

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.

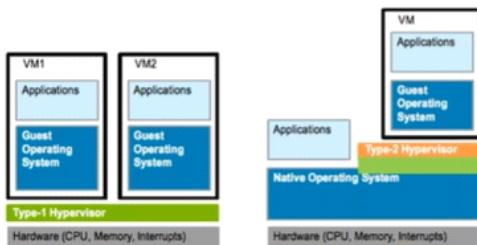


to do this they intercept instructions & emulates if guest os does not know hypervisor does this. 1st Hypervisor gets booted up does same thing as OS but for VM's



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 privileged instructions, but hypervisor will trap them and emulate them.

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

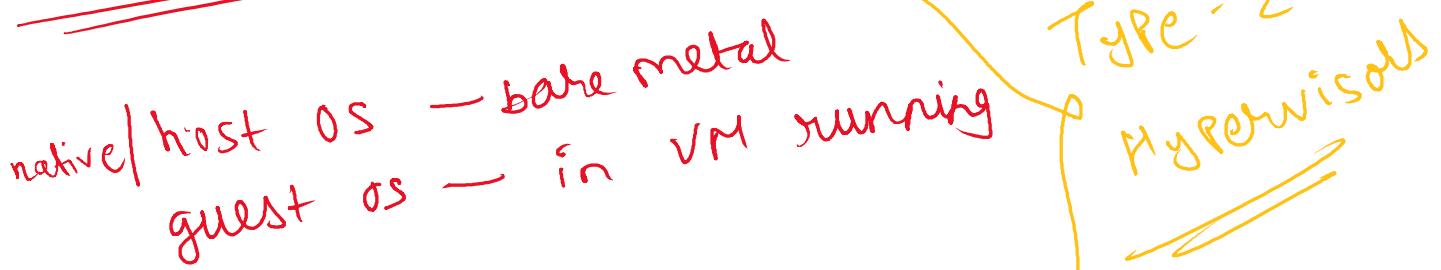
⇒ we're making hypervisor part of OS & user space.
↳ ... we'll divide that is necessary to access

⇒ We're making Hypervisor part of OS & now we
⇒ 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.



Put Hypervisor in kernel
module in OS to take over
privileged 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 "hypervcalls" 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.



guest os modified
not all the instruction
trapped & immediate

only sensitive
info.] one set

which -> ve executed through hyp

& rest instr. -> be system co

Hypervisor not fully transpar

→ often + additional hy
i...ll...

