2019 4th International Conference on Information Systems and Computer Networks (ISCON)
GLA University, Mathura, UP, India. Nov 21-22, 2019

Comprehensive Analysis of Container Technology

Tamanna Siddiqui
Associate Professor
Department of Computer Science
Aligarh Muslim University, Aligarh, India
ja_zu_siddiqui@hotmail.com

Shadab Alam Siddiqui
Research Scholar
Department of Computer Science
Aligarh Muslim University, Aligarh, India
siddiqui.shadab@gmail.com

Najeeb Ahmad Khan
Director
Royal Achievers Academy
Aligarh, India
naahkh@gmail.com

***Abstract*—Containers is the new buzz word within the computer technology space. With the technology growing at an exponential pace and within a short span of time, the industry has evolved from native to virtualized, to cloud and now to container technology. This evolution is focused more on decoupling the software with the hardware and making software or applications more lean and agile. Containers has done a lot of good in achieving this objective and has also helped to cope up with the deficiencies faced in virtualized and cloud based environment. This paper tries to do a literature review of the evolution of containers and will also attempt to analyze container eco system. This article will also discuss the differences between the containers and virtual machines and will highlight the efficiencies that container technology brings over the virtualized environment.**

Keywords—Containers, Virtualization, Cloud, Microservices, Orchestration.

I. INTRODUCTION

Since last decade or so there has been an exponential advancement in terms of hardware and software technology. In accordance to these advancements there has also been an evolution of user demands and business cases. These end user demands coupled with the reachability and affordability of internet has put a lot of pressure on application providers and services. This vast all round development has led to the demands of utilizing resources efficiently and lowering the CAPEX and OPEX of any service provider including data center owners. Various technologies over the past decade has tried to address these requirement. Some of the most prominent breakthrough technologies which have addressed various aspects of these requirements are Virtualization, Cloud Computing and Container technology.

Container technology in its vanilla form is simply called as container. As the name suggests, it is an instrument of holding an item and in technological definition it can hold applications. It provides applications an environment to execute with its dependencies and isolates it from irrelevant programs and processes of the underlying Operating System. Thus, it could be defined as an application packaging mechanism having only its dependencies. It is analogous to the containers used in shipping industry, where shipping individual items is done by packaging them in pre-built containers rather than as items of different size. This gives operational efficiency to the shipping industry and synonymously computer container technology provides operational efficiency for applications bundled within containers.

The requirement of containers arises from the fact that developers had to face a daunting task in porting applications from development environment to testing/lab environments and finally to production environments. Every time an application is shifted to a new environment, it faces issues of migration because usually no two environments are similar in terms of both software and hardware capabilities. Thus, to overcome this drawback and bring in more operational efficiency, container technology comes into picture. It helps to package the application with all its required dependencies in terms of software, libraries, configuration files, binaries and other required stuff to execute the application. This helps the application to run smoothly in any environment as it is packaged with all its dependencies.

II. RELATED WORK

There has been a constant motivation upon academicians and research scholars to delve into the details of container and its related technologies. A lot of effort has been made to track the development of containers, whose evolution is more of implementation based rather than research oriented. It is one of the few technologies where industry based demand has resulted into the implementation of technology faster than the research devoted to it. Over the period of time, several key technologies have, in some way or the other, helped in developing and motivating containers, as the way it is in its current form. Some of the major concepts and technologies are reviewed underneath.

A. Microservices

These are small and single functional elements which can work independently in isolation. Dragoni N. et. al. [3] defines microservices as an architectural style and as "the minimal independent process that interact via messaging". Alan Sill in [10] has extensively talked about the design and architecture of the microservices. Microservices, as the name suggests, are single process which could be a basic application or several microservices could be externally coupled with each other by means of API's (Application Programming Interface) to build a larger application. These are reusable and independent components which can be used as a building block of various large and enterprise application.

B. Virtualization

Virtualization refers to the abstraction of hardware layer and divide hardware resources to multiple virtual resources which can then be used as different machines. The diagram in figure 1 pictorially describes the division of actual hardware

978-1-7281-3651-6/19/$31.00 ©2019 IEEE

resources to several virtual resources by a virtualization layer. The software that helps actual hardware to be virtualized is called hypervisor. This set of virtual resources can then be used as separate machines known as virtual machines which can now host an OS and application on top of it.

