DOCKER

By ANKIT SHARMA

Docker

Docker is a software which provides centralized platform to execute your application. It wraps software component into a complete standardized unit which contain everything require to run. Whether it is code, runtime environment,tool or libraries. It guarantees the software will always run as expected.

Docker is an Open platform for developing,shipping and running applications, Docker enable you to separate your application form your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your application. By taking advantage of docker's methodologies for shipping,testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

History of Virtualization

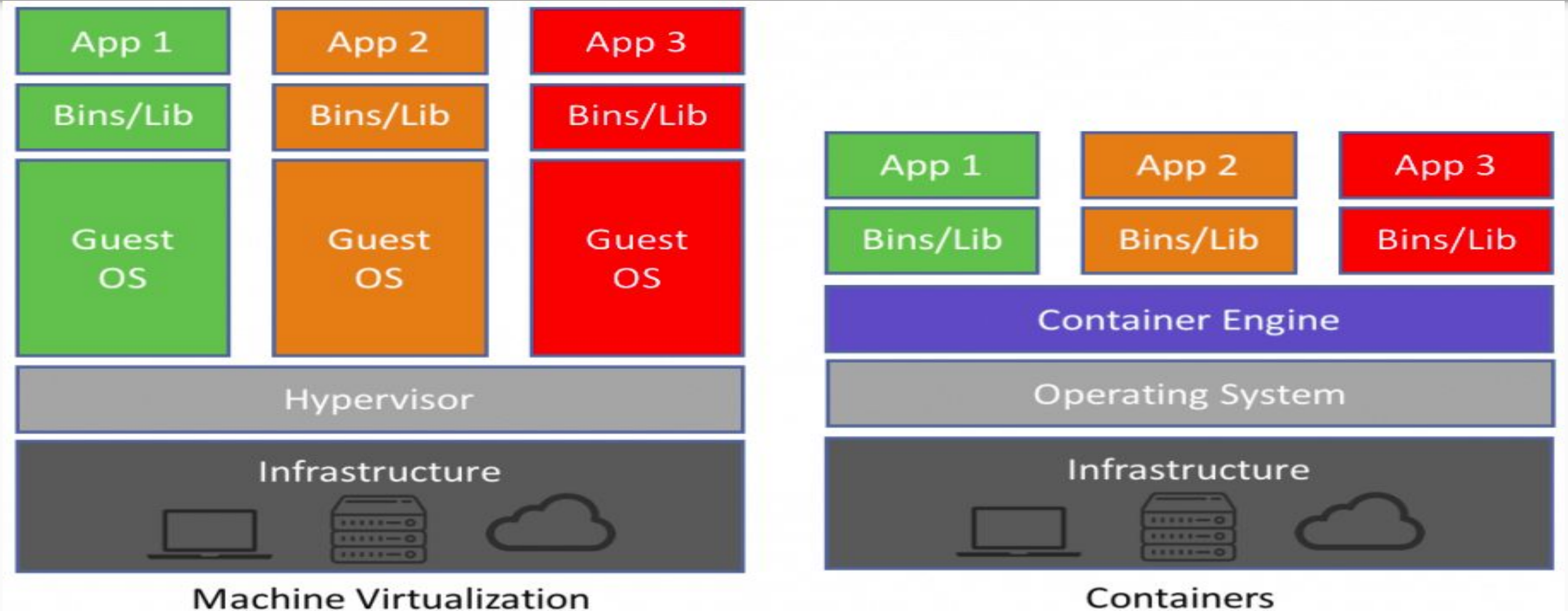
Earlier, the process for deploying a service was slow and painf ul. First,the developers were writing code, the operations team would deploy it on bare metal machines, where they had to look out for library versions,patches and language compilers for the code to work.if there were some bugs or errors, the process would start all over again, the developers would fix it, and then again the operational team was there to deploy.

There was an improvement with the creation of ***hypervisors***. ***Hypervisors*** have multiple ***virtual machines*** or *VMs* on the same host, which may be running or turned off. *VMs* decreased the waiting time for deploying code and bugs fixing in a big manner, but the real game changer was ***docker containers.***

What is Docker ?

Docker is computer software used for virtualization in order to have multiple operating system running on the same host. Unlike *Hypervisors* which are used for creating VMs,*virtualization* in Docker is performed on system-level in so-called *Docker* .containers. As you can see the difference in the image in the ***next slide,*** *Docker containers* run on the host’s operation system. This helps you to improves efficiency. Moreover, we can run more containers on the same *infrastructure* than we can *Virtual machines* because containers use fewer resources. Unlike the VMs which can *communicate* with the hardware of the host (ex: *Ethernet adapter* to create more virtual adapters) *Docker containers* run in an isolated environment on top of the host's OS. Even if your host runs Windows OS, you can have Linux images running in containers with the help of *Hyper-V,* which automatically creates small VM to virtualize the system's base image, in this case, Linux.

