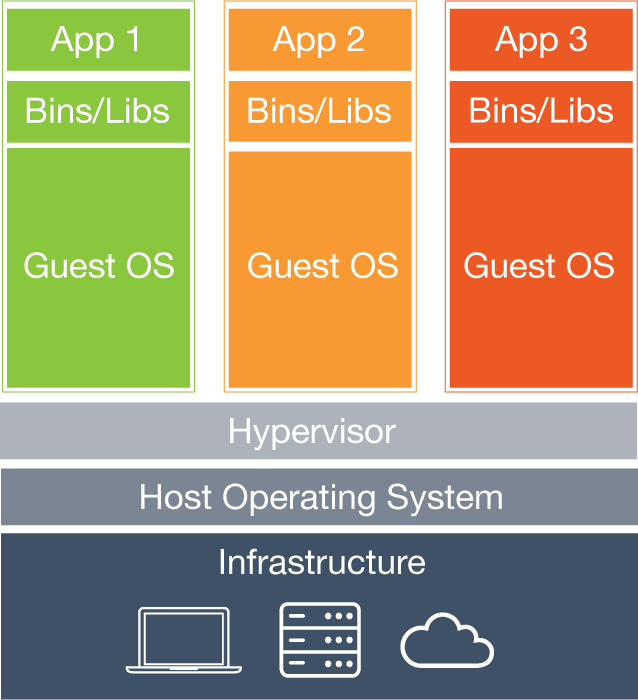
We develop application in our machine. But when we run it on another machine we have to set up the environment and all its dependencies. Although an application runs on the built environment, it would not run on the another. It wastes the time setting up the environment. What is the solution???? Docker is the solution for that. So the idea behind the docker is that it enables developers to create the applications to from components that can be run on laptops, servers or the the cloud. Docker is a open source platform that can manage the containers which run on Linux. It is a way to build and ship the applications in containers.

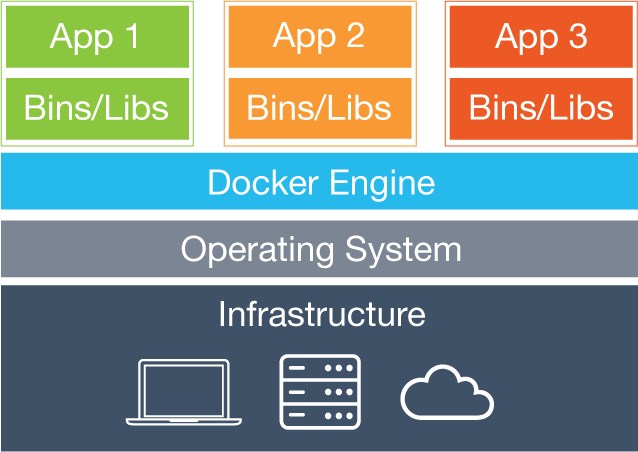
Did you hear a new word? Containers??? What is it...

*“Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They’re also not tied to any specific infrastructure – Docker containers run on any computer, on any infrastructure and in any cloud. ”*

*“Docker allows to package an application with all dependencies into a single unit for software development.”* This is the description of docker in Docker.com. This unit includes everything that the software need to run, code, run time,system libraries and anything that can install on a server. Therefore docker guarantees that the software will run the same, regardless of the environment it is running on. Docker has emerged as a leading container program with support and partnerships from major linux open source powers such as Canonical, Red Hat an Amazon.

When we move to Docker, virtualization is a vary important point. In the traditional virtualization we used virtual machines. Docker is bit like virtual machine. But it is not a a actual virtual machine. Shall we look the difference between docker vs virtual machines.

