Cloud computing

Tema 2. Fundamentos

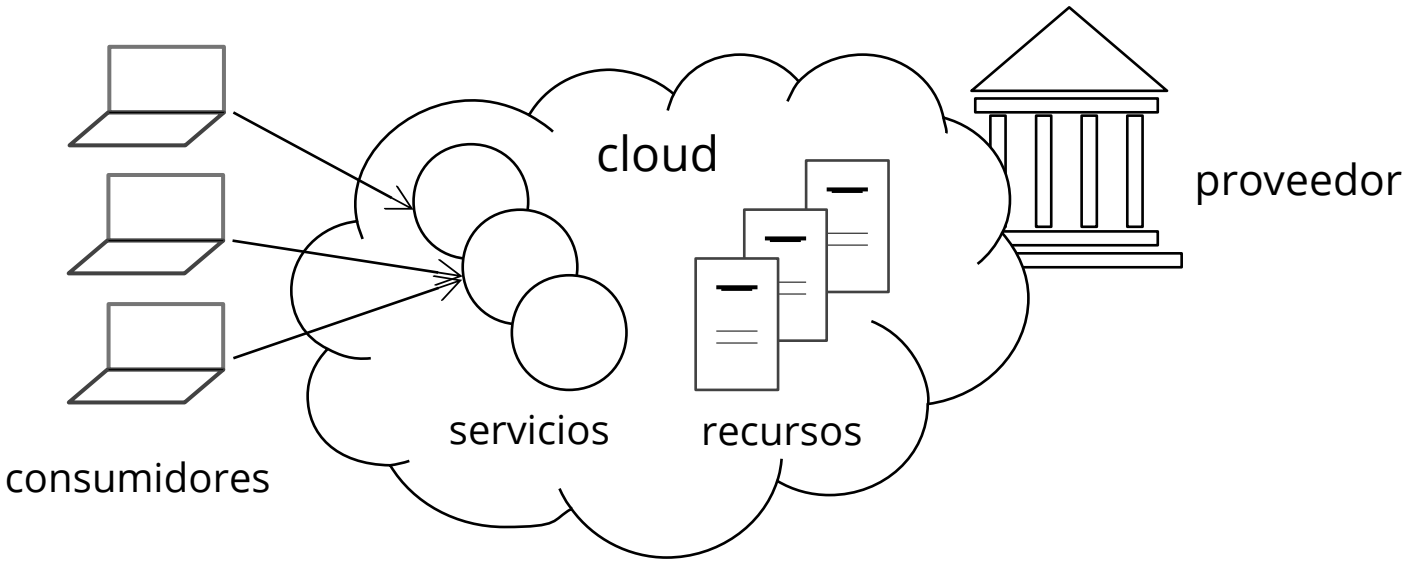
Contenido

- Introducción
- SOA
  - Servicios
  - Principios de diseño
  - Servicios RESTful
- Virtualización
  - Virtualización hardware
  - Libvirt
- Contenedores
  - Docker

Introducción

El modelo

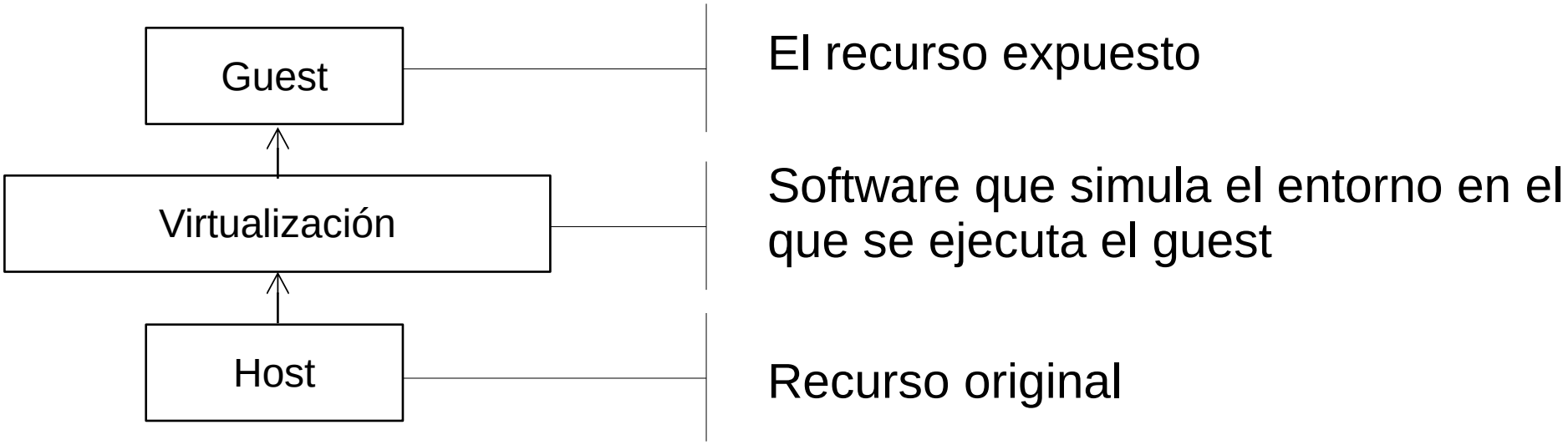
- Recursos computacionales que son consumidos a través de servicios
- Los recursos computacionales se implementan con técnicas de virtualización



Virtualización

¿Qué es?

- El proceso de crear una versión virtual (no real), abstracta o lógica de algo (hardware, dispositivo, …)
- Se identifican tres capas: guest, virtualización y host
- Desacople entre guest y host



Virtualización

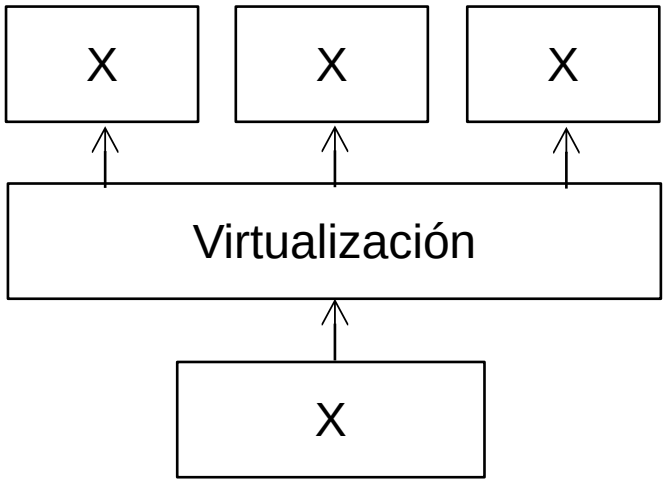
Estrategias de virtualización

- Multiplexado (o compartición)
- Agregación
- Emulación

Virtualización

Estrategias de virtualización

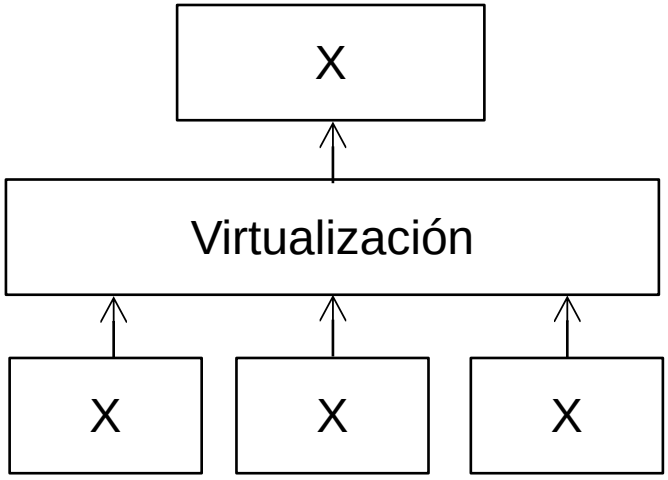
- Multiplexado (o compartición)
  - Múltiples recursos virtuales a partir de uno físico
  - Mayor tasa de utilización
  - En el espacio (páginas memoria) y en el tiempo (CPU)



Virtualización

Estrategias de virtualización

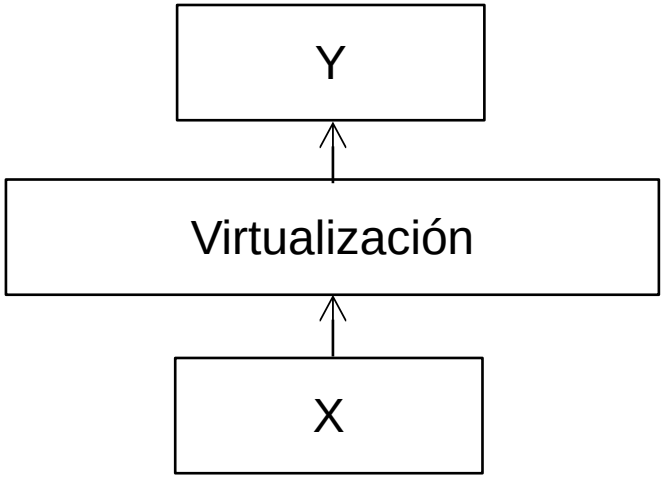
- Multiplexado (o compartición)
- Agregación
  - Múltiples recursos físicos se exponen como uno sólo (e.g. RAID)



Virtualización

Estrategias de virtualización

- Multiplexado (o compartición)
- Agregación
- Emulación
  - Recurso virtual diferente al recurso físico



Virtualización

Estrategias de virtualización

- Multiplexado (o compartición)
- Agregación
- Emulación
- Las tecnologías de virtualización actuales combinan todas estas técnicas de virtualización

