Docker
Overview and Use Case

--Jannat

Topics

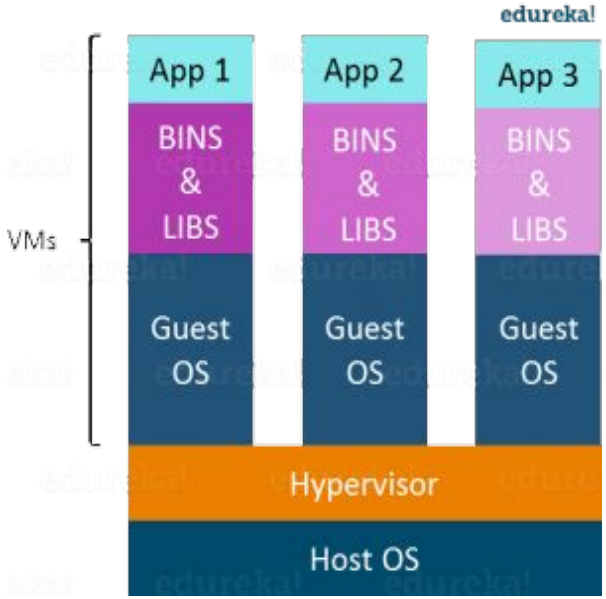
- What is Virtualization?
- What is Containerization
- Advantages of Containerization over Virtualization
- Introduction to Docker
- Benefits of Docker
- Virtualization vs Containerization
- Docker Installation
- Dockerfile, Docker Image & Docker Container
- What is Docker Hub?
- Docker Architecture
- Docker Compose
- Basic command outputs

Preliminary

- **Docker** is gaining popularity and its usage is spreading like wildfire.The reason for Docker's growing popularity is the extent to which it can be used in an IT organization. Very few tools out there have the functionality to find itself useful to both developers and as well as system administrators. **Docker is one such tool that truly lives up to its promise of Build, Ship and Run.**
- In simple words, Docker is a software containerization platform, meaning you can build your application, package them along with their dependencies into a container and then these containers can be easily shipped to run on other machines.
- *For example:* Let's consider a linux based application which has been written both in Ruby and Python. This application requires a specific version of linux, Ruby and Python. In order to avoid any version conflicts on user's end, a linux docker container can be created with the required versions of Ruby and Python installed along with the application. Now the end users can use the application easily by running this container without worrying about the dependencies or any version conflicts.
- These containers uses Containerization which can be considered as an evolved version of Virtualization. The same task can also be achieved using Virtual Machines, however it is not very efficient.

Virtualization

- Virtualization is the technique of importing a Guest operating system on top of a Host operating system.
- This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host. This eliminated the need for extra hardware resource.
- The advantages of Virtual Machines or Virtualization are:
  a. Multiple operating systems can run on the same machine
  b. Maintenance and Recovery were easy in case of failure conditions
  c. Total cost of ownership was also less due to the reduced need for infrastructure



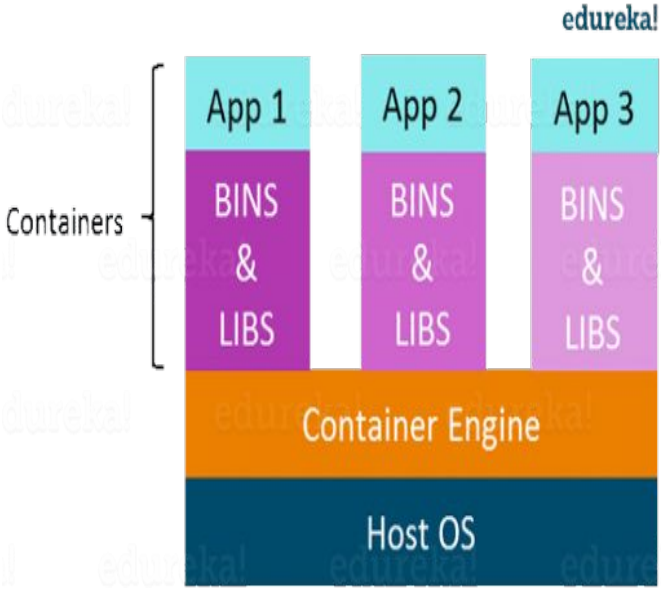
Virtualization (Cont.)

