DOCKER TUTORIAL

simplilearn

What's in it for you?

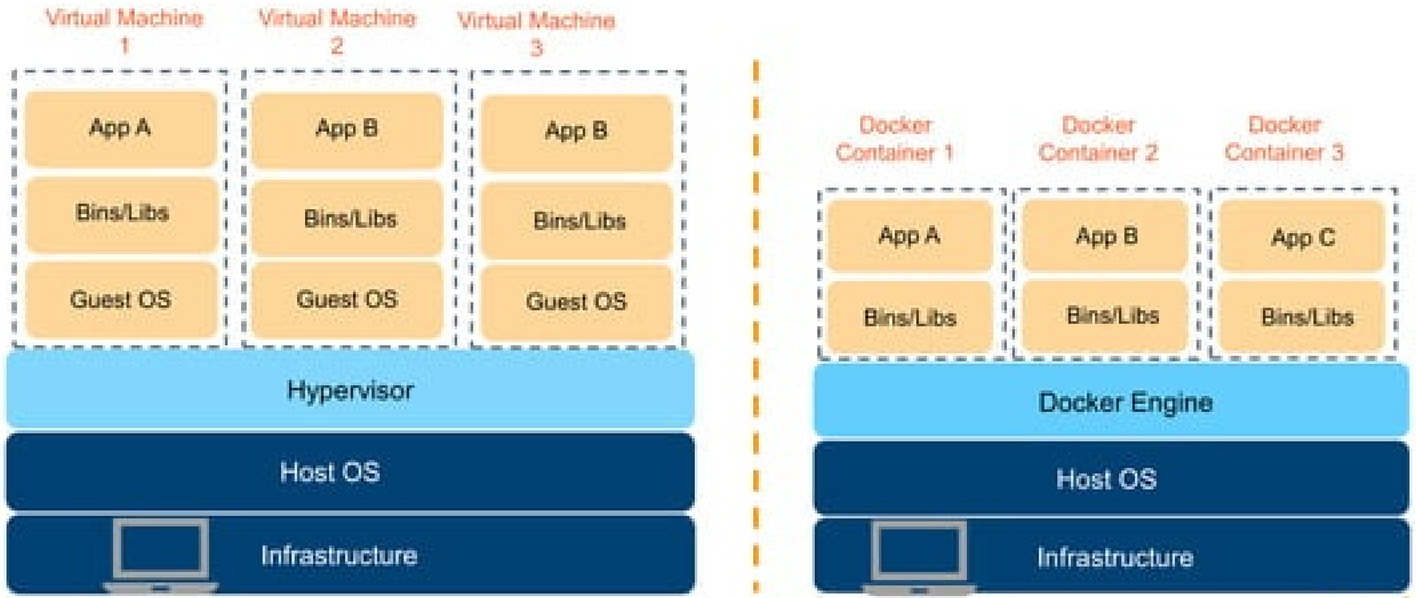
- Virtual Machine vs Docker
- What is Docker?
- Advantages of Docker
- How does Docker work?
- Components of Docker
- Advanced concepts in Docker
- Basic Docker commands
- Demo

Let's get started

Virtual Machine vs Docker

simplilearn

Virtual Machine vs Docker



Virtual Machine vs Docker

Major differences are:

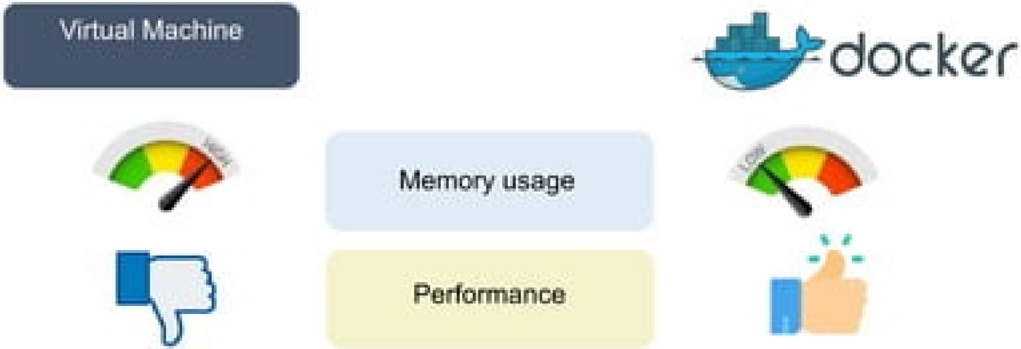
Virtual Machine

docker

Memory usage

Virtual Machine vs Docker

Major differences are:



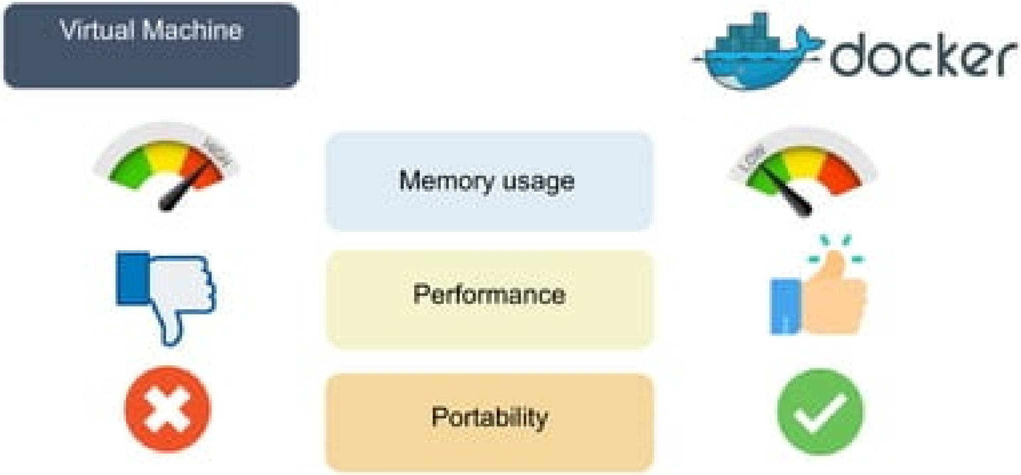
Virtual Machine

Memory usage

Performance

Virtual Machine vs Docker

Major differences are:



Virtual Machine

docker

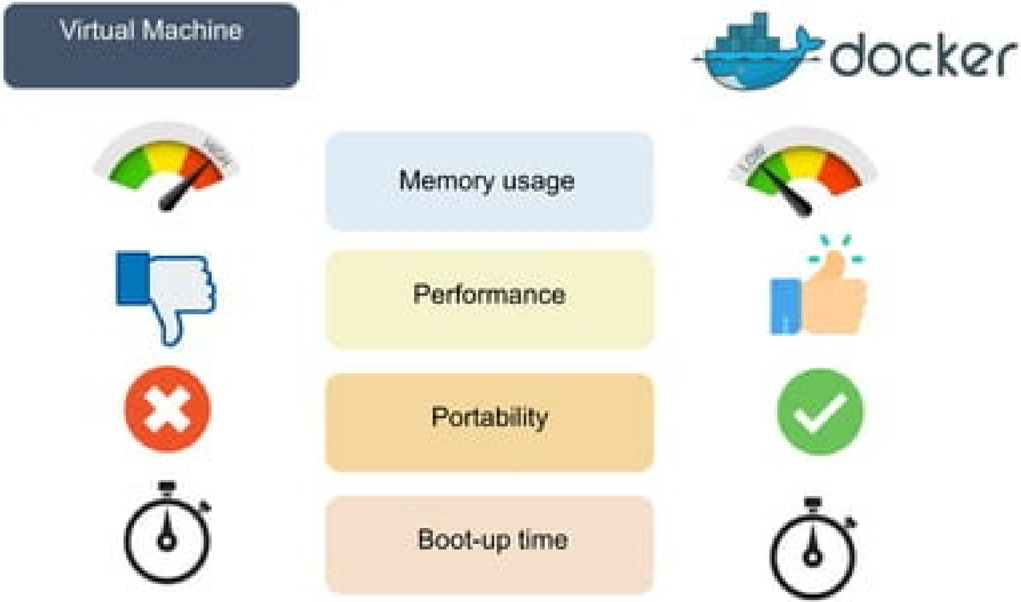
Memory usage

Performance

Portability

Virtual Machine vs Docker

Major differences are:



Virtual Machine

docker

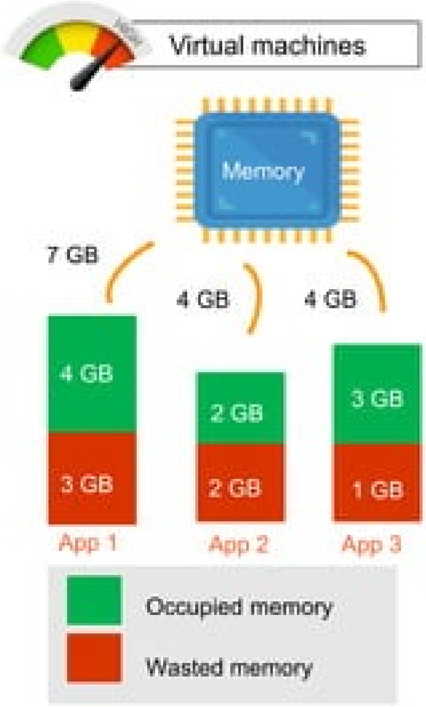
Memory usage

Performance

Portability

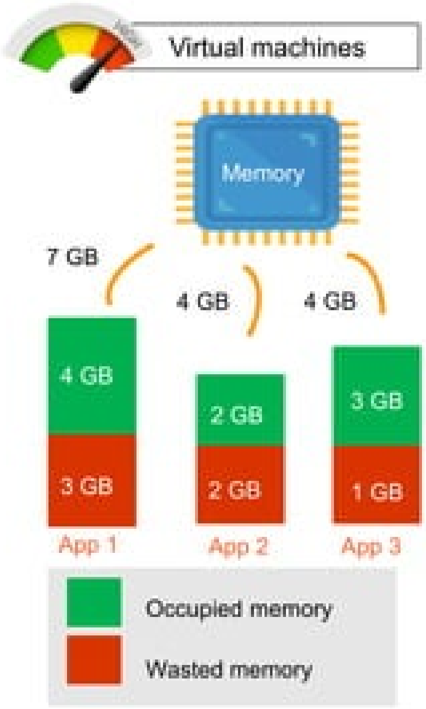
Boot-up time

Virtual Machine vs Docker



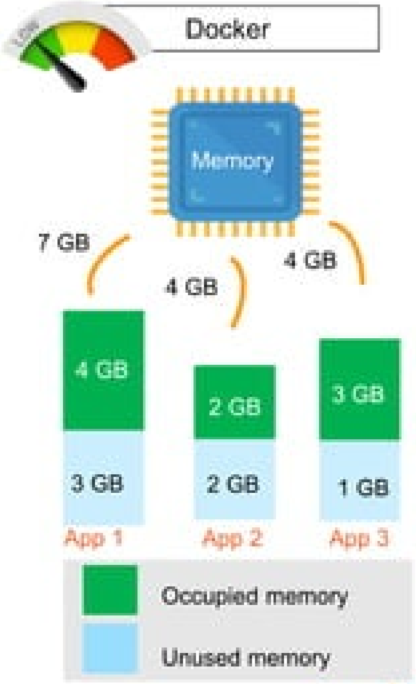
VM - Only 9 GB of memory is used whereas the remaining 6 GB of unused memory cannot be reused

Virtual Machine vs Docker



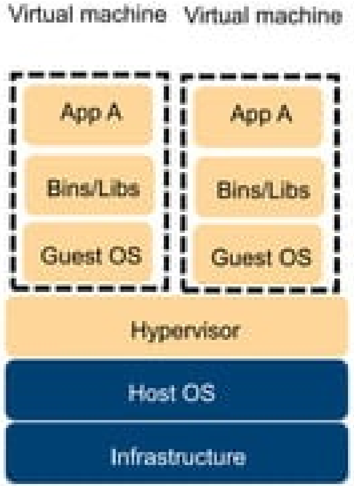
VM - Only 9 GB of memory is used whereas the remaining 6 GB of unused memory cannot be reused

Docker - Only 9 GB of memory is used whereas the remaining 6 GB of memory can be reused for a new container



Virtual Machine vs Docker

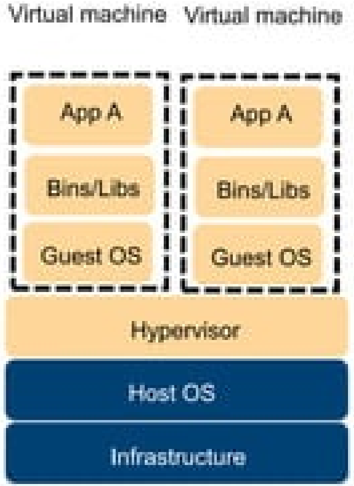
Virtual machines



VM - Running multiple virtual machines leads to unstable performance

Virtual Machine vs Docker

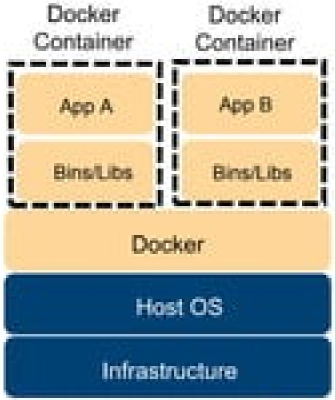
Virtual machines



VM - Running multiple virtual machines leads to unstable performance

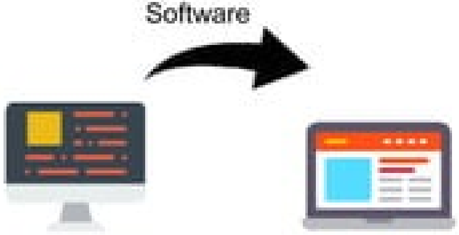
Docker - Containers have a better performance as they are hosted on a single Docker engine

