IIT Roorkee

eict IIT Roorkee

IntelliPaat

Advanced Certification in Cloud Computing & DevOps

Virtualization

Agenda

01 Virtualization

02 Virtual Machine

03 Hardware virtualization

04 Hypervisor

05 Open Flow

06 Containerization and Docker

07 Docker Components

08 Foundation Engine

Cloud Computing – Course (CSN-520)

Dr. Neetesh Kumar

Assistant Professor (MTech. & PhD, JNU)

Department of Computer Science & Engineering (IIT-R)

Write to me: neetesh@cs.iitr.ac.in

Web link: https://www.iitr.ac.in/~CSE/Neetesh_Kumar

Google Scholar: https://scholar.google.co.in/citations?user=Ut_l01AAAAAJ&hl=en

Source Contents Credit to:

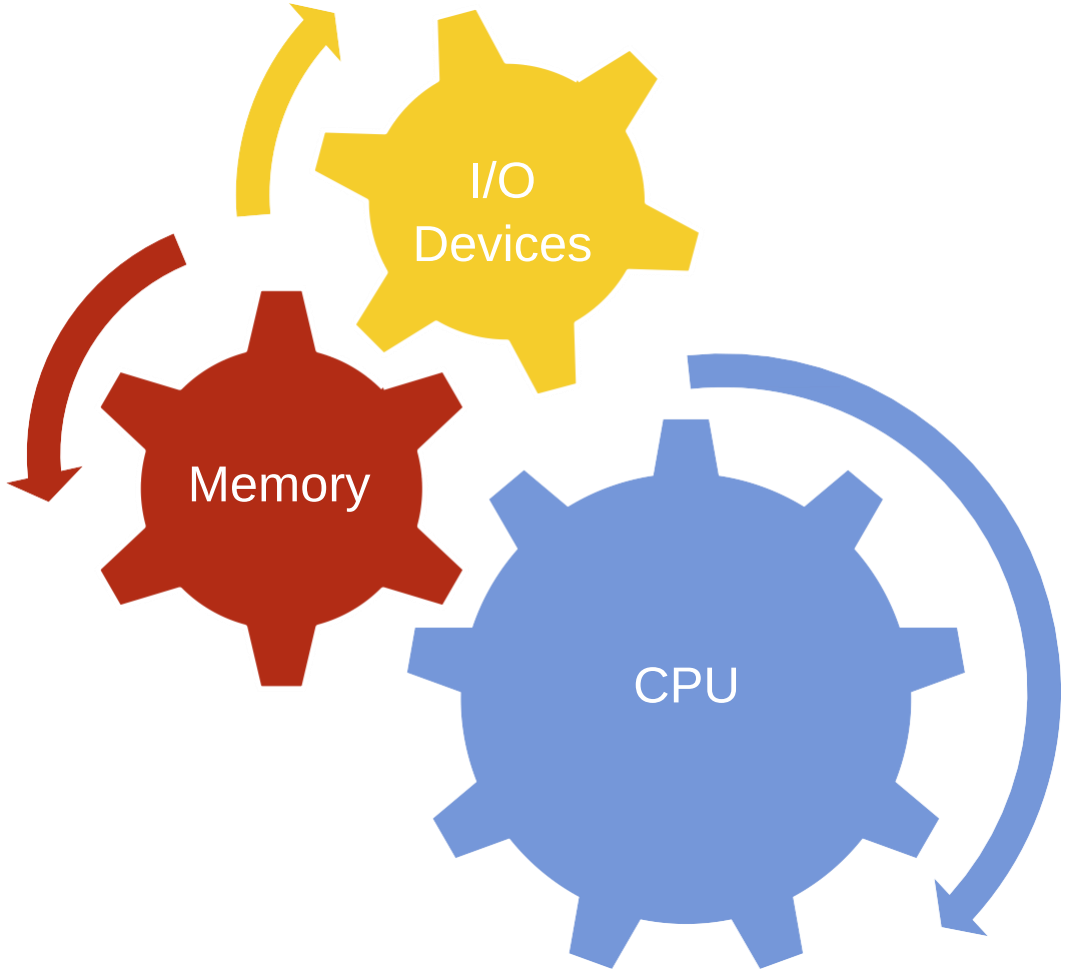
- **Pro. Rajkumar Buyya** (*University of Melbourne, Australia*.)
- **Prof. D. P. Vidyarthi** (*Jawaharlal Nehru University (JNU), India*.)
- **Dr. Rajiv Ranjan and Devki Nandan Jha** (*Computing Science, Newcastle University, UK*.)

List of books

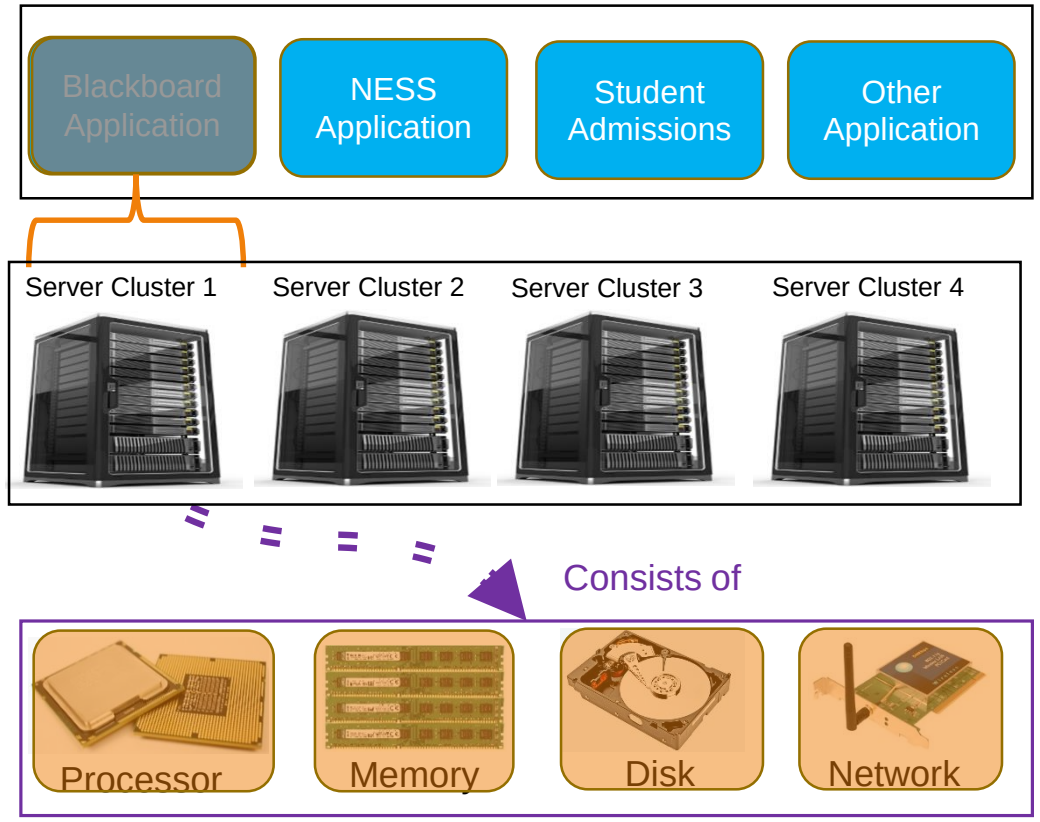
- Mastering Cloud computing, Rajkumar Buyya, Christian Vacchiola, S Thamarai Selvi, McGraw Hill.
- Advance Computer Architect: Parallelism, Scalability, Programmability or Scalable Parallel Computing: Technology, Architecture, Programming by Kai hawang at el.
- Cloud Computing Principles and Paradigms, Rajkumar Buyya, James Broberg, Andrzej Goscinski, Wiley Publishers
- Rest of book resources are mentioned in your Syllabus.

Virtualization

COMPUTER ORGANISATION

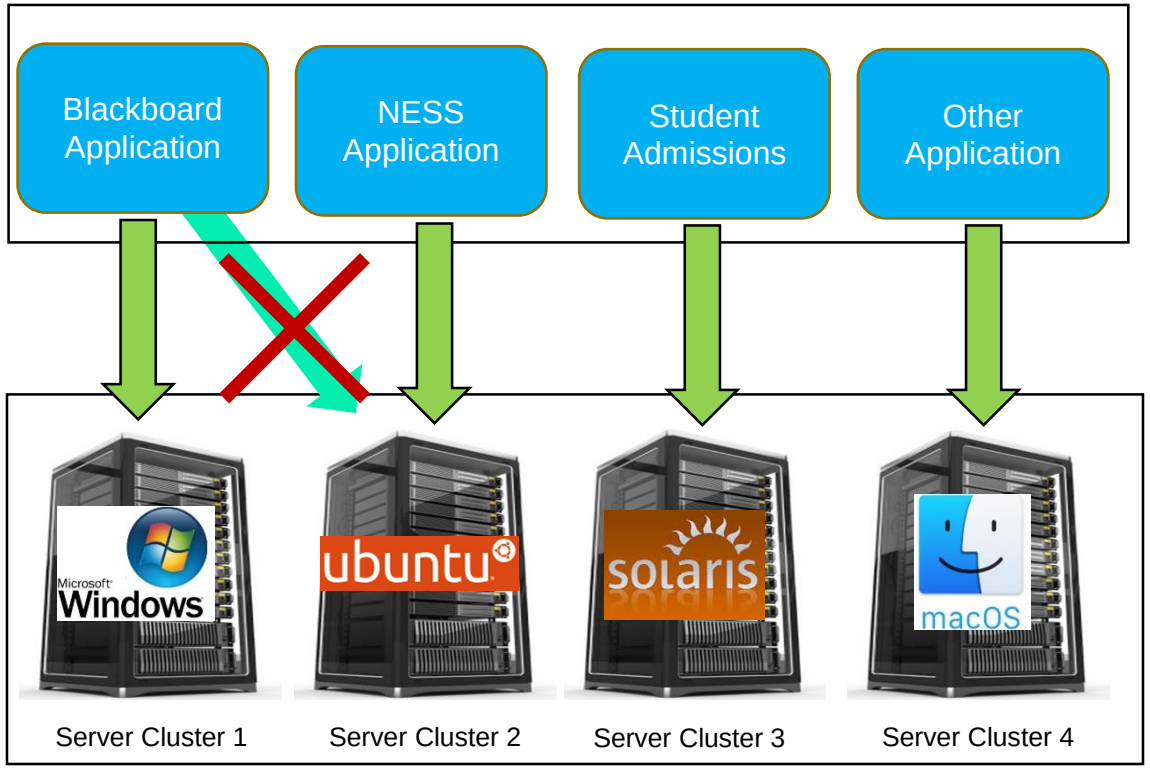


Non Virtualized System

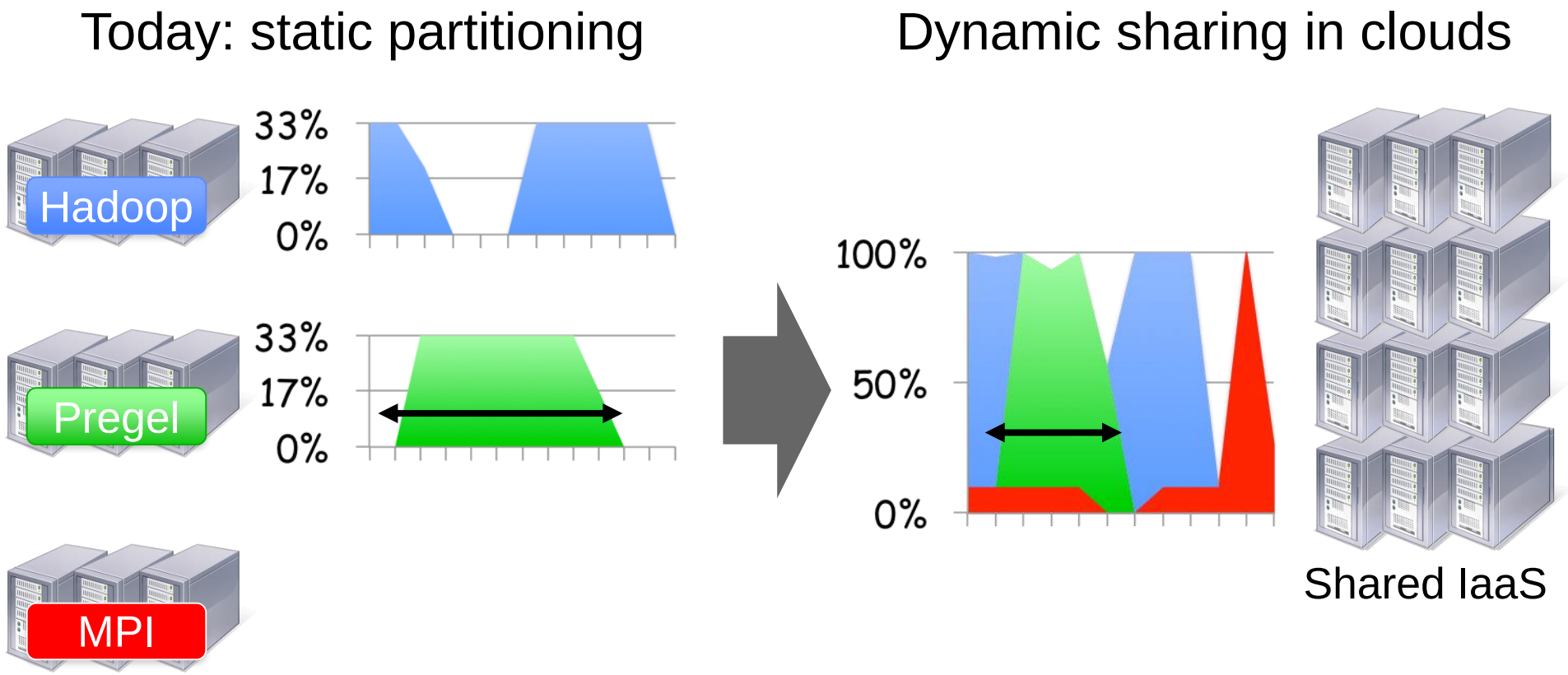


Hardware Failure

Non Virtualized System

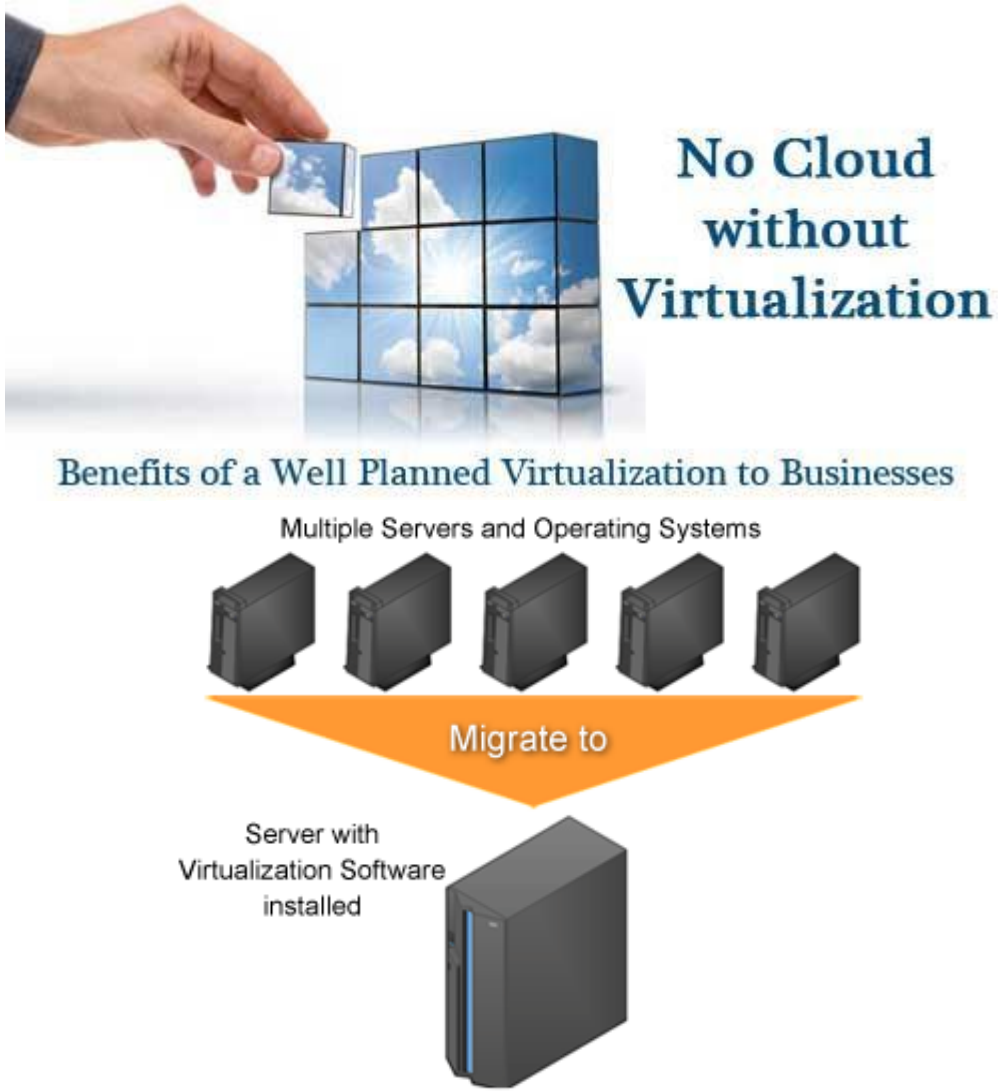


What do we want to achieve with cloud?



Benefits of Virtualisation

- Heterogeneous system consolidation
- System testing and workload isolation
- Hardware optimization
- Application high availability
- Elasticity



Virtualization (Overview)

- Fundamental component of cloud computing, especially in IaaS.
- Allows the creation of secure, customizable, and isolated execution environment, even if untrusted, **NOT** affecting other users' applications.
- **Basic idea:** ability of a computer program (software and hardware) **to emulate** an executing environment separate from the one that hosts such programs.
- For example, Windows OS can run on top of a virtual machine, which itself is running on Linux OS.
- A great opportunity to build elastically scalable systems
- Provision capability with minimum costs.
- Virtualization used to deliver customizable computing environments on demand.

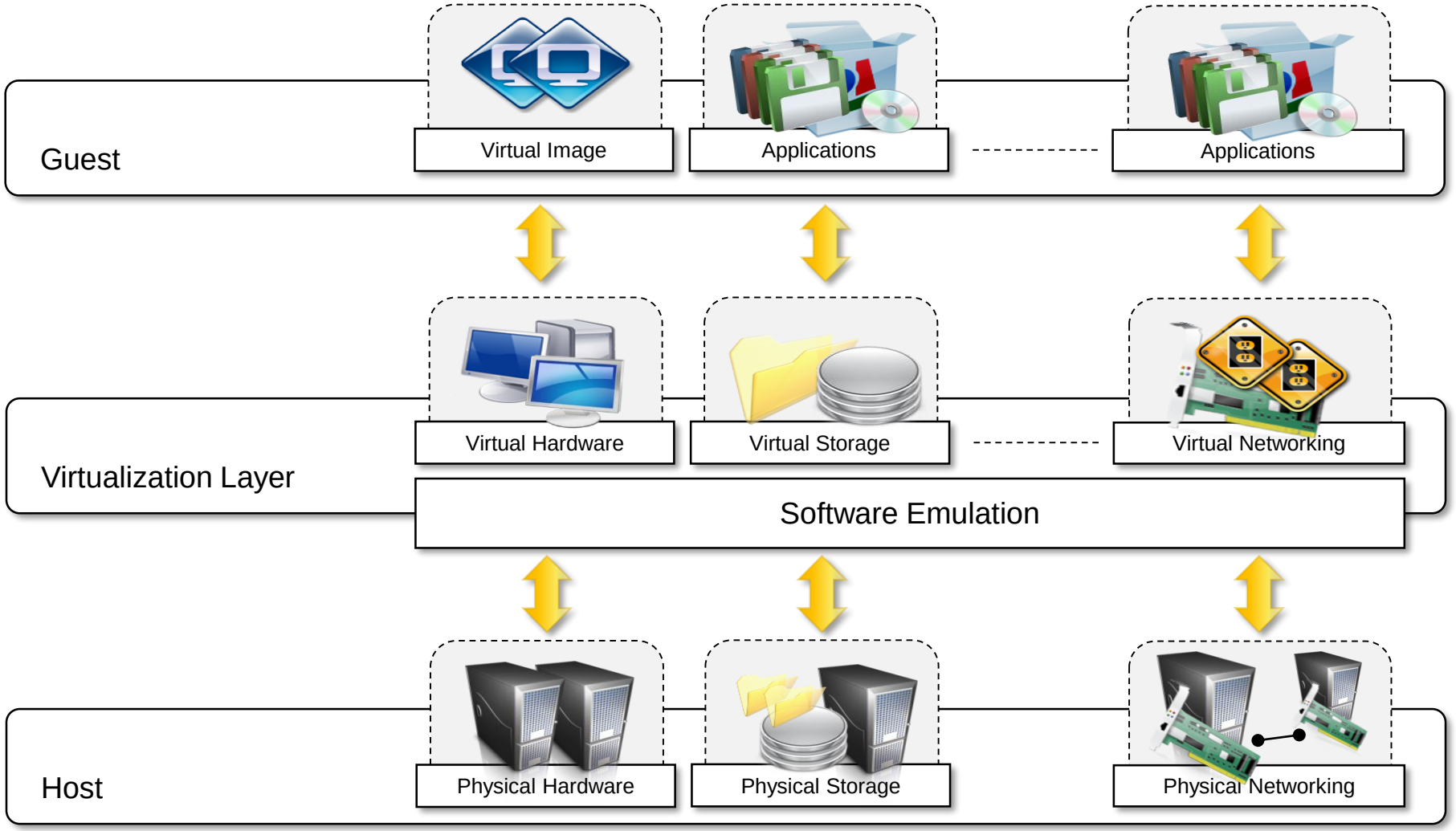
Virtualization (Overview)

- Large umbrella of technologies & concepts to provide abstract environment (**virtual hardware: processing elements (PEs), storage, memory, and networking or an operating system**).
- Synonymous with hardware virtualization to deliver IaaS.
- Operating system level, programming language level and application level.
- **Interested phenomena:**
  - Increased performance and computing capacity.
  - Underutilized hardware and software resources.
  - Lack of space.
  - Greening initiatives.
  - Administrative costs.

Virtualization Reference Model

- Virtual version of hardware, a software environment, storage, or a network.
- Three major components: Guest, Host and, Virtualization layer.
- The Guest represents the system component that interacts with the virtualization layer.
- Host represents original environment, the guest is supposed to be managed.
- Virtualization layer recreates the environment where the guest will operate.
- Most intuitive application is represented by hardware virtualization, the guest is represented by a system image comprising an operating system and installed applications.

Virtualization Reference Model

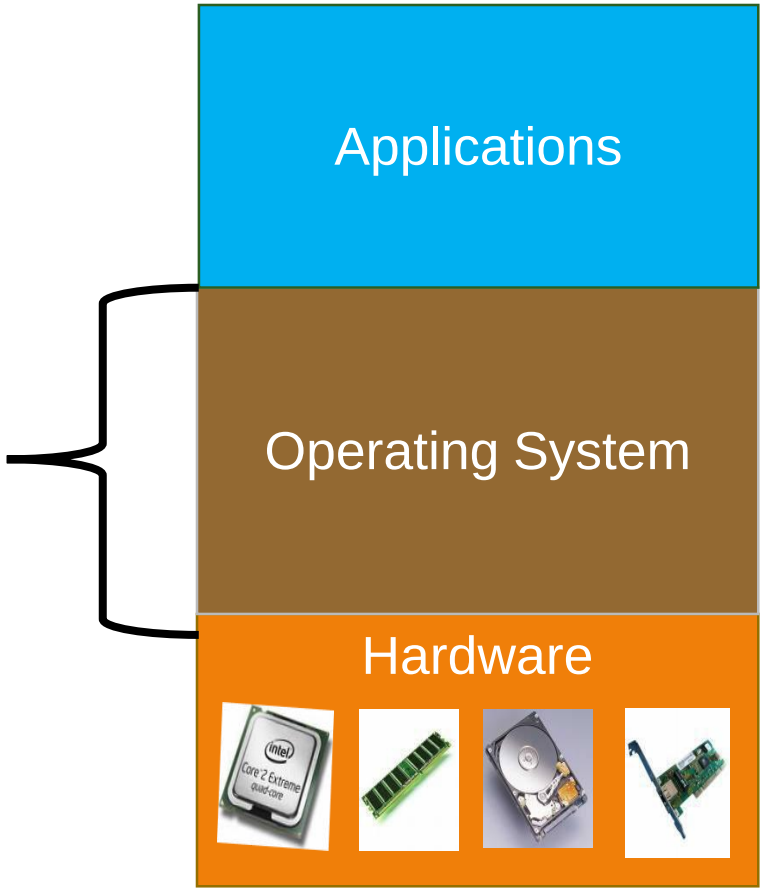


Virtualization Reference Model

- Guests are installed on top of virtual hardware, controlled and managed by the virtualization layer (virtual machine manager ((VMM)).
- Host is instead represented by the physical hardware, and in some cases the operating system that defines the environment where the VMM is running.
- **Virtual storage**: guest might be client applications interact with the virtual storage management software deployed on top of the real storage system.
- Virtual networking: guest interacts with a virtual network (VPN) managed by specific software (VPN client) using the physical network on the node.
- VPNs are useful for creating the illusion of being within a different physical network and thus accessing the resources in it, which would otherwise not be available.

