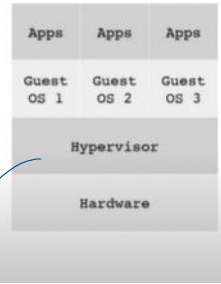
System VM

Hypervisors

- Also called Virtual Machine Monitor (VMM)
- A hypervisor is an operating system for operating systems
  - Provides a virtual execution environment for an entire OS and its applications
  - Controls access to hardware resources
  - When guest OS executes a privileged instruction, Hypervisor intercepts the instruction, checks for correctness and emulates the instruction.



VM tries to stay completely invisible

to do this they intercept instructions & emulates it.

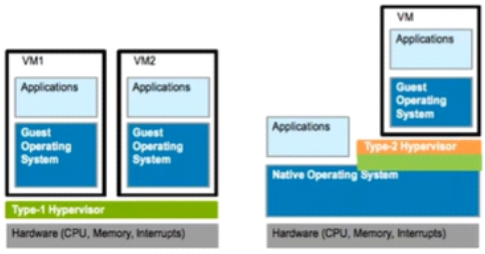
guest OS does not know hypervisor does this.

1st Hypervisor gets booted up

does same thing as OS but for VM's.

Type of Hypervisors

- Type 1 Hypervisor (bare metal, native): supports multiple virtual machines and runs directly on the hardware (e.g., VMware ESX , Xen, Denali)
- Type 2 Hypervisor (hosted) VM - runs under a host operating system (e.g., user-mode Linux)



https://microkerneldude.files.wordpress.com/2012/01/type1-vs-2.png

VM's will initiate & invoke previlleged instructions, But Hypervisior will trap them and emulate them.

⇒ Type-2 Hypervisior gives some flexibility in terms of implementation.

⇒ We're making hypervisor part of OS & user space.

⇒ it must have all drivers that is necessary to access

⇒ we're making Hypervisor part of OS & must
⇒ it must've all drivers that is necessary to access hardware.

⇒ why'll write all drivers from scratch ?

- already present in OS.
- why write which Type-1 Hypervisor doing.

we take advantage of already present drivers.

native/host OS — bare metal
guest OS — in VM running

Type-2 Hypervisors

Put Hypervisor in kernel module in OS to take over Previlleges from host OS & some lib. that are required to run apps.

Para-virtualized VMs

- Modify guest OS for better performance
- Traditional Hypervisors provide full-virtualization
  - They expose to VMs virtual hardware that is functionally identical to the underlying physical hardware.
  - Advantage : allows unmodified guest OS to execute
  - Disadvantage: Sensitive instructions must be trapped and emulated by Hypervisor.
  - E.g. KVM and VMWare ESX provide full virtualization
- Para-virtualized VM
  - Sees a virtual hardware abstraction that is similar, but not identical to the real hardware.
  - Guest OS is modified to replace sensitive instructions with "hypercalls" to the Hypervisor.
  - Advantage: Results in lower performance overhead

- Para-virtualized VM
  - Sees a virtual hardware abstraction that is similar, but not identical to the real hardware.
  - Guest OS is modified to replace sensitive instructions with "hypercalls" to the Hypervisor.
  - Advantage: Results in lower performance overhead
  - Disadvantage: Needs modification to the guest OS.
  - Xen provides both para-virtual as well as full-virtualization

→ if Hypervisor wants to get fully transparent all the instructi. must be trapped & emulated.

