Chapter – 4

Docker– Containers

Develop faster. Run anywhere.

docker

Accelerate how you build, share, and run modern applications.

18 million +
developers

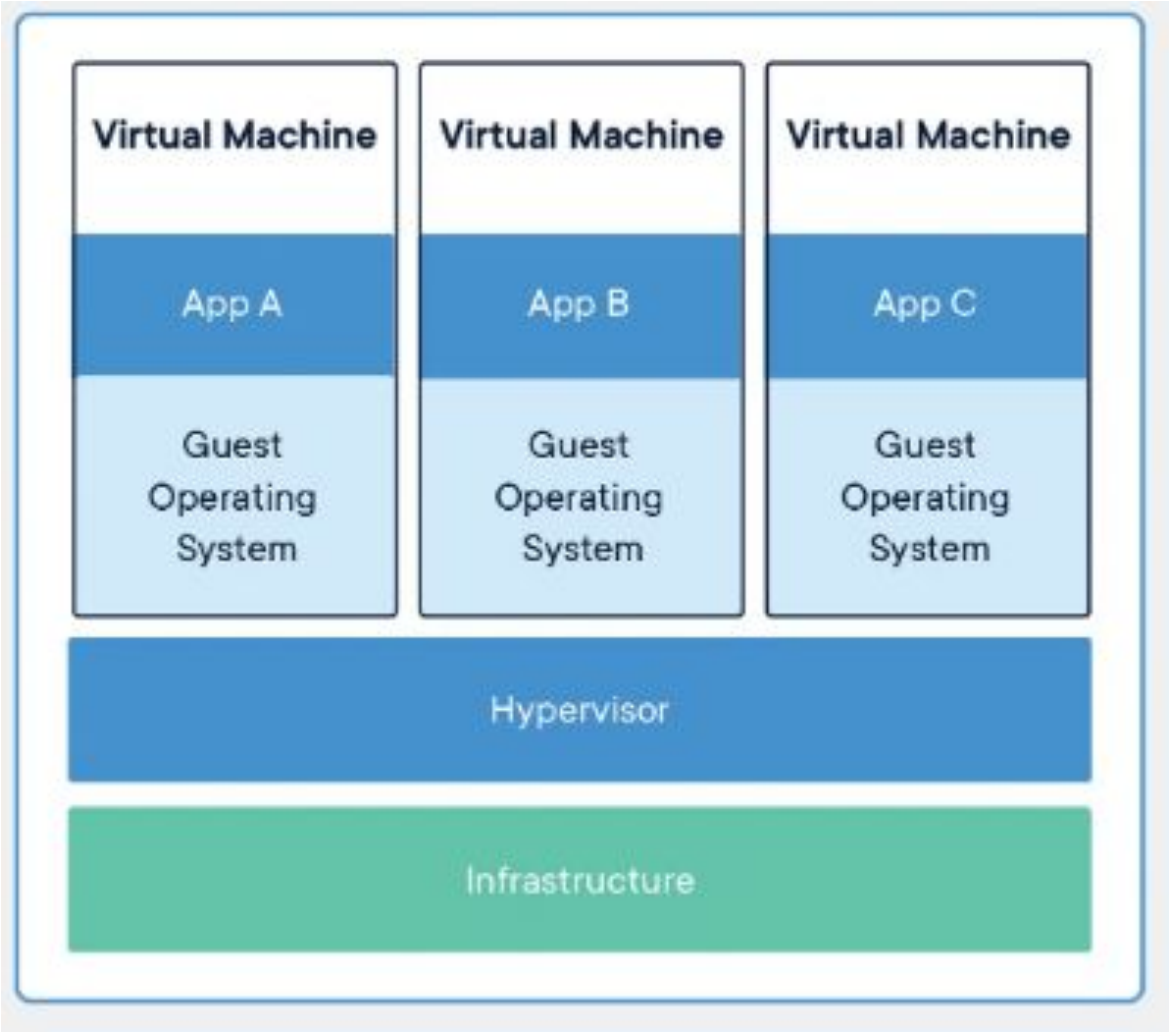
7 million +
applications

13 billion +
monthly image downloads

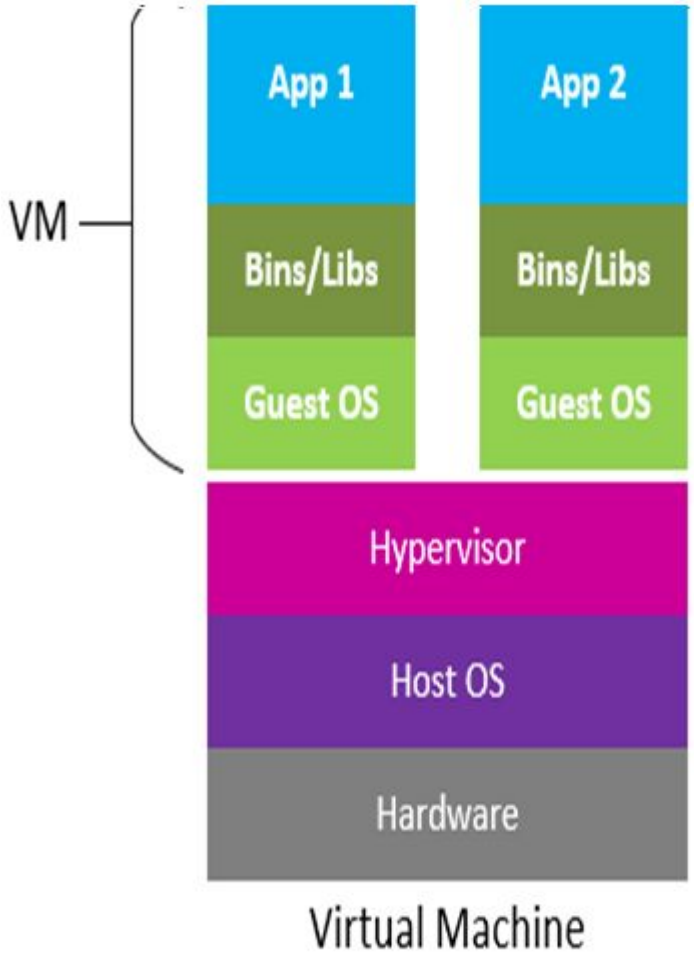
4	4. Docker– Containers & Build tool- Maven 4.1. Introduction: What is a Docker, Use case of Docker, Platforms for Docker, Dockers vs. Virtualization 4.2. Architecture: Docker Architecture., Understanding the Docker components 4.3. Installation: Installing Docker on Linux. Understanding Installation of Docker on windows. Some Docker commands. Provisioning. 4.4. Docker Hub.: Downloading Docker images. Uploading the images in Docker Registry and AWS ECS, Understanding the containers, Running commands in container. Running multiple containers. 4.5. Custom images: Creating a custom image. Running a container from the custom image. Publishing the custom image. 4.6. Docker Networking: Accessing containers, linking containers, Exposing container ports, Container Routing.	30	15

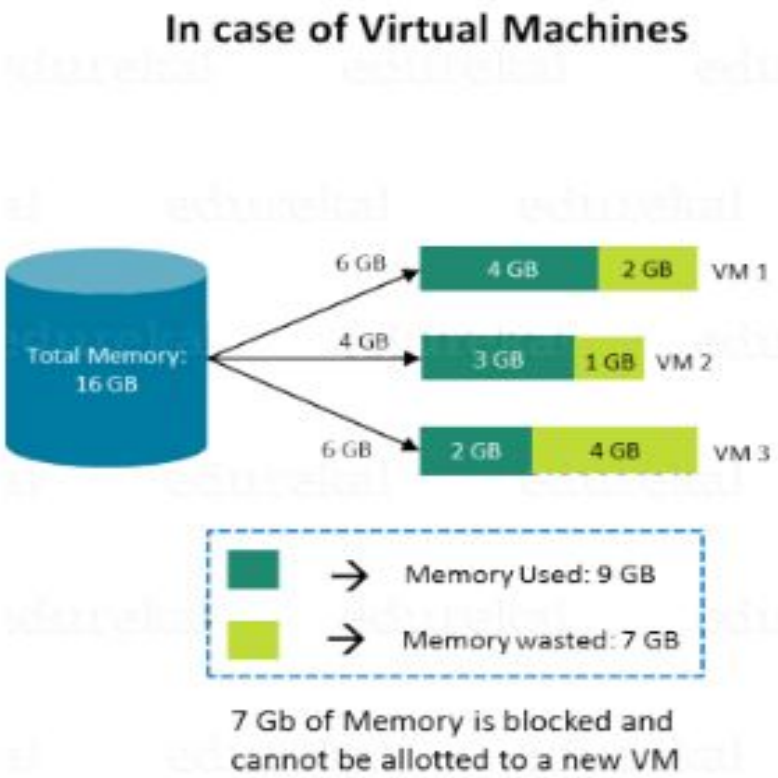
VIRTUAL MACHINES ?

VIRTUAL MACHINES :



VIRTUAL MACHINES :





VIRTUAL MACHINES :

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers.

VIRTUAL MACHINES :

The hypervisor allows multiple VMs to run on a single machine.

VIRTUAL MACHINES :

Each VM includes a full copy of an operating system, the application, necessary binaries and libraries – taking up tens of GBs. VMs can also be slow to boot.

VIRTUAL MACHINES :

Major Drawbacks :

1] Performance degradation due to heavyweight.

2] The lack of application portability.

3] Slowness in provisioning of IT resources.

Docker makes development efficient and predictable

Docker makes development efficient and predictable :

Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development – desktop and cloud.

Docker makes development efficient and predictable :

Docker's comprehensive end to end platform includes UIs, CLIs, APIs and security that are engineered to work together across the entire application delivery lifecycle.

Build

- Get a head start on your coding by leveraging Docker images to efficiently develop your own unique applications on Windows and Mac. Create your multi-container application using Docker Compose.

Build

- Integrate with your favorite tools throughout your development pipeline – Docker works with all development tools you use including VS Code, CircleCI and GitHub.

Build

- Package applications as portable container images to run in any environment consistently from on-premises Kubernetes to AWS ECS, Azure ACI, Google GKE and more.

Share

- Leverage Docker Trusted Content, including Docker Official Images and images from Docker Verified Publishers from the Docker Hub repository.

Share

- Innovate by collaborating with team members and other developers and by easily publishing images to Docker Hub.

Share

- Personalize developer access to images with roles based access control and get insights into activity history with Docker Hub Audit Logs.

Run

Deliver multiple applications hassle free and have them run the same way on all your environments including design, testing, staging and production – desktop or cloud-native.

Run

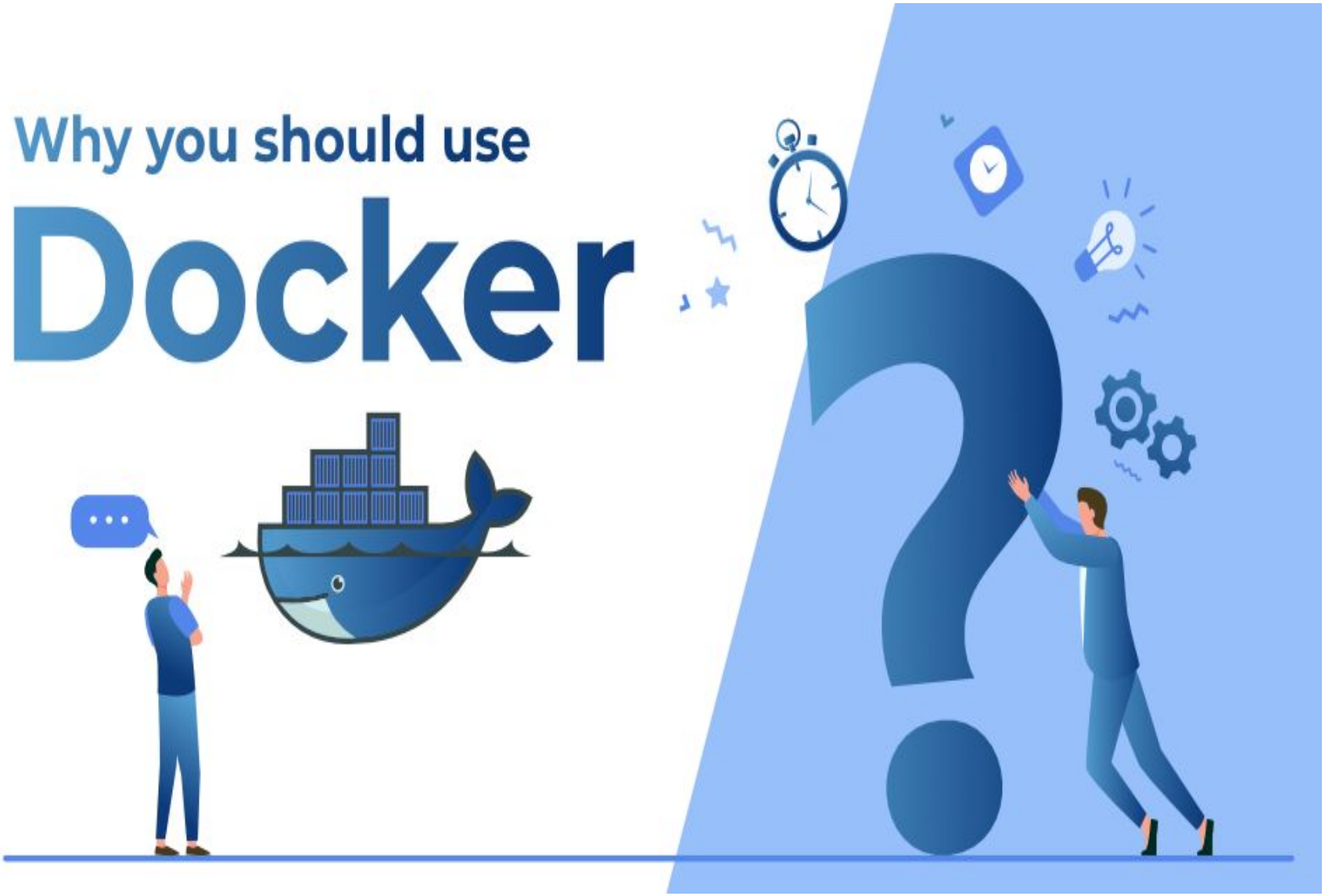
Deploy your applications in separate containers independently and in different languages. Reduce the risk of conflict between languages, libraries or frameworks.

Run

Speed development with the simplicity of Docker Compose CLI and with one command, launch your applications locally and on the cloud with AWS ECS and Azure ACI.

