|  |
| --- |
|  |
| Explore Docker |
| Basics :: Installation: POC |
|  |
| **Vibranarayanan** |
| **1/1/2018** |

|  |
| --- |
|  |

Table of Contents

[Change Log 2](#_Toc514263179)

[Docker Terms 2](#_Toc514263180)

[Docker work flow 3](#_Toc514263181)

[Virtualization Vs Containerization 3](#_Toc514263182)

[Containers on VM 4](#_Toc514263183)

[Docker Client-server architecture 4](#_Toc514263184)

[Docker Benefits 5](#_Toc514263185)

[Portability 5](#_Toc514263186)

[Version Control 5](#_Toc514263187)

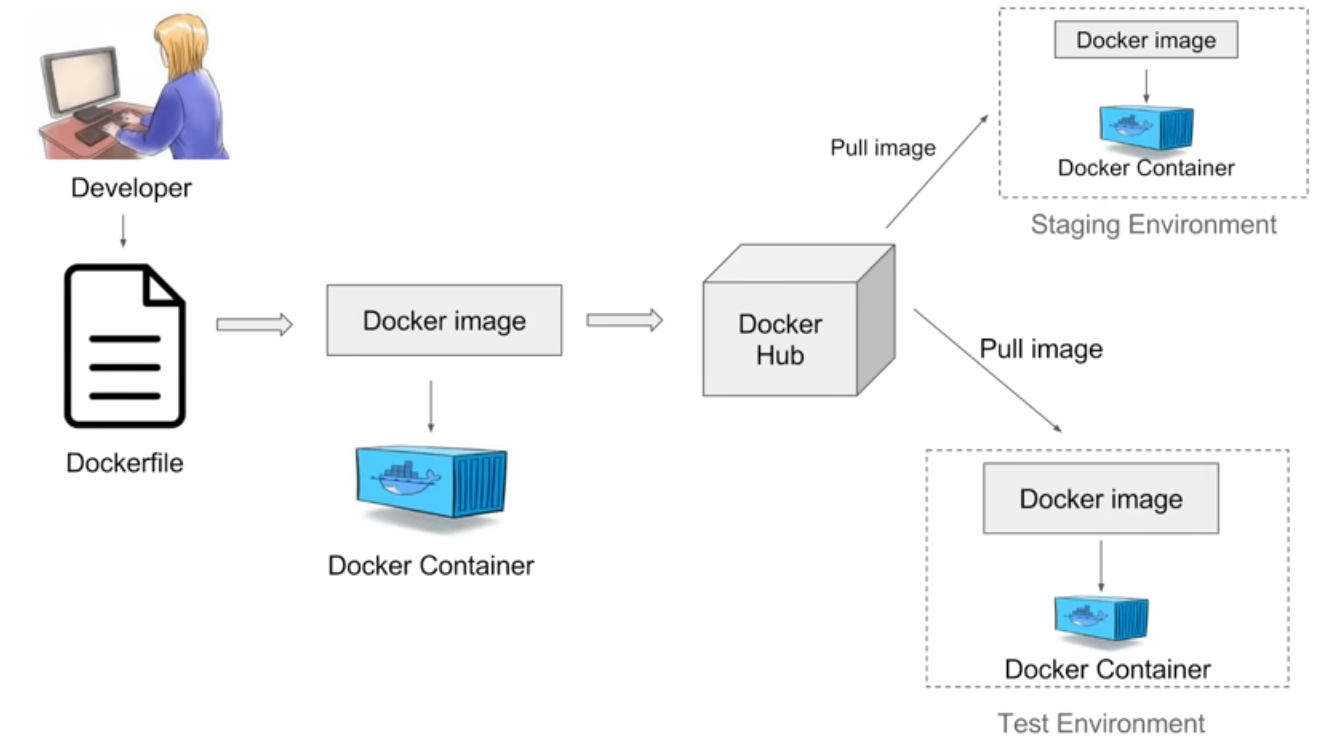
## Change Log

|  |  |
| --- | --- |
| Date | Details |
| 05/16/2018 | * Docker terms * Docker work flow * Virtualization vs Containerization * Client server architecture * Benifites |
|  |  |

