docker

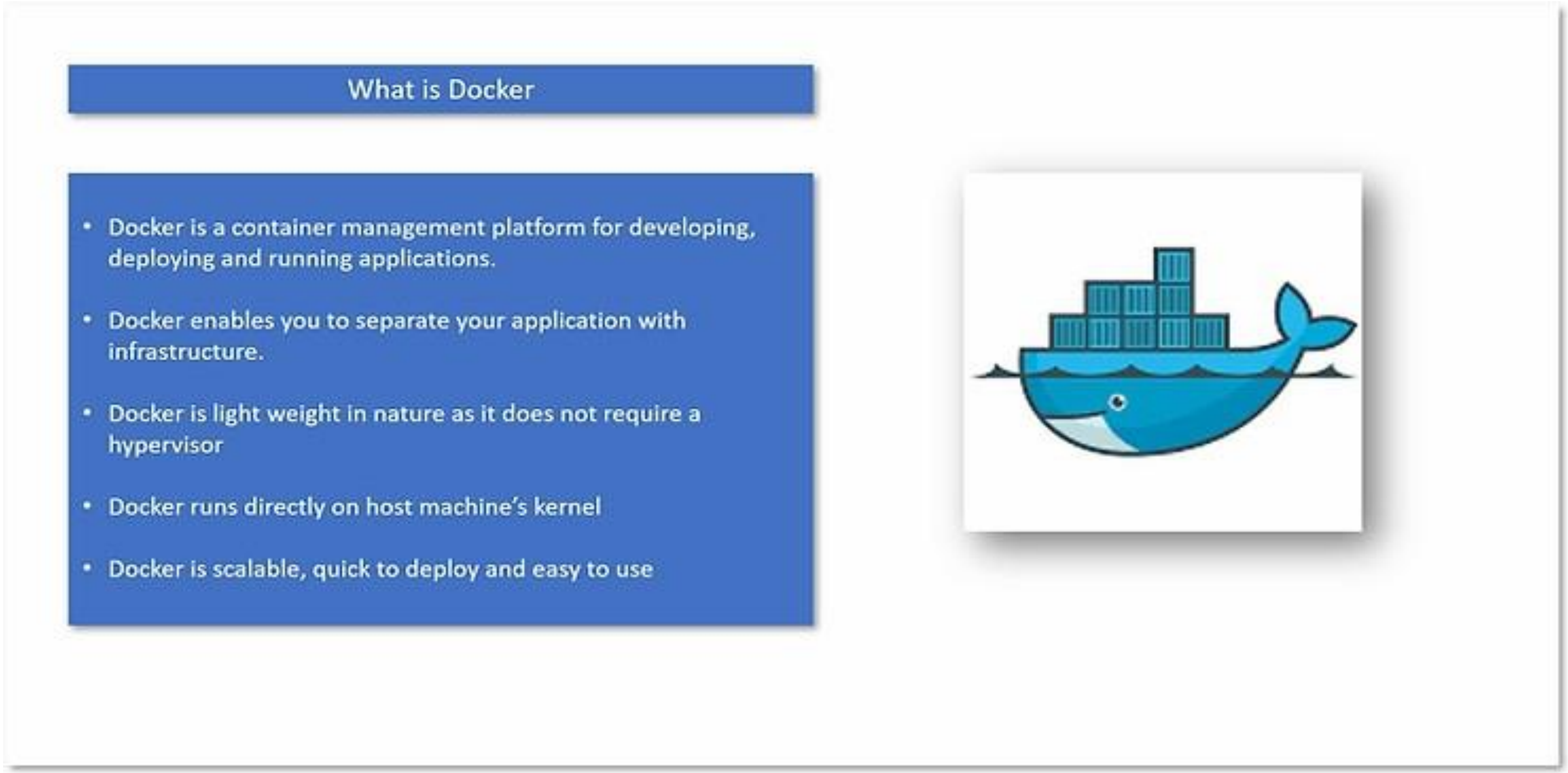
Mithun Technologies

Mithun Technologies
devopstrainingblr@gmail.com

Program Agenda

- Introduction
- Virtualization and Containerization
- Virtual Machine and Docker
- Installation
- Dockerfile, Docker Image and Docker Container
- Docker Commands
- Build the Dockerfile
- Create Docker Custom images and Containers, push to Docker Hub
- Docker Compose
- Docker Swarm