Virtualización

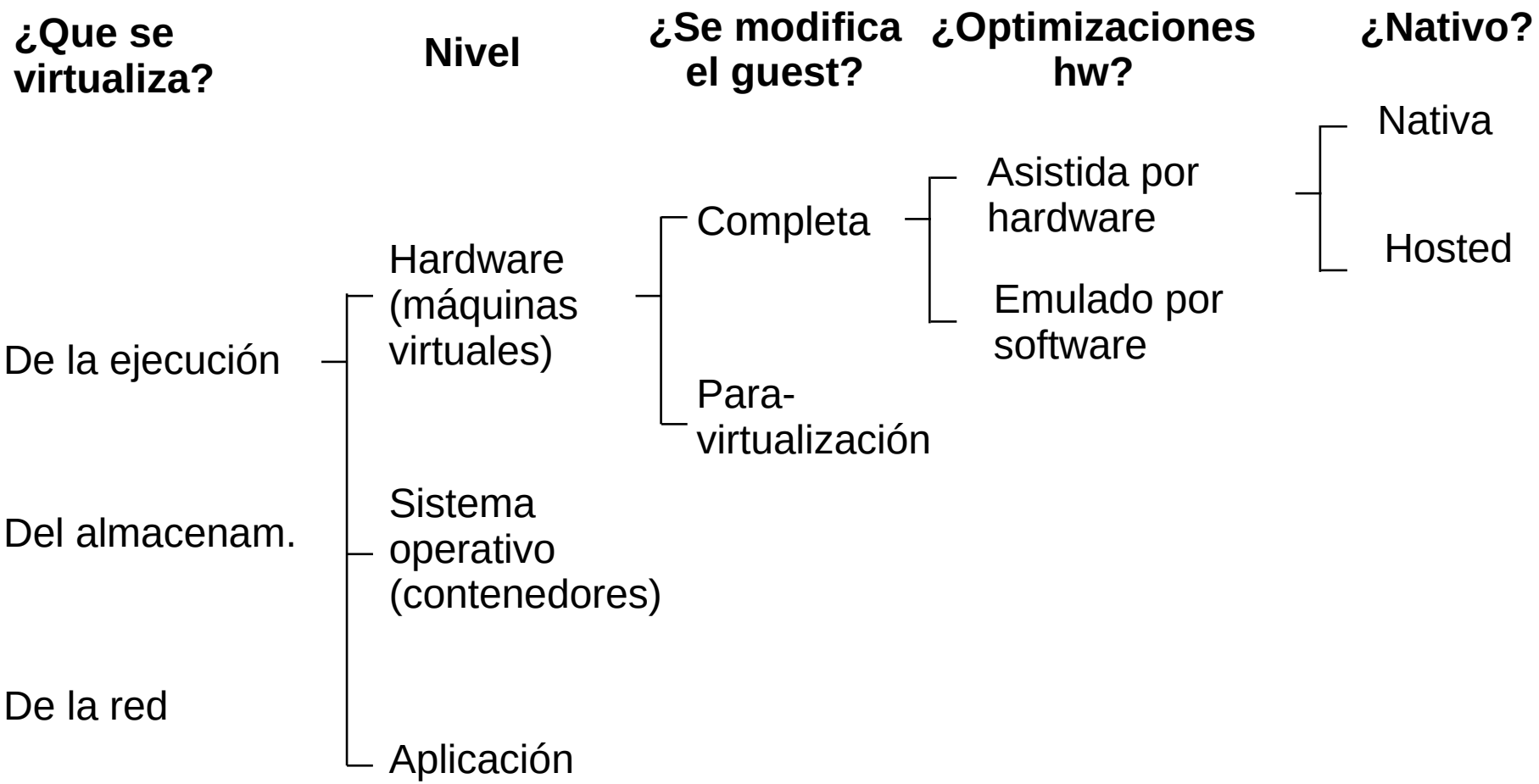
Beneficios

- Utilización de recursos: consolidación de servidores
- Impacto medioambiental
- Eficiencia y productividad
- Reducción de costes
- Seguridad
- Escalabilidad, fiabilidad y resiliencia
- Portabilidad

Virtualización

Clasificación de técnicas de virtualización

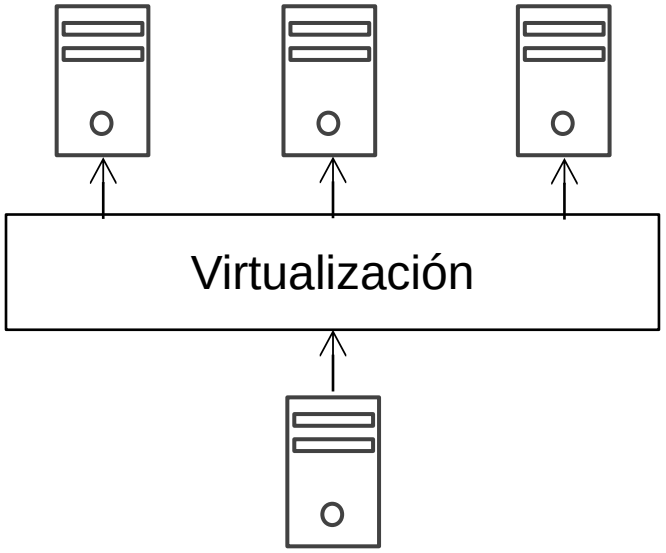
- No completa



Virtualización

Tipos de virtualización según el recurso

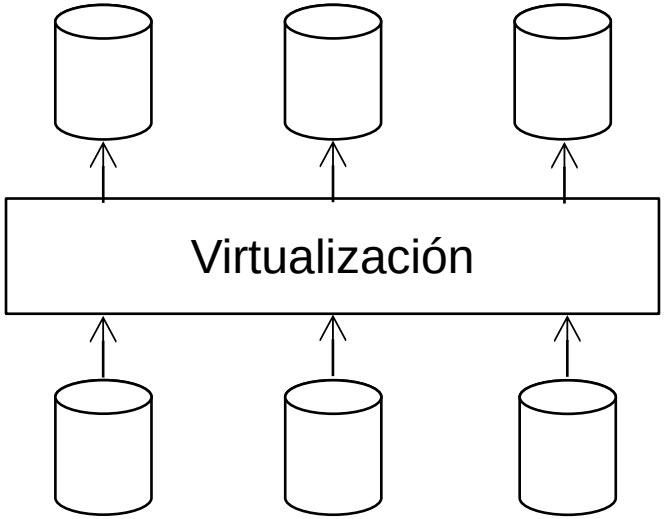
- Virtualización de la ejecución
  - Emulación de un entorno de ejecución
  - Máquinas virtuales (e.g. VirtualBox, JVM)



Virtualización

Tipos de virtualización según el recurso

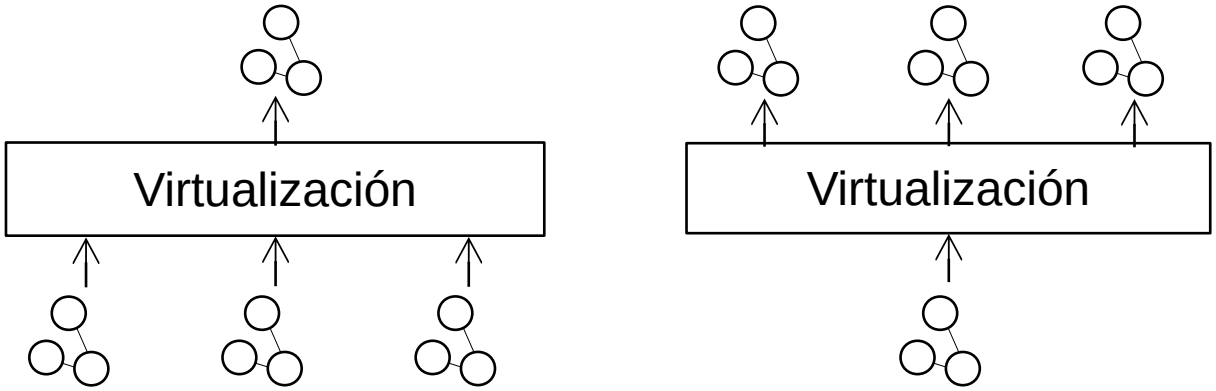
- Virtualización de la ejecución
- Virtualización del almacenamiento
  - Organización física diferente a la organización lógica
  - SDS (*Software Defined Storage*)



