Reorganizing Virtual Machines as Docker Containers for Efficient Data Centres

N. VasanthaKumari and R. Arulmurugan

Abstract With increase in the use of virtual machines in various fields, there is a need to enhance the balancing of huge workload in data centres. The traditional way of implementing the cloud is usually done with virtual machines. Working of virtualization can be reconsidered with different and enhanced technology, such as containers in data centre. Docker containers are gaining popularity due to its features such as increase in productivity by reducing the number of resources. In this chapter, virtual machines are compared with the containers, and virtualization techniques are examined so that the user can work with respect to the requirements. Unlike a virtual machine, a container does not have another software layer called Hypervisor. Due to these reasons, containerized applications have better performance characteristics than virtual machine-based applications. We will discuss the benefits of containers over hypervisor in containers. Nevertheless, containers execute directly in the kernel of the virtual machine. Docker containers use engine of the docker as an alternative to hypervisor.

Keywords Container · Docker · Virtual machine

1 Introduction

Subsequent paragraphs, however, are indented. Many establishments have already moved/to move to cloud computing services with the growth of Cloud technologies. Cloud computing also permits to share different resources (hardware/software) over

N. VasanthaKumari (✉)
Presidency College, Bangalore, Karnataka, India

R. Arulmurugan
Presidency University, Bangalore, Karnataka, India
e-mail: arulmurugan@presidencyuniversity.in

A. Haldorai et al. (eds.), *3rd EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing*, EAI/Springer Innovations in Communication and Computing, https://doi.org/10.1007/978-3-030-78750-9_14

the network [1]. Traditional method of implementing cloud is by the usage of virtual machines. In recent days, Dockers have got more prominence than virtual machines due to its characteristics. Virtual machines and Dockers have its own importance and features. Dockers are the newest technology in cloud IT, and virtual machines have been used for many years and play an important role in data centre with variety of sizes [2]. Virtualization works where several applications are hosted on a single server, which also has mapping techniques.

Virtual machine (VM) is a computer that is present in other host OS and which provides variety of services to users. With the use of virtual machines, customers can get various services from the providers of Cloud. There are many devices of the system which can also be virtualized. When working on virtual machines, the user works with the assumption of working on physical machines [3]. Virtual machines are fabricated as a layer in the infrastructure by executing programs. Hypervisor is a primary part which is sandwiched between the hardware and OS for running the virtual machines. There are many types of virtualization (at server level/network level, etc.) executed at various levels.

Docker containers also have the same conception of virtual machine except in terms of time and code. Images of the containers become containers only when they run on Docker engine. With only the transformation in infrastructure, containers are providing services to both windows and Linux-based applications.

This chapter is divided into different components/sections. This chapter mainly focuses on comparison of virtual machines and docker containers with respect to benefits, features, etc. Section 2 overviews on limitations of virtual machines. Section 3 on docker containers, benefits, and its different models. Section 4 explains about differences between virtual machines and Dockers followed by conclusion and future work.

2 Background

In this section, we will discuss the parameters that distinguish virtual machines from containers and associate the infrastructure of both and describes why these two technologies is becoming increasingly familiar [4]. Containers and virtual machines jointly have become an important factor in technology, which also ease to deploy the applications.

2.1 Virtual Machines and Containers

Both virtual machines and containers have the same way of segregation of resources and allocation. Virtual machines act similar to physical computers. There is much dissimilarity between a virtual machine and container, but the main difference is

