**. What is a Container?**

* **Theory:**  
  A container is a lightweight, standalone, executable software package that includes everything needed to run a piece of software, including code, runtime, system tools, libraries, and settings. Containers are isolated from one another but share the same OS kernel, making them efficient.
* A container is like a "box" for your app. It holds everything your app needs to run (like code, libraries, settings), but it's smaller and lighter than a virtual machine (VM). Think of a container as a way to carry your app from one computer to another, and it works the same everywhere.
* Example : Imagine you have a jar of jam. No matter where you take that jar (your kitchen, someone else's house), the jam inside tastes the same. Similarly, a container ensures your app runs the same no matter where it's placed.

**2. What is Docker?**

* **Theory:**  
  Docker is an open-source platform for automating the deployment, scaling, and management of applications using containerization. Docker allows developers to package applications into containers and simplifies development and operations (DevOps).

Docker is like a tool that helps you create and manage containers. It makes it easy to pack your app and move it around without worrying if it will work on different computers.

**Example:**  
If you’ve ever baked cookies and packed them in boxes to give to friends, Docker is the tool that helps you pack the cookies into these boxes (containers) quickly and reliably

**3. Why Do We Need Docker?**

* **Theory:**  
  Docker ensures that your applications behave the same regardless of where they are deployed. It enables microservices architecture, simplifies CI/CD, and ensures rapid deployment and scalability of applications.
* **Theory:**  
  Docker makes sure your app behaves the same everywhere. You don’t have to worry about "it works on my machine" but breaks on another. It’s especially useful when working in teams, deploying apps, or using cloud services.
* **Example:**  
  If you travel to different places and want the same food everywhere, you pack your own food. Docker helps "pack" your app, so it works the same wherever it goes.

**4. Difference Between Docker and VM**

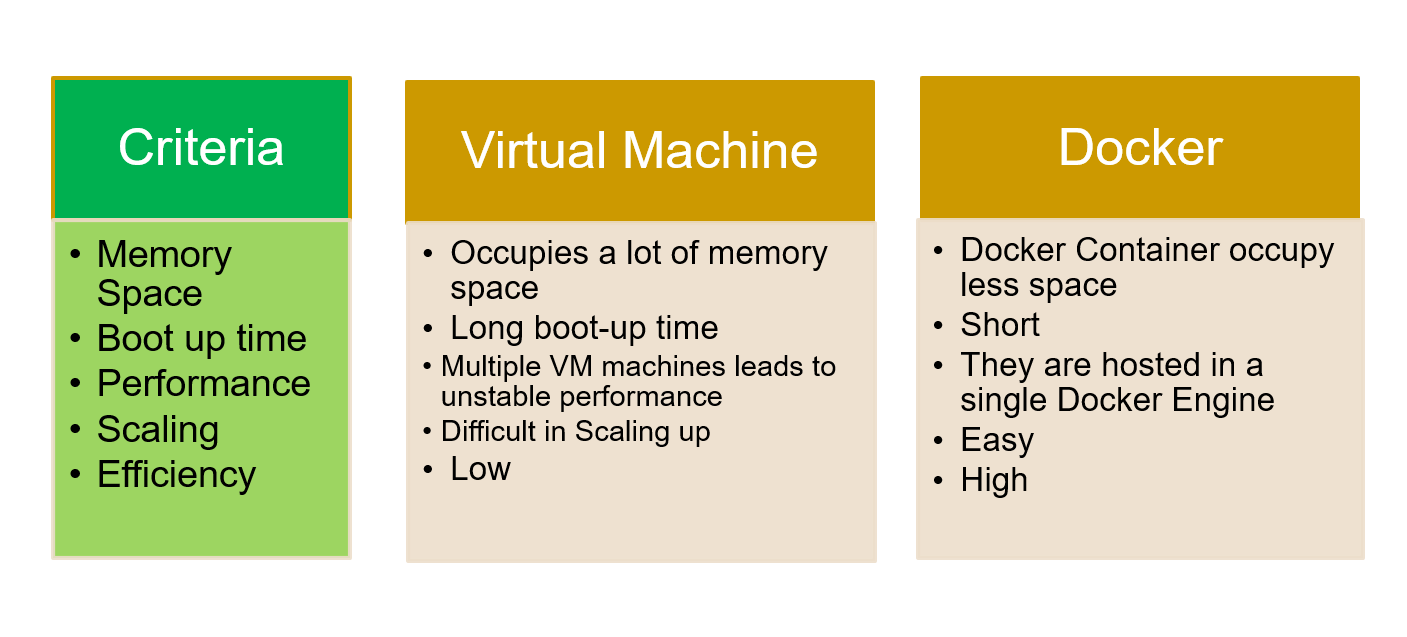
**VM:** Requires a full OS, including its own kernel. Heavy on resources.

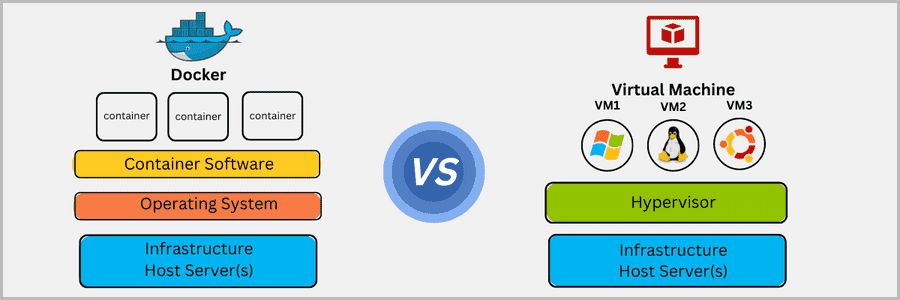
**Docker:** Uses OS-level virtualization, sharing the host OS kernel, making it lightweight and faster.

