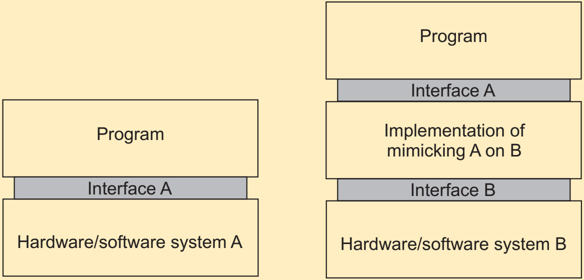
# Virtualization

## Observation

Virtualization is important:

- Hardware changes faster than software
- Ease of portability and code migration
- Isolation of failing or attacked components

## Principle: mimicking interfaces


Program
Interface A
Hardware/software system A
Program
Interface A
Implementation of mimicking A on B
Interface B
Hardware/software system B

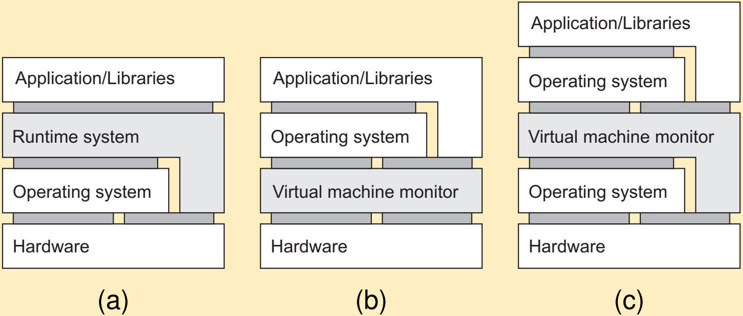
# Mimicking interfaces

### Four types of interfaces at three different levels

1. Instruction set architecture: the set of machine instructions, with two subsets:
    - Privileged instructions: allowed to be executed only by the operating system.
    - General instructions: can be executed by any program.
2. System calls as offered by an operating system.
3. Library calls, known as an application programming interface (API)

# Ways of virtualization

## (a) Process VM, (b) Native VMM, (c) Hosted VMM



## Differences

(a) Separate set of instructions, an interpreter/emulator, running atop an OS.
(b) Low-level instructions, along with bare-bones minimal operating system
(c) Low-level instructions, but delegating most work to a full-fledged OS.

# VMs and cloud computing

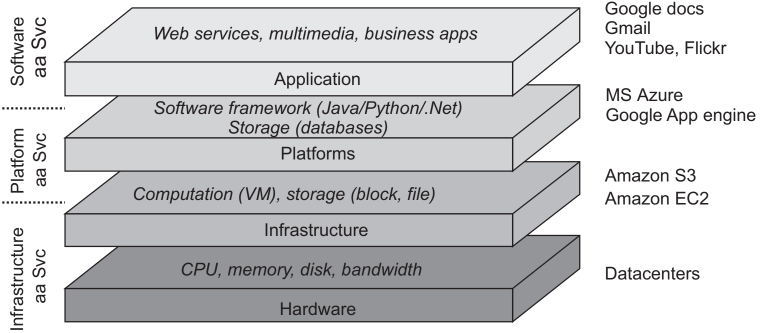
### Three types of cloud services

- Infrastructure-as-a-Service covering the basic infrastructure
- Platform-as-a-Service covering system-level services
- Software-as-a-Service containing actual applications

### IaaS

Instead of renting out a physical machine, a cloud provider will rent out a VM (or VMM) that may possibly be sharing a physical machine with other customers ⇒ almost complete isolation between customers (although performance isolation may not be reached).

# Cloud computing



# Cloud computing

## Make a distinction between four layers

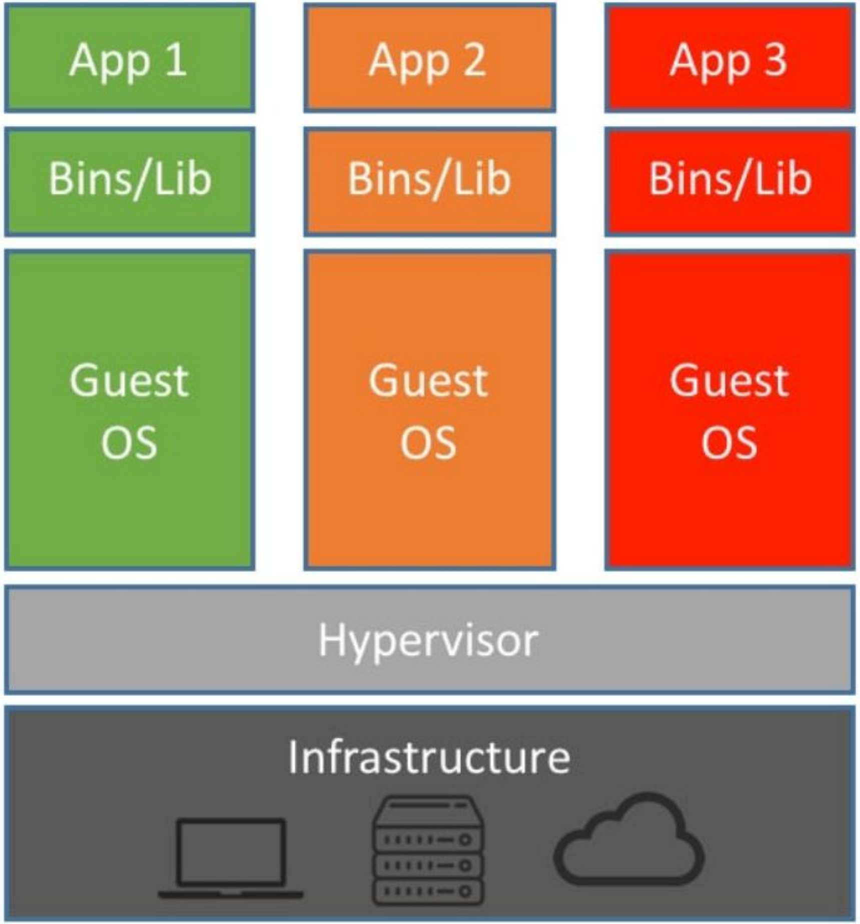
- Hardware: Processors, routers, power and cooling systems. Customers normally never get to see these.
- Infrastructure: Deploys virtualization techniques. Evolves around allocating and managing virtual storage devices and virtual servers.
- Platform: Provides higher-level abstractions for storage and such. Example: Amazon S3 storage system offers an API for (locally created) files to be organized and stored in so-called buckets.
- Application: Actual applications, such as office suites (text processors, spreadsheet applications, presentation applications). Comparable to the suite of apps shipped with OSes.