Para-virtualized VMs →

guest os modified
not all the instruction
trapped. & imulate

#only sensitive info.] one set

which -ll ve executed through hyp

& rest instr. -ll be system c

hypervisor not fully transp

→ often additional hyp

are partially Pa

device drivers in g

Para-virtualized

CPU & memory ma

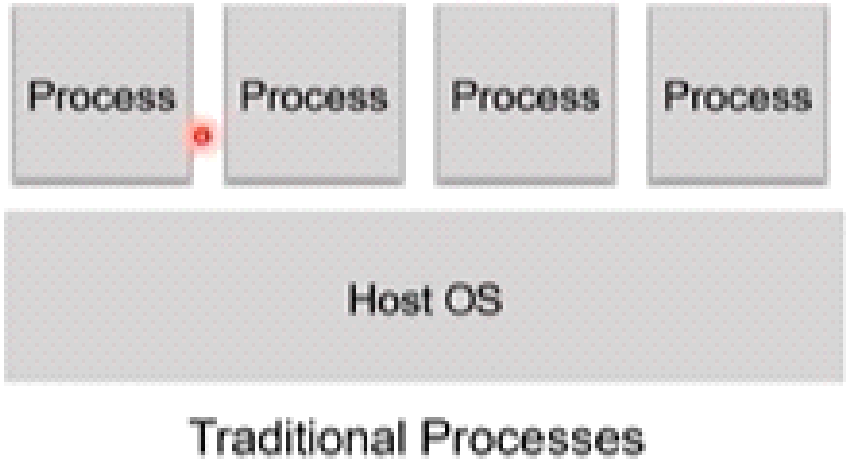
ra - virtualized

est OS may be

. whereas

y be fully virtualized

Traditional Process vs traditiona

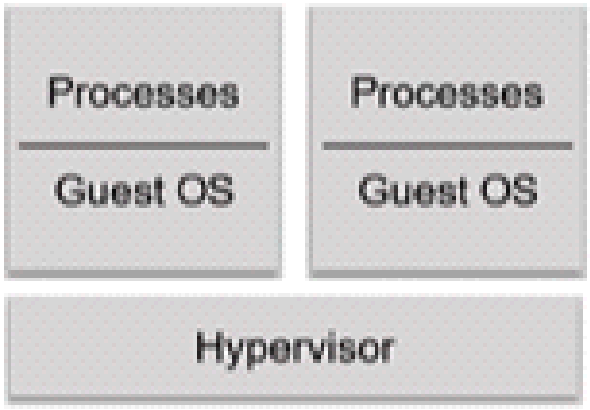


Traditional Processes

Resources shared among Processes.

→ some virtualization / multi

l VMs



Traditional VMs

isolation blw Process that are running on top of these two guest os is completely

plexing

→ Some Virtualization / multi
done by OS in this lev

→ Loosely isolated

→ Processes will talk thr
System calls & Eg.
Sys calls. in Linux.

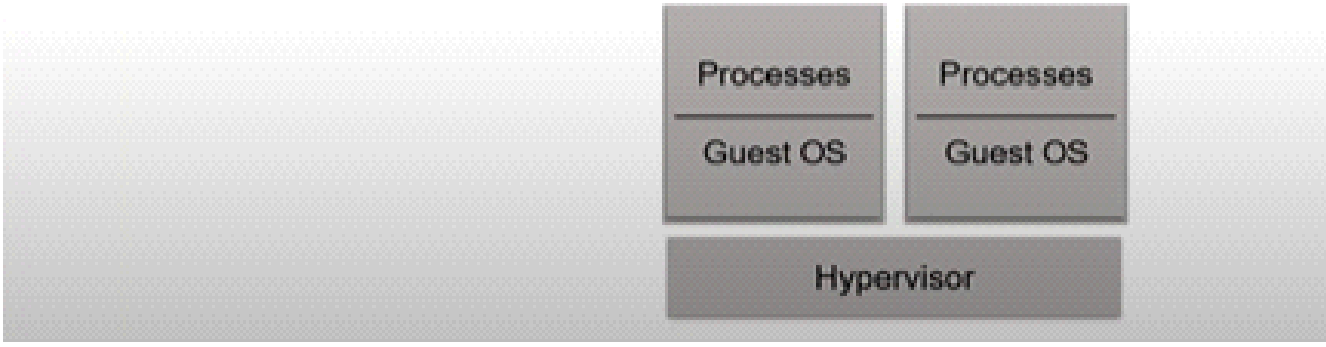
→ all sys. calls available
resources.

Completely rigid.