Virtualización

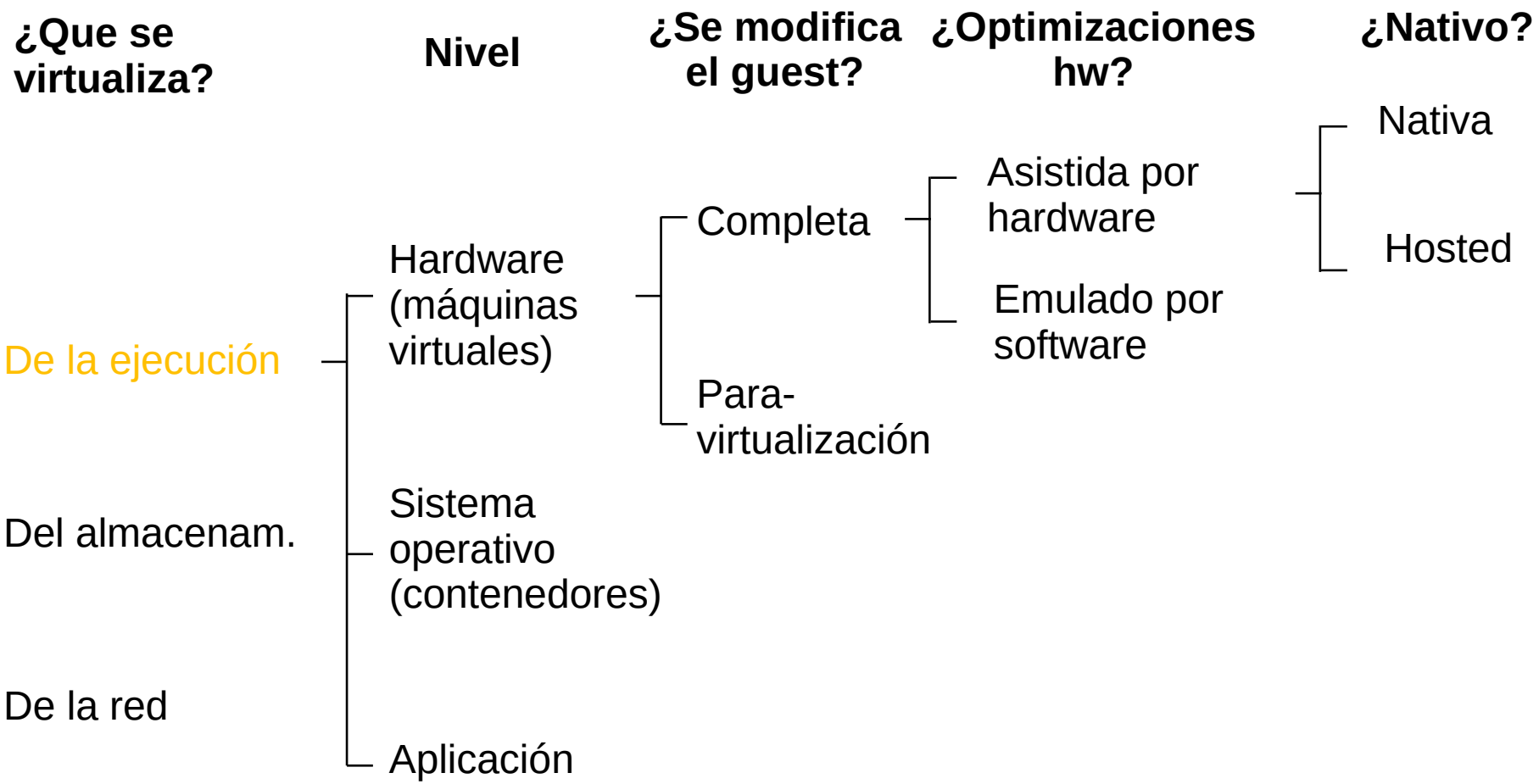
Tipos de virtualización según el recurso

- Virtualización de la ejecución
- Virtualización del almacenamiento
- Virtualización de la red
  - Externa (VLAN) vs interna
  - SDN (*Software Defined Network*)



Virtualización de la ejecución

Virtualización de la ejecución



Virtualización de la ejecución

¿Qué es?

- Se emula un entorno de ejecución
- Sistema computacional como sistema en capas

Application Programming Interface

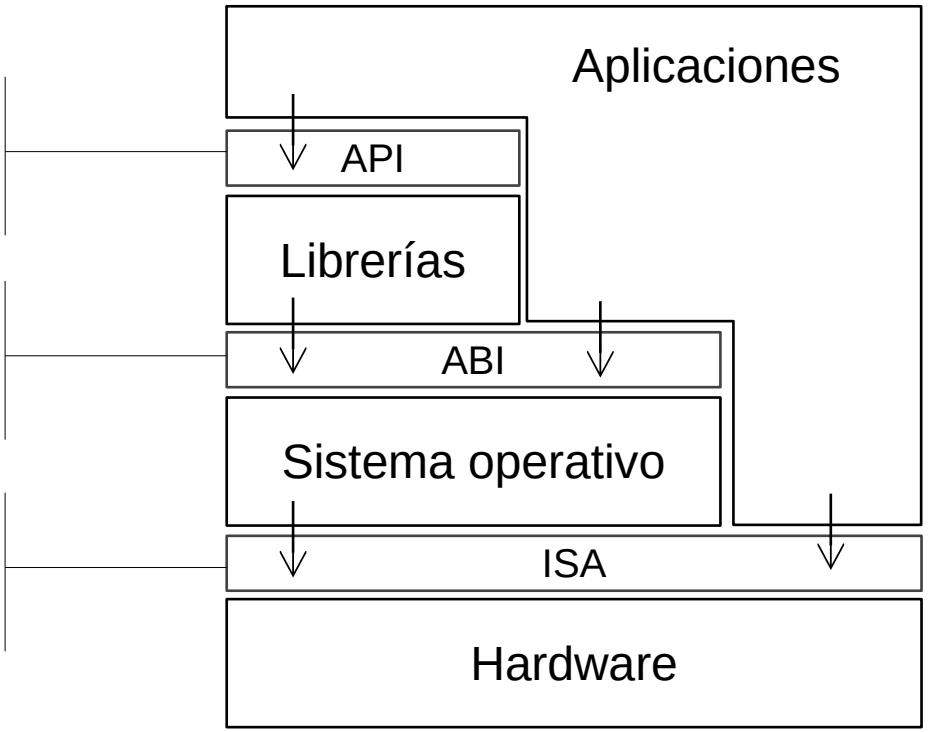
- Accedidas por las aplicaciones

Application Binary Interface

- Llamadas al sistema, ejecutables, memoria, funciones, etc.

Instruction Set Architecture

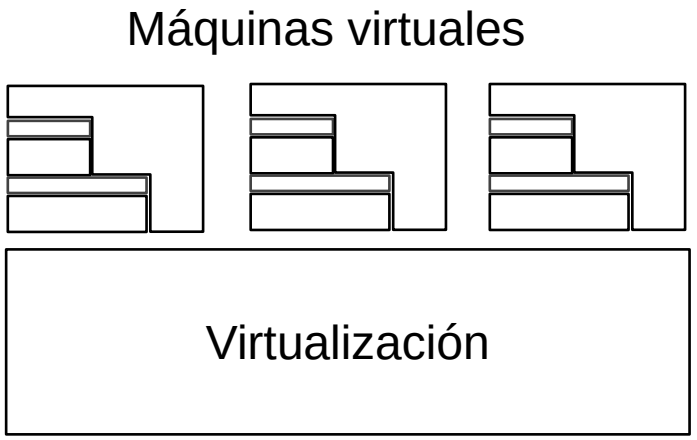
- Juego instrucciones del procesador

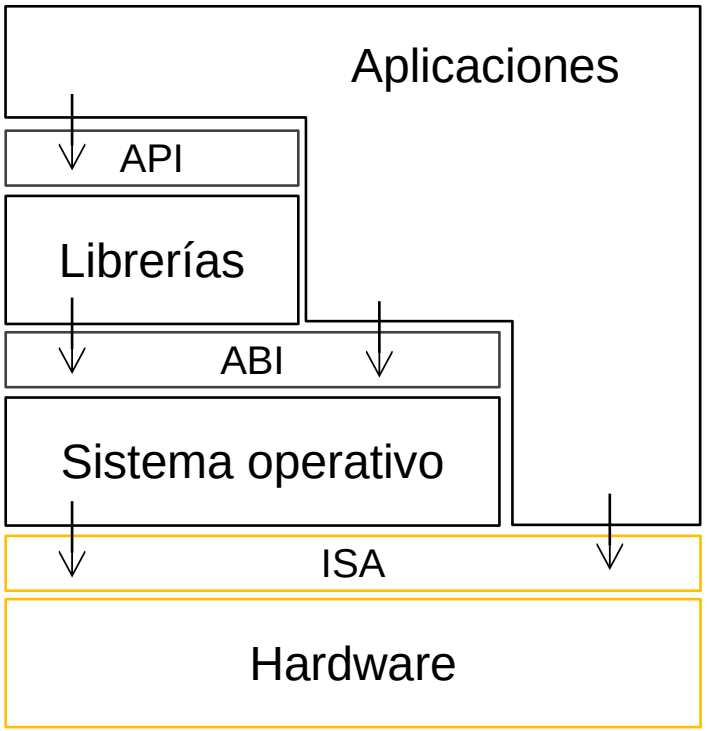


Virtualización de la ejecución

Niveles de virtualización

- Según la capa que se virtualice
- Virtualización de hardware
  - Máquinas virtuales