Docker



Virtual Machine vs Docker

✖ Virtual machines



Software

Software works on system A

The same software doesn't work on system B

VM - Portability issues while executing applications in different platforms

Virtual Machine vs Docker

☒ Virtual machines

☑ Docker

Software works on system A

The same software doesn't work on system B

VM - Portability issues while executing applications in different platforms

Docker – Multiple software can be encapsulated in a single container and can be easily deployed to different platforms

Virtual Machine vs Docker

Virtual machines

VM – Takes long boot-up time (minutes)

Virtual Machine vs Docker

Virtual machines

Docker

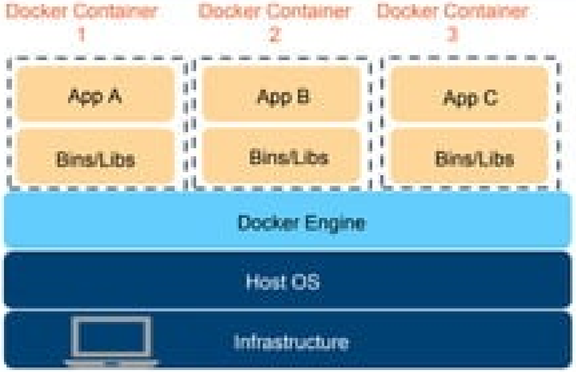
VM – Takes long boot-up time (minutes)

Docker – Takes less boot-up time (milliseconds)

What is Docker?

What is Docker?

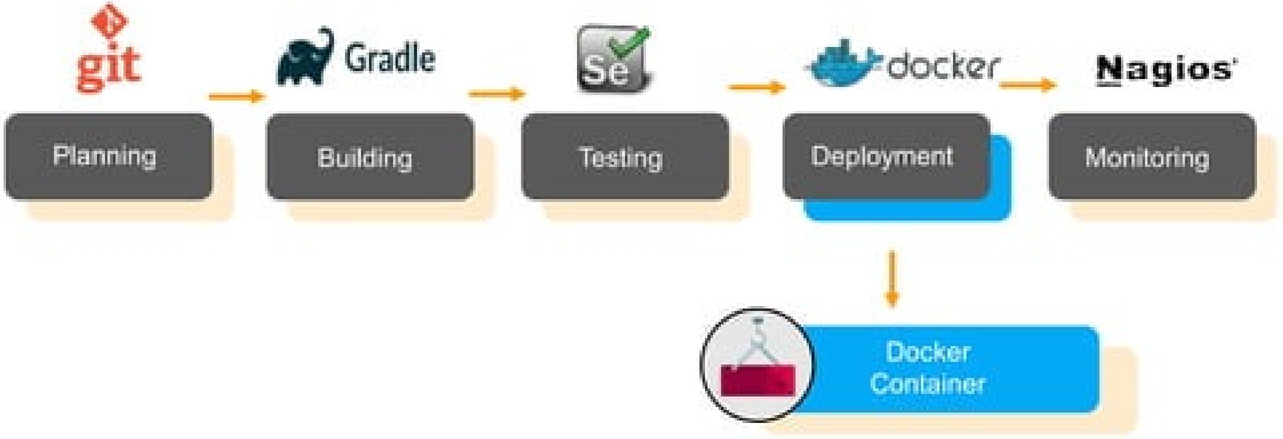
Docker is an OS-level virtualization software platform that enables developers and IT administrators to create, deploy and run applications in a Docker Container with all their dependencies



Note: Docker Container is a lightweight software package that includes all the dependencies (frameworks, libraries, etc.) required to execute an application

What is Docker?

Where is Docker used in DevOps?



Advantages of Docker

Advantages of Docker

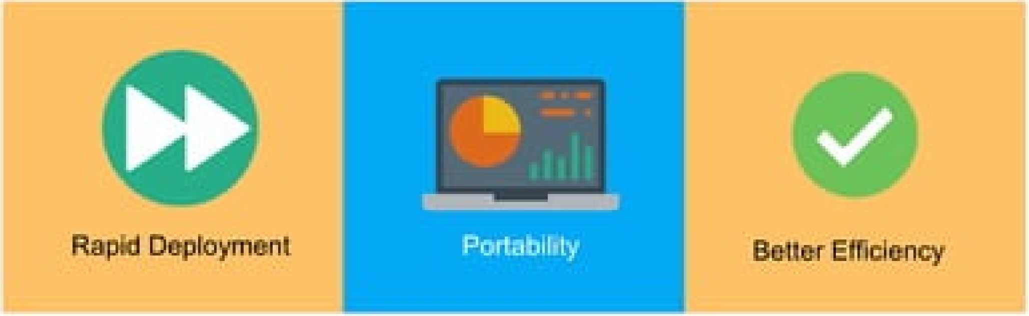
Rapid Deployment

Advantages of Docker

Rapid Deployment

Portability

Advantages of Docker



Rapid Deployment

Portability

Better Efficiency

Advantages of Docker

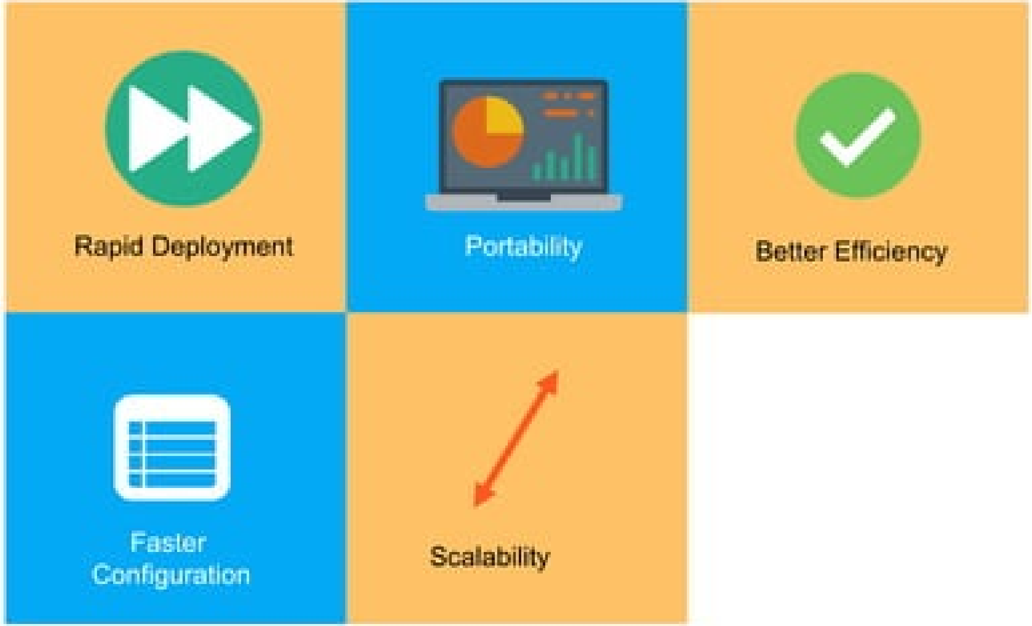
Rapid Deployment

Portability

Better Efficiency

Faster Configuration

Advantages of Docker



Rapid Deployment

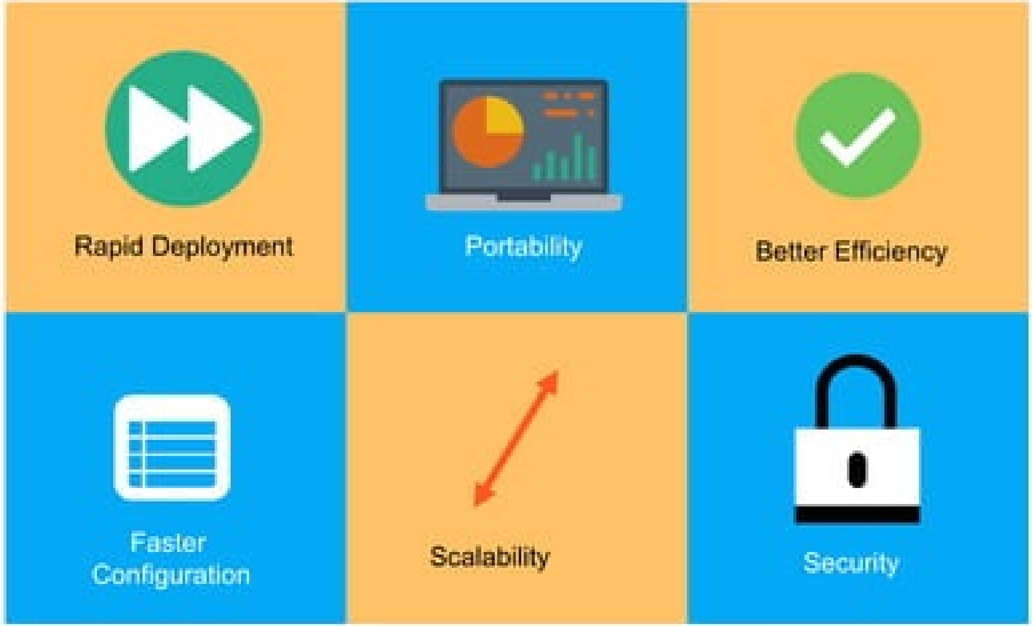
Portability

Better Efficiency

Faster Configuration

Scalability

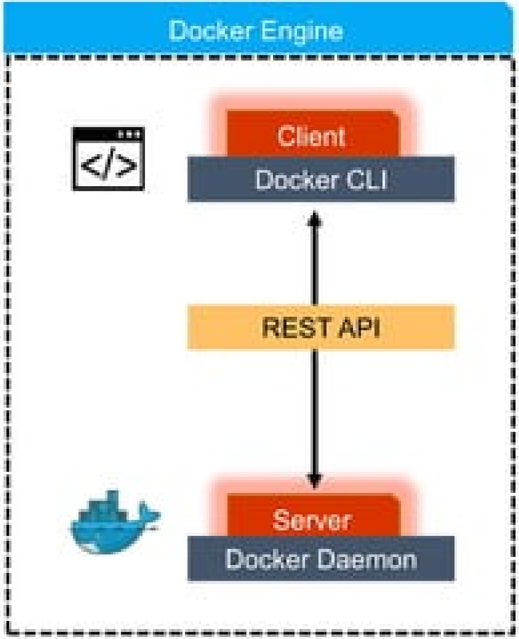
Advantages of Docker



- Rapid Deployment
- Portability
- Better Efficiency
- Faster Configuration
- Scalability
- Security

How does Docker work?

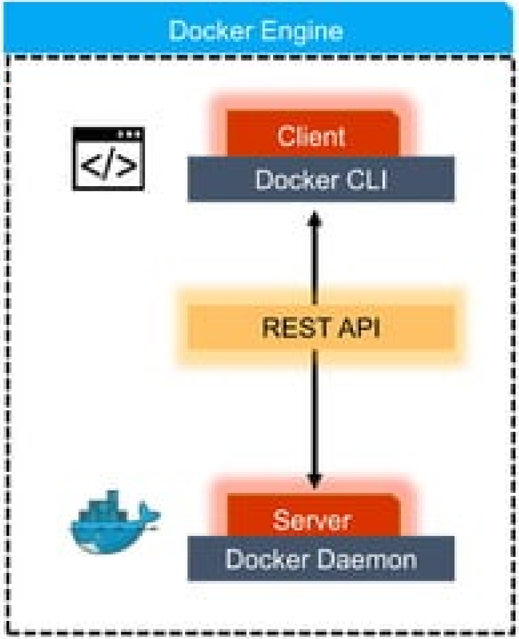
How does Docker work?



Docker Engine

- Docker engine or Docker is a client server application that builds and executes containers using Docker components

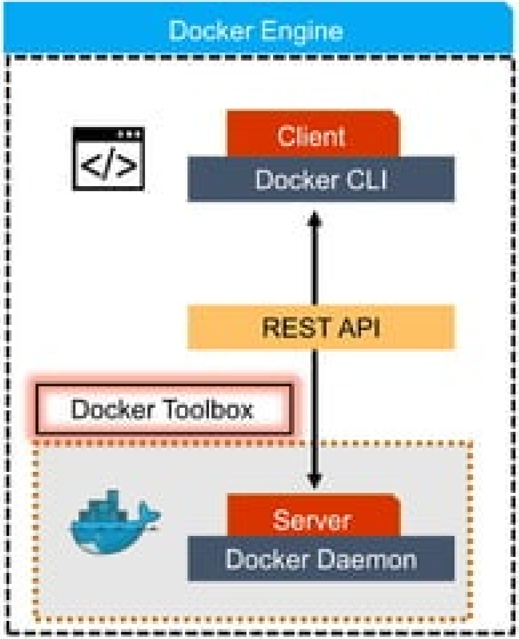
How does Docker work?