4	4. Docker– Containers & Build tool- Maven 4.1. Introduction: What is a Docker, Use case of Docker, Platforms for Docker, Dockers vs. Virtualization 4.2. Architecture: Docker Architecture., Understanding the Docker components 4.3. Installation: Installing Docker on Linux. Understanding Installation of Docker on windows. Some Docker commands. Provisioning. 4.4. Docker Hub.: Downloading Docker images. Uploading the images in Docker Registry and AWS ECS, Understanding the containers, Running commands in container. Running multiple containers. 4.5. Custom images: Creating a custom image. Running a container from the custom image. Publishing the custom image. 4.6. Docker Networking: Accessing containers, linking containers, Exposing container ports, Container Routing.	30	15

Why you should use Docker



Suppose there are four developers in a team working on a single project. Meanwhile, one is having a Windows system, the second is owning a Linux system, and the third & fourth ones are working with macOS.

Now, as you see, they are using the distinct environments for creating a single application or software they will be required to carry on the things in accordance with their respective machines such as the installation of different libraries & files for their system, etc. And such situations, especially on an organizational or larger level, often cause numerous conflicts and problems throughout the entire software development life cycle.

However, the containerization tools such as Docker eliminates this problem.

Issues we faced before Containerization

Developer

Operating System

Pytest 5.4.3

Pycharm IDE

Backend Code

Dependencies

Tester

Operating System

Pytest 5.3.0

Spyder IDE

Backend Code

Dependencies

Use containers to Build,

Share and Run your applications

Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere.

The whole idea of **Docker** is for developers to easily develop applications, **ship them into containers** which can then be **deployed anywhere**.

Package Software into Standardized Units for Development, Shipment and Deployment

The initial release of Docker was in March 2013.

Definition :

Docker is an open source containerization engine, which automates the packaging, shipping, and deployment of any software applications that are presented as lightweight, portable, and self-sufficient containers, that will run virtually anywhere.

Definition :

A Docker container is a software bucket comprising everything necessary to run the software independently.

Definition :

There can be multiple docker containers in a single machine and containers are completely isolated from one another as well as from the host machine.

Definition :

A docker container includes a software component along with all of its dependencies (binaries, libraries, configuration files, script, jars, and so on)

Definition :

Docker's technology is unique because it focuses on the requirements of developers and systems operators to separate application dependencies from infrastructure.

Definition :

Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

Definition :

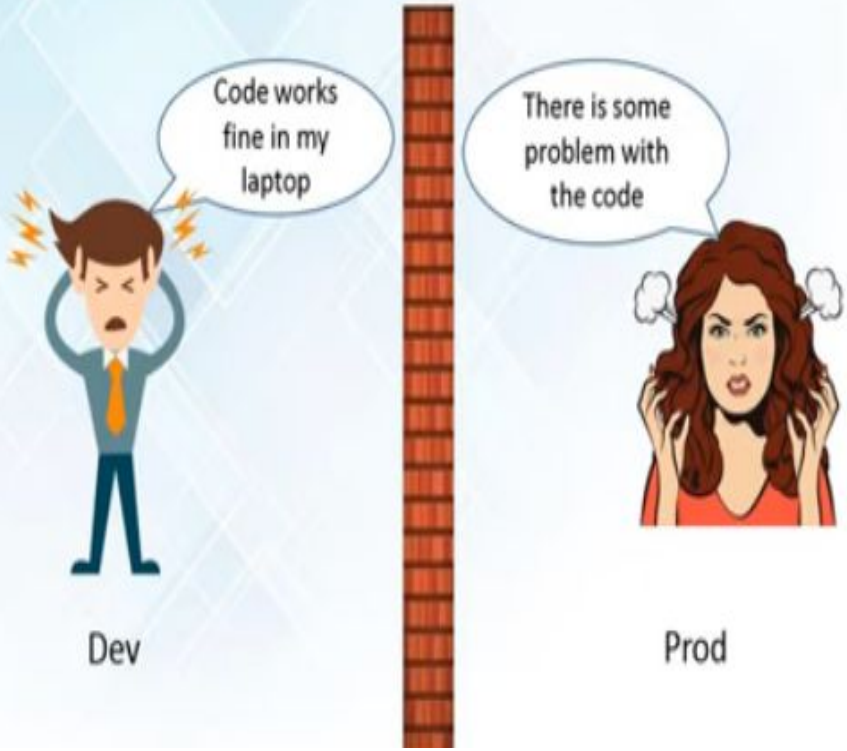
With Docker, you can manage your infrastructure in the same ways you manage your applications.

Definition :

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.

Problems Before Docker

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.



In Dev there can be a software that is upgraded and in Prod the old version of software might be present

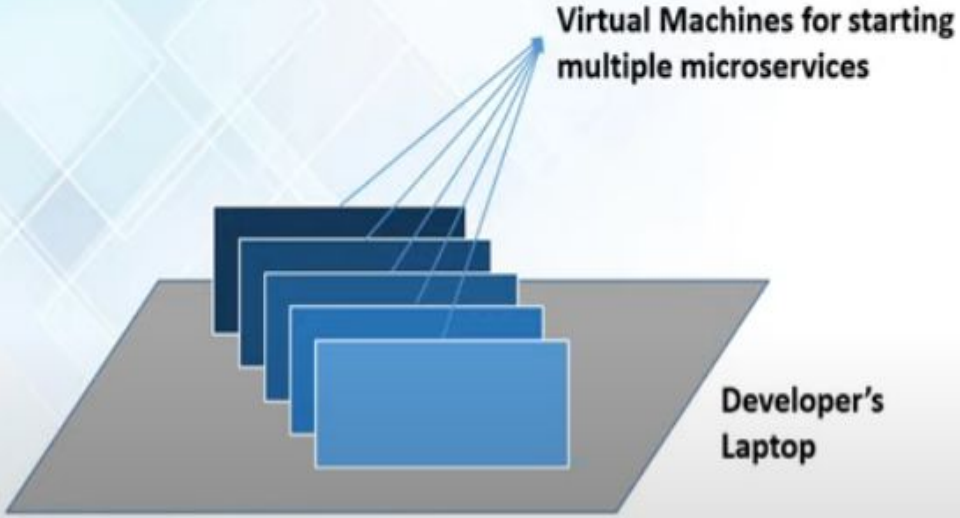
Problems Before Docker

The idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts

Problems Before Docker

Developing an application requires starting several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine.



The Docker platform

The Docker Platform :

Docker provides the ability to package and run an application in a loosely isolated environment called a container.

The Docker Platform :

The isolation and security allows you to run many containers simultaneously on a given host.

The Docker Platform :

Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host.

The Docker Platform :

You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

Develop your application and its supporting components using containers.

The container becomes the unit for distributing and testing your application.

When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

Features of Docker

Features of Docker :

Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.

Features of Docker :

With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.

Features of Docker :

You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.

Features of Docker :

Since Docker containers are pretty lightweight, they are very easily scalable.

It Ensures Scalability & Flexibility

Features of Docker :

Consistent & Isolated Environment :

Docker provides you with a consistent and isolated environment. It takes the responsibility of isolating and segregating your apps and resources in such a way that each container becomes able to access all the required resources in an isolated manner i.e., without disturbing or depending on another container. It eventually allows you to run multiple containers simultaneously on the same host.

Features of Docker :

Rapid Application Deployment :

Docker indeed fastens the application deployment process to a greater extent. It efficiently organizes the entire development lifecycle by providing a standardized working environment to the developers.

Docker is very preferable for Continuous Integration and Continuous Delivery (CI/CD) workflows.

Features of Docker :

Better Portability :

Another enriching advantage of Docker is Portability! The applications created with Docker containers are immensely portable. The Docker containers can run on any platform whether it be **Amazon EC2, Google Cloud Platform, VirtualBox, Rackspace server, or any other – though the host OS should support Docker.**

Features of Docker :

Cost-Effective :

Docker reduce overall cost without compromising with the **standard workflow** or **product quality**.

Features of Docker :

In-Built Version Control System :

Docker – it comes up with an **in-built version control** system. The Docker containers allow you to commit changes to the Docker images and version control them conveniently.

For instance – if you are having some issues with the current or upgraded version of the image – you can quickly roll back to a previous stable version of the Docker image.

Features of Docker :

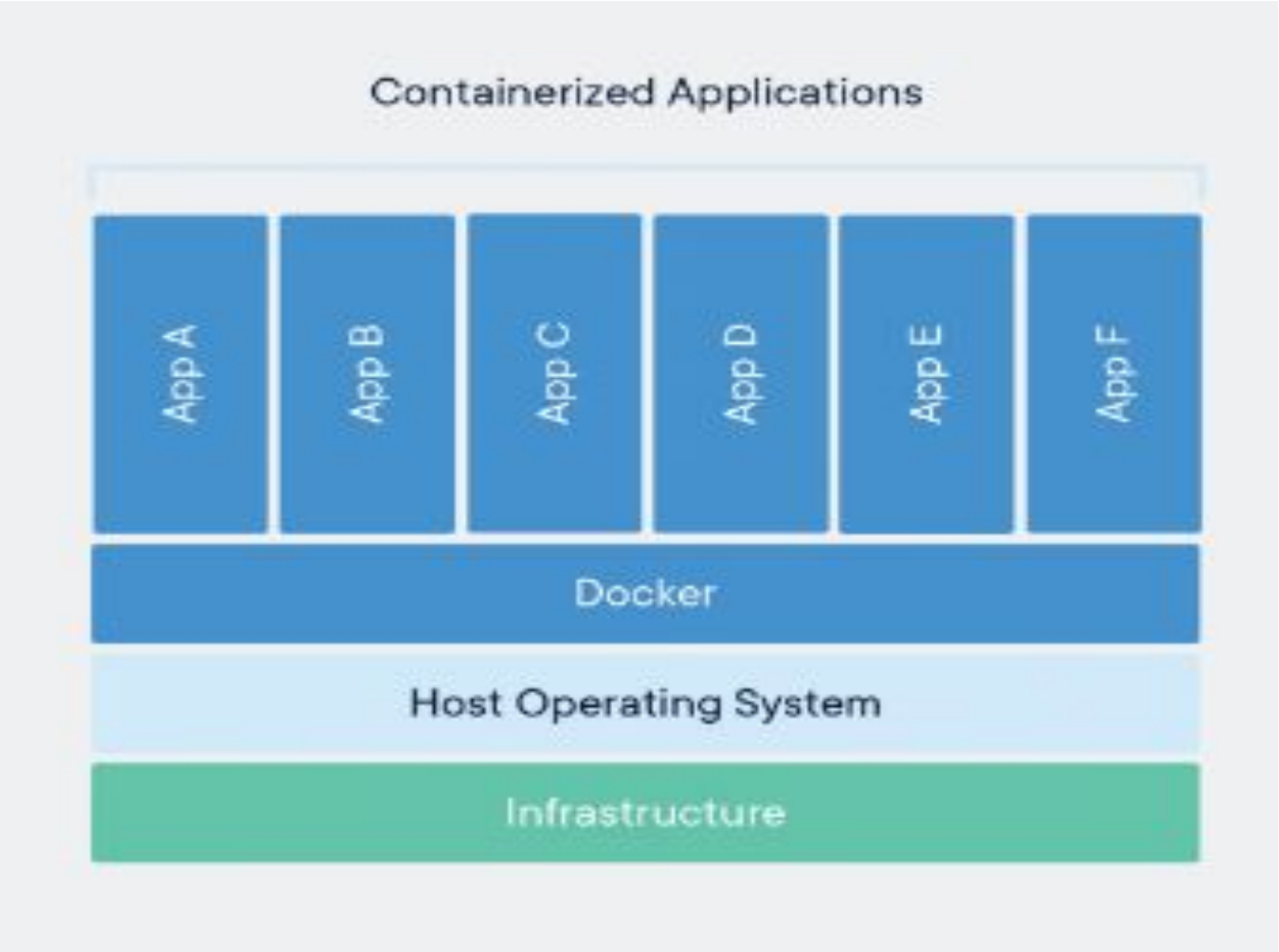
Security :

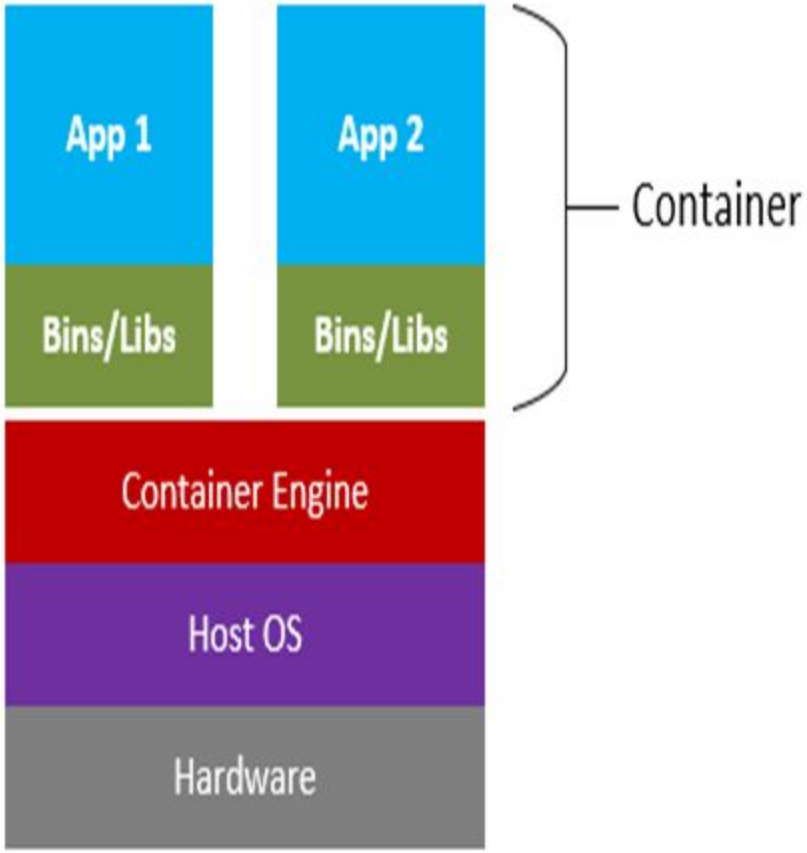
Docker takes the responsibility of complete isolation and segregation of applications running within the Docker containers with each other – the developers have complete control over the traffic course.

A particular container cannot access the data of another container without having authorized access.