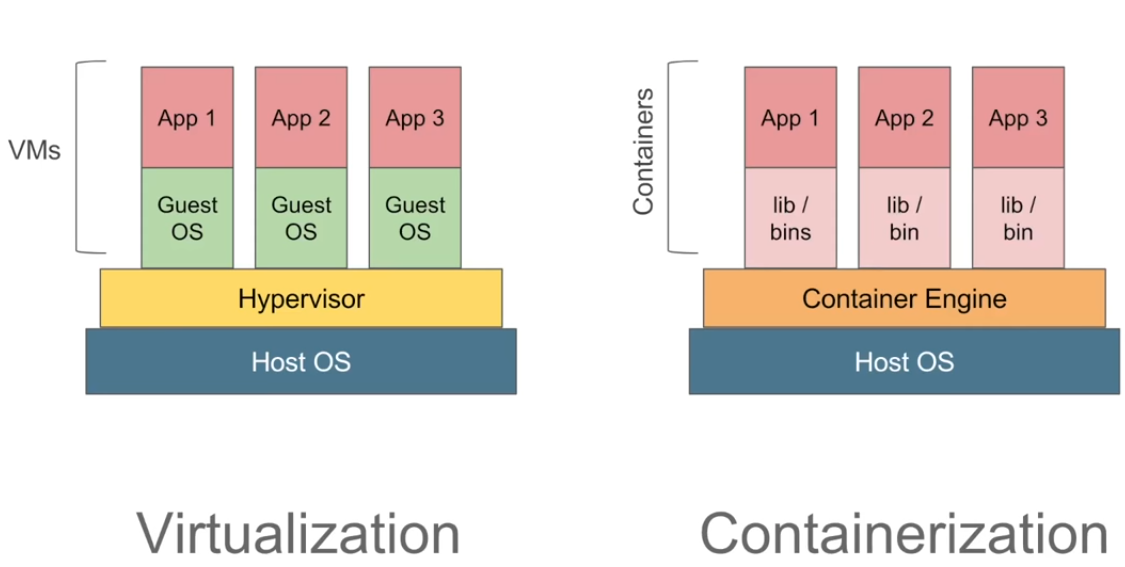
## Docker Terms

|  |  |
| --- | --- |
| Term | Explanation |
| Docker file | Describes steps to create Docker image |
| Docker image | * Created using Docker file * contain application requirements and its dependencies |
| Docker container | * Created by running Docker image. * Run time instances of Docker image. |
| Docker Hub | * Online repository, images can be stored in this repository |
| Docker Client | * Command line inter face to interact with Docker server. |
| Docker Server/Daemon | * Will have all the containers |
| Docker Engine | * Combination of Docker Client and server component. |

## Docker work flow



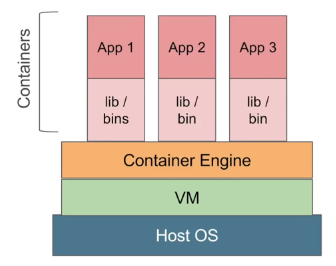
## Virtualization Vs Containerization



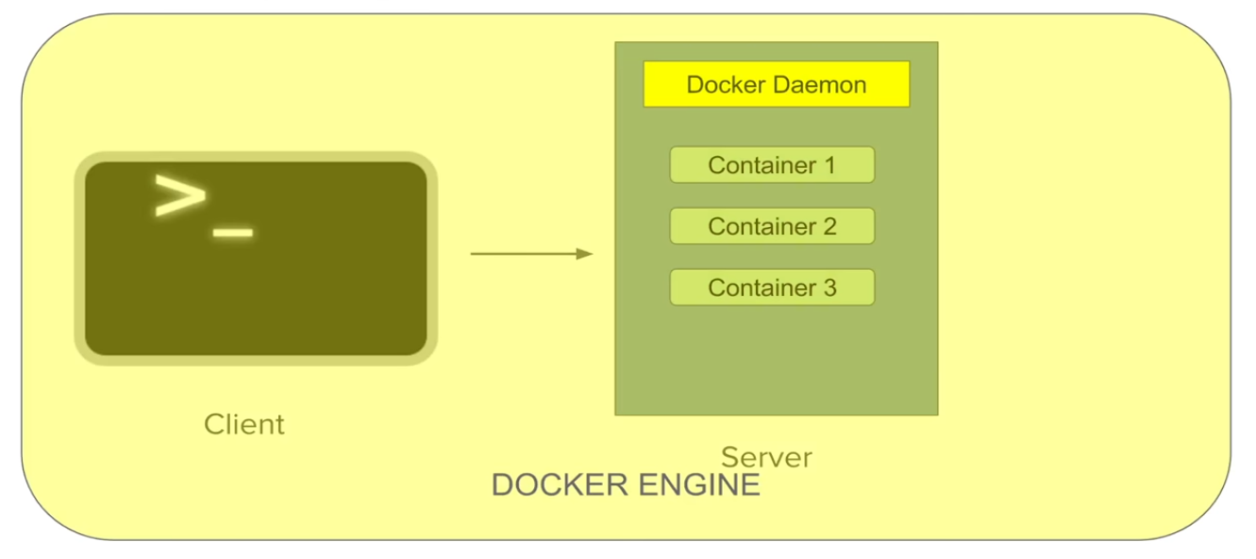
|  |  |
| --- | --- |
| **Virtualization** | **Containerization** |
| * Create Multiple VMs using Hypervisor in Host OS * VMs have their own OS and does not use host OS. * Overhead on Host platform * Each VM fixed memory need to be allocated. this leads to wastage of memory. | * Containers are light weight * Container engine will manage/use Host OS configurations/memory based on applications in containers. * No memory/space overhead. |

## Containers on VM

There may be a scenarios like container need to be run on VM. In that case Containers can be managed in below stacks.



## Docker Client-server architecture



* Docker server/Docker Daemon receives command from client in this form of CLI or rest API.
* All the Docker Client and server together form Docker engine.
* Docker client or Docker Daemon can be present in same host or in different host.

## Docker Benefits

* An Application inside a container runs on any system that Docker installed.
* Build application only once and no need to configure multiple times.
* Docker images can be maintained in any repository , later images can be pulled for usage.
* Test your application inside your container and ship it inside the container, This means Environment in which you test is identical with the app in production.
* *Isolation* is the Key, with Docker every application works in isolation in its own container. Does not interferes with other application running on the same system.
* Removal of an application is easy by deleting a container.
* Developer can package software with all its dependency and Docker will take care if running those application in different platform.
* *Productivity* is an another key. Docker allows faster and efficient deployment without worrying about, application running on different planform.

### Portability



* Docker containers can run on any platform. This can run on local system and Amazon ec2 or Google Cloud, etc..
* Container running on AWS can easily be ported to Virtual machine.

### Version Control

* Like Git Docker has in-build version control.
* Docker containers work just like a GIT repo, allow you to commit your changes into Docker images and version control them.
* Docker images can be tracked in other version control system like GIT.