Know Docker

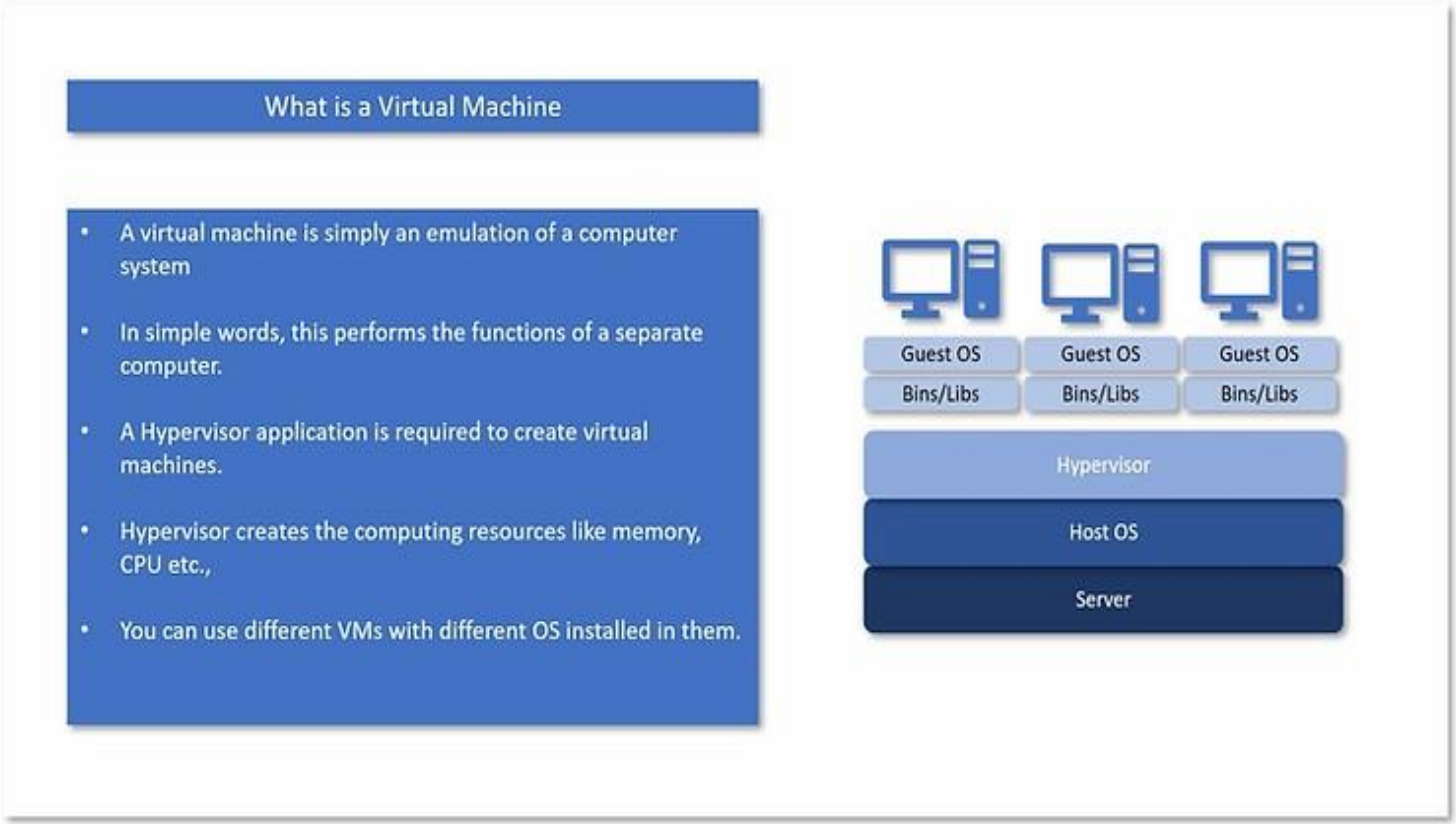


What is Docker

- Docker is a container management platform for developing, deploying and running applications.
- Docker enables you to separate your application with infrastructure.
- Docker is light weight in nature as it does not require a hypervisor
- Docker runs directly on host machine's kernel
- Docker is scalable, quick to deploy and easy to use

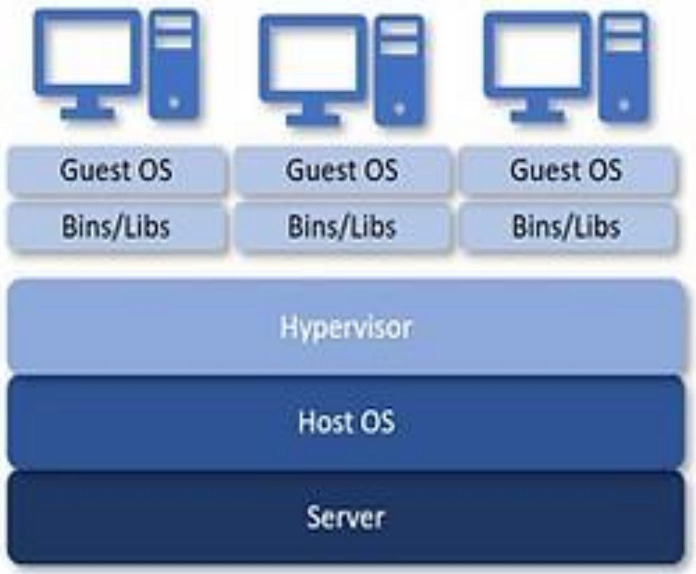
- ❖ *Docker is a containerisation platform which packages your app and its all dependencies together in the form of containers. so as to ensure that your application works seamlessly in any environment be it dev or test or prod.*
- ❖ *Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications*
- ❖ *Docker is available in two editions: Community Edition (CE) and Enterprise Edition (EE).*

To understand Docker better, we need to know what Virtual Machine is and how it works.

