Mayank Kamboj Assignment 1

What Is Docker?

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. 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.

Docker Container

Docker containers are the live, running instances of Docker images. While Docker images are read-only files, containers are live, ephemeral, executable content. Users can interact with them, and administrators can adjust their settings and conditions using docker commands.

What can I use Docker for?

Fast, consistent delivery of your applications

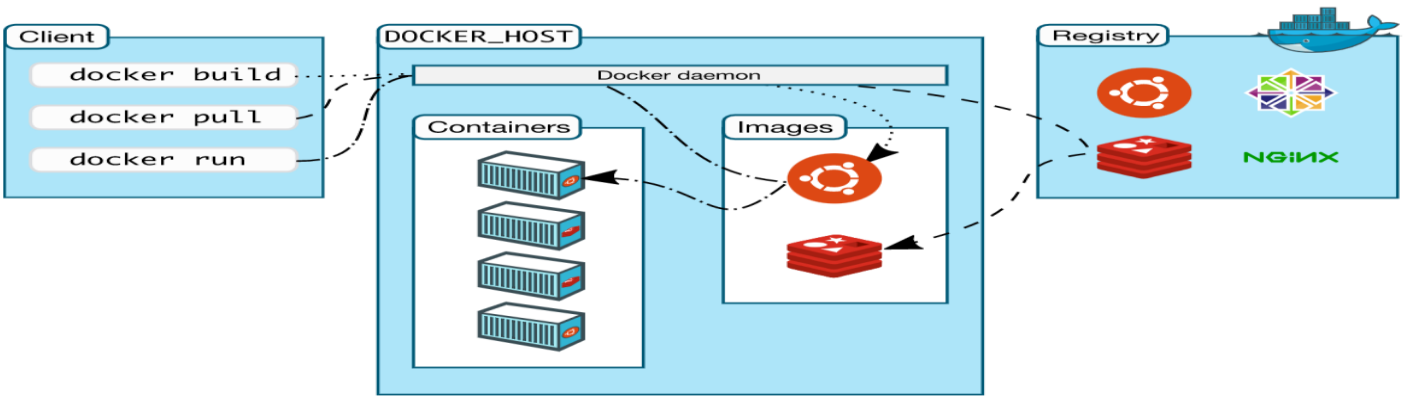
Docker streamlines the development lifecycle by allowing developers to work in standardized environments using local containers which provide your applications and services. Containers are great for continuous integration and continuous delivery (CI/CD) workflows.

Consider the following example scenario:

- Your developers write code locally and share their work with their colleagues using Docker containers.
- They use Docker to push their applications into a test environment and execute automated and manual tests.
- When developers find bugs, they can fix them in the development environment and redeploy them to the test environment for testing and validation.
- When testing is complete, getting the fix to the customer is as simple as pushing the updated image to the production environment.

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. 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.



The 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.

Docker Desktop

Docker Desktop is an easy-to-install application for your Mac, Windows or Linux environment that enables you to build and share containerized applications and microservices. Docker Desktop includes the Docker daemon (`dockerd`), the Docker client (`docker`), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.

What Is EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Features of Amazon EC2

Amazon EC2 provides the following features:

1. Virtual computing environments, known as *instances*
2. Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
3. Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*
4. Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
5. Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as *instance store volumes*
6. Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*
7. Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
8. A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
9. Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
10. Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
11. Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs).