A container is a **standard unit of software** that packages up code and all its **dependencies** so the application runs quickly and reliably from one computing environment to another.

A **Docker container** image is a **lightweight**, **standalone**, **executable package of software** that includes everything needed to run an application: **code, runtime, system tools, system libraries and settings.**


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure


App 1
App 2
Bins/Libs
Bins/Libs
Container
Container Engine
Host OS
Hardware

Container images become containers at runtime and in the case of Docker containers – images become containers when they run on Docker Engine.

Docker developed a **Linux container technology** – one that is portable, flexible and easy to deploy.

Available for both **Linux** and **Windows-based** applications, containerized software will always run the same, regardless of the infrastructure.

Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Docker containers that run on Docker Engine

Docker containers that run on Docker Engine:

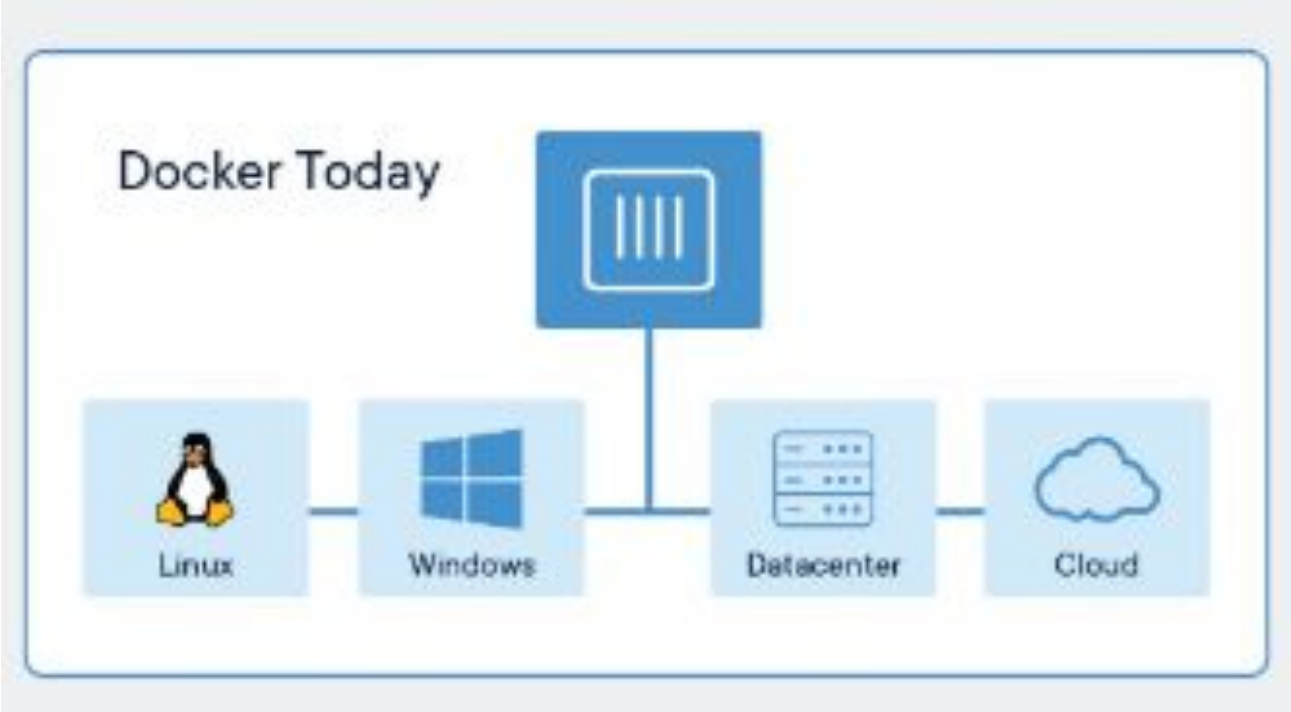
Standard: Docker created the industry standard for containers, so they could be **portable anywhere**.

Docker containers that run on Docker Engine:

Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs

Docker containers that run on Docker Engine:

Secure : Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.

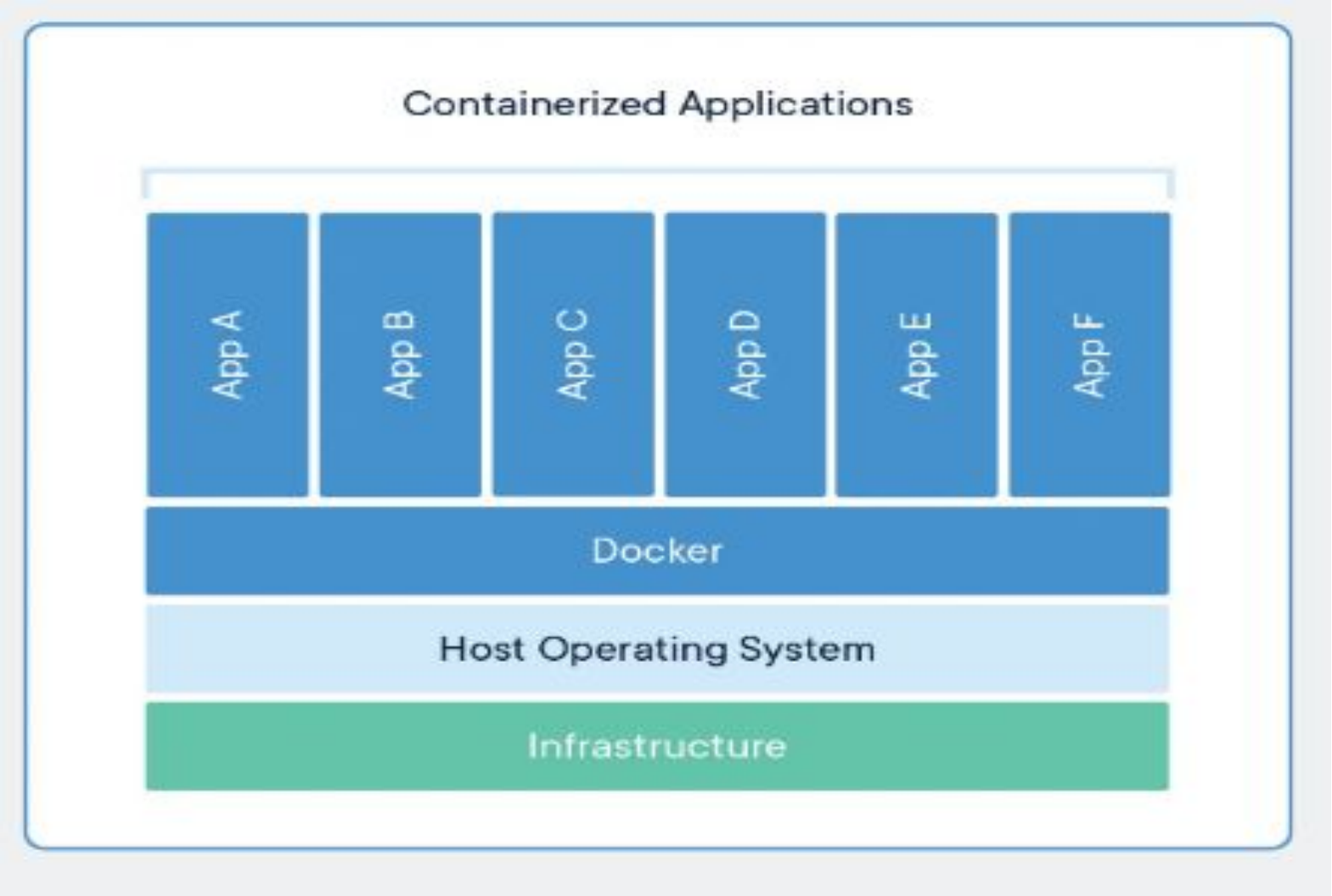


Docker Containers Are Everywhere: Linux, Windows, Data center, Cloud, Serverless, etc.

Docker container technology was launched in **2013** as an **open source** Docker Engine.

Docker's technology is unique because it focuses on the requirements of developers and systems operators to **separate application dependencies from infrastructure.**

CONTAINERS :


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

CONTAINERS :

Containers are an abstraction at the app layer that packages code and dependencies together.

CONTAINERS :

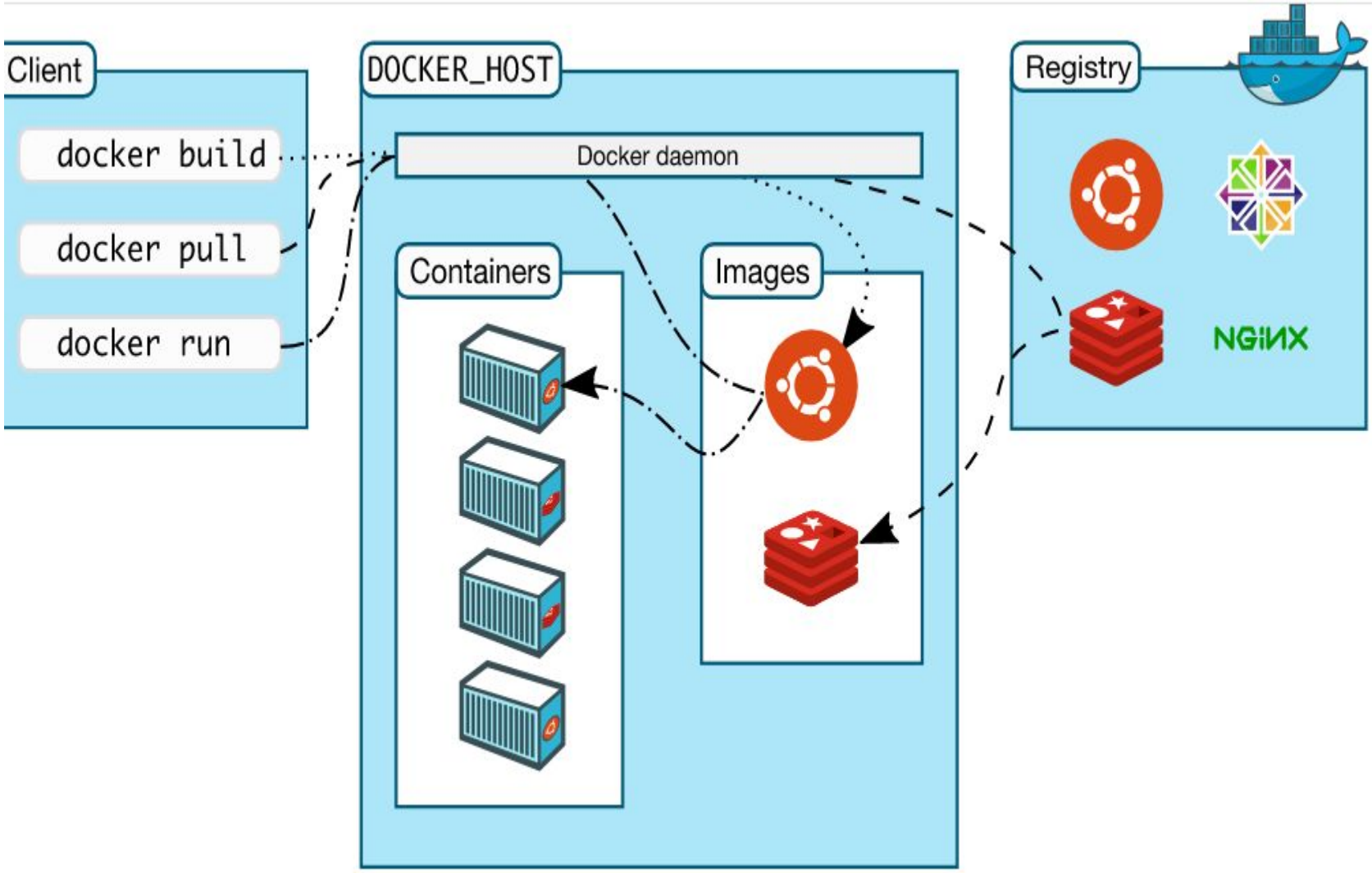
Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.

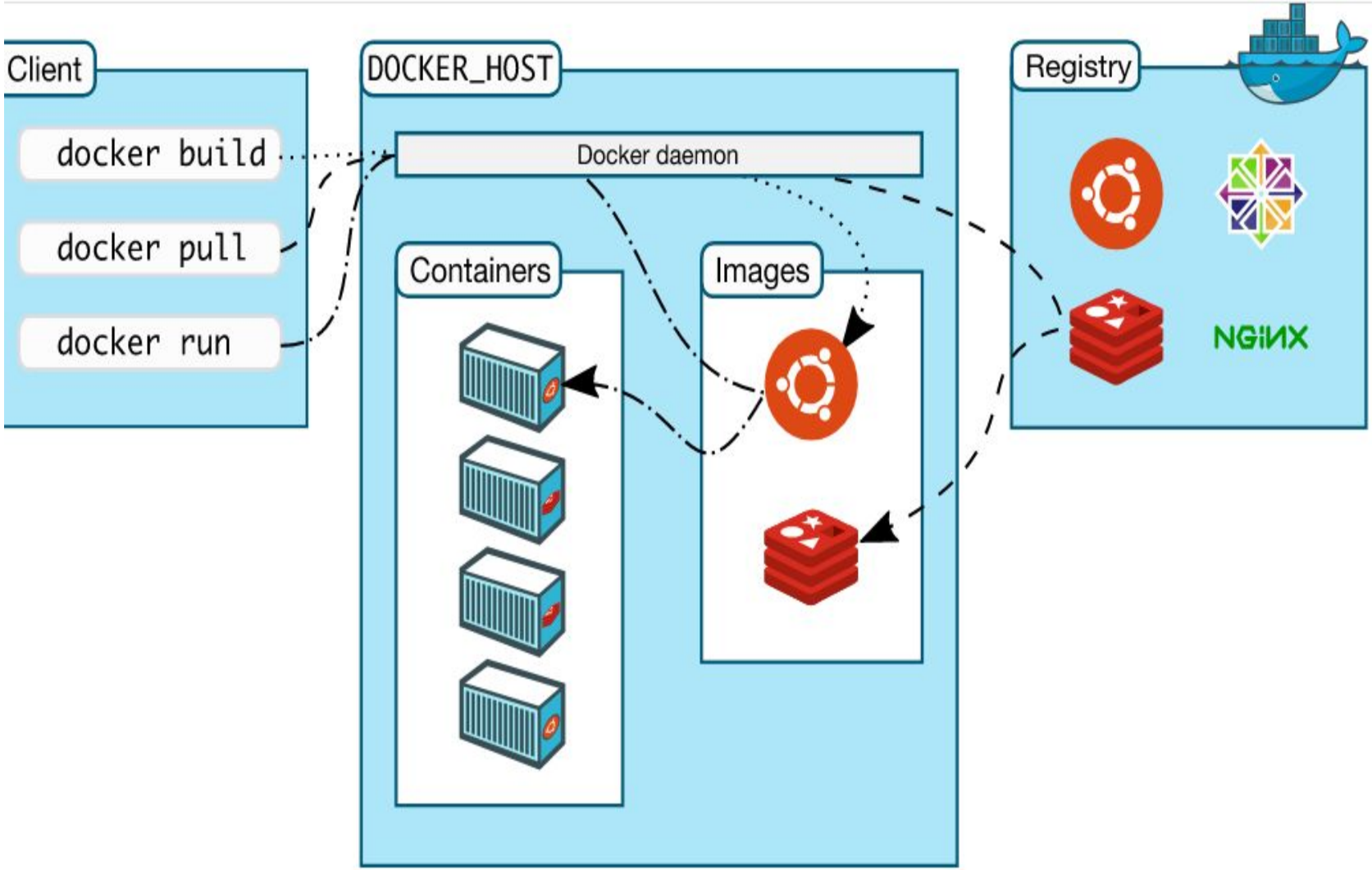
CONTAINERS :

Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

4	4. Docker– Containers & Build tool- Maven 4.1. Introduction: What is a Docker, Use case of Docker, Platforms for Docker, Dockers vs. Virtualization 4.2. Architecture: Docker Architecture., Understanding the Docker components 4.3. Installation: Installing Docker on Linux. Understanding Installation of Docker on windows. Some Docker commands. Provisioning. 4.4. Docker Hub.: Downloading Docker images. Uploading the images in Docker Registry and AWS ECS, Understanding the containers, Running commands in container. Running multiple containers. 4.5. Custom images: Creating a custom image. Running a container from the custom image. Publishing the custom image. 4.6. Docker Networking: Accessing containers, linking containers, Exposing container ports, Container Routing.	30	15

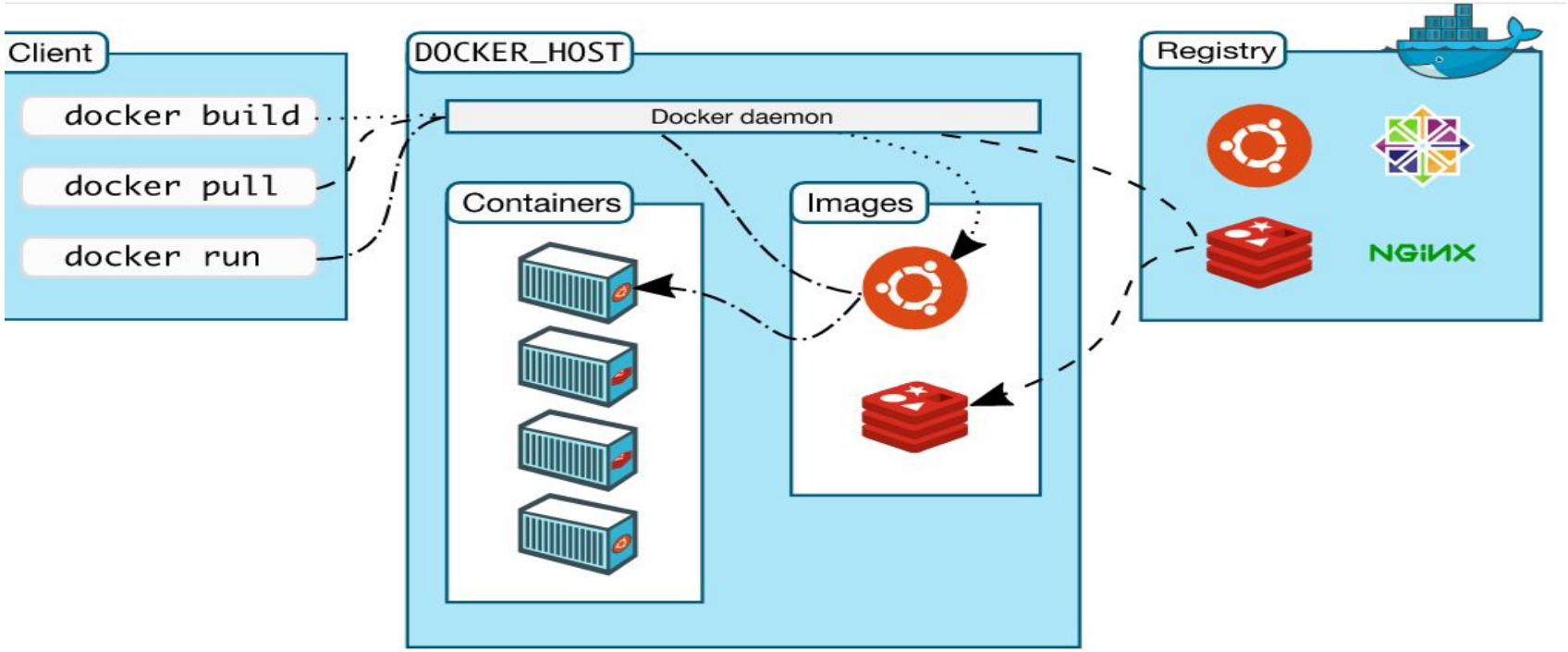
Docker Architecture


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

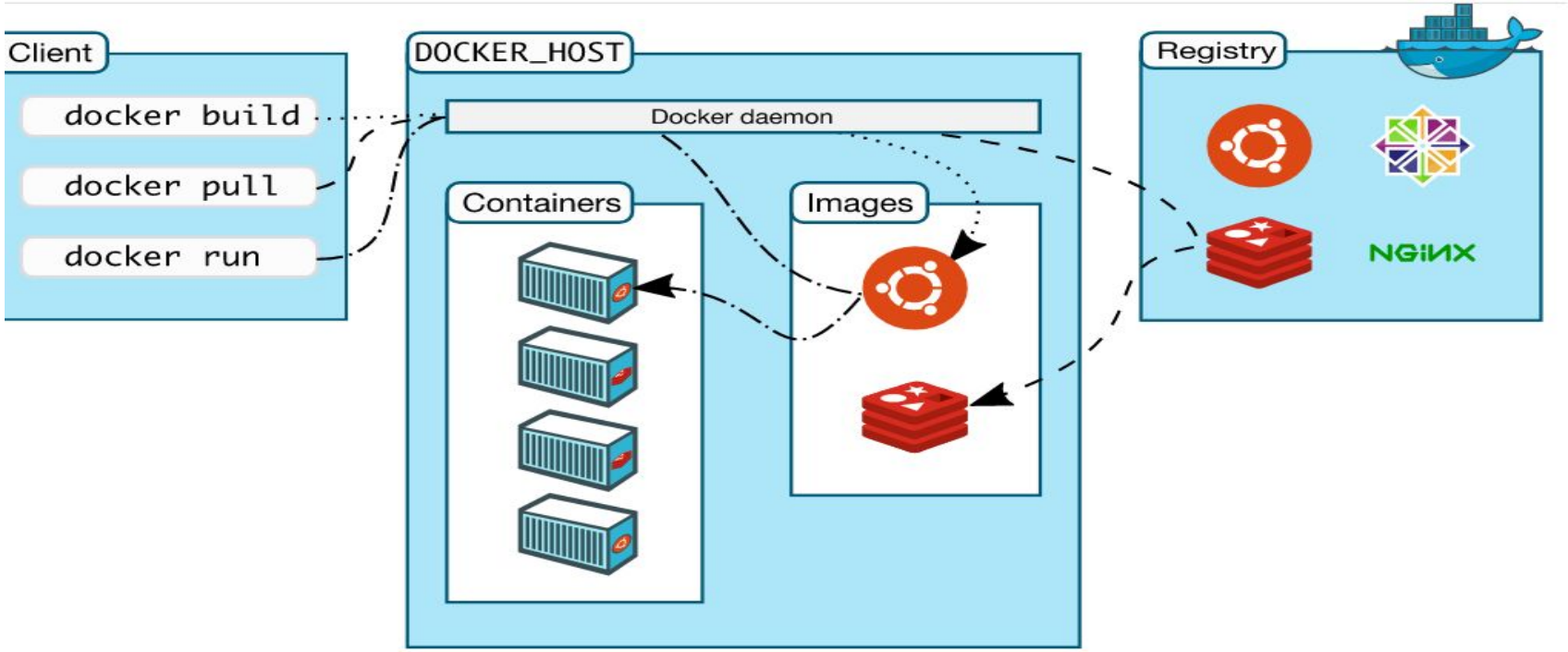


Docker uses a **client-server architecture.**

The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.

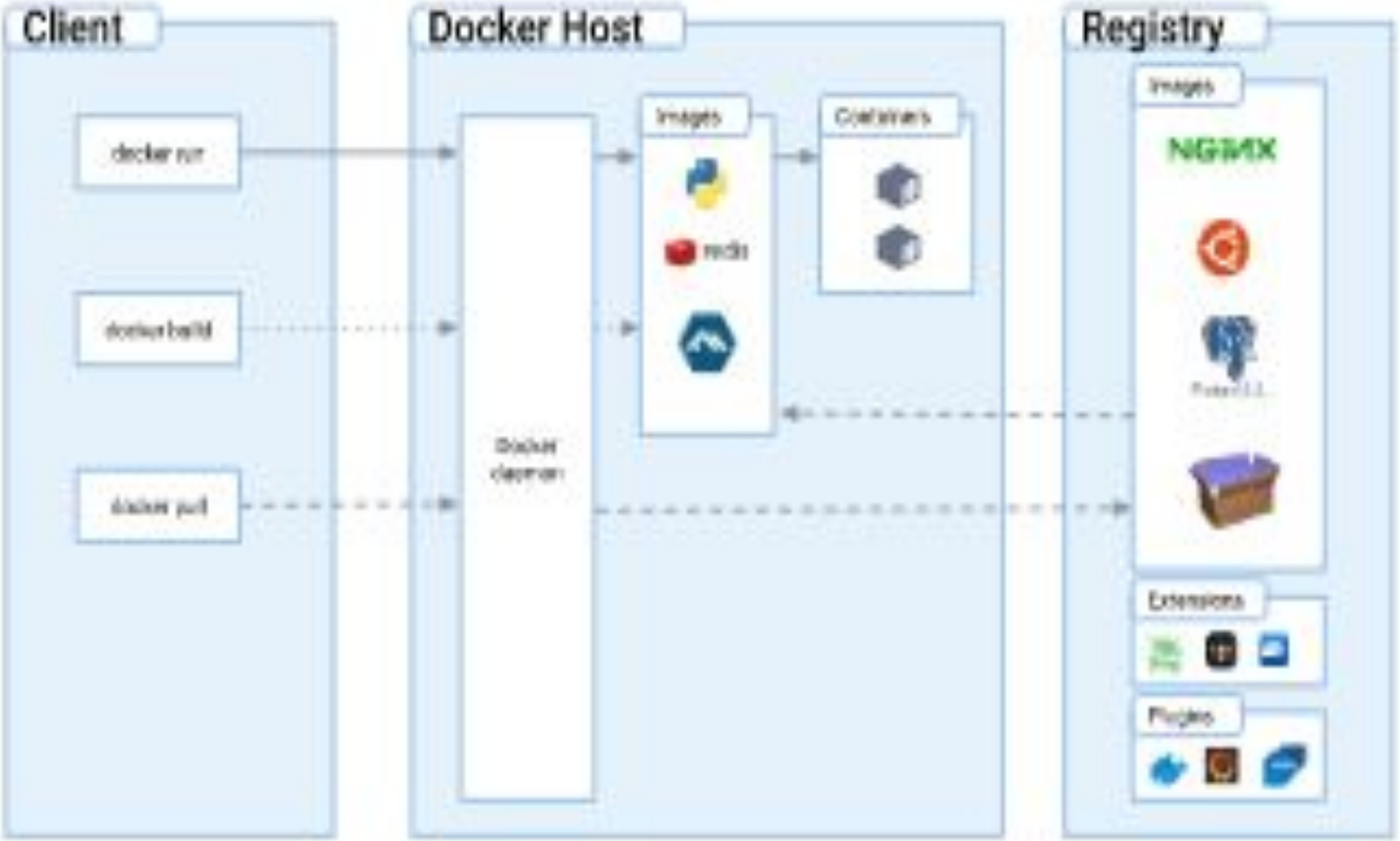


The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon.