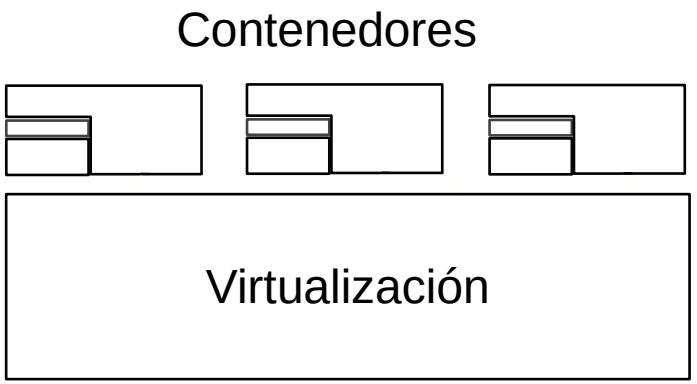


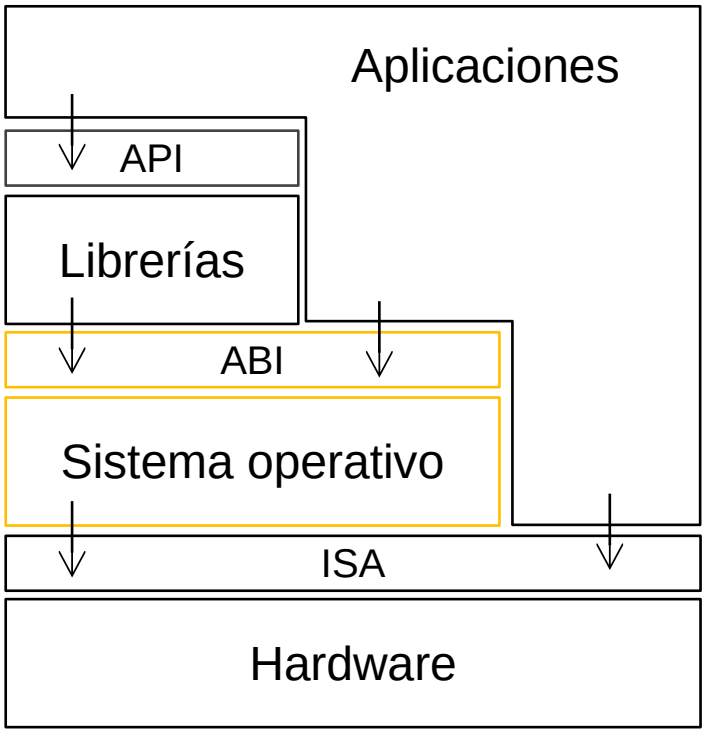


Virtualización de la ejecución

Niveles de virtualización

- Según la capa que se virtualice
- Virtualización de hardware
- Virtualización de S.O.
  - Contenedores

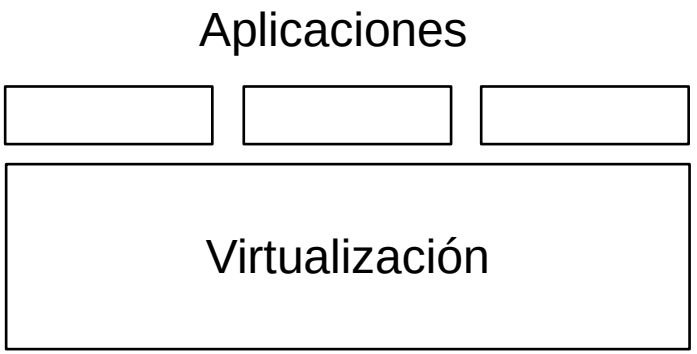


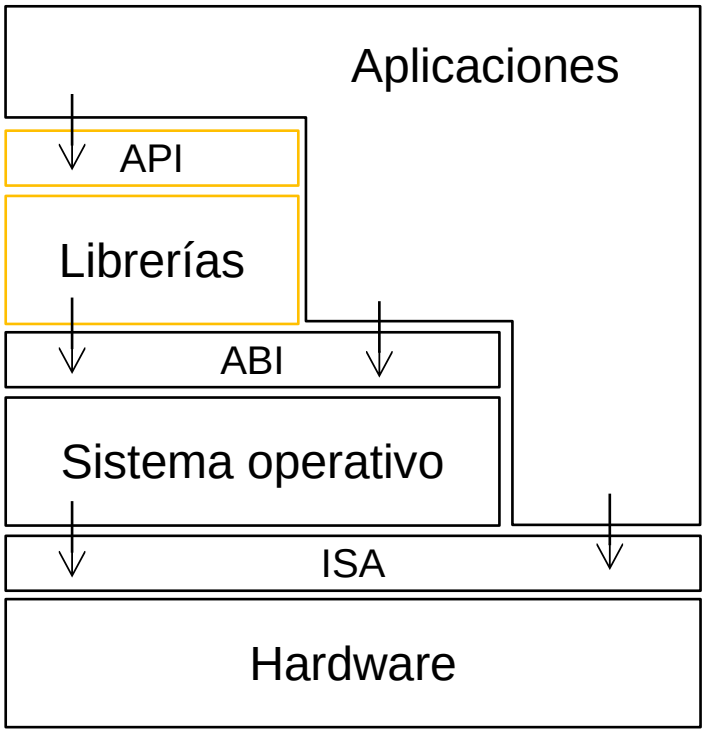


Virtualización de la ejecución

Niveles de virtualización

- Según la capa que se virtualice
- Virtualización de hardware
- Virtualización de S.O.
- Virtualización de app
  - JVM, Wine, etc.

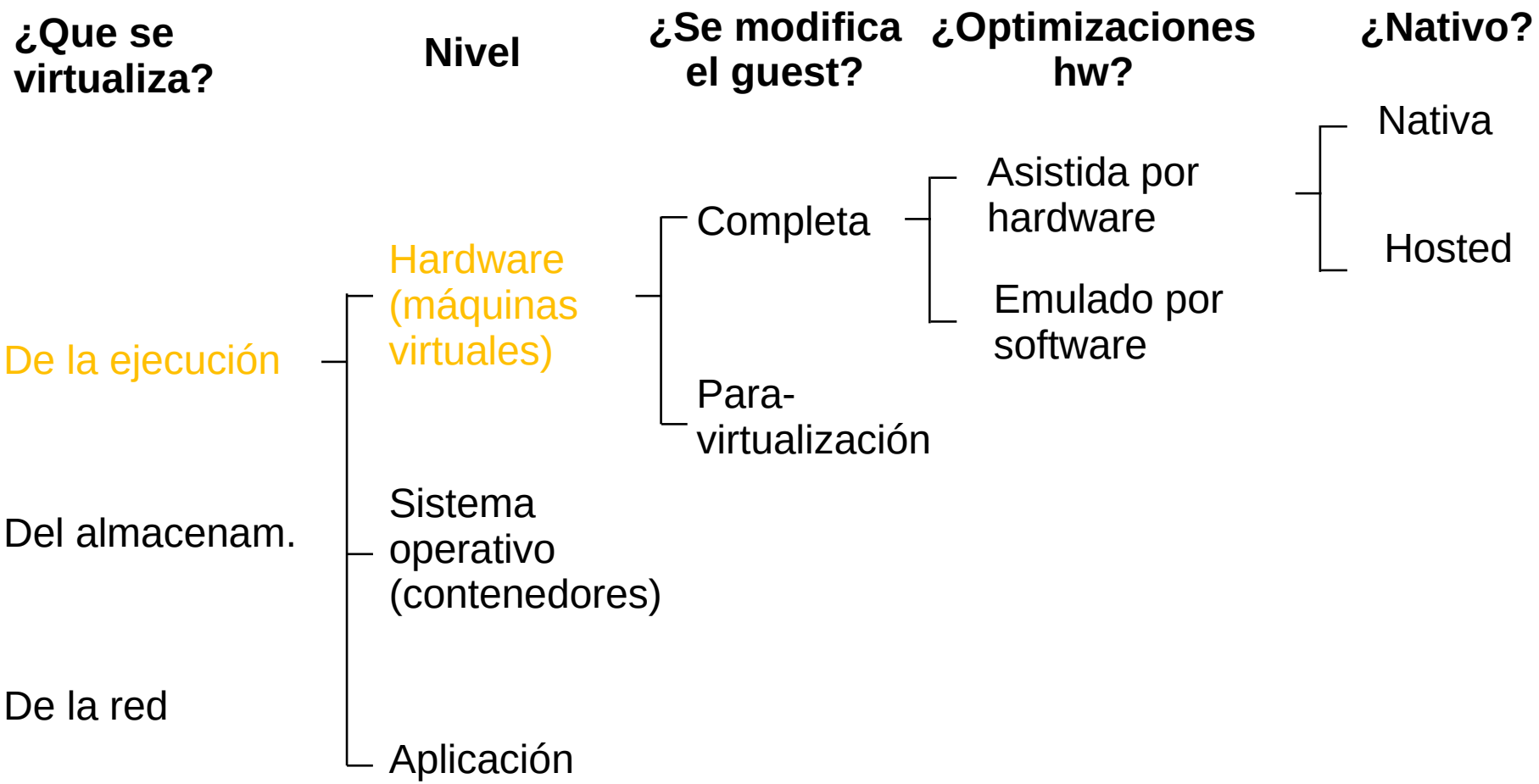




Virtualización hardware

Virtualización hardware

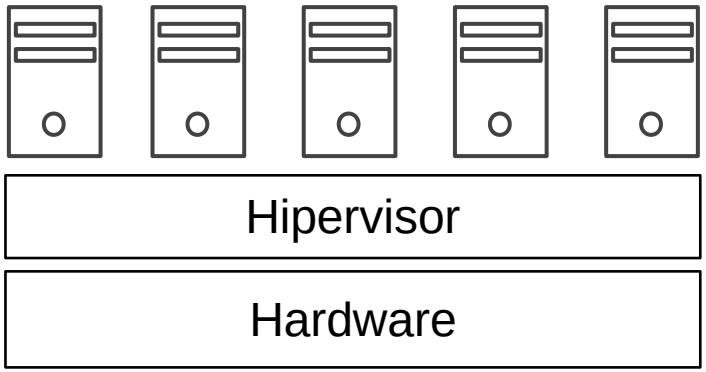
- No completa



Virtualización hardware

¿Qué es?