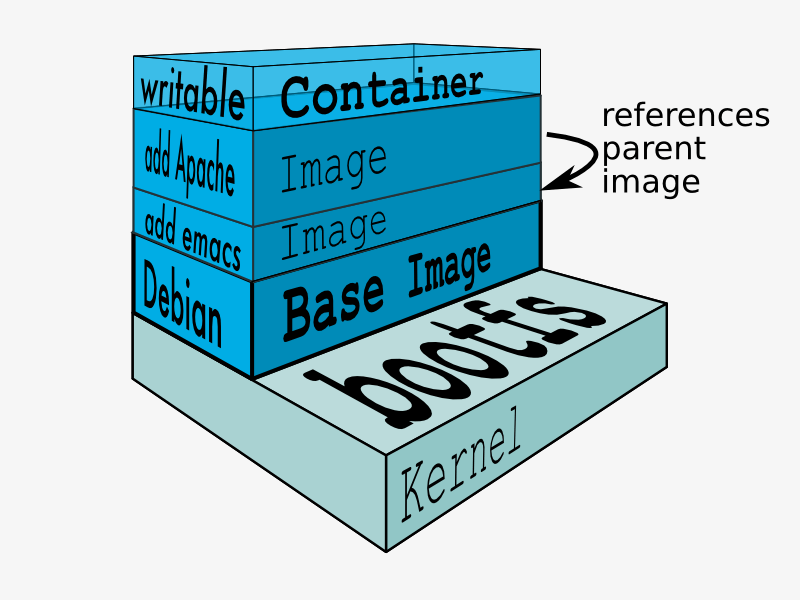




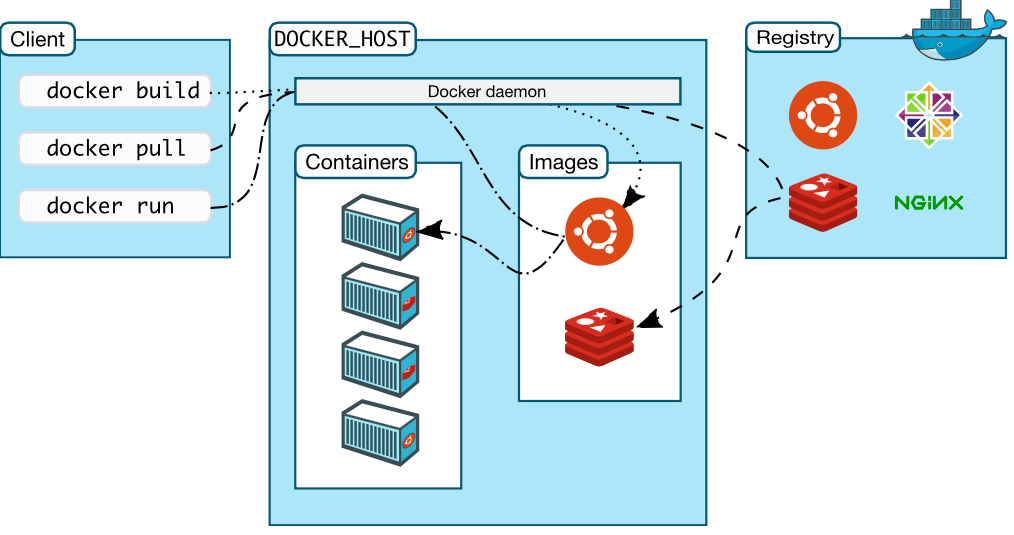
Virtual Machines creates a whole virtual operating system. Docker allows applications to use the same linux kernal as the system that they are running on. Docker containers run as isolation processes while sharing the the kernal with other containers.

Basic Docker Components

* Docker client
* Images
* Registry



* Containers



First we have to understand about image. Docker images are the templates from which containers are created. In other word docker containers are the runtime instances of docker Images. Docker images are consists of layers, and typically created with Docker files that include the instructions for building the image and stored in a docker registry that may be private or public. Docker Hub is the registry by Docker.

Actually, docker clients are the person who use the docker. We are the docker clients. Docker clients run the docker commands.

**Getting started with Docker**

If you have basic idea about docker now you can get start with docker. By following this blog you will learn the fundamental Docker features. These are the basics : How to

* install Docker
* run a software image in a container
* browse for an image on Docker Hub
* create your own image and run it in a container
* create a Docker Hub account and an image repository
* create an image of your own
* push your image to Docker Hub for others to use

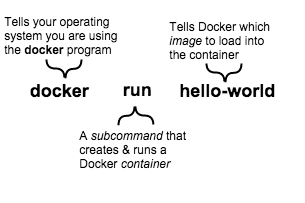
**install Docker**

$ wget -qO- https://get.docker.com/ | sh

The system prompts you for your sudo password. Then, it downloads and installs Docker and its dependencies.

Verify docker is installed correctly.

$ docker run hello-world



If Docker has been installed correctly then image hello-world will download from the docker hub and run.

**run a software image in a container**

Let's run a docker image in a container.

Command :

docker run Image Name

* + e.g :- docker run docker/whalesay cowsay boo

here cowsay boo is a command

***“cowsay*** *is a program which generates* [*ASCII*](https://en.wikipedia.org/wiki/ASCII) *pictures of a cow with a message.**[[2]](https://en.wikipedia.org/wiki/Cowsay" \l "cite_note-2) It can also generate pictures using pre-made images of other animals, such as Tux the Penguin, the Linux mascot.”*

\_\_\_\_\_

< boo >

-----

\

\

\

## .

## ## ## ==

## ## ## ## ===

/""""""""""""""""\_\_\_/ ===

~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ / ===- ~~~

\\_\_\_\_\_\_ o \_\_/

\ \ \_\_/

\\_\_\_\_\\_\_\_\_\_\_/

A container is a stripped-to-basics version of a Linux operating system. An image is software you load into a container. When you run the command, the Docker software checks whether you have or not the image. If you have not then download the image from the Docker Hub and load the image into the container and run it.

