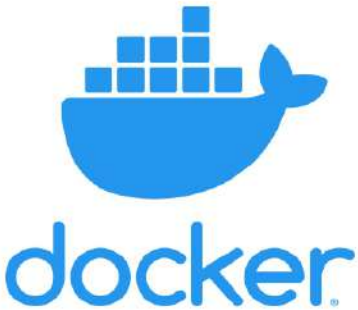


MAIL: Rnraju4u@gmail.com **NUMBER:** +91 9848363431 **YOUTUBE:** SYSGEEKS

DOCKER

- Docker is an open platform tool for developing, shipping, & running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system.
- Docker provides a way to run applications securely isolated in a container, packaged with all its dependencies and libraries.
- It is designed to benefit both developers and system administrators, making it a part of many DevOps tool chains.

VIRTUALIZATION (VT)

- **VT** is a software technology that makes computing environments independent of physical infrastructure.
- It is a process of creating virtual applications, virtual servers, storage & n/w.
- It is the single most effective way to reduce IT expenses while boosting efficiency & agility for all size businesses.

VIRTUALIZATION BENEFITS:

- Reduced capital and operating costs.
- Minimized or eliminated downtime.
- Increased IT productivity, efficiency, agility and responsiveness.
- Faster provisioning of applications and resources.
- Greater business continuity and disaster recovery.
- Simplified data center management.
- Availability of a true Software-Defined Data Center.

VIRTUALIZATION TYPES:

- Server Virtualization
- Network Virtualization
- Desktop Virtualization
- Para-virtualization
- Hardware-level virtualization

HYPERVISORS

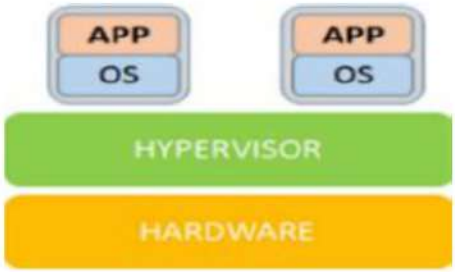
- A hypervisor is a hardware virtualization technique that **allows multiple guest operating systems** to run on a single host system at the same time.
- Guest OS shares hardware of the host computer, have its own processor, memory and other h/w resources.
- A hypervisor is also known as a **Virtual Machine Manager (VMM).**

HYPERVISIOR TYPES:

TYPE1:

- **Type1** is on bare metal. VM resources are scheduled directly to the hardware by the hypervisor.

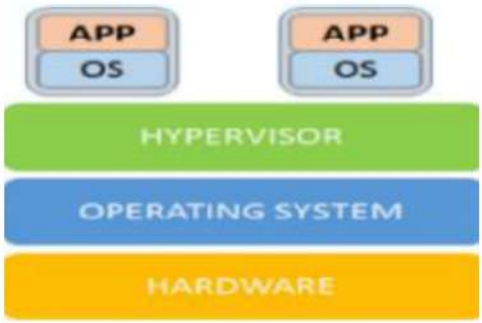
  Eg: VMware ESXI, Citrix XenServer, Microsoft Hyper-V, Linux KVM.



TYPE2:

- **Type2** is hosted. VM resources are scheduled against a host operating system, which is then executed against the hardware.

  Eg: VMware workstation and Oracle virtual box.



VIRTUAL MACHINE (VM)

- **A VM** is a virtual environment that functions as a virtual computer system with its own CPU, memory, network, and storage, created on a physical.
- Most enterprises use a combination of physical and virtual infrastructure to balance the corresponding advantages and disadvantages.

KEY PROPERTIES OF VIRTUAL MACHINE:

PARTITIONING:

- Run multiple operating systems on one physical machine.
- Divide system resources between virtual machines.

ISOLATION:

- Provide fault and security isolation at the hardware level.
- Preserve performance with advanced resource controls.

ENCAPSULATION:

- Save the entire state of a virtual machine to files.
- Move and copy virtual machines as easily as moving and copying files.

