Docker Basics

# Docker basics

In DevOps era Every organization wants to streamline their deployment strategy. To streamline deployments, the center of attraction is IT infrastructure to be precise different kind of servers

use to maintain application.

Every Organization needs their application should be scalable. But the term scalable may be easy to say but believe me it is very hard to implement.

**What do we mean by scalable?**

Suppose a project maintain images, in a normal day it receives 10,000 requests. Project has a load balancer under the balancer it maintains two pools.

Pool A and Pool B. Pools are act as Active/Passive mode. For a given moment one pool is active

Each pool contains say 5 web servers.

Each webserver can take maximum 10000 request.

So Total capacity of this project

5\*10,000=50,000 requests at given moment.

I can say this project can handle 50,000 requests at a time.

But consider a situation say in upcoming RIO Olympics there will be more traffic than a normal day

After all this project hosted Images.

Say in Rio Olympic this projects got 1,000000 (lakhs) requests, what happens then?

Obviously servers are gone down as it can maximum handle 50000 requests for a moment so every moment there will 50000 extra requests in pool and another 100000 comes so next moment it has to serve 150000 requests. So requests in queue increase in exponential order so system will down eventually.

**How to manage such situation?**

**There is two ways to handle such situation**

1. **Vertical scaling: That is make your server a super computer so it can serve many more requests. But Cost of Super computer is Very high most of the organization can’t effort it.**
2. **Horizontal scaling: Another way is add many more commodity servers under load balancer so load balancer can distribute requests.**

**Second one look promising right…**

But in Current Infrastructure the problem is to add new server to the pool. To add the servers you need to configure that server that is install OS , add necessary software’s, add network configuration, add firewall, update load balancer etc.

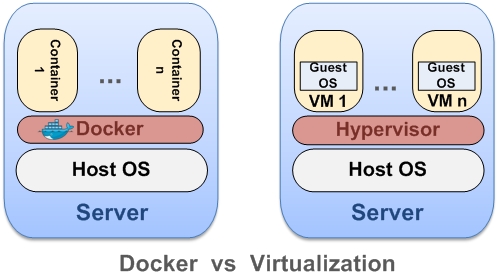
Which takes much more times even if you have golden image still it takes time.

**So the Solution is Docker.**

***Docker is nothing but a slick container. It runs just above the Operating systems and access your hardware but it can spawn multiple containers and Each container act as separate m/c and each has separate address spaces each has separate guest OS with minimum requirement to hosting a software, so If a software needs one WebLogic server so a container only contains a Guest OS with WebLogic server.***

***Docker is faster than Virtual machine (VM) as Virtual machine contains one or more guest OS which run on top of your machine OS and in between there is a Hypervisor layer which manages guest OS calls or it’s requirements and change that requirements in to your machine specific OS call. Mainlyuse for your OS and Guest OS communication.***

***Unlike VM, Docker just run on top of your OS it shares your OS network, hardware everything but Docker container maintains a separate OS which has its own address space and only contains require software to work with. So Docker are very light-weight and easy to spawn later you can see by a command you can spawn a container so Horizontal Scale up is very easy***

******

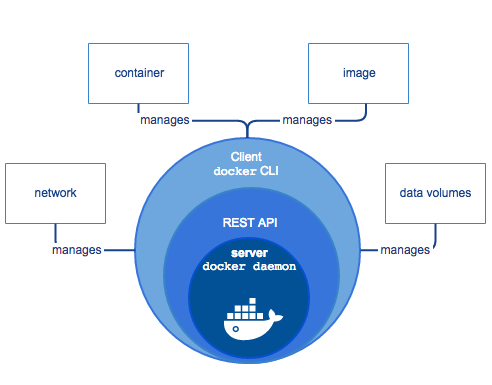
***Difference between Docker and VM***

**Docker Architecture:**

**Docker contains Three main parts**

Docker maintains client server architecture where Docker client talks with Docker daemon via socket or Restful API. Docker daemon and client can be in same host ore Docker daemon can deployed on a different Host. If so Docker client has to communicate with Remote Docker daemon.

1. **Docker Daemon:** Docker Daemon does most of the work according to Docker client or Restful API commands. According to command it builds image, spawn a container, run a container, can push an image etc.
2. **Restful API:** Docker publish Restful API so if you want to control Docker daemon through program you can call this API and controls Docker daemon.
3. **Docker client:** Docker client is a CLI (command Line Interface) where you can fire command then it talks to Docker daemon and perform the task for you. You can consider it as Linux terminal which talks to kernel.

****

**Docker Architecture – Picture Courtesy: Docker Site**

**To understand How docker works you need to know some Docker terms**

1. **Docker images:** Docker images is nothing but a logical template based on this Docker container spawns. So as the Project states above it need as Image which has a OS say Ubuntu 13.04 with a WebLogic server. So an image can be created with this requirement.

Image can be created locally using Docker file or can be pull from a global repository (If it is exists) which is managed by Docker. we call it Docker Hub or registry.

1. **Docker Hub/Registry:** Docker hub contains images which can be pull to satisfy your requirements. In Docker hub there can be OS images and can be Hybrid images such as Solaris with Tomcat. One can push its own image to Docker Hub to do it you need Docker account.

Docker Hub is an central repository you can have thought it as an maven Repository. And like GIT you can push your image in to it.

1. **Docker Containers**: Docker containers is runtime environment from an image a runtime can spawn you can though it as a separate machine.

**Some important commands:**

***To download a Image from Docker Hub:***

In Docker client type

***Docker pull <image-name>*** //that is name of the image

***To spawn a container and execute a command:***

***Docker run -i -t <image-name> <command name>***

***Docker run -I -t ubuntu13.03 ls***

***To Push an image in Docker Hub:***

***Docker push <image-name>***

***I will discuss more command in next Docker section where I will guide you to set up a Docker container.***