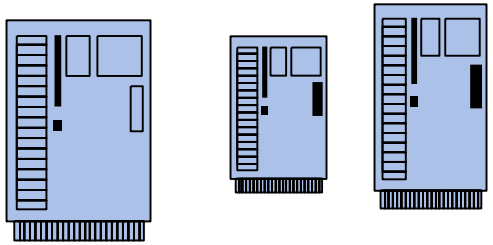
Virtualisation in Conventional Software Infrastructure

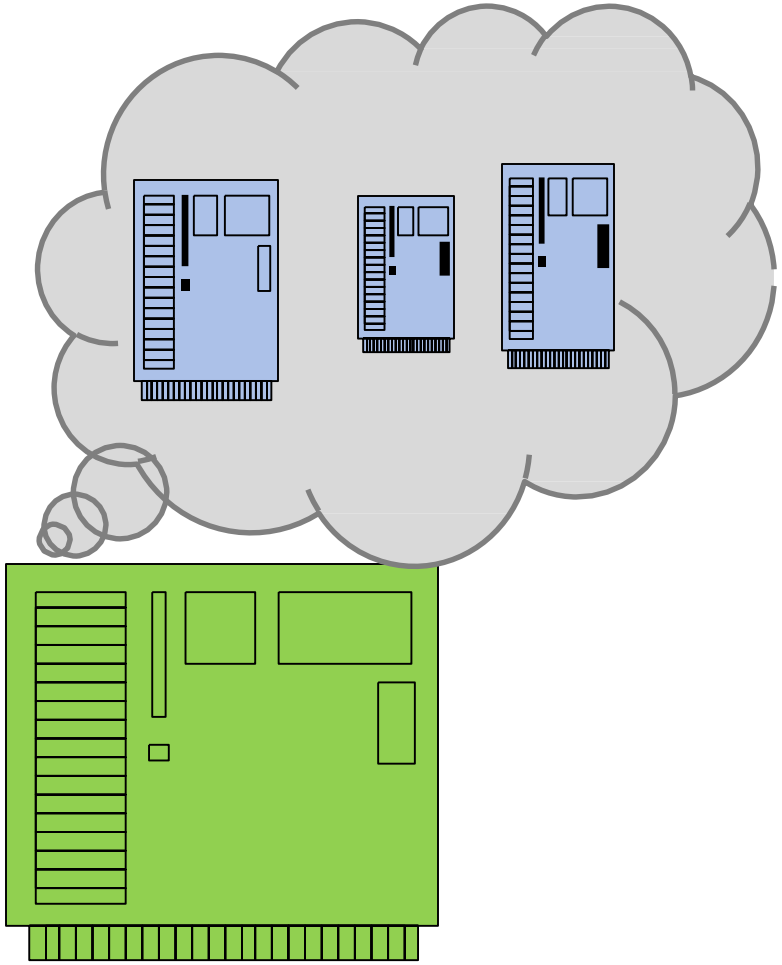
Physical Resource	Virtual Resource
CPU	Process
Memory	Virtual Memory
Disk	Files
Network Card	TCP/UDP Sockets
Screen	Windows



What do we want to achieve with Virtualization in clouds

For one thing, replace several machines with (a bigger) one without changing any software.





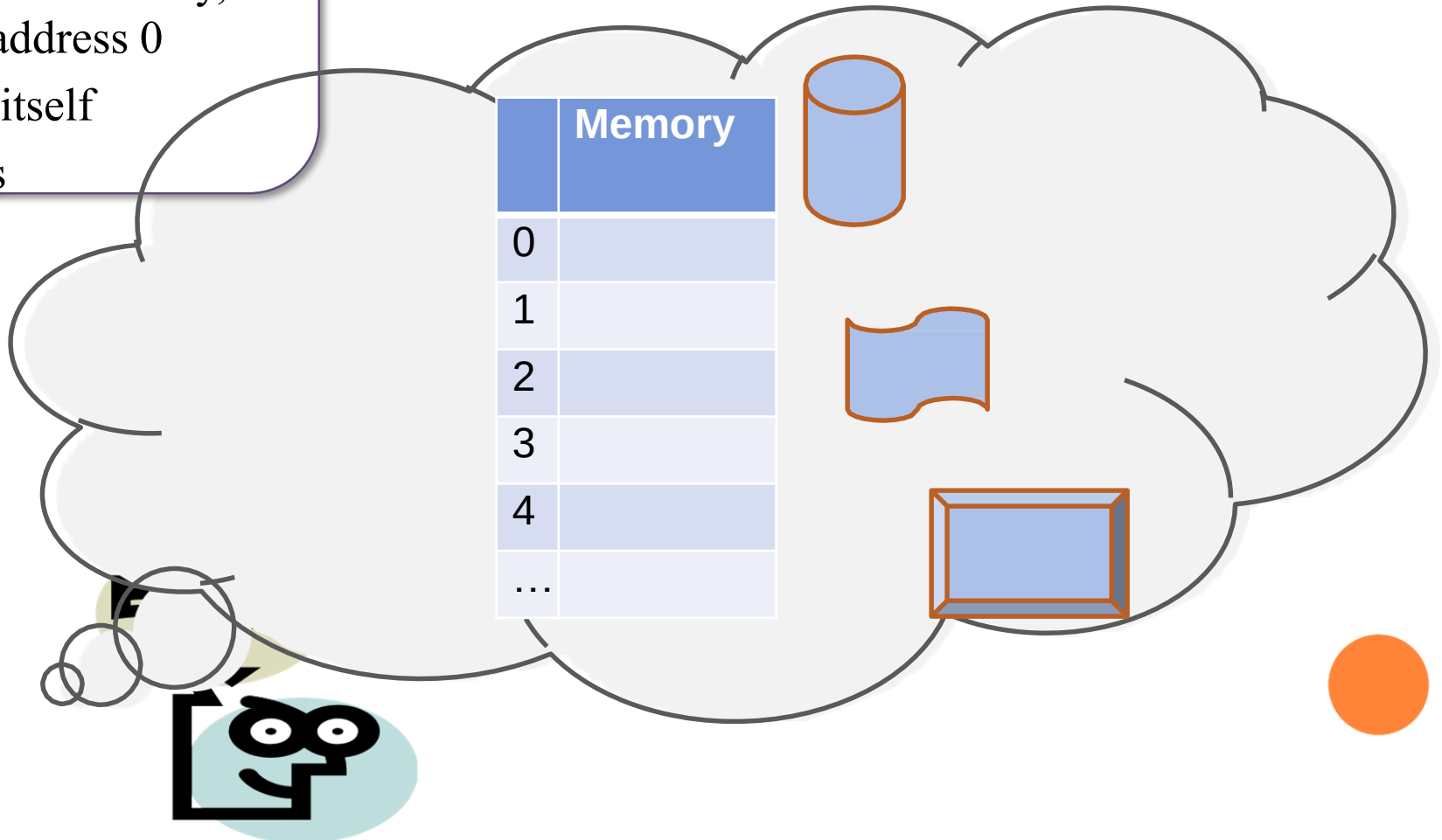
IBM did that with VM/370 in 1972

WHY DO THAT?

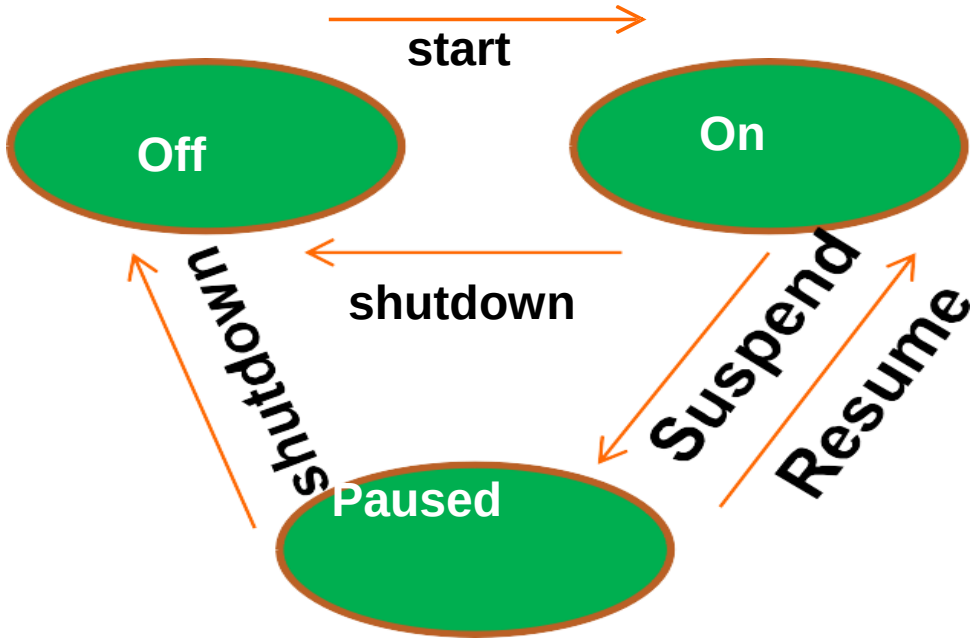
- Save the cost (space, energy, personnel) of running several machines in place of one—a **green** aspect, too!
- Use the (otherwise wasted) CPU power
- *Clone* servers (for example, for debugging) at low cost
- *Migrate* a machine (for example, when the load increases) at low cost
- *Isolate* an appliance—a server for a specific purpose (such as security)—without buying new hardware

VIRTUALIZATION IN MODERN OPERATING SYSTEM

- Each virtualized cloud application thinks that it has
  - Huge contiguous memory, starting from address 0
  - CPU power to itself
  - All I/O devices



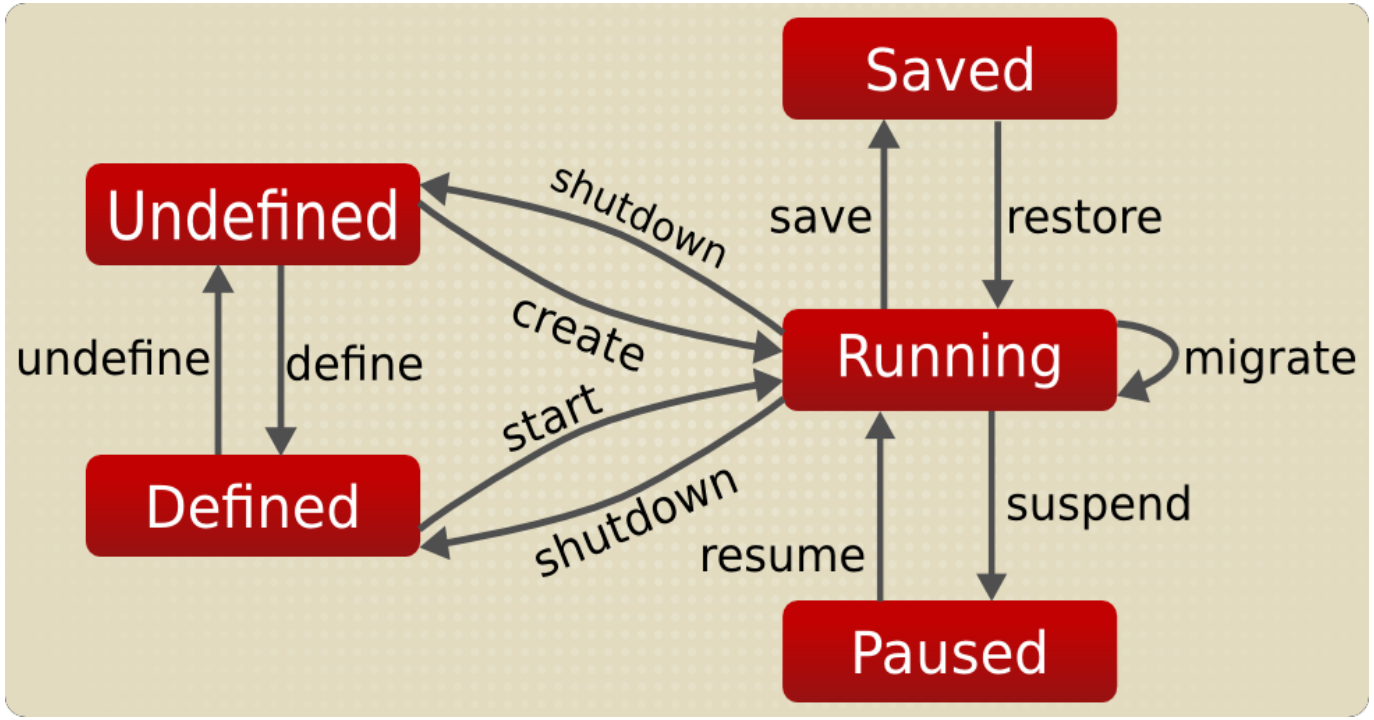
CASE STUDY: LIFECYCLE



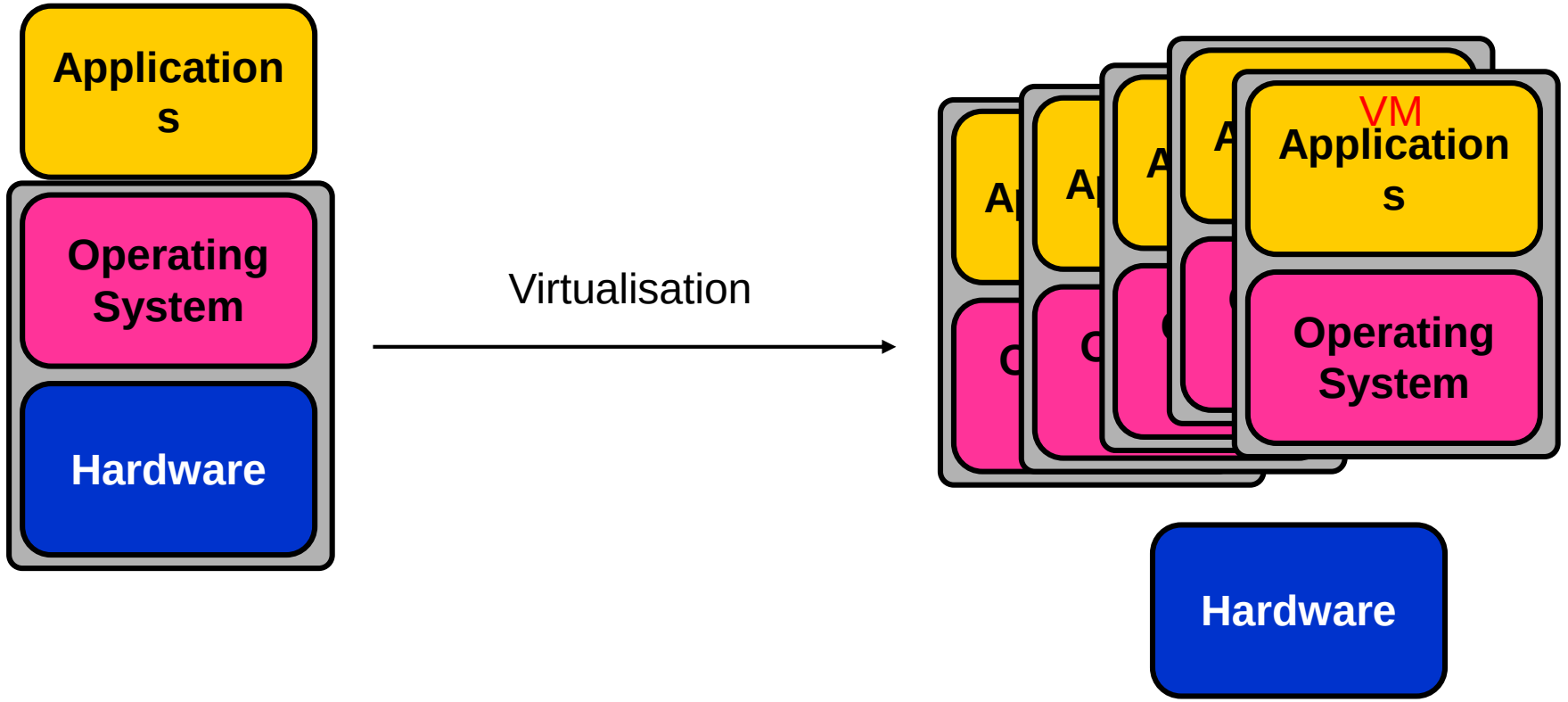
Physical Machine

Virtual Machine (VM)

after
http://wiki.libvirt.org/page/VM_lifecycle
Virtual_Machine_Lifecycle



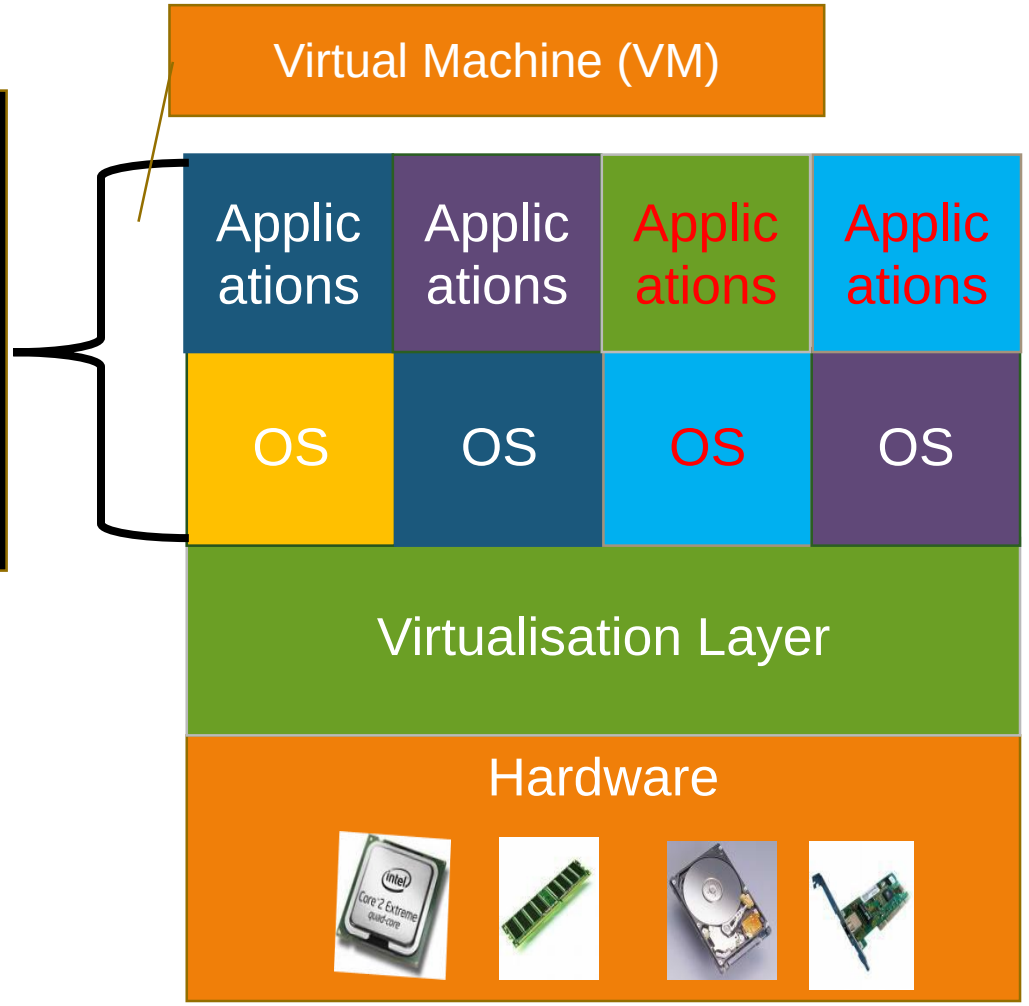
Virtualization: a Technique for Multiple Applications/OS



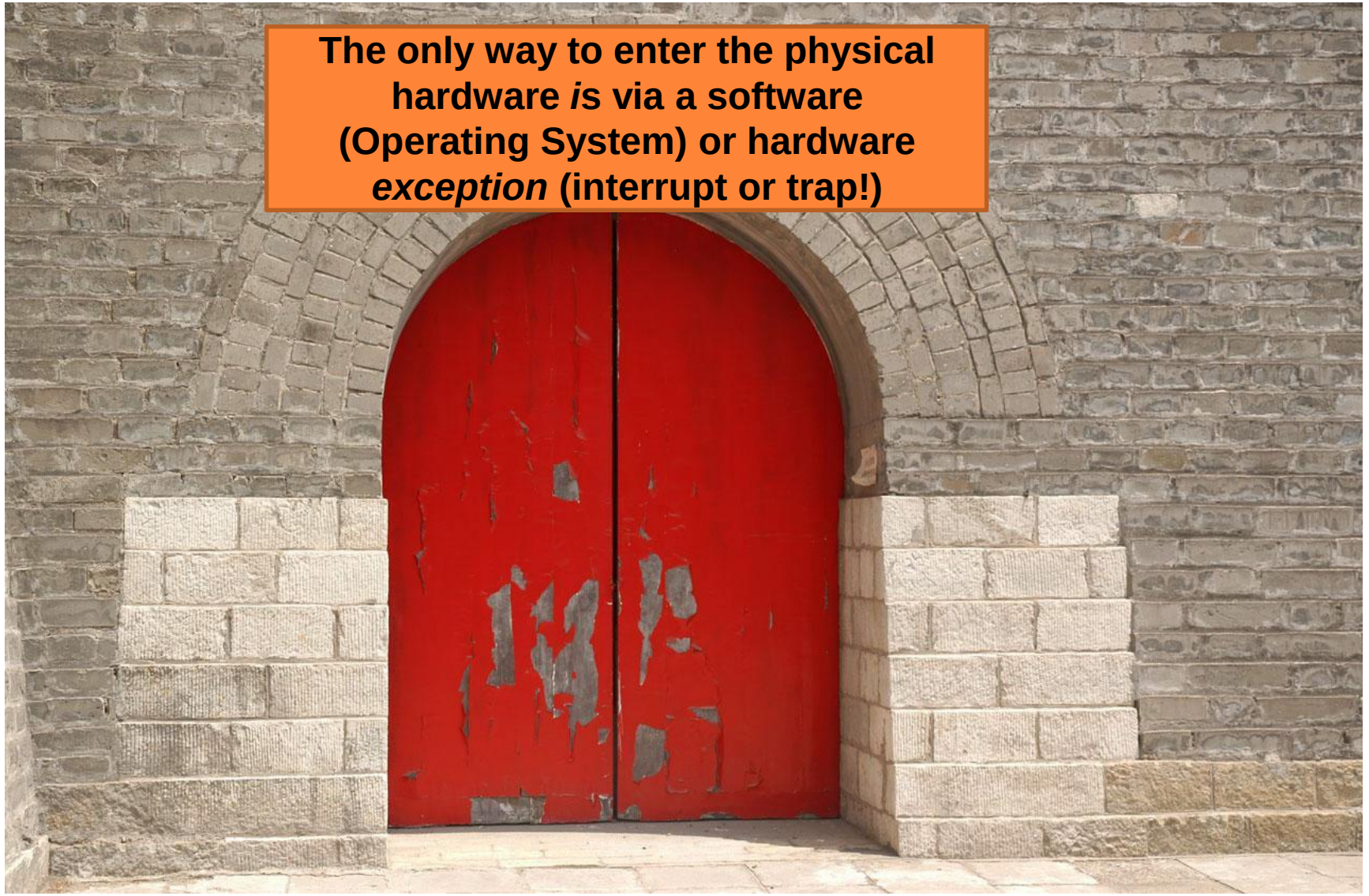
Virtualised Infrastructure in Clouds

NOW: Physical Machine
(CPU+Mem+Disk+Net+…)

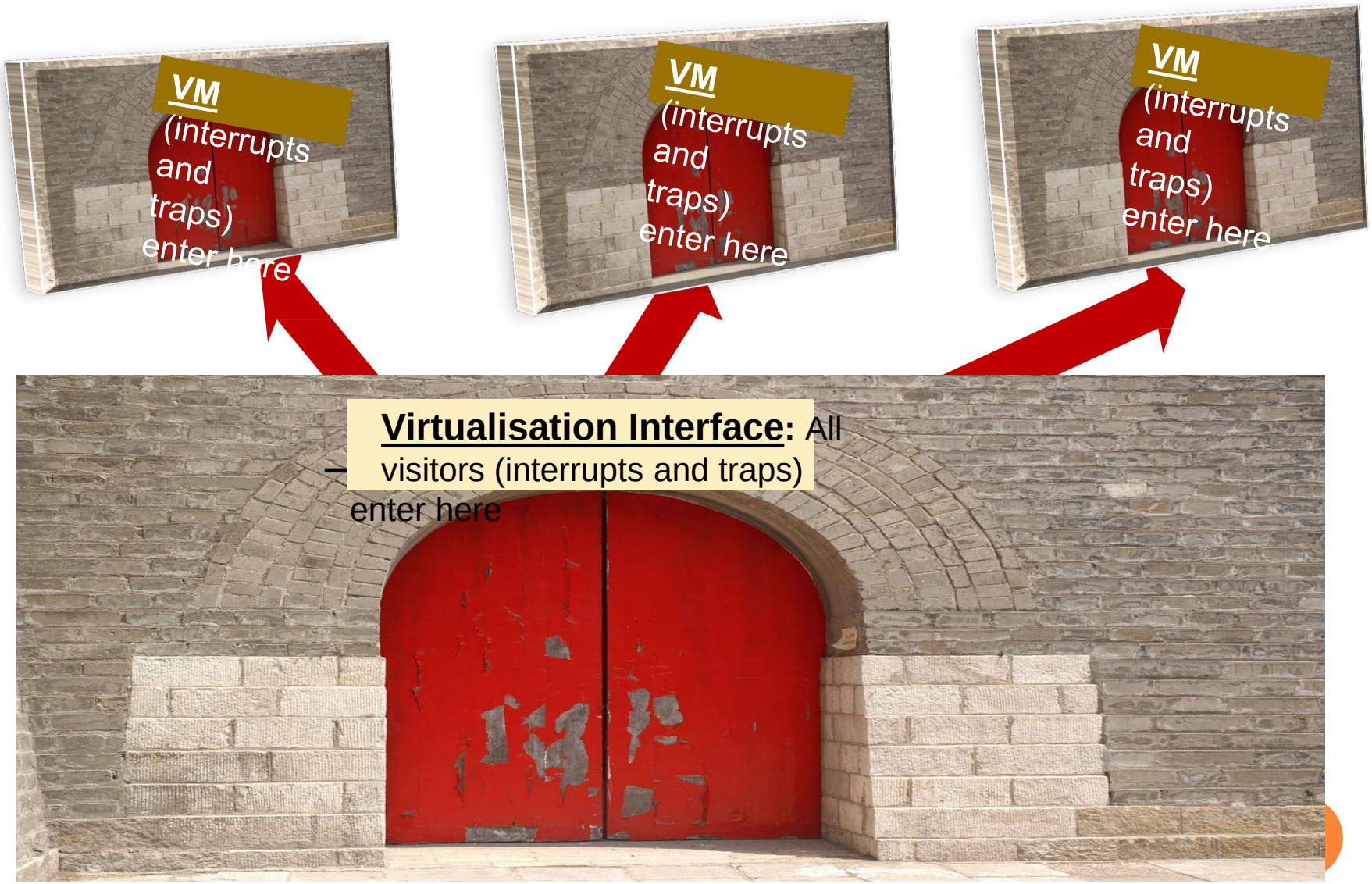
Virtual Cloud Machine
(VCPU+Vmem+Vdisk+Vnet+…)