⇒ resources shared b/w guest OS're completely managed by Hypervisor

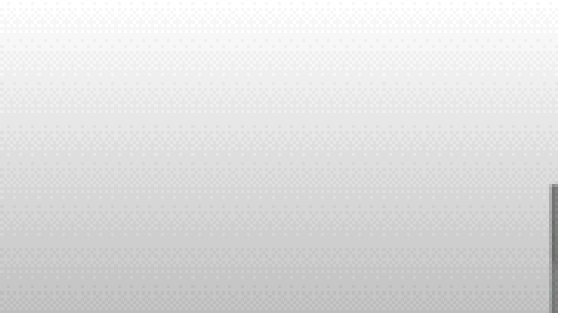
System VMs

- Each VM has its own
  - Guest OS
  - Guest physical memory ("virtualized" view of m
  - One or more virtual CPUs
  - Virtual I/O devices: virtual disk, virtual network
- Ideally: Co-located VMs don't see/share ANYTHING



emory seen by guest OS

S



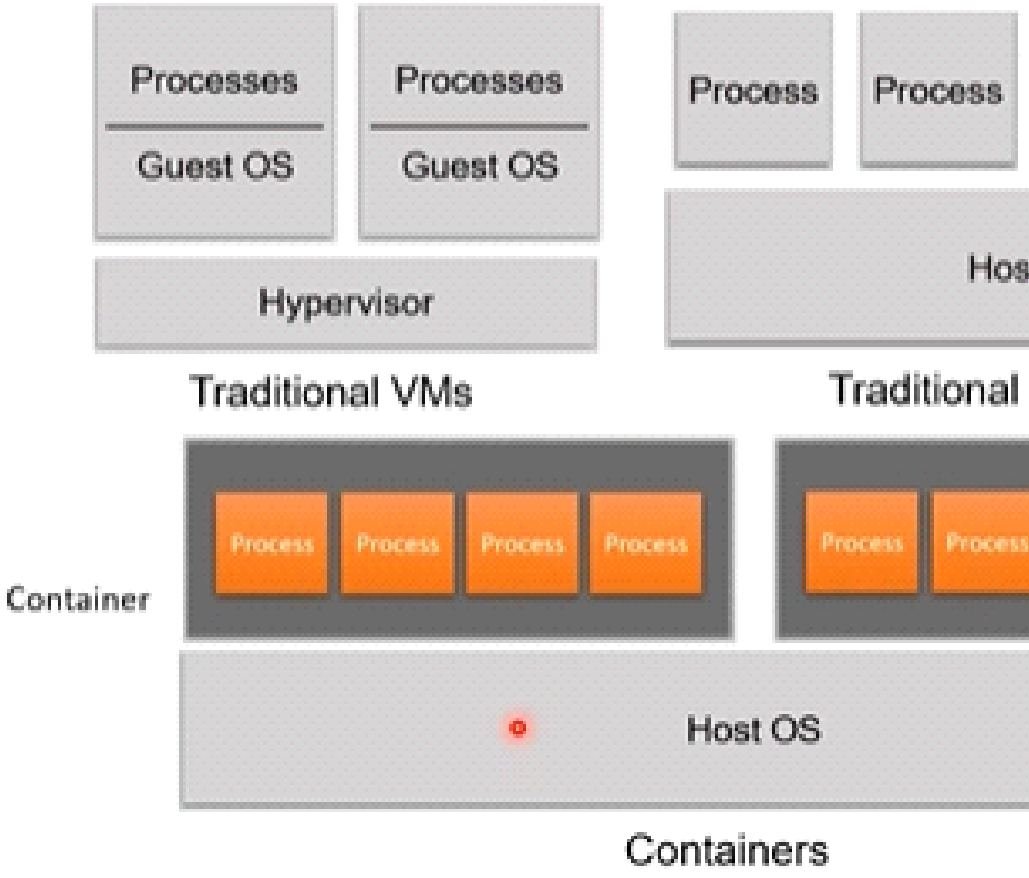
Isolation is important

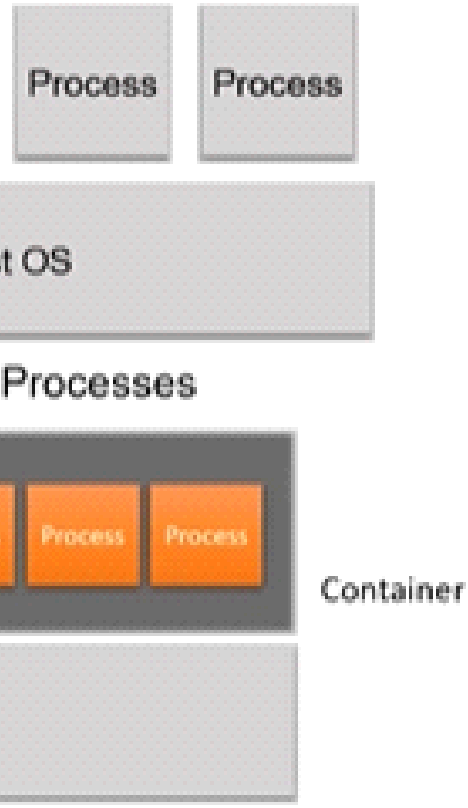
- Limiting what/who a process/application can see.
- Limiting who can see a process/application

- Processes share too much
  - Great performance but not isolated enough
  - System VMs are too heavy
- Great Isolation but too heavy due to separate gues system-level virtualization
  - Multiple isolated user-spaces
  - Share one kernel
  - Native performance

t OS per VM Operating-

