Author: Kharim Mchatta

Title: Introduction to docker

Date: 3/10/2022

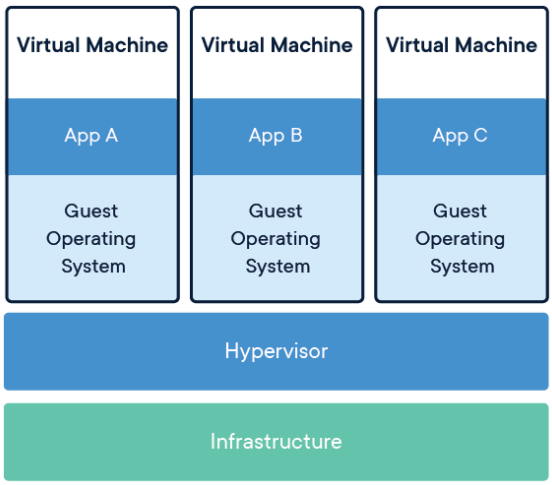
INTRODUCTION

Before talking about docker we need to have a walk back in time before docker became to be. We need to have an understanding how applications were hosted, what solution came after that, and how did we arrive to docker.

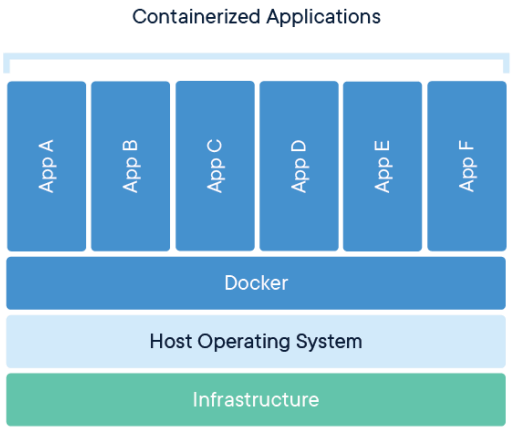
It all started back in the days when all applications were being hosted on a single server which had its own resources (its own dedicated memory, CPU and disk space). This proved to be very costly to run especially those applications with high requirement of system resources. Meaning that the more applications you wanted to run the more resources you needed which includes having more server in place.

Later virtualizations was born. In 1950 virtual memory was invented by Tom Killburn in the Atlas system in Manchester University, then we moved forward to virtual computing which happened in the 1960 to be precise in 1963 which was invented by group of researchers (David Sayre and Les Belady who were led by Bob Nelson) that created a technique of segmenting an IBM 7044 machine into a submachine that were 7044 images with less memory. The team did an extensive research on performance studies of the virtual memory load control and scheduling policies.

This was very significant because in the 1960's people were able to run multiple applications on each virtual machine each being a segmentation of the actual physical server and having its own dedicated resources (its own dedicated memory, CPU and disk space). Although this was a perfect solution for back in the days, it had its drawbacks, and the drawbacks were each virtual machine required its own OS with other costs like licensing costs, required security patching, driver support, admin time etc which proved to be of a problem.



On march 20 2013 docker was born which intended to solve the problems that had risen from having a virtual machine. A docker is a software platform that facilitates a user to build, run and manage applications on a server and the cloud. it is done through virtualizing the operating system of the computer on which it is installed and running. This can be seen on the image below.