Virtualised Infrastructure in Clouds

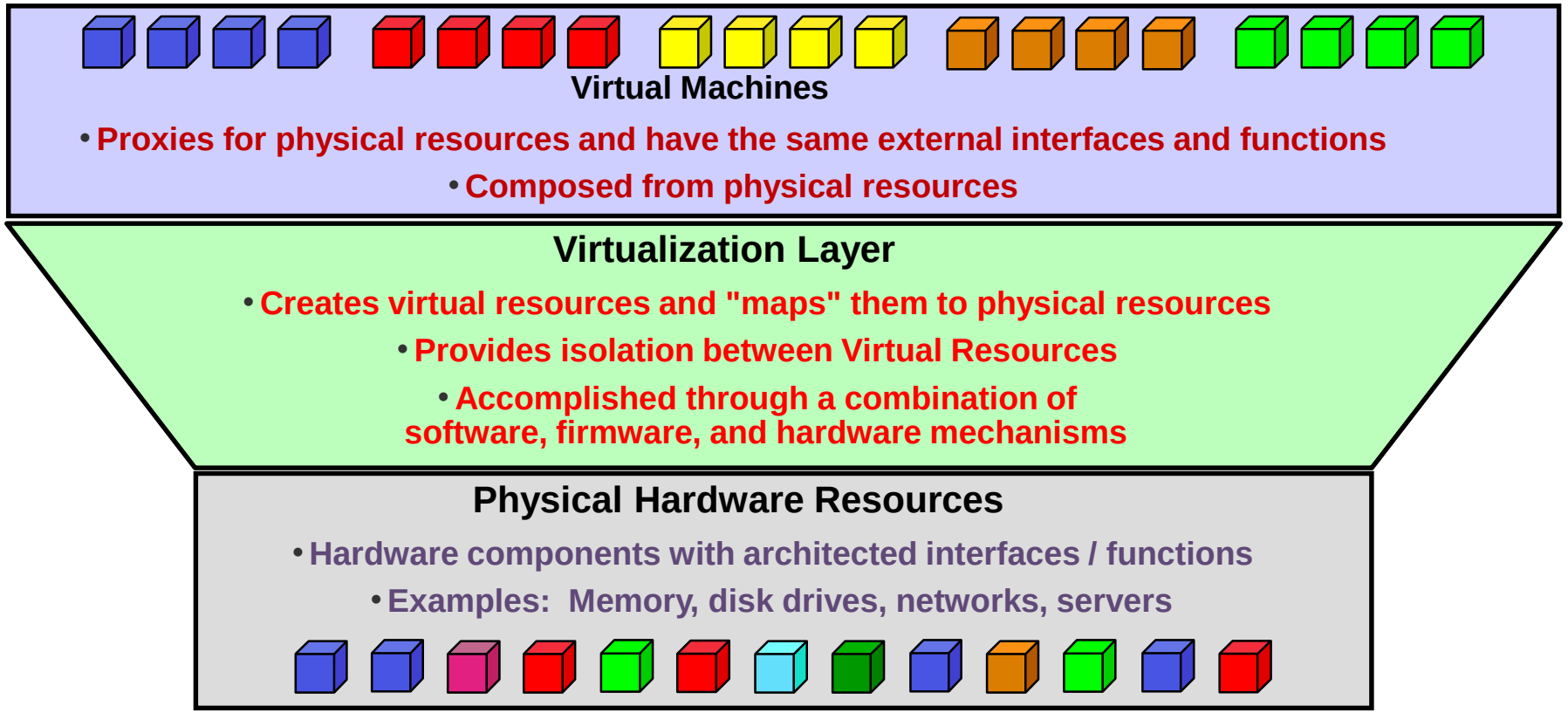


Virtualised Infrastructure in Clouds



Virtualised Infrastructure in Clouds

Server Virtualization – The division of a physical system's resources (e.g. processors, memory, I/O, and storage), where each such set of resources operates independently with its own System Image instance and applications



Characteristics of virtualized environments

Increased security :

- For guest, a completely transparent and controlled execution environment.
- An emulated environment in which the guest is executed.
- Guest operations performed on VM, It translates and applies them to the host.
- VMM controls and filters the guest activity, thus preventing some harmful operations from being performed, Host is hidden or protected from the guest.
- Sensitive information naturally hidden without installing complex security policies.
- Increased security is a requirement when dealing with untrusted code. For example, applets downloaded from the Internet run in a sandboxed version o f the Java Virtual Machine(JVM), which provides them with limited access to the hosting operating system resources.

Characteristics of virtualized environments

Increased security :

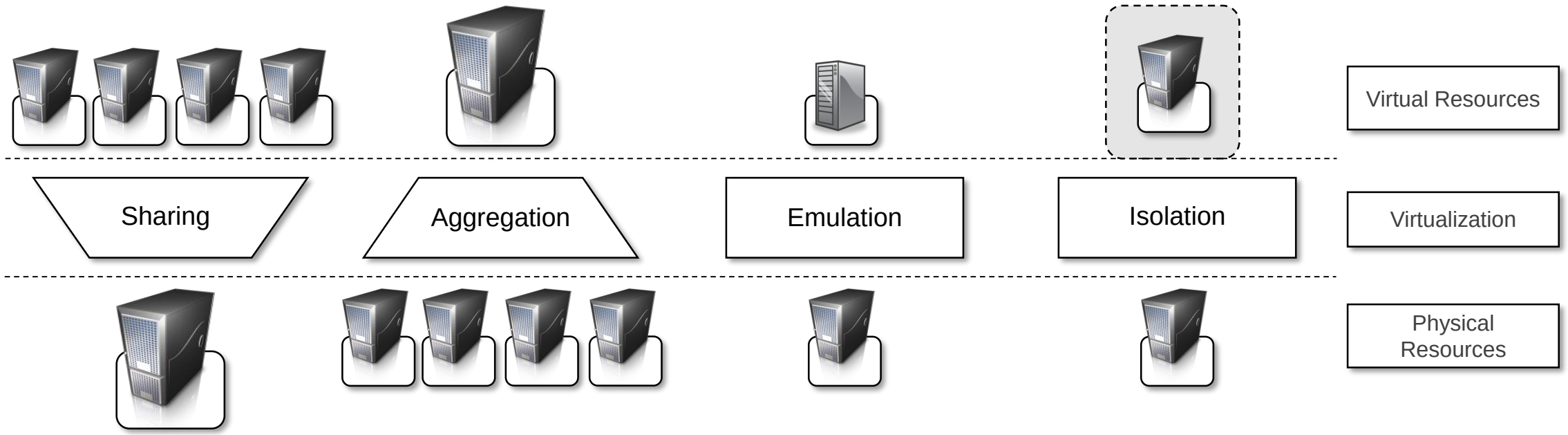
- Hardware virtualization solutions (VMware Desktop, Virtual Box, and Parallels) create a virtual computer with customized virtual hardware on top of which a new operating system can be installed.
- By default, the file system exposed by the virtual computer is completely separated from the one of the host machine.

Characteristics of virtualized environments

Managed execution :

- Sharing: sharing in virtualized data centers, to reduce the number of active servers and limit power consumption
- Aggregation: A group of separate hosts tied together and represented to guests as a single virtual host, implemented in middleware for distributed computing.
- Emulation Controlling and tuning the environment that is exposed to guests.
- Isolation Allows multiple guests to run on the same host without interfering with each other, Also, Separation between the host and the guest.

Characteristics of virtualized environments



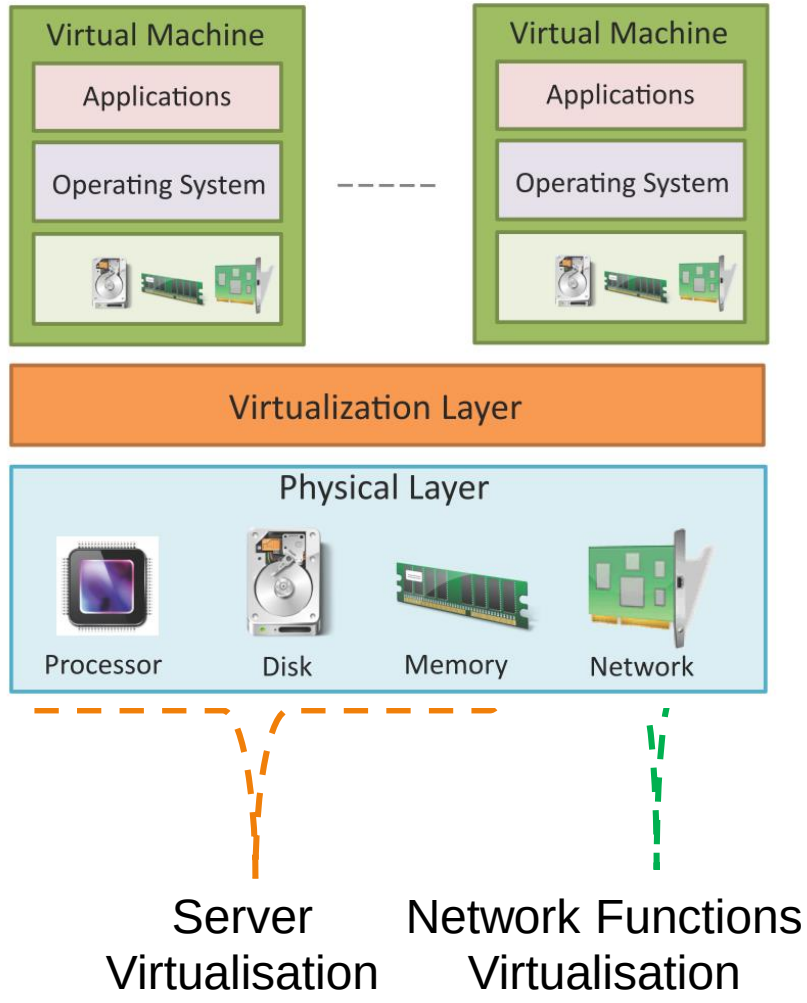
Characteristics of virtualized environments

Portability:

- **Hardware virtualization solution:** guest packaged into a virtual image that can be safely moved and executed on top of different virtual machines.
- **Programming-level virtualization: I**mplemented by JVM or .NET runtime, binary code can be run without any recompilation.
- This makes flexible and very straight forward: One version is able to run on different platforms with no changes.
- Allows your own system always with you and ready to use as long as the required virtual machine manager is available (services you need available to you anywhere you go).

Virtualization in Clouds

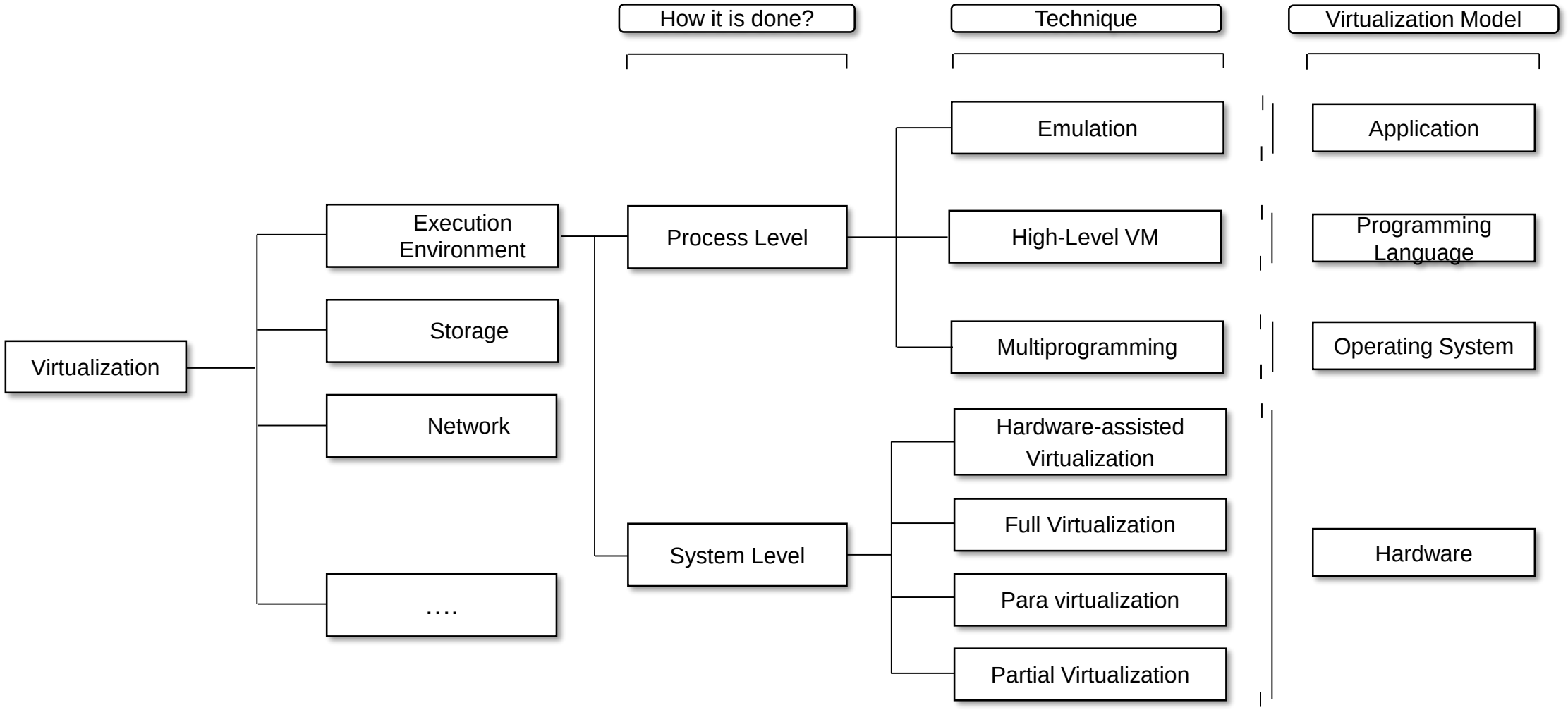
- Virtualization refers to the partitioning the resources of a physical computing hardware (such as computing, storage, network and memory) into multiple virtual resources or virtual machines (VMs).
- Key enabling technology of cloud computing that allow pooling of resources.
- In cloud computing, resources are pooled to serve multiple users using multi-tenancy.
- There are two major types of Virtualization approaches of cloud hardware at physical layer
  - Server Virtualization
  - Network Functions Virtualization



Virtual Machine (VM)

- VM can be implemented by adding a software layer to a real machine to support desired architecture, can be applied to entire machine, lie at architected interfaces.
- **Architecture:** a formal specification to an interface in the system, including the logical behavior of the resources managed via the interface.
- Implementation describes the actual embodiment of an architecture.
- Abstraction levels correspond to implementation layers, having its own interface or architecture.
- **Guest** – system component interacts with Virtualization Layer.
- **Host** – original environment where guest runs.
- **Virtualization Layer** – recreate the same or different environment where guest will run.

Virtualization



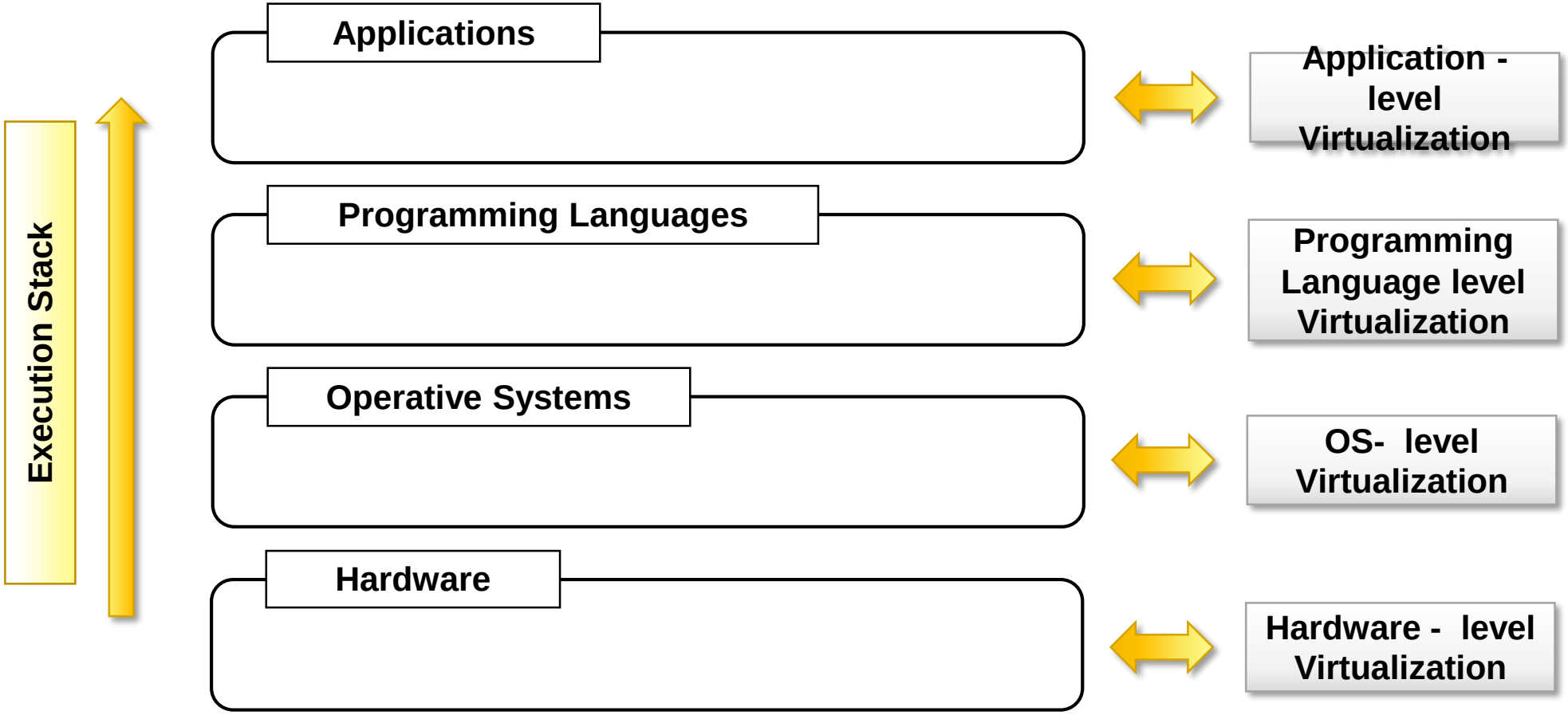
Execution Virtualization

- It emulates an Execution Environment (EE), separated from host, implemented on top of H/W by OS, application (libraries) dynamically or statically.

Machine reference model

- Virtualizing an execution environment at different levels of **computing stack** requires a reference model that defines interfaces between levels of abstractions.
- Virtualization techniques actually replace one of the layers.
- A clear separation between layers simplifies their implementation.
- It requires the emulation of interfaces and interaction with underlying layer.

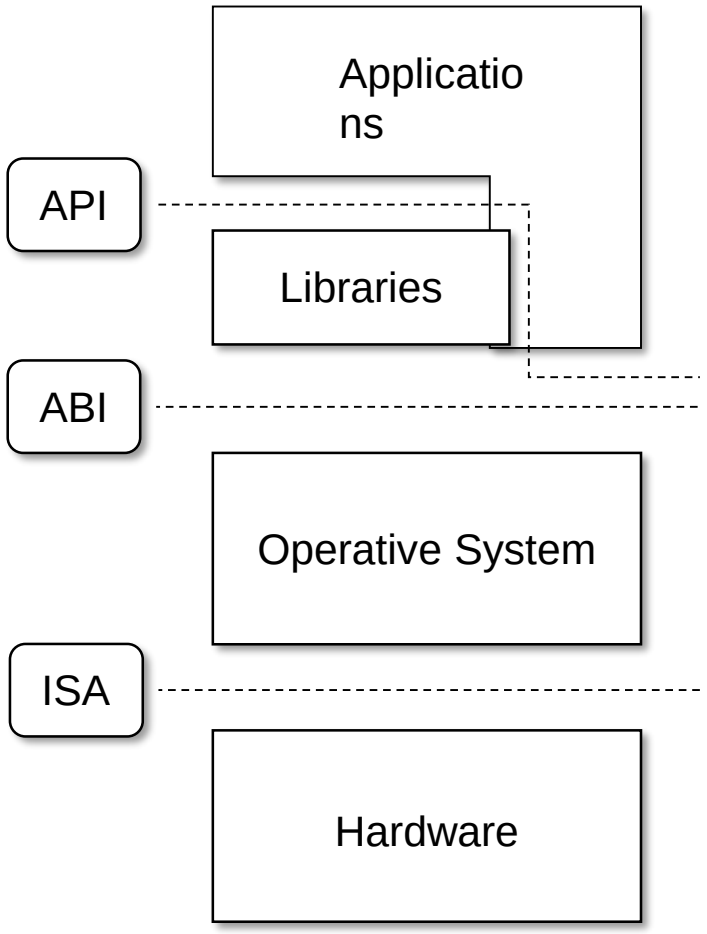
Computing Stack

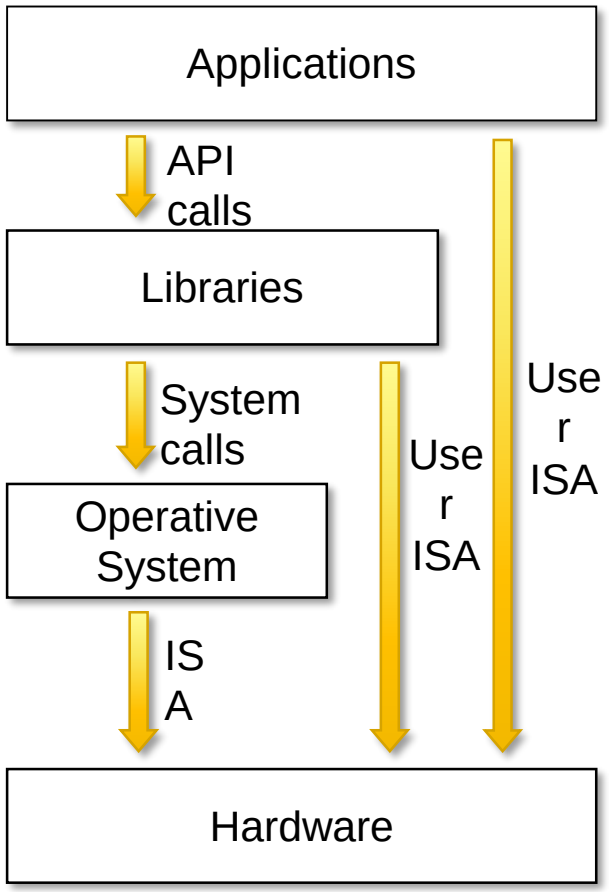


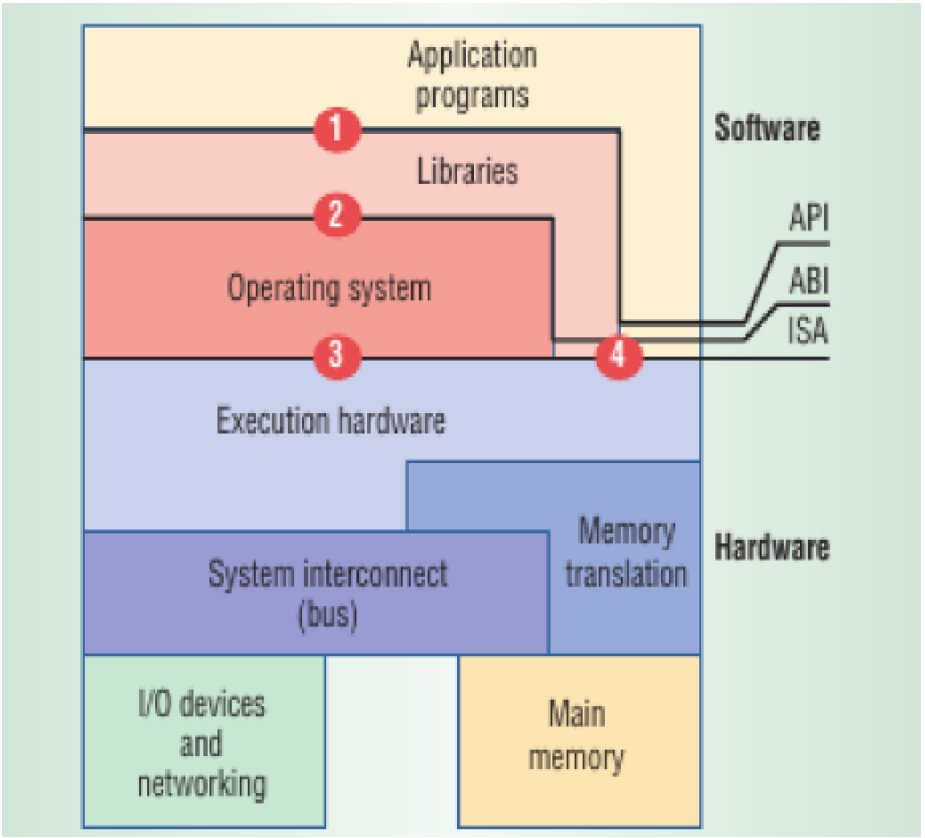
Execution virtualization: Interfaces

- **Virtualization:** extend or replace an existing interface to mimic the behavior of another system.
- **Instruction Set Architecture (ISA) :** processor, registers, memory and the interrupt management or H/W.
- **Application Binary Interface (ABI):** separates OS layer from application and libraries which are managed by the OS, System Calls defined, allows portability of applications and libraries across OS.
- **Application Programming Interfaces (API):** interfaces applications to libraries and/or the underlying OS.
- ISA has been divided into two security classes**: Privileged** Instructions and **Non-privileged** Instructions.

