Revolutionizing Software Development: An In-Depth Analysis of Docker's Impact on the Industry

Pushpalatha K S, Mohd Ashar, Prince Kumar, Rishabh Dubey, Suryoday Kaushik
Information Science and Engineering, Acharya Institute of Technology

Abstract - Docker is a software platform that allows developers to create, deploy, and run applications inside containers. Containers are lightweight, portable, and self-contained environments that include all the necessary dependencies and libraries for an application to run. Docker simplifies the deployment process by allowing developers to create a single container image that can be run on any platform that supports Docker.

1. Introduction

Docker is an open-source platform designed to simplify the development and distribution of applications by packaging them with their dependencies into containers. These containers operate in isolation on top of the host operating system's kernel. However, the additional layer of abstraction introduced by containers can potentially impact performance.

Although container technologies have been around for over a decade, Docker has emerged as a highly successful innovation due to its unique features. One key advantage is its ability to create and manage containers easily, allowing applications to be deployed consistently across different environments without requiring modifications. Docker also excels at efficiently running multiple virtual environments on the same hardware. Furthermore, it provides seamless integration with third-party tools, streamlining the deployment process

This paper aims to conduct a systematic literature review on Docker's technology and evaluate its performance. The subsequent sections will present an introduction to Docker's technology, provide a detailed description of its components, briefly compare Docker with virtual machines, and discuss the advantages and disadvantages of Docker containers.

2. Docker

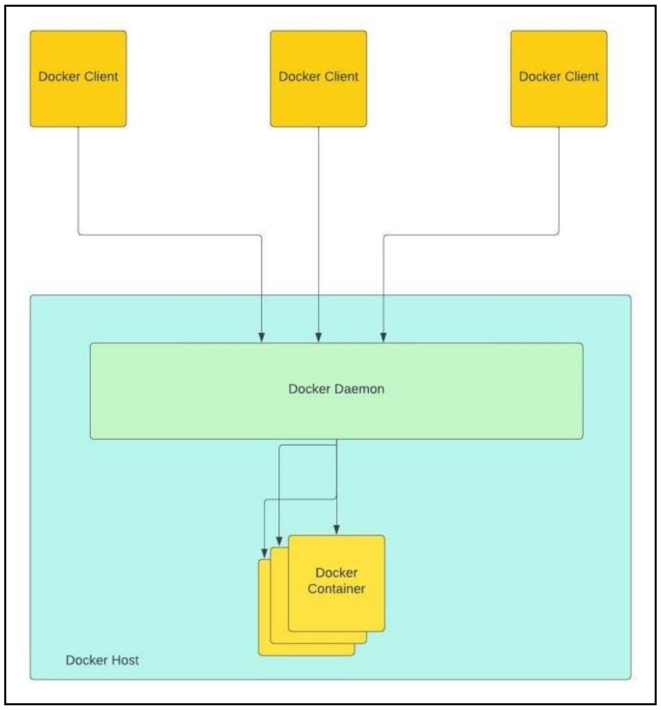
Docker offers automation capabilities for deploying applications into containers. It adds a deployment engine on top of the virtualized container environment, allowing for efficient execution of code. Docker is designed to provide a lightweight and fast environment for running code, and it also facilitates a streamlined workflow by enabling code testing before production deployment. According to Russell (2015), Docker allows for quick code testing and deployment into production environments. Turnbull (2014) emphasizes that Docker is remarkably straightforward. Getting started with Docker only requires a simple configuration system and the Docker binary with a Linux kernel.

3. Docker Architecture

Docker utilizes a client-server architecture where the Docker client communicates with the Docker daemon. The Docker daemon is responsible for performing tasks such as building, running, and distributing Docker containers. The client and daemon can be located on the same system, or the client can connect to a remote daemon. Communication between the client and daemon occurs via a REST API, either through UNIX sockets or a network interface. Docker Compose is another client that allows working with applications composed of multiple containers.

3.1 The Docker Daemon

The Docker daemon, also known as dockerd, is responsible for receiving and processing Docker API requests. It handles various Docker objects, including images, containers, networks, and volumes. Additionally, a daemon has the ability to communicate with other daemons for managing Docker services.