HARDWARE INDEPENDENCE:

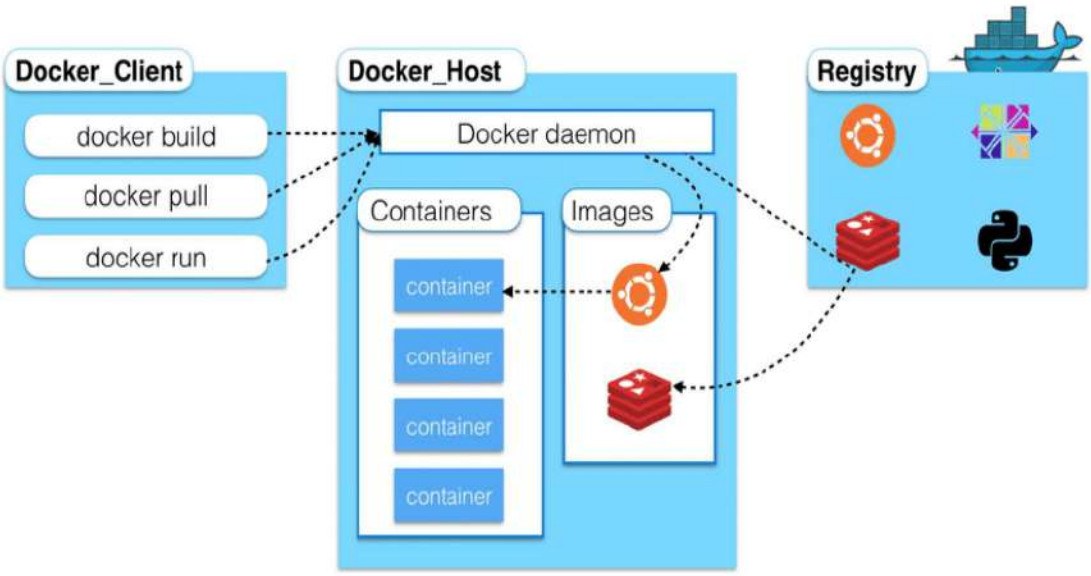
- Provision or migrate any virtual machine to any physical server.

VIRTUALIZATION vs. CLOUD COMPUTING

- Virtualization is software that makes computing environments independent of physical infrastructure.
- Cloud computing is a service that delivers shared computing resources (software and/or data) on demand via the Internet.
- As complementary solutions, organizations can begin by virtualizing their servers and then moving to cloud computing for even greater agility and self-service.

DOCKER ARCHITECTURE

- 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**.



DOCKER DAEMON:

- The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.

DOCKER CLIENT:

- The Docker client (docker) is the primary way that many Docker users interact with Docker.

DOCKER REGISTRIES:

- 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.

DOCKER OBJECTS

IMAGES:

- An *image* is a read-only template with instructions for creating a Docker container.
- A container is launched by running an image. An **image** is an executable package that includes everything needed to run an application–the code, a runtime, libraries, environment variables, and configuration files.

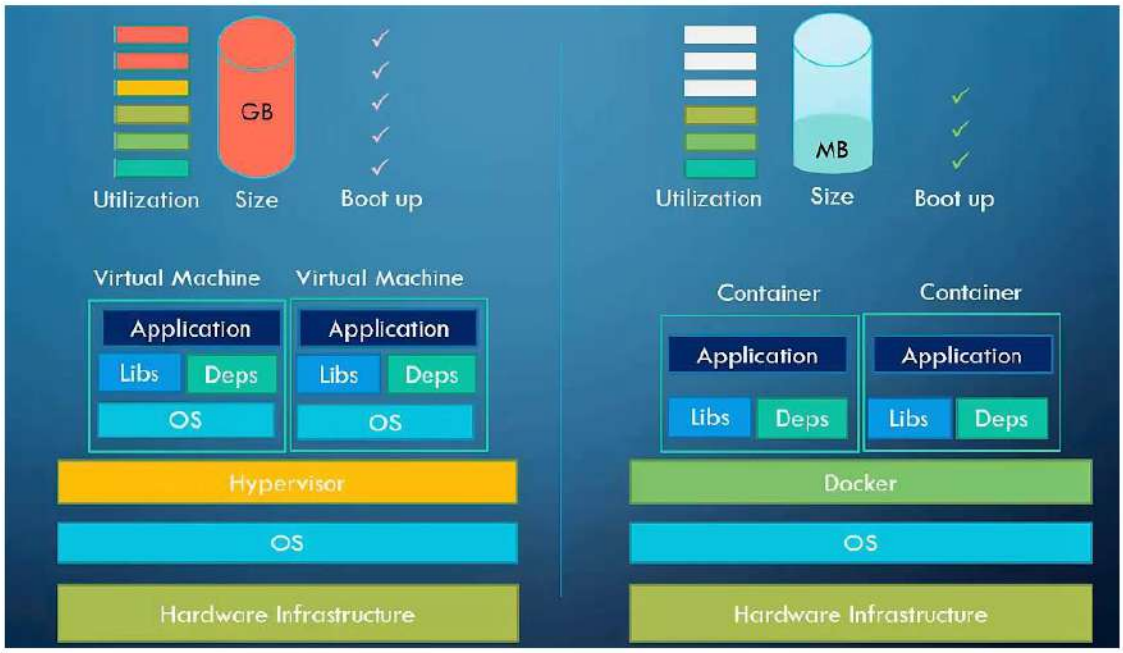
CONTAINERS:

- 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.
- Docker Containers are:
  - **Flexible** : The most complex applications can be containerized.
  - **Lightweight** : Containers leverage and share the host kernel.
  - **Interchangeable:** You can deploy updates and upgrades on-the-fly.
  - **Portable** : Build locally, deploy to the cloud, and run anywhere.
  - **Scalable** : Increase and automatically distribute container replicas.
  - **Stackable** : You can stack services vertically and on-the-fly.

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.
- 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 MACHINES VS CONTAINERS



VIRTUAL MACHINES:

- A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical. In other words, creating a computer within a computer.
- Multiple virtual machines can run simultaneously on the same physical computer.

CONTAINERS:

- A container is a running instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI.

DOCKER INSTALLATION

DOCKER ENGINE OVERVIEW:

- Docker Engine is an open-source containerization technology for building and containerizing your applications.
- Docker Engine acts as a client-server application with:
  - A server with a long-running daemon process dockerd.
  - APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
  - A command line interface (CLI) client docker.

SUPPORTED PLATFORMS:

- Docker Desktop for Mac (macOS)
- Docker Desktop for Windows
- Linux distributions:
  Red Hat, Centos, Fedora, Debian, Ubuntu…etc.
- Cloud Platforms:
  AWS, AZURE, GCP, Digital Ocean…. etc.

INSTALL DOCKER ENGINE ON LINUX:

INSTALL ON CENTOS / RHEL 9:

- To install Docker Engine, you need a maintained version:
  - **CentOS 7 or 8**
  - **RHEL 7 or 8**
  - **Fedora 33 or 34**

INSTALL ON UBUNTU:

- To install Docker Engine, you need the **64-bit** version of one of these Ubuntu versions:
  - Ubuntu Hirsute 21.04
  - Ubuntu Groovy 20.10
  - Ubuntu Focal 20.04 (LTS)
  - Ubuntu Bionic 18.04 (LTS)

STEP 1: Setting Up Hostname

#hostname Docker

#vim /etc/hostname

Docker

#bash

STEP2: Security-Enhanced Linux is being disabled or in permissive mode.

#sed -i 's/SELINUX=.*/SELINUX=disabled/g' /etc/selinux/config

#setenforce 0

STEP 3: Update current OS

#yum update -y

STEP 4: Unistall Old Versions

```
yum remove docker \
        docker-client \
        docker-client-latest \
        docker-common \
        docker-latest \
        docker-latest-logrotate \
        docker-logrotate \
        docker-engine \
        podman \
        runc
```

STEP 5: Set up the repository

#yum install -y yum-utils

#yum-config-manager --add-repo
https://download.docker.com/linux/rhel/docker-ce.repo

STEP 6: Install Docker Engine

#yum install docker-ce -y

NOTE: Getting any error, plese change repo lines

#vim /etc/yum.repos.d/docker-ce.repo

```
[docker-ce-stable]
name=Docker CE Stable - $basearch
baseurl=https://download.docker.com/linux/centos/$releasever/$basearch/stable
enabled=1
gpgcheck=1
gpgkey=https://download.docker.com/linux/centos/gpg
```

#yum install docker-ce -y

#docker --version

STEP 7: Start and Enable docker service

#systemctl start docker

#systemctl enable docker

#systemctl status docker

INSTALL DOCKER DESKTOP ON WINDOWS

- Windows 10 64-bit: Home or Pro 2004 (build 19041) or higher, or Enterprise or Education 1909 or higher.
- Enable the WSL 2 feature on Windows. (Windows Subsystem for Linux, version 2)
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10:
  - 64-bit processor with Second Level Address Translation (SLAT)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings.
  - Download and install the Linux kernel update package: https://docs.microsoft.com/en-us/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package

