DOI: http://dx.doi.org/10.26483/ijarcs.v8i8.4618

ISSN No. 0976-5697

Volume 8, No. 8, September-October 2017

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

THE ROAD TO DOCKER: A SURVEY

Gaurav Bhatia
Department of Computer Science & Engineering
Sardar Patel University of Police, Security & Criminal Justice
Jodhpur, India

Arjun Choudhary
Department of Computer Science & Engineering
Sardar Patel University of Police, Security & Criminal Justice
Jodhpur, India

Vipin Gupta
U-Net Solutions
Moga, India

Abstract: The interest on conventional technologies is declining with the increasing demand on new technologies. In the virtualization industry, container based technology has become the most powerful technologies in the last couple of years. With the arrival of Docker, implementation of containerization technology has become more simplified and efficient. Unlike other virtualization platform, Docker is an open source software container platform that provides some facilities, which are useful for developers and administrators. Docker has the traits of providing fast and lightweight virtualization on operating system-level, because of which Docker has become popular technology to serve variety of cloud platforms. Development cost and time can be brought down tremendously by simply replacing traditional on-going virtual machine with Docker container. Also with the use of Docker, the cost of re-developing the cloud platform can be reduced to great extent.

Keywords: Docker; Docker Container; Virtual Machine; Virtualization; Hypervisors; Operating System

I. INTRODUCTION

Docker is an open standard platform for building, exporting, and running applications. It can also be viewed as command-line program with a background daemon running and a set of remote utilities that take a systematic approach to solve common software problems and simplifying user experience of installing, running, publishing, and removing applications. Docker provides facility to separate your applications from your infrastructure to deliver software quickly. A Docker container is a software bucket comprising of all the supporting dependencies necessary to run the software independently. Also the isolation and security feature in Docker allow you to run multiple containers simultaneously on a single host machine. [1]

This paper provides insight view on technology of Docker, and we will try to examine how Docker has changed the dynamics of whole virtualization industry. The insight of the paper is systemized as follows. In section 2, we will see what Docker actually is. In section 3, we will focus upon Docker architecture and its components. In section 4, we will compare light weight virtualization using Docker and heavy weight virtualization. Section 5 and 6 will give you a small idea of why to use or not to use Docker container. Finally, in section 7 we will try to compare virtualization with containers following with conclusion and future scope of Docker technology in section 8.

II. DOCKER CONCEPT

Docker is a platform designed to make it easier to create, deploy, and run virtualized application containers on a common operating system (OS), with an ecosystem of allied tools. Initially, Docker was created to work on the Linux platform, but has extended to offer greater support for non-Linux operating platforms, including Mac OSX and Microsoft Windows. Versions of Docker for Amazon Web Services (AWS) and Microsoft Azure are also released. [2]

Before Docker, deploying software to different environments required huge efforts. Even though these efforts were encapsulated in virtual machines, a lot of time was spent in managing and deployment of these machines, waiting for them to install and boot, and managing the overhead of resource use they created.

With Docker, image can be pulled down from public or private repositories and is ready to run, consuming fewer resources and contained so that it doesn't interfere with other environments.

User don't need to worry about whether his container will going to be shipped to an Ubuntu machine, a CentOS machine, or any other machine; as long as it has Docker installed on it. [3]

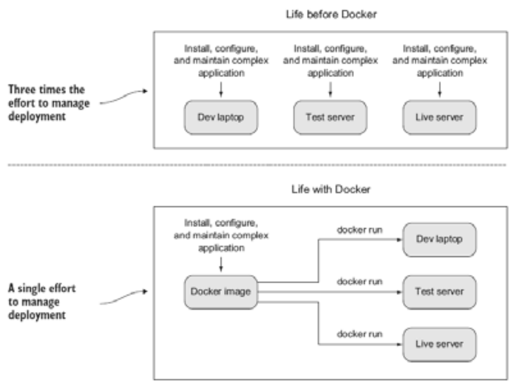
Life before Docker
Three times the effort to manage deployment
Install, configure, and maintain complex application
Dev laptop
Test server
Live server
Life with Docker
A single effort to manage deployment
Install, configure, and maintain complex application
Docker image
docker run
Dev laptop
Test server
Live server

Figure 1. Software delivery before and after Docker [3]

III. Docker Architeture

Docker can be thought of as application based on client server architecture, as shown in Figure 2. Docker on your host machine is divided into two parts—a Server Docker Daemon with a REST API and a Client Docker CLI that talks to the daemon. A REST API is one that uses standard HTTP request types such as GET, POST, and DELETE through which programs can communicate with daemon.