permissive
cells.
agent

permissive
inhalized

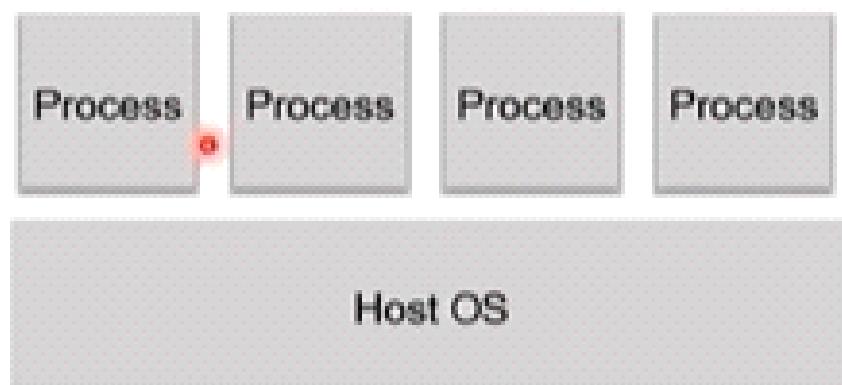
are partially Pa

→ device drivers in g
Para-virtualized
CPU & memory ma

partially virtualized
host OS may be
, whereas
they be fully virtualized



Traditional Process vs traditional

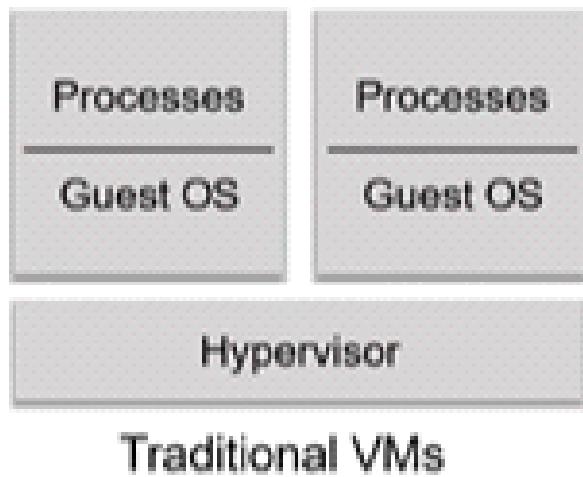


Traditional Processes

Resources shared
among processes.

→ some virtualization / multi

Virtual VMs



isolation b/w processes
that are running
on top of these
two guest os is
completely
plexing

- some virtualization / memory management done by OS in this layer
- loosely isolated
- Processes will talk through system calls & Eg. sys calls in Linux.
- all sys. calls available resources.

Completely
rigid.

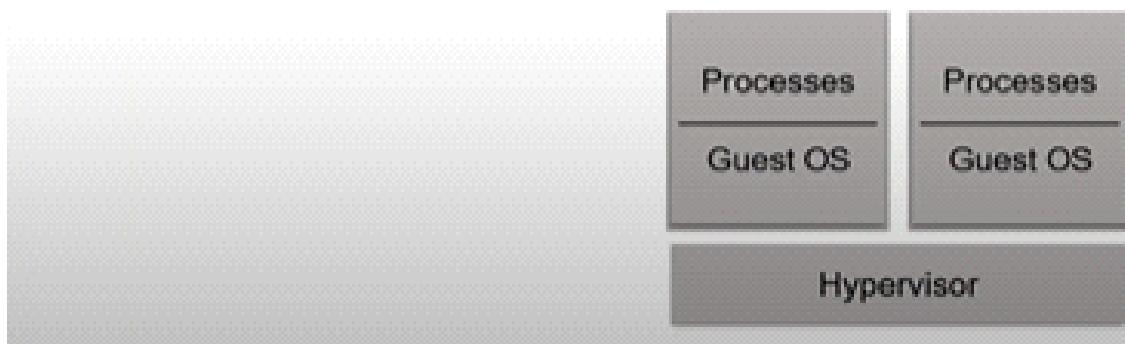
→ resources
shared b/w
guest os's
completely
managed by
Hypervisor

through
400
for all



System VMs

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



memory seen by guest OS'