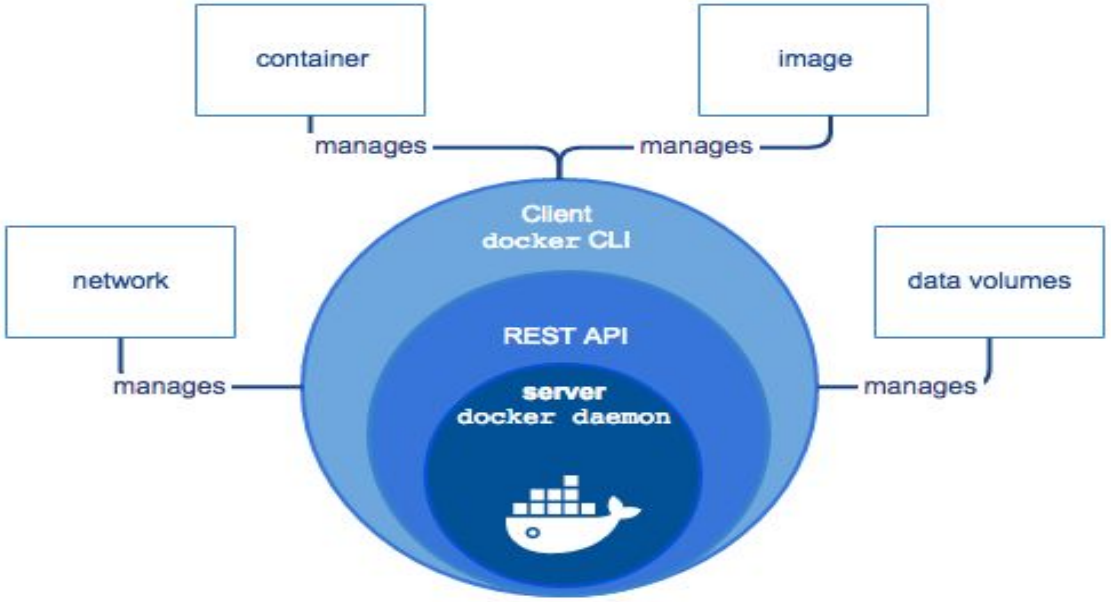
Virtualization VS Containers



Docker Engine

It is a client server application that contains the following major components.

- A server which is a type of long-running program called a daemon process.
- The REST API is used to specify interface that programs can use to talk to the daemon and instruct it what to do.
- A command line interface client.



Docker Features

- Easy and Faster configuration
- Increase productivity
- Application Isolation
- Swarm (*It is a clustering and Scheduling tool for Docker containers*)
- Routing Mesh
- Services
- Security Management