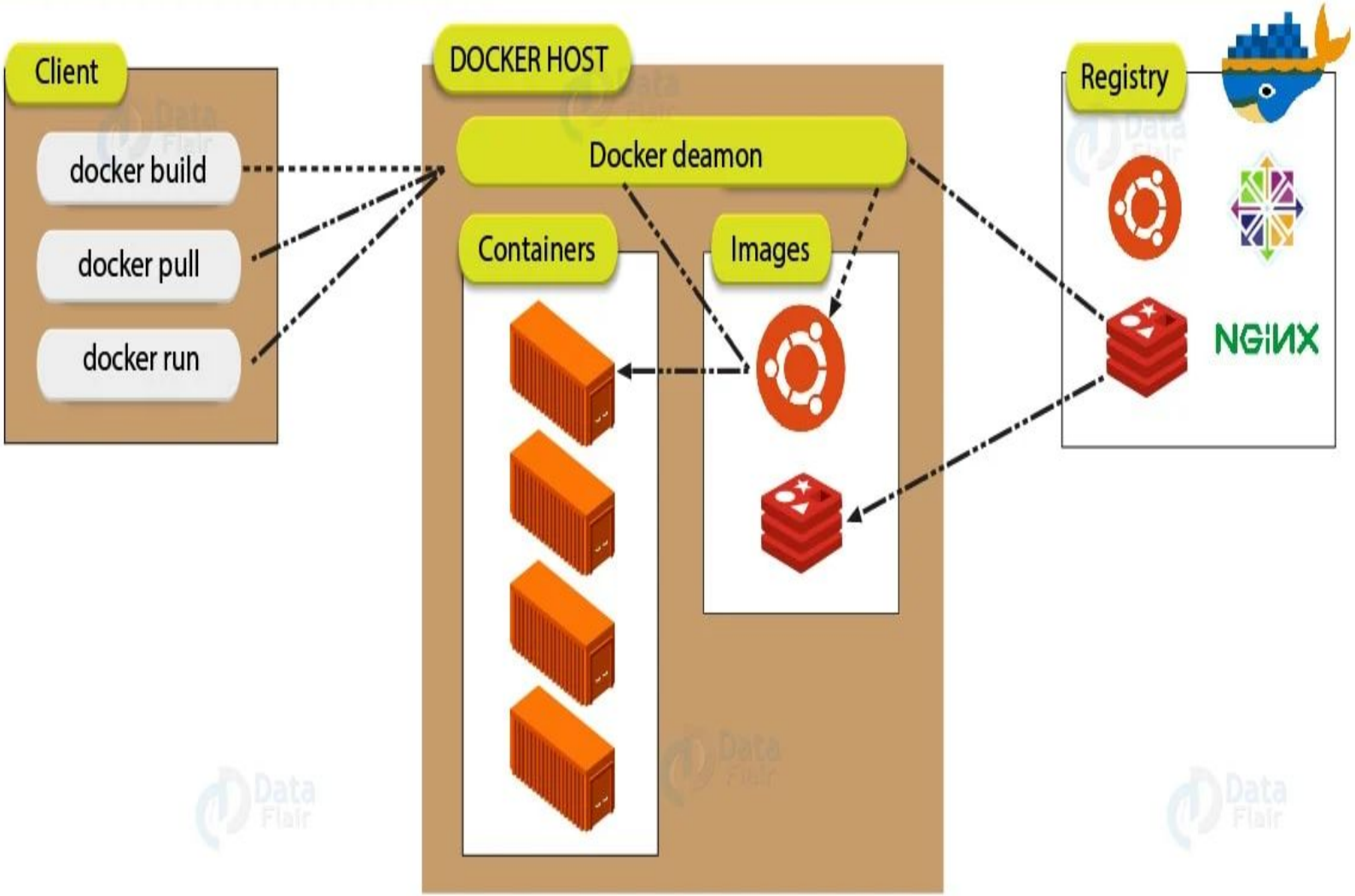


The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

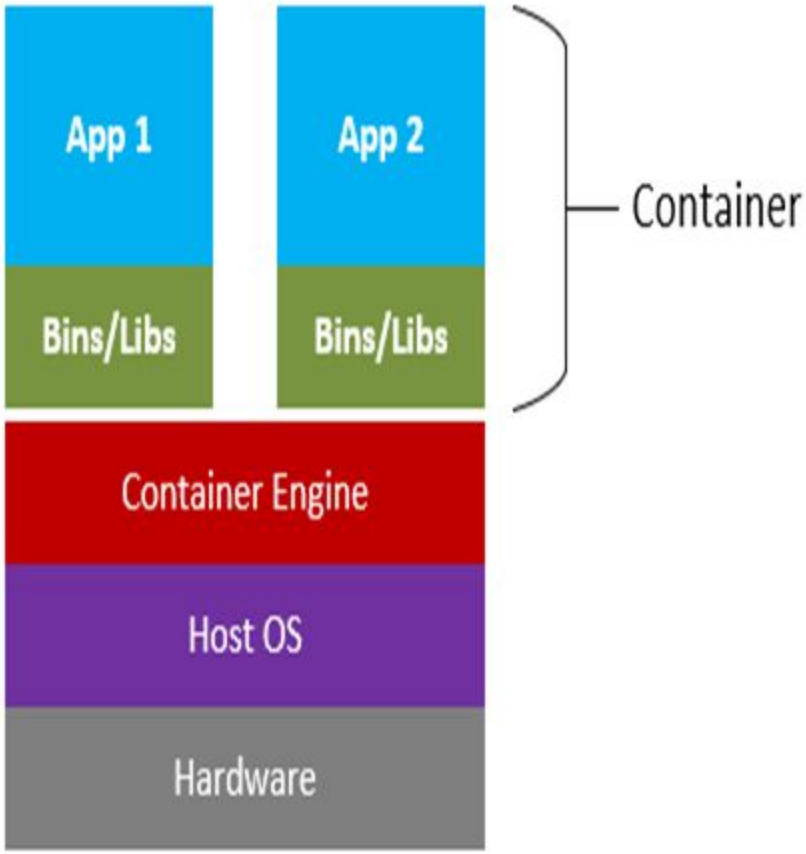
Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.


Client
Docker Host
Registry
docker run
docker build
docker pull
Images
Containers
redis
Images
NGINX
Extensions
Plugins

Docker Architecture



Components of Docker :



Components of Docker :

Docker Engine / Docker Deamon– It is used for building Docker images and creating Docker containers. The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

Components of Docker :

Docker Hub / Registry : A Docker *registry* stores 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 even run your own private registry.

Components of Docker :

Docker Hub / Registry :

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 image is pushed to your configured registry.

Components of Docker :

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.

Components of Docker :

Containers :

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

Components of Docker :

Containers :

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

Components of Docker :

Containers :

A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

The underlying technology :

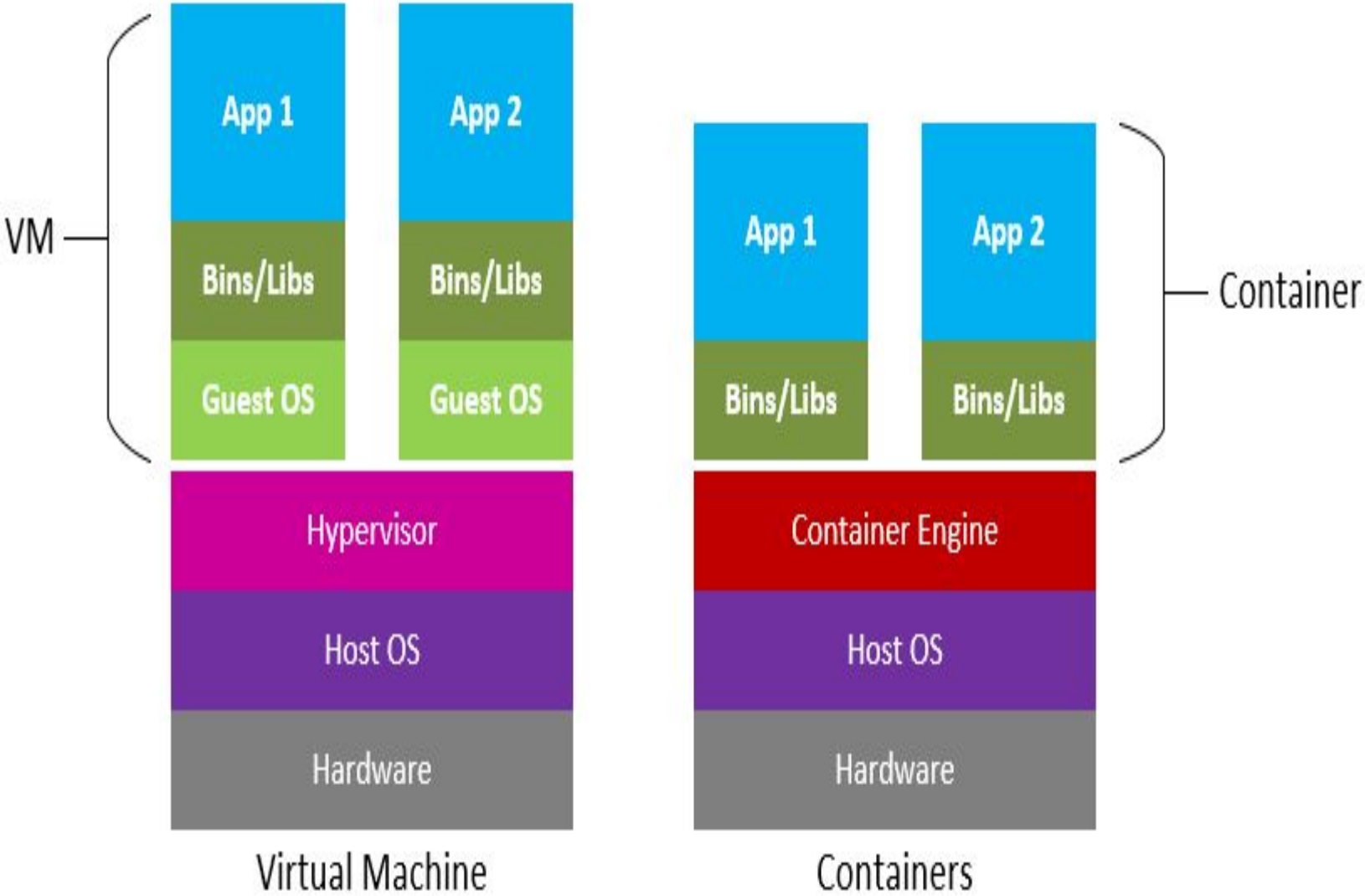
Docker is written in the **Go programming language** and takes advantage of several features of the Linux kernel to deliver its functionality.

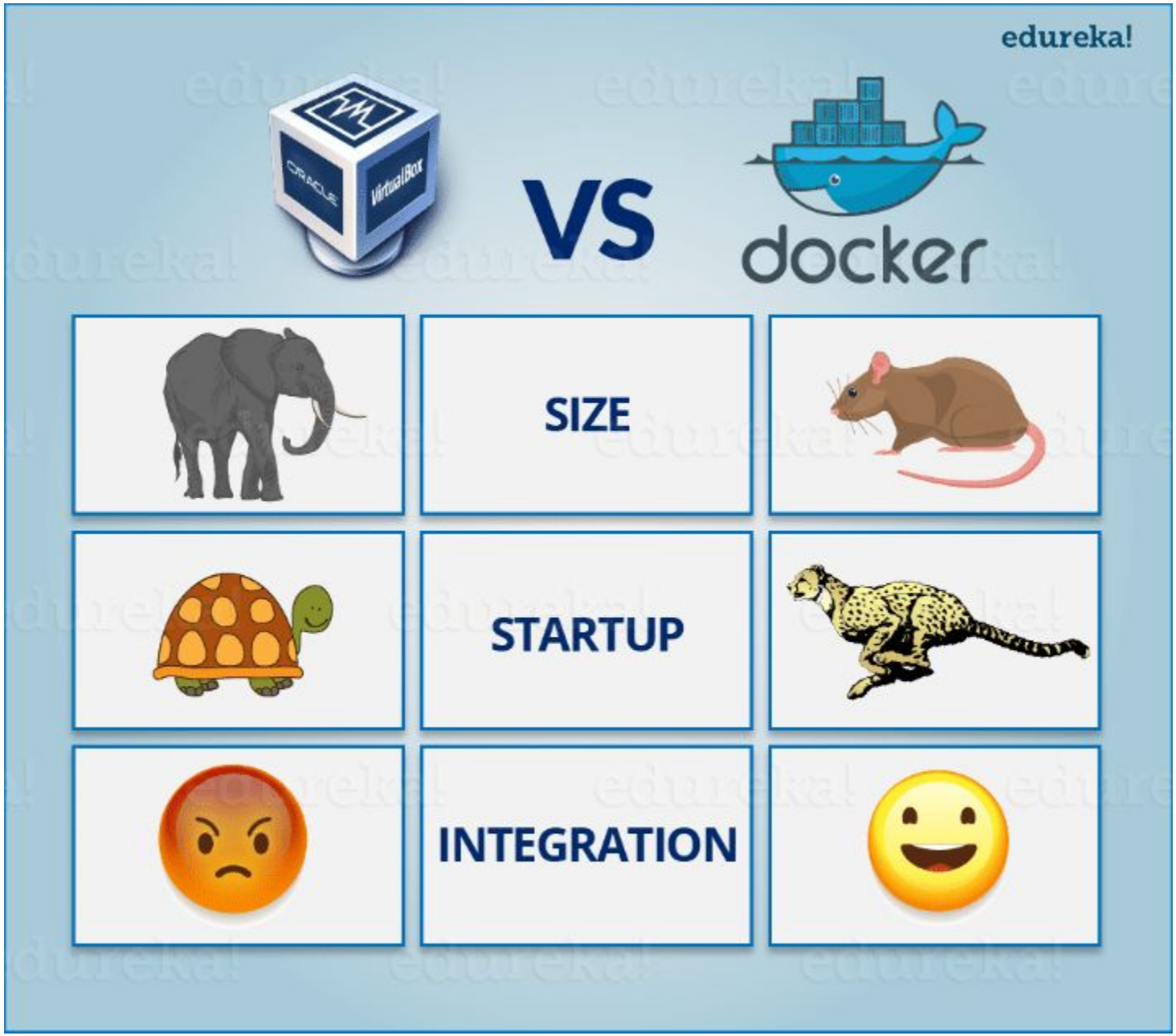
Docker uses a technology called namespaces to provide the isolated workspace called the *container*. When you run a container, Docker creates a set of *namespaces* for that container.

The underlying technology :

These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

Virtual Machine vs Docker


VM
App 1
App 2
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Hardware
Virtual Machine
App 1
App 2
Bins/Libs
Bins/Libs
Container
Container Engine
Host OS
Hardware
Containers


edureka!
VS
docker
SIZE
STARTUP
INTEGRATION

Virtual Machine	Docker Container
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find	Pre-built docker containers are easily available

Virtual Machine	Docker Container
VMs can move to new host easily	Containers are destroyed and re-created rather than moving
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

Installation

docs.docker.com/desktop/install/windows-install/

docker docs | Search the docs | Home | Guides | Manuals | Reference

- Install on Apple silicon
- Install on Windows
- Install on Linux
- Installation per Linux distro
- Quick Start Guide and sign in
- Explore Docker Desktop
- Change settings
- Troubleshoot and diagnose
- Additional resources

Install Docker Desktop on Windows

Estimated reading time: 10 minutes

Update to the Docker Desktop terms

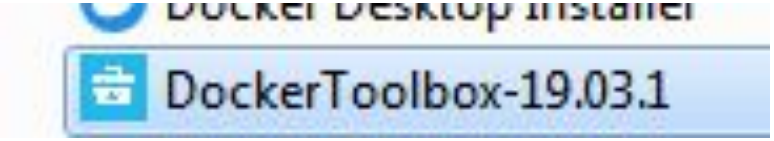
Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) now requires a paid subscription.

Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and

https://docs.docker.com/desktop/install/windows-install/

https://docs.docker.com/desktop/install/windows-install/

windows 10 onwards version



docker toolbox for windows 7

Docker for Windows

Docker has out-of-the-box support for Windows, but you need to have the following configuration in order to install Docker for Windows.