- Virtualization also has some **shortcomings.**
- Running multiple Virtual Machines in the same host operating system leads to performance degradation. This is because of the guest OS running on top of the host OS, which will have its own kernel and set of libraries and dependencies. This takes up a large chunk of system resources, i.e. hard disk, processor and especially RAM.
- Another problem with Virtual Machines which uses virtualization is that it takes almost a minute to boot-up. This is very critical in case of real-time applications.
- Following are the disadvantages of Virtualization:
  - Running multiple Virtual Machines leads to unstable performance
  - Hypervisors are not as efficient as the host operating system
  - Boot up process is long and takes time

These drawbacks led to the emergence of a new technique called Containerization.

Containerization

- Containerization is the technique of bringing virtualization to the operating system level.
- While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system.
- Do note that Containerization is also a type of Virtualization.
- Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines.



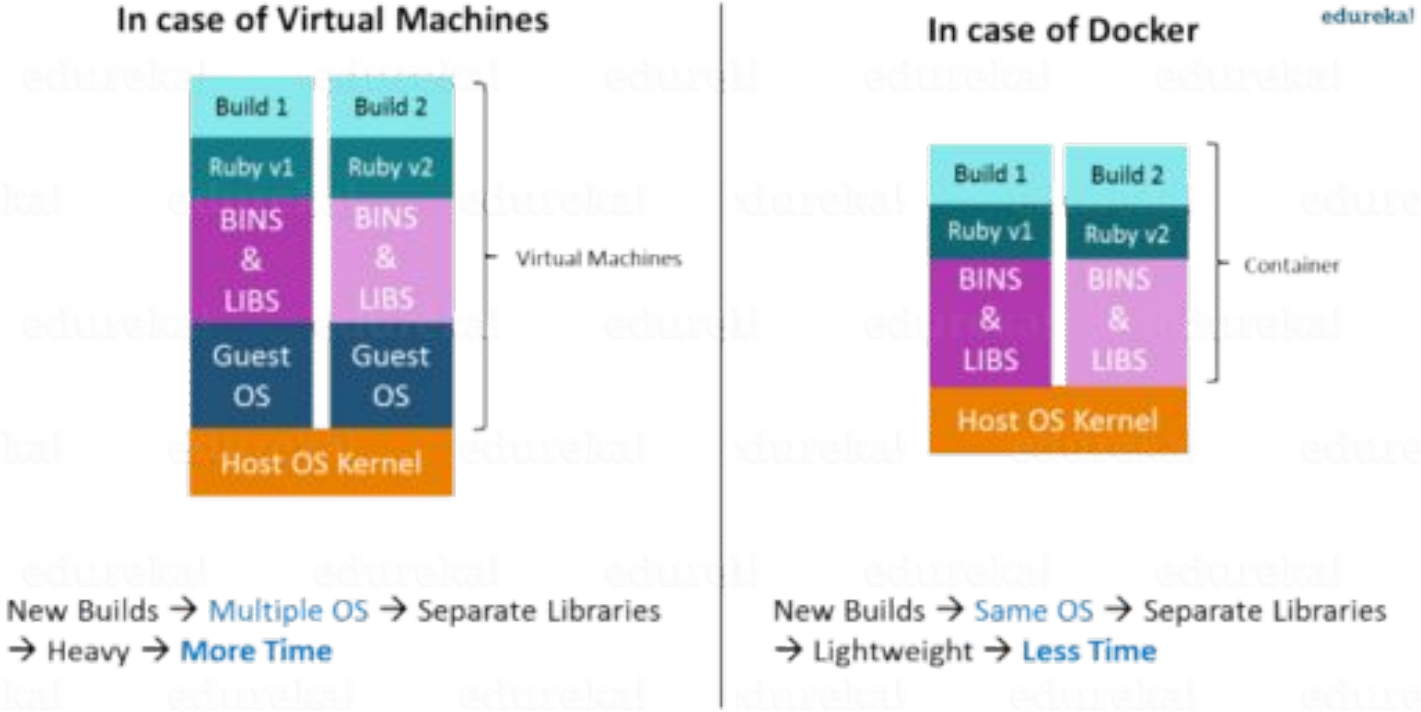
Containerization (Cont.)

- Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast.
- Even booting-up a container takes only a fraction of a second. Because all the containers share, host operating system and holds only the application related binaries & libraries.
- They are lightweight and faster than Virtual Machines.
- Advantages of Containerization over Virtualization:
  - Containers on the same OS kernel are lighter and smaller
  - Better resource utilization compared to VMs
  - Boot-up process is short and takes few seconds

Benefits of Docker

- **As a developer, I can build a container which has different applications installed on it and give it to my QA team who will only need to run the container to replicate the developer environment.**
- Now, the QA team need not install all the dependent software and applications to test the code and this helps them save lots of time and energy.
- This also ensures that the working environment is consistent across all the individuals involved in the process, starting from development to deployment.
- The number of systems can be scaled up easily and the code can be deployed on them effortlessly.

Virtualization vs Containerization



- **Virtualization and Containerization both let you run multiple operating systems inside a host machine.**
- **Virtualization deals with creating many operating systems in a single host machine. Containerization on the other hand will create multiple containers for every type of application as required.**

Docker Installation

```
jannat@jannat-lp:~$ sudo apt-get remove docker docker-engine docker.io containerd runc
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
jannat@jannat-lp:~$ sudo apt-key fingerprint 0EBFCD88
jannat@jannat-lp:~$ sudo add-apt-repository \
>    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
>    $(lsb_release -cs) \
>    stable"
jannat@jannat-lp:~$ sudo apt-get update
jannat@jannat-lp:~$  sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Docker Installation

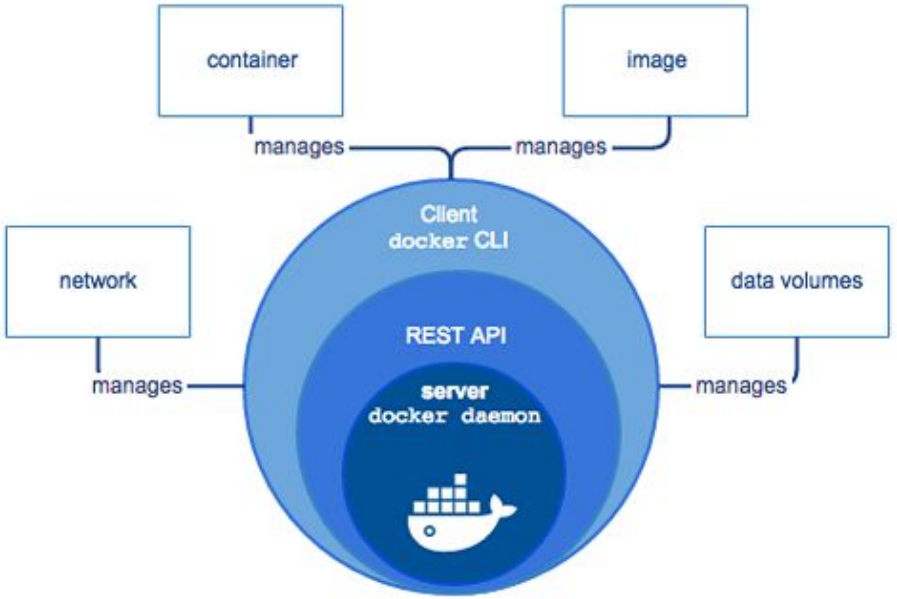
```
jannat@jannat-lp:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker's Workflow