Docker Engine

- Docker engine or Docker is a client server application that builds and executes containers using Docker components
- REST API is a primary mode of communication between Docker Client and Docker Daemon

How does Docker work?



Docker Engine

Client

Docker CLI

REST API

Docker Toolbox

Server

Docker Daemon

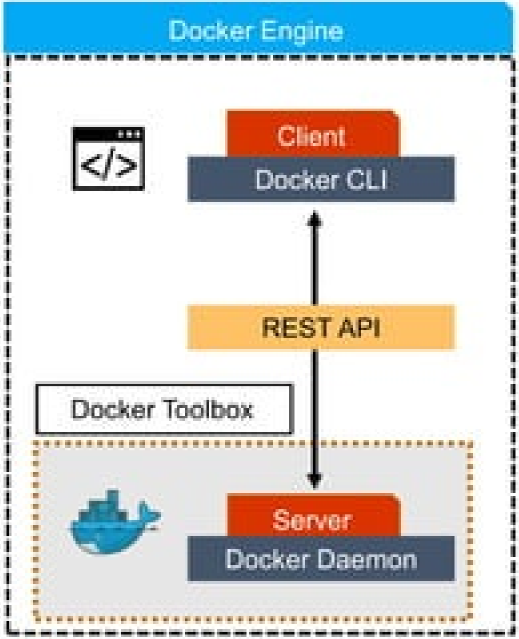
Docker Engine

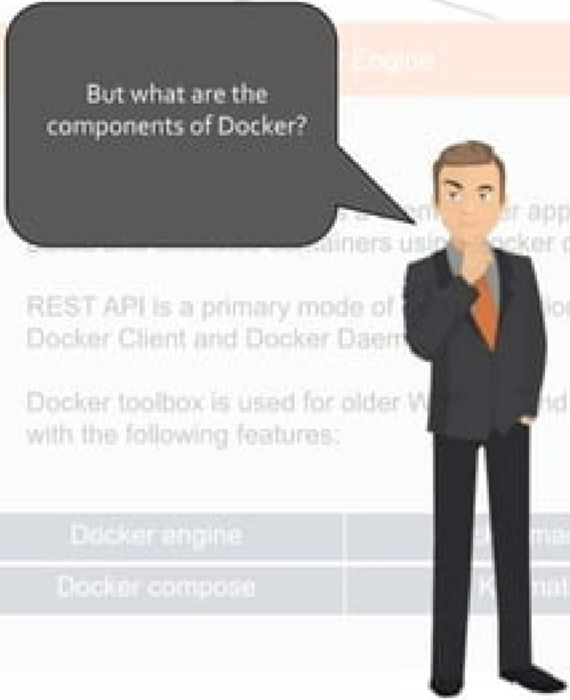
- Docker engine or Docker is a client server application that builds and executes containers using Docker components
- REST API is a primary mode of communication between Docker Client and Docker Daemon
- Docker toolbox is used for older Windows and Mac systems with the following features:

Docker engine	Docker machine
Docker compose	Kitematic

simplilearn

How does Docker work?





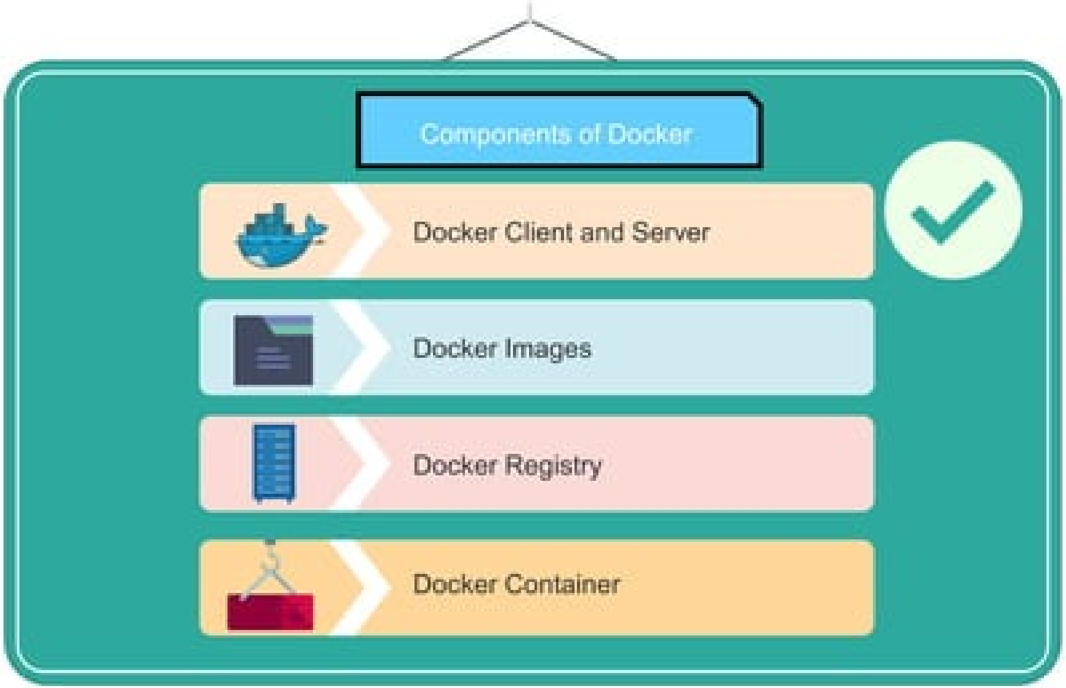
[illegible] Engine

- [illegible] r application that [illegible] ainers us[illegible] ocker components
- REST API is a primary mode of [illegible] ion between Docker Client and Docker Daem[illegible]
- Docker toolbox is used for older W[illegible] nd Mac systems with the following features:

Docker engine	[illegible] machine
Docker compose	K[illegible] matic

Components of Docker

Components of Docker



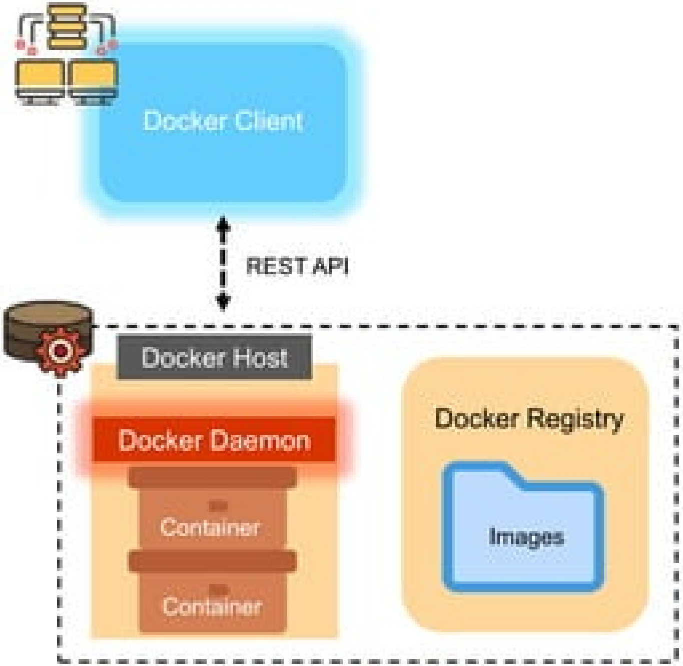
Components of Docker

- Docker Client and Server
- Docker Images
- Docker Registry
- Docker Container

Docker Client and Server - Components of Docker

Docker Client

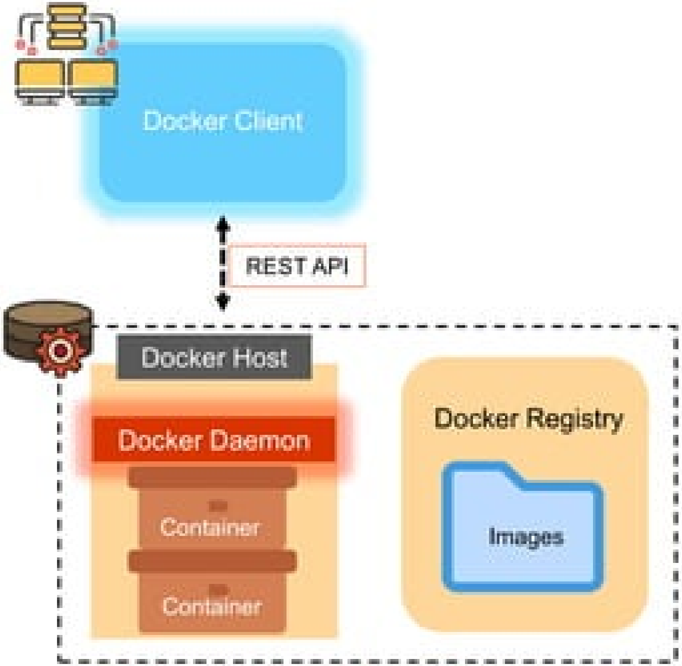
- Docker Client consist of the CLI command which is used to issue commands to the Docker Daemon



Docker Client and Server - Components of Docker

Docker Client

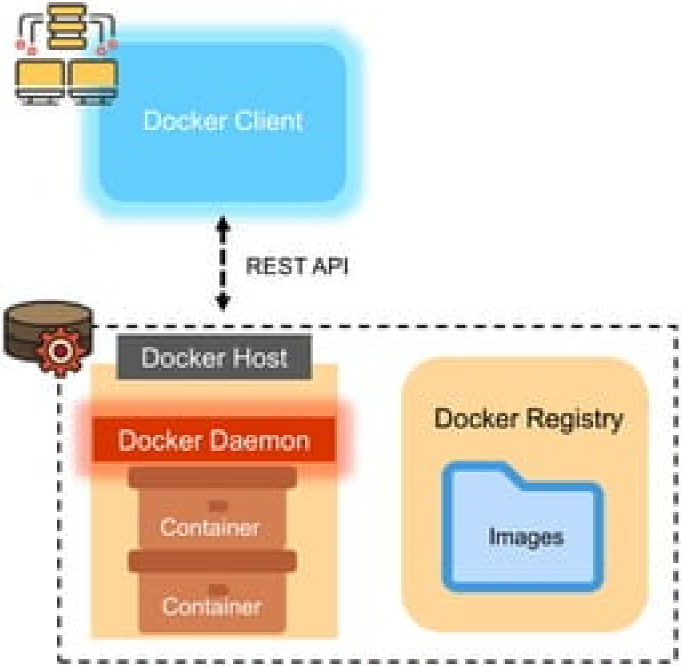
- Docker Client consist of the CLI command which is used to issue commands to the Docker Daemon
- Docker Client uses REST API to issue commands to Docker Daemon through scripting or direct CLI commands



Docker Client and Server - Components of Docker

Docker Client

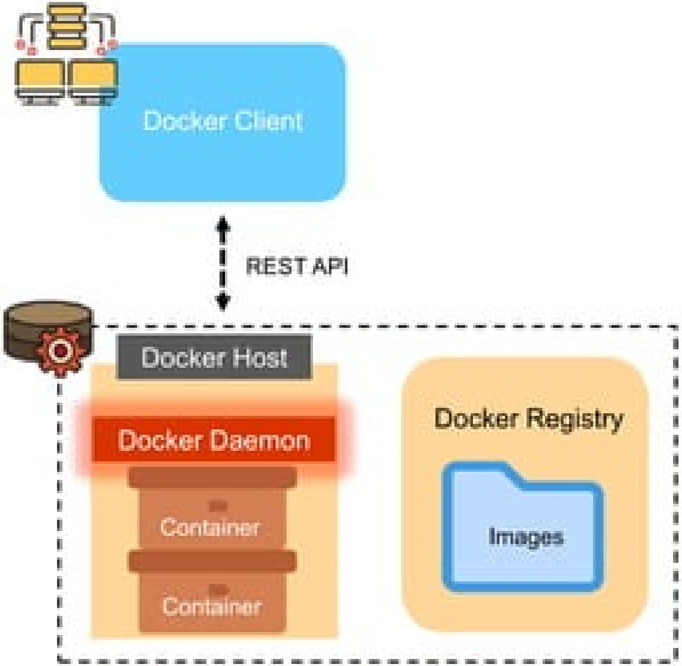
- Docker Client consist of the CLI command which is used to issue commands to the Docker Daemon
- Docker Client uses REST API to issue commands to Docker Daemon through scripting or direct CLI commands
- For example, when you use a *docker pull* command, the client sends this command to daemon, which performs the operation by interacting with other components (Image, Container, Registry)



Docker Client and Server - Components of Docker

Docker Daemon (server)