Machine Reference Model





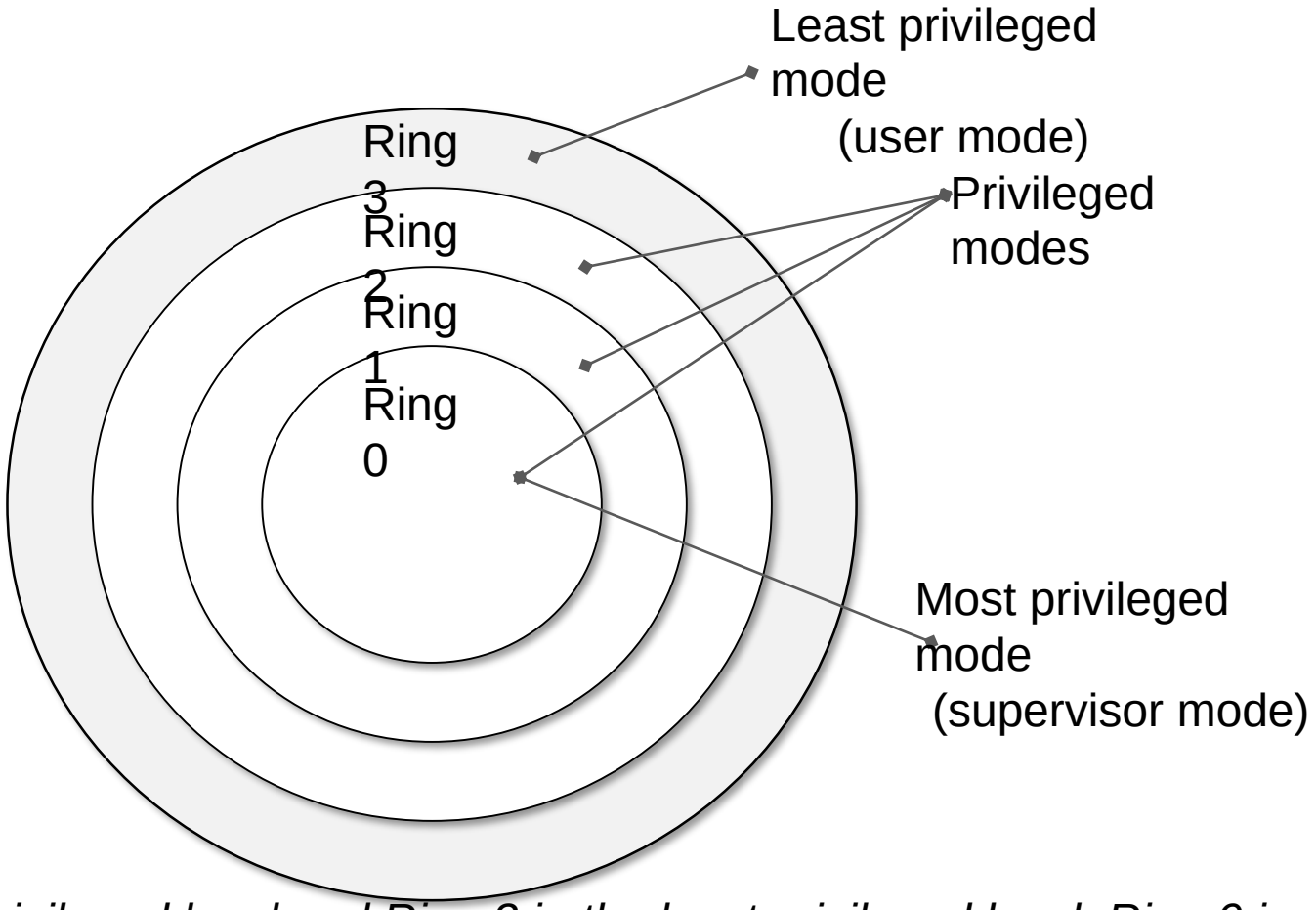


- Depending on what is replaced /mimicked, we obtain different techniques of virtualization.

Execution virtualization

- **Non-privileged** instructions can be used without interfering with other tasks because they do not access shared resources (floating, fixed-point, and arithmetic instructions).
- **Privileged instructions** are executed under specific restrictions and mostly used for sensitive operations: behavior-sensitive: operate on the I/O) or control-sensitive: alter the state of the CPU.
- Some architectures feature more than one class of privileged instructions
- For instance, a hierarchy of privileges see (*Fig. Next Slide*), in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3.

Security Rings and Privilege Modes.



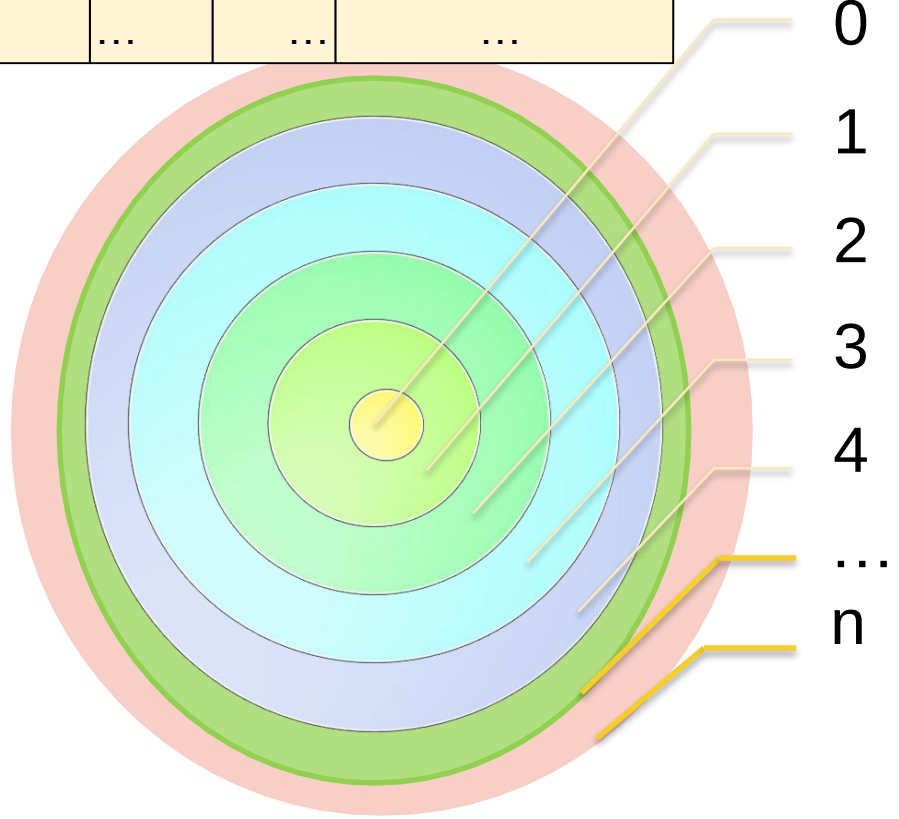
Ring 0 is in the most privileged level and Ring 3 in the least privileged level. Ring 0 is used by the kernel of the OS, rings 1 and 2 are used by the OS-level services, and Ring 3 is used by the user.

Security Rings and Privilege Modes.

Segment Table revisited

… …	…		…
Start address	size	**ring**	Access type
…	…	…	…

= {read, write, execute}



Code permitted to access ring *i* may access ring *j* if and only if *j* > *i*,

Execution virtualization

- Distinction between user & supervisor mode signifies role of the **hypervisor.**
- Hypervisors run in supervisor mode, and division between privileged and non-privileged instructions makes challenging design of virtual machine managers.
- Sensitive instructions execute in privileged mode, requires supervisor mode to avoid traps.
- Without this assumption it is impossible to fully emulate and manage the status of the CPU for guest operating systems.
- This is not true for the original ISA, allows 17 sensitive instructions to be called in user mode, prevents multiple OSs managed by a single hypervisor to be isolated from each other.
- Since, they access the privileged state of the processor and change it.

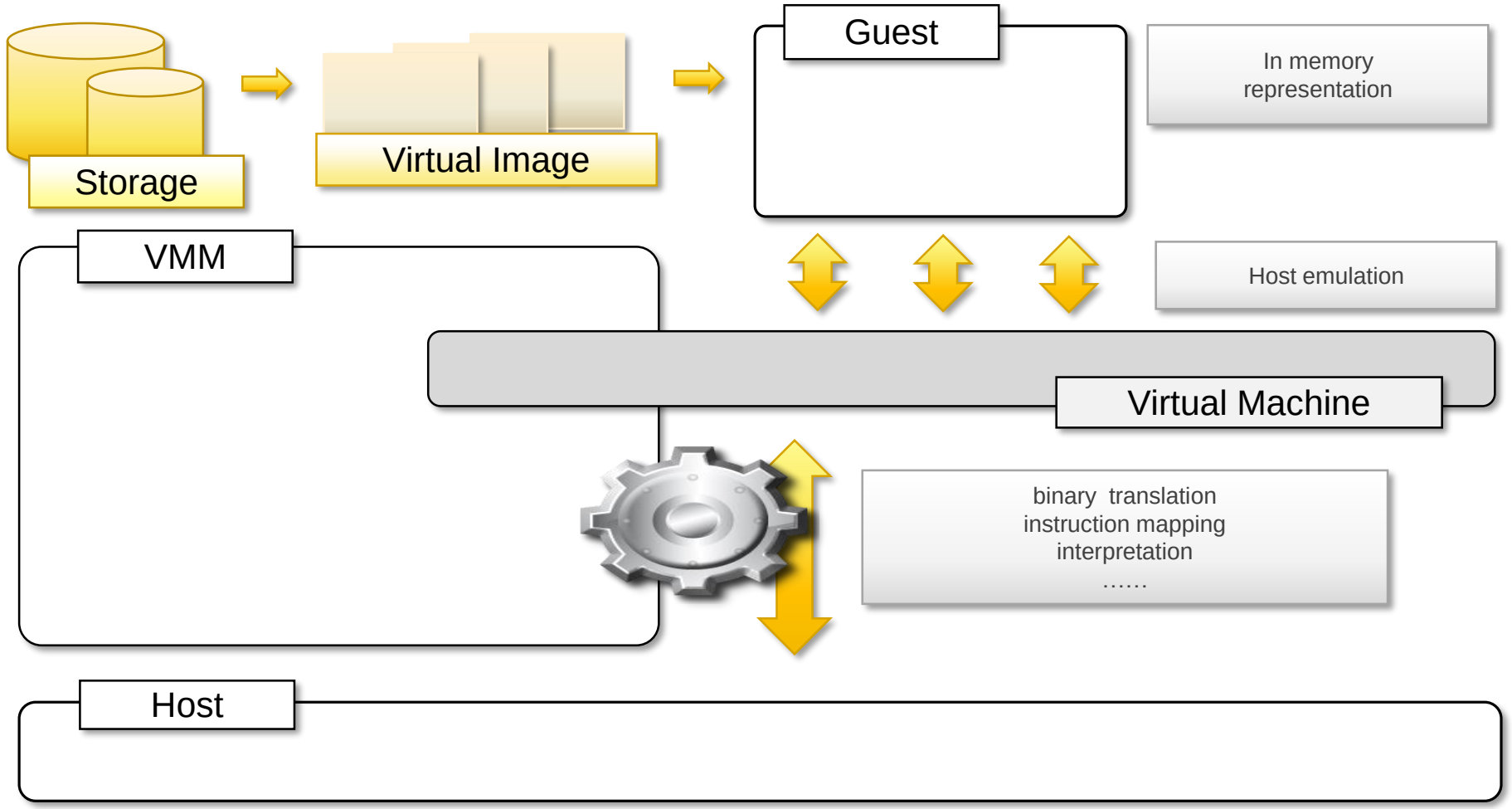
Execution virtualization

- Recent implementations of ISA (**Intel VT and AMD Pacifica**) solved this problem by **redesigning such instructions as privileged ones**.
- In hypervisor-managed environment, all guest OS code will run in user mode in order to prevent it from directly accessing the status of the CPU.
- If there are sensitive instructions that can be called in user mode, it is no longer possible to completely isolate the guest OS.

Hardware Virtualization

- It provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can be run.
- The **guest** is represented by the operating system, the **host** by the physical computer hardware, the **virtual machine** by its emulation, and the **virtual machine manager** by the hypervisor (see Figure).
- The **hypervisor** is generally a **program** or a combination of **software** and **hardware** that allows the abstraction of the underlying physical hardware.
- Hardware-level virtualization is also called System level Virtualization.
- Since, it provides ISA to virtual machines, which is the representation of the hardware interface of a system.
- This is to differentiate it from process virtual machines, which expose ABI to virtual machines.

A hardware virtualization referen



Hardware Virtualization

Hypervisors(or VMM):

- Hypervisor runs above the physical hardware.
- It runs in supervisor mode.
- It recreates a h/w environment in which guest operating systems are installed.
- It is a piece of s/w that enables us to run one or more VMs on a physical server(host).
- Two major types of hypervisor
- – ***Type -I***
- – ***Type-II***

Hardware Virtualization

Hypervisors (Type-I):

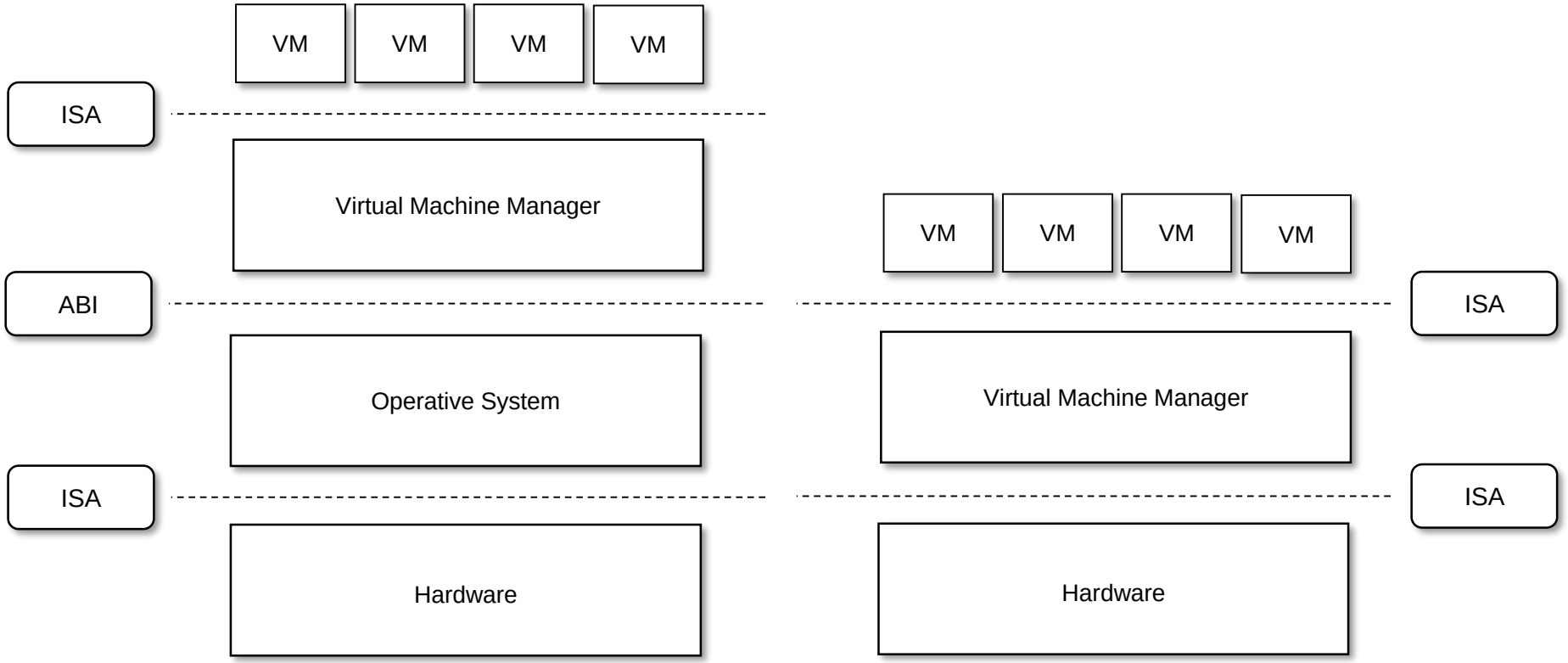
- It runs directly on top of the hardware, takes place of OS.
- Directly interact with the ISA exposed by the underlying hardware, and emulate this interface in order to allow the management of guest OS.
- Also known as *native virtual machine.*

Hypervisors (Type-2):

- It requires the support of an operating system to provide virtualization services.
- Programs managed by the OS, interact with OS through the ABI.
- Emulate the ISA of virtual h/w for guest OS.
- Also called hosted virtual machine.

No modification to Operating System code for both Type-1 and Type-2 hypervisors

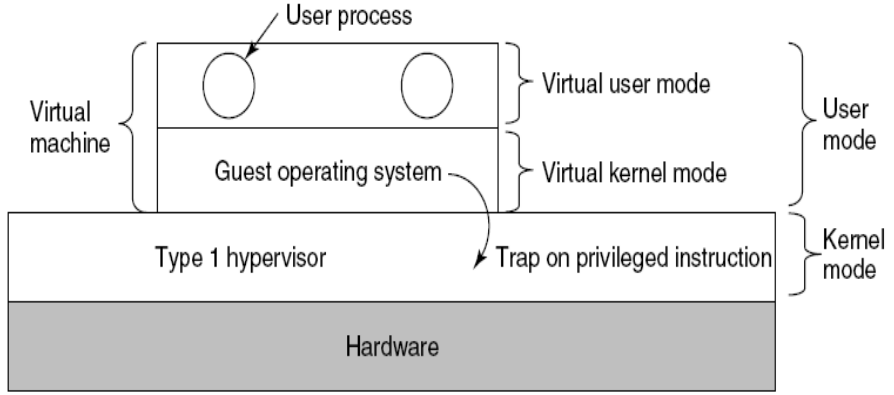
Hosted and *native Hypervisors*



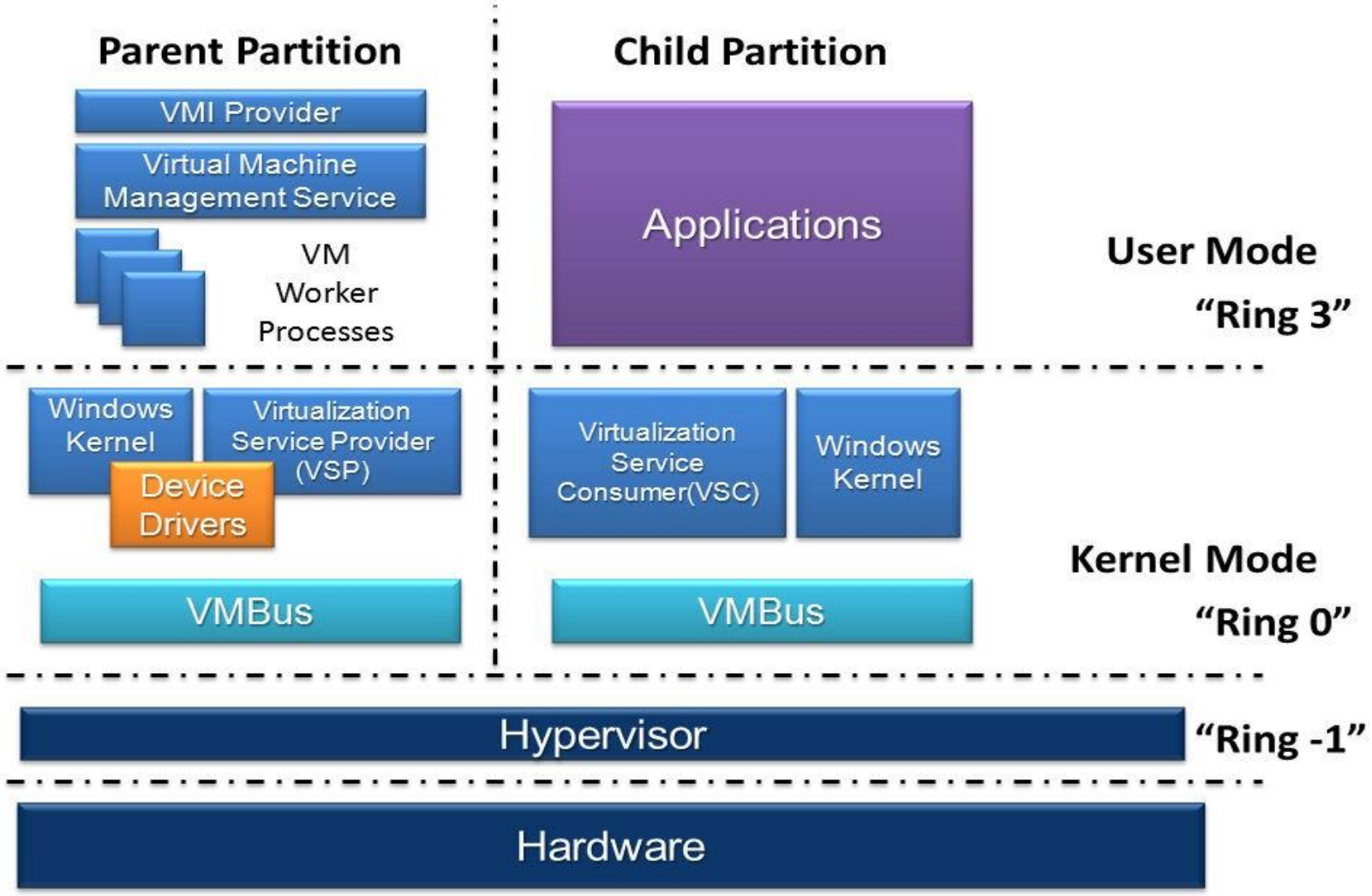
Type 1 Hypervisor

- Unmodified Operating System is running in user mode
  - But it thinks it is running in kernel mode (*virtual kernel mode)*
  - privileged instructions trap;
  - Hypervisor is the “real kernel”
    - Upon trap, executes privileged operations
    - Or emulates what the hardware would do

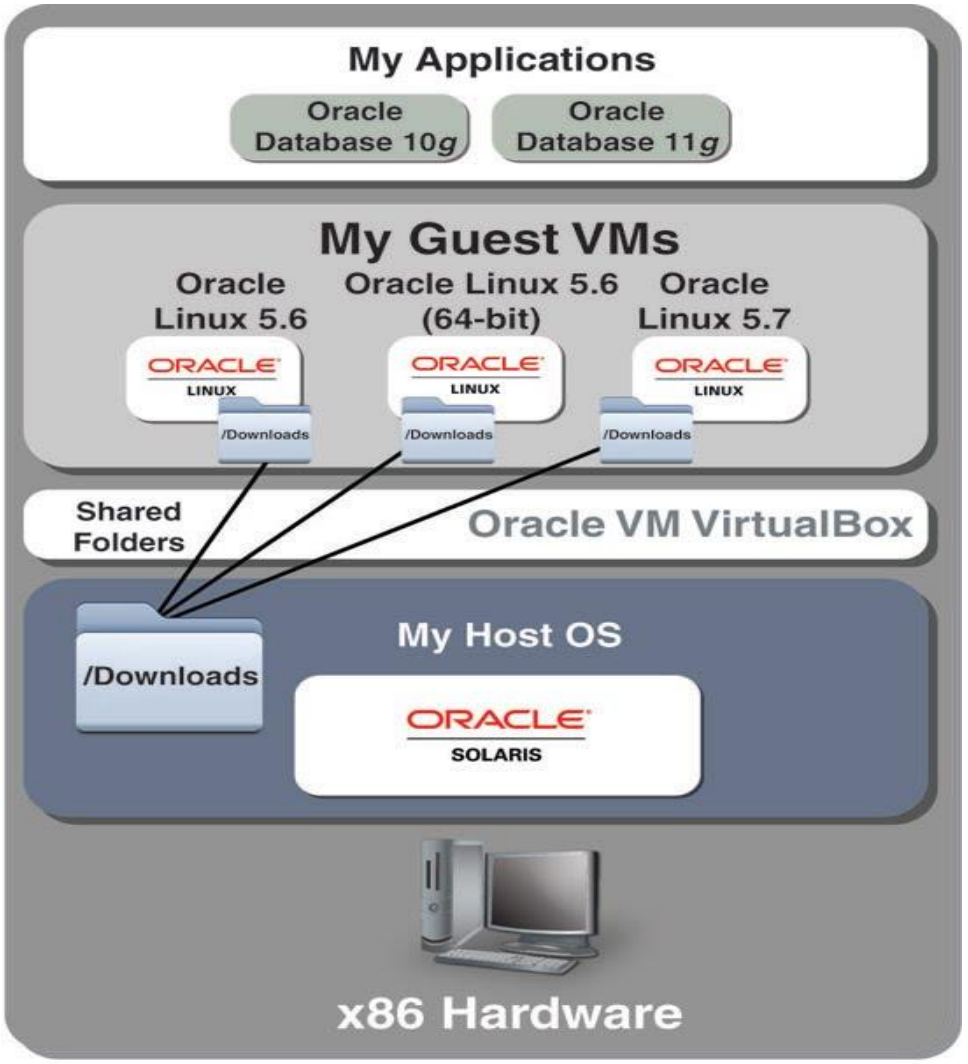
  Ex: VMware ESXi or Microsoft Hyper-V



Type 1 Hypervisor CASE STUDY MICROSOFT HYPER-V



Type 2 Hypervisor



- VMWare example
  - Type 2 hypervisor Upon loading program: scans code for basic blocks
  - If sensitive instructions, replace by VMware procedure
    - Binary translation
  - Cache modified basic block in VMWare cache
    - Execute; load next basic block etc.
- Type 2 hypervisors work without VT (no automatic trapping of privileged instructions by hypervisors) support.

Hardware Virtualization

VMM: Main Modules coordinate to emulate the underlying hardware:-

Dispatcher: Entry Point of VMM

- Reroutes the instructions issued by VM instance.

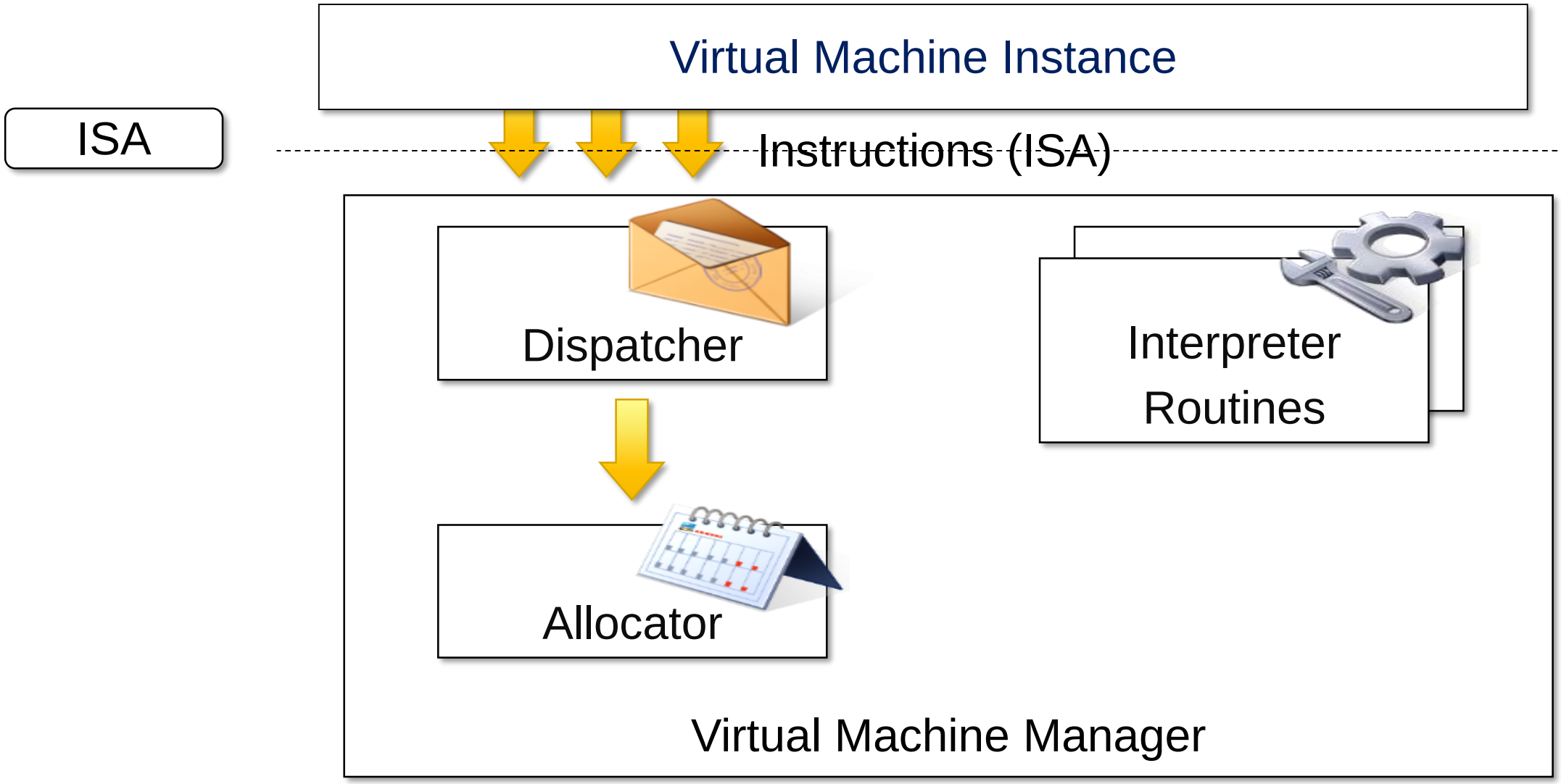
Allocator: Decides system resources to be provided to the VM;

- a VM executes an instruction, results in changing machine resources associated with that VM.
- Invoked by dispatcher

Interpreter: Consists of interpreter routines

- Executed whenever a VM executes a privileged instruction.
- Trap is triggered and the corresponding routine is executed.