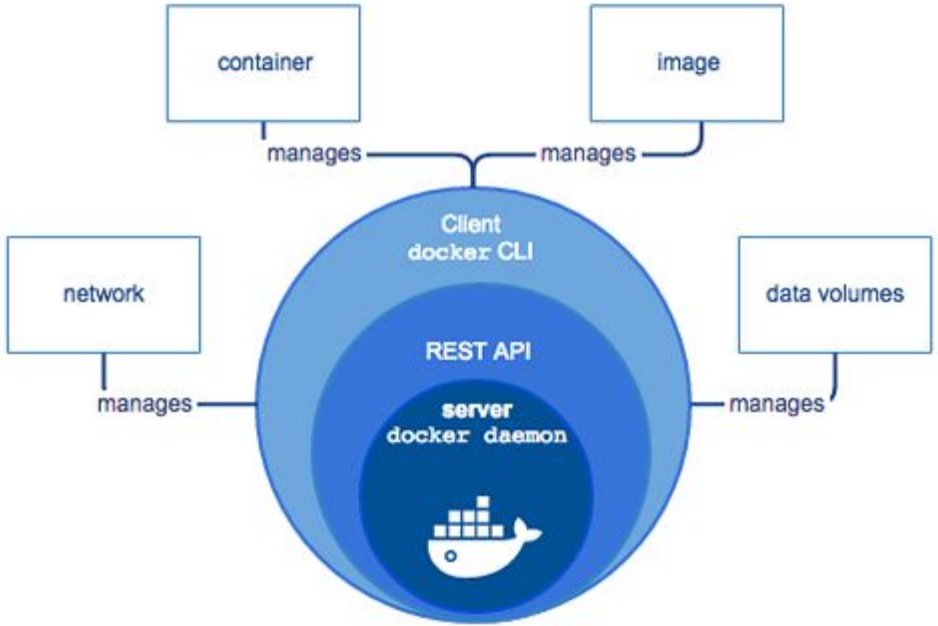
First, let us take a look at Docker Engine and its components so we have a basic idea of how the system works. Docker Engine allows you to develop, assemble, ship, and run applications using the **following components:**

1. **Docker Daemon:** A persistent background process that manages Docker images, containers, networks, and storage volumes. The Docker daemon constantly listens for Docker API requests and processes them.
2. **Docker Engine REST API:** An API is used by applications to interact with the Docker daemon. It can be accessed by an HTTP client.
3. **Docker CLI:** A command-line interface client for interacting with the Docker daemon. It significantly simplifies how you manage container instances and is one of the key reasons why developers love using Docker.



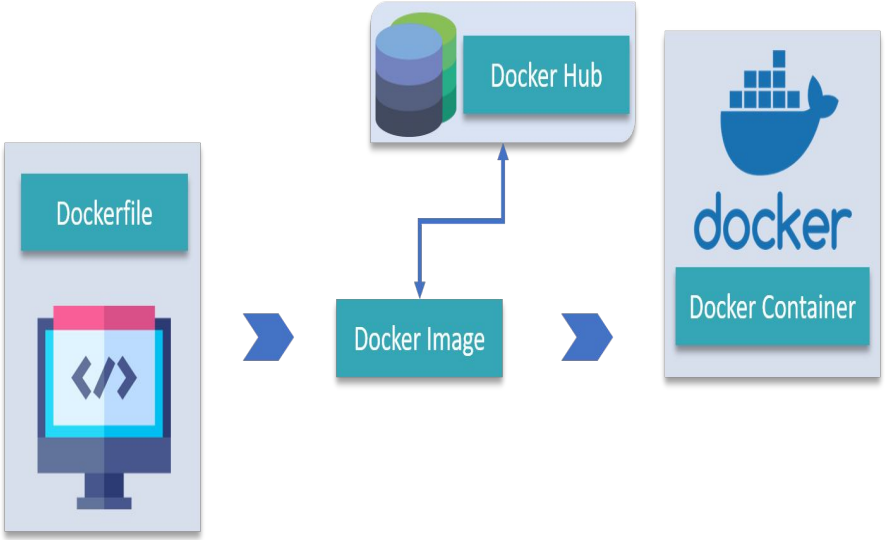
Docker’s Workflow (Cont.)

- At first, Docker client talks to the Docker daemon, which performs the heavy lifting of the building, running, as well as distributing our Docker containers.
- Fundamentally, both the Docker client and daemon can run on the same system.
- We can also connect a Docker client to a remote Docker daemon.
- In addition, by using a REST API, the Docker client and daemon, communicate, over UNIX sockets or a network interface.



Few important concepts: Dockerfile, Docker Image, Docker Container, Docker Hub

1. **Docker Image** is created by the sequence of commands written in a file called as **Dockerfile**.
2. When this Dockerfile is executed using a docker command it results into a Docker Image with a name.
3. When this Image is executed by **"docker run"** command it will by itself start whatever application or service it must start on its execution.



Docker Hub:

Docker Hub is like GitHub for Docker Images. It is basically a cloud registry where you can find Docker Images uploaded by different communities, also you can develop your own image and upload on Docker Hub, but first, you need to create an account on DockerHub.