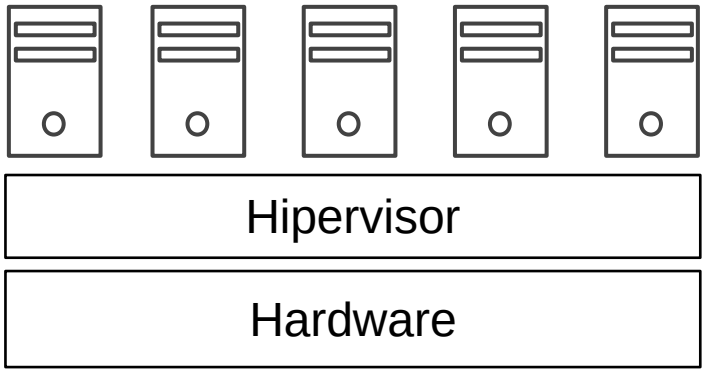
- Crea un entorno de ejecución que simula el hardware de un computador
- Hipervisor o *Virtual Machine Manager* (VMM)
- Las máquinas virtuales se ejecutan sobre VMM
- Las vm contienen el S.O., librerías y aplicaciones



Virtualización hardware

Hipervisor > Condiciones de corrección

- La ejecución del guest sobre el hipervisor debe ser equivalente a su ejecución sobre el host
- Penalización mínima de rendimiento
- El hipervisor tiene control de los recursos



Virtualización hardware

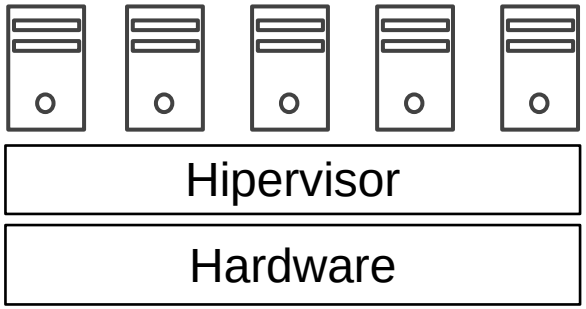
Hipervisor > Tipos

- Hipervisor de tipo I (nativo)
- Hipervisor de tipo II (hosted)

Virtualización hardware

Hipervisor > Tipos

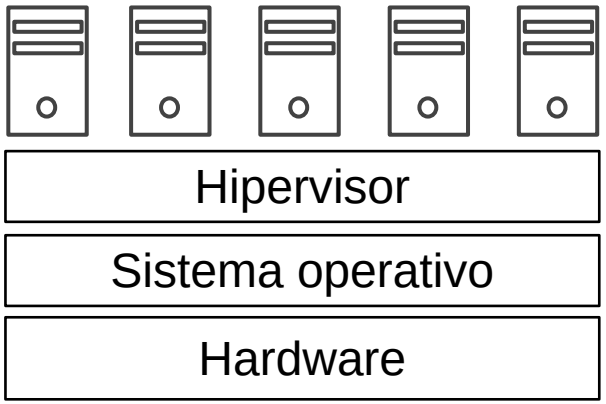
- Hipervisor de tipo I (nativo)
  - Se ejecuta directamente sobre el hardware
  - Control total, más seguro y eficiente, pero más difícil de configurar (soporte del hardware)
  - Xen, VMWare ESX Server, MS Hyper-V



Virtualización hardware

Hipervisor > Tipos

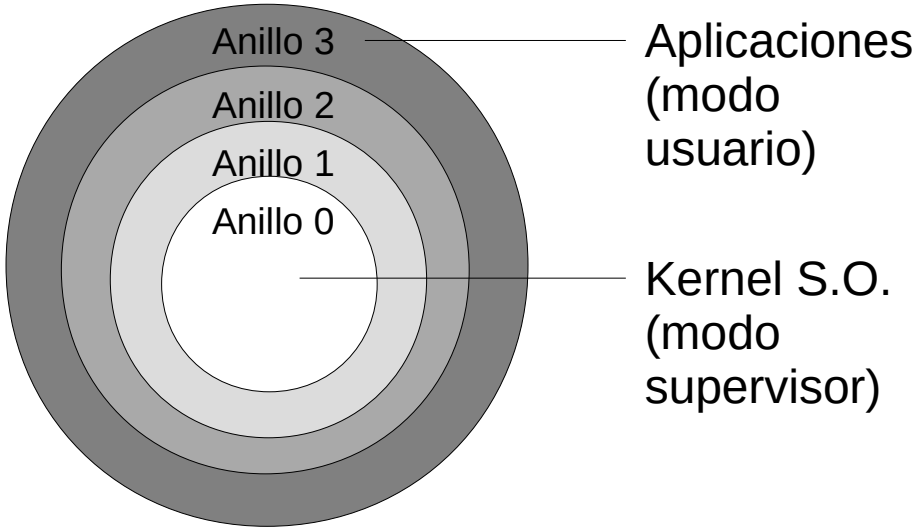
- Hipervisor de tipo II (hosted)
  - Se ejecuta en un sistema operativo, como proceso
  - Menos control, eficiencia y seguridad pero fácil configuración (el s.o. gestiona el hw)
  - WMWare Workstation, VirtualBox, Ms VirtualPC, ...



Virtualización hardware

Anillos de protección

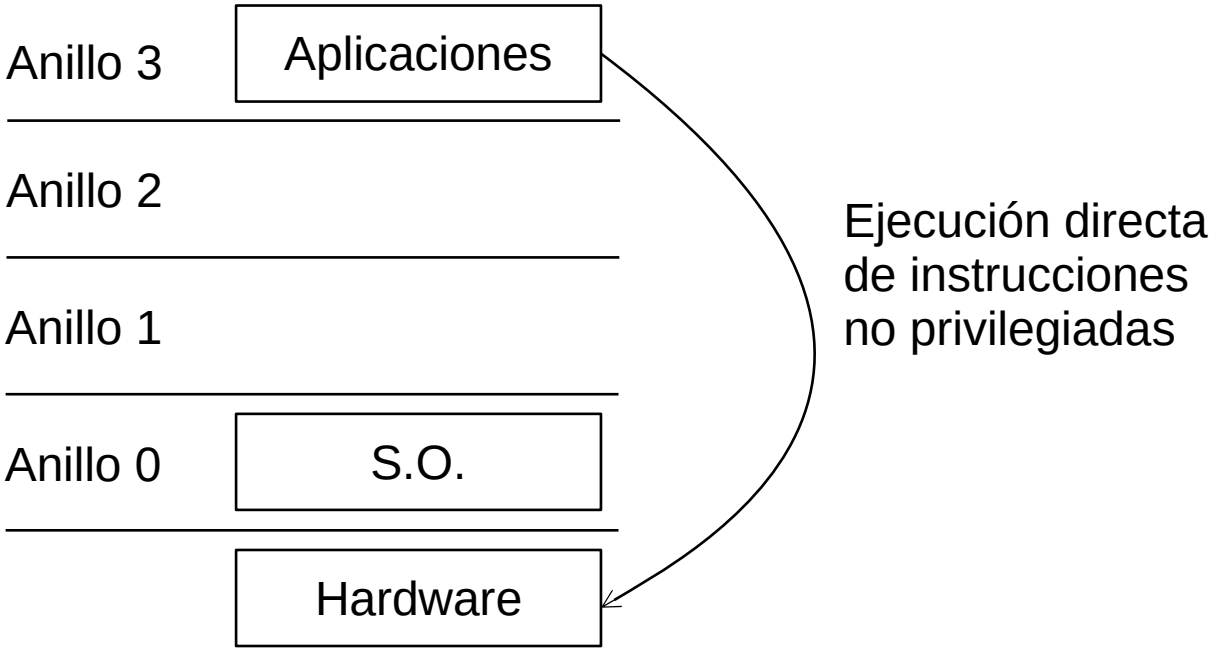
- Mecanismo de seguridad implementado por sistema operativo y hardware
- El código se ejecuta en anillos concéntricos
- Cada anillo representa un nivel de privilegios



Virtualización hardware

Anillos de protección

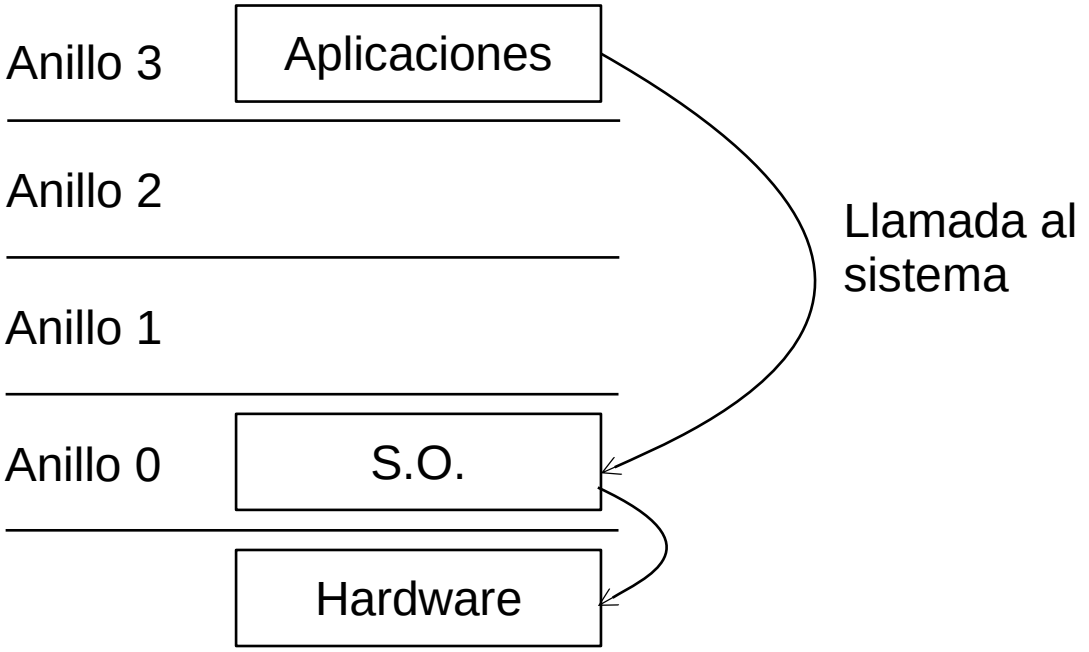
- Las aplicaciones se ejecutan en modo usuario