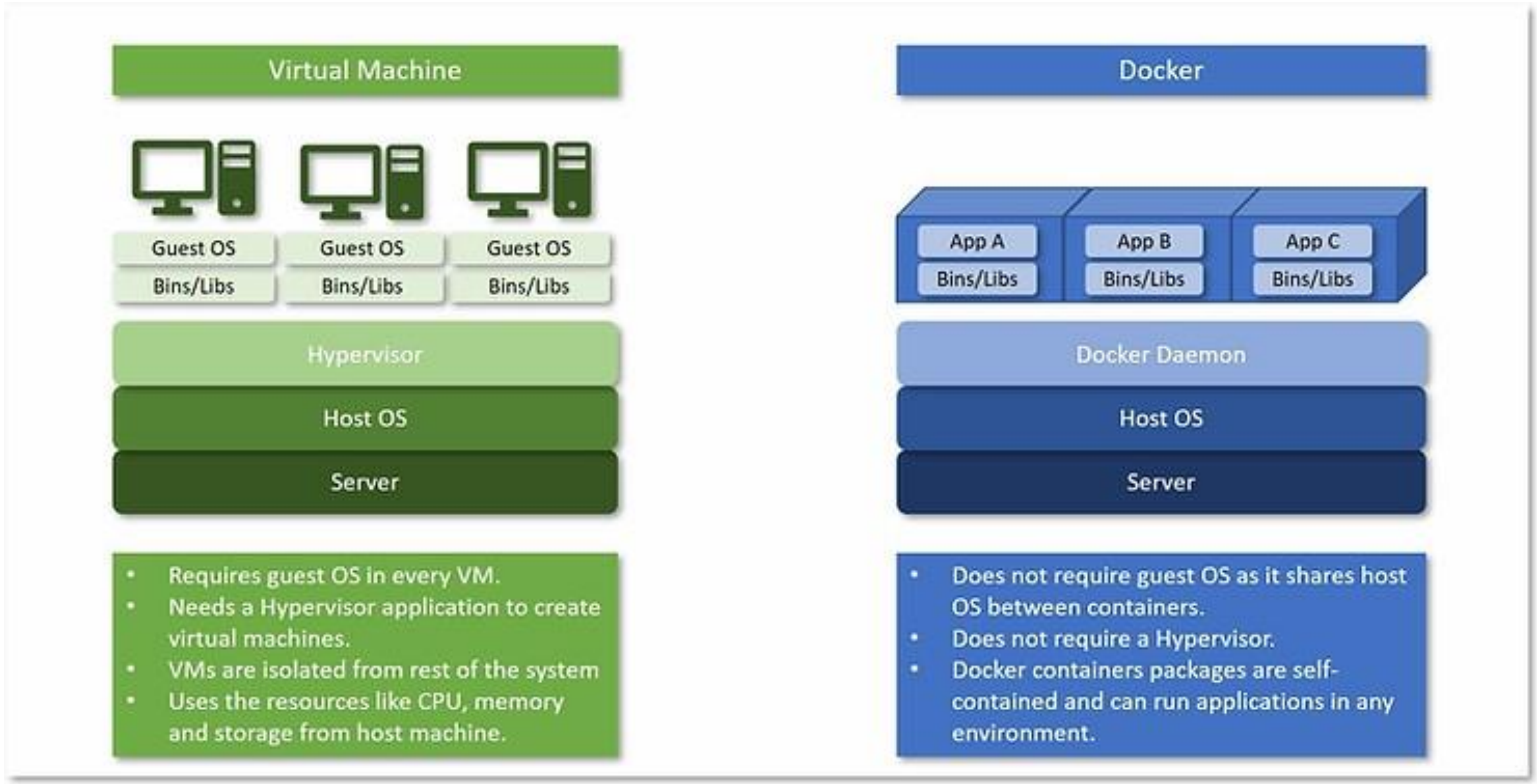


What is a Virtual Machine

- A virtual machine is simply an emulation of a computer system
- In simple words, this performs the functions of a separate computer.
- A Hypervisor application is required to create virtual machines.
- Hypervisor creates the computing resources like memory, CPU etc.,
- You can use different VMs with different OS installed in them.



Know How Virtual Machine and Docker works

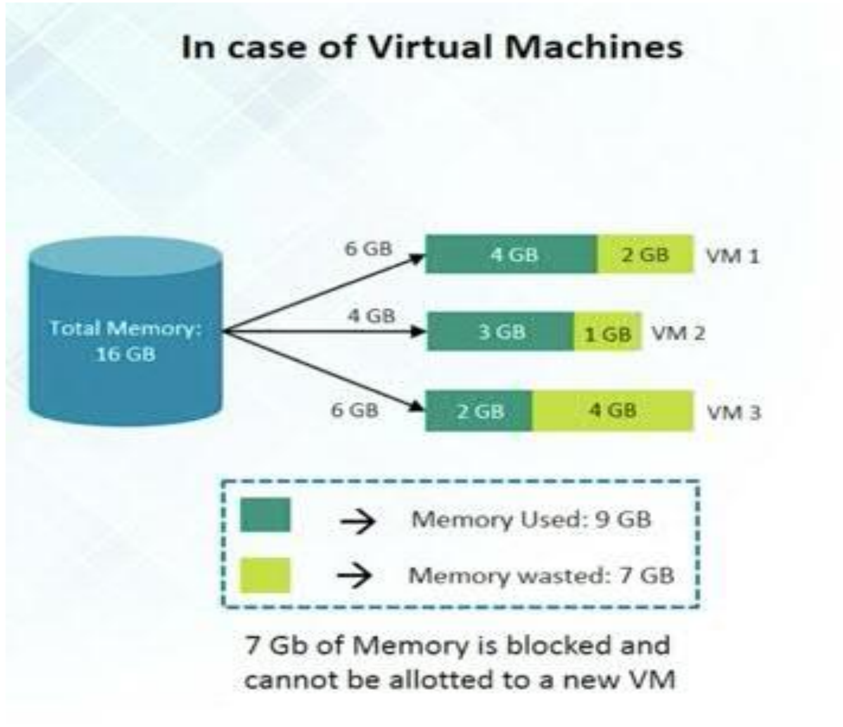


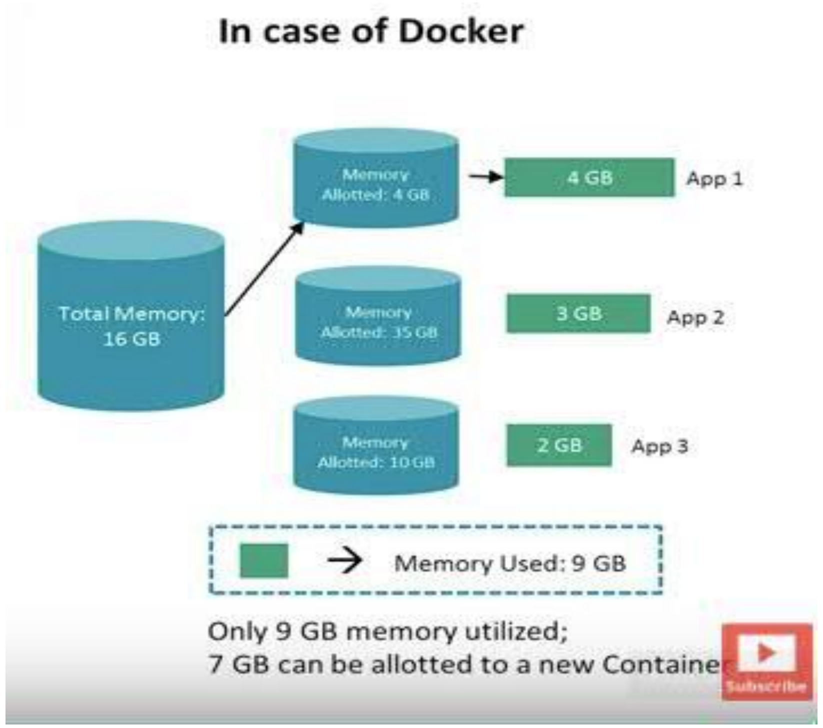
- Requires guest OS in every VM.
- Needs a Hypervisor application to create virtual machines.
- VMs are isolated from rest of the system
- Uses the resources like CPU, memory and storage from host machine.