NOTE: Please follow the bellow link for docker installation. https://docs.docker.com/engine/install/

DOCKER IMAGES & CONTAINERS

DOCKER IMAGES:

#docker search centos

#docker pull centos

#docker images (or) #docker image ls

Image History:

#docker image history a9d583973f65

Image Details:

#docker image inspect ubuntu

Removing dangling images:

A dangling image is an image that is not tagged and is not used by any container.

To remove dangling images:

#docker image prune

Remove Image:

#docker rmi imageid or docker images rm imageid

Removing all unused images

#docker image prune -a

MANIPULATING DOCKER IMAGES:

#docker run -i -t <imagename>:<tag> /bin/bash

Options: -i : Gives us an interactive shell into the running container

-t : Will allocate a pseudo-tty

-d : The daemon mode

Create a first Container with name Test1

#docker run -d -it --name Test1 centos

To List Running Containers

#docker ps (or) #docker container ls

To list all containers

#docker ps -a

Docker Exec: Execute a command in a running container.

Execute a command on a container:

#docker exec -d Test1 touch /opt/aws

#docker exec -d Test1 ls /opt/

Execute an interactive bash shell on the container:

#docker exec -it Test1 bash

#exit

To Stop Container (Application shutdown gracefully)

#docker stop cid

#docker start -a cid : -a attach mode

Rename Container:

#docker rename <current_container_name> <new_container_name>

Container stats:

#docker stats <container_name>

#docker stats cid

Monitor Container:

#docker top cid/name

Container Pause:

#docker container pause containerID

Container Unpause:

#docker container unpause cid

Kill one or more Containers:

#docker kill cid [no time for proper shutdown of Application]

Removing Containers:

#docker container rm cid (or) #docker rm cid

Removing all stopped containers:

#docker container prune

Docker Logs: Fetch the logs of a container

Check Docker Logs:

#docker logs cid

Logs to verify Consensually:

#docker logs -f cid

Docker System and stats:

Docker System will manage docker (Host system)

Show docker disk usage:

#docker system df

Display system-wide information

#docker system info :

Get real time events from the server:

#docker system events

Use docker events to get real-time events from the server:

Open two terminals, on first one run:

#docker system events

On second terminal launch or stop any container

#docker stop CID / #docker run -d -it --name Test2 centos

Now to check the events on first terminal

Delete all stopped and unused containers:

#docker system prune

Delete all images and stopped containers:

#docker system prune -a

Docker stats: The docker stats command returns a live data stream for running containers.

#docker stats CID

PORT FORWARDING:

#docker run -d -it -p <host_port>:<container_port> <image>:<tag>

#docker pull nginx

#docker run -d -it --name mynginx -p 8090:80 nginx

#docker ps

To check the running port of the container

Go To Web Browser

http://IP-Address:8090/

To check Container Logs

#docker logs cid

Runtime options with Memory & CPUs:

By default, a container has no resource constraints and can use as much of a given resource as the host's kernel scheduler allows.

Docker provides ways to control how much memory, or CPU a container can use, setting runtime configuration flags of the docker run command.

Container memory limit 512m max

#docker run -d -it –name sample1 -p 800:80 -m 512m nginx

#docker status sample1

cpulimit 50000(50%) total cpu size 100thousend

#docker run -d -it --name sample2 -p 900:80 --cpu-quota=50000 nginx

#docker status sample2

To run a container from the centos image, assigning 1 GB of RAM for the container to use and reserving 1 GB of RAM for swap memory, type:

#docker run -d -it --name sample3 --memory="1g" --memory-swap="2g" centos

DYNAMICALLY UPDATES CONTAINER CONFIGURATION:

Docker Update: Update configuration of one or more containers

Update a container's cpu-shares

#docker update --cpu-shares 512 cid

Update a container with cpu-shares and memory

#docker update --cpu-shares 512 -m 300M cid

If you started a container with this command:

#docker run -dit --name test4 --kernel-memory 50M centos bash

You can update kernel memory while the container is running:

#docker update --kernel-memory 80M test4