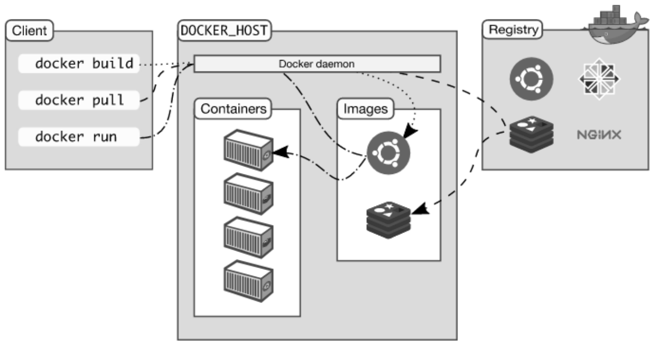

Figure 2. Docker Architecture [1]

The Docker daemon is the core component of Docker architecture which receives instructions from Docker client regarding building, running and shipping the Docker containers. Docker provides facility to run the Docker client and the Docker daemon on same device, or client can connected to the Docker daemon running remotely. Communication between Docker client and Docker daemon is done using a REST API, over a network interface.

A. *The Docker Daemon*

The Docker daemon (see figure 3) is the hub of your interactions with Docker, which waits for Docker API requests and manages the state of your Docker objects accordingly.

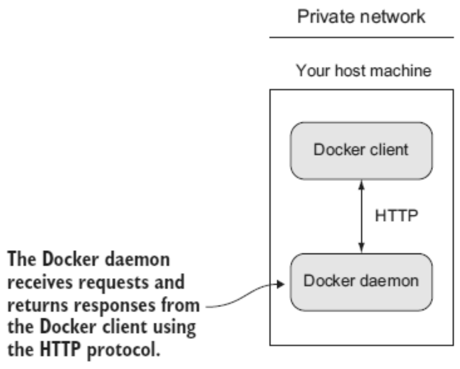
Private network
Your host machine
Docker client
HTTP
Docker daemon
The Docker daemon receives requests and returns responses from the Docker client using the HTTP protocol.

Figure 3. The Docker Daemon [3]

B. *The Docker Client*

The Docker client (see figure 4) is the simplest component in the Docker architecture, which helps Docker users to communicate with Docker. The Docker client is called when user type commands like docker run or docker pull on his machine. Its job is to communicate with the Docker daemon by sending HTTP requests.

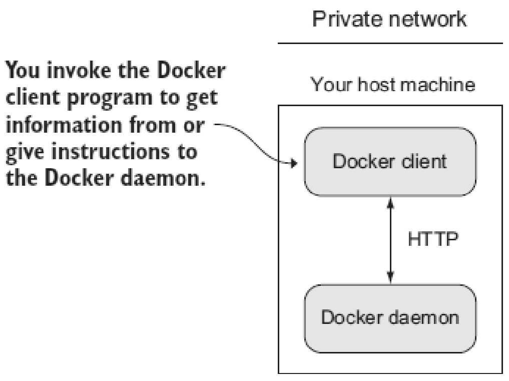
Private network
Your host machine
You invoke the Docker client program to get information from or give instructions to the Docker daemon.
Docker client
HTTP
Docker daemon

Figure 4. The Docker Client [3]

C. *Docker Registries*

Suppose user has created his own images, and he wants to share them with other users. This is where the concept of the Docker registry comes in. Docker registries provide a platform to store the Docker images. Docker Hub and Docker Cloud are two public registries that can be used by anyone to pull or push images, and by default Docker is designed to search for images on Docker Hub. Many companies set up private registries to store and share their proprietary images internally. You can pull any image using docker pull or docker run command. To push image to your configured registry, use docker push command.

D. *Docker Images*

In Docker, everything is based on 'Images'. Images act as a read-only template for creating new containers. The platform for creating a new image is a base image. For example, your base image can be Ubuntu 16.04 LTS with an Apache web server and you web application installed on it. This approach of creating a new image is called "committing a change". User can create and run his own Docker image by defining steps in a Dockerfile. Each instruction written in a Dockerfile creates an additional layer in the image, as depicted in Figure 5. When user makes a change in Dockerfile and tries to recreate the Docker image, only those layers are recreated in which changes have been made by that particular user. This is what makes images so lightweight, small, fast and reliable, when compared to other pre-existing virtualization technologies.