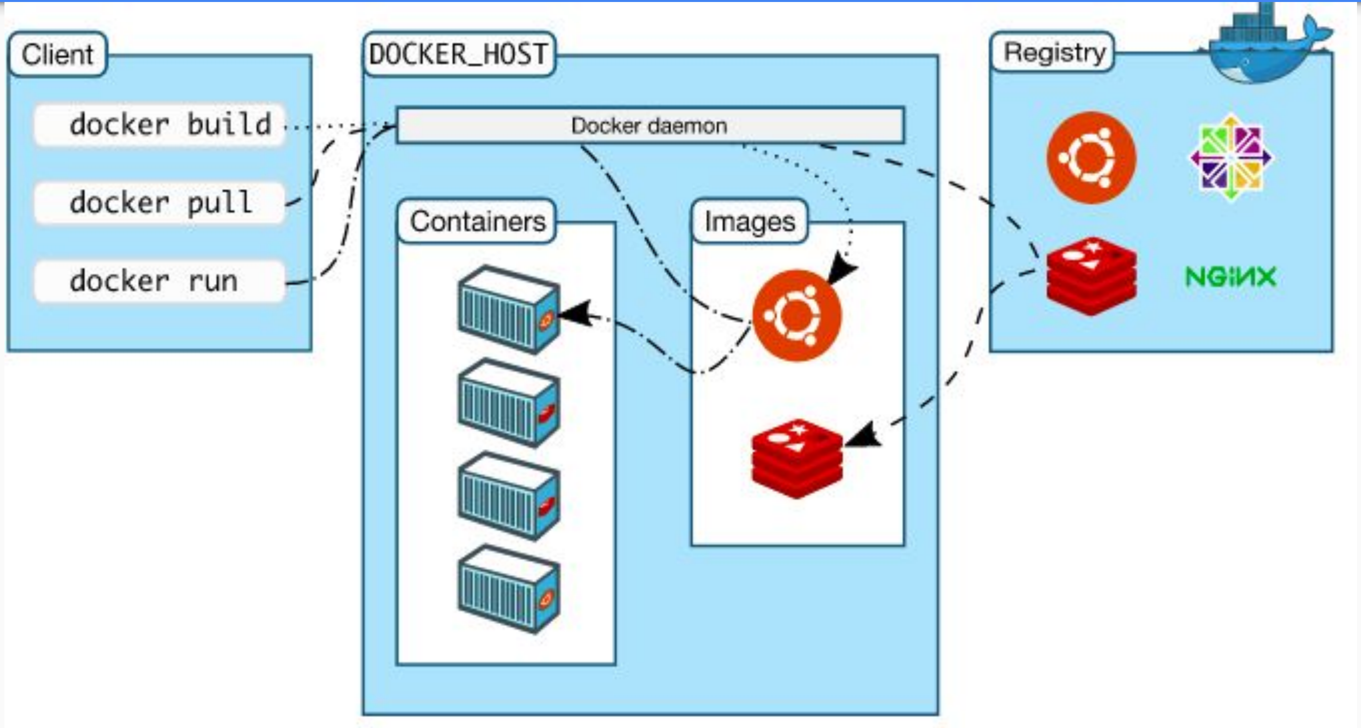
Docker Architecture

1) ***<u>Client</u>*** **: Docker provides command Line Interface (CLI) tool to client to interact with *Docker Daemon*. Client can build,run and stop application . Client can also interact to docker_Host remotely.**

2) ***<u>Docker_Host</u>*** **: It contains containers,image and docker daemon. It provides complete environment to execute and run your application.**

3) ***<u>Registry</u>*** **: It is global repository of images. You can access and use these images to run your application in Docker environment.**

Docker Architecture



Docker Architecture: *Docker daemon*

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 Docker Client : The Docker client (Docker) is the primary way that many Docker users interact with *Docker*. When you use commands such as Docker run , the client sends these commands to dockerd , which carries them out. The *docker* command uses the Docker API. The Docker client can communicate with more than one *daemon*.

Docker architecture : *Docker registries*

A *docker registry* store *docker Images. Docker Hub* is a public registry that anyone can use and Docker is configured to look for images on *Docker Hub* by default.You can ever run your own private registry. If you use *Docker Datacenter (DDC)*, it includes *Docker Trusted registry (DTR).*

When you use the *docker pull* or *docker run* commands , the required images are pulled from your configured registry. When you use the *docker push* command, your images is pushed to your configured registry.

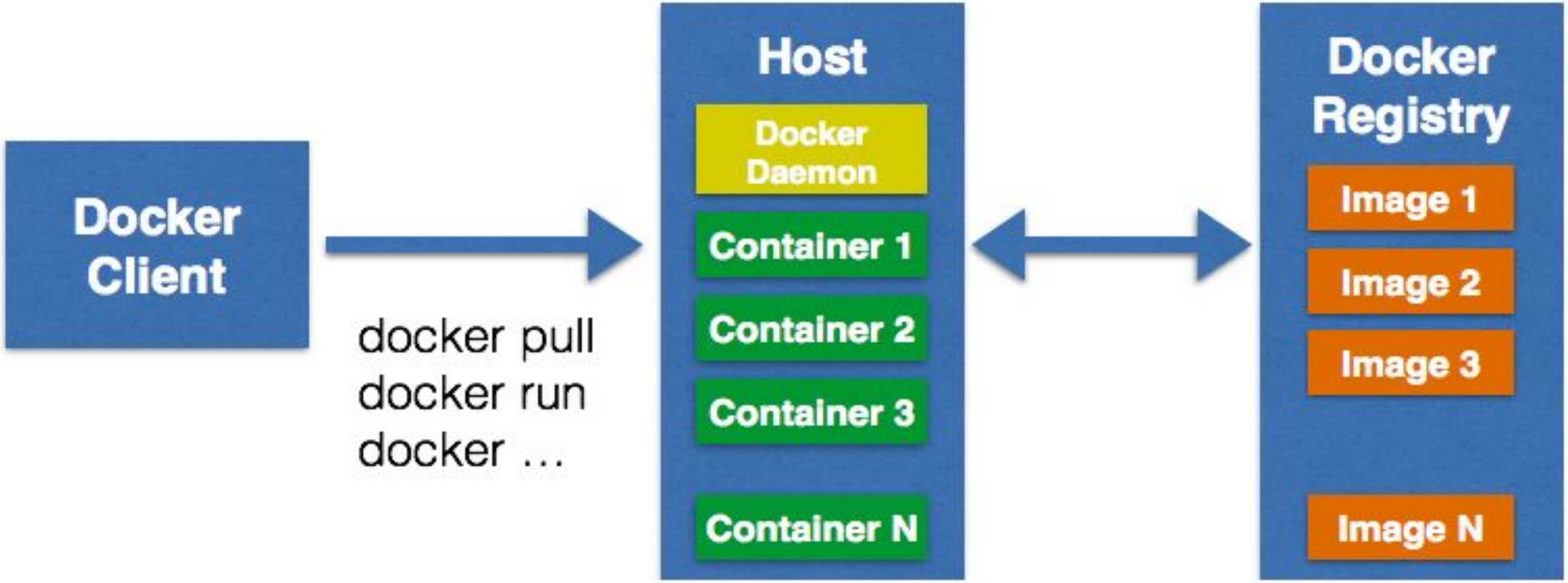
Docker Object : When you use Docker, you are creating and using images, containers, networks , volumes , plugins and other objects.