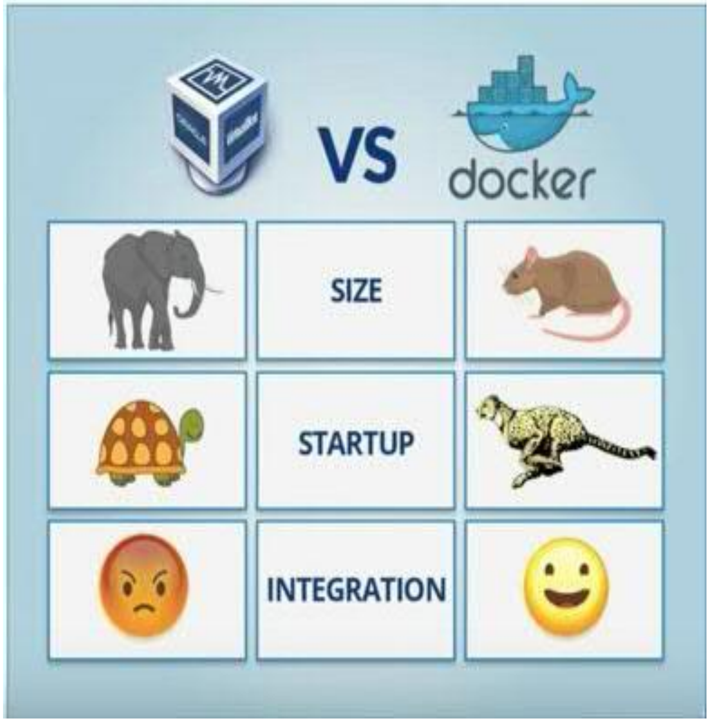
- Does not require guest OS as it shares host OS between containers.
- Does not require a Hypervisor.
- Docker containers packages are self-contained and can run applications in any environment.