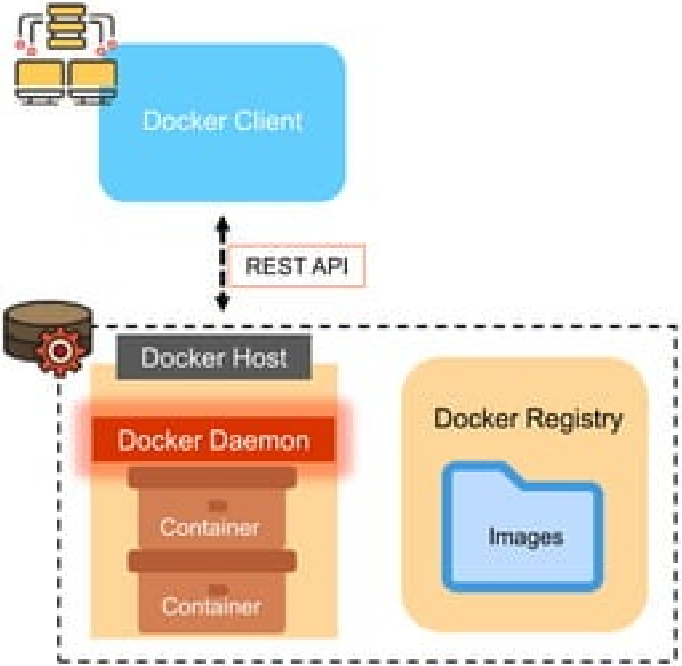
- Docker Daemon is a server which interacts with the operating system and performs all kind of services



Docker Client and Server - Components of Docker

Docker Daemon (server)

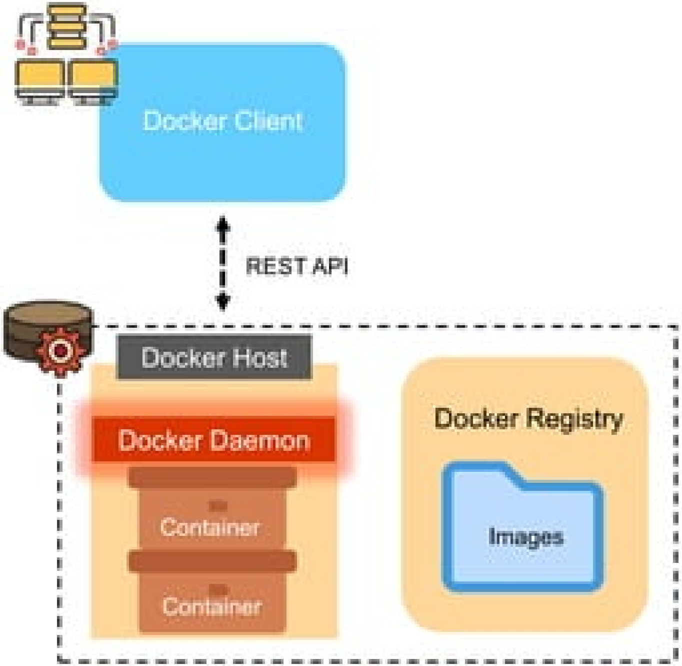
- Docker Daemon is a server which interacts with the operating system and performs all kind of services
- The Docker Daemon listens for REST API request and performs the operation



Docker Client and Server - Components of Docker

Docker Daemon (server)

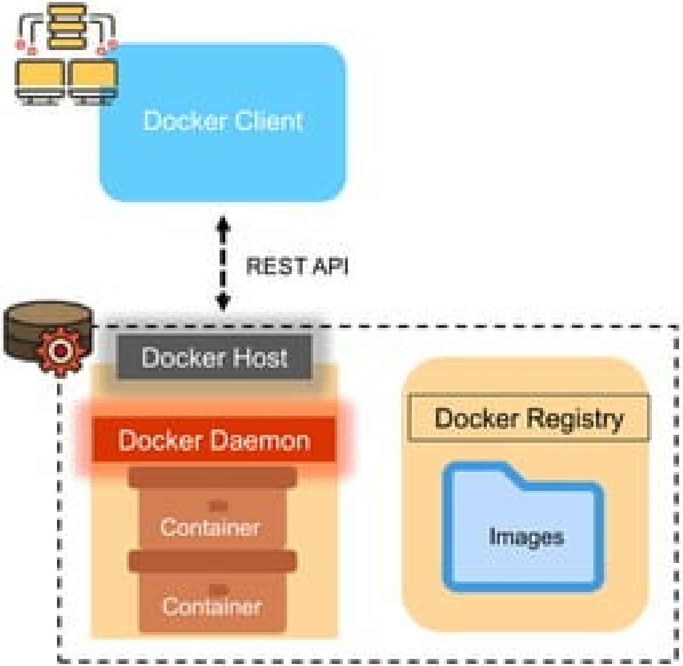
- Docker Daemon is a server which interacts with the operating system and performs all kind of services
- The Docker Daemon listens for REST API request and performs the operation
- A command *dockerd is* used to start a Docker Daemon



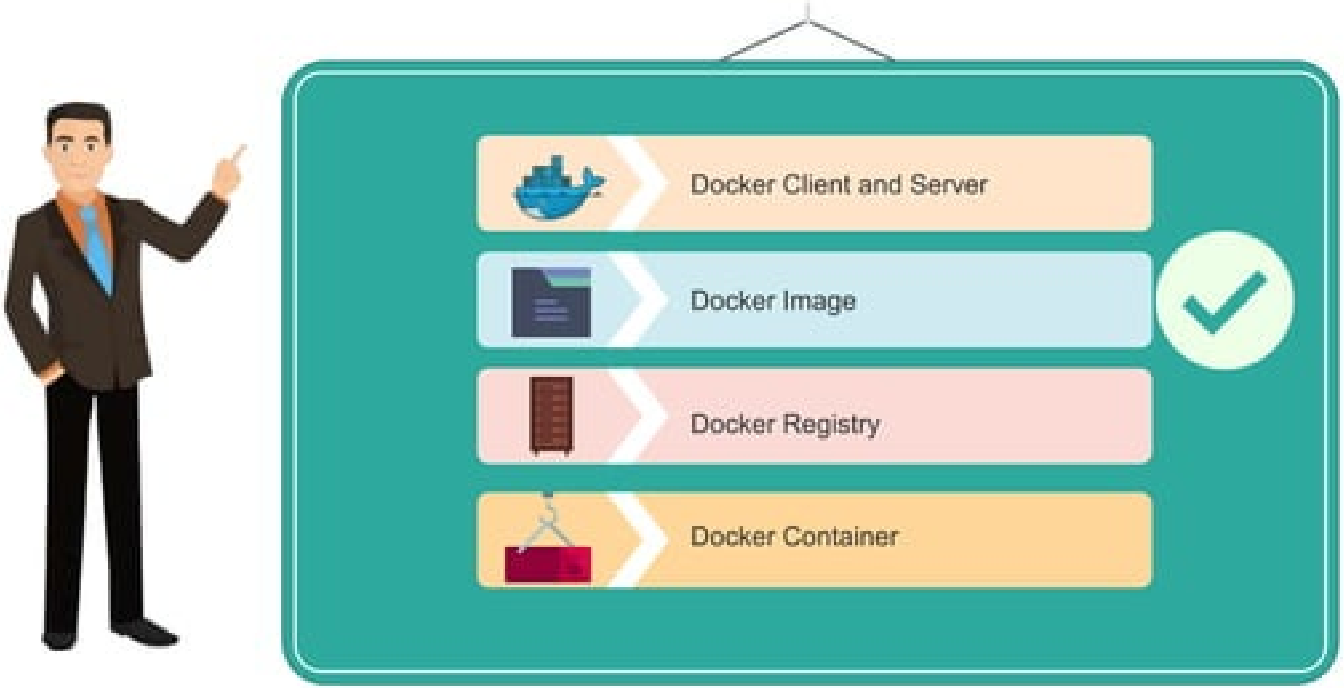
Docker Client and Server - Components of Docker

Docker Daemon (server)

- Docker Daemon is a server which interacts with the operating system and performs all kind of services
- The Docker Daemon listens for REST API request and performs the operation
- A command *dockerd is* used to start a Docker Daemon
- Docker Host runs the Docker Daemon and Registry



Docker Image - Components of Docker

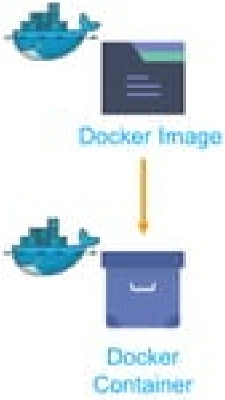


- Docker Client and Server
- Docker Image ✓
- Docker Registry
- Docker Container

Docker Image - Components of Docker

Docker Image

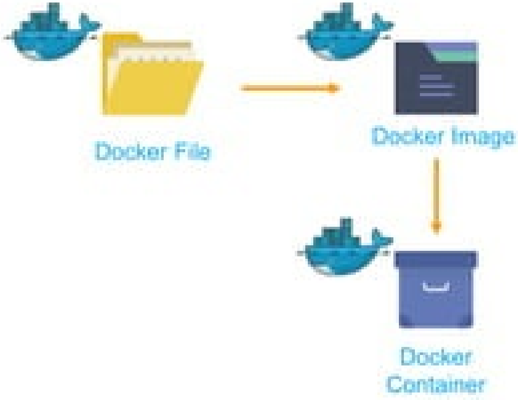
- A Docker Image is a template of instructions which is used to create containers



Docker Image - Components of Docker

Docker Image

- A Docker Image is a template of instructions which is used to create containers
- A Docker Image is built using a file called Docker File

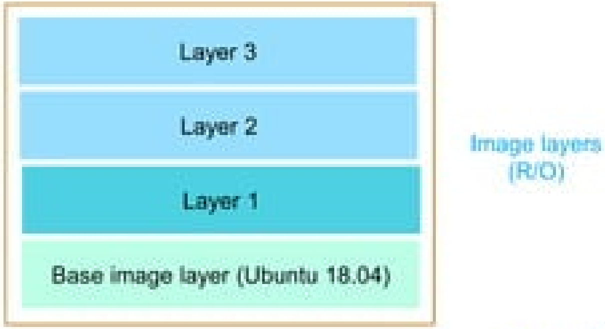


simplilearn

Docker Image - Components of Docker

Docker Image

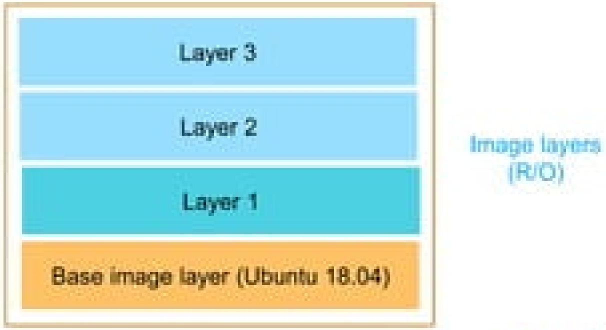
- A Docker Image is a template of instructions which is used to create containers
- A Docker Image is built using a file called Docker File
- It is comprised of multiple layers



Docker Image - Components of Docker

Docker Image

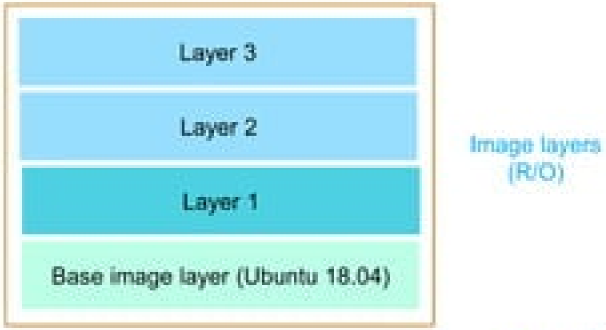
- A Docker Image is a template of instructions which is used to create containers
- A Docker Image is built using a file called Docker File
- It is comprised of multiple layers
- By default, Docker Image starts with a base layer



Docker Image - Components of Docker

Docker Image

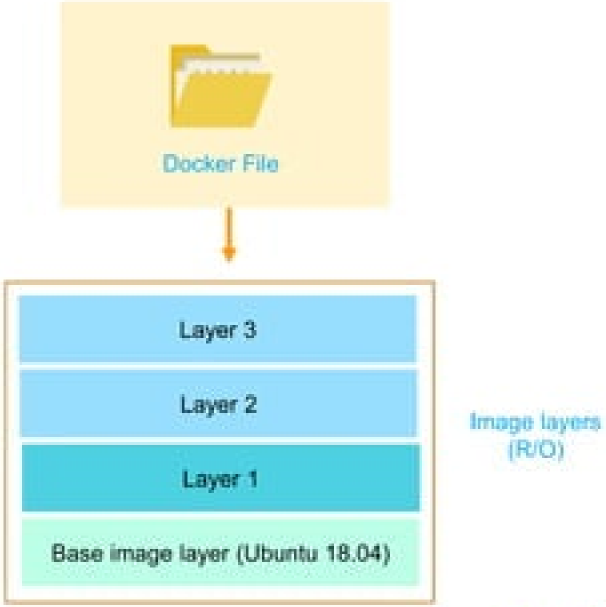
- A Docker Image is a template of instructions which is used to create containers
- A Docker Image is built using a file called Docker File
- It is comprised of multiple layers
- By default, Docker Image starts with a base layer
- Here, each layer depends on the layer below it