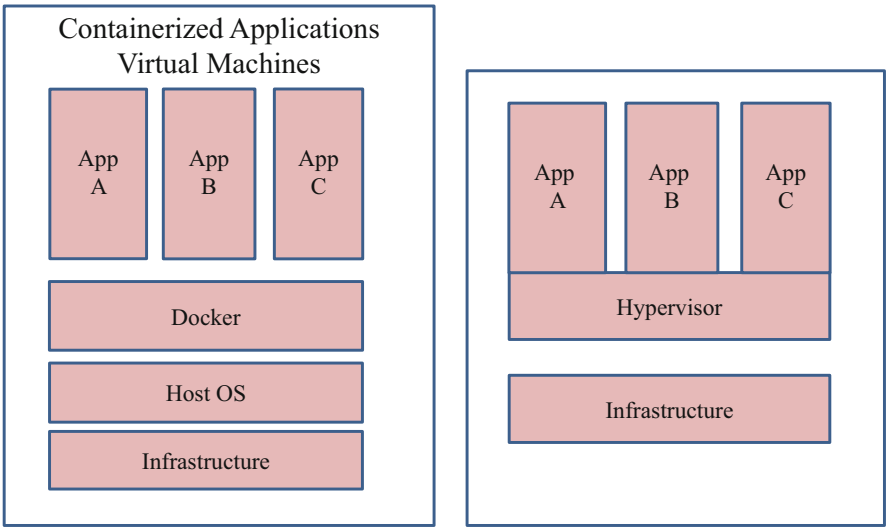


Fig. 1 Container and virtual machines

that containers offer a method to virtualize an OS so that several workloads can execute on one machine. Containers execute with less memory space than virtual machines. Both tools have an impression that single host's task is executed on multiple machines. Both tools are compared with few components like [5] shown in Fig. 1.

Planning of the host machine: In case of containers, kernel is shared between the hosts where the virtual machine has the advantage to execute the kernel which is unlike the machine kernel.

Machine startup: Containers use less resources and the system will start promptly in few seconds. Machine will start in operating system customarily and speed depends on the applications.

Compactness: In virtual machine, there is an issue where applications running on a single host cannot be ported to another host machine, which is allowed in containers. Containers are available in suites that can be used to execute several applications. By nature, containers are lightweight compared to virtual machine. As the containers are lightweight, applications can be easily deployed on server host machines.

Assessment of performance: Virtual machines and containers have its own prominence; hence comparison will not be much successful. Performance is calculated based on many criteria such as speed, cost. Containers use fewer resources and hence it is lightweight. Making a copy is easy in containers compared to virtual machines.

2.2 *Application Deployment in Virtual Machines and Container-Based Environment*

Containerized applications are easier to deploy and scale than virtual machine-based applications. In most environments, deployment of applications is performed using a process called Continuous Integration and Continuous Deployment. It refers to automation of software pipeline to integrate code from the various application developers and deliver into production. Cloud applications in large-scale industries need deployment with a good performance with that demand; deployment can be done with either virtual machines or dockers. As the applications grow, it is important to manage the resources and deploy manually. Hence, methods called continuous integration and continuous deployment are used to deploy automatically whenever the changes happen in the code. In Continuous Integration(CI), a memory is shared for storing the code written by the programmers and when programmer tries to loads the code a snippet from the memory storage activates and does integration of both and executes the test cases. In the market, there are few tools available for performing continuous integration like Jenkins, Bamboo, etc. Git is one of the tools available where developers write the code and merge with the master code [6].

Continuous Delivery (CD) is another tool used to deploy the errors that was fixed, new structures into the server whenever it is needed. Users can do different types of testing that is beyond other types of testing such as integration testing. With these tools, deployment is common which does not have human intrusion. CI/CD tools automates the building of new code into server [7].

3 Related Work

With innovation in research, there are a lot of benefits of containers over virtual machines in terms of security and performance.

Docker containers provide solutions to the concerns with much research by providing lightweight images to ease installation of software [8]. explored that the Docker has the capability to execute multiple containers like Genomic pipelines. According to [9], there are many issues with virtual machines like scheduling, managing the resources which has been determined using containers. In comparing the performance between virtual machines and docker containers, [10] explored that for all the experimentations docker containers indicate request, consume lower execution times compared to virtual machines.

In many researches, more comparisons are done with virtual machines and containers. Virtual machines are delineated to physical machines. It has been considered as a challenge to the researchers to design an effective placement algorithm [11]. The research paper [12] says that dockers can be used in wider sense to reduce traffic in load balancing environments to increase the performance

of the data centre. The paper [13] compared Linux containers with virtual machines to improve the performance of systems, which uses the extensions of virtualization.