Mithun Technologies

Virtual Machines vs. Containers







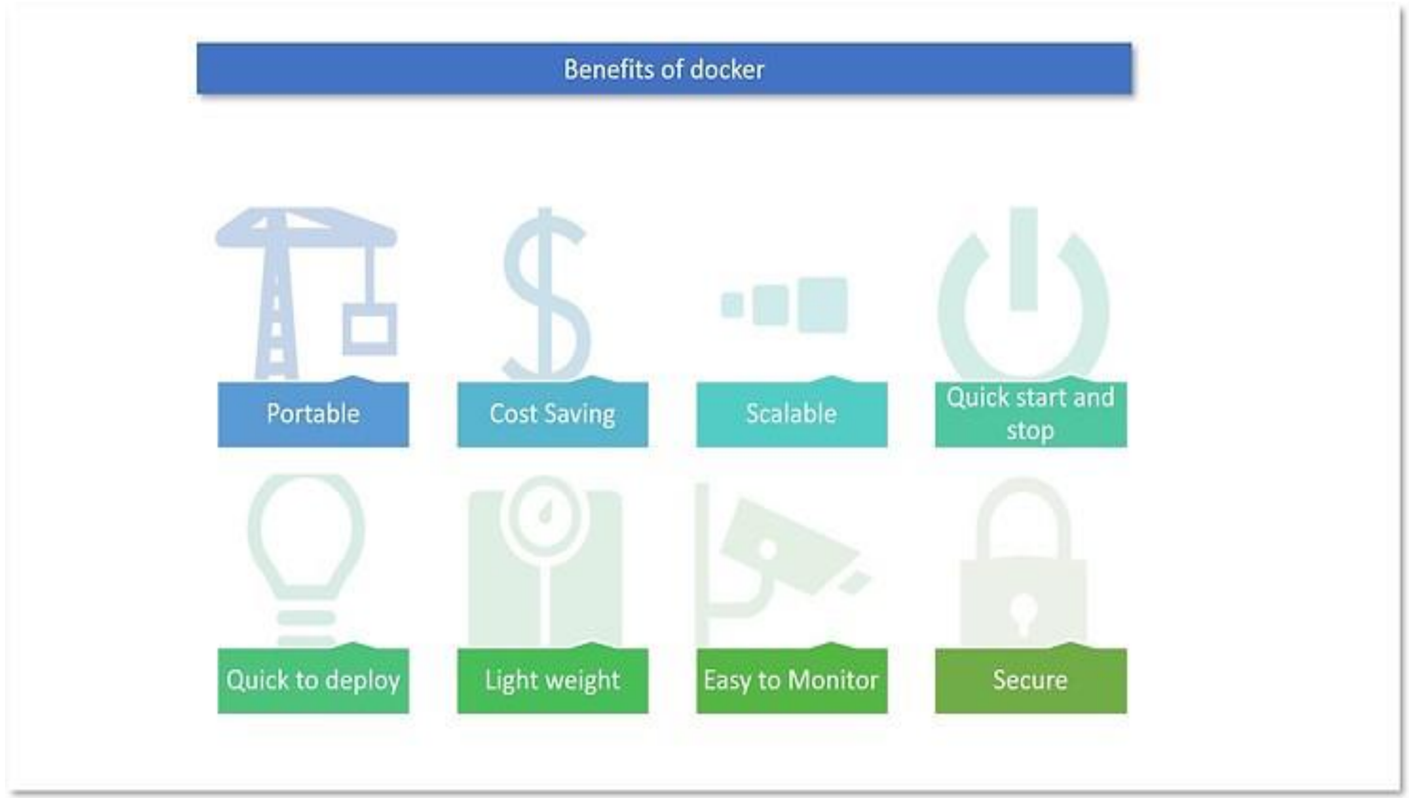
Virtual Machines vs. Containers

Features	Virtual Machines (VMs)	Application Containers	Application Container Benefits
Virtualization	In VMs, the virtualization occurs in server hardware.	In containers, the virtualization occurs at the OS level.	Faster time-to-market
Standardization	VMs are standardized systems that have capabilities similar to that of bare metal computers.	Containers are not standardized, as their kernel OS has varying degrees of complexity.	Makes applications more portable by isolating them from the underlying infrastructure
Resource management	Each VM has its own kernel OS, binary, and library.	Containers share the same host OS, resources and Kernel.	Lower overhead costs
Density	VMs require a few gigabytes (GB) of space that limits them into a single server.	Containers require a few megabytes (MB) of space that enables many containers to fit into a single server.	Lighter weight than VMs
Boot time	Minutes	Seconds	Easy to update and release newer versions of applications

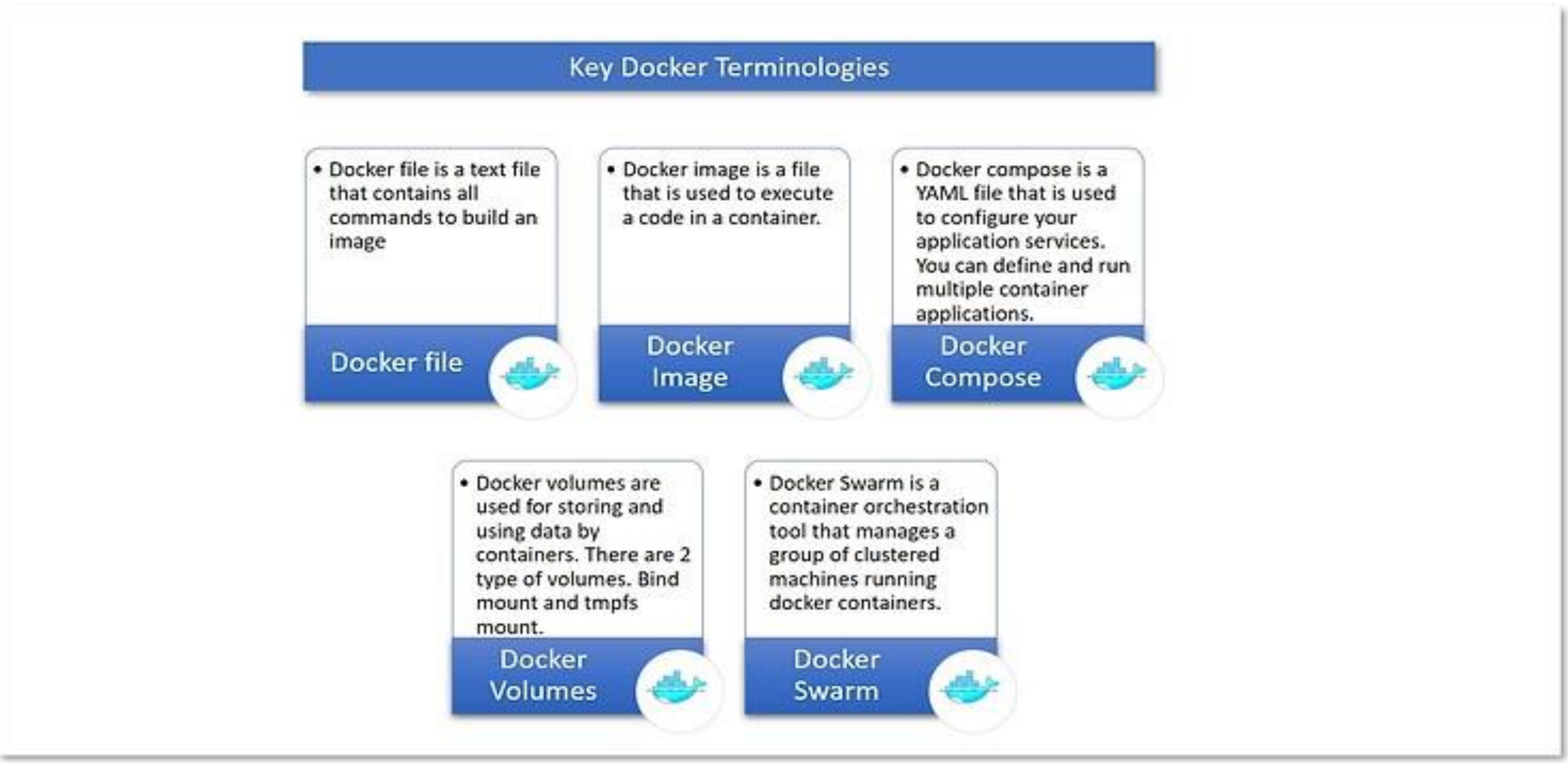
Type	Containerization Tool
Vendor	Docker
Is Open Source?	Yes – Some extent
Operating system	Cross Platform
URL	https://www.docker.com/

Download Docker and install from below url
https://www.docker.com/products/docker-toolbox

Register in https://hub.docker.com/



Docker Key Concepts



Key Docker Terminologies

Docker file
- Docker file is a text file that contains all commands to build an image

Docker Image
- Docker image is a file that is used to execute a code in a container.

