Introduction

What is Docker?

Wikipedia defines Docker as> 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" refers to several things. This includes an open-source community project which started in 2013; tools from the open-source project; Docker Inc., the company that is the primary supporter of that project; and the tools that the company formally supports.

Tips: From Wikipedia
Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources.

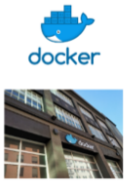
- Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. Linux.
- The key benefit of Docker is that it allows users to package an application with all of its dependencies into a standardized unit for software development. Unlike virtual machines, containers do not have high overhead and hence enable more efficient usage of the underlying system and resources.

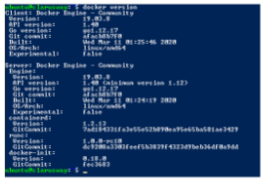
Here's a quick explanation:

- The IT software "Docker" is containerization technology that enables the creation and use of Linux® containers.
- The open source Docker community works to improve these technologies to benefit all users
- The company, Docker Inc., builds on the work of the Docker community, makes it more secure, and shares those advancements back to the greater community.

Tips:
With DOCKER, you can treat containers like extremely lightweight, modular virtual machines. And you get flexibility with those containers—you can create, deploy, copy, and move them from environment to environment, which helps optimize your apps for the cloud.

- Docker as a "Company"
- Docker as a "Product"
- Docker as a "Platform"
- Docker as a "CLI Tool"
- Docker as a "Computer Program"



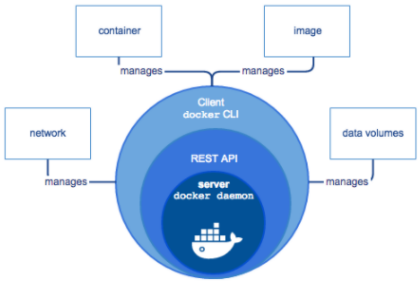


> Q: What is a Docker?
> A: **Docker** is defined as the platform for containerizing the applications to isolate it from each other in order to ensure high availability and more efficiency irrespective of the environments such as **Development, Testing or Production**. All the application related dependencies such as libraries, jar files, server related configurations, infrastructure-related elements will be packaged and formed as a container called containerized applications which does not need any dependency and works independently. It ensures the application to be run irrespective of the external factors. Containers in Docker have support from Docker Engine and Host Operating System to support all the operational or infrastructural related dependencies.
>
> — - Interview Q&A

Docker Engine

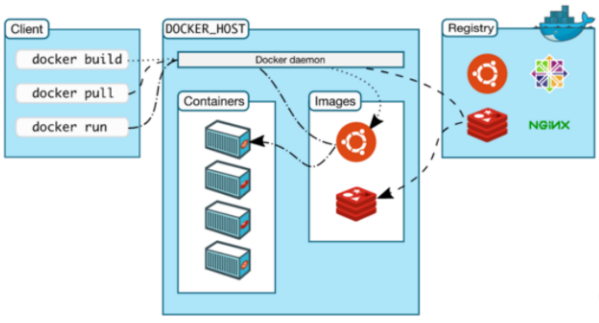
Docker Engine is a client-server application with these major components:

- 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.



Docker Architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets, or a network interface.



The core components that compose Docker:>

- *The Docker client and server also called the Docker Engine.*
- *Docker Images*
- *Registries*
- *Docker Containers*

Docker Client and Server

Docker is a client-server application. The Docker client talks to the Docker server or daemon, which, in turn, does all the work. You'll also sometimes see the Docker daemon called the Docker Engine.

Docker images

Images are the building blocks of the Docker world. Containers are launched from images. Docker images are light-weight, portable, reproduce-able, and declarative.

Registries

Docker stores the images you build in registries. There are two types of registries: public and private. Docker, Inc., operates the public registry for images, called the Docker Hub. You can create an account on the Docker Hub and use it to share and store your images. You can also re-use the `registry` repository capability for free at your premises to store your images.

Containers

Docker helps you build and deploy containers that encapsulate your packages, applications, infrastructures, and services. Docker containers try to maximize resource sharing between containers and allow us to isolate those parts of the applications that are different and need their own space.

Because of **high resource sharing**, we can build Docker containers with a **small footprint** that makes application **distribution easier** and it makes container **startup times faster**.

> Q: What are the components of Docker Architecture and explain?
> A: The Docker works on a client-server architecture. The Docker client establishes communication with the Docker Daemon. The Docker client and Daemon can run on the same system. A Docker client can also be connected to a remote Docker Daemon.
>
> - **Docker Client:** This performs Docker build pull and run operations to establish communication with the Docker Host. The Docker command uses Docker API to call the queries to be run.
> - **Docker Host:** This component contains Docker Daemon, Containers and its images. The images will be the kind of metadata for the applications which are containerized in the containers. The Docker Daemon establishes a connection with Registry.
> - **Registry:** This component will be storing the Docker images. The public registries are Docker Hub and Docker Cloud which can be used by anyone.
>
> — Interview Q&A

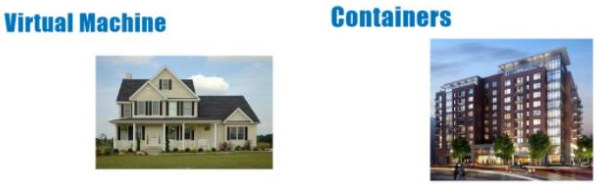
Docker vs. VM's

What are VMs?