Virtualización hardware

Anillos de protección

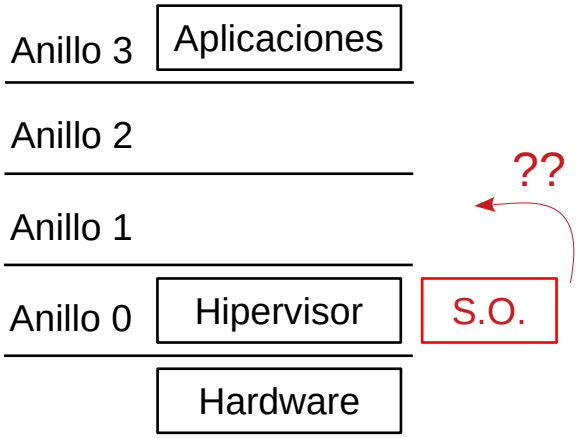
- Las aplicaciones se ejecutan en modo usuario
- Cuando ejecutan instrucciones privilegiadas (entrada/salida) efectúan llamadas al sistema



Virtualización hardware

Anillos de protección

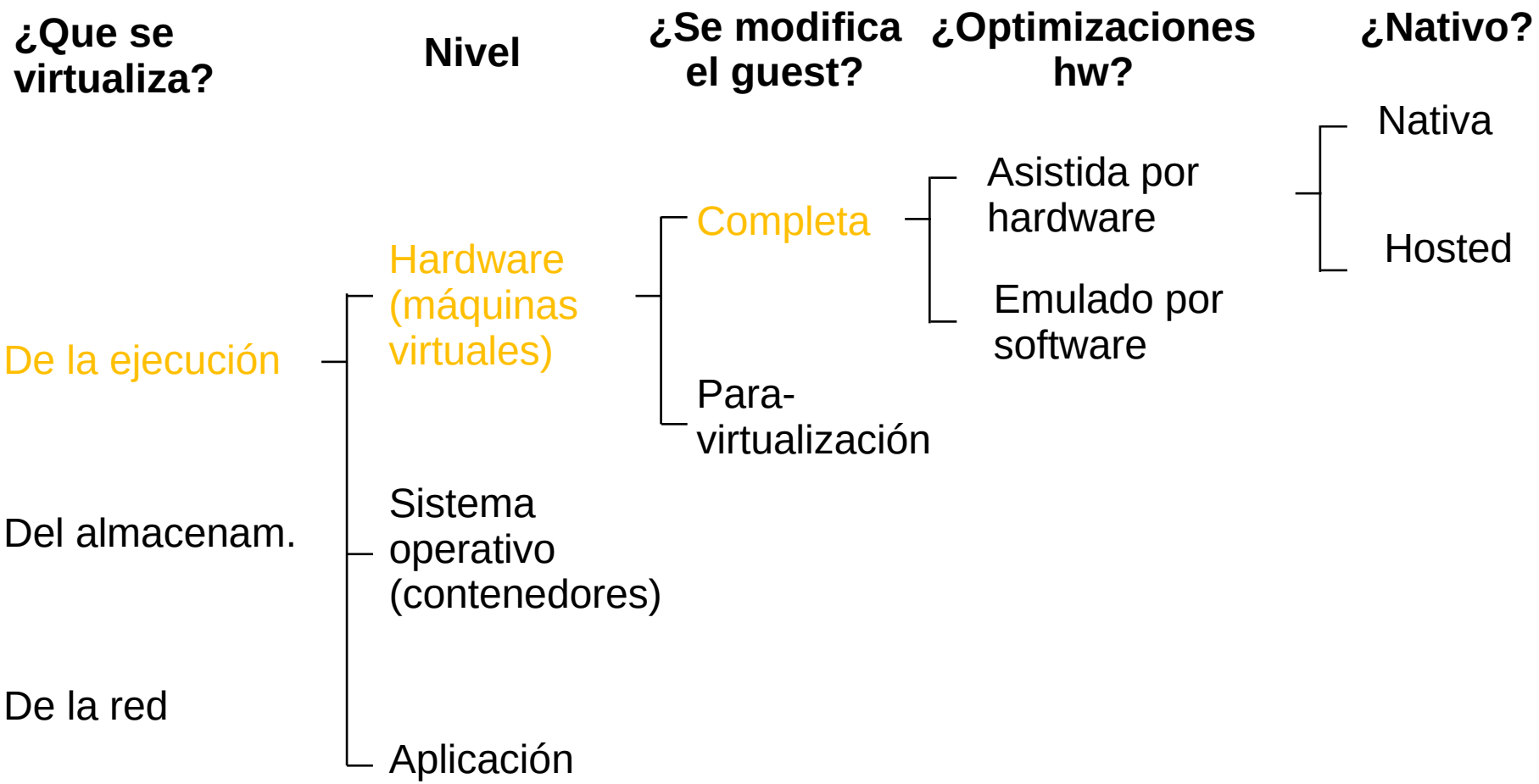
- El hipervisor debe ejecutarse en el anillo 0
- El S.O. guest se ubica en un nivel superior, con menos privilegios
- Tres estrategias:
  - Virtualización completa
  - Paravirtualización
  - Virtualización asistida por hardware



Virtualización hardware

Virtualización completa

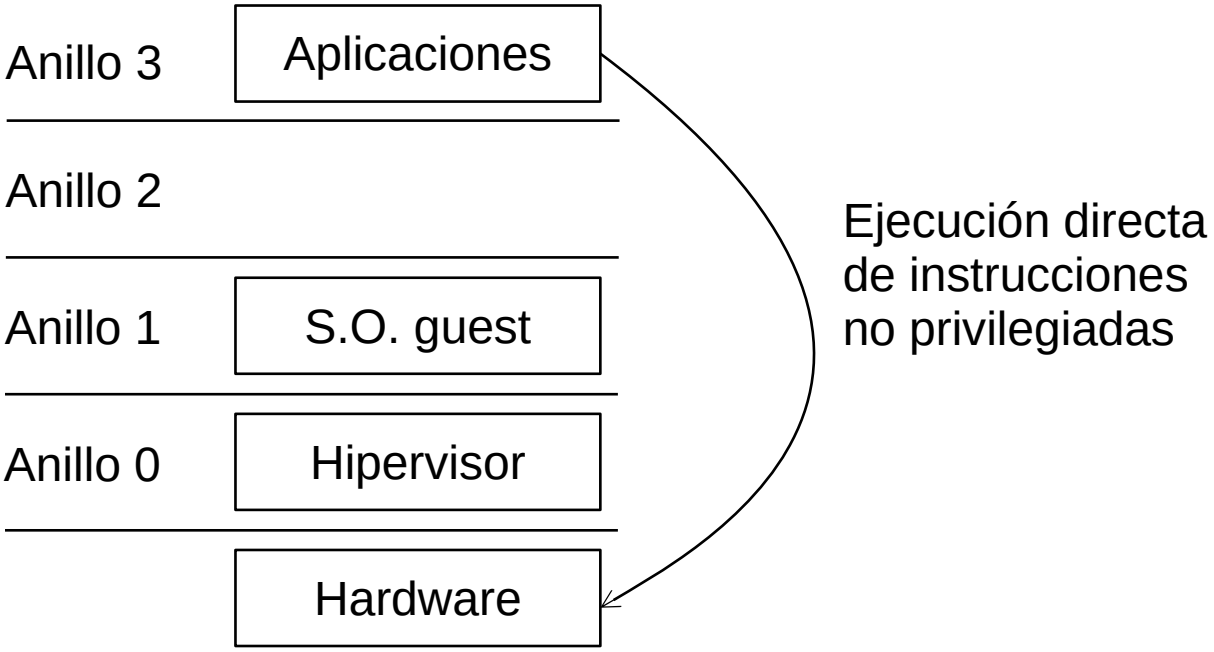
- No completa



Virtualización hardware

Virtualización completa

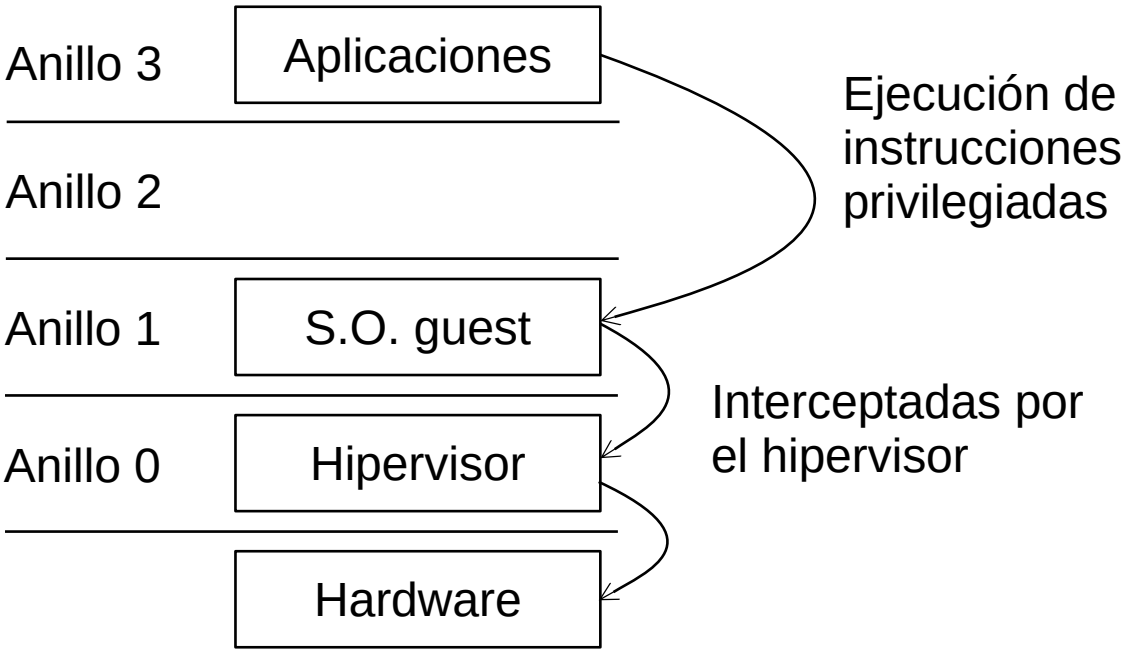
- El SO guest se ejecuta sin modificación en anillo 1



