**About Docker**

Its an open-source project that automates the deployment of software applications inside **containers** by providing an additional layer of abstraction and automation of **OS-level virtualization** on Linux.

Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

Docker is Developed in GO Language

|  |
| --- |
| Go (often referred to as golang) is a [free](https://en.wikipedia.org/wiki/Free_software) and [open source](https://en.wikipedia.org/wiki/Open-source) [programming language](https://en.wikipedia.org/wiki/Programming_language) created at [Google](https://en.wikipedia.org/wiki/Google) in 2007 by Robert Griesemer, [Rob Pike](https://en.wikipedia.org/wiki/Rob_Pike), and [Ken Thompson](https://en.wikipedia.org/wiki/Ken_Thompson). It is a [compiled](https://en.wikipedia.org/wiki/Compiler), [statically typed](https://en.wikipedia.org/wiki/Static_typing) language in the tradition of [Algol](https://en.wikipedia.org/wiki/ALGOL) and [C](https://en.wikipedia.org/wiki/C_%28programming_language%29), with [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29), limited [structural typing](https://en.wikipedia.org/wiki/Structural_type_system), [memory safety](https://en.wikipedia.org/wiki/Memory_safety) features and [CSP](https://en.wikipedia.org/wiki/Communicating_sequential_processes)-style [concurrent programming](https://en.wikipedia.org/wiki/Concurrent_programming) features added.[[](https://en.wikipedia.org/wiki/Go_%28programming_language%29#cite_note-boldly-14) |

**About Docker History/info**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | | |  | | | [**Original author(s)**](https://en.wikipedia.org/wiki/Software_developer) | [Solomon Hykes](https://en.wikipedia.org/wiki/Solomon_Hykes) | | [**Developer(s)**](https://en.wikipedia.org/wiki/Software_developer) | [Docker, Inc.](https://en.wikipedia.org/wiki/Docker,_Inc.) | | **Initial release** | 13 March 2013; 4 years ago | |  | | | [**Stable release**](https://en.wikipedia.org/wiki/Software_release_life_cycle) | 17.03.0-ce[[1]](https://en.wikipedia.org/wiki/Docker_%28software%29#cite_note-1) / 1 March 2017; 40 days ago | | [**Repository**](https://en.wikipedia.org/wiki/Repository_%28version_control%29) | <https://github.com/docker/docker>, <https://github.com/docker/docker.git> | | **Written in** | [Go](https://en.wikipedia.org/wiki/Go_%28programming_language%29) | | [**Operating system**](https://en.wikipedia.org/wiki/Operating_system) | [Linux](https://en.wikipedia.org/wiki/Linux), [Windows](https://en.wikipedia.org/wiki/Windows) | | [**Platform**](https://en.wikipedia.org/wiki/Computing_platform) | [x86-64](https://en.wikipedia.org/wiki/X86-64), [ARM](https://en.wikipedia.org/wiki/ARM_architecture) (experimental) with modern Linux kernel, or [x86-64](https://en.wikipedia.org/wiki/X86-64) Windows with Hyper-V capabilities | | [**Type**](https://en.wikipedia.org/wiki/Software_categories#Broad_categories) | [Operating-system-level virtualization](https://en.wikipedia.org/wiki/Operating-system-level_virtualization) | | [**License**](https://en.wikipedia.org/wiki/Software_license) | [Apache License](https://en.wikipedia.org/wiki/Apache_License) 2.0 | | **Website** | [www.docker.com](https://www.docker.com/) | |

### What are containers?

The industry standard today is to use Virtual Machines (VMs) to run software applications. VMs run applications inside a guest Operating System, which runs on virtual hardware powered by the server’s host OS.

VMs are great at providing full process isolation for applications: there are very few ways a problem in the host operating system can affect the software running in the guest operating system, and vice-versa

Containers take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the isolation of virtual machines at a fraction of the computing power.

## **Features of Docker**

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.