3.2 Docker Images

There are two approaches to building a Docker image. The first method involves using a read-only template as the foundation. Typically, operating system images like Ubuntu 14.04 LTS or Fedora 20 serve as base images, providing a complete running OS within a container. Alternatively, a base image can be created from scratch, and additional applications can be added by modifying it. However, this requires building a new image, which is achieved through a process known as "committing a change." The second method involves creating a Dockerfile, which is a text file containing a set of instructions. When the "Docker build" command is executed from the command line, it follows the instructions specified in the Dockerfile to build an image. This approach offers an automated way of constructing an image by defining the desired configuration in the Dockerfile.

3.3 Docker Registries

Docker images are stored in Docker registries, similar to source code repositories. Registries provide a centralized location for pushing and pulling images. There are two types of registries: public and private. Public registries, such as Docker Hub, allow users to access and pull existing images, as well as push their own images without starting from scratch. Docker Hub serves as a widely accessible public registry, enabling easy distribution of images. It offers a convenient way to share and distribute images to a specific audience, whether it is a public or private setting.

4. Virtual Machine vs. Docker

Virtualization is a well-established concept that has been extensively utilized in cloud computing. It gained prominence with the acceptance of Infrastructure-as-a-Service (IaaS) as a critical approach for system configuration, resource allocation, and multi-tenancy. Virtualized resources are pivotal in addressing various challenges within cloud computing. They form the foundation for the fundamental techniques employed in cloud computing. Figure 2 depicts the architecture of a virtual machine.

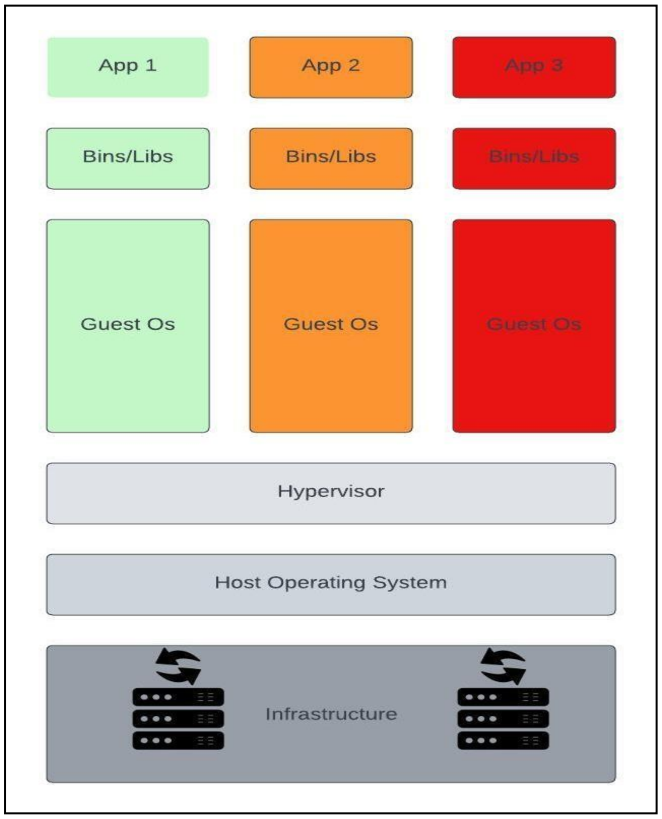


Fig. 2 Virtual Machine architecture [11].

A hypervisor serves as a virtual platform situated between the host and guest operating systems. It enables the management of multiple operating systems on a single server, operating as an intermediary between the operating system and the CPU. Virtualization is divided into two main categories: Para-Virtualization and Full Virtualization. Para-Virtualization involves modifying the guest operating system to interact with the hypervisor, while Full Virtualization emulates the complete hardware environment for each guest operating system. On the other hand, Docker containers are managed by the Docker tool and leverage operating system-level virtualization. Linux containers, depicted in Figure 3, are isolated within a single control host.

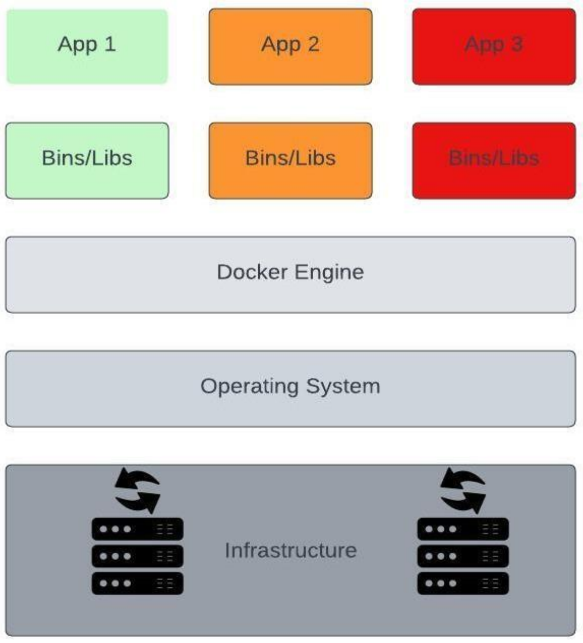


Fig. 3 Docker Container architecture [11].