Virtualización hardware

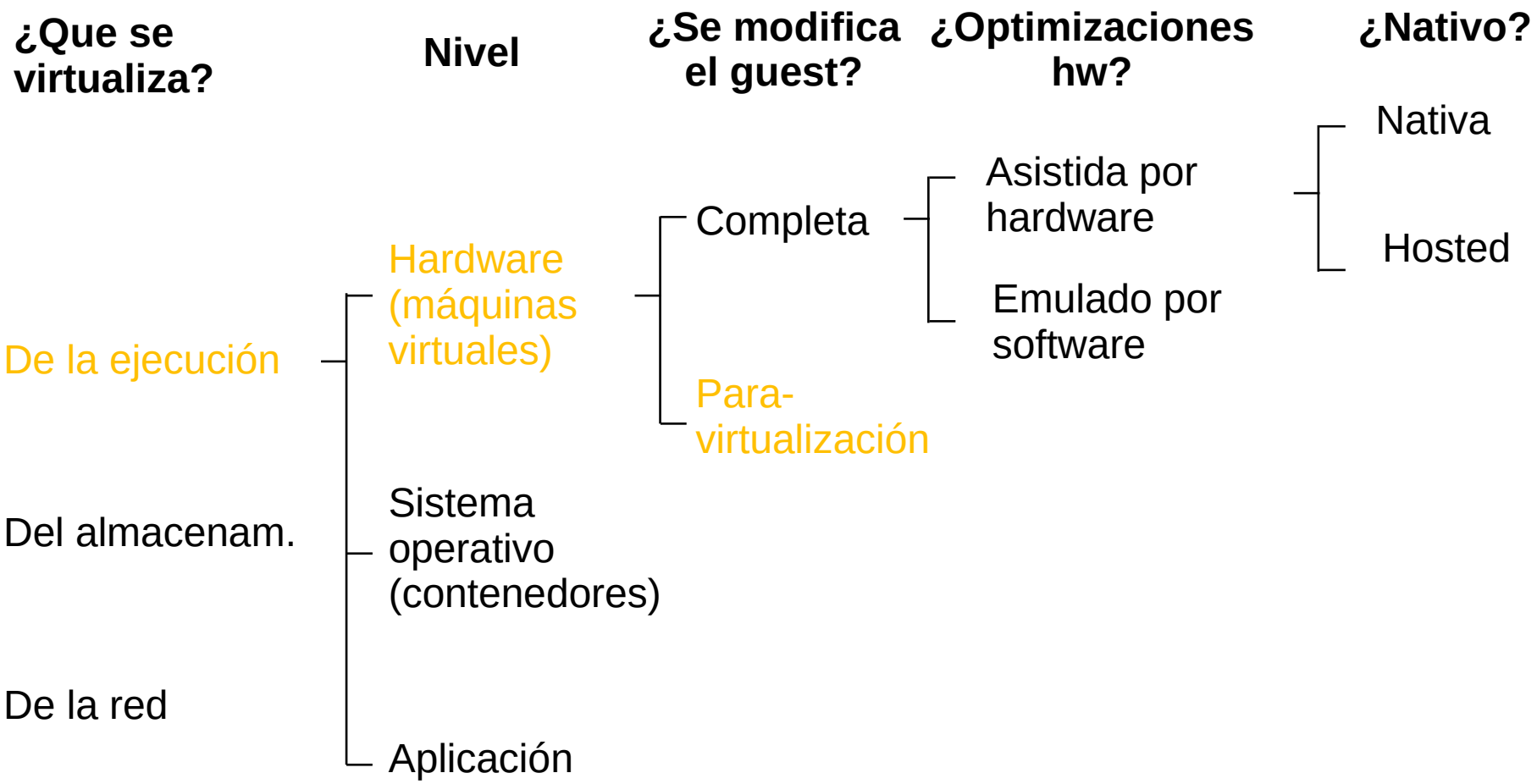
Virtualización completa

- El SO guest se ejecuta sin modificación en anillo 1
- Intercepta instrucciones privilegiadas y traduce
- Costoso. QEMU, VirtualBox, VirtualPC.