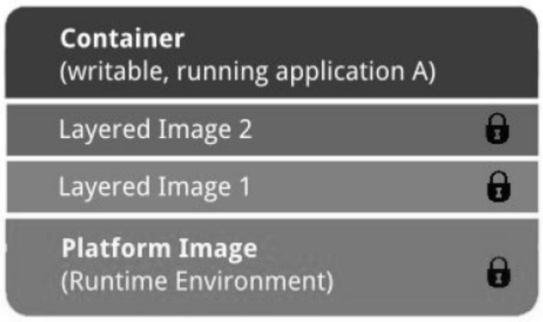

Figure 5. Image created using Dockerfile

E. *Docker Containers*

At first glance, we can say that Docker container appears to be a lightweight form of virtual machines. Docker container is created using Docker images or we can say container is a runnable instance of a Docker image. Containers hold the complete set of dependencies required by an application to run in more confined way. Using the Docker API or CLI, you can easily create, run, move, stop or remove a container.

Docker containers will always be in one of the four states according to the diagram in Figure 6. [4]

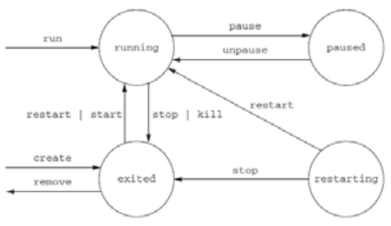

Figure 6. The state translation diagram for Docker container [4]

IV. DOCKER V/S VIRTUAL MACHINE

Before we compare Docker with Virtual Machine, let us first see what does the term heavy weight virtualization and light weight virtualization means.

Heavy weight virtualization means each and every virtual machine having its own independent operating system. With the help of virtualization software, you can install any guest operating system such as Windows, Linux and Mac on your host operating system. Virtualization software such as VMware Workstation, KVM, Oracle Virtual Box, and Hyper-V comes under Type-2 Virtualization, which means we cannot directly ran these software's on the system hardware. So the new type of virtualization technology called hypervisor came into the existence which can be installed directly on system hardware. It comes under Type-1 Virtualization. [5]

But the problem with these software's are that we need to install whole Guest OS also for doing small experiments like checking network connectivity between two virtual machines using simple ping command, which is sheer wastage of resources.

Nowadays, the focus is moving towards lightweight container based virtualization technology known as Docker as shown in Figure 7. With Docker, application can be placed inside the Docker container, providing Container as a Service (CAAS) platform. These containers containing application are easy to handle and can be placed on any type of platform including cloud. Also these types of light weight technology are fast and consume almost negligible resources.

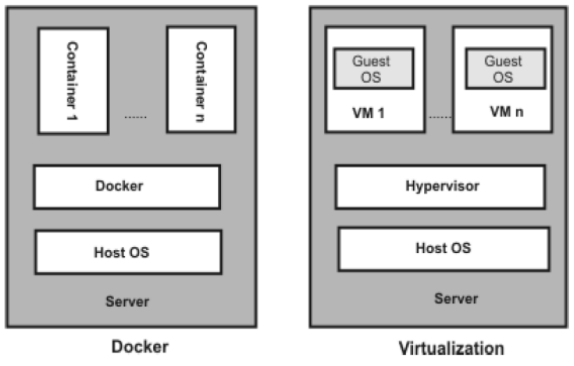

Figure 7. Docker v/s Virtualization [5]

In early stages, if we try to compare Docker containers to virtual machines we will see that Docker containers have ability to share single kernel and application library. Also Docker container possess lower system overhead(i.e. computation time, memory etc.), due to which the performance of the application running inside a container is generally better as compared to application running within a virtual machine.

Capability of virtual machines can be enhanced using technologies like Intel's VT-x/EPT and VT-d which is already provided by VMware. An Intel VT-x/EPT and VT-d technology provides same processing power to your guest operating system as your host operating system gets and also increases network performance. Also by using these technologies, virtual machine can achieve ring-1 hardware isolation [6]. Ring works as a protected shield to protect sensitive data and functionality from errors and abnormal behavior by improving fault tolerance and computer security [7].

According to Gupta V (2017), creation and launching time of Docker container is much lesser than that of virtual machine. [5] Also containers consume resources according to the need of particular service or an application. Because of which it is possible to execute more containers on system having not that powerful configuration.