Docker Compose
- Docker compose is a YAML file that is used to configure your application services. You can define and run multiple container applications.

Docker Volumes
- Docker volumes are used for storing and using data by containers. There are 2 type of volumes. Bind mount and tmpfs mount.

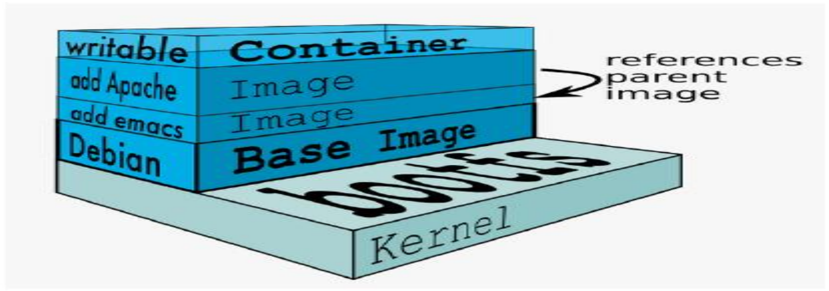
Docker Swarm
- Docker Swarm is a container orchestration tool that manages a group of clustered machines running docker containers.

Dockerfile

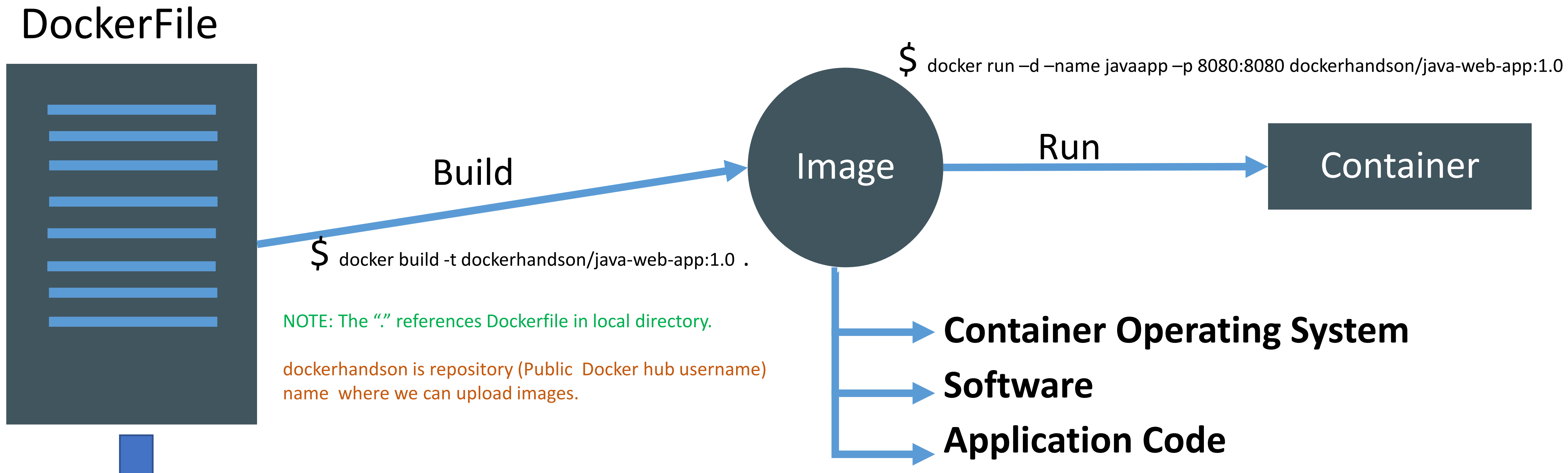
- Dockerfile is a file, it will describe the steps in order to create an image quickly.
- The Docker daemon runs the instructions in the Dockerfile are processed from the top down, so you should order them accordingly.
- The Dockerfile is the source code of the Docker Image.

Docker Image

- An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files.



Docker Flow

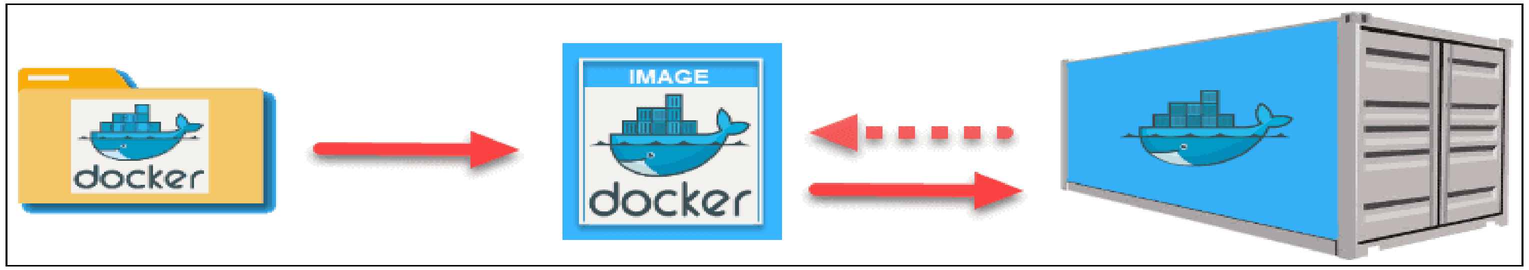


```
$ docker build -t dockerhandson/java-web-app:1.0  .
```

```
$ docker run –d –name javaapp –p 8080:8080 dockerhandson/java-web-app:1.0
```

NOTE: The "." references Dockerfile in local directory.

dockerhandson is repository (Public Docker hub username) name where we can upload images.



```
# Default name of docker file is Dockerfile
FROM tomcat:8.0.20-jre8
COPY target/java-web-app*.war /usr/local/tomcat/webapps/java-web-app.war
```

Why Containers?