2



Isolation is important

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

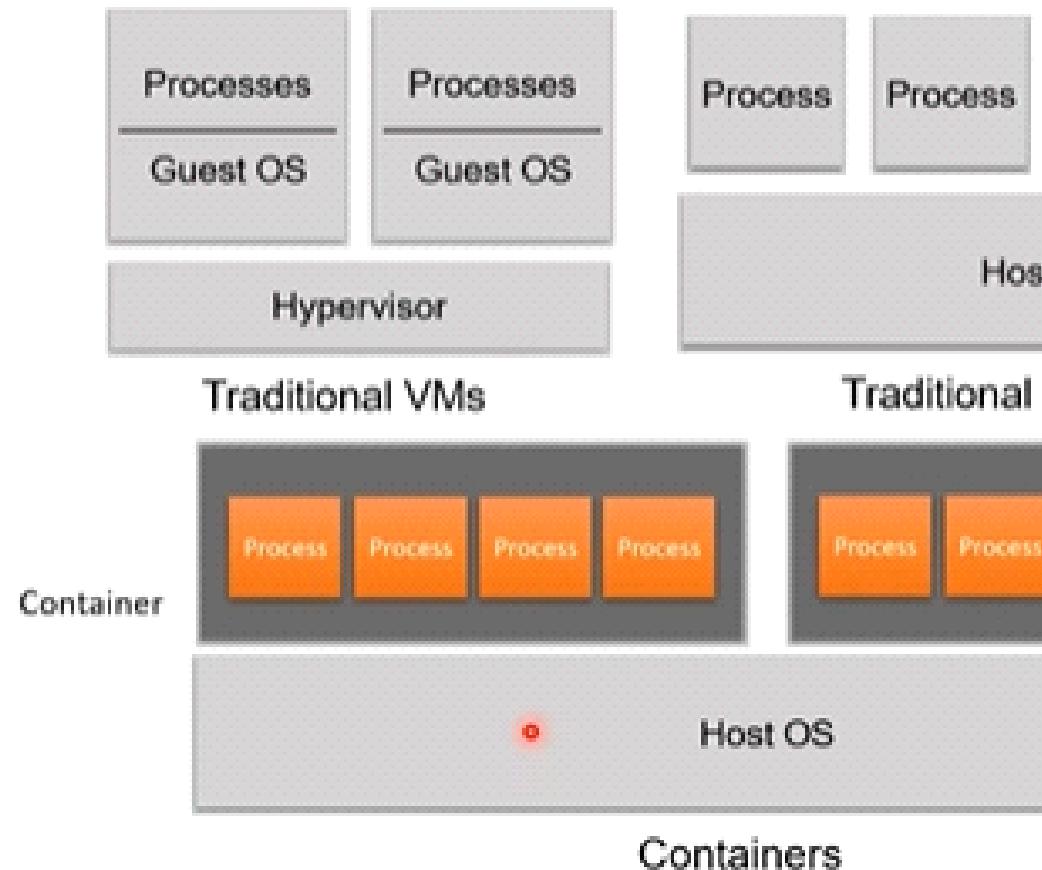
- Processes share too much
 - Great performance but not isolated enough
 - System VMs are too heavy
- Great Isolation but too heavy due to separate guest system-level virtualization
 - Multiple isolated user-spaces
 - Share one kernel
 - Native performance