A virtual machine (VM) is an emulation of a computer system. Put simply, it makes it possible to run what appears to be many separate computers on hardware that is actually one computer.

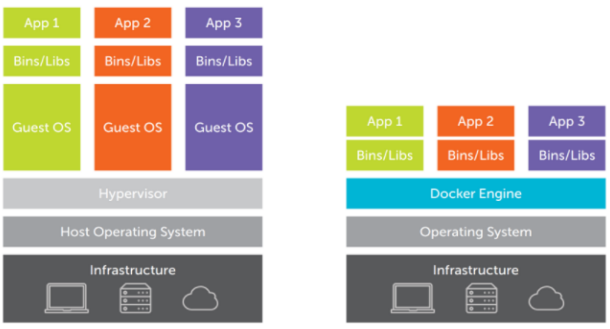
Difference Between VM and Docker

Virtual machines have a full OS with its own memory management installed with the associated overhead of virtual device drivers. In a virtual machine, valuable resources are emulated for the guest OS and hypervisor, which makes it possible to run many instances of one or more operating systems in parallel on a single machine (or host). Every guest OS runs as an individual entity from the host system. Hence, we can look at it an independent full-fledged house where we don't share any resources as shown below:



Docker containers are executed with the Docker engine rather than the hypervisor. Containers are therefore smaller than Virtual Machines and enable **faster startup** with **better performance**, **less isolation** and **greater compatibility** possible due to **sharing** of the host's kernel. Hence, it looks very similar to the residential flats system where we share resources of the building.

Tips:
From Wikipedia: A hypervisor (or virtual machine monitor, VMM) is computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.



Virtual Machines Containers

Virtual Machines are built over the physical hardware, there is a layer of Hypervisor which sits between physical hardware and operating systems. In a broader view, Hypervisor is used to virtualize the hardware which is then configured with the way a user wants it to. Unlike virtual machines where hypervisor divides physical hardware into parts, Containers are like normal operating system processes.

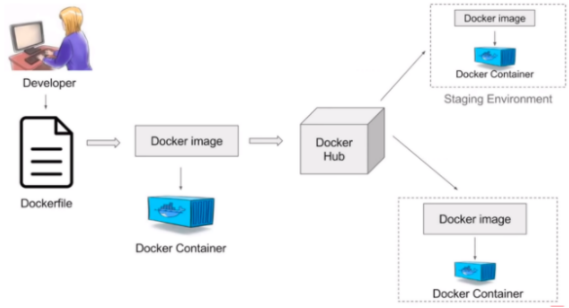
Docker Containers versus Virtual Machines (VM)

Docker	Virtual Machines
All containers share the same kernel of the host	Each VM runs its own OS
Containers initiated in seconds	Boot-up time is in minutes
Images are built incrementally on top of another like layers. Lots of images/snapshots	VMs snapshots are used sparingly
Images can be diffed and can be version controlled. Dockerhub is like GitHub	Not effective diffs. Not version controlled
Can run many Docker containers on a laptop.	Cannot run more than a couple of VMS on an average laptop
Multiple Docker containers can be started from one Docker image	Only one VM can be started from one set of VMX and VMDK files

> Q: What is the advantage of Docker over hypervisors?
> A: Docker is light weight and more efficient in terms of resource uses because it uses the host underlying kernel rather than creating its own hypervisor.
>
> — Interview Q&A

Terminology

> In this course, we will use a lot of Docker-specific jargon which might be confusing to some. So before we go further, we need to clarify some terminology that is used frequently in the Docker ecosystem.



Docker Editions

- **Docker Community Edition (CE)** is ideal for Developers who are looking for experimenting docker and creating container-based applications. It's free.
- **Docker Enterprise Edition (EE)** is a Containers-as-a-Service (CaaS) platform. Enterprise Edition Subscription packages include an integrated Docker platform and tooling for container management and security.

Docker ID

- Your free Docker ID grants you access to Docker services such as the Docker Store, Docker Cloud, Docker Hub repositories, and some beta programs. Your Docker ID becomes repository namespace used by hosted services such as Docker Hub and Docker Cloud. All you need is an email address.

Registry

- A Docker registry stores Docker images.
- **Docker Hub (Like GitHub)** is a cloud-based registry service that allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.
- **Docker Cloud** uses the hosted Docker Cloud Registry, which allows you to publish Dockerized images on the internet either publicly or privately. Docker Cloud can also store pre-built images, or link to your source code so it can build the code into Docker images, and optionally test the resulting images before pushing them to a repository.

Docker Client

- The command-line tool that allows the user to interact with the daemon. It is the primary user interface to Docker. Accepts commands from the user and communicates back and forth with a Docker daemon.

Docker Daemon

- The background service running on the host that manages the building, running and distributing Docker containers. Runs on a host machine.

Dockerfile

- Dockerfile is a text document that contains all commands a user could call on the command line to create an image.

Docker Image

- Docker image is a read-only template with instructions for creating a Docker container.

Docker Container

- Created from Docker images and run the actual application. It is a runnable instance of an image.

Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.

> Q: What are Docker Image and Docker Hub?
> A: The Docker Image is a set of files and a combination of parameters that will allow creating the instances to run in separate containers as an isolated process. The Docker hub is a kind of repository to the images where these images can be stored and this access is public. The Docker run command can be used to create the instance called container which can be run using the Docker image. Docker hub is the largest public repository of the image containers which is being maintained by the community of developers and individual contributors.
>
> — - Interview Q&A

Complementary Lesson about What Docker is