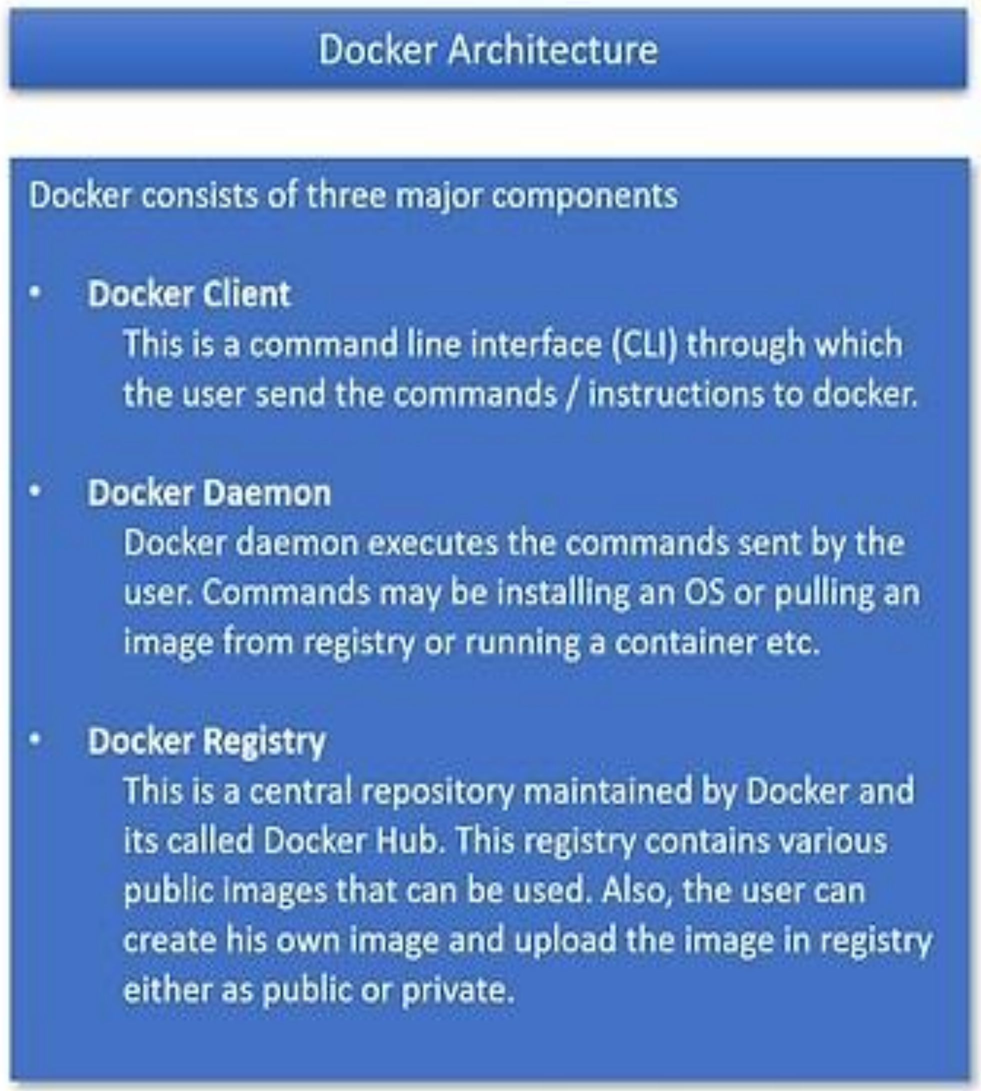
Developers care because:

- Quickly create ready-to-run packaged applications, low cost deployment and replay
- Automate testing, integration, packaging
- Reduce / eliminate platform compatibility issues (“It works in dev!”)
- Support next gen applications (microservices)

IT cares because:

- Improve **speed** and frequency of releases, reliability of deployments
- Makes app lifecycle efficient, consistent and repeatable – configure once, run many times
- Eliminate environment inconsistencies between development, test, production
- Improve production application resiliency and scale out / in on demand

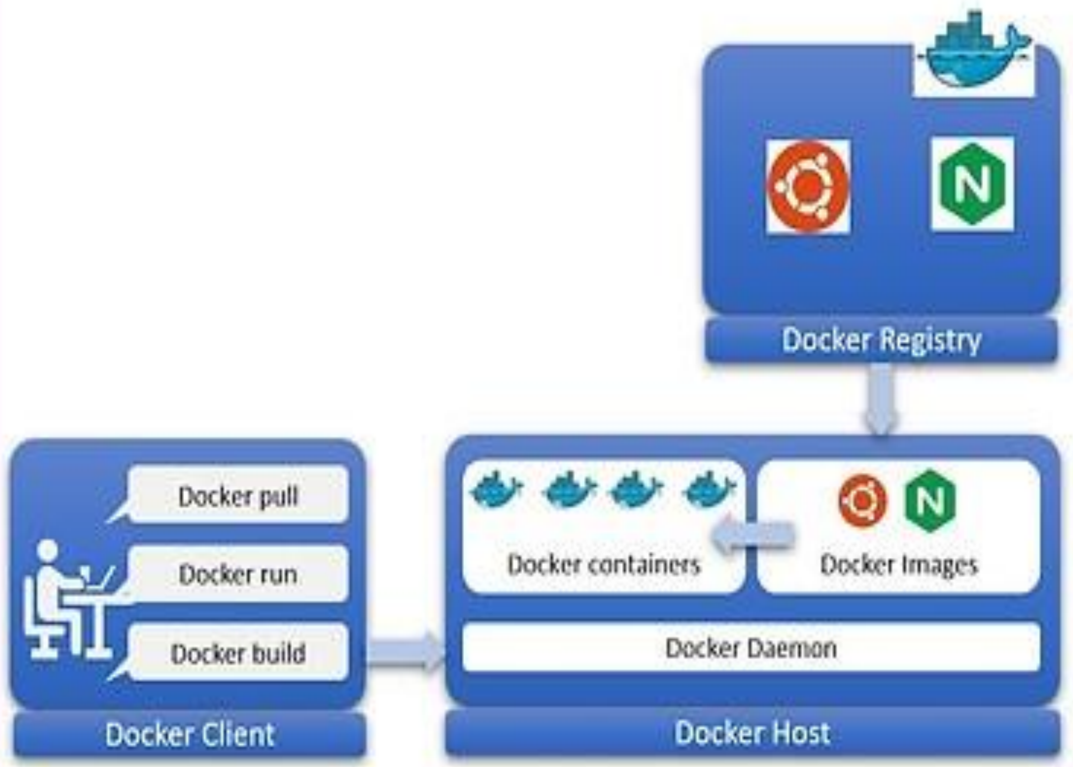
Architecture of Docker showing how it works