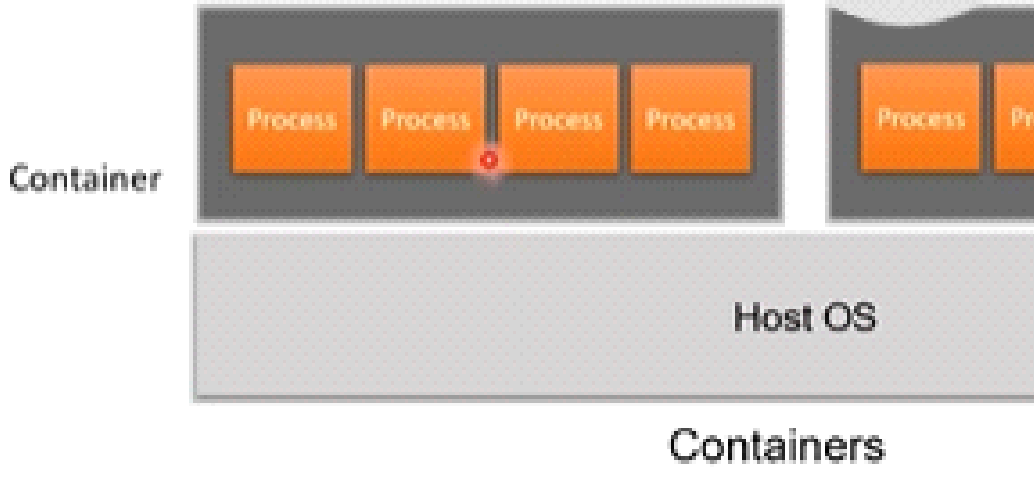
Process VMs and Containers




Process
Process
t OS
Processes
Process
Process
Container

Containers

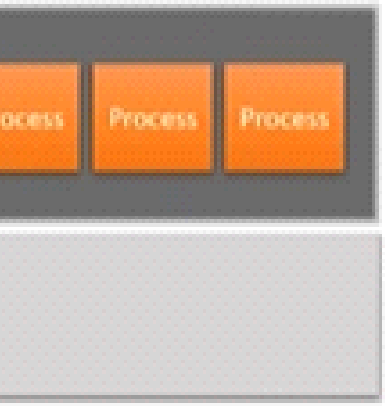
- Containers
  - group traditional processes together and
  - restrict what resources they can see/access
- In Linux, containers consist of
  - Namespaces
  - Control Groups (cgroups)



Containers

Processes share names
create one seperate
group of processes.

.



spaces - So if you
namespace & control

⇒ diff. containers will inv
to resource OS.

⇒ in terms of sys level v
one cannot get full
But with containers, u
approach, where we c
for these Processes & b/w these containers.

oke host sys. calls

Ms or fully virtualized VMs

isolation of Processes.

e can get Big resource

an've namespace seperated

isolated resource sharing

Cloud Computing

- Virtualized distributed processing, storage
  service.