When you run the “docker run docker/whalesay cowsay boo” command you will get the above out put. But when we run an image at its' first docker will download from the registry. The docker will run the image.

When you run the command at very first, you will see this output in the terminal.

It means the image is not locally available. Therefore Docker should download the image from the Docker Hub to your computer. So image will pull from the **docker.io/library/docker/** which is the domain name for the registry.

**Unable to find image 'docker/whalesay:latest' locally**

**Pulling repository docker.io/library/docker/whalesay**

Docker downloads the image to your computer. This local copy of the image saves your time. Docker only downloads the image again if the image’s source changes on the hub.

**create your own image and run it in a container**

We browse images in the docker hub and ran them. Shall we create our own image and run. As the first step to create a docker image we should write a docker file. Let's try an example.

Write a Dockerfile and write these lines :

**FROM docker/whalesay:latest**

**RUN apt-get -y update && apt-get install -y fortunes**

**CMD /usr/games/fortune -a | cowsay**

In this file,

the FROM keyword tells Docker which image your image is based on

Then, Docker installs the new fortunes software.

Finally, Docker finishes the build and reports its outcome.

To build the Docker File you should run the following command.

$ docker build -t docker-whale .

This will follow the steps in docker file and build the docker image. Yeah! Now you have your own docker image. Shall we run it. So run the following command.

$ docker run docker-whale

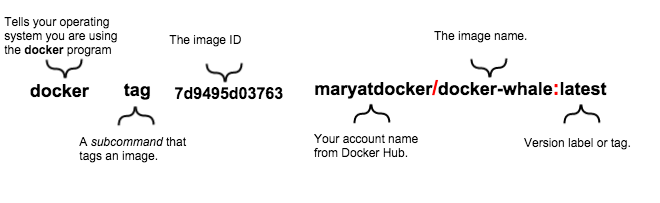
Now you are done. :)

Then create a Docker Hub account and Registry.

**Tag, push, and pull your image**

**Link :**

After creating a docker registry in docker Hub you can store your images in it and others can use those images.



As an example :

$ docker tag 908ac9e2f833 nadeeshani/docker-whale:latest

Here 908ac9e2f833 is the image id and nadeeshani is the account name of the Docker Hub. When we run the above command it will create a copy of the image which the image id is 908ac9e2f833.

Now you can push the images to Docker Hub and Pull the images from it. But first you should login into the Docker Hub using command line.

**$ docker login –username=(account name of Docker Hub) --password=password --email=usrname@gmail.com**

As an example,

$ docker login --username=nadeeshani --password=1234 --email=pnadeeshani65@gmail.com

Now you have logged into the Docker Hub through the terminal. Now you can push images to your account.

**$ docker push yourusername/docker-whale**

$ docker push nadeeshani/docker-whale

Now check the account from browser and you will see the docker-whale image in your account. Now you can pull the images to your computer too.

**$ docker pull yourusername/docker-whale**

$ docker push nadeeshani/docker-whale

After going through this, you should have good understanding of fundamental docker features. Here are some important docker commands.

Link: Docker Commands

**Create Your Own Registry**

Are you familiar with Docker? Did you have built and run docker images? OK. Then did you familiar with the tag, push, pull commands that are used to store and retrieve docker images. If yes you can go easily through this with a better understanding.

If not first familiar with Docker(Link) and fundamentals of docker (Link).

When we push and pull images normally we use the docker Hub because when we use the **docker/whalesay** it refers to **docker.io/docker/whalesay** repository. We don't use docker.io. B ut it is the domain name for the Docker Hub. When we run **docker/whalesay,** this cpmmmand points to the docker Hub repositories. But it is not always good to keep images in another repository. We may need our own registry to store images. Then we don't want to access the docker hub.

This registry also an image. So we can run this image with the ports that map to the host port and the container port.

Docker run registry..

Now you can store images in that registry using push and pull commands.

First tag an image with the domain name of your registry.

Docker tag 98888886887 localhost:5000/ubuntu

Now you can push images to the registry.

docker push localhost:5000/ubuntu

when we run this command, ubuntu image will be push into the rigistry with tha domain name localhost:5000. If you want you can pull the images from registry too.

Docker pull localhost:5000/ubuntu