IMAGES

An *images* is a read-only template with instructions for creating a *Docker container*. Often, an image is based on another image, with some *additional customization*. For example , you may build an image which is based on the *ubuntu* images, but installs the *Apache web server* and your application, as well as the configuration details needed to make your application run.

You might create your own *images* or you might only use those created by others and published in a *registry*. To build your own *images*, you own *images*, you create a *Dockerfile* with a simple syntax for defining the step needed to create the image and *run it*. Each *instruction in a Dockerfile create a layer in the image*. When you change the *Dockerfile* and *rebuild the image*, only those layer which have changed are *rebuild the image*, only those layer which have *changed* are rebuilt. *This is part of what makes images so lightweight, small and fast when compared to other virtualization technologies.*

Docker Architecture Diagram



Dockerfile

A Dockerfile is a text document that contains commands that are used to assemble an image.we can use any command that call on the command line. Docker build images automatically by reading the instruction from the dockerfile.

```
#This is a sample Image

FROM ubuntu

MAINTAINER ********@gmail.com

RUN apt-get update

RUN apt-get install –y nginx

CMD [“echo”,”Image created”]
```

Docker Useful Commands

1. Check Docker version : *$docker version*
2. Build Docker Image from a dockerfile : *$docker build -t image-name docker-file-location (-t is used to tag docker image with provided name).*
3. Run Docker image : $ *docker run -d image-name (-d used to create a daemon process)*
4. Check available Docker Images : *$docker image*
5. Check all running containers : *$docker ps -a*
6. Check all latest running container : *$docker ps -l*
7. Stop running container : *$docker stop container_id*
8. Delete an image : *$docker rmi image-name*
9. Delete all image : *$docker rmi $(docker images -q)*
10. Delete all image forcefully : *$docker rmi -r $(docker images -q)*
11. Delete all containers : *$docker rm $(docker ps -a -q)*
12. Enter into Docker container : *$docker exec -it container-id bash*

Docker Cloud

Docker provides us the facility to store and fetch docker images on the cloud registry. We can store dockerized images either privately or publicly. It is a full GUI interface that allows us to manage builds, images, swarms, nodes and apps.

We need to have Docker ID to access and control images. If we don't have, create it first.

Docker Compose

It is a tool which is used to create and start Docker application by using a single command.
We can use it to file to configure our application's services.
It is a great tool for development, testing, and staging environments.
It provides the following commands for managing the whole lifecycle of our application.

- Start, stop and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

To implement compose, it consists the following steps.

1. Put Application environment variables inside the Dockerfile to access publicly.
2. Provide services name in the docker-compose.yml file so they can be run together in an isolated environment.
3. run docker-compose up and Compose will start and run your entire app.

Summary

- Earlier, the process for deploying a service was slow and painful but, VMs decreased the waiting time for deploying code and bug fixing in a big manner
- Docker is computer software used for Virtualization in order to have multiple Operating systems running on the same host
- Docker is the client-server type of application which means we have clients who relay to the server
- Docker images are the "source code" for our containers; we use them to build
- Docker has two types of registries 1.) public and 2)private registries
- Containers are the organizational units of Docker. In simple terms, an image is a template, and a container is a copy of that template. You can have multiple containers (copies) of the same image.

docker