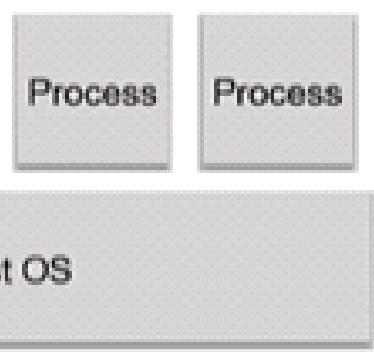
et OS per VM Operating-



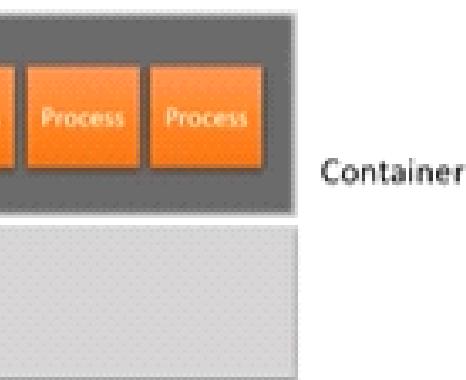


Process VMs and Containers





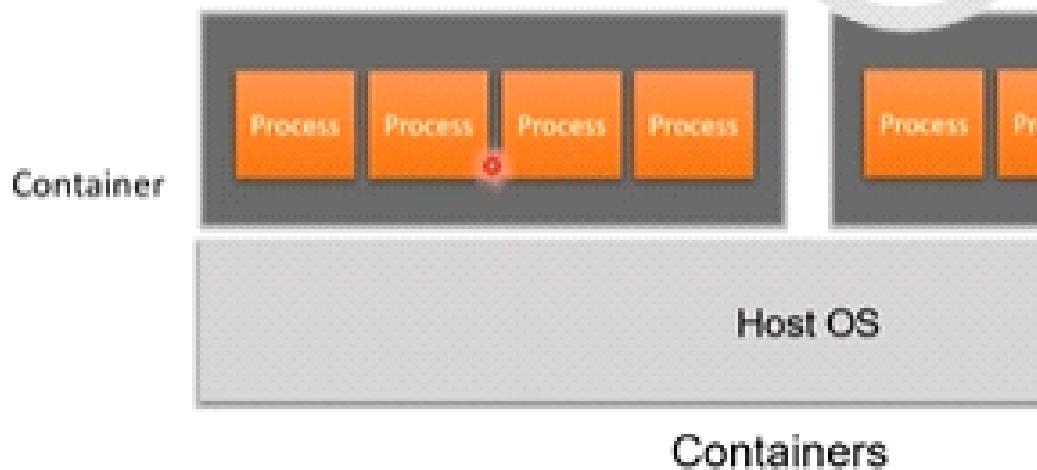
Processes



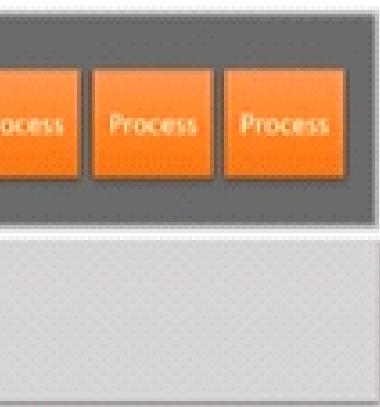


Containers

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



Processes share names
create one separate
group of processes .



spaces, So if you
namespace & control

⇒ diff. containers will invoke
to resource OS.

⇒ in terms of sys level v.
one cannot get full
But with container , we can
approach , where we can
run these processes . &
blw these containers -

invoke host sys. calls

M_s or fully virtualized VM_s

isolation of processes.

We can get big resource

an've namespace separated

isolated resource sharing



Cloud Computing

- Virtualized distributed processing, storage service.
- Delivering computing as a on-demand, pay
- "A model for enabling convenient, on-demand pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider definition"

, and software resources and a

y-as-you-go service.