VMM


Virtual Machine Instance
ISA
Instructions (ISA)
Dispatcher
Interpreter
Routines
Allocator
Virtual Machine Manager

Hardware Virtualization

Criteria of VMM*:

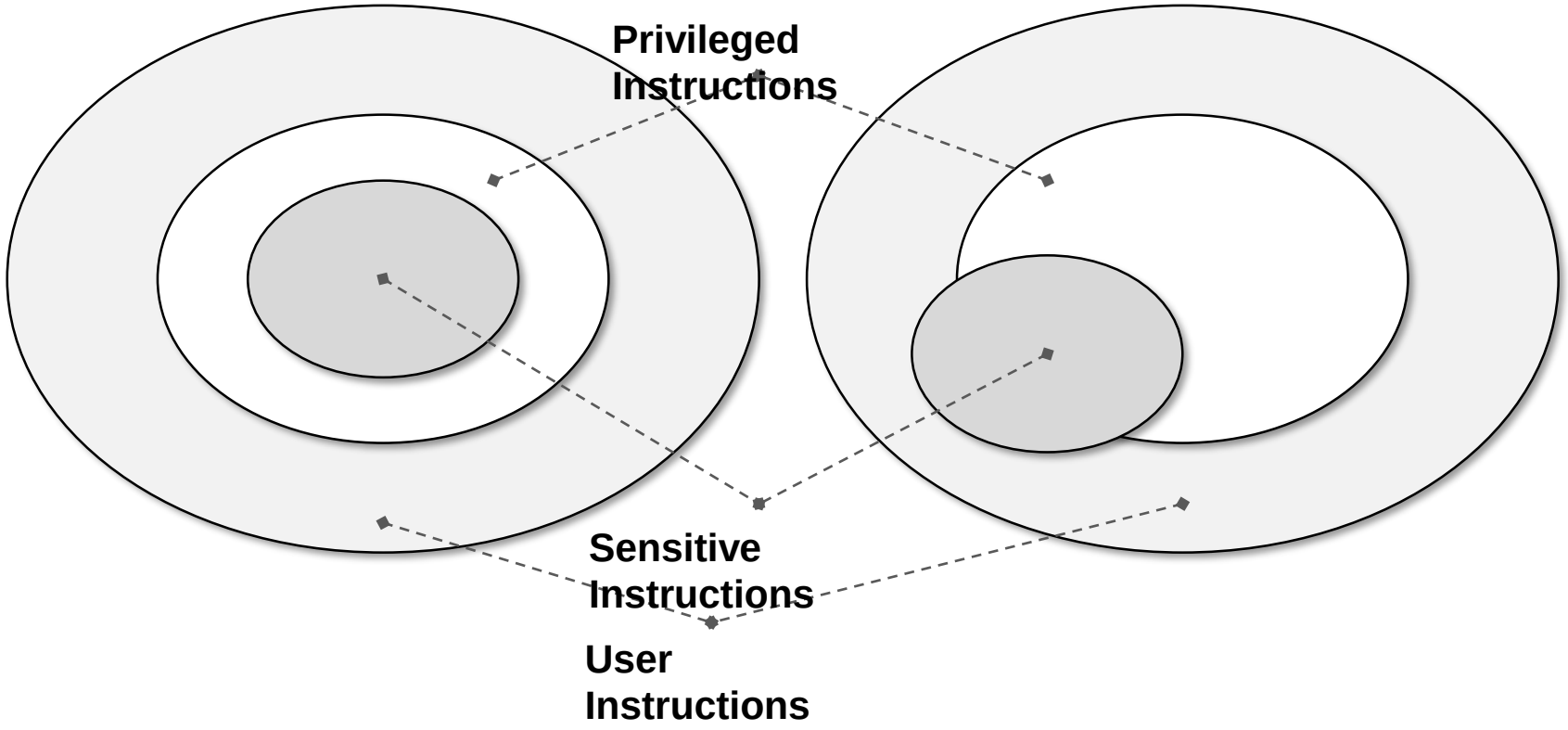
- **Equivalence** – same behavior as when it is ***executed directly*** on the physical host when it was running under control of VMM.
- **Resource control** – VMM should be in ***complete control of virtualized resources***.
- **Efficiency** – a statistically dominant fraction of the machine instructions should be ***executed without intervention*** from the VMM.
- **Theorems*: C*lassification of the instruction set*** and proposed three theorems that define the properties that ***hardware instructions need to satisfy*** in order to efficiently support virtualization.
- Classification of IS:
- Privileged Instructions: trap if the processor is in user mode
- Control sensitive Instructions
- Behavior sensitive Instructions

* *criteria's are established by Goldberg and Popek in 1974*

Hardware Virtualization

Theorem 1*:

- For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.



Hardware Virtualization

Theorem 2 *:

- A conventional third-generation computers is recursively virtualizable if:
- It is virtualizable and
- A VMM without any timing dependencies can be constructed for it.
- **Theorem 3 *:**
- A hybrid VMM may be constructed third-generation machine in which the set of user-sensitive instructions is a subset of the set of privileged instructions.
- *In HVM, more instructions are interpreted rather than being executed directly.*

Hardware Virtualization

Hardware virtualization Techniques:

- CPU installed on the host is only one set, but each VM that runs on the host requires their own CPU.
- CPU needs to be virtualized, done by hypervisor.

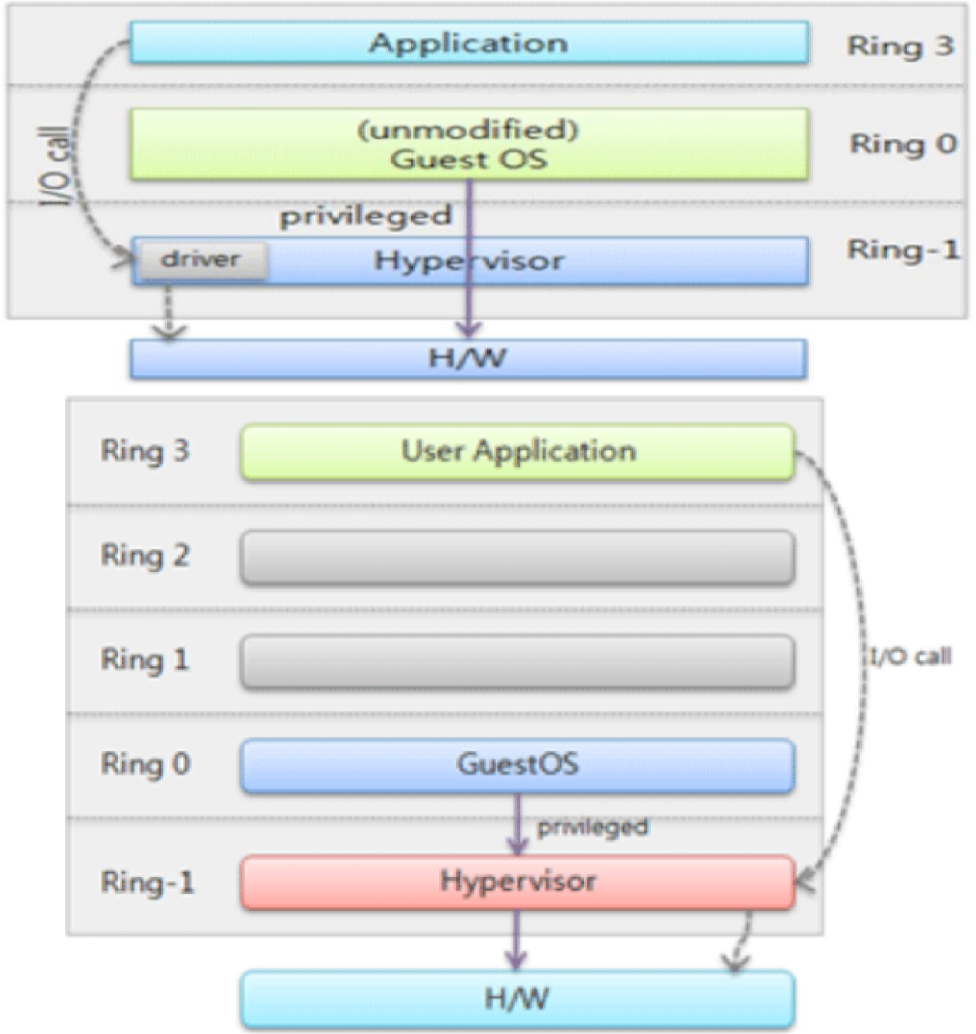
Types of HVT:

- **Full virtualization**
- **Hardware-assisted virtualization**
- **Para virtualization**
- **Partial virtualization**

Hardware Virtualization

Hardware-assisted virtualization:

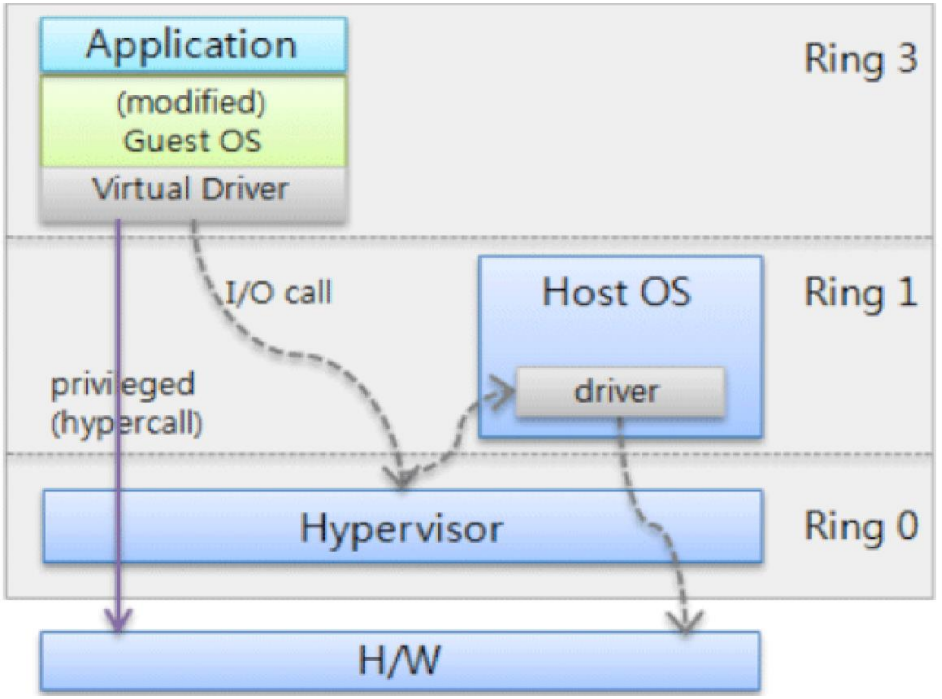
- In this hardware provides architectural support for building a VMM able to run a guest OS in complete isolation.
- ***Intel VT*** and ***AMD V*** extensions.
- Early products were using ***binary translation to trap some sensitive instructions*** and provide an emulated version.
- Additional Ring -1
- ***No binary translation*** of privileged instructions.
- Commands are executed directly to h/w via the hypervisor.



Hardware Virtualization

Para virtualization:

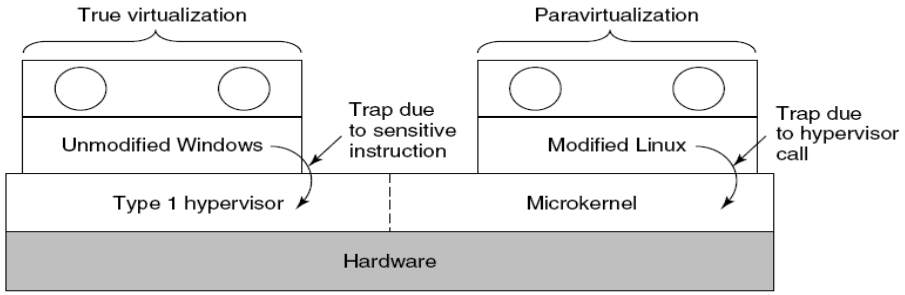
- Not-transparent virtualization
- – Thin VMM
- – Expose software interface to the virtual machine that is slightly modified from the host.
- – Guest OS needs to be modified.
- – Simply transfer the execution of instructions which were hard to virtualized, directly to the host.

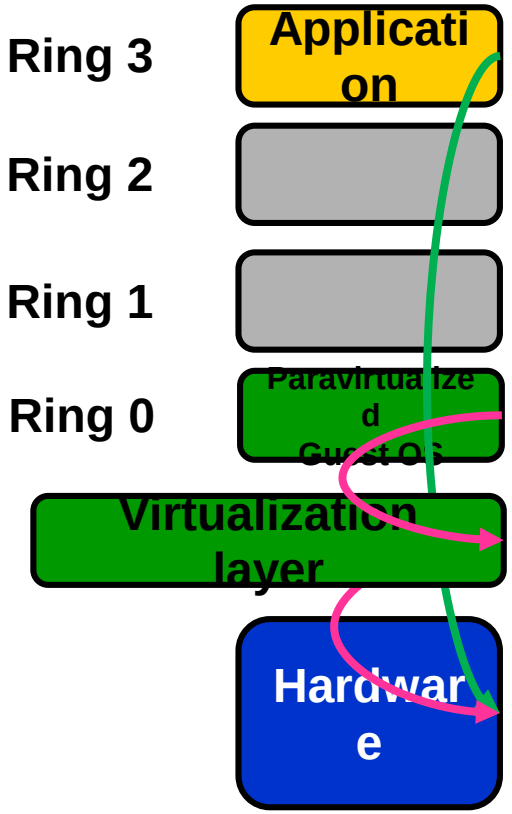


- Privileged instructions of guest ***OS is delivered to the hypervisor*** by using hyper calls
- Hyper calls handles these instructions and accesses the h/w and return the result.
- Guest has authority to ***directly control*** of resources.

Para virtualisation

- Both type 1 and 2 hypervisors work on unmodified OS
- Para virtualisation: modify OS kernel to replace all sensitive instructions with hyper calls
  - OS behaves like a user program making system calls
  - Hypervisor executes the privileged operation invoked by hyper call.





Direct Execution of user and OS Requests

Hyper calls to the Virtualization Layer replace non-virtualiable OS instructions

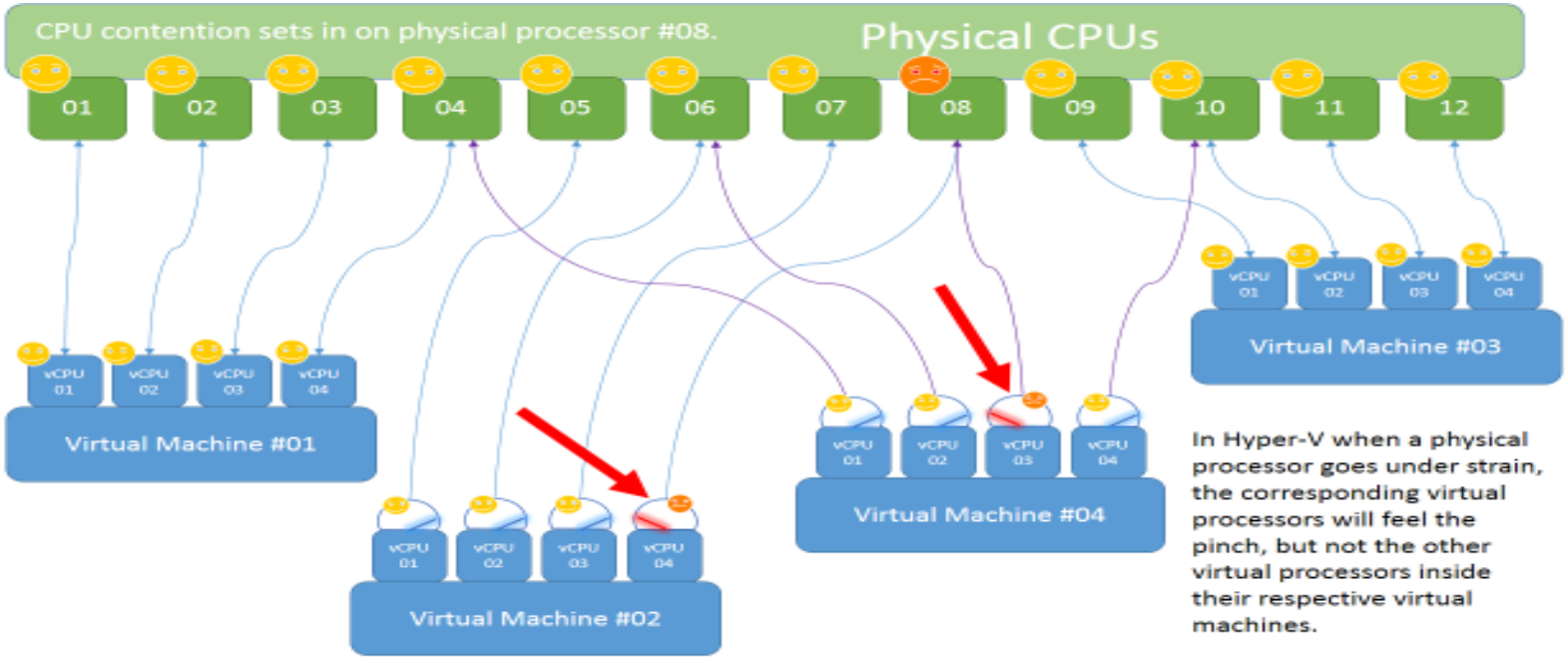
Hardware Virtualization

Partial virtualization:

- Partial emulation of the underlying hardware
- Not allow complete isolation to guest OS.
- Address space virtualization is a common feature of
- Contemporary operating systems.
- Address space virtualization used in time-sharing system.

Virtual CPU (VM) scheduling

- Based on fair-share balancing algorithm
  - Each VM gets equal time on the processor
- The VMs are allocated to physical processor core based on (Availability, Utilization, Priority (Research!!!))
- How privileged instructions requested by VMs depends on
  - Type of virtualization (Full virtualization, Para Virtualization, …)



Operating System-level virtualization

- To create different and ***separated execution environments*** for applications that are managed concurrently.
- No VMM or hypervisor.
- Virtualization is in single OS.
- OS kernel allows for multiple isolated user space instances.
- OS kernel is responsible for sharing the system resources among instances and other management.
- A user space instance has a proper view of the file system (isolated), separate IP addresses, software configurations, and access to devices.
- OS supporting virtualization are general-purpose, time-shared operating systems with the capability to provide resource isolation.

Operating System-level virtualization

- Considered an evolution of the Chroot mechanism in Unix systems.
- Compared to H/W virtualization, no overhead because applications directly use OS system calls and there is no need for emulation.
- No need to modify applications, nor to modify any specific hardware.
- Good for server consolidation; multiple application servers share the same technology: operating system, application server framework etc.
- Different servers are aggregated into one physical server, each server is run in a different user space, completely isolated from the others. Ex: OpenVZ, IBM Logical Partition (LPAR), Solaris Zones and Containers.

Programming language-level virtualization

- To ease deployment of applications, managed execution, Security and portability across different platforms (OSs).
- A virtual machine executes the byte code of a program.
- VMs constitute a simplification of the underlying hardware instruction set and provide some high-level instructions that map some of the features of the languages compiled for them.
- At runtime, byte code can be either interpreted or compiled on the fly (JIT) against underlying hardware instruction set.
- Both Java and the Common Language Infrastructure (CLI), are stack-based virtual machines.
- Stack-based virtual machines possess the property of being easily interpreted and executed simply by lexical analysis and hence are easily portable over different architectures.

Application-level virtualization

- Applications run in runtime environments that do not natively support all the features required by such applications.
- Applications are not installed in the expected runtime environment but are run as though they were.
- Mostly concerned with partial file systems, libraries, and operating system component emulation.
- Such emulation is performed by a thin layer—a program or an operating system component—that is in charge
- Emulation can also be used to execute program binaries compiled for different hardware architectures.

Application-level virtualization

Emulation strategies implemented:

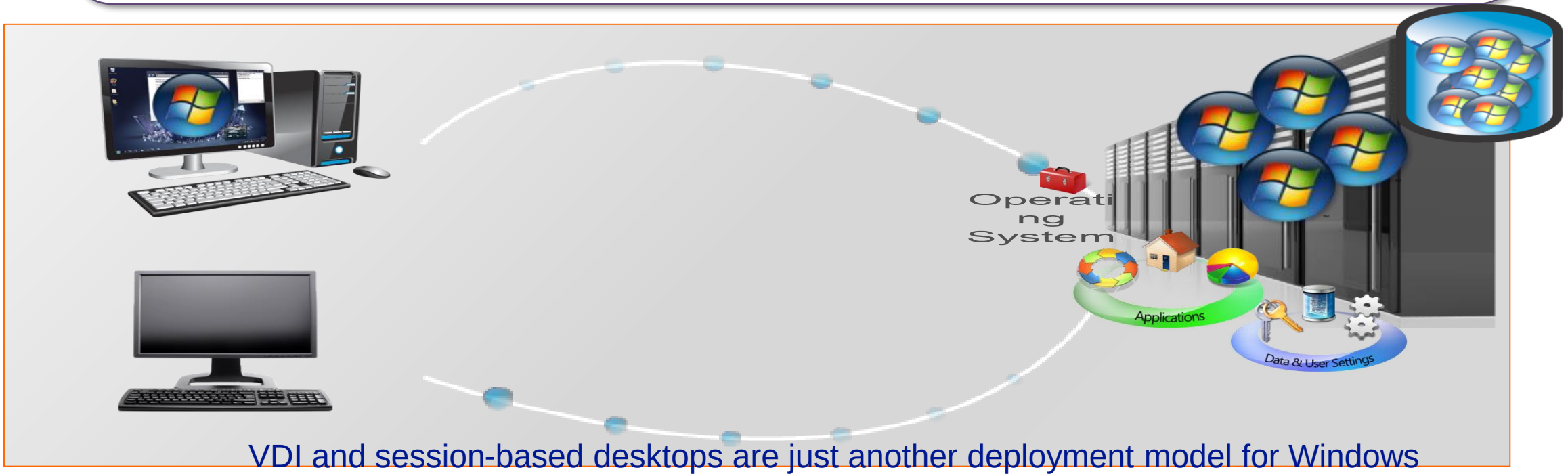
- **Interpretation**: source instruction is ***interpreted*** by an emulator for executing ***native ISA instructions***, ***minimal*** start up cost but ***huge overhead***.
- **Binary translation:** source instruction is ***converted to native*** instructions with equivalent functions.
- Block of instructions ***translated*** , ***cached*** and ***reused***.
- Large ***overhead cost*** , but over time it is subject to ***better performance***.
- In h/w virtualization , it allows the execution of a program ***compiled against a different h/w***.
- In Application level emulation , complete ***h/w environment***.

Other types of virtualization

- **Desktop virtualization** : abstracts the desktop environment available on a personal computer to access it using a client/server approach.
- Same outcome of hardware virtualization but serves a different purpose, makes accessible a different system as though it were natively installed on the host, but this system is remotely stored on a different host and accessed through a network connection.
- Same desktop environment accessible from everywhere.
- A highly available data center ensures the accessibility and persistence of the data, that is required.

What is a Virtual Desktop?

Virtual Desktop Infrastructure (VDI) and Remote Desktop Services session-based desktops are the key technologies that enable virtual desktops, whereby a desktop that runs in the data center can be delivered to the end-user's device using Remote Desktop Protocol (RDP). When combined with technologies that enable application and user state virtualization, organizations can achieve a high degree of desktop optimization and security and reduced TCO.



VDI and session-based desktops are just another deployment model for Windows

Other types of virtualization

- **Storage virtualization**: A system administration practices decoupling the physical organization of the hardware from its logical representation.
- No worried about the specific location of their data, it can be identified using a logical path.
- Harness a wide range of storage facilities and represent them under a single logical file system.
- Storage Area Network (SANs) use a network-accessible device through a large bandwidth connection to provide storage facilities.