# Containers

- ***Observation***: many applications are strongly dependent on a set of (versions of) libraries and other processes than anything else.
- ***Essence***: Why not use packages of those dependencies and have apps run in isolated environments containing exactly those libraries etc.?

Instead of virtualizing the underlying hardware, **containers virtualize the operating system** (typically Linux or Windows) so each individual container contains only the application and its libraries and dependencies. Containers are small, fast, and portable because, unlike a virtual machine, **containers do not need to include a guest OS in every instance** and can, instead, simply leverage the features and resources of the host OS.

https://www.ibm.com/cloud/blog/containers-vs-vms



App 1 | App 2 | App 3

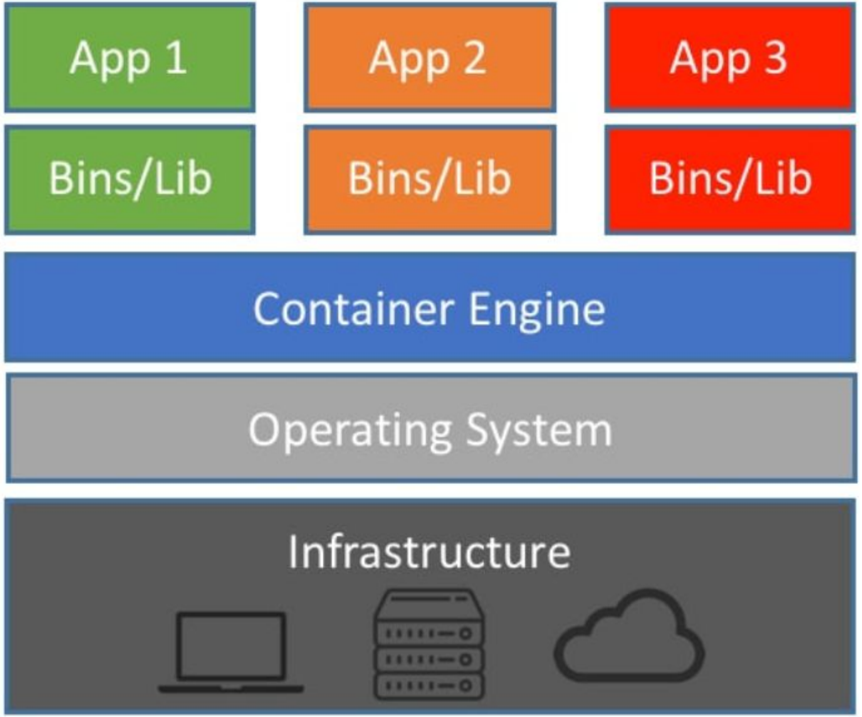
Bins/Lib | Bins/Lib | Bins/Lib

Guest OS | Guest OS | Guest OS

Hypervisor

Infrastructure

Machine Virtualization



App 1 | App 2 | App 3

Bins/Lib | Bins/Lib | Bins/Lib

Container Engine

Operating System

Infrastructure

Containers

Source: https://www.netapp.com/blog/containers-vs-vms/

- Mimic the environment of an application
- Make sure that namespaces are in order

**An example: PlanetLab**

**Explore:**

**Docker Container:**

Package Software into Standardized Units for Development, Shipment and Deployment

https://www.docker.com/resources/what-container

# Example: PlanetLab

## Essence

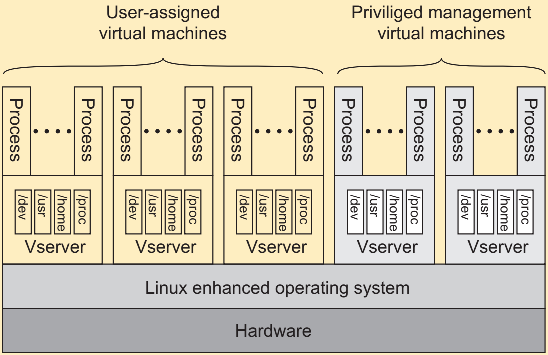
Different organizations contribute machines, which they subsequently share for various experiments.

## Problem

We need to ensure that different distributed applications do not get into each other’s way ⇒ virtualization

# PlanetLab basic organization

## Overview


User-assigned virtual machines
Priviliged management virtual machines
Process
....
Process
/dev
/usr
/home
/proc
Vserver
Linux enhanced operating system
Hardware

## Vserver

Independent and protected environment with its own libraries, server versions, and so on. Distributed applications are assigned a collection of vservers distributed across multiple machines