System Requirements

Windows OS	Windows 10 64 bit
Memory	2 GB RAM (recommended)

Setup - Docker Toolbox

Welcome to the Docker Toolbox Setup Wizard

This will install Docker Toolbox version 19.03.1 on your computer.

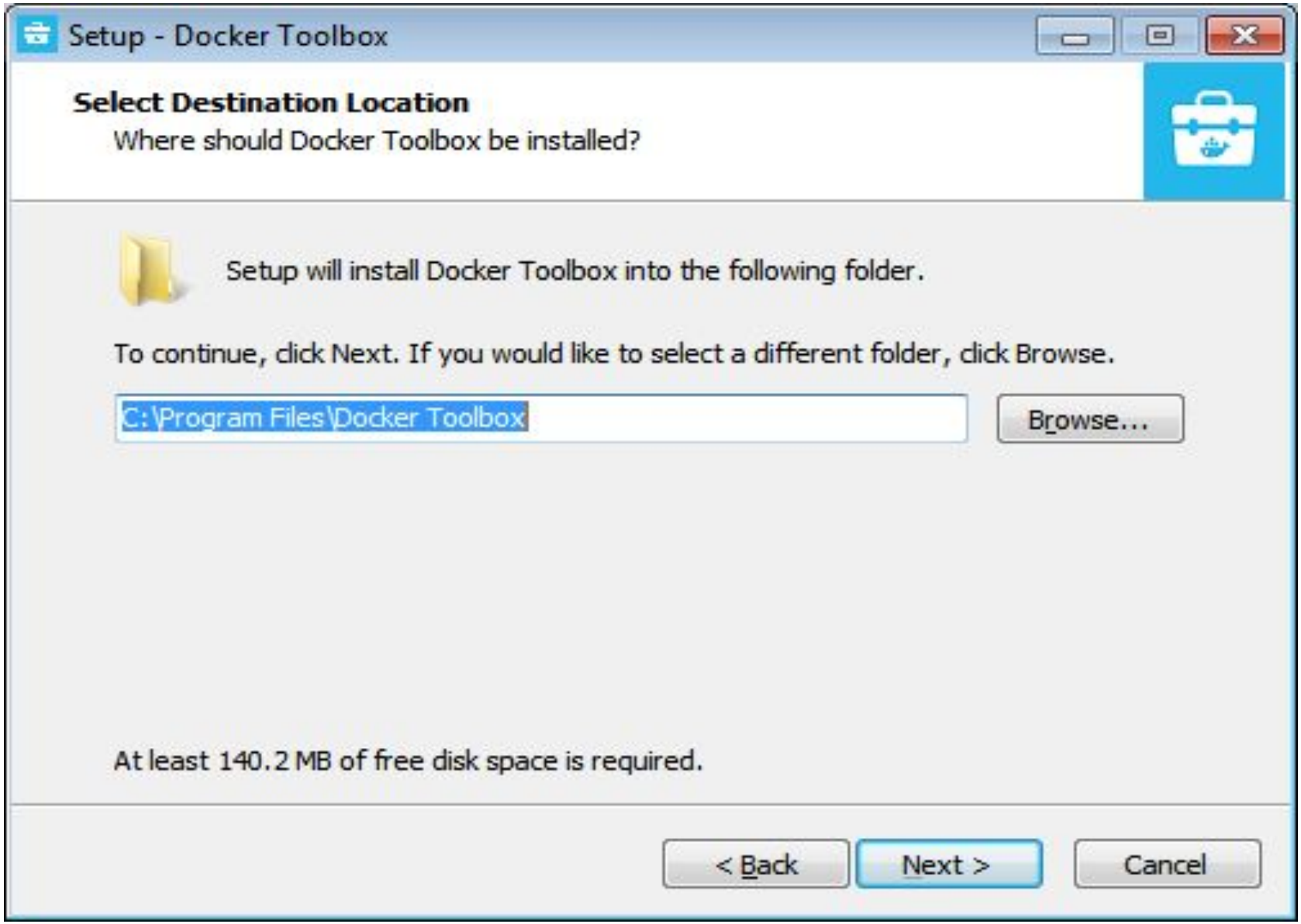
It is recommended that you close all other applications before continuing.

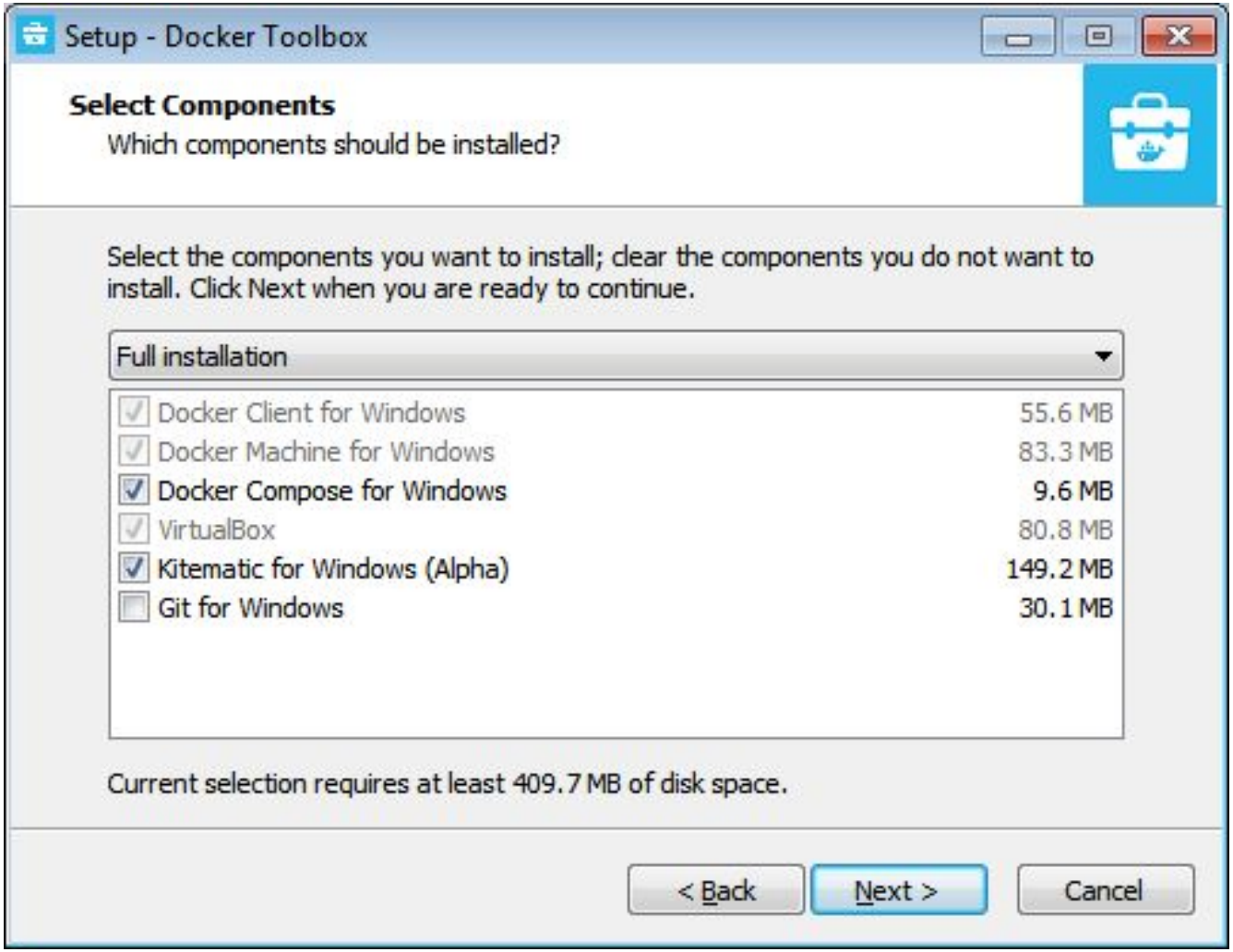
Click Next to continue, or Cancel to exit Setup.

☑ Help Docker improve Toolbox.

This collects anonymous data to help us detect installation problems and improve the overall experience. We only use it to aggregate statistics and will never share it with third parties.

Next > | Cancel


Setup - Docker Toolbox
Select Destination Location
Where should Docker Toolbox be installed?
Setup will install Docker Toolbox into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Docker Toolbox
Browse...
At least 140.2 MB of free disk space is required.
< Back
Next >
Cancel


Setup - Docker Toolbox
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Full installation
Docker Client for Windows 55.6 MB
Docker Machine for Windows 83.3 MB
Docker Compose for Windows 9.6 MB
VirtualBox 80.8 MB
Kitematic for Windows (Alpha) 149.2 MB
Git for Windows 30.1 MB
Current selection requires at least 409.7 MB of disk space.
< Back
Next >
Cancel

Setup - Docker Toolbox

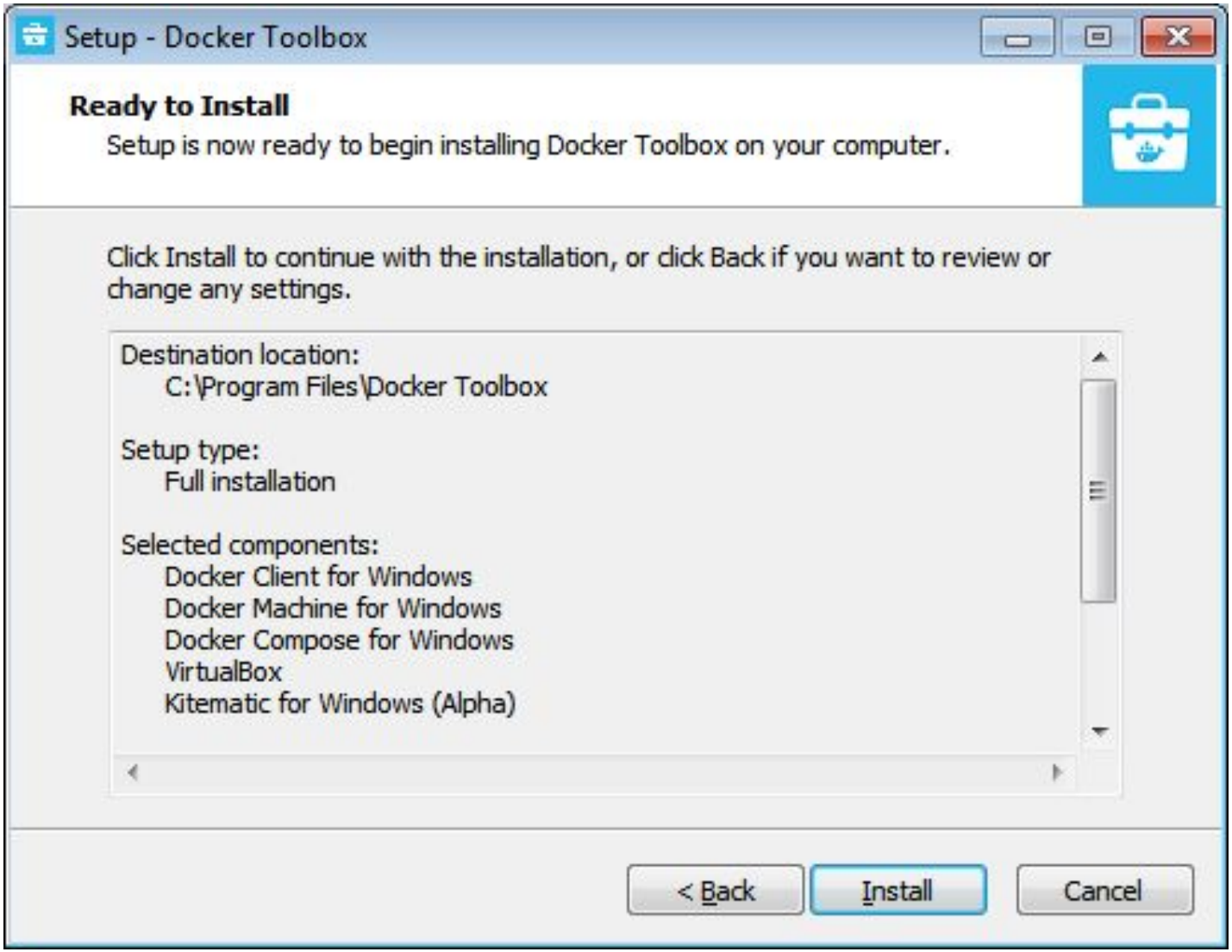
Select Additional Tasks

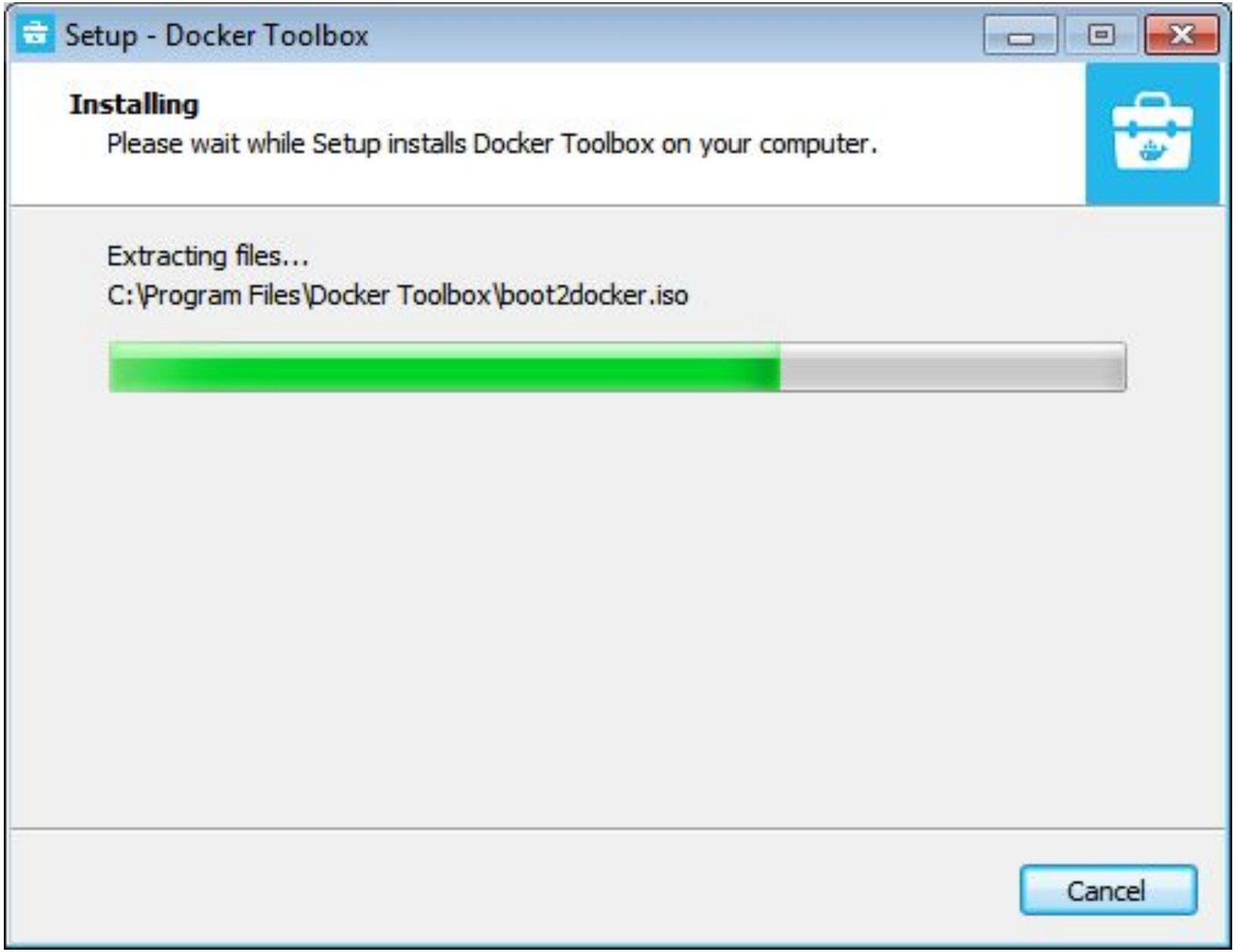
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Docker Toolbox, then click Next.