VM (Virtual Machine): A VM is like a full computer inside your computer. It runs its own operating system (OS), which takes a lot of space and memory.

**Docker:** Uses OS-level virtualization, sharing the host OS kernel, making it lightweight and faster.

Docker: Docker containers are lighter. They share the main computer's OS, making them faster and using fewer resources.





Example:

Running a VM is like having multiple TV sets in your house—each with its own power supply. Running Docker is like having different shows on the same TV.

**5. What is OS-Level Virtualization?**

* **Theory:**  
  OS-level virtualization allows multiple isolated user-space instances (containers) to run on a single host OS kernel. Containers are more lightweight than VMs, which require a full OS.
* This means running multiple containers on the same operating system (OS). Each container thinks it has its own OS, but really they all share the same one. It’s much faster and lighter than giving each container its own OS.

**6. What is a Hypervisor?**

* **Theory:**  
  A hypervisor is software that allows multiple VMs to run on a single hardware host by managing and allocating resources to each VM.
* A hypervisor is software that helps you create and run virtual machines. It splits up the computer’s resources (memory, CPU) so that each VM feels like it’s running on its own hardware.
* A hypervisor is like the landlord of an apartment building who makes sure each apartment (VM) has enough resources like water, electricity, and gas.

**7. Difference Between Deploying an Application on VM and Docker**

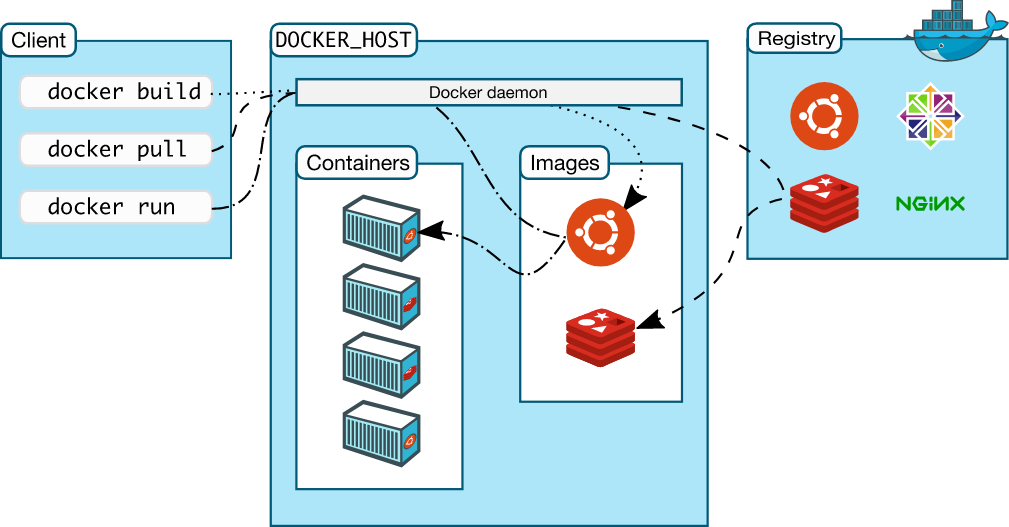
* **Theory:**  
  Deploying on a VM involves installing an OS and setting up the environment for the app. In Docker, containers are pre-configured environments with .

**Lab:**

* + Deploy a simple app on a VM and Docker, compare resource usage.

**8. Docker Architecture**

* **Theory:**  
  Docker architecture includes three main components: Docker Client, Docker Daemon, and Docker Registry. The client interacts with the daemon to build and run containers, and the registry stores container images.



**9. What is Dockerfile, Docker Image, Docker Container?**

* **Theory:**
  + **Dockerfile:** A script of instructions to build a Docker image.
  + **Docker Image:** A read-only template containing the application and its dependencies.
  + **Docker Container:** A runtime instance of a Docker image.

**10. Docker Installation Using yum/apt and RPM**

* **Theory:**  
  Docker can be installed on various Linux distributions via yum, apt, or by manually downloading the .rpm package.

**Lab:**

* + Install Docker on Ubuntu:

**sudo apt update**

**sudo apt install docker.io**

* + Install Docker on CentOS using yum:

**sudo yum install docker**

**11. Docker Commands**

* **Theory:**  
  Common Docker commands include docker run, docker ps, docker stop, docker build, docker exec, and docker logs.

**Lab:**

* + Run a simple container and interact with it:

bash

Copy code

docker run -it ubuntu bash

**12. Docker Compose**

* **Theory:**  
  Docker Compose allows you to define and run multi-container Docker applications. You can define services, networks, and volumes in a YAML file (docker-compose.yml).

**Lab:**

* + Create a docker-compose.yml file and start services:

yaml

Copy code

version: '3'

services:

web:

image: nginx

ports:

- "80:80"

bash

Copy code

docker-compose up

**13. Docker Volumes**

* **Theory:**  
  Docker volumes are used to persist data generated by containers. They allow sharing data between containers and the host machine.

**Lab:**

* + Create a volume and attach it to a container:

bash

Copy code

docker volume create mydata

docker run -v mydata:/data -it ubuntu bash

**14. Docker Networks**

* **Theory:**  
  Docker networking allows containers to communicate with each other, either within the same host or across different hosts.

**Lab:**

* + List Docker networks and create a custom network:

bash

Copy code

docker network ls

docker network create mynetwork

This structure covers both the theoretical concepts and practical labs. Let me know if you need any additional resources or exercises!