Linux containers have emerged as an architecture to efficiently manage CPU resources, as stated by Waldspurger (2002). Unlike Hyper-V or VMWare, where running more than ten virtual machines can be challenging due to overhead, containers solve this problem by utilizing only the necessary resources for services or applications. This enables the execution of over 50 container requests on a low-configured machine.

For example, consider an organization providing email security services. These services primarily involve tasks such as virus and spam checking, malware detection, message forwarding, log management, and reporting. Since these functions typically do not require associated dependencies, OS-level libraries, or kernel data structures, it is beneficial to containerize each component using technologies like OpenVZ or Docker, rather than using virtual machines.

In enterprises, virtual machines are commonly used for component testing, which consumes significant CPU and memory resources. Container technology offers resource efficiency guarantees, ensuring that excessive workloads do not affect resource performance. Moreover, containers have faster installation times compared to virtual machines, making them more adaptable.

However, it is important to note that both Docker and OpenVZ have faced scrutiny regarding their security aspects. Reduced isolation in containers can impact security, as the root users of the host Linux system can potentially access the containers since they share the same kernel and operating system. Although Docker isolates the applications running within containers from the host, its isolation is not as strong as that of virtual machines. Additionally, certain applications may require a different operating system and cannot be run within a containerized environment.

5. Advantages of Docker Container

Docker containers offer numerous advantages that have greatly enhanced the efficiency, consistency, scalability, and security of software development and deployment. The key advantages of utilizing Docker containers are as follows:

5.1 Speed

Speed is a highly acclaimed advantage of using containers, and it is crucial to highlight this aspect when discussing the benefits of Docker. Containers are known for their fast performance, which is attributed to their small size. Building a container is a quick process compared to other methods. The compact nature of containers enables faster development, testing, and deployment. Once a container is built, it can be easily pushed for testing and seamlessly transitioned to the production environment. This swift process significantly speeds up the overall software development lifecycle.

5.2 Portability

Docker containers provide a consistent environment that can be easily packaged and deployed across different systems, including development, testing, and production environments. This ensures that applications run reliably and consistently regardless of the underlying infrastructure

5.3 Scalability

Docker provides seamless scalability by allowing the easy replication and distribution of containers across multiple hosts or cloud environments. Containers can be quickly spun up or down to meet changing demands, enabling efficient scaling of applications.

5.4 Consistency

Docker containers play a crucial role in maintaining consistency throughout the development and deployment process. Developers have the ability to package their applications along with all the required components, such as the operating system, dependencies, and libraries. This ensures that the application runs consistently across different environments, regardless of the underlying infrastructure

5.5 Security

Docker maximizes resource utilization by leveraging available resources more efficiently without the need for a hypervisor. Unlike traditional virtualization methods, Docker containers offer enhanced security by isolating applications from the underlying host system.

5.6 Isolation

Docker containers encapsulate applications and their dependencies, isolating them from the host system and other containers. This isolation enhances security and prevents conflicts between different applications or services running on the same system.

6. Disadvantages of Docker Container

There are some drawbacks of docker containers, which are listed below :

- Complete virtualization is not provided by a docker because it depends on the Linux kernel, which is provided by the local host.
- Currently, docker does not run on older machines. It only supports 64-bit local machines.
- The complete virtualized environment must be provided by the docker container for Windows and Mac machines. Even though the *boot2docker* tool fills this gap, but still, it should be checked whether it makes obstructions to acceptance by users of these systems or the integration and performance with the host machine's operating system are adequate [4].
- It is necessary that the possibility of security issues should be evaluated. Building off trusting binaries could be made easier by digitally signing docker images, for future support.
- An important concern is to check if the teaching community or scientific researcher will significantly think of adopting docker.