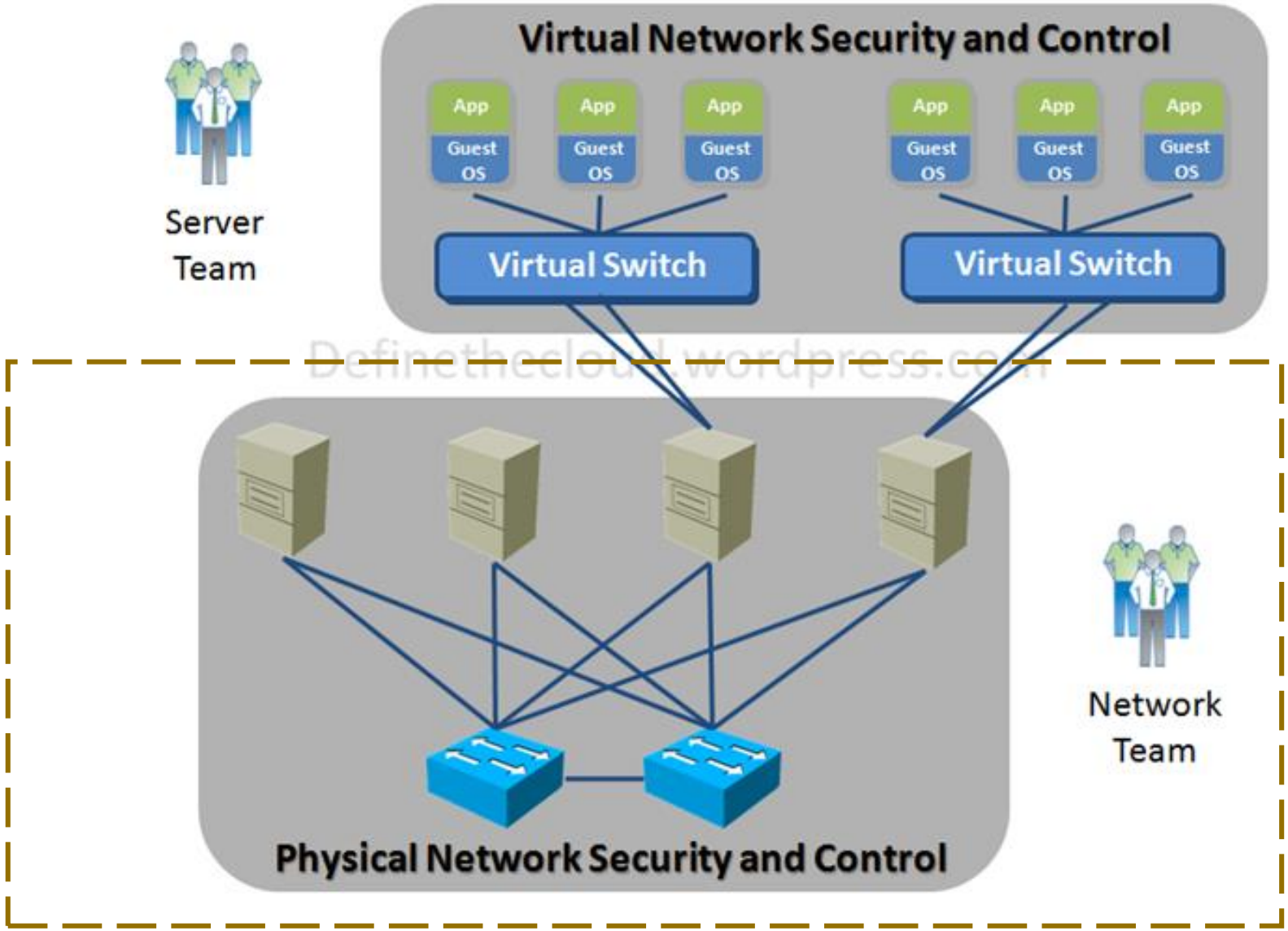
Other types of virtualization

- **Network Virtualization**: combines hardware appliances and specific software for the creation and management of a virtual network.
- Aggregate different physical networks into a single logical network (external network virtualization).
- Provide network-like functionality to an operating system partition (internal network virtualization)
- The result of external network virtualization is generally a virtual LAN.
- A VLAN is an aggregation of hosts that communicate with each other as though they were located under the same broadcasting domain.
- Internal network virtualization is applied together with hardware and operating system-level virtualization, in which the guests obtain a virtual network interface to communicate with.

Datacentre Network Function Virtualization

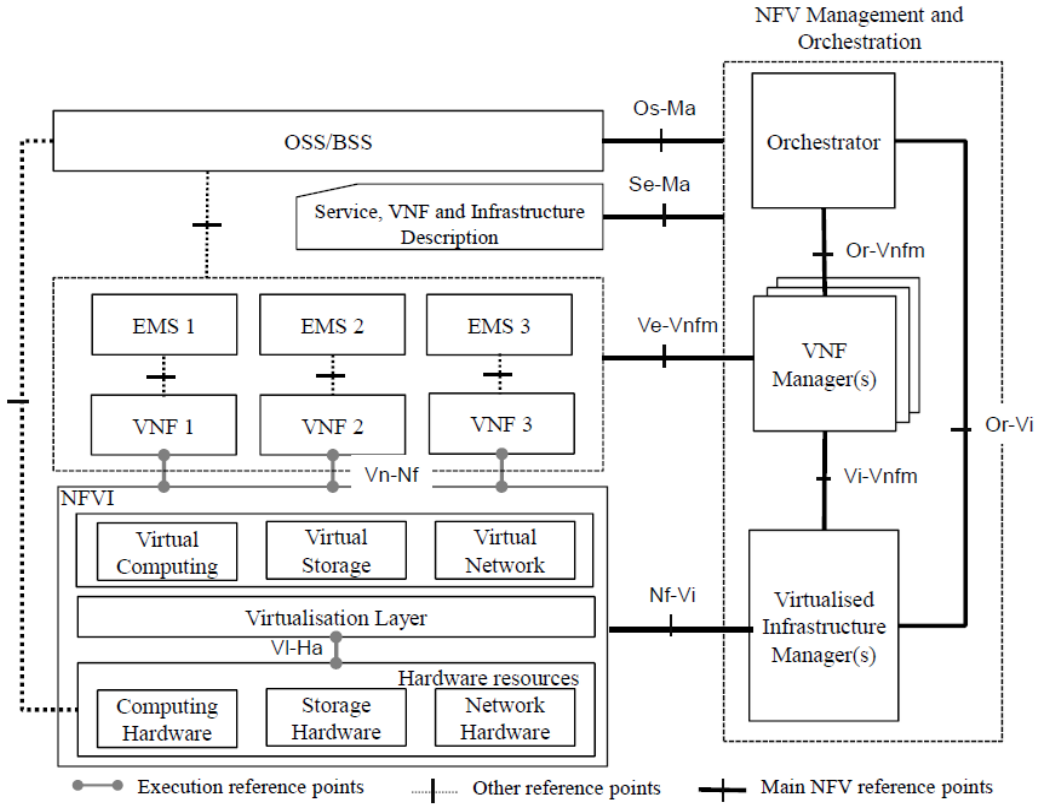
- Network Function Virtualization (NFV) is a technology that leverages virtualization to consolidate the heterogeneous network devices onto industry standard high volume servers, switches and storage.
- Relationship to SDN (Software Defined Networking)
  - NFV is complementary to SDN as NFV can provide the infrastructure on which SDN can run.
  - NFV and SDN are mutually beneficial to each other but not dependent.
  - Network functions can be virtualized without SDN, similarly, SDN can run without NFV.
- NFV comprises of network functions implemented in software that run on virtualized resources in the cloud.
- NFV enables a separation the network functions which are implemented in software from the underlying hardware.

Cloud Datacentre Network



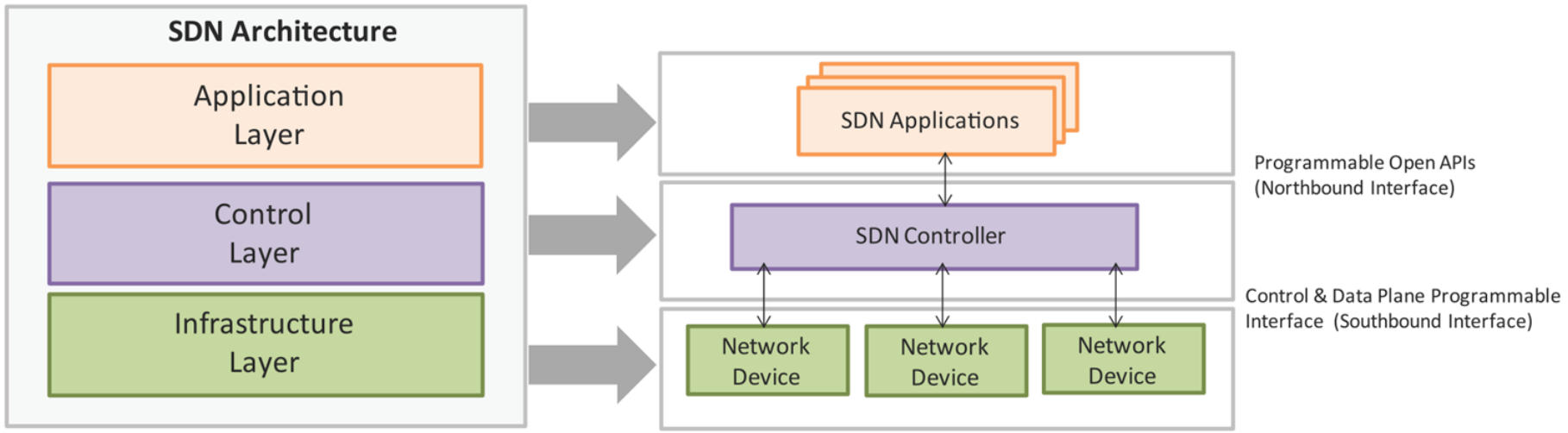
NFV Architecture

- Key elements of the NFV architecture are
  - Virtualized Network Function (VNF): VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).
  - NFV is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.



Software Defined Networking

- Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.
- Conventional network architecture
  - The control plane and data plane are coupled. Control plane is the part of the network that carries the signaling and routing message traffic while the data plane is the part of the network that carries the payload data traffic.
- SDN Architecture
  - The control and data planes are decoupled and the network controller is centralized.

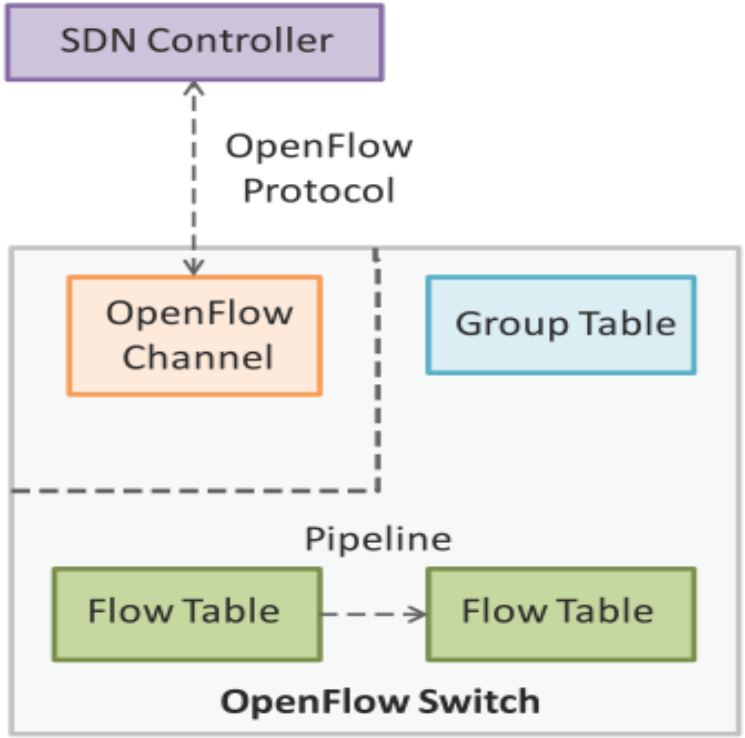


SDN - Key Elements

- Centralized Network Controller
  - With decoupled the control and data planes and centralized network controller, the network administrators can rapidly configure the network.
- Programmable Open APIs
  - SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface). These open APIs that allow implementing various network services such as routing, quality of service (QoS), access control, etc.
- Standard Communication Interface (Open Flow)
  - SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface). Open Flow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface.

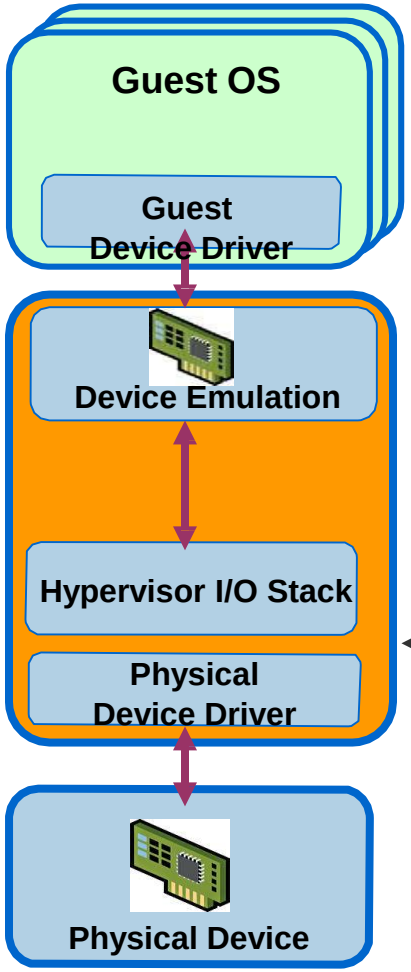
Open Flow

- Open Flow is the broadly accepted SDN protocol for the Southbound interface.
- With Open Flow, the forwarding plane of the network devices can be directly accessed and manipulated.
- Open Flow uses the concept of flows to identify network traffic based on pre-defined match rules.
- Flows can be programmed statically or dynamically by the SDN control software.
- Open Flow protocol is implemented on both sides of the interface between the controller and the network devices.



Open Flow switch comprising of one or more flow tables and a group table, which perform packet lookups and forwarding, and Open Flow channel to an external controller.

Local DISK I/O Virtualization



- Each guest OS thinks it "owns" the disk
- Hypervisor creates "virtual disks"
  - Large empty files on the physical disk that appear as "disks" to the guest OS
    - Hypervisor converts block # to file offset for I/O
  - Direct Memory Access (DMA) needs physical addresses
    - Hypervisor needs to translate

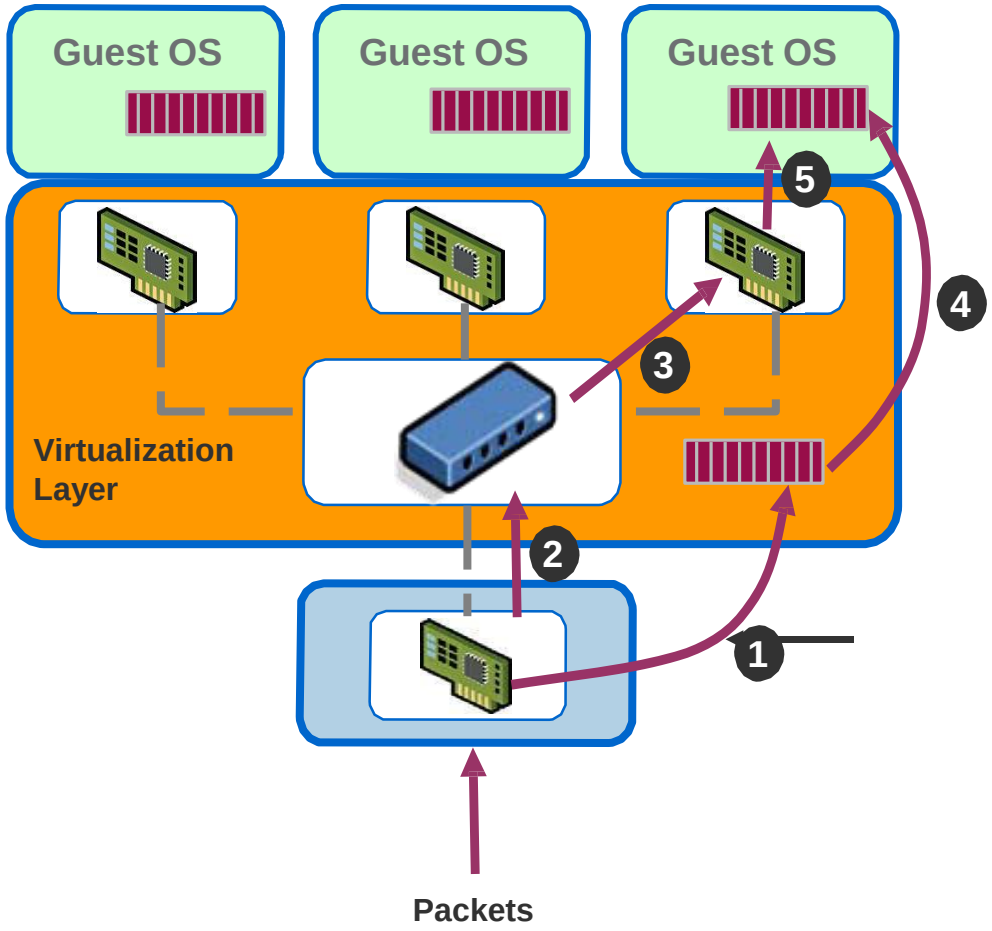
Hypervisor I/O stack

translates guest I/O addresses to host addresses handles inter VM communication

multiplexes I/O requests from/to the physical device

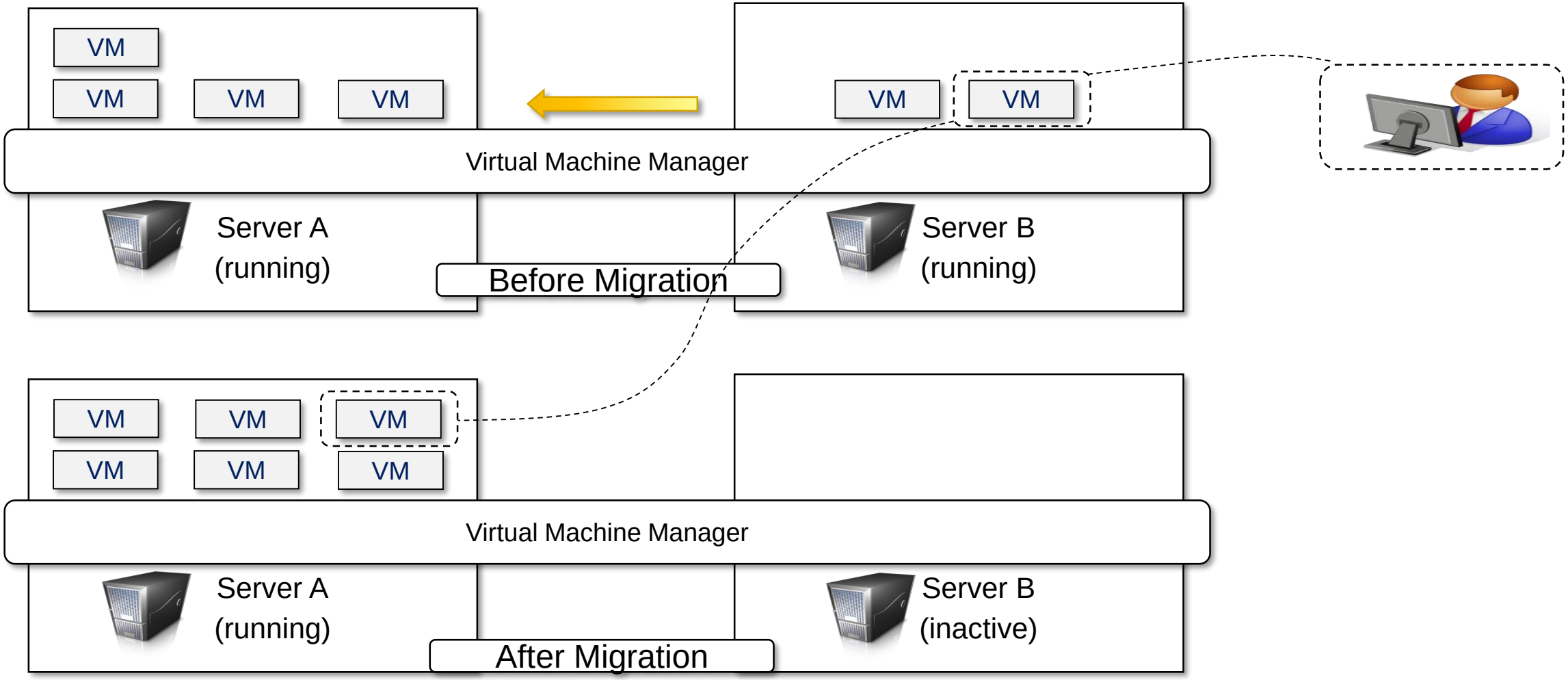
provides enterprise-class I/O features

Local Network Function (I/O) Virtualization



1. DMA packet into receive buffers owned by VM kernel
2. Raise physical Interrupt
3. Parse packet to find destination VM
4. Copy packet into guest receive queue
5. Raise virtual NIC interrupt

Virtualization and Cloud Computing (VM Migration)



Virtualization and Cloud Computing

- Degree of **customization, security, isolation, and manageability.**
- Virtual computing environment
- Hardware virtualization is Infrastructure-as-a-Service (IaaS).
- Programming language virtualization is Platform-as-a-Service (PaaS) offerings.
- Attractive business opportunity for companies featuring a large computing infrastructure.
- Isolated and controllable environments
- **Server consolidation:** underutilized active resources reduced by aggregating VMs to small number of resources, can be fully utilized**.**
- Movement of virtual machine instances (VM **migration).**

IntelliPaat

eict IIT Roorkee

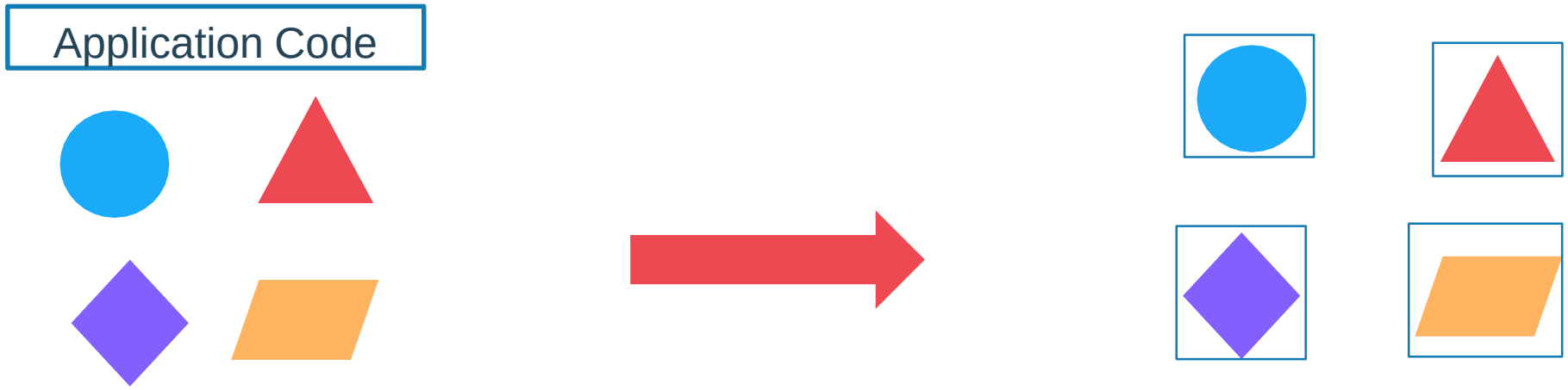
IIT Roorkee

Containerization and Docker

Movement in the cloud

- Migrate workloads to cloud
- Portability across environments
- Want to avoid cloud vendor lock-in

Application Modernization



Developer Issues:

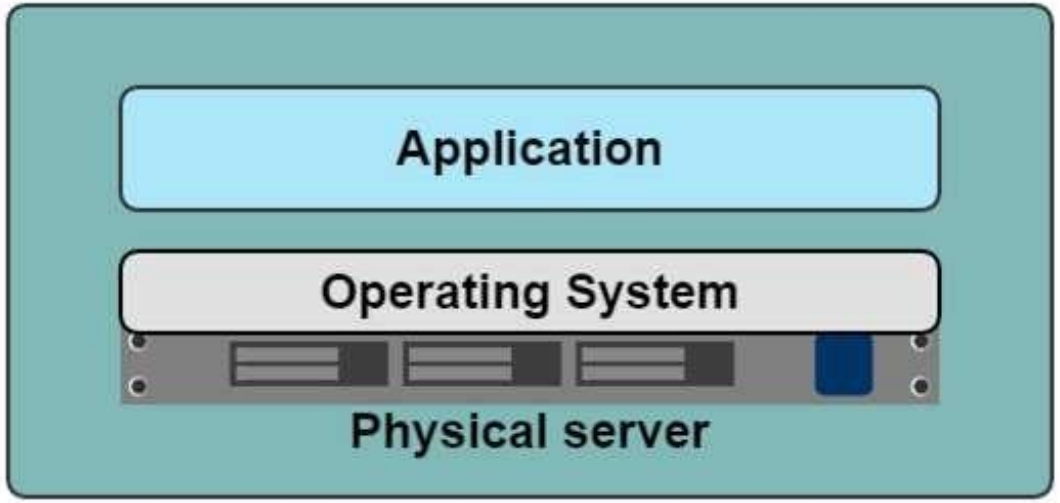
- Minor code changes require full re-compile and re-test
- Application becomes single point of failure
- Application is difficult to scale

Micro services: Break application into separate operations

12-Factor Apps: Make the app independently scalable, stateless, highly available by design

