growing use of networks, driven by new mobile customers, and to meet the demand for new network services and performance guarantees, telecom service providers leverage virtualization on their network by implementing network services in virtual machines. Telecom service providers have begun to lose existing revenue, while suffering increased capital expenditures and operating costs that cannot be offset by rising subscription costs. To meet the challenges mentioned above, service providers have begun to migrate network infrastructure to software.

Some popular marginal devices are presented along with their release date, architecture, CPU, and memory parameters. The list includes residential equipment for large-scale customers.

Customer Device	Architecture	CPU	Memory
Residential CPE home routers			
Virgin SuperHub 3(Arris TG2492s)	Intel Atom	2x1.4 GHz	2x256 MB
Google Fiber Network Box GFRG110	ARM v5	1.6 GHz	N/A
Orange Livebox 4	Cortex A9	1 GHz	1 GB
Commodity wireless routers			
TP-LINK Archer C9 home router	ARM v7	2x1 GHz	128 MB
Ubiquity Edge Router Lite 3	Cavium MIPS	2500 MHz	512 MB
Netgear R7500 Smart Wifi Router	Qualcomm Atheros	2x1.4 GHz	384 MB
IoT edge gateways			
Dell Edge Gateway 5000	Intel Atom	1.33 GHz	2 GB
NEXCOM CPS 200 Industrial IoT Edge Gateway	Intel Celeron	4x2.0 GHz	4 GB
HPE Edgeline EL4000	Intel Xeon	4x3.0 GHz	Up to 64 GB

Conclusion

- Containers give a false sense of security. There are many pitfalls when it comes to securing applications. It is wrong to assume that one way to secure them is to place them in containers. Containers do not provide anything in themselves. If someone keeps your web application containerized, it could be locked into namespaces, but there are several ways to get rid of it depending on your configuration. We need to be aware of this and put as much effort into security
- There are Docker images that require the exposure of more than 20 ports for different applications inside a container. Docker's philosophy is that a container should do a single job and you should compose them instead of making them heavier. If you end up packing all your tools in one container, you lose all the benefits, you may have different versions of Java or Python inside, and you end up with a 20GB image that can't be managed.
- The paper managed to provide an overview of the current options for implementing "fog computing": whether it is devices already in production at the end user, or we are talking about native equipment, dedicated to perform functions specific to information processing on the periphery network. Observing these things, a variant that can be adopted is the use of dedicated servers at the periphery of the networks on which to build special functions using Docker technology. This has the advantage of a fast implementation of functions, their rapid scaling as well as the advantage of having a platform shared by many entities given the isolation.
- In this paper I have managed to provide an overview of "Docker" technology and how this technology can contribute to a better exploitation of virtual network functions. This is because Docker provides very good isolation between instances and at the same time does not require the presence of dedicated software (hypervisor), offering greater flexibility than classic virtual machines.

Thank you!