7. Docker Performance

Seo et al. (2014) used two servers with the same configuration in the cloud environment. One server was used for docker and the other one was for an Open Stack platform for KVM by means of a virtualization tool [8]. According to him, a VM works docker does not contain a guest operating system. Therefore, it takes very little time in distributing and gathering images. The boot time is also veryshort. These are the technologies to benchmark some applications [7]. He explains that Xen would be a better choice in the sense of equally distributing resources, performance is not dependent on the other tasks, and it is executed on the same machine. However, LXC is much better in the sense of getting most of the hardware resources or for the execution of smaller isolated processes. In private and dot clouds, LXC is a better option.
Felter et al. (2014) evaluate the performance of three different environments, Native, Docker, and KVM [3]. He clarifies that containers and VMs are both mature innovation that has profited from last 10 years of incremental equipment and programming enhancements. According to this research, docker is equivalent to or surpasses KVM execution for each situation they tried. Their outcomes demonstrate that both KVM and docker present irrelevant overhead for CPU and memory execution. It has also been shown that the overall performance of docker is better than the Local Host, as the applications were executed and responded faster than in Local Host. Moreover, fewer hardware resources were used in docker container to perform the tasks.
Docker is really a future demanding technology. As users and developers would know more about the docker and its capabilities then they would consider replacing traditional virtualization with docker technology. Docker provides many simple and useful features. To get the best performance and results, it is highly recommended to move up from the default configuration. Containers provide advanced density, better performance, scalability, and usability as compared with traditional virtualization because containers smartly utilize its resources, which reduce the chance of unnecessary overhead. Containers are better in performance than virtual machine, because containers take less start-up time. Docker has removed the biggest issue of "dependency". Now containers have all of their required dependencies, which help containers to be properly built, and to execute them in any docker environment. An additional layer of isolation is provided by the container, which increases the containers' security. Docker is not as insecure as people normally think, but it provides a complete protection.

8. Docker vs. other Container Technology

In this area, the execution of application virtualization and the execution of the docker holder will be discussed, and the assessment of other containerize technology will be compared and looked into. Seo et al.(2014) summarize that there's no visitor OS of docker in the cloud, so the capacity and the wastage of CPU assets areless. The pictures are not exasperates; boot time is faster and the time of producing the pictures is brief. These are the benefits of docker cloud in comparison with VM Cloud.They utilized two comparative servers with the same configuration in the cloud environment. One server was utilized for docker and the other one was for an Open Stack stage for KVM IJCSNS Universal Diary of Computer Science and Organize Security, by implies of a virtualization apparatus. Ubuntu Server was used as a base stage. To calculate the surmised boot-time, 20 pictures were generated on each server and boot time was checked. Boot time of docker was lesser that the boot time of KVM. Docker employments the Have OS, while KVM uses Visitor OS.

Virtual Machines vs. Containers

Virtual machines utilize extra layers between the have OS and the visitor OS. This layer is called a hypervisor. Docker includes an extra layer of has that runs the system and where the application is virtualized and runs, it's called the Docker motor. Since Docker does not utilize the visitor work, this makes the docker holder and the virtual machines have a tremendous contrast in execution. Docker beats KVM in terms of startup time and computation speed, concurring to Seo et al (2014), Felter et al (2014) shows that between Docker and KVM is no contrast in squandering assets (overhead), but may be a contrast in execution as KVM is speedier than Docker. Scheepers (2014) found that LXC takes longer to total a errand, whereas Xen Server takes less. LXC is superior in terms of investing less whereas Xen is way better in terms of designating on normal.

9. Summary

Docker computerizes the applications when they are containerized. An additional layer of docker motor is included to the have working framework. The execution of docker is faster than virtual machines because it has no visitor operating system and less asset overhead.

References

[1] Boettiger, C. (2015). An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1), 71-79.
[2] Bui, T. (2015). Analysis of docker security. arXiv preprint arXiv:1501.02967.
[3] Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. technology, 28, 32.
[4] Harji, A. S., Buhr, P. A., & Brecht, T. (2013). Our troubles with Linux Kernel upgrades and why you should care. ACM SIGOPS Operating Systems Review, 47(2), 66-72.
[5] Joy, A. M. (2015). Performance comparison between Linux containers and virtual machines. Paper presented at the Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in.
[6] Russell, B. (2015). Passive Benchmarking with docker LXC,
[7] Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison.
[8] Seo, K.-T., Hwang, H.-S., Moon, I.-Y., Kwon, O.-Y., & Kim, B.-J. (2014). Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud.
[9] Turnbull, J. (2014). The Docker Book: Containerization is the new virtualization.
[10] Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on, 16(9), 1128-1142.