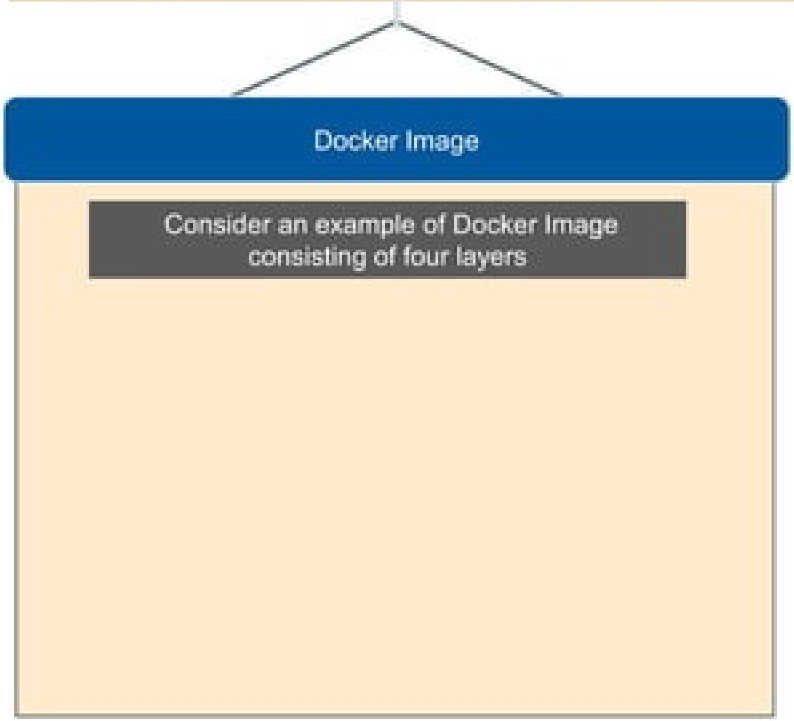
Docker Image - Components of Docker

Docker Image

- A Docker Image is a template of instructions which is used to create containers
- A Docker Image is built using a file called Docker File
- It is comprised of multiple layers
- By default, Docker Image starts with a base layer
- Here, each layer depends on the layer below it
- Image layers are created by executing each command in the Dockerfile and are in the read-only format



Docker Image - Components of Docker



Docker Image

Consider an example of Docker Image consisting of four layers

simplilearn

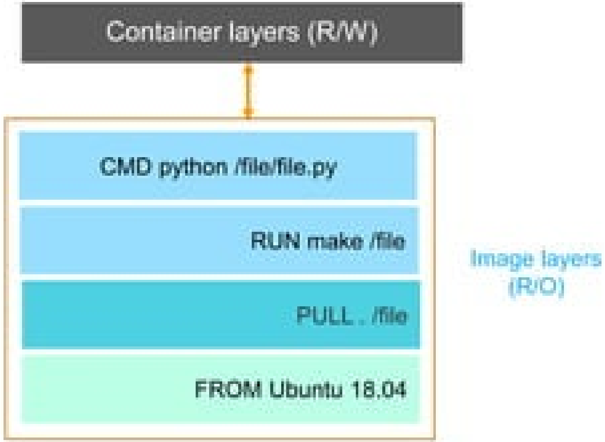
Docker Image - Components of Docker

Docker Image

Consider an example of Docker Image consisting of four layers

```
FROM ubuntu:18.04
PULL . /file
RUN make /file
CMD python /file/file.py
```

- FROM - Creates a layer from the ubuntu:18.04
- PULL - Adds files from your Docker repository
- RUN - Builds your container
- CMD - Specifies what command to run within the container

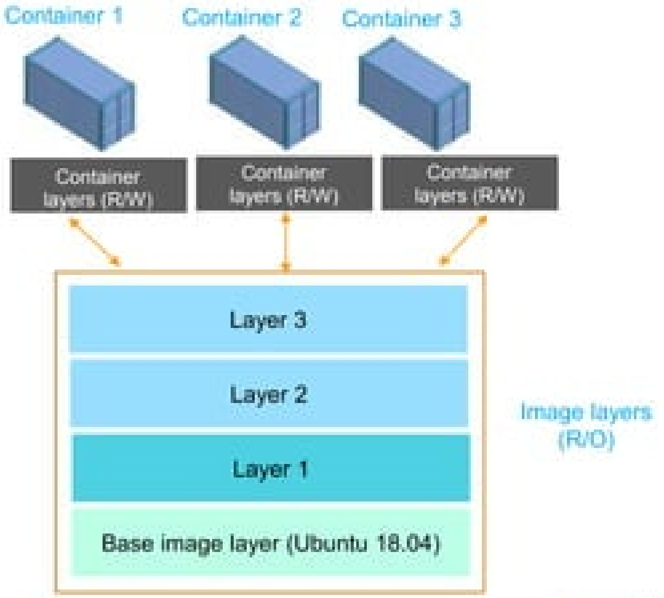


Docker Image - Components of Docker

Docker Image

- Whenever a user creates a container, a new layer is formed on top of the image layers called container layer

Container based on the Ubuntu 18.04 image:

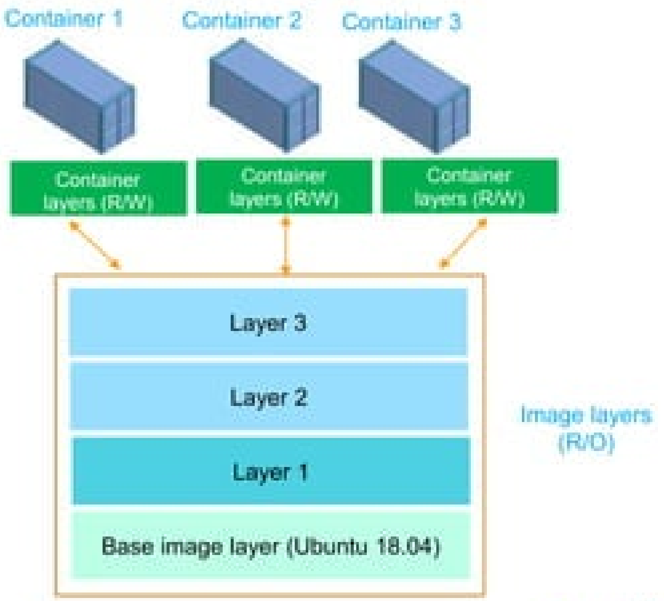


Docker Image - Components of Docker

Docker Image

- Whenever a user creates a container, a new layer is formed on top of the image layers called container layer
- Every container has a separate (R/W) container layer and any modification in a container is reflected upon the container layer alone

Container based on the Ubuntu 18.04 image:

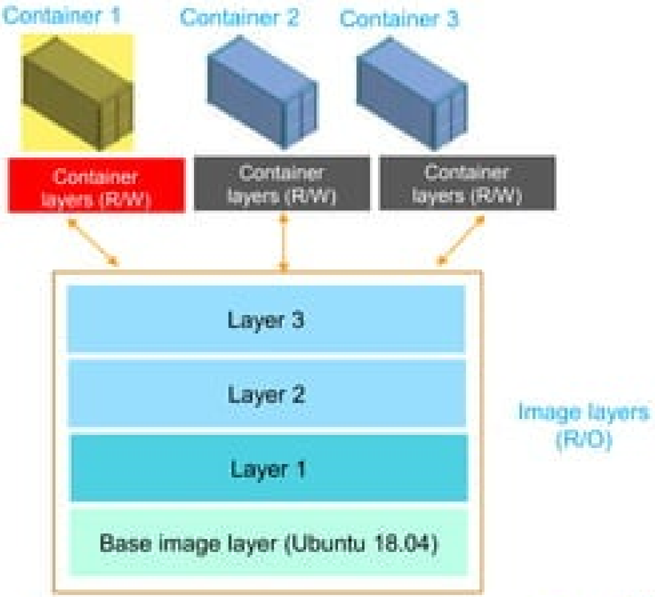


Docker Image - Components of Docker

Docker Image

- Whenever a user creates a container, a new layer is formed on top of the image layers called container layer
- Every container has a separate (R/W) container layer and any modification in a container is reflected upon the container layer alone
- When a container is deleted, the top layer also gets deleted

Container based on the Ubuntu 18.04 image:



Docker Image - Components of Docker

Docker Image

What should be done when there is a change in image layer?

Docker Image - Components of Docker

Docker Image

What should be done when there is a change in image layer?

- Users can add a new layer to the base image
- But, users cannot modify any of the existing image layers

Docker Image - Components of Docker

DID YOU KNOW?

- Base layers are in the read only format
- The layers can be combined in a union file system to create a single image
- Union file system saves memory space by avoiding duplicating of files
- This allows a file system to appear as writable (but without modifying the file) which is known as copy-on-write

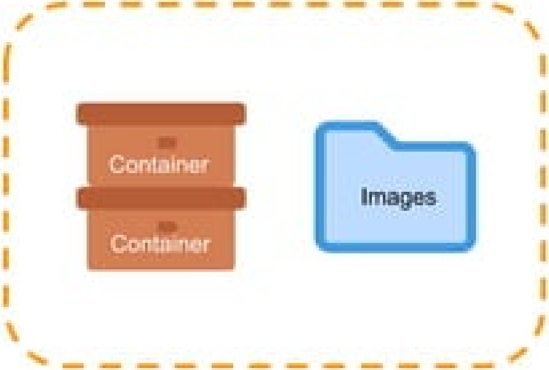
???

simplilearn

Docker Image - Components of Docker

Docker Image

- Docker uses a copy-on-write strategy with both Docker Images and Docker Containers

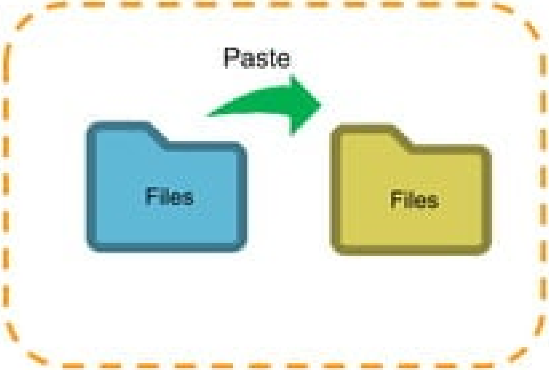


simplilearn

Docker Image - Components of Docker

Docker Image

- Docker uses a copy-on-write strategy with both Docker Images and Docker Containers
- CoW is a strategy to share and copy files for better efficiency



Docker Image - Components of Docker

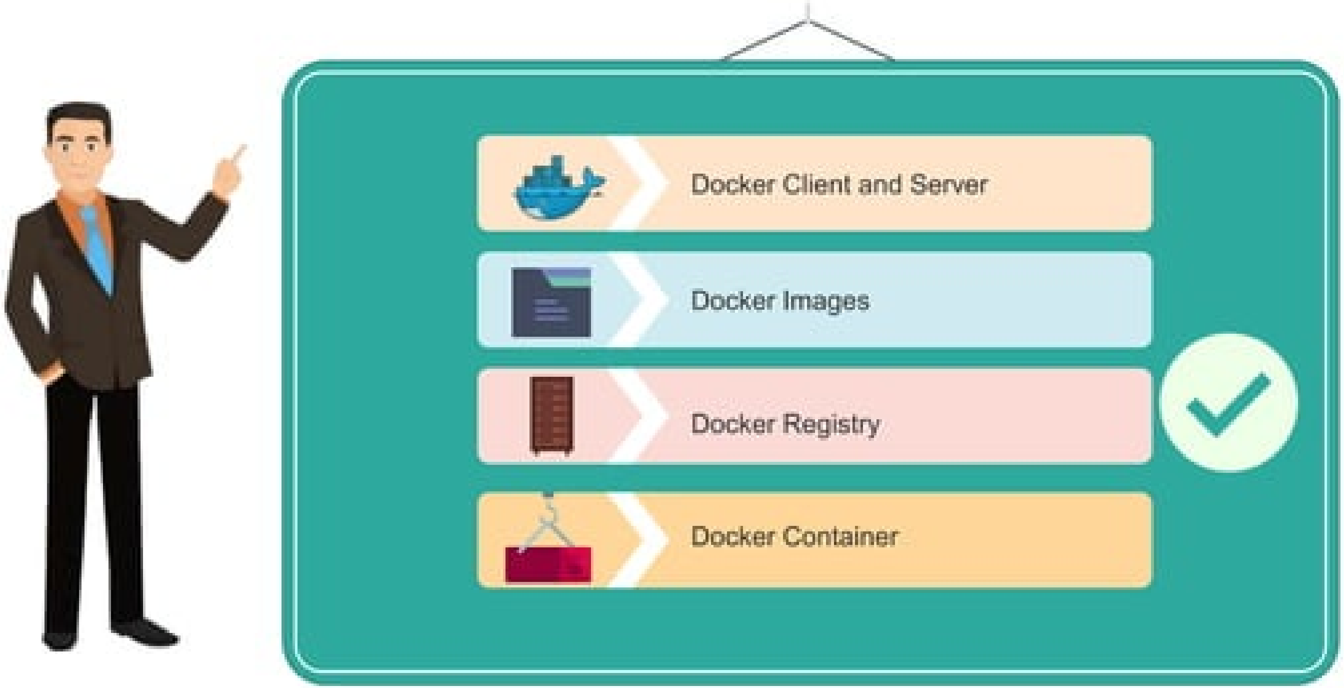
Docker Image

- Docker uses a copy-on-write strategy with both Docker Images and Docker Containers
- CoW is a strategy to share and copy files for better efficiency
- CoW strategy makes Docker efficient by reducing the usage of disk space and increasing the performance of container