## **Components of Docker**

Docker has the following components

* **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
* **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
* **Docker for Windows** − It allows one to run Docker containers on the Windows OS.
* **Docker Engine** − It is used for building Docker images and creating Docker containers.
* **Docker Hub** − This is the registry which is used to host various Docker images.
* **Docker Compose** − This is used to define applications using multiple Docker containers.

## **The Docker platform**

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight because they don’t need the extra load of a hypervisor, but run directly within the host machine’s kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines. You can even run Docker containers within host machines that are actually virtual machines!

Docker provides tooling and a platform to manage the lifecycle of your containers:

* Develop your application and its supporting components using containers.
* The container becomes the unit for distributing and testing your application.
* When you’re ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

## **Docker Engine**

Docker Engine is a client-server application with these major components:

* A server which is a type of long-running program called a daemon process (the dockerd command).
* A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface (CLI) client (the docker command).



The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

The daemon creates and manages Docker objects, such as images, containers, networks, and volumes.

**Note**: Docker is licensed under the open source Apache 2.0 license.

## **What can I use Docker for?**

**Fast, consistent delivery of your applications**

Docker streamlines the development lifecycle by allowing developers to work in standardized environments using local containers which provide your applications and services. Containers are great for continuous integration and continuous development (CI/CD) workflows.

Consider the following example scenario.

* Your developers write code locally and share their work with their colleagues using Docker containers.
* They use Docker to push their applications into a test environment and execute automated and manual tests.
* When developers find bugs, they can fix them in the development environment and redeploy them to the test environment for testing and validation.
* When testing is complete, getting the fix to the customer is as simple as pushing the updated image to the production environment.

**Responsive deployment and scaling**

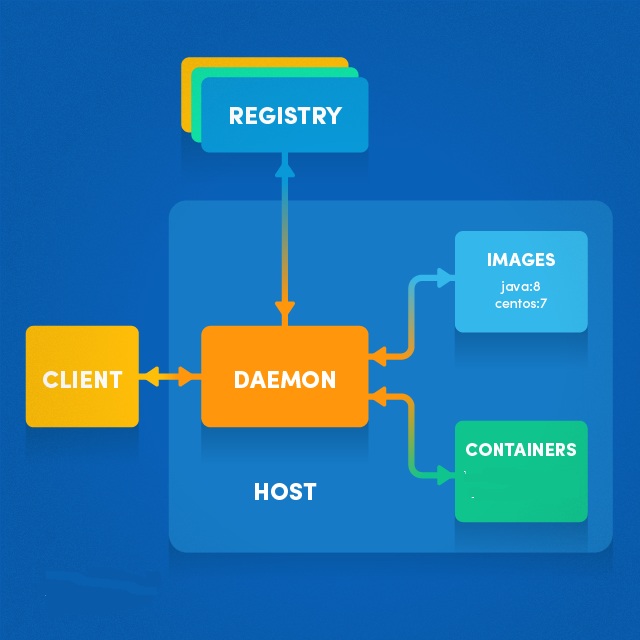
Docker’s container-based platform allows for highly portable workloads. Docker containers can run on a developer’s local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.

Docker’s portability and lightweight nature also make it easy to dynamically manage workloads, scaling up or tearing down applications and services as business needs dictate, in near real time.

**Running more workloads on the same hardware**

Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your compute capacity to achieve your business goals. Docker is perfect for high density environments and for small and medium deployments where you need to do more with fewer resources.

## **Docker Architecture**



By default, the main **registry** is the Docker Hub which hosts public and official images. Organizations can also host their private registries if they desire.

**Images** can be downloaded from registries explicitly from Docker Repositry (docker pull imageName) or implicitly when starting a container. Once the image is downloaded it is cached locally.

**Docker daemon** responsible for creating, running, and monitoring containers, It also takes care of building and storing images

**Docker Client**- talks to the daemon via HTTP.

**Containers** are the instances of images - they are the living thing. There could be multiple containers running based on the same image.