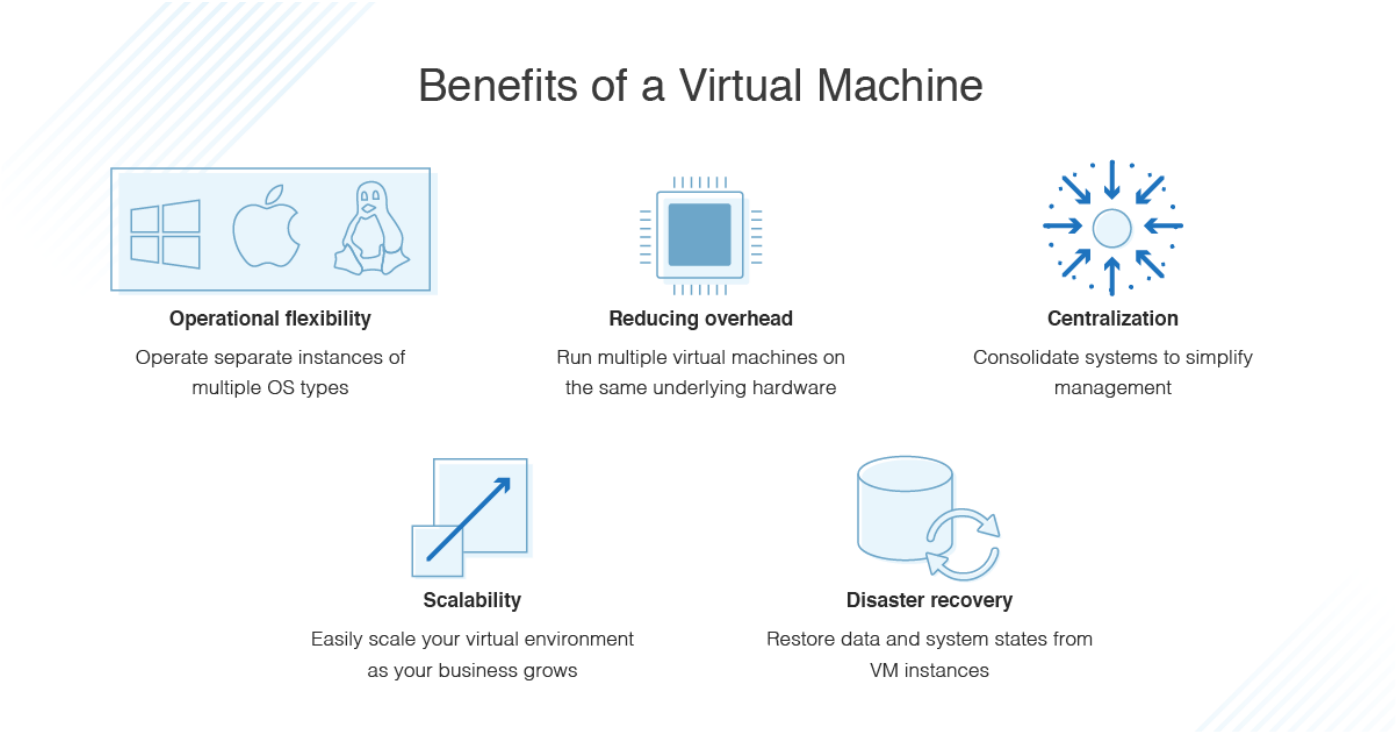
What Is Virtual Machine?

A Virtual Machine (VM), on the other hand, is created to perform tasks that if otherwise performed directly on the host environment, may prove to be risky. VMs are isolated from the rest of the system; the software inside the virtual machine cannot tamper with the host computer. Therefore, implementing tasks such as accessing virus-infected data and testing of operating systems are done using virtual machines. We can define a virtual machine as:

A virtual machine is a computer file or software usually termed as a guest, or an image that is created within a computing environment called the host.

VMs are broadly divided into two categories depending upon their use:

1. System Virtual Machines: A platform that allows multiple VMs, each running with its copy of the operating system to share the physical resources of the host system. Hypervisor, which is also a software layer, provides the virtualization technique. The hypervisor executes at the top of the operating system or the hardware alone.
2. Process Virtual Machine: Provides a platform-independent programming environment. The process virtual machine is designed to hide the information of the underlying hardware and operating system and allows the program to execute in the same manner on every given platform.



Difference Between Docker and Virtual Machine

Virtual Machines	Docker
Process in one VM can't see processes in other VMs	Process in one container can't see processes in other container
Each VM has its own root filesystem	Each container has its own root file system(Not Kernel)
Each VM gets its own virtual network adapter	Docker can get virtual network adapter. It can have separate IP and ports
VM is a running instance of physical files(.VMX and .VMDK)	Docker containers are running instances of Docker Image
Host OS can be different from guest OS	Host OS can be different from Container OS

Containers Vs VMs