- [x] Create a desktop shortcut
- [x] Add docker binaries to PATH
- [x] Upgrade Boot2Docker VM
- [x] Install VirtualBox with NDIS5 driver [default NDIS6]

< Back | Next > | Cancel


Setup - Docker Toolbox
Ready to Install
Setup is now ready to begin installing Docker Toolbox on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files\Docker Toolbox
Setup type:
Full installation
Selected components:
Docker Client for Windows
Docker Machine for Windows
Docker Compose for Windows
VirtualBox
Kitematic for Windows (Alpha)
< Back
Install
Cancel


Setup - Docker Toolbox
Installing
Please wait while Setup installs Docker Toolbox on your computer.
Extracting files...
C:\Program Files\Docker Toolbox\boot2docker.iso
Cancel

Double click on docker Quickstart Terminal

```
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com


Start interactive shell

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
$ docker -v
Docker version 19.03.1, build 74b1e89e8a

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker --version
Docker version 19.03.1, build 74b1e89e8a
```

docker info :

Display system-wide information.

$ docker info [OPTIONS]

```
$ docker info
Client:
 Debug Mode: false

Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
 Server Version: 19.03.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk s
yslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.19.130-boot2docker
 Operating System: Boot2Docker 19.03.12 (TCL 10.1)
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 985.4MiB
 Name: default
 ID: FVJF:A2JI:U6V2:46LV:67YD:SC2P:2OC4:45AI:NPXJ:XHMV:X6SU:NC6Y
 Docker Root Dir: /mnt/sda1/var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
  provider=virtualbox
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
```

Docker Commands

$ docker search

We can use the command docker search to search for public images on the Docker hub. It will return information about the image name, description, stars, official and automated.

$ docker search hello-world

```
$ docker search hello-world
NAME                                                      DESCRIPTION
                STARS                 OFFICIAL                  AUTOMATED
hello-world                                               Hello World! (an example of minimal D
ockeriz   2020                  [OK]
kitematic/hello-world-nginx                               A light-weight nginx container that d
emonstr   153
tutum/hello-world                                         Image to test docker deployments. Has
 Apache   90                                                     [OK]
dockercloud/hello-world                                   Hello World!
                20                                               [OK]
crccheck/hello-world                                      Hello World web server in under 2.5 M
B               15                                               [OK]
vad1mo/hello-world-rest                                   A simple REST Service that echoes bac
k all t   7                                                     [OK]
rancher/hello-world
                4
ansibleplaybookbundle/hello-world-db-apb                  An APB which deploys a sample Hello W
orld! a   2                                                     [OK]
ppc64le/hello-world                                       Hello World! (an example of minimal D
ockeriz   2
thomaspoignant/hello-world-rest-json                      This project is a REST hello-world AP
I to bu   2
ansibleplaybookbundle/hello-world-apb                     An APB which deploys a sample Hello W
orld! a   1                                                     [OK]
businessgeeks00/hello-world-nodejs
                0
okteto/hello-world
                0
strimzi/hello-world-producer
                0
strimzi/hello-world-consumer
                0
golift/hello-world                                        Hello World Go-App built by Go Lift A
pplicat   0
koudaiii/hello-world
                0
freddiedevops/hello-world-spring-boot
                0
strimzi/hello-world-streams
                0
garystafford/hello-world                                  Simple hello-world Spring Boot servic
e for t   0                                                     [OK]
```

Docker Commands

$ docker search

We can use the command docker search to search for public images on the Docker hub. It will return information about the image name, description, stars, official and automated.

$ docker search MySQL

Docker Commands

```
$ docker search MySQL
NAME                                    DESCRIPTION
STARS                OFFICIAL                   AUTOMATED
mysql                                   MySQL is a widely used, open-source relation
898                  [OK]
mariadb                                 MariaDB Server is a high performing open sou
45                   [OK]
percona                                 Percona Server is a fork of the MySQL relati
2                    [OK]
phpmyadmin                              phpMyAdmin - A web interface for MySQL and M
7                    [OK]
bitnami/mysql                           Bitnami MySQL Docker Image
2                                                   [OK]
linuxserver/mysql-workbench
7
linuxserver/mysql                       A Mysql container, brought to you by LinuxSe

ubuntu/mysql                            MySQL open source fast, stable, multi-thread

circleci/mysql                          MySQL is a widely used, open-source relation

google/mysql                            MySQL server for Google Compute Engine
1                                                   [OK]
rapidfort/mysql                         RapidFort optimized, hardened image for MySQL
3
bitnami/mysqld-exporter

ibmcom/mysql-s390x                      Docker image for mysql-s390x

nasqueron/mysql
                                                    [OK]
newrelic/mysql-plugin                   New Relic Plugin for monitoring MySQL databa
                                                   [OK]
vitess/mysqlctld                        vitess/mysqlctld
                                                    [OK]
silintl/mysql-backup-restore            Simple docker image to perform mysql backups
                                                   [OK]
docksal/mysql                           MySQL service images for Docksal - https://d
```

Type the following command and press *Enter*:

```
$ docker run hello-world
```

Docker will download and run the "Hello world" container. A confirmation message will be displayed in the terminal.

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

This indicates that your Docker installation is successful.

docker pull :

```
$ docker pull <image name>
```

This command is used to pull images from the docker repository(hub.docker.com)

MINGW64:/c/Program Files/Docker Toolbox

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

Docker Commands

$docker pull

Now that we know the name of the image, we can pull that from the Docker hub using the command docker pull. Here, we are setting the platform option as well.

$ docker pull --platform linux/x86_64 mysql

Docker Commands

Tags are used to identify images inside a repository. If we don't specify a tag Docker engine uses the **:latest** tag by default. So, in the previous example, Docker pulled the **mysql:latest** image.

Docker Commands

If our application depends on a specific version of an image, we can specify that using a tag name.

$docker pull --platform linux/arm64/v8 mysql:5.6

Docker Commands

Since we can have multiple images under one repository, we can pull all the images using the --all-tags option. The following command will pull all the images from the mysql repository.

```
$ docker pull --all-tags mysql
```

Docker Commands

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker pull --all-tags mysql
51f5c6a04d83: Pull complete
a3ed95caeb02: Pull complete
260d7505d8f9: Pull complete
a65f47c75fe3: Pull complete
729d0217f8db: Pull complete
b947959cc91d: Pull complete
b47bda57598f: Pull complete
ca4716b2a2da: Extracting   26.2MB/26.2MB
6bfeb4cf0b17: Download complete
1ea53cc4e6ff: Download complete
586629f61f81: Download complete
abcdaaf08d0f: Pull complete
b947959cc91d: Downloading  85.85kB/8.213MB
5-debian: Pulling from library/mysql
ac2fb615420c: Pull complete
c67721b86bd6: Pull complete
8a459e3867bf: Pull complete
4146b33aaf1f: Pull complete
51f5c6a04d83: Extracting   21.5MB/51.36MB
7eae6e50dbb1: Pull complete
ba7a8db4d7e2: Pull complete
54cbb2253505: Pull complete
2fa6b64c13ff: Pull complete
29961c6b0e84: Pull complete
51f5c6a04d83: Extracting  18.35MB/51.36MB
Digest: sha256:367d1dbfd4483258ef7a652728e5b9952fbcecb85279e402aa9e5d87de8231a0
5-oracle: Pulling from library/mysql
Digest: sha256:bbe0e2b0a33ef5c3a983e490dcb3c1a42d623db1d5679e82f65cce3f32c8f254
5.5: Pulling from library/mysql
743f2d6c1f65: Pull complete
3f0c413ee255: Pull complete
aef1ef8f1aac: Pull complete
```

Downloading image

Let's say you need to pull the docker image from dockerhub (docker repository). The following example of pulling the Apache HTTP server image.

$ docker pull httpd

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
26c5c85e47da: Pull complete
2d29d3837df5: Pull complete
2483414a5e59: Pull complete
e78016c4ba87: Pull complete
757908175415: Pull complete
Digest: sha256:a182ef2350699f04b8f8e736747104eb273e255e818cd55b6d7aa50a1490ed0c
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

Docker Commands

$ docker images

Brilliant, now we should have some images in our local machine, and to confirm, let's run the following command to list all the local images.

$ docker images

Docker Commands

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hello-world         latest              feb5d9fea6a5        19 months ago
13.3kB

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

Docker Commands

```
MINGW64:/c/Program Files/Docker Toolbox

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
httpd               latest              4b7fc736cb48        2 weeks ago
145MB
hello-world         latest              feb5d9fea6a5        19 months ago
13.3kB

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

Docker Commands

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
mysql               5                   459651132a11        8 days ago
429MB
mysql               5-oracle            459651132a11        8 days ago
429MB
mysql               5-debian            c39b1bfebc65        9 days ago
462MB
mysql               5.6                 dd3b2a5dcb48        7 months ago
303MB
mysql               5.5                 d404d78aa797        3 years ago
205MB
mysql               5.5.62              d404d78aa797        3 years ago
205MB
mysql               5.5.61              d306c8812ec3        3 years ago
205MB
mysql               5.5.60              2cd7ceff3e53        3 years ago
205MB
mysql               5.5.59              0da48351c371        4 years ago
205MB
mysql               5.5.58              1d35aff2d8ca        4 years ago
257MB
mysql               5.5.57              6929c9b0e536        4 years ago
257MB
mysql               5.5.56              17e9a1fe381e        5 years ago
256MB
mysql               5.5.55              c0fb485deb70        5 years ago
256MB
mysql               5.5.54              9913483658d3        5 years ago
256MB
mysql               5.5.53              0e21a45a5660        5 years ago
255MB
mysql               5.5.52              402df65b897e        5 years ago
255MB
mysql               5.5.51              51ae120c924c        5 years ago
```

docker run :

```
$ docker run -it -d <image name>
```

This command is used to create a container from an image.

docker run :

$ docker run -it -d <image name>

```
MINGW64:/c/Program Files/Docker Toolbox
$ docker run -it -d hello-world
3515898f679714beeda2475e2ef834ec24c9929af156506043d2422e173a9e3f

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

docker run :

$ docker run -it -d <image name>

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it -d httpd
b26d375374c4e76ccfa2ae9f438407bff5514221e5dbd0191c2add8fd062454a

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

docker run :

$ docker run -it -d <image name>

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it -d hello-world
188514e947c9772fc568b2b970fcd3925425f7873a26d76ca6101d422cabf2fc
```

docker ps :

This command is used to list the running containers.

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
 STATUS              PORTS               NAMES
b26d375374c4        httpd               "httpd-foreground"       2 minutes ago
 Up 2 minutes        80/tcp              hardcore_hertz
```

docker ps -a

This command is used to show all the running and exited containers.

```
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
  STATUS                          PORTS                    NAMES
271154eeba3c        hello-world         "/hello"                 About a minute ago
  Exited (0) About a minute ago                            xenodochial_burnell
b26d375374c4        httpd               "httpd-foreground"       4 minutes ago
  Up 4 minutes                    80/tcp                   hardcore_hertz
188514e947c9        hello-world         "/hello"                 24 hours ago
  Exited (0) 24 hours ago                                  flamboyant_tharp
d6e01391ea67        hello-world         "/hello"                 24 hours ago
  Exited (0) 24 hours ago                                  priceless_burnell
3515898f6797        hello-world         "/hello"                 25 hours ago
  Exited (0) 25 hours ago                                  pedantic_bell
82fdb423fcba        hello-world         "/hello"                 25 hours ago
  Exited (0) 25 hours ago                                  objective_mcclintock
```

Docker Commands

$ docker rmi

Finally, if we want to free some disk space, we can use the docker rmi command with the image id to remove an image.