Docker Architecture

Docker consists of three major components

- **Docker Client**
  This is a command line interface (CLI) through which the user send the commands / instructions to docker.
- **Docker Daemon**
  Docker daemon executes the commands sent by the user. Commands may be installing an OS or pulling an image from registry or running a container etc.
- **Docker Registry**
  This is a central repository maintained by Docker and its called Docker Hub. This registry contains various public images that can be used. Also, the user can create his own image and upload the image in registry either as public or private.



Advantages of Docker

- ***Rapid application deployment*** *– containers include the minimal run time requirements of the application, reducing their size and allowing them to be deployed quickly.*

- ***Portability across machines*** *– an application and all its dependencies can be bundled into a single container that is independent from the host version of Linux kernel, platform distribution, or deployment model. This container can be transferred to another machine that runs Docker using it's docker image, and executed there without compatibility issues.*

- ***Version control and component reuse*** *– you can track successive versions of a images, roll-back to previous versions. Containers reuse components from the preceding layers(base images), which makes them noticeably lightweight.*

- ***Sharing*** *– you can use a remote repository to share your image with others. And it is also possible to configure your own private repository.*

- ***Lightweight footprint and minimal overhead*** *– Docker images are typically very small, which facilitates rapid delivery and reduces the time to deploy new application containers. Allowing for more efficient use of computing resources, both in terms of energy consumption and cost effectiveness.*

- ***Simplified maintenance*** *– Docker reduces effort and risk of problems with application dependencies. Multiple containerized apps on a single server don't mess up each other. If you update an app, you just build a new image, run fresh containers and don't have to worry about other ones on the machine breaking*

Advantages of Docker

- ***Self-sufficient***

  A Docker container contains everything it needs to run

  Minimal Base OS

  Libraries and frameworks

  Application code

  A Docker container should be able to run anywhere that Docker can run.

- ***Docker shines for microservices architecture.***

  NOTE: Containers aren't required to implement microservices, but they are perfectly suited to the microservices approach and to agile development processes generally.

Build & Deployment Process

To be more clear,
In 1990'S we had physical servers where apps are hosted and served.

In 2000'S we had Virtual Machines where apps are hosted and served ,running on top of the Physical servers. Virtualization started here.

In early 2013 the concept of Containerization picks up where we can package the application and its dependencies so that whatever a developer released will work regardless of which environment it runs under(DEV, QA & PROD).

Before Docker
Build

- A developer pushes a change to SCM tools.
- The CI sees that new code is available. It rebuilds the project, runs the tests and generates a application package(like jar/war/ear ..etc.)
- The CI saves application package as build artifacts into repositories.

Deployment

- Setup dependencies(Software's) and configuration files in App(Deployment) Servers(Physical/Virtual Machines).
- Download the application package
- Start the application

Build & Deployment Process

Cons:

- Packaging and deploying applications in multiple environments is a real and challenging job.
- Setup Environments before deploying an application.
- Applications may work in one environment which may not work in another environment. The reasons for this are varied; different operating system, different dependencies, different libraries, software versions.

After Docker

Build

- A developer pushes a change to SCM tools.
- The CI sees that new code is available it. It rebuilds the project, runs the tests and generates an application package(jar/war/ear ..etc) & and also we will create docker image(application code(jar/ear/war..etc), dependencies (Software's) and configuration files).
- CI Will push docker image to the docker repo(Public Repo(Docker Hub) or Private Repo(Nexus/JFrog/DTR. Etc.)).

Deployment

- Download the Docker image(package).
- Start the docker image which will create a container where our application will be running.

Advantages:

Application works seamlessly in any environment be it dev or test or prod.

Monolithic VS Microservices Architecture

Monolithic Architecture

Monolithic application has single code base with multiple modules. Modules are divided as either for business features or technical features. It has single build system which build entire application and/or dependency. It also has single executable or deployable binary.

Microservices Architecture

Microservices are a software development technique that arranges an application as a collection of loosely coupled services. Which can be developed, deployed, and maintained independently. Each of these services is responsible for discrete task and can communicate with other services through simple APIs to solve a larger complex business problem.

Monolithic vs. Micro Services

The following table shows a side-by-side comparison of important criteria between monoliths (the old way of architecting applications) and microservices (the new way).

Category	Monolithic architecture	Microservices architecture
Architecture	Built as a single logical executable	Built as a suite of small services
Modularity	Based on language features	Based on business capabilities
Agility	Changes involve building and deploying a new version of the entire application	Changes can be applied to each service independently
Scaling	Entire application is scaled when only one part is the bottleneck	Each service is scaled independently when needed
Implementation	Typically, entirely developed in one programming language	Each service can be developed in a different programming language
Maintainability	Large code base is intimidating to new developers	Smaller code bases are easier to manage
Deployment	Complex deployments with maintenance windows and scheduled downtimes	Simple deployment as each service can be deployed individually, with minimal downtime

Questions ?

Mithun Technologies

Thank You

Mithun Technologies