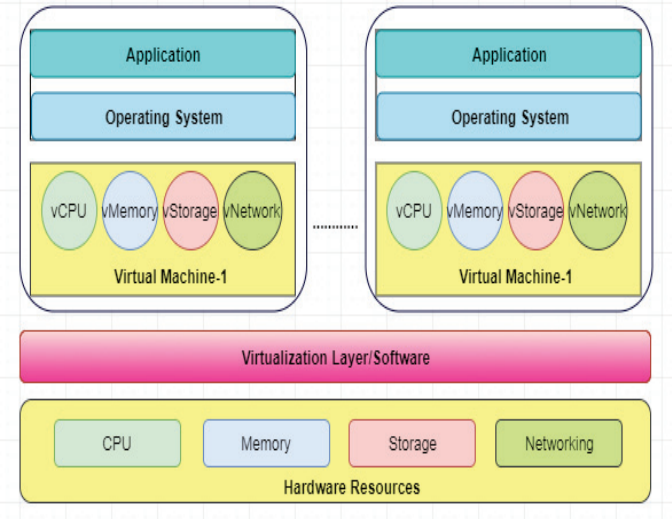

Fig. 1. Virtual Machine Architecture

C. *Cloud*

Cloud computing technology efficiently reduces the costs of IT infrastructures by offering computer resources (e.g., networks, servers, storage, applications, and services) as services. It also provides applications as services over the Internet, offers hardware and systems software in the data centers that are required to build services. Cloud model is composed of five essential characteristics –

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity and
- Measured service.

There are three prominent service models for cloud computing that is provided by cloud service vendors –

- Software as a Service (SaaS)
- Platform as a Service (PaaS) and
- Infrastructure as a Service (IaaS).

Cloud can be deployed in four types of distinct deployments – Public cloud, Private cloud, Hybrid cloud and Community cloud models [5].

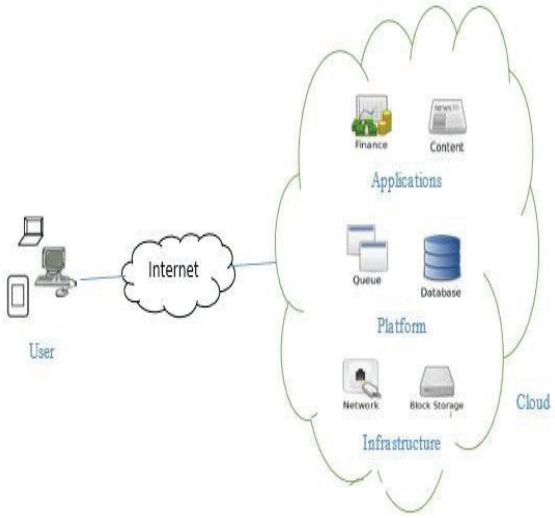

Fig. 2. Cloud Model

D. *Containers*

Containers have come of age and are now at the forefront of technology. It is simply defined as packaging unit for any application or it can also be referred to as operating system level virtualization [12]. This brings agility and ease of Life cycle management of any application. Major behemoth of IT services companies like Google, Amazon, Microsoft etc. have adopted container technology. This has helped them manage their cloud infrastructure in easier, faster and in an efficient manner.

III. METHODOLOGY USED

In-order to study the reasons of containers peculiar evolution, several aspects of the container technology as well as its related technology has been studied, analyzed and is being presented in this paper. The overall methodology and approach used to accomplish the review work can be broadly categorized in three phases, as shown in the below figure 3:

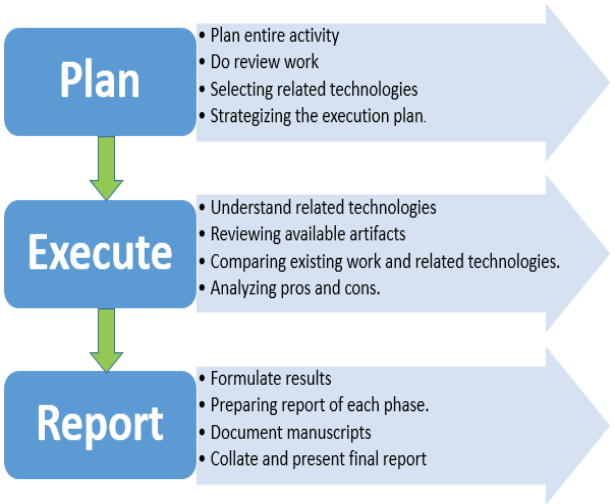

Fig. 3. Adopted Methodology

A. *Plan and Strategize*

Planning the review work and selecting related technologies which has impacted the overall evolution of container technology. Further strategizing the overall plan so that it could be executed.

B. *Execute*

Once the overall plan was chalked out, strategy was in place and topics were identified, an extensive effort was utilized to execute the plan. The execution step primarily included to understand each related topic and technology to its depth by thoroughly understanding the available work around it and systematically comparing its effectiveness, drawbacks and other aspects which helped in development of container technology.

C. *Report*

The final phase incorporated the work of streamlining and documenting the results of first two phases. Careful reporting of each technology study was documented and later all such documentation was collated to prepare a manuscript of entire exercise. And finally it resulted in structuring this paper with the crux of the entire exercise

IV. REVIEW PRESENTMENT

Various aspects of container technologies and its related fields have been studied and presented in this section. This section elaborates the most important factors in the development of containers technology. It depicts and reflects the motivation of its evolution from baby steps to leading technology in today's context.

A. *Paradox of Microservices and Containers*

When talking about containers, there is a lot of discussion on Microservices at its relation with containers. In a nutshell, microservices are not a building block for containers. In fact, microservices are lightweight software components with singular scope. From architectural point of view, microservices are a single function executable which communicates with outside world (other Softwares/services) using light weight API's. Nuha et. al. [1] have done a detailed study on microservices architecture and their implementation.

On the other hand, containers are not microservices but can be viewed as packaging on top of micro-services. Microservices being independent and tiny functional executables need a service that makes its deployment very agile, scalable and portable in nature which can ease its life-cycle management. Containers fit this requirement perfectly and this means that a container can host a micro-services and enhance their ease of use and maintainability. The below figure 3 illustrates the relationship between containers and microservices. As pictorially represented in the diagram below, containers act as a packaging unit for the encapsulating the microservice along with its dependencies such as binaries and libraries.

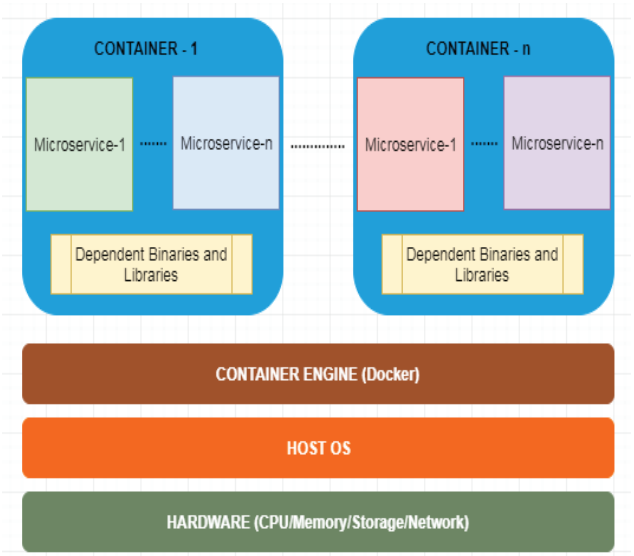

Fig. 4. Relationship between Microservices and Containers

A container can host one or more than one microservices as envisaged by the application design. In addition to this, in a virtualized environment there could exists one or more containers and each container could host one or more microservices. This environment could be related to a single application or several applications running independently.

B. *The Container Chronology*

The evolution of containers started way back in 1970's and since then it has been a stop gap based development. Container technology or Containerization is not a knee jerk invention but rather it is a culmination of years of effort which is now finding a vast audience. Osnat [2] in her blog captures elaborately, the time wise journey of Container evolution.

The concept of containers is not new and its existence can be tracked back to 1979 when chroot system call was introduced in Unix V7 operating system. Since then, several isolated implementations have come up around container logic but a complete implementation of container technology was done by Linux in 2008 and it was named as LXC (Linux Containers). Using LXC, Dockers came into existence in 2013 and made containers very popular amongst the technologist and as of now is a leading container technology provider. RightScale in its cloud survey has indicated that Docker usage expanded from 35 to 49 percent [13]. Docker offers an entire ecosystem revolving around containers, managers, orchestrators and other suits of use cases. Docker started its journey with LXC but now it has replaced LXC with its own libcontainer library.

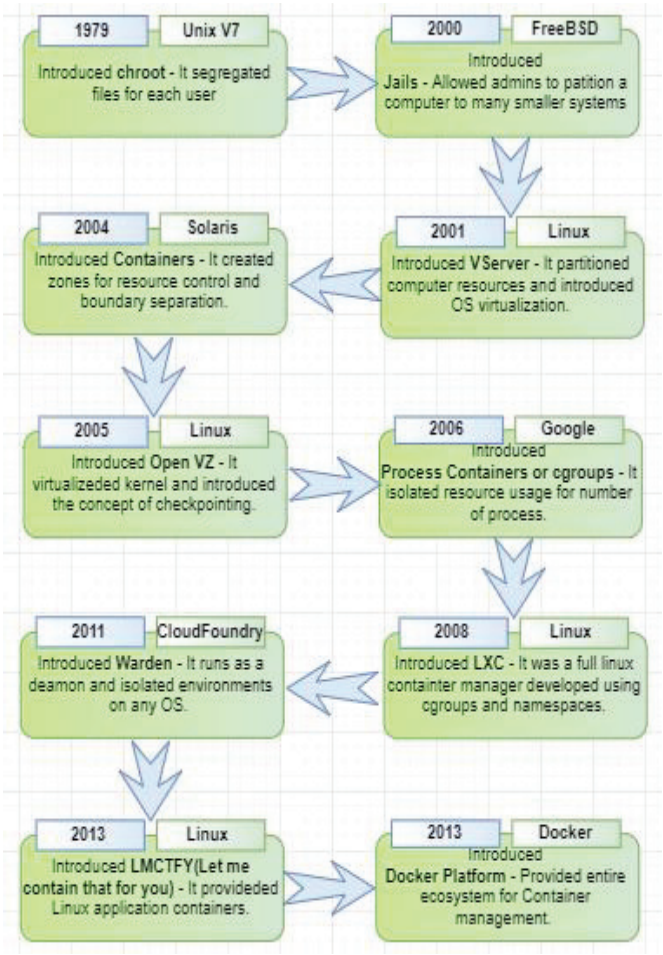

Fig. 5. Chronology of Container Evolution

C. *Containers versus Virtualization*

There is a lot of debate over the pros and cons of container and virtualization. And few novice technologists even miss-interpret both technologies to an extent that they hardly can differentiate between the two. This is because both these technologies aim to abstract and decouple the underlying hardware from the application. There is a marked difference of how both these technologies present themselves to achieve the objective of software decoupling from the hardware. Both these technologies have been effectively compared in [6] over several parameters.

The primary level of virtualization in a Virtual Machines is the virtualization of the underlying hardware whereas Containers aim to virtualize the operating system itself thus giving each containered application a separate area of execution within the operating system which in turns results in sharing of the operating system by various container based applications or services. VMs are an ideal choice when the applications have different OS whereas containers would be optimal choice when the underlined share OS is same [14].

Marcelo et. Al. in [11] have done detailed experiment on the CPU performance, Overhead of virtual container creation, overhead of nested container creation and network performance of containers. There are several characteristics where both containers and virtualization can be judged and compared. Table 1 captures the gist of these parameter.

TABLE I. COMPARING CONTAINERS AND VIRTUAL MACHINES

Characteristic	Containers	VMs
Virtualization	OS Kernel	Hardware
OS	Shared	Dedicated
Size	Small	Large
Boot time	Low	High
Performance	Lighter	Heavy
Security	Access Control	Hypervisor
Communication	Ethernet	IPC
QoS	Better	Average

The authors in this review work have come up with a detailed study of both the technologies for their several characteristics or parameters.

- **Virtualization layer –** The VM virtualization is at the hardware level where a hypervisor virtualizes the actual hardware and makes virtual copies of the hardware to efficiently using the hardware. Scott et. Al. in [4] report that a virtual machine monitor is a very thin layer of software that sits on top of the hardware and virtualizes all its resources. On the other hand Containers virtualize the OS kernel by creating several user-space entities which are independent of each other. We can also say that containers offer Operating System virtualization.
- **Operating System –** Virtual Machines have a dedicated OS of their own. In contrast all the containers share the same operating system and its kernel.
- **Size –** Containers as a bundle take limited storage when compared to Virtual Machines. This is because each VMs have a full copy of OS whereas containers only have required binaries and other dependencies and the base OS is shared amongst various containers.
- **Boot-up time –** Boot/Start-up time of containers is considerably lower in comparison to the VMs. The start-up time for VMs is higher because it also requires the booting up of the dedicated OS whereas in containers, full-fledged OS is not part of the package. Thus application running within containers come up faster in comparison to the same application when it is deployed over a Virtual Machine.
- **Performance –** The performance and the response time of Containers are quicker in comparison to virtual machines because instruction in VMs needs translation

from guest OS to host OS through the hypervisor layer whereas this overhead is not there in containers. Felter et. al. have experimentally proved in [7] that containers usage of CPU vis-à-vis Virtual machines is lighter.

- **Security –** Security of VMs are dependent on security implementation of the underlying hypervisor and to an extent to the hardening of the guest OS. In contrast, Security in containers is achieved by access control mechanisms like namespaces.
- **Communication Mechanism –** Networking or communication mechanism employed between VMs is through Ethernet devices (IP and port) whereas Containers communicate with each other using various Inter Process Communication (IPC) mechanism like message queues, pipes etc.
- **Quality of Service (QoS) –** This parameter basically drives the user experience and is very important when devising an application and its deployment model. It has been established by Ismael et. al. in [8] that containers provide a better QoS as compared to VMs because of its compact size which can consolidate more services with the same amount of resources.

D. *Container Eco-System*

At present, containers are at the forefront of technology evolution and thus an entire eco system has developed around it. Various tools, samples, Proof of Concepts (PoC), Use cases and excessive amount of documentation has been made available to help ease of use of containers. Cito et. Al. [9] have done an extensive analysis of docker ecosystem for the docker related files in the github repository. This quick and broad spectrum development around container technology have helped to its popularity amongst the technology enthusiasts. Some of the important building blocks of this eco-system has been captured below in table 2, identifying its usability and its major providers.

TABLE II. CONTAINER ECO-SYSTEM

CONTAINER ECO-SYSTEM	
Container Platform	Helps in bundling applications with pre-requisite dependencies.
	Major Players: Dockers, kubernetes, Mesos, Vagrant etc.
Container Orchestration	Helps in Lifecycle Management of software within containers.
	Major Players: Kubernetes, Docker Swarm, Mesosphere Marathon, Nomad etc.
Container Repositories	It is a collection of container images of commonly used container and its versions.
	Major Players: Docker Hub, Amazon ECR etc.
Container as a Service	CaaS is a new service that helps provide ready to use container technology for naïve users.
	Major Players: Amazon Elastic Container Service, Google Kubernetes Engine, Microsoft Azure Container Instances.

- **Container Platform –** Several container platforms are available which help in bundling applications with pre-requisite dependencies. Most prominent player in this space is Docker whose popularity in this space is akin to the container platform. Other major container platforms include kubernetes, Mesos, Vagrant, and RancherOS..
- **Container Orchestration –** Lifecycle Management of software is an integral part of the software business. Softwares deployed on containers also require lifecycle management services for upgrades, scaling, monitoring, load balancing and other regular activities. Container orchestration refers to life cycle management of containers and several tools have been developed for container orchestration. Kubernetes, developed by google is the most popular Container orchestration software. Other example of Container orchestration tools available are Docker Swarm, Mesosphere Marathon, Nomad etc.
- **Container Repositories –** It refers to the collection of container image repositories of commonly used container and its different versions. This repository bases reduces the workload of packaging common application within containers and can be downloaded from the available repositories. All the major container provider have their container repository and the most common container repository is provided by Docker. Users can easily pull and push an image to these repositories for their use and storage.
- **Container as a Service (CaaS) -** All prominent cloud service providers now look into containers as a new business model have Container as a Service (CaaS) product for its user base. This has made using container technology even easier for naïve users to use and get the benefit of Container technology. Some of the well-known CaaS providers are Amazon Elastic Container Service (ECS), Google Kubernetes Engine (GKE), Microsoft Azure Container Instances (ACI) and IBM Cloud Kubernetes Service, among many more players in this market.

V. CONCLUSION

Containers provide a new way of virtualization which is lightweight and agile. This has helped in the popularity of containers. Adapting to containers for new applications or migrating to container architecture for existing applications have lots of benefits for the application life cycle management. It helps the application and its host to independently plan for upgrades and maintenance without the other being hampered. Although the design of the application needs to be studied for migration or before choosing container as it packaging architecture.

In this piece of work, we did an extensive study regarding the overall container technology, its eco-system and also did comparative study with virtualization in general. To conclude, containers are a perfect fit for application demanding quick life cycle management. Containers also help in reducing CAPEX and OPEX of data center manager as well as application

owners. Adapting container technology is a win-win situation for all provided the application are designed to use containers effectively.

REFERENCES

[1] Alshuqayran, Nuha, Nour Ali, and Roger Evans. "A systematic mapping study in microservice architecture." Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on. IEEE, 2016.

[2] Rani Osnat, "A Brief History of Containers: From the 1970s to 2017", 2018.

[3] Dragoni N. et al. (2017) Microservices: Yesterday, Today, and Tomorrow. In: Mazzara M., Meyer B. (eds) Present and Ulterior Software Engineering. Springer, Cham.

[4] S. Devine, E. Bugnion, and M. Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. US Patent, 6397242, Oct. 1998.

[5] Siddiqui, Tamanna, and Riaz Ahmad. "Cloud Testing – A Systematic Review." International Research Journal of Engineering and Technology (2015).

[6] R. Dua, A. R. Raja, D. Kakadia, "Virtualization vs containerization to support paas", Proceedings of the 2014 IEEE International Conference on Cloud Engineering ser. IC2E '14, pp. 610-614, 2014.

[7] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), March 2015, pp. 171–172.

[8] Ismael Cuadrado-Cordero, Anne-Cécile Orgerie, Jean-Marc Menaud, "Comparative experimental analysis of the quality-of-service and energy-efficiency of VMs and containers' consolidation for cloud applications", Software Telecommunications and Computer Networks (SoftCOM) 2017 25th International Conference on, pp. 1-6, 2017.

[9] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, and H. C. Gall. "An Empirical Analysis of the Docker Container Ecosystem on GitHub", Proceedings of the 14th Working Conference on Mining Software Repositories (MSR '17), IEEE Press, pp 323-333, 2017.

[10] A. Sill, "The Design and Architecture of Microservices," in *IEEE Cloud Computing*, vol. 3, no. 5, pp. 76-80, Sept.-Oct. 2016.

[11] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar and M. Steinder, "Performance Evaluation of Microservices Architectures Using Containers," *2015 IEEE 14th International Symposium on Network Computing and Applications*, Cambridge, MA, 2015, pp. 27-34.

[12] Chae, M., H. Lee, and K. Lee, A performance comparison of linux containers and virtual machines using Docker and KVM. Cluster Computing, 2017: p. 1-11.

[13] Weins, K., Cloud Computing Trends: 2018 State of the Cloud Survey. 2018.

[14] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81-84, Sept. 2014.