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



Service models

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

- Virtual Machines
- Virtual Networks

IaaS

- Auto Elastic
- Continuous Integration

PaaS

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

using programming tools supported
provider's cloud infrastrcture

nning on provider's cloud

- Built for Cloud
- Uses PaaS

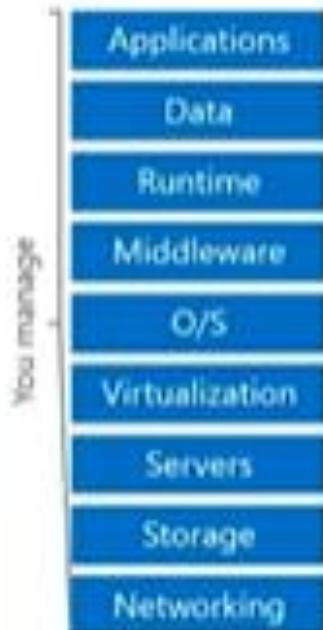
SaaS



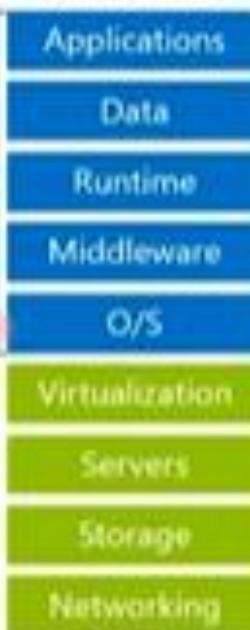


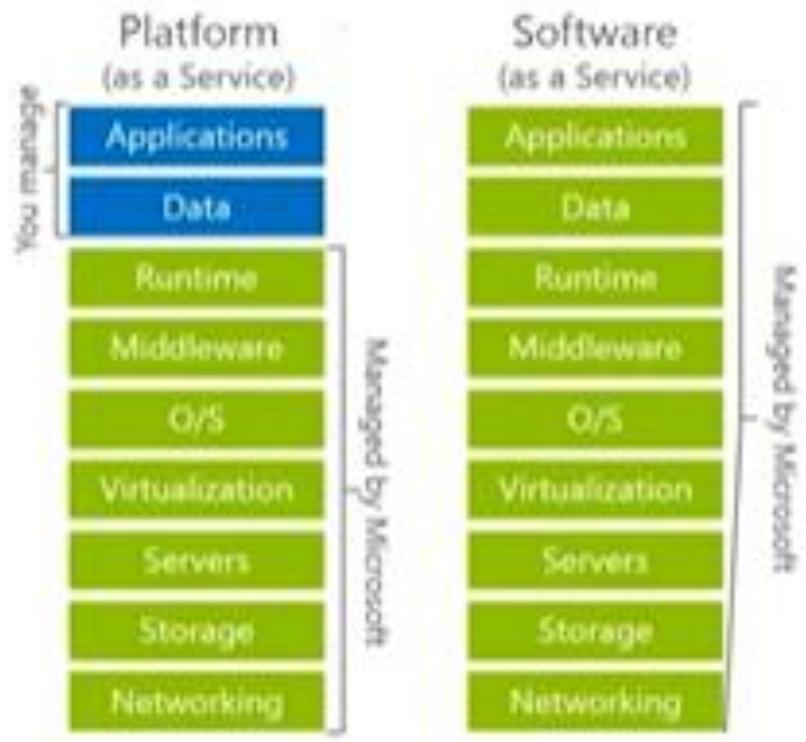
Service models

On-Premises



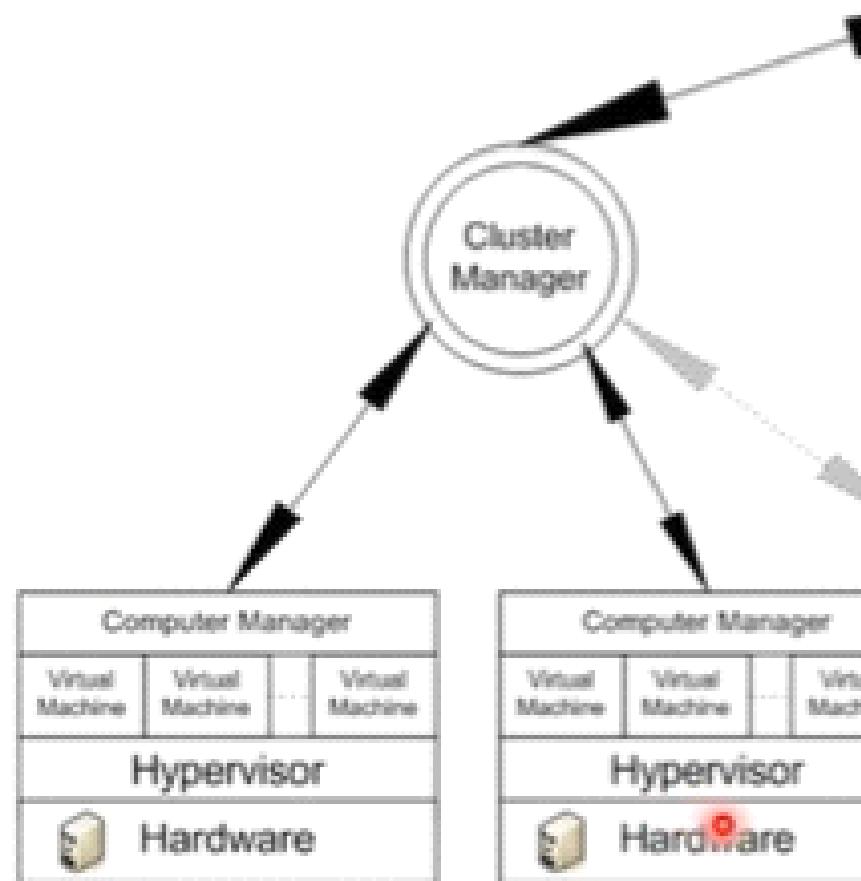
Infrastructure (as a Service)





IAAS cloud acrhitectur

IaaS Architecture



re