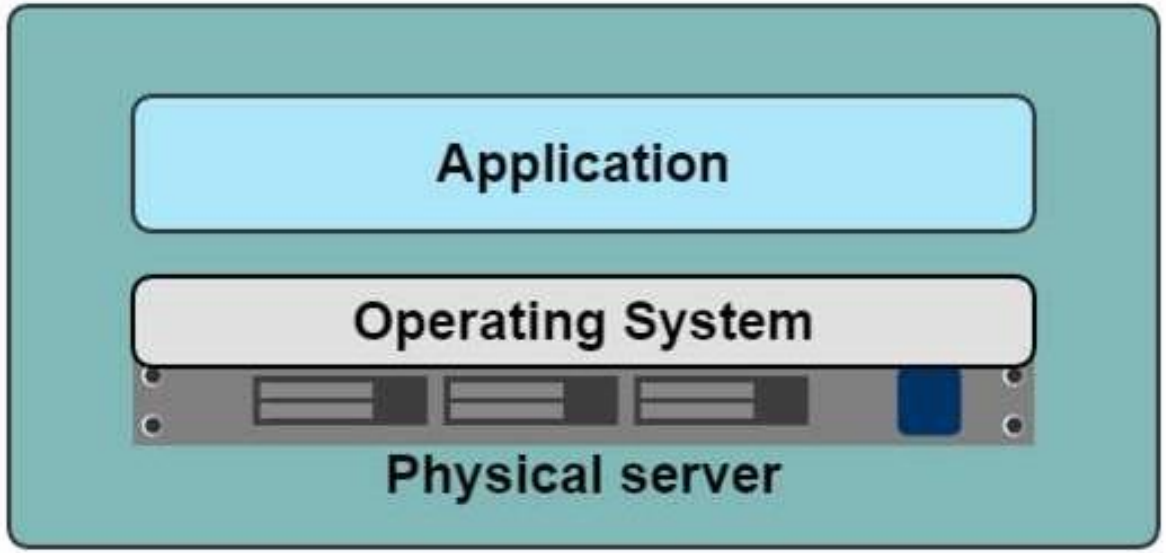
A History Lesson

One application on one physical server



Historical limitations of application deployment

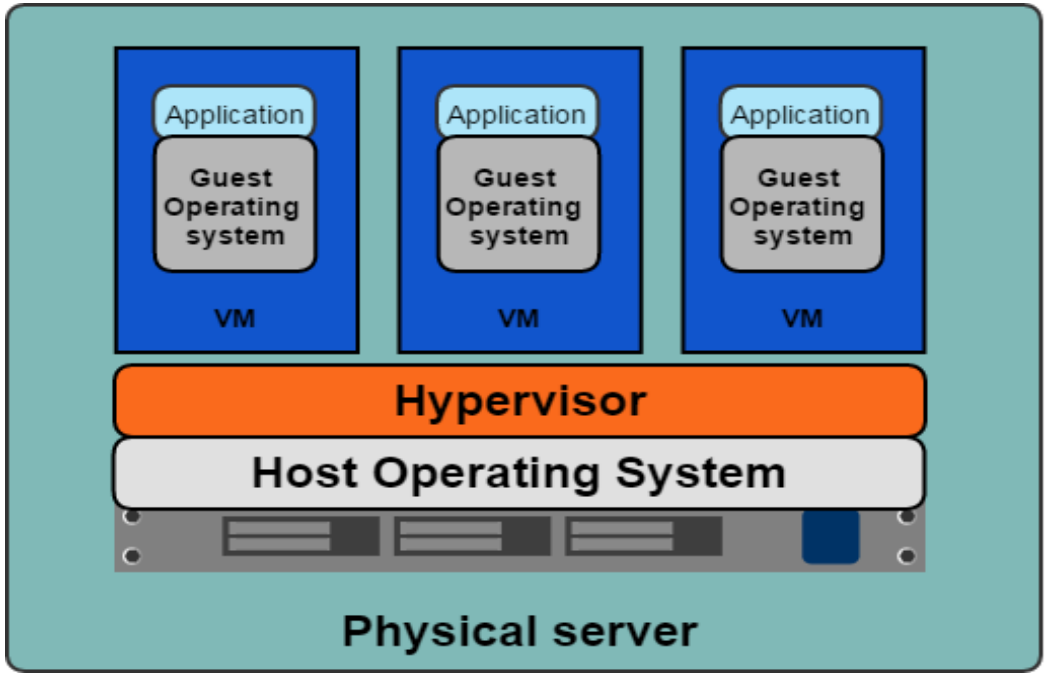
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



A History Lesson

Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



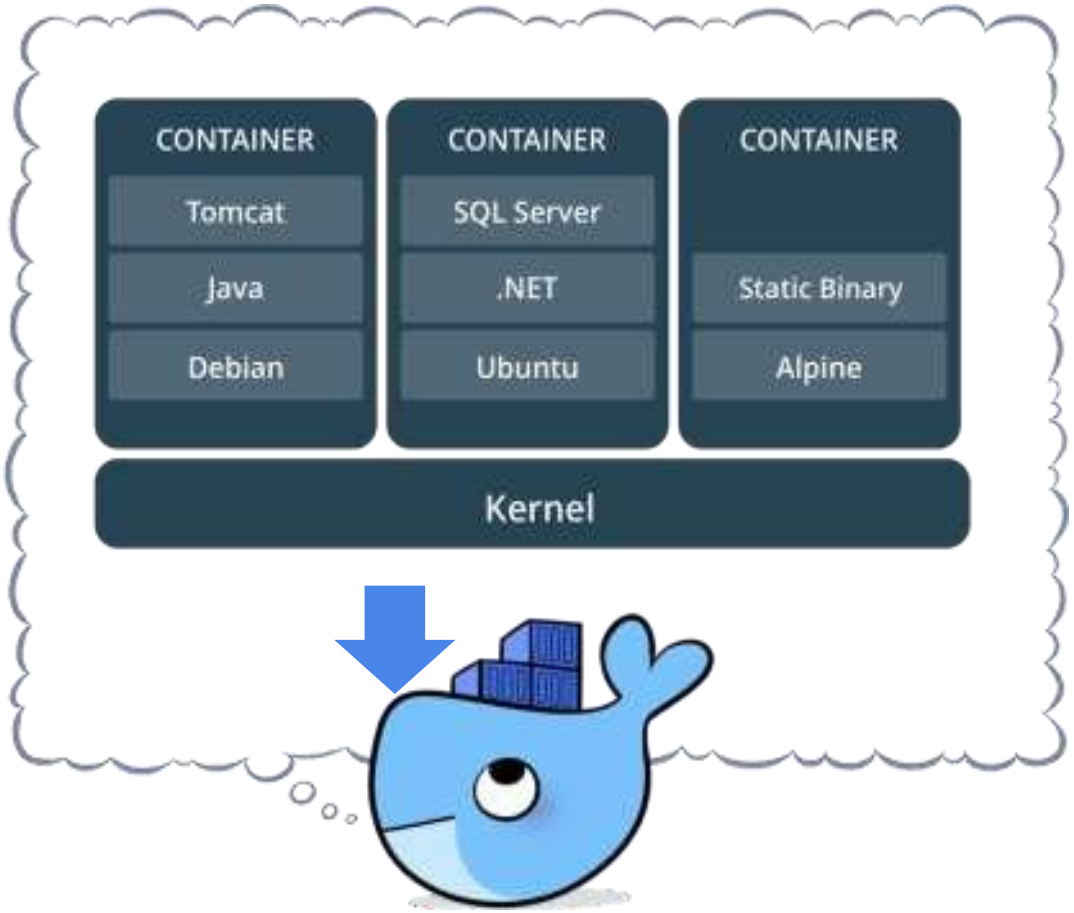
Benefits of VMs

- Better resource pooling
  - One physical machine divided into multiple virtual machines
- Easier to scale
- VMs in the cloud
  - Rapid elasticity
  - Pay as you go model

Limitations of VMs

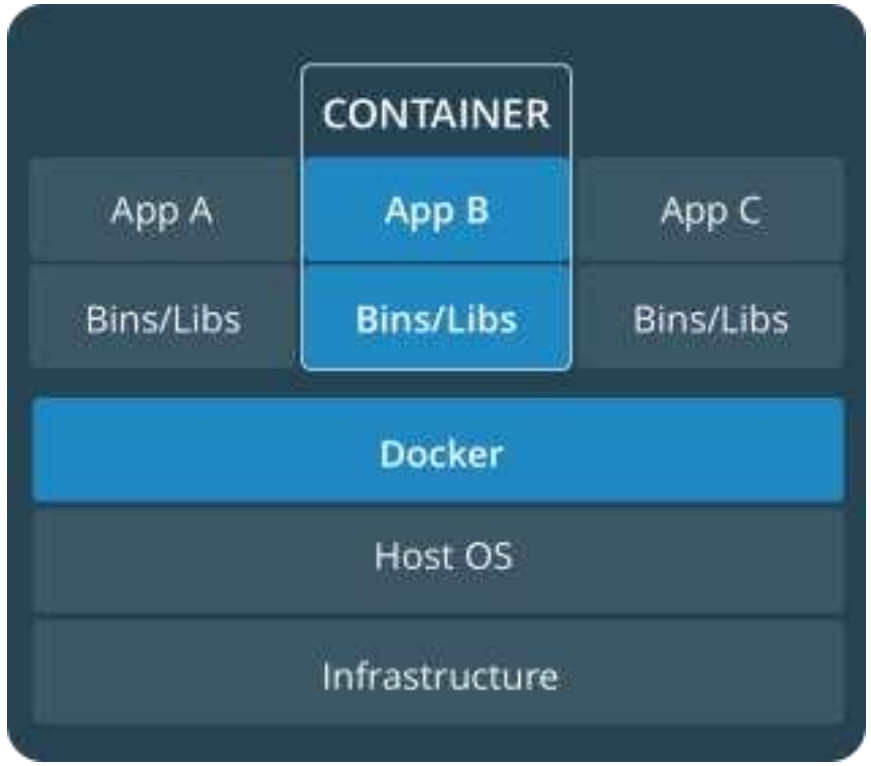
- Each VM stills requires
  - CPU allocation
  - Storage
  - RAM
  - An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed

What is a container?

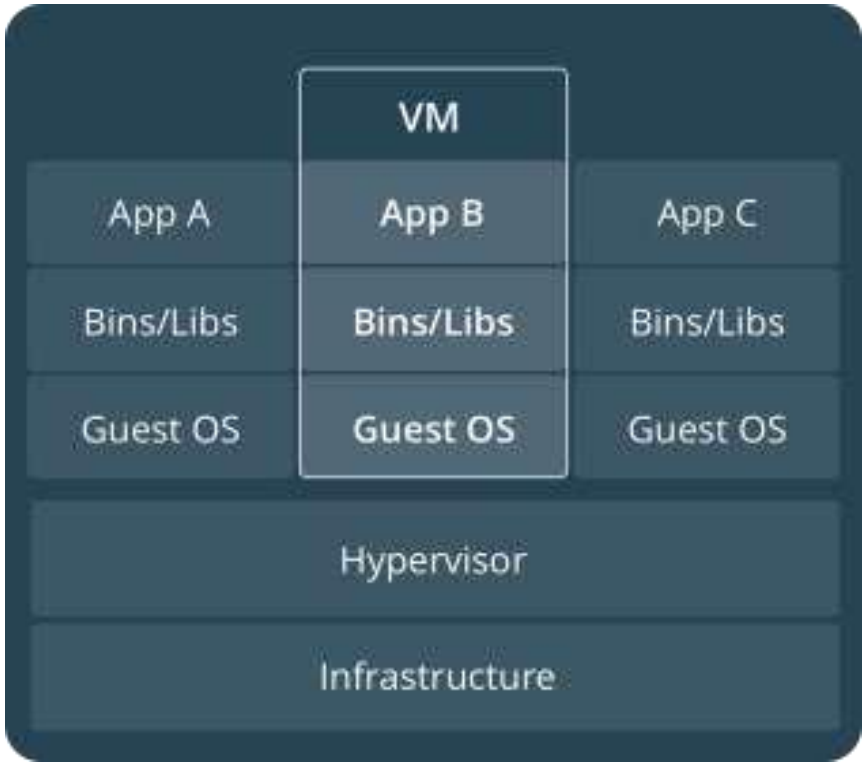


- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

Comparing Containers and VMs

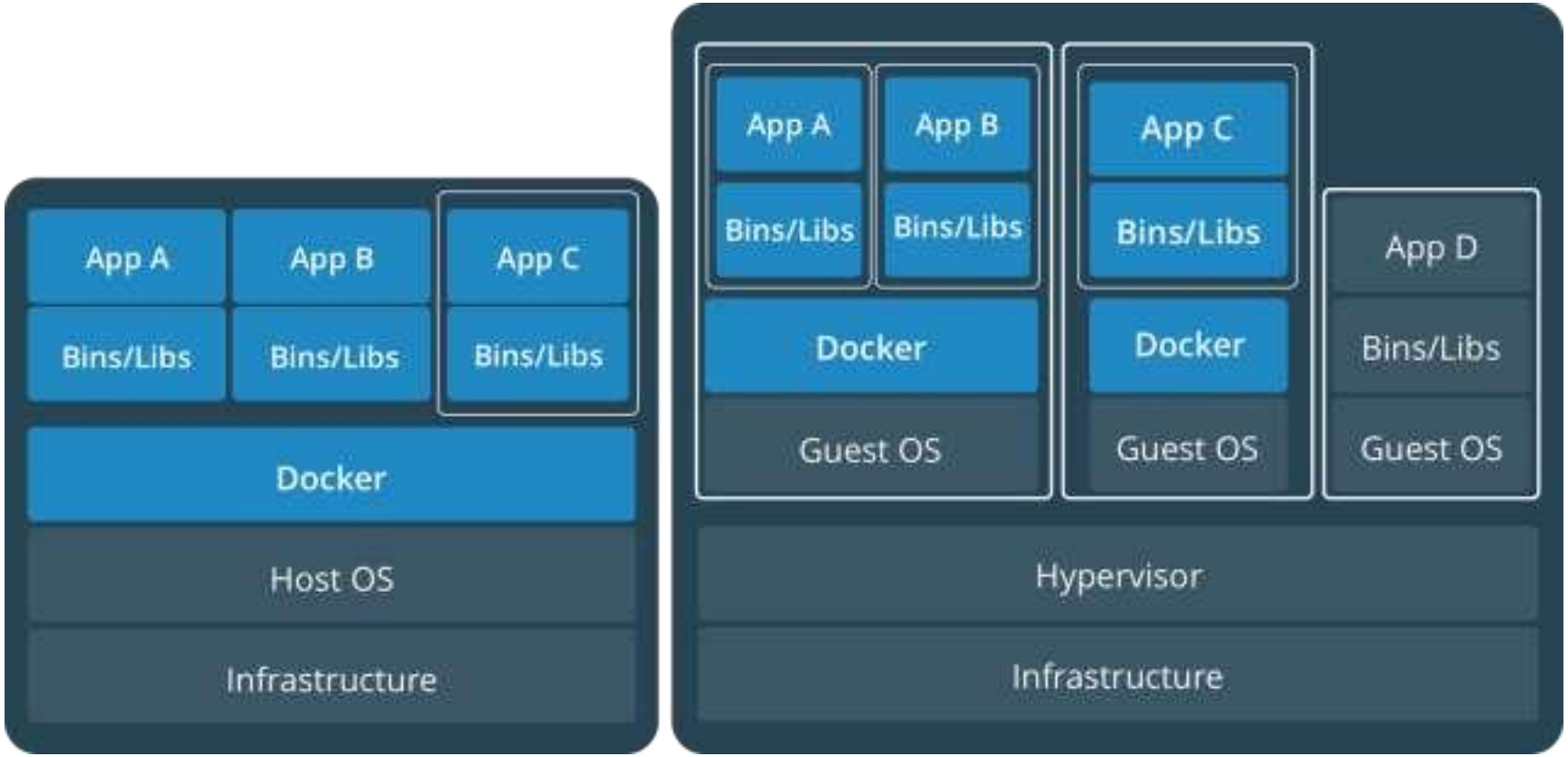


Containers are an app level construct



VMs are an infrastructure level construct to turn one machine into many servers

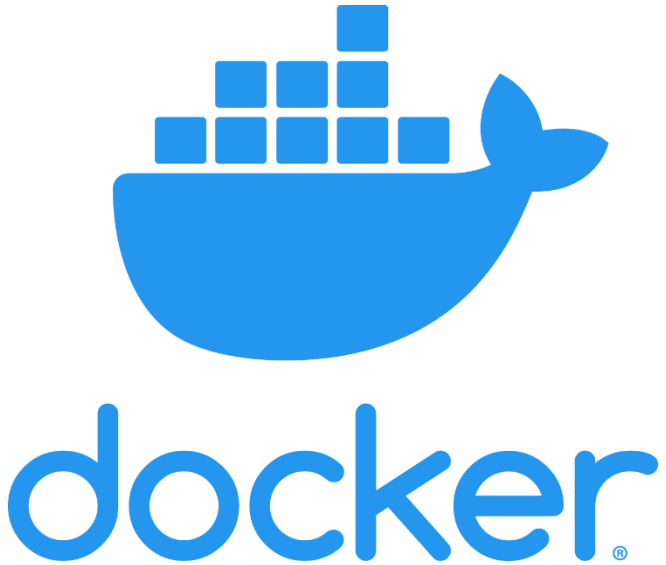
Containers and VMs together



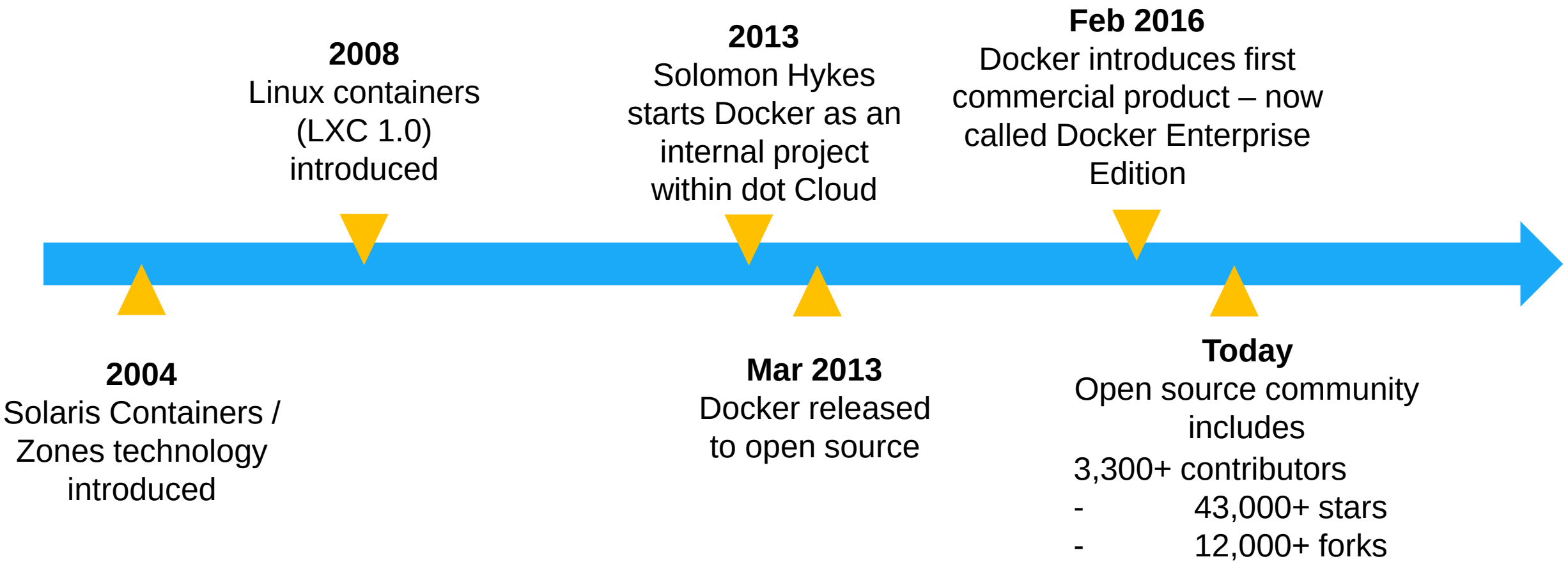
Containers and VMs together provide a tremendous amount of flexibility for IT to optimally deploy and manage apps.

What is Docker?

Docker is a platform for developing, shipping & running application using container based virtualization technology.



History of Docker



Incredible adoption in just 4 years

Key Benefits of Docker Containers

Speed

- No OS to boot = applications online in seconds

Portability

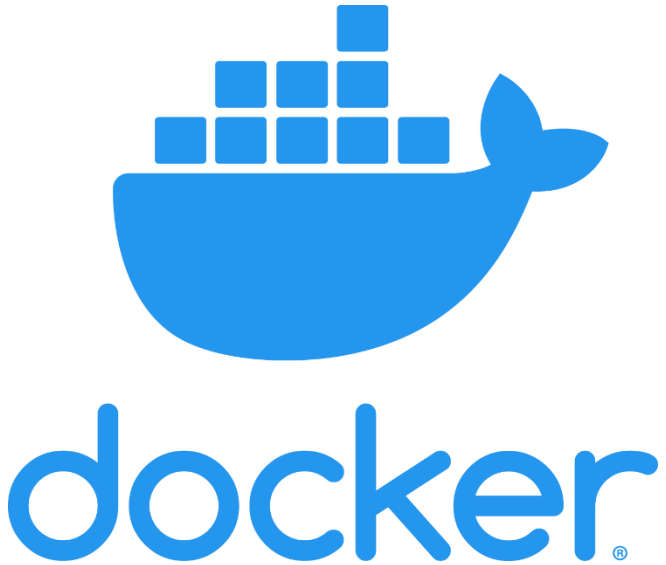
- Less dependencies between process layers = ability to move between infrastructure

Efficiency

- Less OS overhead
- Improved VM density

Docker Components: Docker Image

A Docker image is a binary that includes all of the requirements for running a single Docker container, as well as metadata describing its needs and capabilities.



Docker Components: Docker File

- A Docker file is a text document that contains all the commands a user could call on the command line to assemble an image. Using Docker build users can create an automated build that executes several command-line instructions in succession.

- Docker file is used for automation of work by specifying

all step that we want on docker image

Docker Components: Docker Container

- Images are read only containers used to create containers.
- Built by you or other Docker Users.
- Stored in Docker hub or your local repository

Docker Components: Docker Engine

- Docker Engine is the program that enables containers to build shipped & run.
- Docker Engine uses Linux Kernel namespace & control group.
- Namespace gives us the isolated workspace.

Installing Docker & Running Hello World

Install docker with command

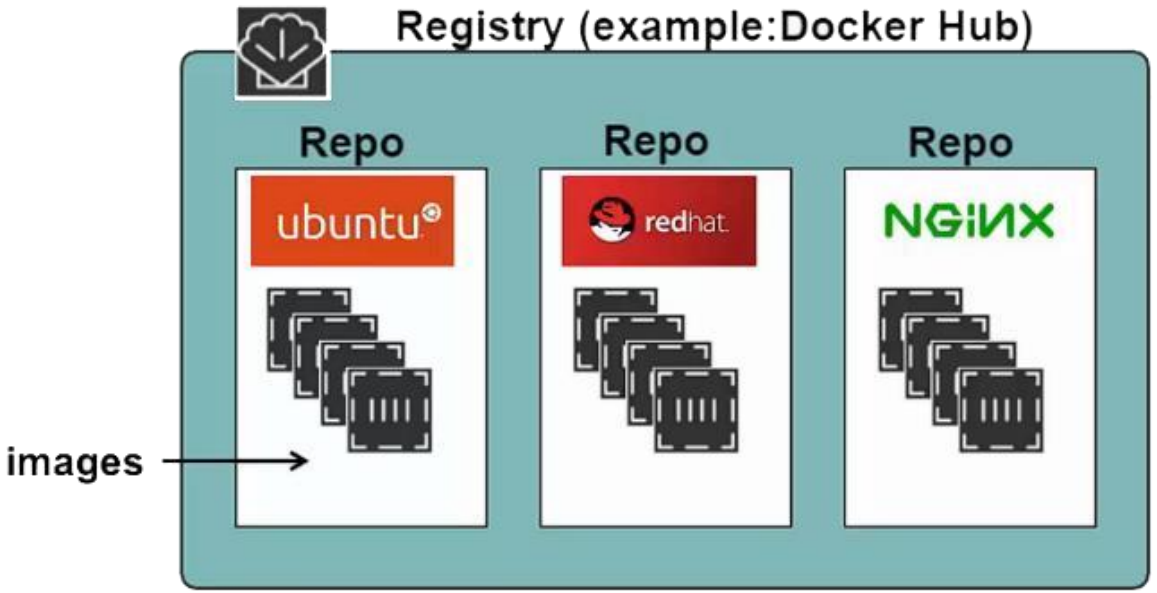
```
# curl -sSL https://get.docker.com/ | sh
```

Run the hello world container to test your installation #

```
sudo docker run hello-world
```

Docker Components :Registry / Repository

- Where we store our image is known as registry
- You can use your own registry or docker's public registry. Known as Docker Hub



Docker Components: Docker Hub

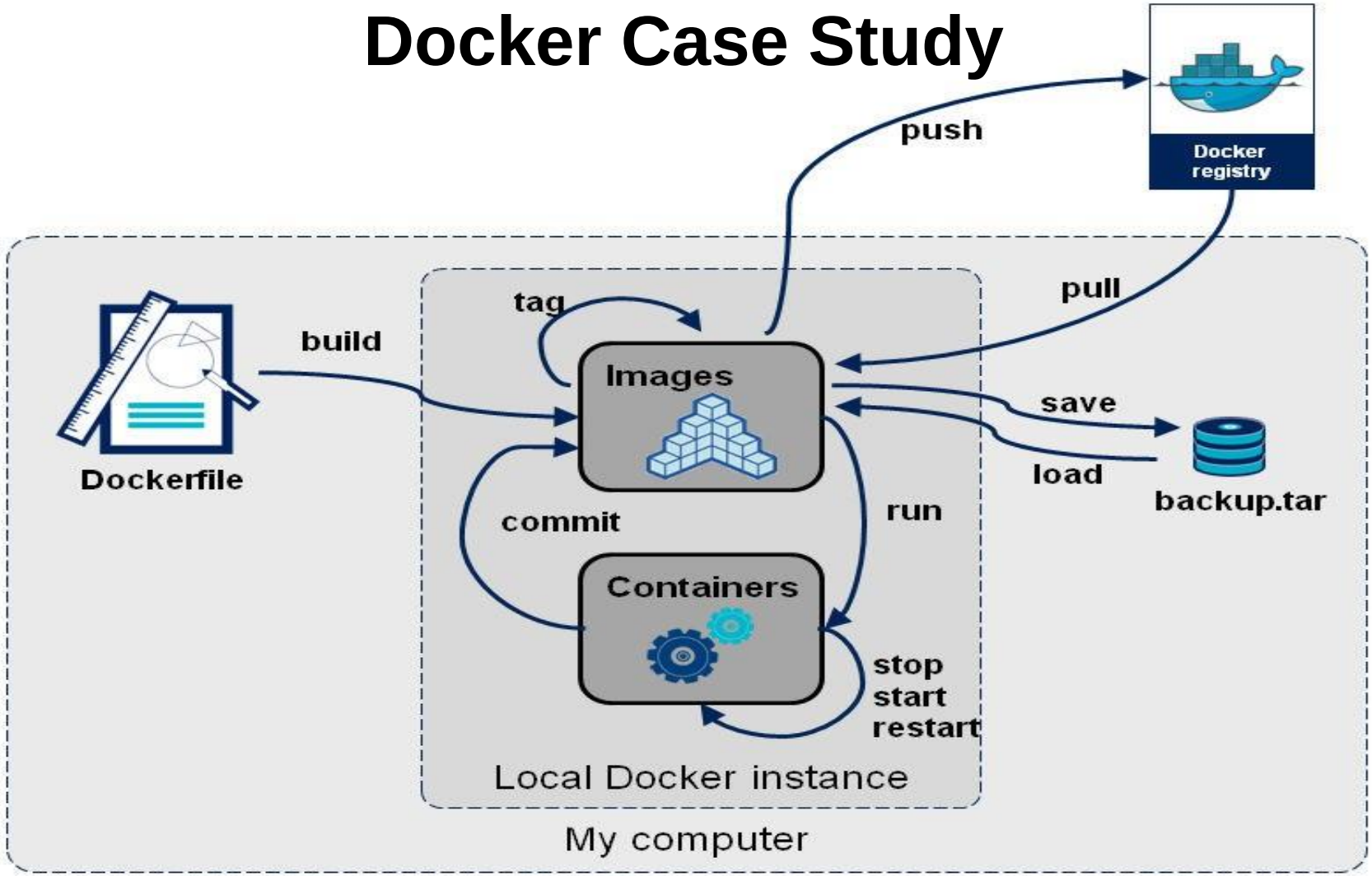
- Docker hub is the public registry that contains large amount of images available for your use.
- Official Repositories are available at https://hub.docker.com/explore/

Docker Components: Intro to Images

1. Go to https://hub.docker.com/ and sign up for an account.
2. Find your confirmation email and active your account.
3. Explore images from docker hub.
4. Understanding official Images, Tags
5. Search images on docker hub.

