Docker for Accelerating Development and Deployment in Software Development Life Cycle

Alif Arief Maulana

June 14, 2023

Abstract

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. Containers are created from images that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories. Docker provides tooling and a platform to manage the lifecycle of your containers.

1 Introduction

Docker 1 is a platform for building, packaging, and deploying containerized applications. It consists of several components that work together to provide a complete containerization solution. Here are the main components of Docker:

1. Docker Image: A Docker image is a read-only template that defines the environment for a container. It contains the application and all its dependencies, as well as the instructions for creating the container when it is run for the first time. Docker images are created using a Dockerfile, which is a text file that contains a list of instructions for building the image.

2. Docker Container: A Docker container is a runtime instance of a Docker image. It is a lightweight and portable encapsulation of an application and its dependencies that allows it to run virtually anywhere. Docker containers are created from Docker images and run the actual application. Each container is isolated from the host environment and other containers, and has its own set of resources allocated to it (Abhishek and Rajeswara Rao, 2021).

3. Docker Engine: The core component of Docker is the Docker Engine, which is a lightweight runtime that allows you to run Docker containers. It includes the Docker daemon, which is responsible for managing containers, images, networks, and volumes, as well as the Docker CLI, which provides a command-line interface for interacting with Docker.

4. Docker Hub: Docker Hub is a public registry of Docker images, where developers can find, store, and share Docker images. It provides a convenient way to discover and download pre-built Docker images for popular software applications, as well as a platform for hosting and distributing custom Docker images Wenhao and Zheng (2020).

5. Docker Compose: Docker Compose is a tool for defining and running multi-container Docker applications. It allows developers to define the different services that make up an application, their configuration settings, and how they are connected and deployed.

6. Docker Swarm: Docker Swarm is a native clustering and orchestration tool for Docker containers. It enables developers to deploy and manage a cluster of Docker nodes as a single virtual system, providing a highly available and scalable platform for running distributed applications.

7. Docker Registry: Docker Registry is a private registry that allows organizations to store and share Docker images within their own environment. It provides a secure and scalable way to manage custom Docker images and distribute them across different teams and environments.

8. Docker CLI: The Docker CLI is a command-line interface that allows developers to interact with the Docker Engine and manage Docker containers, images, networks, and volumes. It provides a simple and powerful way to build, deploy, and manage containerized applications.

Overall, Docker is a powerful platform for building and deploying containerized applications, providing a complete set of tools and components for managing the entire container lifecycle.

The use of Docker in software development has gained significant attention in recent years Merkel, 2014; Boettiger, 2015. Docker is a containerization platform that offers various benefits for developers and organizations, including improved portability and scalability. This journal explores the impact of Docker on accelerating development and deployment processes in the software development life cycle (SDLC).

2 Background Study

The background study section provides an in-depth understanding of Docker containerization technology. It explains the fundamental concepts and principles underlying Docker, including containerization, images, and containers. The section highlights the advantages of Docker over traditional virtualization approaches, such as lightweight resource utilization and faster startup times. Furthermore, it explores the key features of Docker that make it an ideal tool for enhancing development and deployment in the SDLC. These features include container isolation, image versioning, and efficient resource management.

To support the benefits of Docker in the SDLC, the background study section also discusses relevant literature on Docker adoption in various industries and domains. It showcases real-world examples of organizations that have successfully leveraged Docker to improve their development and deployment processes. These case studies highlight the transformative potential of Docker in terms of increased efficiency, reduced overhead, and improved collaboration among development teams.

Figure 1: Docker Logo

In this article, we have looked at what containers are and how they are used in software development. We have also looked at the different tools and technologies that are used to create and manage containers, including Docker, Docker Compose, Docker Swarm, and Docker Machine. Finally, we have looked at some of the key features of Docker Swarm, including service discovery, load balancing, scaling, rolling updates, and security.

Virtual Machines	**Containers**
Heavyweight.	Lightweight.
Limited performance.	Native performance.
Each VM runs in its own OS.	All containers share the host OS.
Startup time in minutes.	Startup time in milliseconds.
Allocates required memory.	Requires less memory space.
Fully isolated and hence more secure.	Process-level isolation, possibly less secure.

To understand the fundamental concepts and advantages of Docker, it is essential to delve into the background study. Docker provides lightweight Linux containers that ensure consistent development and deployment environments Merkel, 2014. It offers reproducibility and simplifies the process of creating portable software packages Boettiger, 2015.

References

Merkel, Dirk (2014). “Docker: lightweight linux containers for consistent development and deployment”. In: *Linux journal* 2014.239, p. 2. URL: `https://www.linuxjournal.com/content/docker`.

Boettiger, Carl (2015). “An introduction to docker for reproducible research”. In: *ACM SIGOPS Operating Systems Review* 49.1, pp. 71–79. DOI: `10.1145/2723872.2723882`.

Wenhao, Jiang and Li Zheng (2020). “Vulnerability Analysis and Security Research of Docker Container”. In: *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 354–357. DOI: `10.1109/ICISCAE51034.2020.9236837`.

Abhishek, Manish Kumar and D. Rajeswara Rao (2021). “Framework to Secure Docker Containers”. In: *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, pp. 152–156. DOI: `10.1109/WorldS451998.2021.9514041`.