- Delivering computing as a on-demand, pay

- "A model for enabling convenient, on-dem
  pool of configurable computing resources
  applications, and services) that can be rapi
  minimal management effort or service pro
  definition

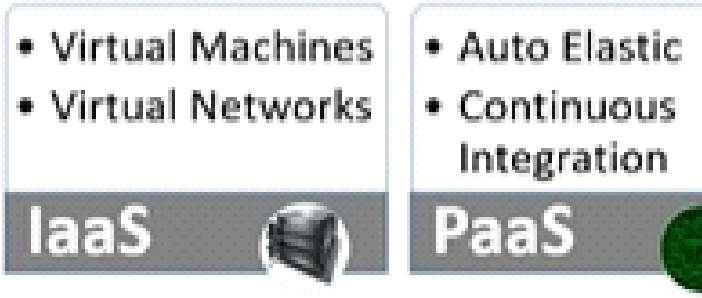
, and software resources and a

y-as-you-go service.

and network access to a shared
(e.g., networks, servers, storage,
idly provisioned and released with
vider interaction" – NIST

Service models

- IaaS: Infrastructure as a Service
  - Consumer can provision computing resourc
    upon which they can deploy and run arbitra
    applications
- PaaS: Platform as a Service
  - Consumer can create custom applications
    by the provider and deploy them onto the
- SaaS: Software as a Service
  - Consumer uses provider's applications runn
    infrastructure



ces within provider's infrastructure
ary software, including OS and

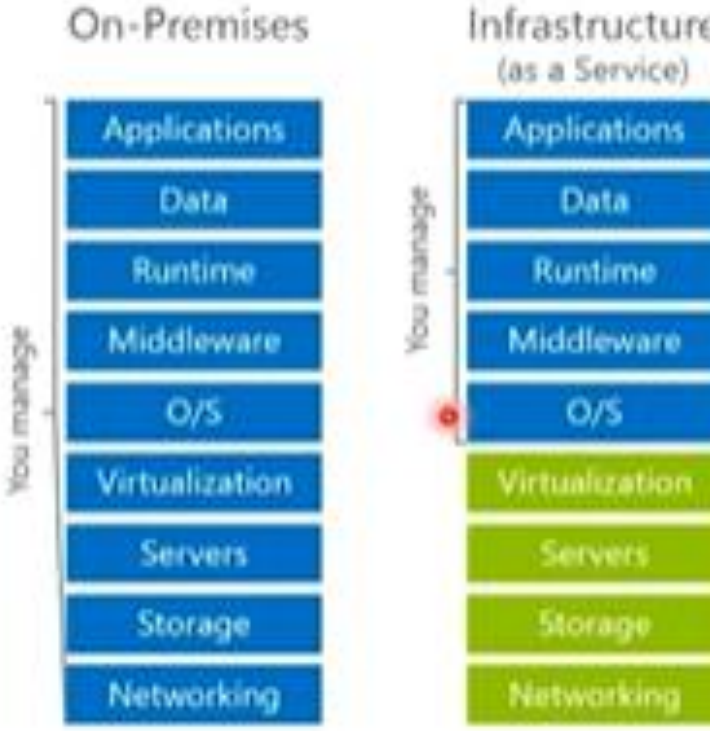
using programming tools supported
provider's cloud infrastructure