3.1 Limitations of Virtual Machines

There are many restrictions of using virtual machines. Few of them are [14]:

1. Virtual machines do not provide security as they communicate and share the data; there are chances that the attacker can exploit the system.
2. Overflow of a buffer is common in virtual machines. It has a limited length for the buffer. So, when user attempts to write beyond the limits of the buffer, then that particular condition is called overflow of a buffer.
3. As there is a single hypervisor, then there are chances for failure of the system as that one hypervisor fails then entire system stops working.
4. The main prerequisite for working on virtual machines is the Internet, which basically leads to security issues.

3.2 Docker Containers

Docker is an open-source podium that executes application and makes the program very easy to develop and distribute. Docker provides us an opportunity to deploy the applications into the containers [15]. Docker containers are used for developing and deploying code very fast by reducing the delay, dockers have become a very important tool based on their performance. Users who need the advantage of dockers should have familiarity about Docker Networks, Docker Storage, Docker image, Docker Swarm, etc.

3.2.1 Components of Docker

To understand the architecture of docker, it is required to know about the components used. There are primarily four components inside. They are Docker client and Server, Docker Images and Registries, Docker Containers [16] as shown in Fig. 2.

3.2.2 Docker Client

Docker uses the concepts of client/server architecture in which we know the functionalities of a client and a server. Docker client communicates with the daemon of a docker for constructing or executing of containers. Both communicate using the network interface or APIs. When the command docker run is executed, client directs

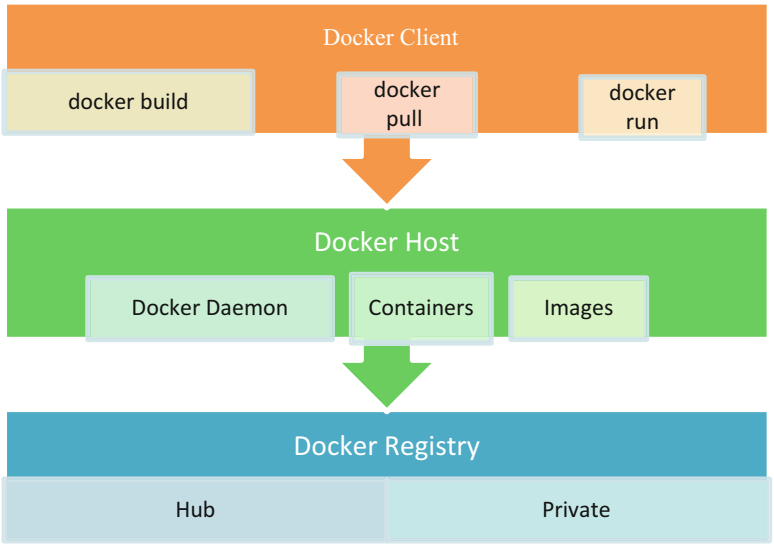


Fig. 2 Architecture of docker container

these commands to dockerd which in turn continues the execution. Docker client can communicate with more than one daemon.

3.2.3 Container

These are the instances of an image. Containers can be created, executed, started or stopped using the API. Containers can be connected to more than one network, store, etc. A container can be explained well by its image and also by the options of arrangement while creating it.

Example: docker run

When this command is executed, docker creates a new network interface to connect to the container. It allows modifying files and directories in the local file system.

3.2.4 Images

There are two ways to build an image. Image is basically a file that is used to run the code in container. When the container executes an image, it turns into instance of that particular container. Images are reusable so that it can be deployed on any machine. Images are comprised of multiple layers. Each layer of docker image can be viewed by a command in Command Line Interface (CLI). There are multiple

commands to process docker image such as docker image build, docker image load, docker image pull, docker image push.

3.2.5 Registries

Docker images are reposited in docker registries. Such images can be used to construct images of other applications. Clients communicate with registry using APIs. Pull command is used to reclaim the image from docker daemon. For instance, IBM cloud has container registry, which is used by various customers, individuals, etc. to dispense the images which comprise operating systems, databases, etc.

3.2.6 Advantages of Docker Containers

Containers execute multiple applications. This has many advantages:

1. **Movability and quickness**: They provide the choice of moving applications between the containers very fast and also easily. An application executed on a host can also be deployed and tested into other host machines.
2. **Performance**: It performs well by executing the task very fast where speed is one of the advantages of the containers. Different levels of execution like deploying; testing will be done faster as containers are very small.
3. **Isolation**: A host contains multiple containers and hence applications that need different versions of software can be executed easily.
4. **Container management**: New containers can be created based on the requirement.
5. **Solidity**: Resources are used by the dockers that are accessible more competently as they will not use hypervisor. Comparatively with the virtual machines, docker containers execute on a single host. Docker containers requires less number of resources than machines which are used in the environment of virtual machines.

3.2.7 Shortcomings of Docker Container

Like other technologies, dockers also have drawbacks. There is a need to understand what the requirement for using dockers in the research is. Containers also have disadvantages. They are listed as follows:

1. Containers make use of resources more efficiently than virtual machines. They do not execute on speeds of bare metal.
2. When the containers are closed, the entire data will be vanished unless work is committed.
3. Dockers do not provision Graphical User Interface (GUI) and hence it is not suitable for graphical-based applications.

4. Docker can be executed only on updated machine versions that are only 64 bit. It will not support older versions.
5. Not all applications will be benefited from containers and also do not provide much security.

3.2.8 Setting Up the Environment for Docker and Execute Container

Initially make sure that secure shell (SSH) in loaded into the virtual machine. Start the docker container, and docker daemon provides the permissions to read/write to the Linux socket to users in a group. Docker group should be created, and current user (vagrant) need to be added to the group. Start running the initial container. The docker run command is executed and a name is assigned to the container. If not, docker daemon will assign a name to the container arbitrarily.

dockerd is a continuous action required to manage containers where different programs are used for daemon and client.

vrant@localhost ~]$ ps -ef | grep docker

root 811 1 0 16:47 ? 00:00:01 /usr/libexec/docker/docker-containerd-current –listen unix:///run/containerd.sock –shim /usr/libexec/docker/docker-containerd-shim-current –start-timeout 2mroot 835 1 0 16:47 ? 00:00:02 /usr/bin/**dockerd-current** –add-runtime oci=/usr/libexec/docker/docker-runc-current –default-runtime=oci –authorization-plugin=rhel-push-plugin –containerd /run/containerd.sock –exec-opt native.cgroupdriver=systemd –userland-proxy-path=/usr/libexec/docker/docker-proxy-current –init-path=/usr/libexec/docker/docker -init-current –seccomp-profile=/etc/docker/seccomp.json –selinux-enabled –log-driver=journald –storage-driver overlay2 –add-registry registry.fedoraproject.org –add-registry registry.access.redhat.com

root 1066 1 0 16:47 ? 00:00:00 /usr/libexec/docker/rhel-push-plugin

vagrant 1115 1089 0 16:55 pts/0 00:00:00 grep –color=auto docker

To check with docker running, following command is used:

[vagrant@localhost ~]$docker –version

ok **version 1.13.1**, build b5e3294/1.13.1

To create docker group and start docker Containers:

vrant@localhost ~]$sudo usermod -aG docker $USER

vrant@localhost ~]$ grep docker /etc/group

To run first docker container:

vrant@localhost ~]$ docker run –name hello hello-world

Hello from Docker!