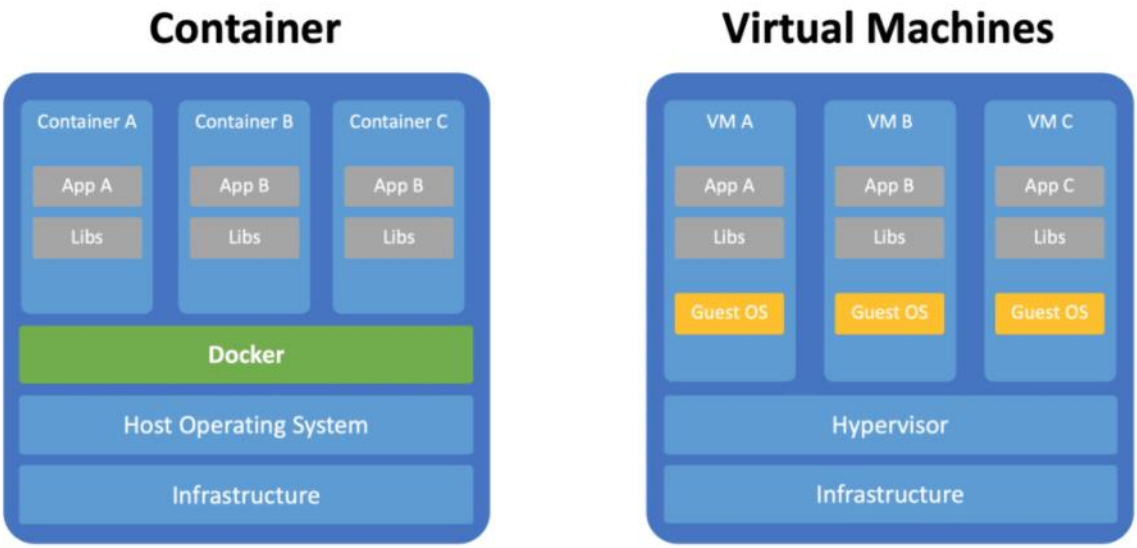


DOCKER VS VIRTUAL MACHINES (VM)

At this point you may be asking what the differences between docker are and Virtual Machines, well here are some key difference between the two technologies

1. Docker can run multiple applications with different requirements and dependencies on the same host, as long as they have the same OS system requirements while Virtual Machines can host multiple applications with different OS system requirements. Example you can run Linux based applications on docker, but you can run windows based application whereas on Virtual Machines you can run any OS.
2. Docker containers consume very little disk space since they are few megabytes in size Virtual Machines consumes large amount of disk space and other resources since each application has its own dedicate OS and other resources.
3. A docker doesn't have an OS, it runs on top of the host OS hence it is consuming very little memory while a Virtual Machines each application would have its dedicated OS which would lead to the consumption of huge amount of memory once each application is booted up
4. Docker is less demanding when it comes to operating it not compared to Virtual Machines

Below is a graphical representations on the difference between a docker and a virtual machine



DOCKER INSTALLATION

The installation of a docker is very simple and straight forward. We are going to learn how to install docker on Debian and install docker on Kali Linux

INSTALLING DOCKER ON KALI LINUX

Let's start by installing **docker.io** where we initially update our system, then install docker and finally enabling it so as it can be active and usable in the command line.

```
kali@kali:~$ sudo apt update
kali@kali:~$
kali@kali:~$ sudo apt install -y docker.io
kali@kali:~$
kali@kali:~$ sudo systemctl enable docker --now
kali@kali:~$
kali@kali:~$ docker
kali@kali:~$
```

Adding a user to the docker group to enable them to use docker just type in the below commands

```
kali@kali:~$ sudo usermod -aG docker $USER
kali@kali:~$
```

Then you can finally logout and login again in the system

To installing **docker -ce** in Kali Linux just follow the following steps:

Since Kali Linux is based on Debian, we start by installing a current Debian stable version as shown below

```
kali@kali:~$ printf '%s\n' "deb https://download.docker.com/linux/debian bullseye stable" |
  sudo tee /etc/apt/sources.list.d/docker-ce.list
```

Import the gpg key

```
kali@kali:~$ curl -fsSL https://download.docker.com/linux/debian/gpg |
  sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker-ce-archive-keyring.gpg
```

Finally install the latest version of docker -ce

```
kali@kali:~$ sudo apt update
kali@kali:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io
```

The difference between docker -io and docker -ce is that docker-io package is still the name used by Debian/Ubuntu for the docker release provided on their official repos while docker-ce is a certified release provided directly by docker.com and can also be built from source. The main reason for using the name docker-io on Debian/Ubuntu platform was to avoid a name conflict with docker system-tray binary.