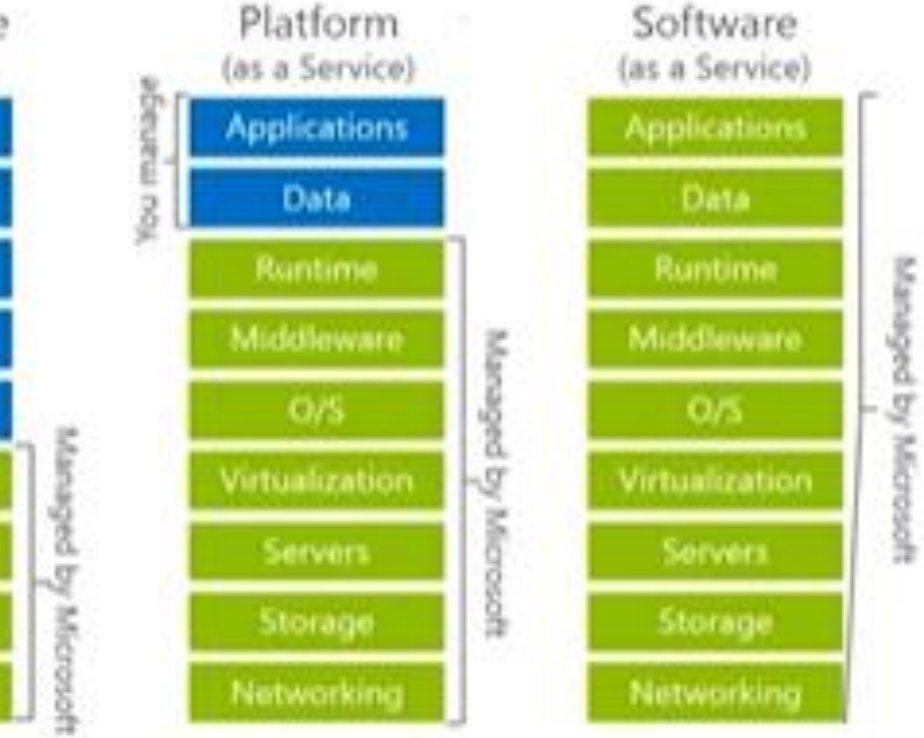
ning on provider's cloud

- Built for Cloud
- Uses PaaS

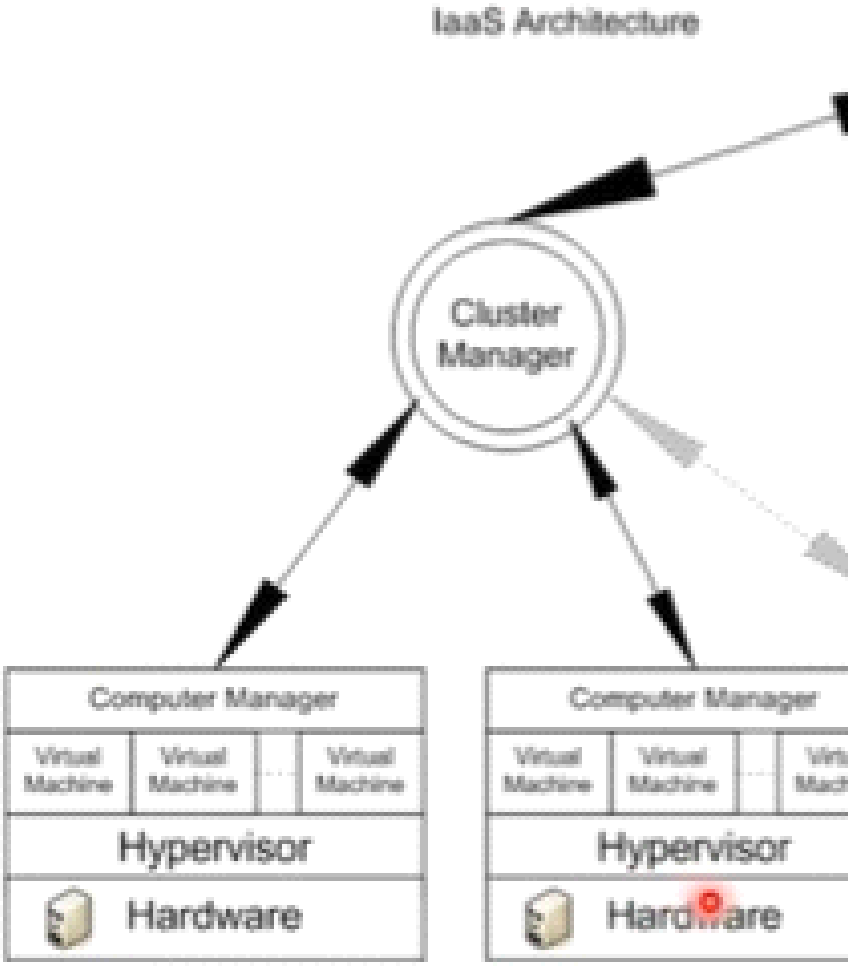
SaaS

Service models

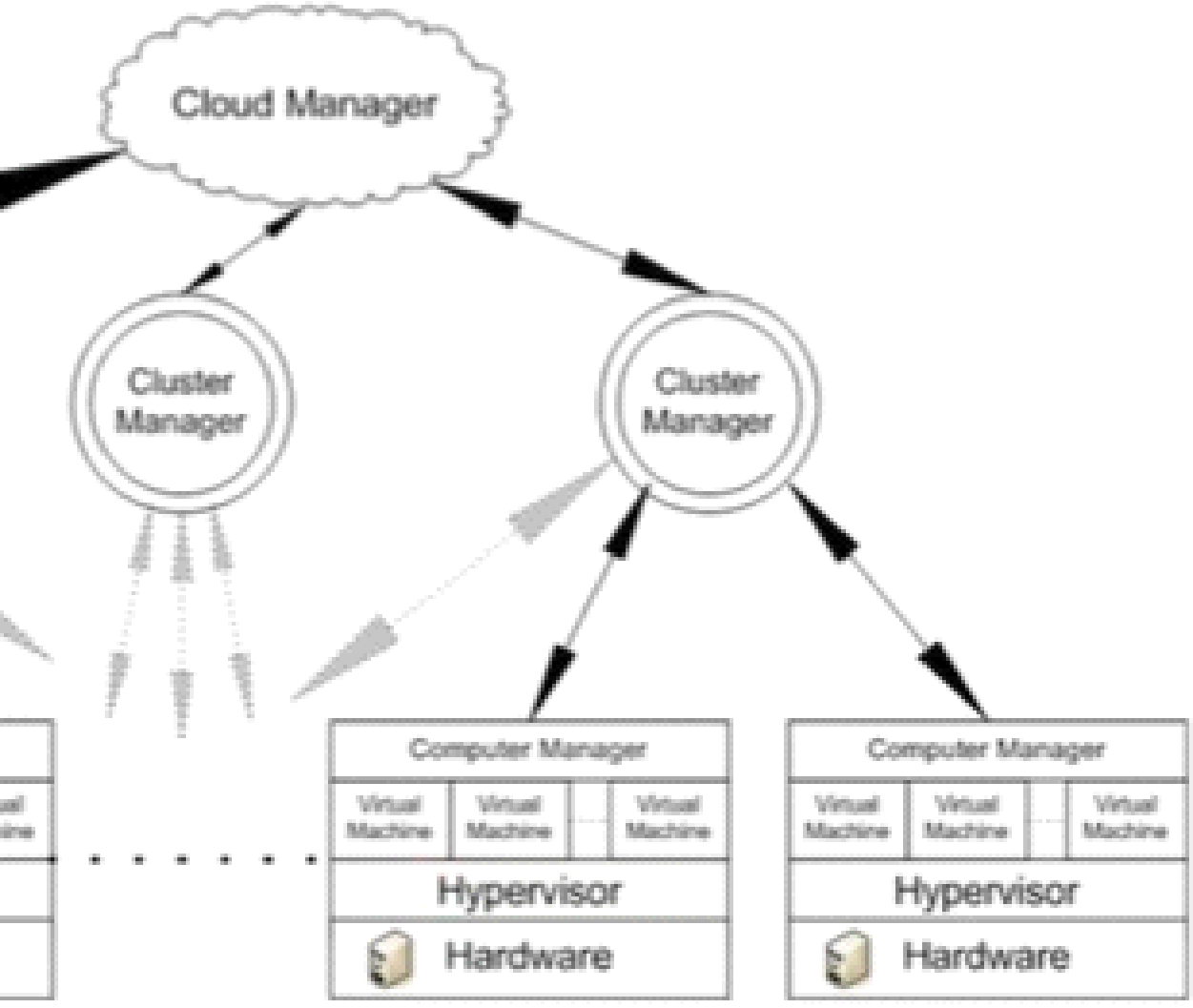



Platform
(as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
Managed by Microsoft
Software
(as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Managed by Microsoft

IAAS cloud acrhitectur



re


Cloud Manager
Cluster Manager
Cluster Manager
Computer Manager
Virtual Machine
Virtual Machine
Virtual Machine
Hypervisor
Hardware
Computer Manager
Virtual Machine
Virtual Machine
Virtual Machine
Hypervisor
Hardware