ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 6, Issue 4, pp: (33-36), Month: October - December 2018, Available at: www.researchpublish.com

The Next Era in Virtualization

Ibraheem Al-Sheikh

Dhahran, Saudi Arabia

Abstract: **This article introduces the new virtualization concept in deploying software and application on a virtual machine (VM). The application and all its requirements would be coupled in a containerized form, also known as containers. This new concept is very efficient in optimizing the hardware resources and has several new features that supersede the current deployment methods on VMs. In this paper we will explain in detail the container methodology and compare it with VMs.**

Keywords: **Containers, Virtualization, Virtual Machine (VM), Docker, Kubernetes, Computing.**

I. INTRODUCTION

Virtualization is defined as a method to create a virtual instance of an actual mimic. The virtualization technology was first introduced in the late 1960s by the IBM mainframe computer systems, and was named Time-sharing. Computers and servers were extremely expensive at that time, so IBM introduced the time-sharing concept to have computing resources (CPU, RAM, Disk, Network, etc.) shared with a larger number of users to efficiently use these resources and to make it more economical for users to acquire computing resources for the development of applications (Margaret Rouse, 2014).

Since then, Information Technology (IT) has dramatically spread throughout the world shaping our everyday activity. Virtualization continues to revolutionize technology and to reform application deployments to be agile enough to cope with never-ending demands.

In the early 2000’s, a new concept emerged from virtualization called containerization or simply containers. Application containerization is an Operating System level (OS-level) virtualization method used to deploy and run distributed applications without launching an entire virtual machine (VM) for each app. Several isolated services or applications access the same OS kernel while running on a single host.

II. CONTAINER IN VIRTUALIZATION

Containers can package applications in an abstract form, from the environment they run in; This method of decoupling containers allows easy deployment, regardless of the environment (public cloud, or one’s personal computer). Technically, a container holds all the dependencies required to run a specific application including the dynamic-link libraries (DLL's), configuration files, and compatible versions that will be stored as an image. Images can be stored in an image database and can be redeployed in the future when needed. A container can be understood as an OS-level isolation at the individual kernel subsystem level (filesystem, process tables, etc.).

In shipping & transport industry, belongings can be safeguarded in a container that will safely travel around the globe till it reaches its final destination, and would be received in the exact format in which they were put inside that container. In a similar perspective, applications can be exported from a test & development environment to production or disaster recovery (DR), without impacting the application functionality. Since all the application dependency is containerized in a container, the behaviour of an application can be predicted whenever it is exported to another environment.

Developers would be able to predict how their application would function in an environment, because the containers isolate the application from the environment. Since the Containers couple the application dependencies to run the application (versioning, libraries), it allows the application to be consistent in any environment deployed. As a result, the IT Operation (IT Ops) team would spend less time in diagnosing the differences between each environment, and focus on operation duties.

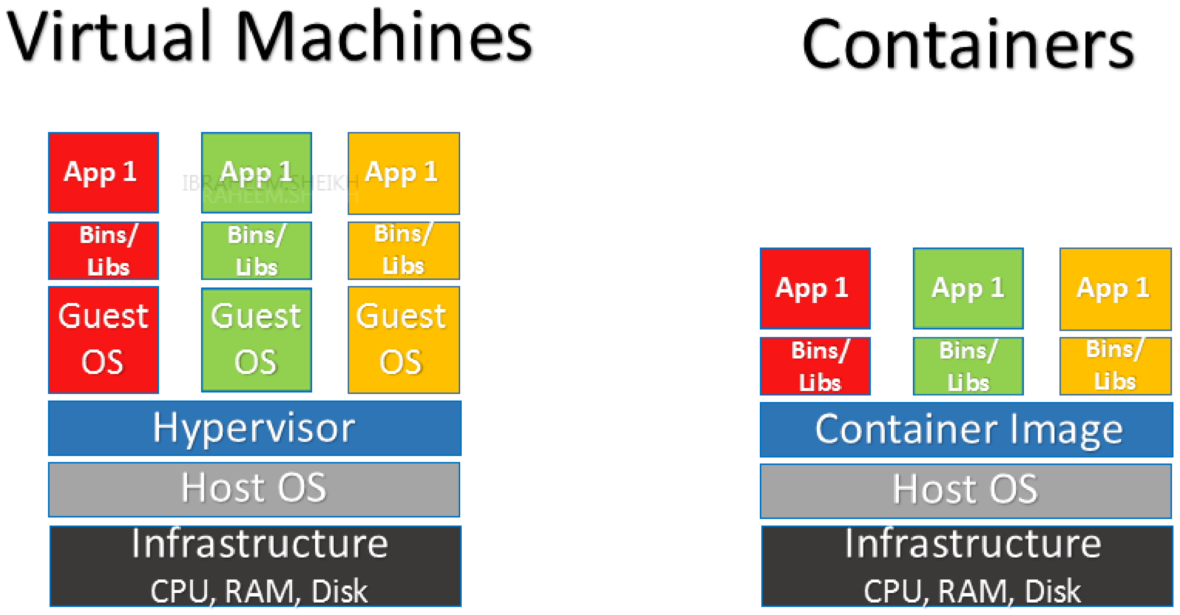
III. CONTAINER ADVANTAGES

Containers supersede VMs in many areas. Containers are lightweight and may take few tens of megabytes (MB's) in size, whereas a VMs may take several gigabytes (GBs) in size as each VM holds an entire Operating System (OS). Thus, a single server can host far more containers than it could with VMs.

Another major benefit is that VMs may take several minutes to boot up their OS, while containers can be started almost instantly (in milliseconds). That allows containers to be created instantly and destroyed instantly, if required.

The concept of Microservices is yet another major benefit of containers. Rather than run an entire complex application in one container, the application can be decoupled and split in to multiple modules (such as the database, application front end, and so on). This makes the development of an application simpler to manage. Developers could upgrade each module separately without affecting the entire service/application (Sahiti 2018, July 25).

The building-block illustration below explains the structural difference between a container and a VM (Pete Brey):



Containers share a single OS, so IT operation management is drastically reduced as compared to VM operations. The operation support staff will be less engaged in the monthly routine patching, bugs, and issues related to the environment.

Container images are stored in a placeholder container registry so that images can be pushed and pulled when needed. And to make things more interesting, one can import or export an image that contains certain features and performs various services from the web, or known providers instead of developing that image himself. To secure these transactions, each image will have a unique signature from the developer. Containers have also introduced a Notary service that adds a signature to an image; this is to verify the authenticity of an image before being pulled to an environment. For example, if one trusts certain vendors or developing agencies, he will only pull images from them and the notary service will authenticate the origin of these images.

IV. CONTAINERS SECURITY

Ever since the advent of containers in the IT industry, the security measures have improved drastically, along with the introduction of interesting methodologies to cope with everyday challenges. As each container holds a small module of one's application or service (microservices), it is easy to isolate a bug or malware within that service without interrupting the entire services. The malware scanner can run a scan on the image registry for vulnerabilities before deploying these images to the environment. Many are taking advantage of the quick deployment of containers to destroy and redeploy a container (with new IP addresses) in a known good state every few seconds to minimize the risk of hacking. There are several security software applications that profile a container-expected behaviour, white-list processes, and check networking activities along with storage practices for any suspicious acts.

Despite the improved security measures, container security is under criticism for its underdevelopment. Many cybersecurity professionals state that there is a lack of mature solutions available for container security. This is plausible, as container security is currently dominated by start-up's and point tools (i.e., Neuvector, Aqua Security, CloudPassage, Cavirin, Layered Insight, StackRox, Aporeto, Twistlock, etc.). Furthermore, portability makes containers more vulnerable to "in motion" compromises. Various security professionals say it's challenging to monitor transient containers and microservices as they appear and disappear. Similar to server virtualization and public cloud workloads of the past, containers remain unfamiliar to many security professionals today. With the growing production of containers and their prolific future deployment plans, it is imminent for security professionals to acquaint themselves with this new technology.

V. CONTAINER USAGE

The concept of the fundamental building blocks of containers is being standardized globally; thus allowing container deployment and importing/exporting of microservices between developers easier. Containers are not meant to replace VMs, but instead complement VMs. The question that arises is when should containers be used in one's environment? Containers can be used to run multiple copies of the same application (i.e., MySQL). For example, if you need to deploy the maximum number of applications in a single hardware, you can utilize containers due to their lightweight characteristics, and their ability to run more instances than Vms. Containers could also be used for efficient mobility and Web applications since containers require significantly fewer resources compared to VMs. Containers can be used for all the Microservices-based applications, or for a single service when deployed at a scale (Preethi Kasireddy 2016, March 4).

In Business Continuity, containers work perfectly for Disaster Recovery (DR) sites. One may create a container on a laptop, run it on a development server in a corporate datacentre, and finally export it to the cloud. Portability is a key feature of containers. It assures application functionality in a different environment, testing the OS version, SSL Library and protocols. Furthermore, it allows network topology, security policies, and storage to one's deployed services in a different environment as well. In the past years, several companies have invested in containers and have made developments to enable others to take full advantage of this emerging technology.

VI. CONTAINER MANAGEMENT

Docker is one of the pioneers in containerization platforms. Dockers was designed by Solomon Hykes on March 13, 2013, and offers automated scaling features (scaling up or down) depending on the usage without the need for human intervention. In a typical use case example, a container runs a web server and web application, while a second container runs a database server that is used by the web application. Containers bundle their own libraries, tools and configuration files, while remaining isolated from each other. Communication between containers with each other are done through well-defined channels (Docker n.d.).

Kubernetes, another pioneer in containerization platforms, is an open source system for container management and automating application deployment (Wikipedia Contributors 2018, June 28). It was originally designed by Google on June 7, 2014, and is maintained by the Cloud Native Computing Foundation. Large part of the Kubernetes API is provided with this extensibility, which is used in containers and extensions as well as by internal components that run on Kubernetes. In Kubernetes, pod is the basic scheduling unit. By grouping containerized components, pod adds a higher level of abstraction. The structure of the pod consists of a single or more than one containers which can share resources and are co-located on the host machine. Unique IP address is assigned to each pod in Kubernetes within the cluster, enabling applications to use ports without any conflict risk. Kubernetes helps Google to run billions of containers a week easily, and in simple words, everything that runs in Google is through containers (Wikipedia Contributors 2018, August 10).

VII. CONCLUSION

There have been conflicting opinions about whether containers will replace VMs. Although this seems unlikely to happen, the best option for performance optimization is to load multiple containers on different VMs. Containerization itself is a rapidly growing technology. According to experts, 40% of the application deployments and services will be running on containers by 2020. The versatile application and diverse features of containers is attracting more users every day (Pete Brey 2018, March 16).

REFERENCES

[1] Pete Brey. (2018, March 16). Containers vs. Virtual Machines (VMs): What's the Difference? Retrieved from https://blog.netapp.com/blogs/containers-vs-vms/.

[2] Sahiti. (2018, July 25). Microservices Tutorial. Retrieved from https://www.edureka.co/blog/-microservices-tutorial-with-example.

[3] Docker. (n.d.). What is a Container. Retrieved from https://www.docker.com/resources/what-container.

[4] Margaret Rouse. (2014). container (containerization or container-based virtualization). Retrieved from https:// searchitoperations.techtarget.com/definition/container-containerization-or-container-based-virtualization.

[5] Goole Cloud. (n.d.). Containers at Google. Retrieved from https://cloud.google.com/-containers/.

[6] Wikipedia Contributors. (2018, August 10). Operating-system-level virtualization. Retrieved from https://en. wikipedia.org/wiki/Operating-system-level_virtualization.

[7] Preethi Kasireddy. (2016, March 4). A Beginner-Friendly Introduction to Containers, VMs and Docker. Retrieved fromhttps://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b.

[8] Wikipedia Contributors. (2018, June 28). Kubernetes. Retrieved from https://en.wikipedia.-org/wiki/Kubernetes.