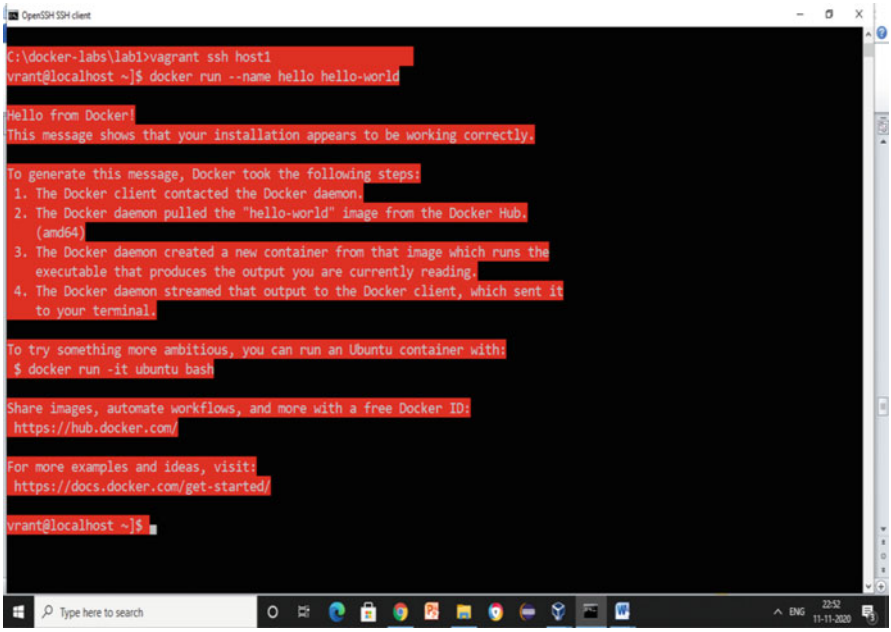
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker Daemon will contact Docker client.
2. “hello-world” image will be pulled from the Docker hub by Docker daemon.

3. A new container is produced by the daemon that executes the image and generates the output.
4. Docker client receives the output from daemon, which in turn is directed to the terminal.

The command is run inside the container. Command to run is specified in the image by the image developer when the image is built.



4 Containers in Cloud Environment

Docker Containers are collection of programs which comprise all the necessary components to execute in any type of environment. Containers anticipates and executes irrespective of the environment. Any machine can have different containers with a single operating system which excludes dependencies. Many algorithms were proposed for container scheduling on physical machines. Many technologies exist for understanding the concept of containers. The most used ones are docker, Linux-Vserver.

5 Virtual Machine Versus Containers

Upon different analysis on benefits and limitations of twin technologies, there exist differences based on criteria given in the following Table 1.

Table 1 Comparison of virtual machine containers

Feature	Virtual machine	Containers
Operating system	Need of operating system serving as guest	OS inside can be common among operating system
Portability	A smaller amount	Can be done more
Right to use resources	There is no use of resources as the crow flies	It is possible
Resources required for execution	Requires more in number	Not much required compared to virtual machine
Number of servers required	Requires more servers	Less in number
Need of memory	Less memory is required	Less memory as host operating system is distributed among various applications
Being away from threat	Provides safety to the data inside the host as there is a hypervisor	As the kernel is shared among the applications, safety cannot be assured
Allocation of library files and others	Distribution is not allowed	It can be distributed using different commands of Linux

6 Future Work and Conclusion

In this chapter, two important technologies are compared based on few parameters like portability, security, etc. Both serve same purpose of virtualization. Containers do not require many resources compared to virtual machines. Choosing one of them is the responsibility of a researcher based on the requirement of the research. Continuous Integration and Continuous Delivery tools automate the deployment by integrating the code. In the future, improvements should be carried out by providing more security while using containers. There are different docker models available for variety of Operating Systems. Docker containers are added asset to the current technologies. They are more comfortable than virtual machines in deployment, testing. In future, still various traits of virtual machines and containers need to be discovered. To conclude, dockers perform faster than virtual machines as they use a smaller number of resources and a guest operating system does not exist. Further, there is a contest for all researchers in energy efficiency in cloud and consolidating services with workloads.