✓

Better Efficiency

Docker Registry - Components of Docker



Docker Client and Server

Docker Images

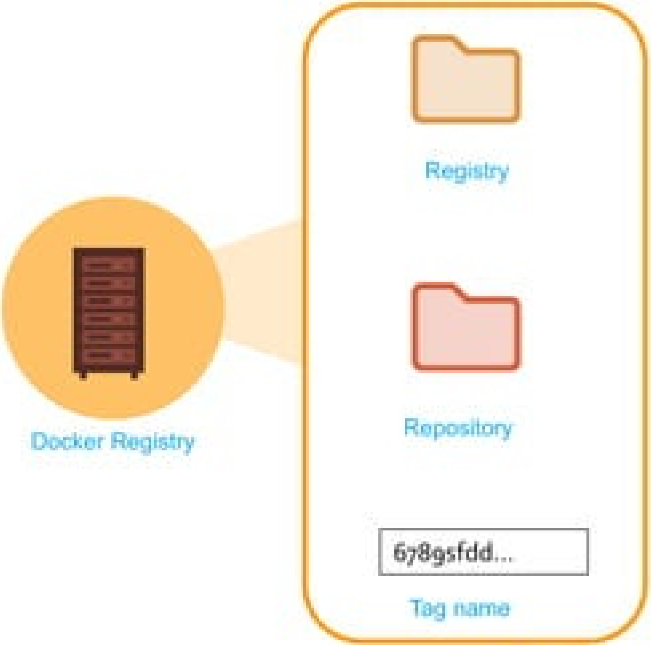
Docker Registry

Docker Container

Docker Registry - Components of Docker

Docker Registry

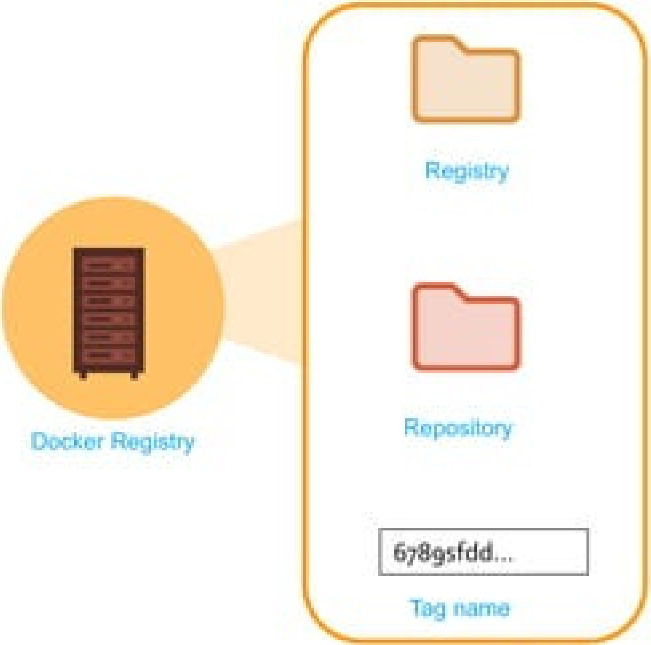
- Docker Registry is a service to host and distribute Docker Images among users



Docker Registry - Components of Docker

Docker Registry

- Docker Registry is a service to host and distribute Docker Images among users
- Repository is a collection of Docker Images

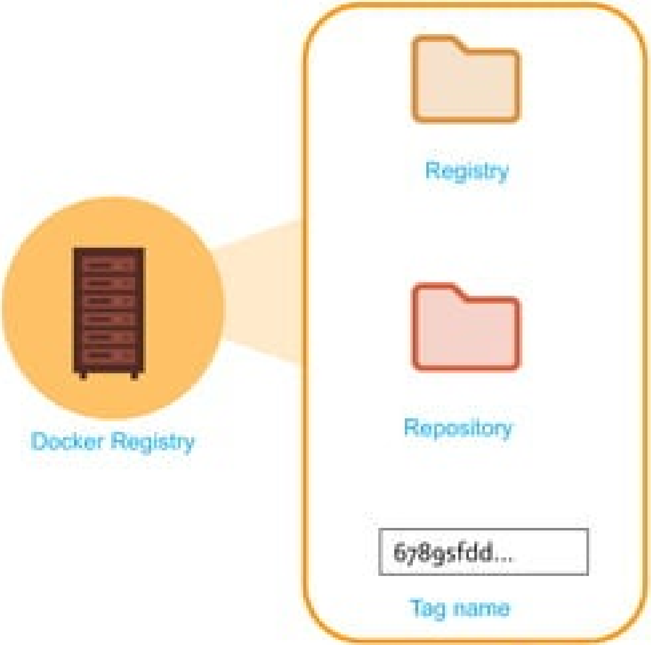


Docker Registry - Components of Docker

Docker Registry

- Docker Registry is a service to host and distribute Docker Images among users
- Repository is a collection of Docker Images
- In Registry, a user can distinguish between Docker Images with their tag names

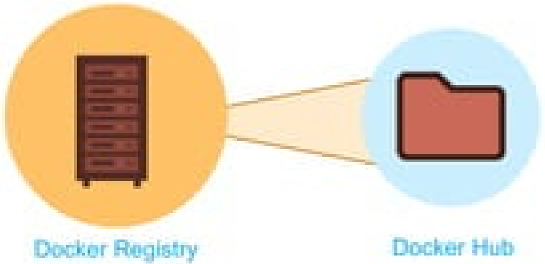
Note: A tag is a alphanumeric identifier attached to an image



Docker Registry - Components of Docker

Docker Registry

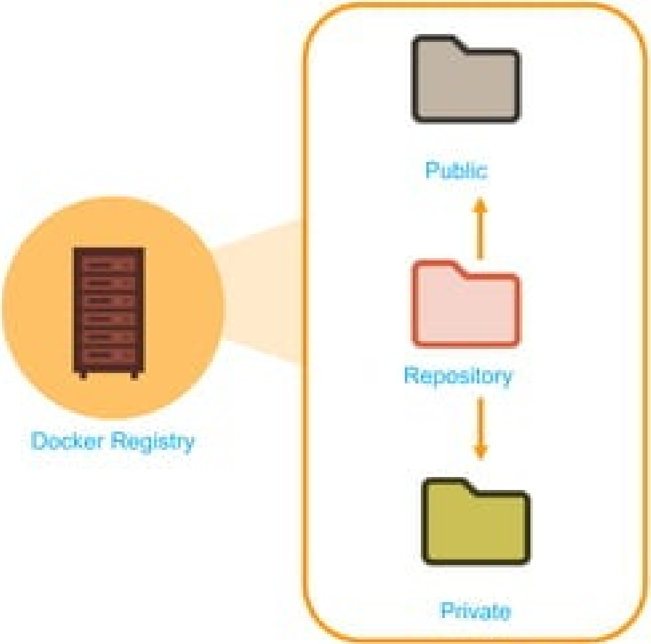
- Docker Registry is a service to host and distribute Docker Images among users
- Repository is a collection of Docker Images
- In Registry, a user can distinguish between Docker Images with their tag names
- Docker has its own cloud based Registry called Docker Hub where users store and distribute container images



Docker Registry - Components of Docker

Docker Registry

- Docker Registry has public and private repositories

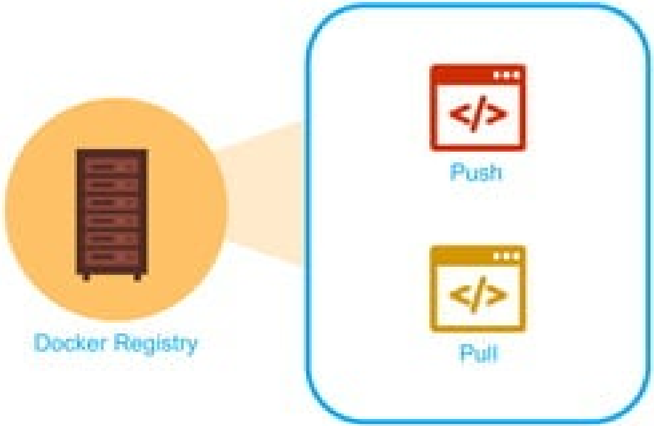


simplilearn

Docker Registry - Components of Docker

Docker Registry

- Docker Registry has public and private repositories
- In Registry, push and pull commands are used to interact with the Docker Images

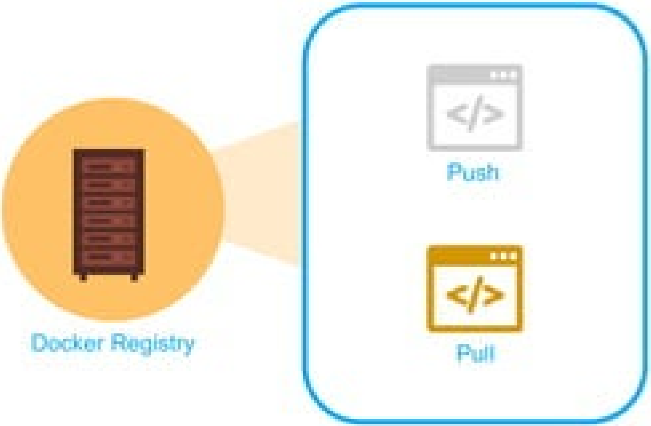


simplilearn

Docker Registry - Components of Docker

Docker Registry

- Docker Registry has public and private repositories
- In Registry, push and pull commands are used to interact with the Docker Images
- Pull command – It pulls (retrieves) a Docker Image from the Docker Registry

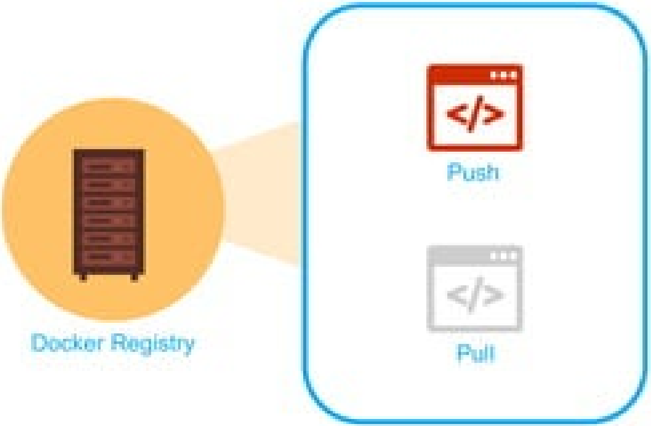


simplilearn

Docker Registry - Components of Docker

Docker Registry

- Docker Registry has public and private repositories
- In Registry, push and pull commands are used to interact with the Docker Images
- Pull command – It pulls (retrieves) a Docker Image from the Docker Registry
- Push command – It pushes (stores) a Docker Image in Docker Registry

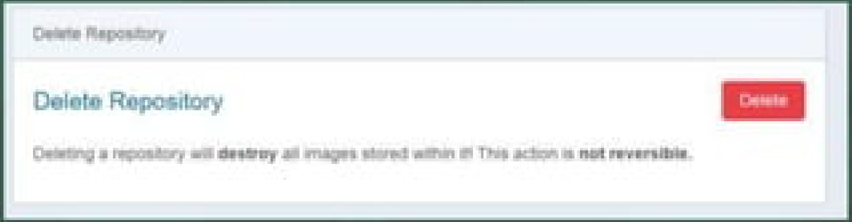


simplilearn

Docker Registry - Components of Docker

DID YOU KNOW?