Deployment Lifecycle Container Image and instance

Docker Case Study



Benefits of using Docker

- Separation of Concerns

  Life becomes easier for System admin

- Fast deployment cycle
- Application portability

  Build in one environment, Ship anywhere.

- Scalability

  Easy sign up new containers if needed.

- Run more apps on host machine

Docker Networking

- Containers can talk to each other without having to expose ports to host.
- Essential for micro service application architecture.
- Example:

  -Container with Tomcat running

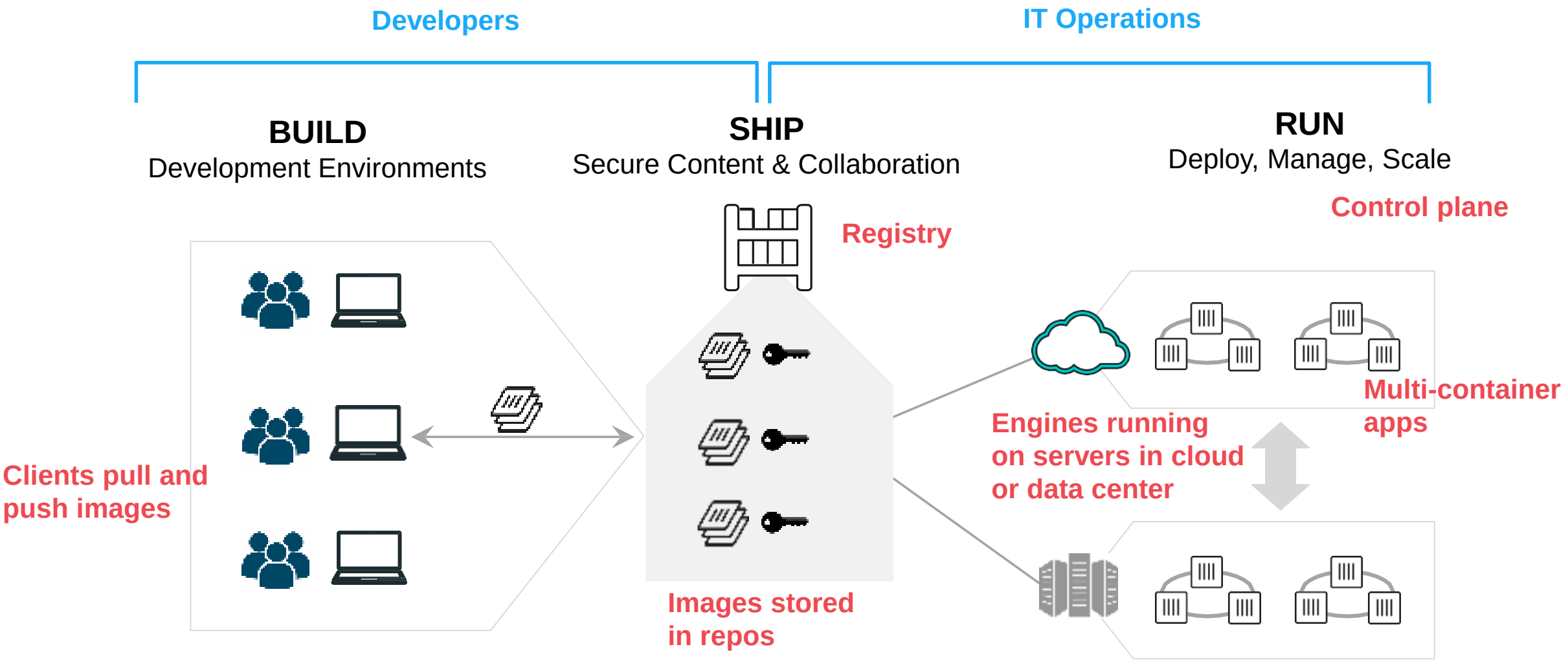
  -Container with MySQL running

  -Application on tomcat needs to connect to MySQL

Container vs VM

- Containers are more lightweight.
- No need to install guest OS.
- Less CPU, RAM, Storage needed.
- More containers per machine than VM.

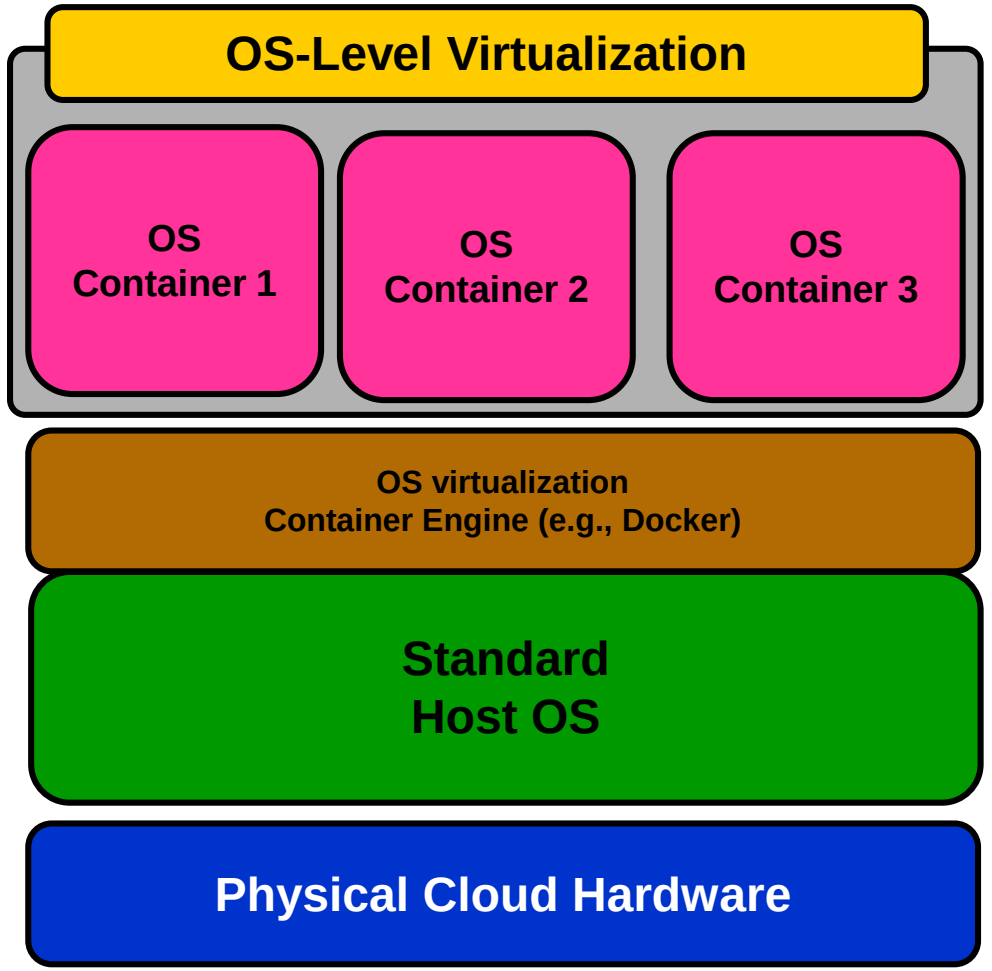
Containers as a Service



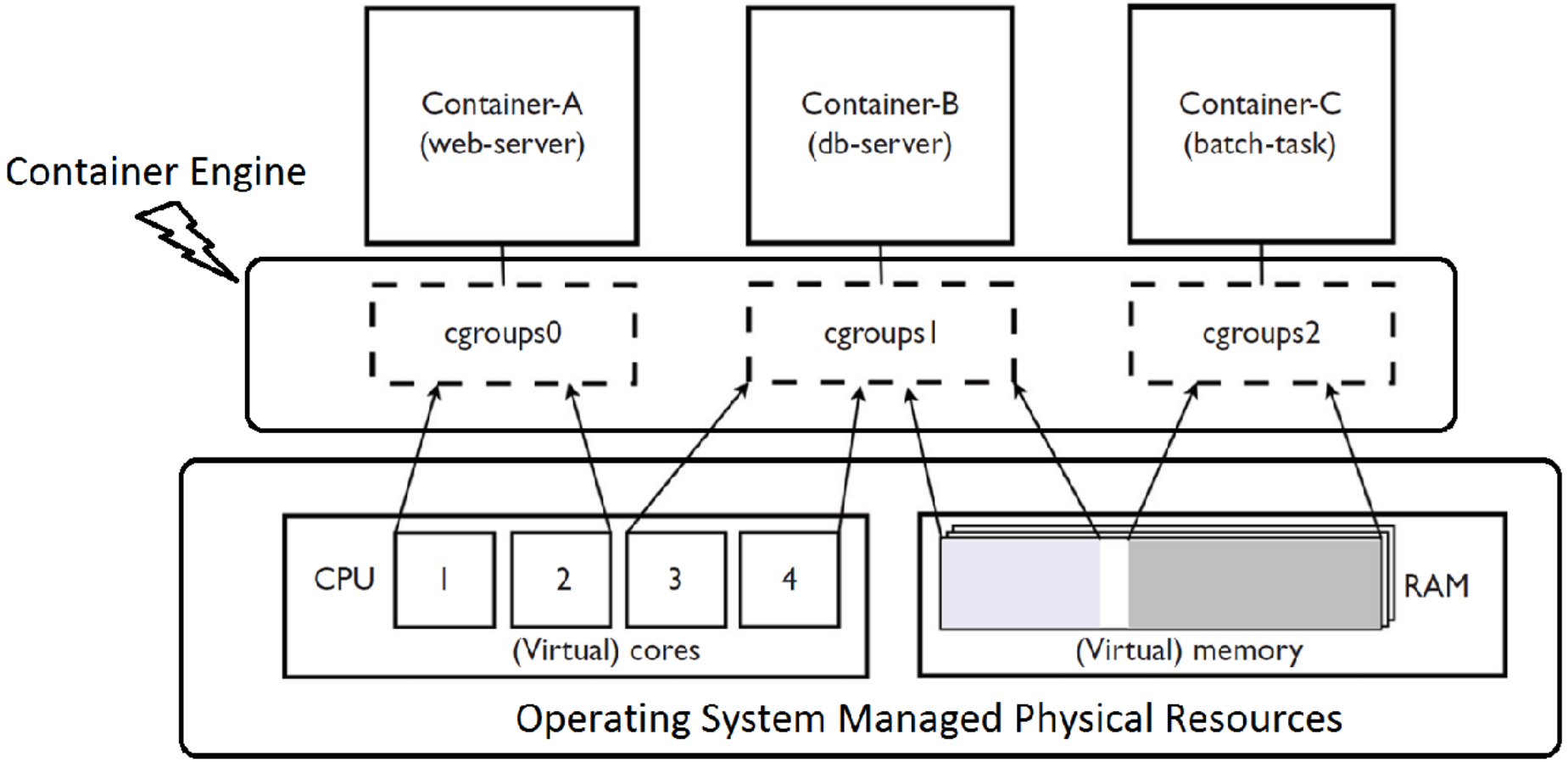
Docker is in the Enterprise

OS-level virtualisation (Containerisati

- Containers are light-weight alternatives to hypervisor-based VMs
- Container is a collection of operating system kernel utilities (bins/runtime libs) configured to manage the physical hardware resources used by a particular micro service (user process or app)
- Popular container engine implementation includes Docker, and Open VZ



Resource sharing (CPU, Disk, I/O) across containers



Difference between VM Image and VM instance

- A virtual Machine Image (VMI) is a file (e.g., XML) that describes the software configurations of a virtual machine.
- From an VMI, you launch an instance, which is a copy of the VMI running as a virtual server in the cloud. An instance type has hardware configurations.

Table 1: VMI Configurations

Name	Example Values
Virtualization Format	Xen, VMWare, ...
Operating System (OS)	Linux, Windows, ...
OS Version	Ubuntu 10.4, ...
Software Feature	Application Server, Load Balancer, Database, ...
Software	JBoss, Nginx, MySQL, ...
Software Vers.	0.8-alpha,...
Implementation Lang.	Java, Perl, Ruby, ...
Supported Impl. Lang.s	Java, Perl, Ruby, ...

Table 2: Numerical Instance Configurations

Name	Influence	Metric	Range
Hourly Price	Negative	$/h	0-∞ $/h
CPU Cores	Positive	Cores	0-∞
RAM Size	Positive	Bit	0-∞
Disk Size	Positive	Bit	0-∞
CPU Perfomance	Positive	Flops	0-∞ Flops
RAM Perfomance	Positive	Ops/s	0-∞ Ops/s
Disk Perfomance	Positive	B/s	0-∞ B/s
Max. Latency	Negative	ms	0-∞ ms
Avg. Latency	Negative	ms	0-∞ ms
Uptime	Positive	%	0-100%
Service Popularity	Positive	%	0-100%
Network Send Price	Negative	$/Byte	0-∞
Network Recieve Price	Negative	$/Byte	0-∞
Internet Send Price	Negative	$/Byte	0-∞
Internet Recieve Price	Negative	$/Byte	0-∞

Difference between Container Image and instance

- A Container Image is a file (e.g., JavaScript Object Notation) that describes the software configurations and is a snapshot (e.g., run-time libs, dependencies, OS type) of a container.
- From Container Image, you launch Container Instance, which is a copy of the image running as a container in the cloud or on your local machine.

Table 1: Container Image Configurations

Name	Example Values
Container Providera	Google, AWS,IBM, Deis, Docker Hub ...
Container Format	Bare, OVF, AKI, ARI, AMI, Docker ...
Disk Format	Raw, VHD, VMDK, VDI ...
Operating System (OS)	Linux, Windows, ...
OS Version	Ubuntu 10.4, ...
Software Feature	Web Server, Load Balancer, DB Server ...
Software	Apache HTTP, JBoss, Nginx, MySQL ...
Software Vers.	0.8-alpha,..
Implementation Language	Java, Perl, Ruby, ...
Supported Impl. Lang.s	Java, Perl, Ruby, ...

VM Image vs. Container Image

Table 1: VMI Configurations

Name	Example Values
Virtualization Format	Xen, VMWare, ...
Operating System (OS)	Linux, Windows, ...
OS Version	Ubuntu 10.4, ...
Software Feature	Application Server, Load Balancer, Database, ...
Software	JBoss, Nginx, MySQL, ...
Software Vers.	0.8-alpha,...
Implementation Lang.	Java, Perl, Ruby, ...
Supported Impl. Lang.s	Java, Perl, Ruby, ...

Table 2: Container Image Configurations

Name	Example Values
Container Providera	Google, AWS,IBM, Deis, Docker Hub ...
Container Format	Bare, OVF, AKI, ARI, AMI, Docker ...
Disk Format	Raw, VHD, VMDK, VDI ...
Operating System (OS)	Linux, Windows, ...
OS Version	Ubuntu 10.4, ...
Software Feature	Web Server, Load Balancer, DB Server ...
Software	Apache HTTP, JBoss, Nginx, MySQL ...
Software Vers.	0.8-alpha,...
Implementation Language	Java, Perl, Ruby, ...
Supported Impl. Lang.s	Java, Perl, Ruby, ...

One platform and one journey for all applications

1

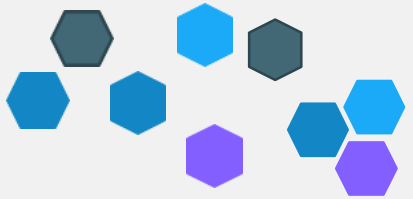
Traditional apps in containers
Gain portability, efficiency and security

2

Transform to Micro services
Look for shared services to transform

3

Accelerate New Applications
Greenfield innovation



Difference between VM Image and VM instance

- A virtual Machine Image (VMI) is a file (e.g., XML) that describes the software configurations of a virtual machine.
- From an VMI, you launch an instance, which is a copy of the VMI running as a virtual server in the cloud. An instance type has hardware configurations.

Table 1: VMI Configurations

Name	Example Values
Virtualization Format	Xen, VMWare, ...
Operating System (OS)	Linux, Windows, ...
OS Version	Ubuntu 10.4, ...
Software Feature	Application Server, Load Balancer, Database, ...
Software	JBoss, Nginx, MySQL, ...
Software Vers.	0.8-alpha,...
Implementation Lang.	Java, Perl, Ruby, ...
Supported Impl. Lang.s	Java, Perl, Ruby, ...

Table 2.1: Non-numerical Instance Configurations

Name	Example Values
Provider	Amazon, Rackspace, ...
Location Country	Germany, Australia, ...
Architecture	32Bit, 64Bit
Owner	Amazon, Rackspace, Bitnami,...

Table 2: Numerical Instance Configurations

Name	Influence	Metric	Range
Hourly Price	Negative	$/h	0-∞ $/h
CPU Cores	Positive	Cores	0-∞
RAM Size	Positive	Bit	0-∞
Disk Size	Positive	Bit	0-∞
CPU Perfomance	Positive	Flops	0-∞ Flops
RAM Perfomance	Positive	Ops/s	0-∞ Ops/s
Disk Perfomance	Positive	B/s	0-∞ B/s
Max. Latency	Negative	ms	0-∞ ms
Avg. Latency	Negative	ms	0-∞ ms
Uptime	Positive	%	0-100%
Service Popularity	Positive	%	0-100%
Network Send Price	Negative	$/Byte	0-∞
Network Recieve Price	Negative	$/Byte	0-∞
Internet Send Price	Negative	$/Byte	0-∞
Internet Recieve Price	Negative	$/Byte	0-∞

Docker Components: Docker Orchestration

Three tools for orchestrating distributed applications with Docker

- Docker Machine

  Tool that provides Docker Hosts and install the Docker Engine on them.

- Docker Swarm

  Tool that clusters many Engines and schedules containers.

- Docker Compose

  Tool to create and manage multi-container application.

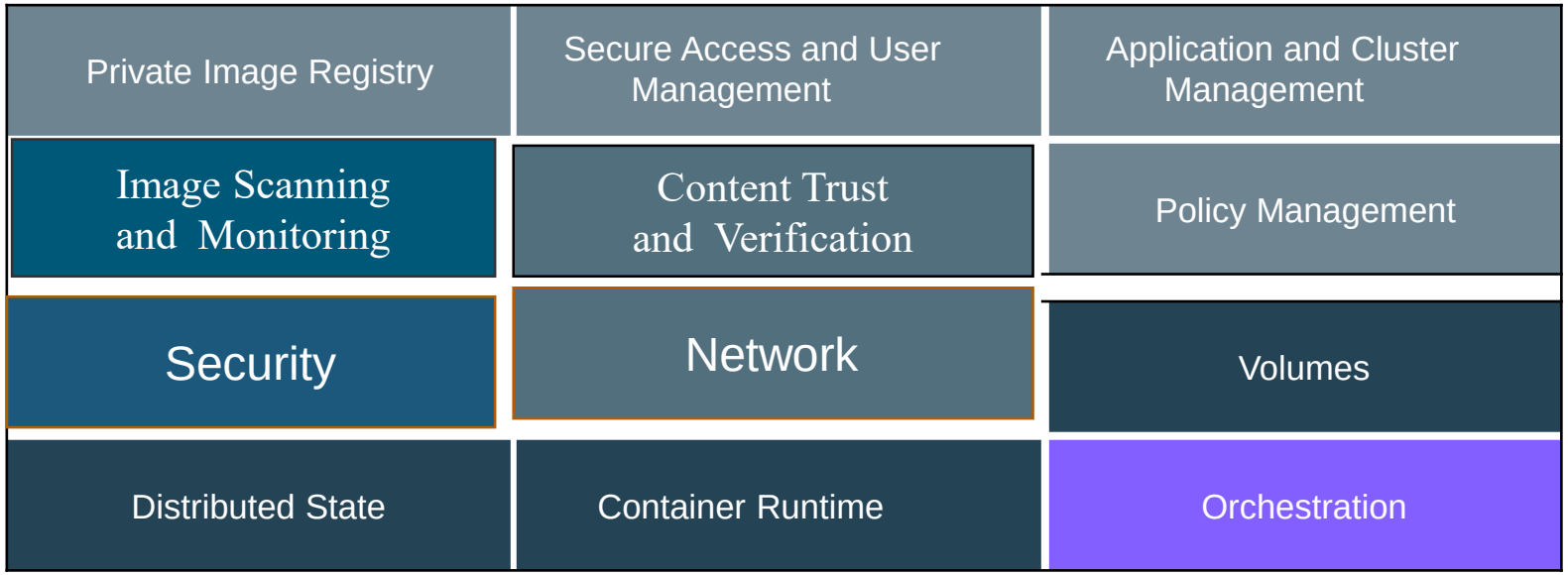
Foundation: Docker Engine

Integrated Security

Security	**Network**	**Volumes**
Distributed State	**Container Runtime**	**Orchestration**

Docker Engine

Building a Secure Supply Chain



Enterprise Edition

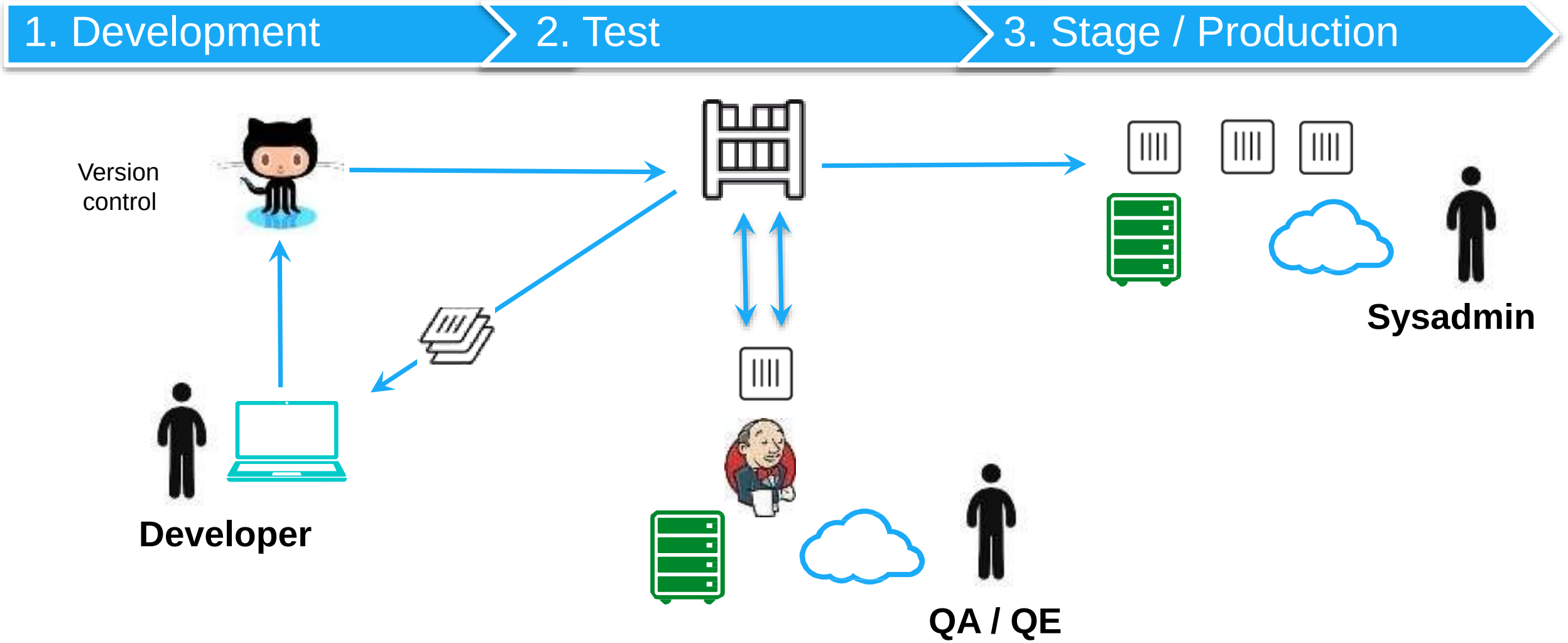
Docker Engine

Usable Security

Trusted Delivery

Portable

Continuous Integration and Delivery



Quick Security Consideration

- Docker demon needs to run as root.
- Ensure that, only trusted can control Docker Demon.
- If binding the demon to a TCP socket, secure it with TLS
- Use linux hardening solutions
  - SELinux
  - GRSEC

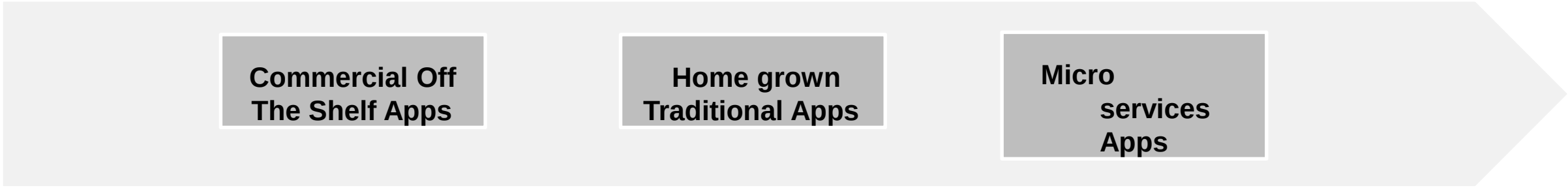
Docker delivers agility, security and cost savings

Hardened containers deliver new levels of security to monoliths on the transition to micro services

Transform monoliths to secure and agile DevOps environments

Reduce maintenance costs by 10X for legacy, commercial and new apps

Docker delivers agility, resiliency, portability security and cost savings for all applications



13X

More software releases

65%

Reduction in developer on boarding time

~47%

Reduction in VMs, OS licensing and Server costs

Eliminate

"works on my machine" issues

62%

Report reduction in MTTR

10X

Cost reduction in maintaining existing applications

IntelliPaat

eict IIT Roorkee

IIT Roorkee

India: +91-7022374614

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor