**========================================================**

**Docker**

**========================================================**

* 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 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 the high overhead and hence enable more efficient usage of the underlying system and resources.

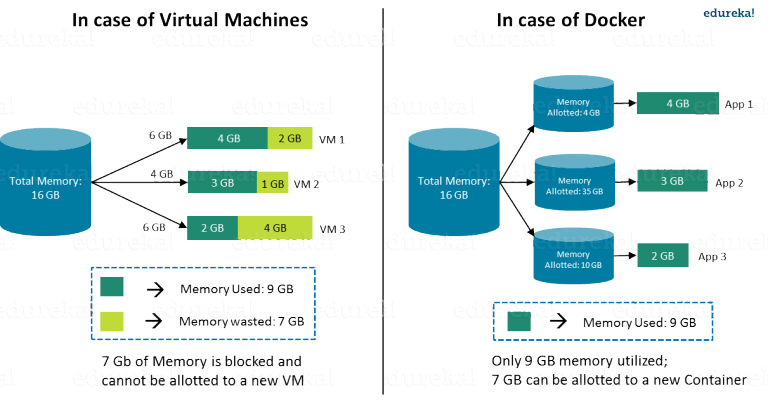
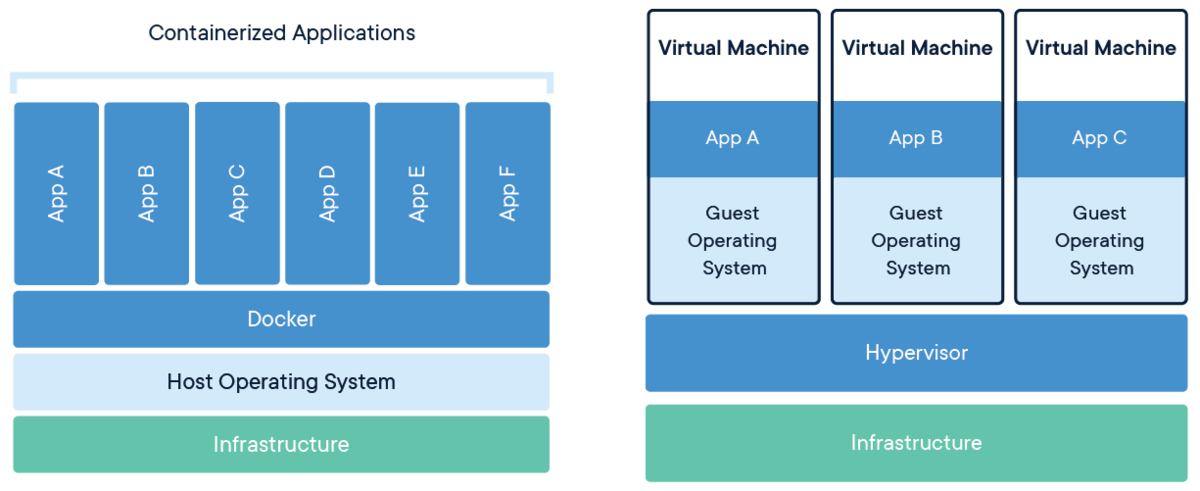
### **What are containers?**

Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOs container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application oriented container like CakePHP container or a Tomcat-Ubuntu container etc.

**Containers Vs. Virtual Machines**

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

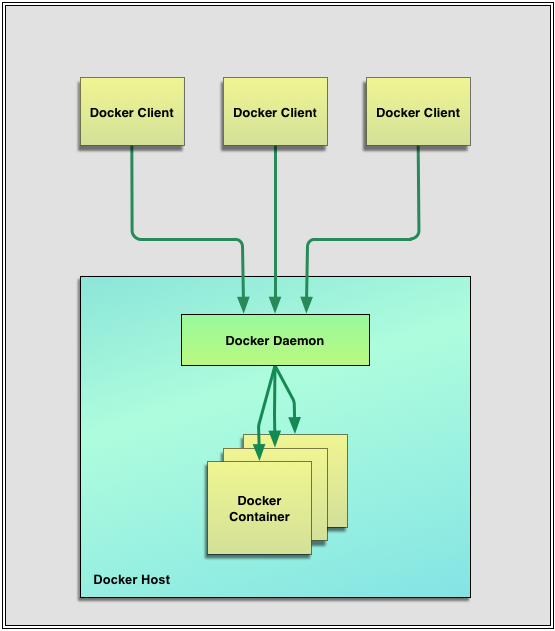
* Size – This parameter will compare Virtual Machine & Docker Container on their resource they utilize.
* Startup – This parameter will compare on the basis of their boot time.
* Integration – This parameter will compare on their ability to integrate with other tools with ease.



**Docker components**

Let's look at the core components that compose Docker:

1. The Docker client and server
2. Docker Images
3. Registries
4. Docker Containers



**Docker images**

Images are the building blocks of the Docker world. You launch your containers from images. Images are the "build" part of Docker's life cycle. They are a layered format, using Union file systems, that are built step-by-step using a series of instructions. For example:

• Add a file.

• Run a command.

• Open a port.

You can consider images to be the "source code" for your containers. They are highly portable and can be shared, stored, and updated.

**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 own images.
* The Docker Hub also contains, at last count, over 10,000 images that other people have built and shared.
* You can also store images that you want to keep private on the Docker Hub. These images might include source code or other proprietary information you want to keep secure or only share with other members of your team or organization.
* You can also run your own private registry, This allows you to store images behind your firewall, which may be a requirement for some organizations.

**Containers**

Docker helps you build and deploy containers inside of which you can package your applications and services. As we've just learnt, containers are launched from images and can contain one or more running processes. You can think about images as the building or packing aspect of Docker and the containers as the running or execution aspect of Docker.

A Docker container is:

• An image format.

• A set of standard operations.

• An execution environment.