$ docker rmi eb0e825dc3cf

Docker Commands

```
$ docker rmi 663a514cfc40
Untagged: mysql:5.5.40
Untagged: mysql@sha256:d06722026288205788 1adc38a8b6df837a196304e0cbbb9591bf53859
40bf9ea
Deleted: sha256:663a514cfc40264d4aa94d51ab93a0e54bbfbc2c1daccb2e4f145bd15f03b7f2
Deleted: sha256:5a1d7acceac0dd1aefaa4074e21240791201a8c4193380e15df5651a4354207d
Deleted: sha256:29ed17ccc01f4c7ce5b24d19f96ed57d3ea4ecf77a5d4cce18e24e47dd2ec55e
Deleted: sha256:8929a9964fe75d1b8ec8240c5233503d6b8b352d6d2a477fd511ad226ba4e3f8
Deleted: sha256:e71cfffb360416ddb74d76c9797f99149b7c94cad2d78d00fbb4e286690542c3
Deleted: sha256:cab32a3b1eefcb074bba1ce4232a886e09a33ef79934381ff56dc4ad7ed69a89
Deleted: sha256:ffa412ce038b43aedb8d1a5e7229a2e776c6b4c04c69f2c7f6c1f0ec6a662525
Deleted: sha256:602c138c778a235f3d2ee5ec0b8860e84c16439f458b132b9f62db929cfebb35
Deleted: sha256:a8f97dede9e0f57c658453aff986b94827d57718031724e088e9901a8428839b
Deleted: sha256:e78dc7fdc3b2ecc919412aaa2035b2695152d07553eb7ed97229a20a9eb3bb67
Deleted: sha256:b8d8069586890a3b3d858dd2f7d0430c1afc2c02e5173471a0025a9a24e7d5ff
Deleted: sha256:f17e492b213dd40b5c72a036c13398f51e652d2d30a77e8968867d8dd778935e
Deleted: sha256:dc991dfc68f0bd2e03e7ed8ced4eb35e667b7c39a385801de1ec2241b68ab422
Deleted: sha256:a8eaa5bfac2a12e437eec1ca8eed232c0bb512fb201a6d66594d9bec13a955a1
Deleted: sha256:4bb5e862604c15b5b2198703e9016ee25324977ef197030cf1c57cb9a0abae37
Deleted: sha256:f45e3c95eefc3a994bf64676e01aea69ee32e7f2a73ab4c97be17a90d46b1666
Deleted: sha256:3a6a49f265a09cd6d85237a96c79dfb7464f9b959ed75a1fcc8347dc1f787f6b
```

Docker Commands

$ docker command_name –help

These commands come with plenty of helpful options. If you want to know about other available options, run the **docker command_name --help** command. For

example:

$ docker logs --help

Docker Commands

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker images --help

Usage:  docker images [OPTIONS] [REPOSITORY[:TAG]]

List images

Options:
  -a, --all             Show all images (default hides intermediate images)
      --digests         Show digests
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print images using a Go template
      --no-trunc        Don't truncate output
  -q, --quiet           Only show numeric IDs

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run --env MYSQL_ROOT_PASSWORD=my-secret-pw --detach mysql --name phund
e
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
e54b73e95ef3: Pull complete
327840d38cb2: Pull complete
642077275f5f: Pull complete
e077469d560d: Pull complete
cbf214d981a6: Pull complete
7d1cc1ea1b3d: Pull complete
d48f3c15cb80: Pull complete
94c3d7b2c9ae: Pull complete
f6cfbf240ed7: Pull complete
e12b159b2a12: Pull complete
4e93c6fd777f: Pull complete
Digest: sha256:152cf187a3efc56afb0b3877b4d21e231d1d6eb828ca9221056590b0ac834c75
Status: Downloaded newer image for mysql:latest
588e31f5cd0c60e2b8db8037688129178ef497dd609d878732be23d0475c32e6

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

Openjdk

```
$ docker pull openjdk

$ docker pull python
```

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker pull python
Using default tag: latest
latest: Pulling from library/python
b0248cf3e63c: Pull complete
127e97b4daf7: Pull complete
0336c50c9f69: Pull complete
1b89f3c7f7da: Pull complete
2d6277217976: Pull complete
273fcda609d8: Pull complete
58568d3a3a00: Pull complete
56fc9fb54f6e: Pull complete
8a22f29afe36: Pull complete
Digest: sha256:f7382f4f9dbc51183c72d621b9c196c1565f713a1fe40c119d215c96
Status: Downloaded newer image for python:latest
docker.io/library/python:latest

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker pull openjdk
Using default tag: latest
latest: Pulling from library/openjdk
197c1adcd755: Pull complete
57b698b7af4b: Pull complete
95a27dbe0150: Pull complete
Digest: sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480b3f1f
Status: Downloaded newer image for openjdk:latest
docker.io/library/openjdk:latest

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

$ docker images

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED
    SIZE
<none>                      <none>              2daff1f5f3ce        21 hours ago
    41MB
mysql                       5-debian            da1f6f04d288        11 days ago
    463MB
mysql                       5                   dd6675b5cfea        11 days ago
    569MB
mysql                       5-oracle            dd6675b5cfea        11 days ago
    569MB
python                      latest              4665a951a37e        2 weeks ago
    921MB
httpd                       latest              4b7fc736cb48        2 weeks ago
    145MB
nginx                       alpine              8e75cbc5b25c        4 weeks ago
    41MB
openjdk                     latest              71260f256d19        2 months ago
    470MB
pegi3s/kakscalculator       latest              0b21c5fa72d1        12 months ago
    320MB
hello-world                 latest              feb5d9fea6a5        19 months ago
    13.3kB
hwasurr/calculator          latest              ac21b39b7019        22 months ago
    123MB
mysql                       5.5                 d404d78aa797        3 years ago
    205MB
mysql                       5.5.44              ebe6ba90997b        7 years ago
    215MB
```

Container Start :

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run python

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        IMAGE                   COMMAND                  CREATED
         STATUS                      PORTS                   NAMES
56627ccec11d        python                  "python3"                31 seconds
ago       Exited (0) 31 seconds ago                           blissful_chatterjee
a6914b4bc33e        nginx:alpine            "/docker-entrypoint."    21 hours ago
        Exited (255) 8 minutes ago   0.0.0.0:80->80/tcp    nginx_base
24eb78afee71        pegi3s/kakscalculator   "/bin/bash"              25 hours ag
o         Exited (255) 21 hours ago                           jovial_brattain
978a6b211107        httpd                   "httpd-foreground"       25 hours ag
o         Exited (255) 21 hours ago    80/tcp                 stupefied_yalow
6b672146c484        hello-world             "/hello"                 25 hours ag
o         Exited (0) 25 hours ago                             admiring_euclid
994304b929b2        pegi3s/kakscalculator   "/bin/bash"              25 hours ag
o         Exited (0) 25 hours ago                             sad_poincare
68ee4a55e020        hello-world             "/hello"                 25 hours ag
o         Exited (0) 25 hours ago                             pensive_maxwell
271154eeba3c        hello-world             "/hello"                 45 hours ag
o         Exited (0) 45 hours ago                             xenodochial_burnell
b26d375374c4        httpd                   "httpd-foreground"       45 hours ag
o         Exited (255) 26 hours ago    80/tcp                 hardcore_hertz
188514e947c9        hello-world             "/hello"                 2 days ago
          Exited (0) 2 days ago                               flamboyant_tharp
d6e01391ea67        hello-world             "/hello"                 2 days ago
          Exited (0) 2 days ago                               priceless_burnell
3515898f6797        hello-world             "/hello"                 2 days ago
          Exited (0) 2 days ago                               pedantic_bell
82fdb423fcba        hello-world             "/hello"                 2 days ago
          Exited (0) 2 days ago                               objective_mcclintock

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

Container Start :

$ docker run python

$ docker ps

$ docker ps –a

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run python

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        IMAGE                   COMMAND                  CREATED
         STATUS                      PORTS                NAMES
56627ccec11d        python                  "python3"                31 seconds
ago      Exited (0) 31 seconds ago                                   blissful_chatterjee
a6914b4bc33e        nginx:alpine            "/docker-entrypoint."    21 hours ago
        Exited (255) 8 minutes ago   0.0.0.0:80->80/tcp   nginx_base
24eb78afee71        pegi3s/kakscalculator   "/bin/bash"              25 hours ag
o        Exited (255) 21 hours ago                                   jovial_brattain
978a6b211107        httpd                   "httpd-foreground"       25 hours ag
o        Exited (255) 21 hours ago    80/tcp                         stupefied_yalow
6b672146c484        hello-world             "/hello"                 25 hours ag
o        Exited (0) 25 hours ago                                     admiring_euclid
994304b929b2        pegi3s/kakscalculator   "/bin/bash"              25 hours ag
o        Exited (0) 25 hours ago                                     sad_poincare
68ee4a55e020        hello-world             "/hello"                 25 hours ag
o        Exited (0) 25 hours ago                                     pensive_maxwell
271154eeba3c        hello-world             "/hello"                 45 hours ag
o        Exited (0) 45 hours ago                                     xenodochial_burnell
b26d375374c4        httpd                   "httpd-foreground"       45 hours ag
o        Exited (255) 26 hours ago    80/tcp                         hardcore_hertz
188514e947c9        hello-world             "/hello"                 2 days ago
         Exited (0) 2 days ago                                       flamboyant_tharp
d6e01391ea67        hello-world             "/hello"                 2 days ago
         Exited (0) 2 days ago                                       priceless_burnell
3515898f6797        hello-world             "/hello"                 2 days ago
         Exited (0) 2 days ago                                       pedantic_bell
82fdb423fcba        hello-world             "/hello"                 2 days ago
         Exited (0) 2 days ago                                       objective_mcclintock

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREAT
    SIZE
<none>                      <none>              2daff1f5f3ce        21 ho
    41MB
mysql                       5-debian            da1f6f04d288        11 da
    463MB
mysql                       5                   dd6675b5cfea        11 da
    569MB
mysql                       5-oracle            dd6675b5cfea        11 da
    569MB
python                      latest              4665a951a37e        2 wee
    921MB
httpd                       latest              4b7fc736cb48        2 wee
    145MB
nginx                       alpine              8e75cbc5b25c        4 wee
    41MB
openjdk                     latest              71260f256d19        2 mon
    470MB
pegi3s/kakscalculator       latest              0b21c5fa72d1        12 mo
    320MB
hello-world                 latest              feb5d9fea6a5        19 mo
    13.3kB
hwasurr/calculator          latest              ac21b39b7019        22 mo
    123MB
mysql                       5.5                 d404d78aa797        3 yea
    205MB
mysql                       5.5.44              ebe6ba90997b        7 yea
    215MB

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name pycon -d 4665a951a37e
36b31aab60cb2274208f1e861d2ac2583bb496c0a51d4ff324c8a397dc616c70

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$
```

$ docker images

$ docker run –name pycon –d 4665a951a37e

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name pycon1 -it -d python
28a2b2f44895906928b71d1efa95bc6ab366697eb68caf19b22523f25448e9f3

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
28a2b2f44895        python              "python3"           9 seconds ago
Up 9 seconds                            pycon1

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ _
```

Container created and expired also

$ docker run –name pycon1 –it –d python

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
28a2b2f44895        python              "python3"           About a minu
 Up About a minute                       pycon1

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker exec -it 28a2b2f44895 python3
Python 3.11.3 (main, Apr 12 2023, 14:31:14) [GCC 10.2.1 20210110] on lin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("welcome MCA 2")
welcome MCA 2
>>>
```

$ docker ps

$ docker exec –it 28a2b2f44895 python3

```
the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
 STATUS              PORTS               NAMES
28a2b2f44895        python              "python3"           About a minu
 Up About a minute                       pycon1

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker exec -it 28a2b2f44895 python3
Python 3.11.3 (main, Apr 12 2023, 14:31:14) [GCC 10.2.1 20210110] on lin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("welcome MCA 2")
welcome MCA 2
>>>
```

Run python commands now

>>> print(“welcome MCA 2”)

```
>>> input("x:")
x:45
'45'
>>>
```

>>> Input(“x:”)

```
>>>
>>>
>>> list=[1,2,3,4,5]
>>> for x in list:
...     print(x, end = "")
...
12345>>>
```

```
list= [1,2,3]
for x in list:
    print (x, end = "")
```

```
$ docker run --name jcon -it -d openjdk
8f78de7053232e0c913ce8f4f928fc19555b305175fbd2bbb38ece5c7035483d

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
8f78de705323        openjdk             "jshell"            7 seconds ag
Up 7 seconds                            jcon
28a2b2f44895        python              "python3"           26 minutes a
Up 26 minutes                           pycon1

the director@computer169 MINGW64 /c/Program Files/Docker Toolbox
$ docker exec -it jcon jshell
|  Welcome to JShell -- Version 18.0.2.1
|  For an introduction type: /help intro

jshell>
```

```
jshell> System.out.println("Welcome MCA 2")
Welcome MCA 2

jshell>
```

Revision

```
$ docker ps –a

$ docker pull openjdk
$ docker images
$ docker run –name java1 –it –d openjdk
$ docker ps -a
$ docker exec –it java1 jshell
```

Revision

$ docker restart container_name

Ex :

$ docker restart java1

Revision

$ docker stop container_name

Ex :

$ docker stop java1

Revision

$ docker login
$ docker commit
$ docker push

Ex :

This command will allow to push image to docker hub

Revision

$ docker copy
$ docker volume
$ docker logout

How to create your own image ?

How to create your own image ?

1] docker pull Ubuntu

2] docker run --name u1 -it ubuntu

3] git --version

4] apt-get update

5] apt-get install –y git

6] git --version

How to create your own image ?

7] exit

8] docker rm –f u1

9] docker run --name u1 -it Ubuntu

10] git --version

How to create your own image ?

create customize new images

1] **docker run --name u1 -it ubuntu**

2] apt-get update

3] apt-get install –y git

4] git –version

5] exit

6] docker commit u1 myubuntu

7] docker rm –f u1

8] docker run --name u1 -it myubuntu

How to create your own image by using docker file?

```
FROM ubuntu
MAINTAINER sangita
<sangita.phunde@rediffmail.com>
RUN apt-get update
CMD ["echo","Hello sangita and MCA"]
```

Note : copy con dockerfile ctrl z to save

1] docker build –t myimg2:1.0 .

2] docker images

3] docker run 7208cb8e9791

Push images to docker hub

1] on command prompt do login

```
C:\> docker login
```

2] docker images

3] docker tag gitubuntu sangitaphunde/ims

4] docker push sangitaphunde/ims

Push images to docker hub

1] on command prompt do login

```
C:\> docker login
```

2] docker images

3] docker tag gitubuntu sangitaphunde/ims

4] docker push sangitaphunde/ims

Push images to docker hub

7] docker tag myimg1:1.0 sangitaphunde/myimg1:1.0

8]docker push sangitaphunde/myimg1:1.0

4	4. Docker– Containers & Build tool- Maven 4.1. Introduction: What is a Docker, Use case of Docker, Platforms for Docker, Dockers vs. Virtualization 4.2. Architecture: Docker Architecture., Understanding the Docker components 4.3. Installation: Installing Docker on Linux. Understanding Installation of Docker on windows. Some Docker commands. Provisioning. 4.4. Docker Hub.: Downloading Docker images. Uploading the images in Docker Registry and AWS ECS, Understanding the containers, Running commands in container. Running multiple containers. 4.5. Custom images: Creating a custom image. Running a container from the custom image. Publishing the custom image. 4.6. Docker Networking: Accessing containers, linking containers, Exposing container ports, Container Routing.	30	15

https://towardsdatascience.com/12-essential-docker-commands-you-should-know-c2d5a7751bb5