V. ADVANTAGES OF DOCKER CONTAINERS

As we know, Docker has been dominating the DevOps conversation since its arrival in 2013. The main reason for its dominance is the benefits provided by Docker container. In this paper, we seek to highlight some of its best features.

A. *Free and Open Source*

Generally we all use virtualization software's like VMware etc. to create the virtual machines, but most of these software's require commercial licenses for enterprise use. In contrast, Docker is completely free and open source which can be easily downloaded and installed. Also no one will ask you buy license in order to use you application. [8]

B. *Ultra Consistent*

While creating virtual machine, sufficient amount of computer resources are required and if resources provided are insufficient virtual machine will not work properly. Whereas in case of Docker, resource consumption is very less because of which higher level of consistency can be achieved. [8]

C. *Rapid Deployment*

In the past, deployment of new hardware used to take few days. But with the arrival of virtual machines, timeframe was reduced down to minutes. Now the appearance of Docker has reduced deployment to mere seconds. Also the cost of creating and destroying the containers is negligible. [9]

D. *Portability*

Applications created inside the Docker containers are portable and light weighted. These portable applications can be treated as a single unit hence can be moved easily, without affecting the performance of the containers. [10]

E. *Isolation*

With Docker, every container created has its own set of resources and each container runs independently without any interfere to other running containers. Docker also ensures easy creation and deletion of your application since each application runs within its own container. Deleting containers won't leave any configuration files on your system. Docker not only also ensures easy creation and deletion of an application but also ensures that each application only uses resources that have been assigned to them. [11]

F. Security

Using Docker is secure because applications and services that are running inside the containers are completely independent from each other. Which means one container cannot poke into another container. Also as discussed earlier, there is no sharing of resources between containers, and each container has its own set of required resources. That means if something goes wrong with one container, the data available in that container will be affected without affecting the rest of the containers. [11]

VI. DISADVANTAGES OF DOCKER CONTAINERS

Before migrating to Docker, there are some facts or drawbacks which should be kept in mind.

A. Persistent Data Storage is Complicated

In docker, when we shut down a container then all of the data residing inside that container gets disappear forever, unless you save it somewhere else first. Although, Docker provide ways to save your data persistently, such as Docker Data Volumes, but using it seamlessly is still very challenging task.

B. Compatibility with Older Machines

Currently, Docker is compatible only with 64-bit local machines. It does not run on 32-bit machines.

C. Containers don't run at Bare Metal Speed

There is no doubt that Docker containers are capable of consuming resources more efficiently than virtual machines. But performance of containers is not up to that level due to overlay networking.

D. Platform Dependant

Docker was initially designed to run on Linux machine. However, nowadays Docker has started supporting Windows, Mac OS X and many other platforms. To make Docker platform independent we will need additional layer between host operating system and Docker. [12]

VII. VIRTUALIZATION V/S CONTAINERS

It totally depends on user needs whether he want to go with virtualization technology or with containerization technology. A container environment often provides rapid deployment, greater efficiency and better resource consumption. If you are looking for better isolation then system virtualization may be more relevant option. But with Docker things are changing rapidly.

Gupta V (2017) in Table 1 has compared heavy weight virtualization i.e. Virtual Box against light weight virtualization i.e. Docker on factors like size, memory, storage, installation time and boot time. [5]

Table I. Heavy Weight Virtualization v/s Light Weight Virtualization

Heavy Weight Virtualization (Virtual Box)		Light Weight Virtualization (Docker)	
Iso Size (Ubuntu 16.04)	667 MB	Image Size (Ubuntu 16.04)	130 MB
RAM	1 GB	RAM	568 KB
Storage	10 GB	Storage	104 KB
VM Installation Time	21 Minutes	Container Creation Time	0.9 Seconds
Virtual Box Software Size	108 MB	Docker Engine Size	19.4 MB
Boot Time	35 Seconds	Boot Time	< 1 Seconds

According to JC Wang (2015) in Table 2, Container starts within a few seconds with faster speed and capable of loading more services by using less space. [13]

Table II. Virtualization v/s Containers

	Virtualization (i.e. KVM, Xen)	Containers (i.e. LXC, Docker)
Typical Server Deployment	10 – 100 VMs	100 - 1000 Containers
Boot Time	Less than a Minute	Seconds
Physical Resources	Each VM has resource reserved for its own use	Shared by all containers