In Docker Registry, deleting a repository is not a reversible action

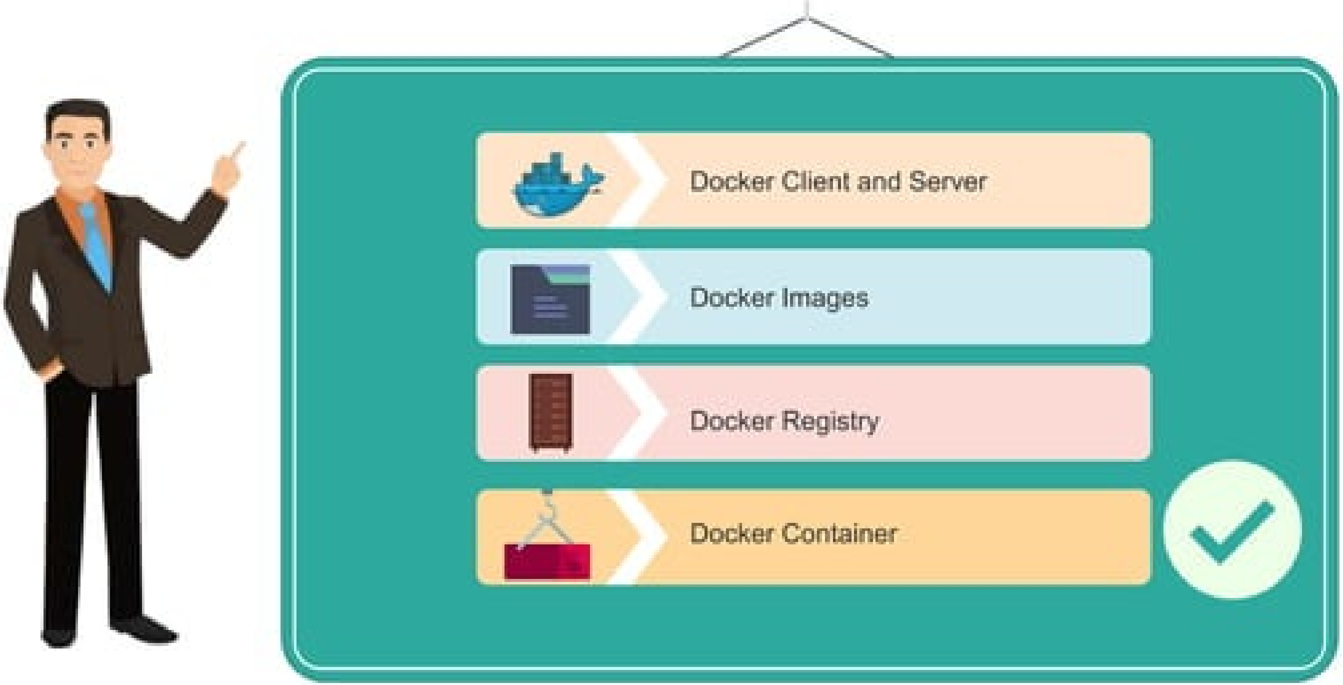


Delete Repository

Delete Repository

Deleting a repository will **destroy** all images stored within it! This action is **not reversible**.

Docker Container - Components of Docker



Docker Client and Server

Docker Images

Docker Registry

Docker Container ✓

Docker Container - Components of Docker

Docker Container

- Docker Container is an executable package of application and its dependencies together

Application
+
Dependencies
=
Docker Container

Docker Container - Components of Docker

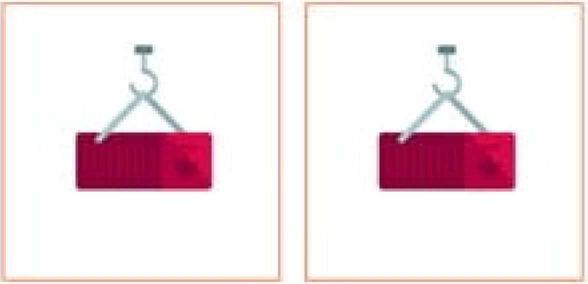
Docker Container

- Docker Container is an executable package of application and its dependencies together
- Since it's light-weight, it can be easily deployed and executed on other computer environments regardless of their host OS/ configurations

Docker Container - Components of Docker

Docker Container

- Docker Container is an executable package of application and its dependencies together
- Since it's light-weight, it can be easily deployed and executed on other computer environments regardless of their host OS/ configurations
- Docker Containers run applications in isolation and also share the OS kernel with other containers

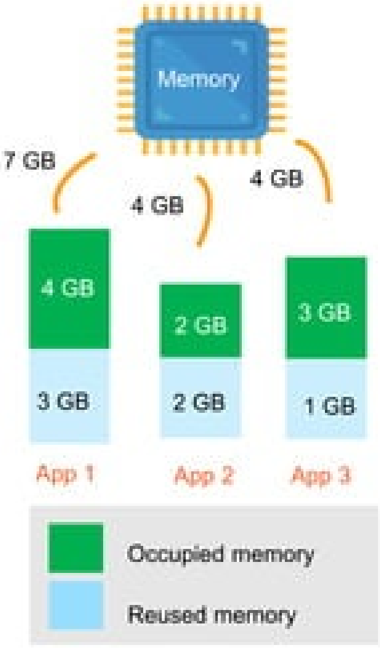


Application runs in isolation

Docker Container - Components of Docker

Docker Container

- Here, data volumes can be shared and reused among multiple containers



Docker Container - Components of Docker

Docker Container

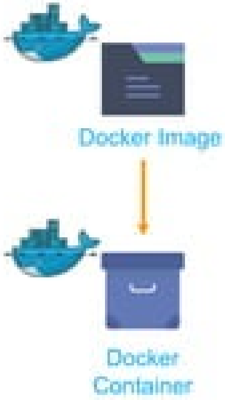
- Here, data volumes can be shared and reused among multiple containers
- It is built using Docker Images

Docker Images

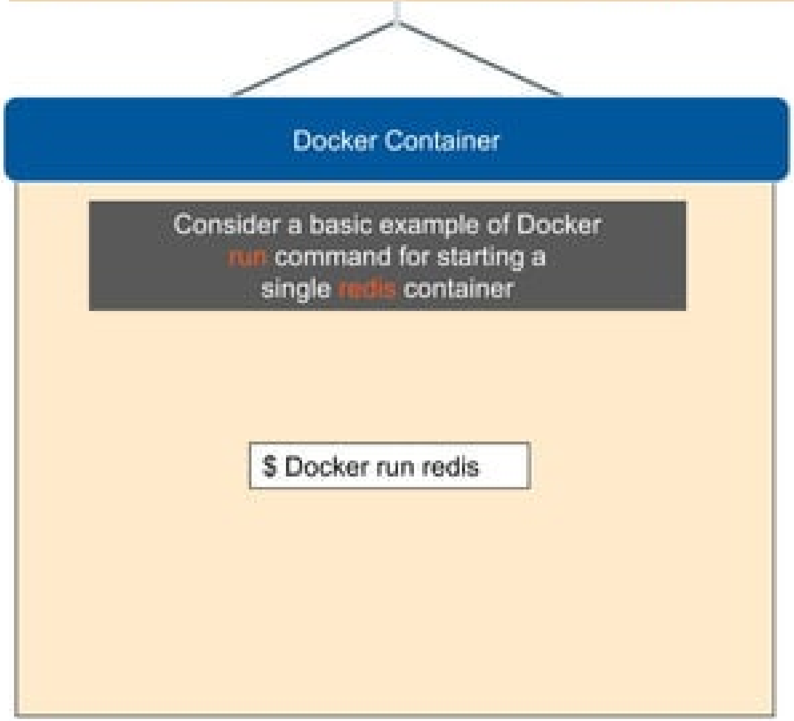
Docker Container - Components of Docker

Docker Container

- Here, data volumes can be shared and reused among multiple containers
- It is built using Docker Images
- Docker *run* command builds a container



Docker Container - Components of Docker



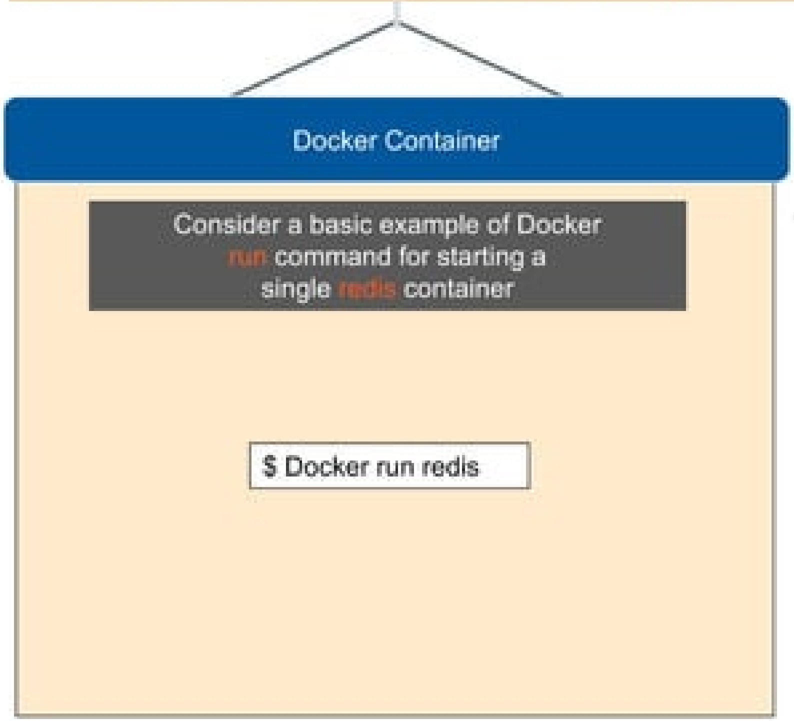
Docker Container

Consider a basic example of Docker run command for starting a single redis container

```
$ Docker run redis
```

simplilearn

Docker Container - Components of Docker



Docker Container

Consider a basic example of Docker run command for starting a single redis container

```
$ Docker run redis
```

Suppose a user runs *$ Docker run redis* command, the following happens:

Docker Container - Components of Docker

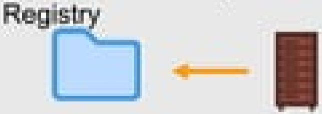
Docker Container

Consider a basic example of Docker run command for starting a single redis container

```
$ Docker run redis
```

Suppose a user runs *$ Docker run redis* command, the following happens:

- In case you don't have a Docker Image locally, the Docker pulls the image from your Registry



Docker Container - Components of Docker

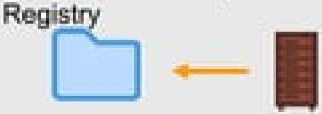
Docker Container

Consider a basic example of Docker run command for starting a single redis container

```
$ Docker run redis
```

Suppose a user runs *$ Docker run redis* command, the following happens:

- In case you don't have a Docker Image locally, the Docker pulls the image from your Registry



- Now, Docker creates a new container redis from the existing Docker Image

redis

Docker Container - Components of Docker

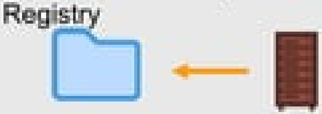
Docker Container

Consider a basic example of Docker run command for starting a single redis container

```
$ Docker run redis
```

Suppose a user runs *$ Docker run redis* command, the following happens:

- In case you don't have a Docker Image locally, the Docker pulls the image from your Registry



- Now, Docker creates a new container redis from the existing Docker Image

redis

- Docker creates a container layer of read-write filesystem

Docker Container - Components of Docker

Docker Container

How are containers lightweight?

Docker Container - Components of Docker

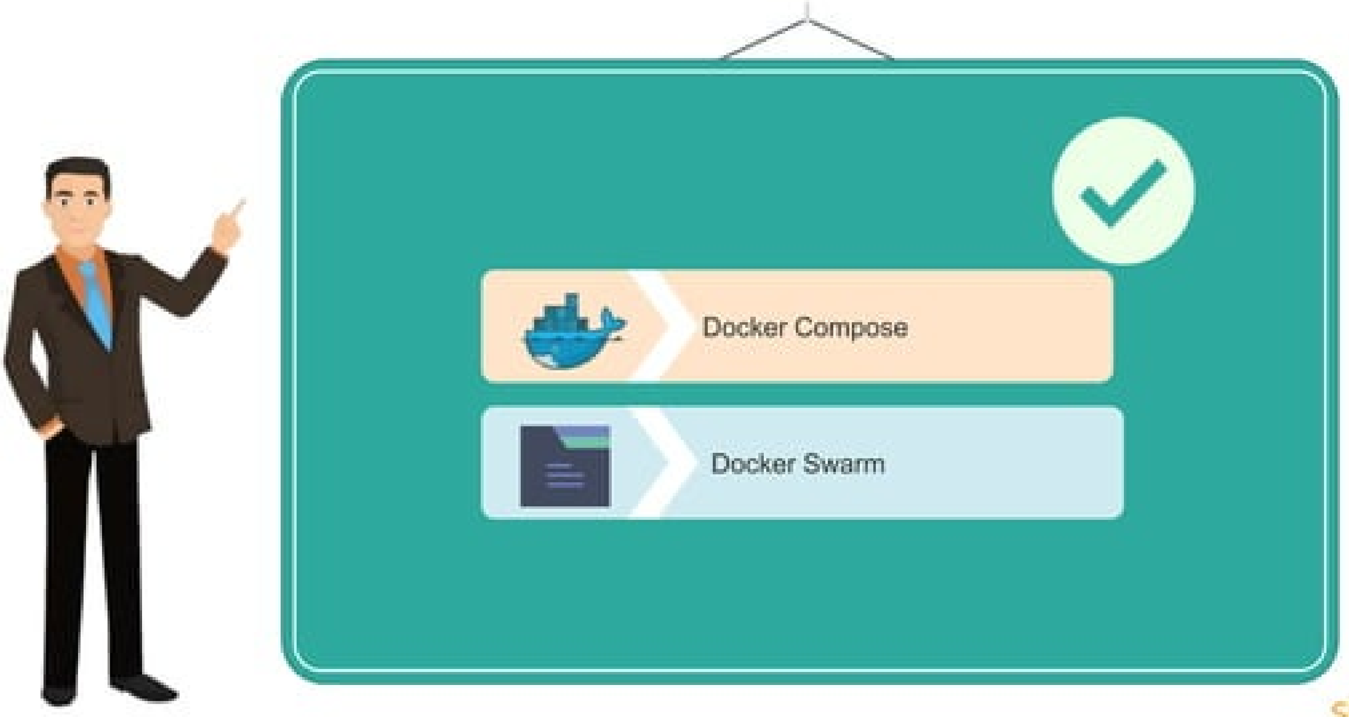
Docker Container

How are containers lightweight?