References

1. Ala'Anzy, M., & Othman, M. (2019). Load balancing and server consolidation in cloud computing environments: A meta-study. *IEEE Access, 7*, 141868–141887. https://doi.org/10.1109/access.2019.2944420.
2. Singh, S., & Singh, N. (2016). Containers & docker: Emerging roles & future of cloud technology. In *2016 2nd international conference on applied and theoretical computing and communication technology (iCATccT)*. https://doi.org/10.1109/icatcct.2016.7912109.

3. Zhao, H., Zheng, Q., Zhang, W., Chen, Y., & Huang, Y. (2015, December). Virtual machine placement based on the VM performance models in cloud. In *2015 IEEE 34th international performance computing and communications conference (IPCCC)*. https://doi.org/10.1109/pccc.2015.7410296.
4. Yadav, R. R., Sousa, E. T. G., & Callout, G. R. A. (2018). Performance comparison between virtual machines and Docker containers. *IEEE Latin America Transactions, 16*(8), 2282–2288. https://doi.org/10.1109/tla.2018.8528247.
5. Guan, X., Wan, X., Choi, B.-Y., Song, S., & Zhu, J. (2017). Application oriented dynamic resource allocation for data centers using Docker containers. *IEEE Communications Letters, 21*(3), 504–507. https://doi.org/10.1109/lcomm.2016.2644658.
6. Beloglazov, A., & Buyya, R. (2011). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience, 24*(13), 1397–1420. https://doi.org/10.1002/cpe.1867.
7. Sharma, O., & Saini, H. (2017). SLA and performance efficient heuristics for virtual machines placement in cloud data centers. *International Journal of Grid and High Performance Computing, 9*(3), 17–33. https://doi.org/10.4018/ijghpc.2017070102.
8. Gao, Y., Lin, W., & Zhou, J. (2019). Cost-efficient and quality of experience-aware provisioning of virtual machines for multiplayer cloud gaming in geographically distributed data centers. *IEEE Access, 7*, 142574–142585. https://doi.org/10.1109/access.2019.2944405.
9. Arianyan, E., Taheri, H., & Sharifian, S. (2015). Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Computers and Electrical Engineering, 47*, 222–240. https://doi.org/10.1016/j.compeleceng.2015.05.006.
10. Rasouli, N., Razavi, R., & Faragardi, H. R. (2020). EPBLA: Energy-efficient consolidation of virtual machines using learning automata in cloud data centers. *Cluster Computing, 23*(4), 3013–3027. https://doi.org/10.1007/s10586-020-03066-6.
11. Soltanshahi, M., Asemi, R., & Shafiei, N. (2019). Energy-aware virtual machines allocation by Krill Herd algorithm in cloud data centers. *Heliyon, 5*(7), e02066. https://doi.org/10.1016/j.heliyon.2019.e02066.
12. Wang, T., & Hamdi, M. (2016). Presto: Towards efficient online virtual network embedding in virtualized cloud data centers. *Computer Networks, 106*, 196–208. https://doi.org/10.1016/j.comnet.2016.06.036.
13. Santos, E. A., McLean, C., Solinas, C., & Hindle, A. (2018). How does docker affect energy consumption? Evaluating workloads in and out of docker containers. *Journal of Systems and Software, 146*, 14–25. https://doi.org/10.1016/j.jss.2018.07.077.
14. Amoon, M. (2018). A multi criteria-based approach for virtual machines consolidation to save electrical power in cloud data centers. *IEEE Access, 6*, 24110–24117. https://doi.org/10.1109/access.2018.2830183.
15. Wang, B., Chang, X., & Liu, J. (2015). Modeling heterogeneous virtual machines on IaaS data centers. *IEEE Communications Letters, 19*(4), 537–540. https://doi.org/10.1109/lcomm.2015.2403832.
16. Tarahomi, M., & Izadi, M. (2019). Energy efficiency in virtual machines allocation for cloud data centers with Lottery algorithm. *International Journal of Electrical and Computer Engineering (IJECE), 9*(1), 546. https://doi.org/10.11591/ijece.v9i1.pp546-553.