Dua et al. (2014) in Table 3 compares Virtual Machines and Containers on multiple factors like performance, isolation security, networking, storage. [14]

Table III. Virtual Machine v/s Containers

Parameter	Virtual Machines	Containers
Guest OS	Each VM runs on virtual hardware and Kernel is loaded into in its own memory region.	All the guests share same OS and Kernel. Kernel image is loaded into the physical memory.
Communication	Will be through Ethernet Devices.	Standard IPC mechanisms like Signals, pipes, sockets etc.
Security	Depends on the implementation of Hypervisor.	Mandatory access control can be leveraged.
Performance	Virtual Machines suffer from a small overhead as the machine instructions are translated from Guest to Host OS.	Containers provide near native performance as compared to the underlying Host OS.
Isolation	Sharing libraries, files etc. between guests and between guests hosts not possible.	Subdirectories can be transparently mounted and can be shared.
Start Up Time	VMs take a few minutes to boot up.	Containers take lower amount of storage as the base OS is shared.
Storage	VMs take much more storage as the whole OS kernel and its associated programs have to be installed and run.	Containers take lower amount of storage as the base OS is shared.

VIII. CONCLUSION AND FUTURE SCOPE

Microservices architecture is a core concept around which Docker is built. This concept has reduced the building and deployment time of Docker containers. Also the demand for resources by a Docker container is very low, and it can achieve higher performance. Docker provides some fine features, which guarantee simplified usability and scalability. However, the Docker container still has some issues regarding security, but tools can be applied to strengthen the base of the container. The main purpose of writing this paper was to give users a brief overview about all the important components of Docker and how Docker technology is different from pre-existing technologies. In future, we will try to look into the other features of Docker like Docker Swarm; a technology for managing a cluster of Docker Engines [15] and NFS Storage; a technology to store and update files on remote computers [16]. We believe, with time Docker technology will get matured and will be deployed more widely.

IX. REFERENCES

[1] "Docker Overview." Internet: https://docs.docker.com/engine/docker-overview/.

[2] Riyaz Faiullabhoy. "Docker Versions." Internet: https://blog.docker.com/2017/06/docker-for-aws- azure-security/, June 2, 2017.

[3] Miell, Ian, and Aidan Hobson Sayers. *Docker in Practice*. Manning Publications Co., 2016.

[4] Nickoloff, J. *Docker in Action*, Manning Publication co, 1st edition, Shelter Island, New York, 2016, page 28.

[5] Gupta V, Kaur K, Kaur S. Performance comparison between light weight virtualization using docker and heavy weight virtualization. International Journal of Advanced Technology in Engineering and Science, Volume No.05, Issue No. 03, March 2017, 509-514.

[6] Sudhi Seshachala. "VT-x and VT-d technologies." Internet: https://devops.com/docker-vs-vms/, November 24, 2014.

[7] Mohamed Fawzi. "Concept of Rings." Internet: https://fawzi.wordpress.com/2009/05/24/virtualization and-protection-rings-welcome-to-ring-1-part-i/, May 25, 2009.

[8] Chris Tozzi. "Open Source and Consistency." Internet: http://www.theserverside.com/feature/The-benefits-of-container-development-with-Docker.

[9] Andrei Manea. "Rapid Deployment." Internet: https://www.cloudhero.io/single-post/top-three-benefits using-docker, March 10, 2017.

[10] Vase, T. (2015). Advantages of Docker.

[11] Ofir Nachmani. "Isolation and Security." Internet: https://dzone.com/articles/5-key-benefits-docker-ci, April 29, 2015.

[12] Chris Tozzi. "Platform Dependent." Internet: https://sweetcode.io/3-pros-3-cons-working-docker-containers/

[13] J. C. Wang, W. F. Cheng, H. C. Chen and H. L. Chien, "Benefit of construct information security environment based on lightweight virtualization technology," 2015 International Carnahan Conference on Security Technology (ICCST), Taipei, 2015, pp. 1-4.

[14] R. Dua, A. R. Raja and D. Kakadia, "Virtualization vs Containerization to Support PaaS," 2014 IEEE International Conference on Cloud Engineering, Boston, MA, 2014, pp. 610-614.

[15] "Docker Swarm." Internet: https://docs.docker.com/engine/swarm/

[16] "NFS Storage." Internet: http://searchenterprisedesktop.techtarget.com/definition/Network-File-System