Docker Containers are lightweight because they do not require an extra layer of a hypervisor and run directly on the host operating system

Advanced concepts in Docker

Advanced concepts in Docker



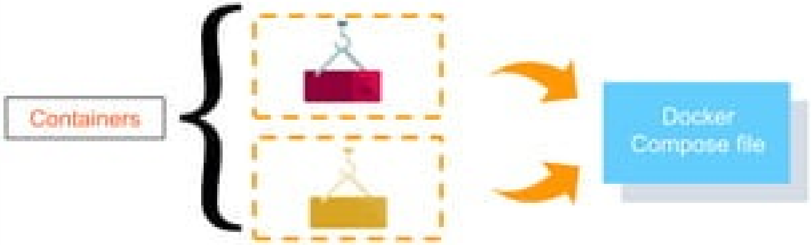
Docker Compose

Docker Swarm

Docker Compose

Docker Compose

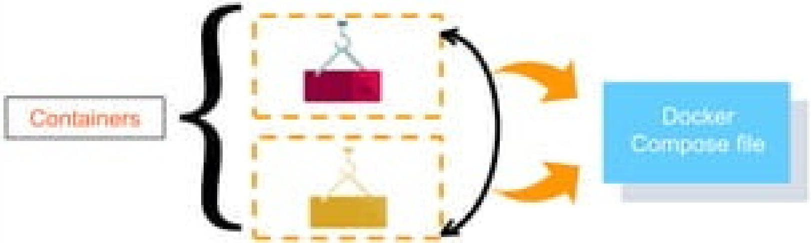
- Docker Compose is used for running multiple containers as a single service



Docker Compose

Docker Compose

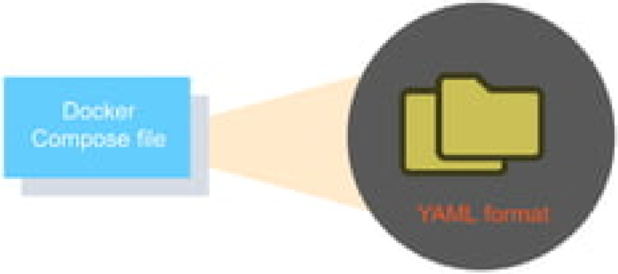
- Docker Compose is used for running multiple containers as a single service
- Here, each container runs in isolation but can interact with each other



Docker Compose

Docker Compose

- Docker Compose is used for running multiple containers as a single service
- Here, each container runs in isolation but can interact with each other
- All Docker Compose files are YAML files

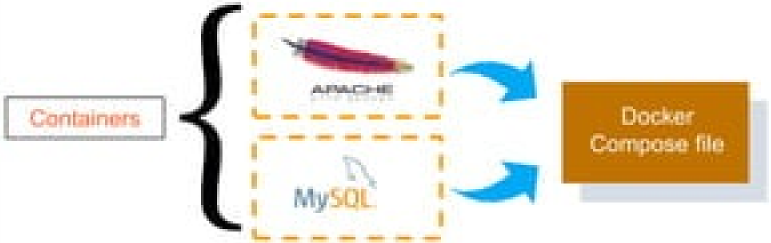


Docker Compose

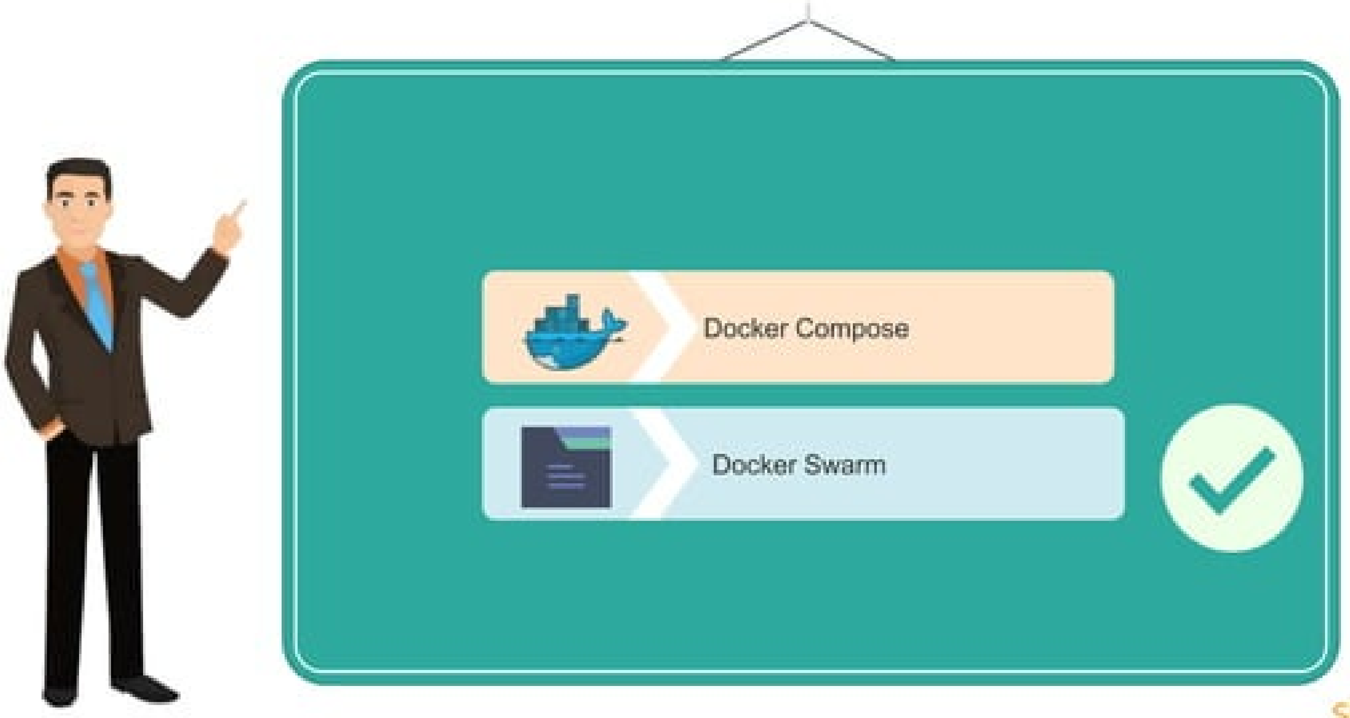
Docker Compose

For example:

If you have an application which requires Apache server and MySQL database, you could create one Docker Compose file which can run both containers as a service without the need to start each one separately



Docker Swarm



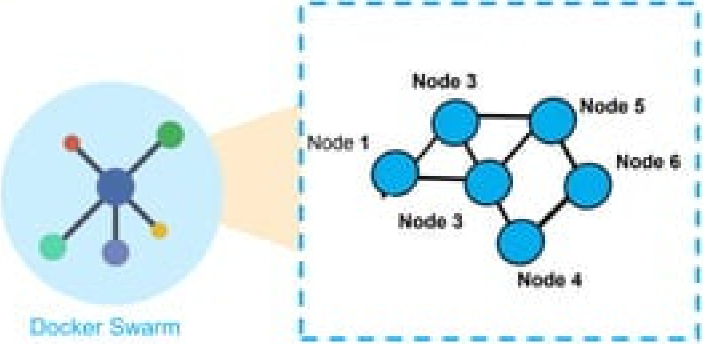
Docker Compose

Docker Swarm ✓

Docker Swarm

Docker Swarm

- Docker Swarm is a service for containers which allows IT administrators and developers to create and manage a cluster of swarm nodes within the Docker platform

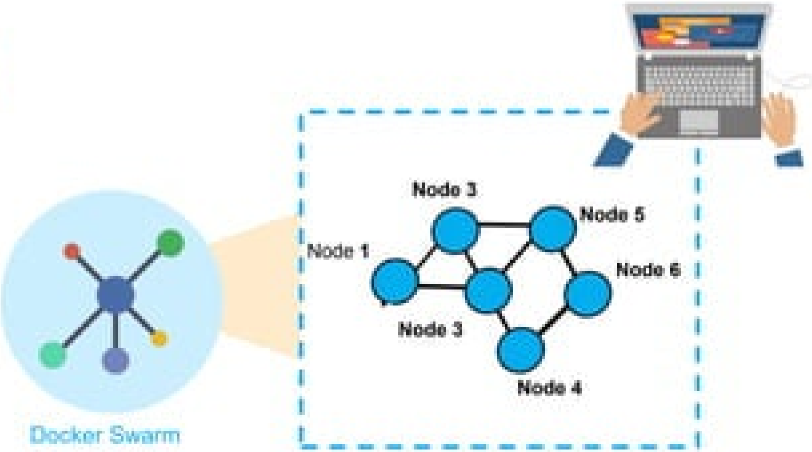


Note: A swarm node is an individual Docker Engine participating in the swarm

Docker Swarm

Docker Swarm

- Docker Swarm is a service for containers which allows IT administrators and developers to create and manage a cluster of swarm nodes within the Docker platform
- Each node of Docker Swarm is a Docker Daemon and all Docker Daemons interact using the Docker API

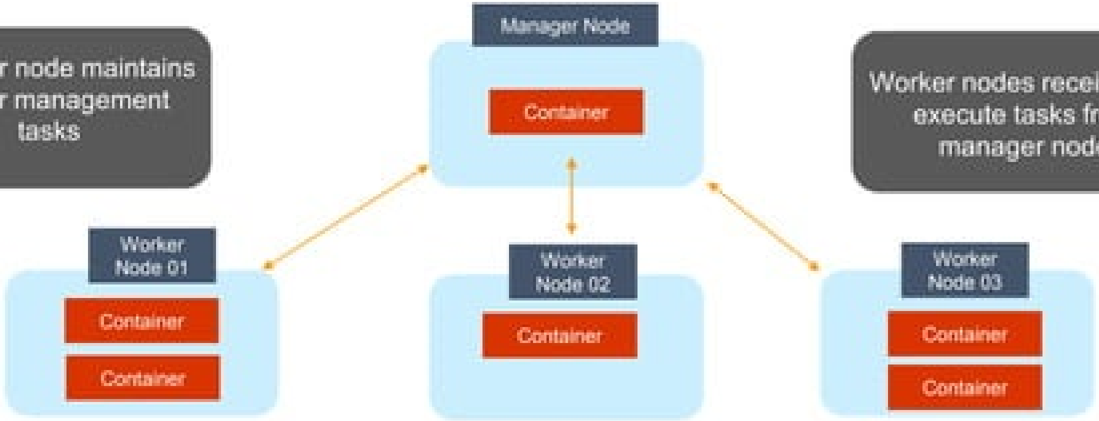


Docker Swarm

Docker Swarm

A swarm consists of two types of nodes:
Manager node and worker node

Manager node maintains cluster management tasks



Worker nodes receive and execute tasks from manager node

Basic Docker commands

Basic Docker commands

Install Docker on your system

```
yum install Docker
```

Basic Docker commands

Install Docker on your system

```
yum install Docker
```

Start the Docker Daemon

```
systemctl start Docker
```

Basic Docker commands

Install Docker on your system

```
yum install Docker
```

Start the Docker Daemon

```
systemctl start Docker
```

Command to remove Docker Image

```
Docker rmi ImageID
```

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Command to run an image
Docker run <image-id>

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Command to run an image
Docker run <image-id>

Command to pull a Docker Image from a Docker hub
Docker pull <image-name:tag>

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Command to run an image
Docker run <image-id>

Command to pull a Docker Image from a Docker hub
Docker pull <image-name:tag>

Command to build an image from a Dockerfile
Docker build –t[image name]:tag

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Command to run an image
Docker run <image-id>

Command to pull a Docker Image from a Docker hub
Docker pull <image-name:tag>

Command to build an image from a Dockerfile
Docker build –t[image name]:tag

Command to shut down the container Docker stop container_ID

Basic Docker commands

Install Docker on your system
yum install Docker

Start the Docker Daemon
systemctl start Docker

Command to remove Docker Image
Docker rmi ImageID

Command to download an image Docker pull image_name

Command to run an image
Docker run <image-id>

Command to pull a Docker Image from a Docker hub
Docker pull <image-name:tag>

Command to build an image from a Dockerfile
Docker build –t[image name]:tag

Command to shut down the container Docker stop container_ID

Command to access a running container
Docker exec it container_ID bash

Demo

THANK YOU

For more information, visit

www.simplilearn.com

simplilearn