INSTALLING DOCKER ON DEBIAN

As we discussed initially there are different ways on which we can install docker, in this section we will be installing docker directly from docker.com. In order to install docker on Debian here are the steps that you need follow

Uninstall old versions

The first step is to start by uninstalling the older versions of docker which were called docker, docker.io, or docker-engine. If these dockers are installed, then you need to uninstall them by running the following commands:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

Installing using the repository

Before getting into the installing of docker for the first time you need to have docker repositories, which docker can be installed and updated from.

a. setting up the repositories
    1. Update the apt package index and install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
     lsb-release
```

2. Add docker official pgp key

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

3. Use the following command to set up the stable repository. To add the nightly or test repository, add the word nightly or test (or both) after the word stable in the commands below. Learn about nightly and test channels.

```
$  echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Installing docker engine

1. First, we need to install the latest version of docker

```
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Then we need to install the test image by typing

```
$ sudo docker run hello-world
```

DOCKER USEFUL COMMANDS

There are many commands that can be used when working with docker images and its containers but in this article, we will be focusing on the commands that you will be frequently using while interacting with docker images and containers. Here are some of the commands:

1. The first step is to install an application on your docker which we call images, now to do that we need

```
$ sudo docker pull <name of image>

Example

$ sudo docker pull Debian
```

2. To list all the images that we have installed we can use the command

```
$ sudo docker images
```

3. To remove a docker image

```
$ sudo docker rmi  <docker container id>

To force remove docker image

$ sudo docker rmi -f <docker container id>
```

4. List docker running process

```
$ sudo docker ps
```

5. Starting a docker

```
$ sudo docker start <docker container id>
```

6. Stopping a docker

```
$ sudo docker stop <docker container id>
```

7. Specify a name for the container

```
$ sudo docker run -t -d - -name <name your container> <docker image>

Example

$ sudo docker run -t -d - -name test-environment  Debian
```

8. Get the terminal of the running container

```
$ sudo docker exec -it <name of the docker container> <sh or /bash or /bin/bash>

Example

$ sudo docker exec -it test-environment sh

$ sudo docker exec -it test-environment bash

$ sudo docker exec -it test-environment /bin/bash
```

9. Bind a port of your host to a port of the container while creating a container:

```
$ sudo docker run -p <host-port>:<container-port> <image>

Example

$ sudo docker run -p 80:80 Debian
```

10. View container logs: (you can also stream the logs using -f option)

```
$ sudo docker logs <container-id/name> -f
```

References

Conroy, S. (2018, January 25). *Uncomplicating Technology*. Retrieved from Uncomplicating Technology: https://www.idkrtm.com/history-of-virtualization/

Denning, P. J. (2001). *ieeexplore.ieee.org.* Retrieved from ieeexplore.ieee.org: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=948908

Docker. (n.d.). *Docker*. Retrieved from Docker: https://docs.docker.com/engine/install/debian/

gamb1t, e. (2021, December 14). *Kali Linux*. Retrieved from Kali Linux: https://www.kali.org/docs/containers/installing-docker-on-kali/

Shahzan. (2019, APril 18). *Free code camp*. Retrieved from Free code camp: https://www.freecodecamp.org/news/docker-simplified-96639a35ff36/#:~:text=In%20simple%20terms%2C%20Docker%20is,it%20is%20installed%20and%20running.

Shilpa. (2019, September 9). *Reverie*. Retrieved from Reverie: https://kriyavikalpa.com/2019/09/09/docker-introduction/

Shilpa. (2021, january 9). *Reverie*. Retrieved from Reverie: https://kriyavikalpa.com/2021/01/09/docker-deep-dive/