Virtualización hardware

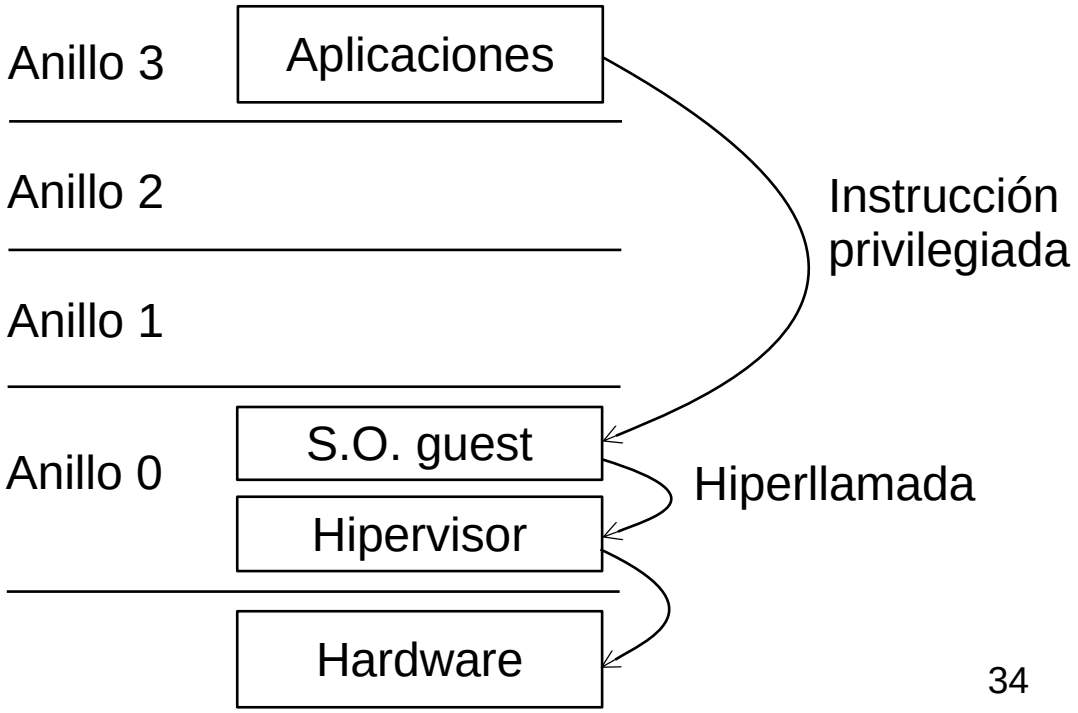
Paravirtualización



Virtualización hardware

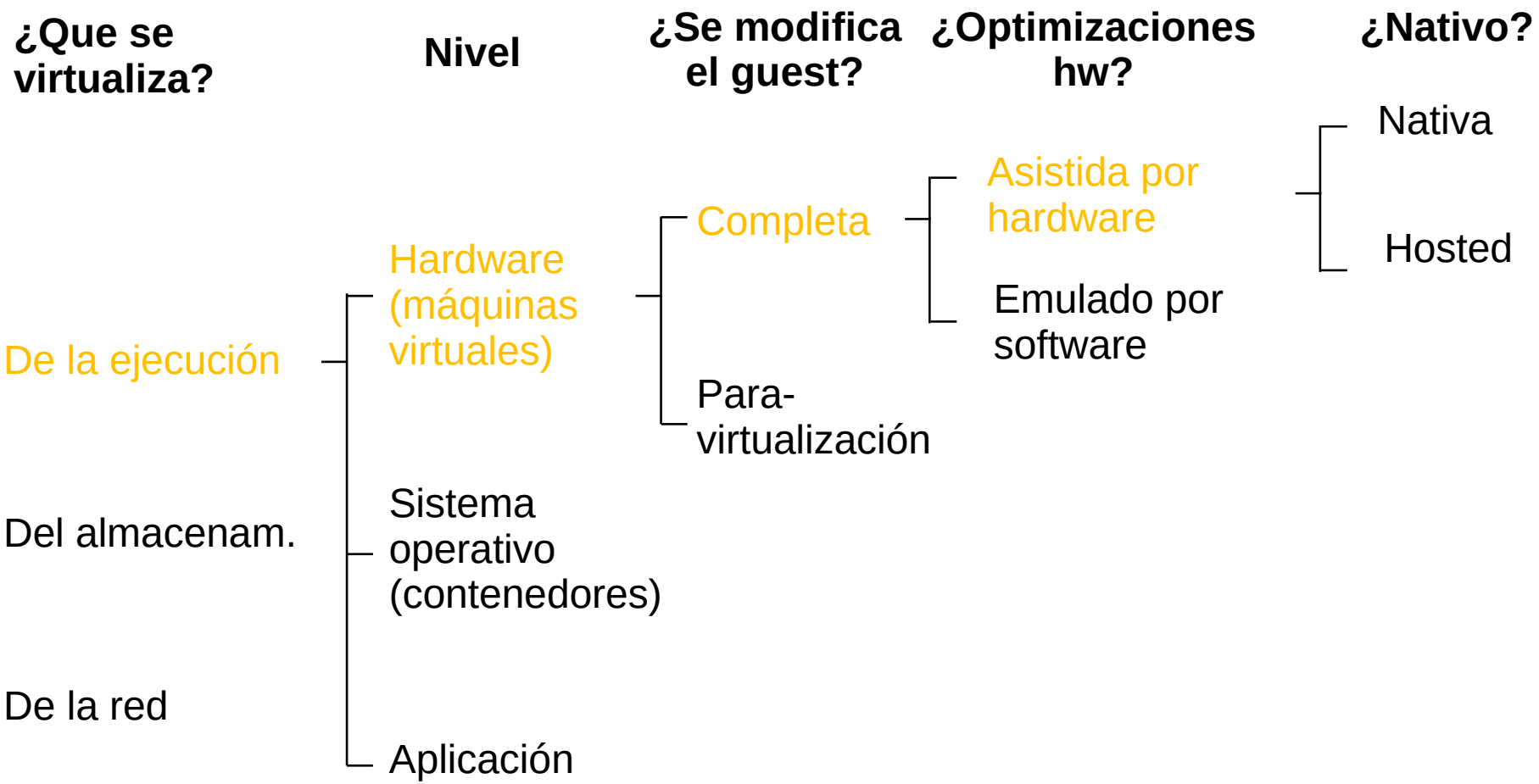
Paravirtualización

- El SO guest se ejecuta modificado en anillo 0
- Se comunica con hipervisor usando hiperllamadas
- Muy eficiente. Xen, Hyper-V



Virtualización hardware

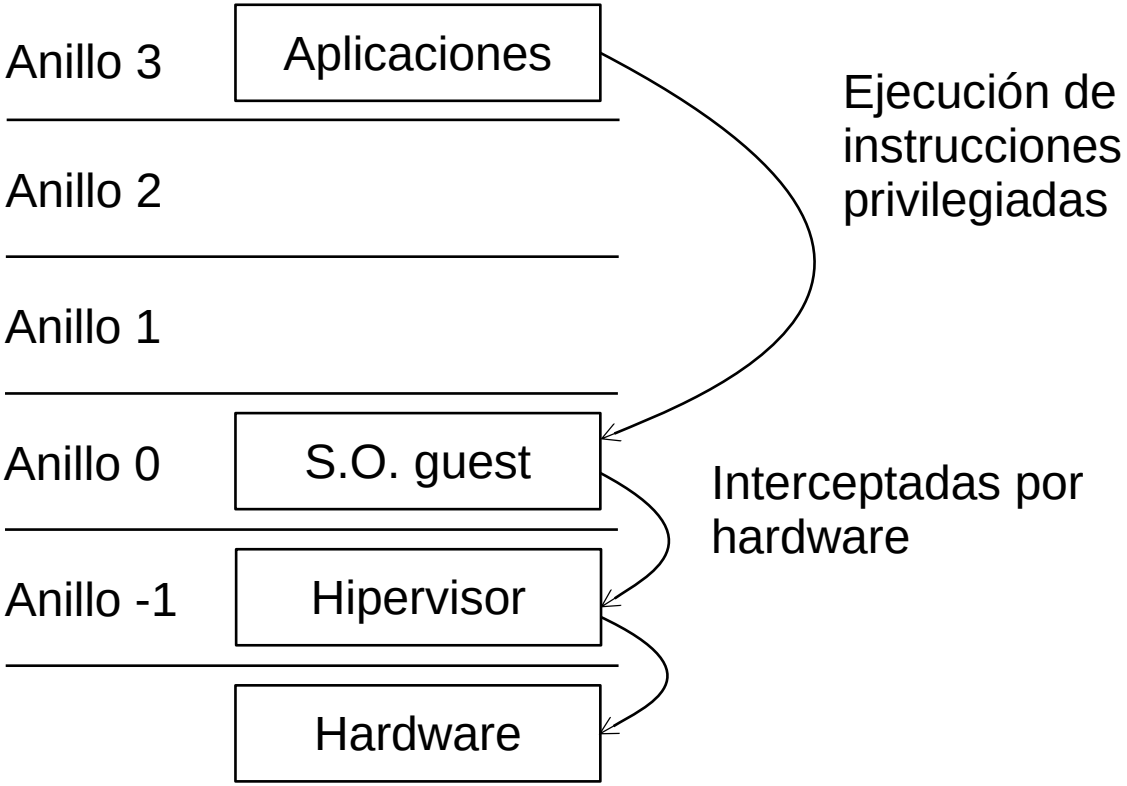
Virtualización asistida por hardware



Virtualización hardware

Virtualización asistida por hardware

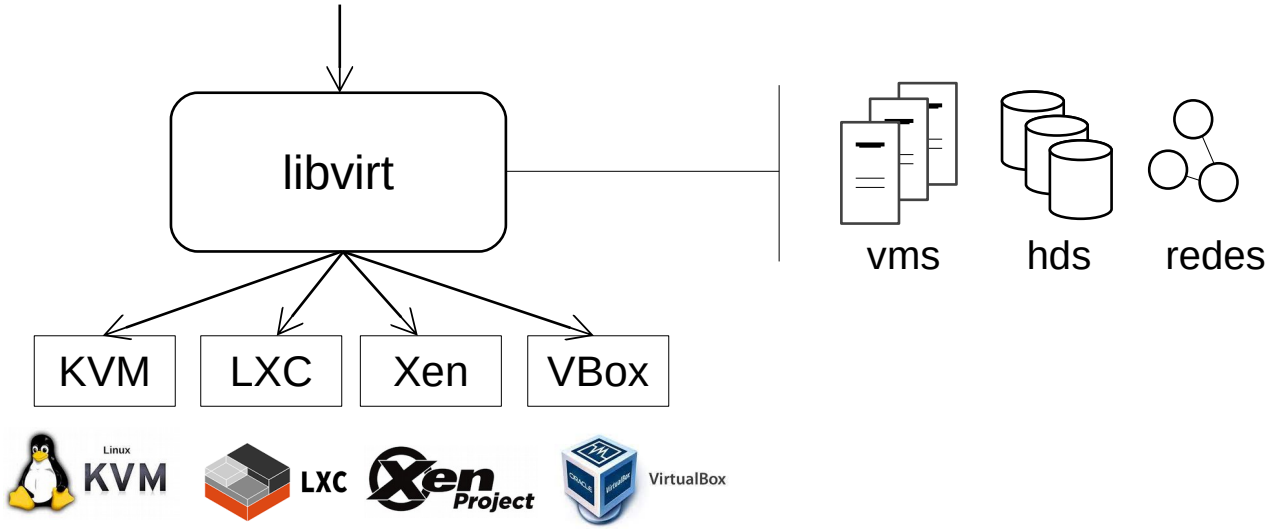
- S.O. guest en anillo 0, hipervisor en anillo -1
- El hardware (Intel VT-x,AMD-V) intercepta y redirige
- Muy eficiente



libvirt

¿Qué es?

- Middleware que gestiona múltiples hipervisores a través de una API común (vms, vols, nets, ...)
- Los drivers traducen a los hipervisores
- Se utiliza como base para otras soluciones (cloud)

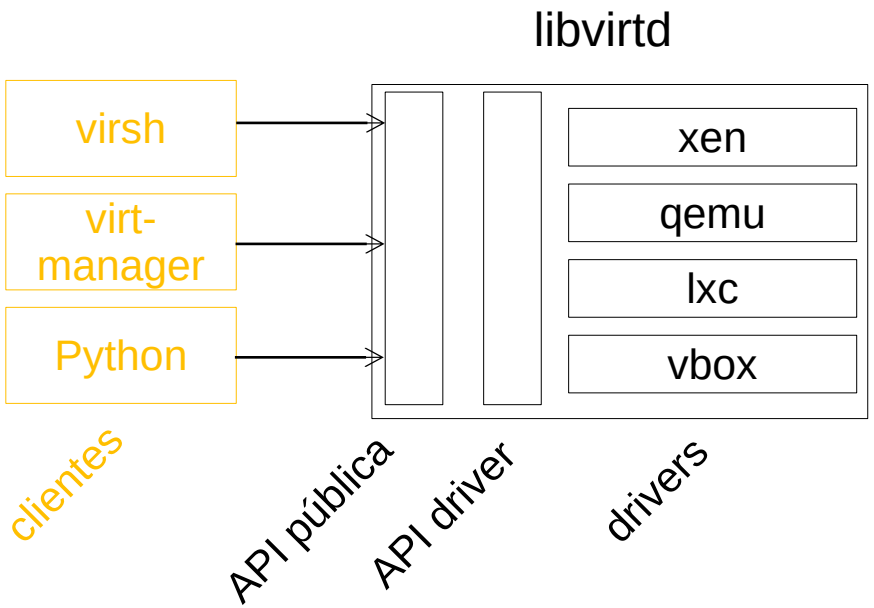


libvirt

Arquitectura

Clientes

- virsh: línea de commandos CLI
- virt-manager: GUI
- Librerías (bindings)

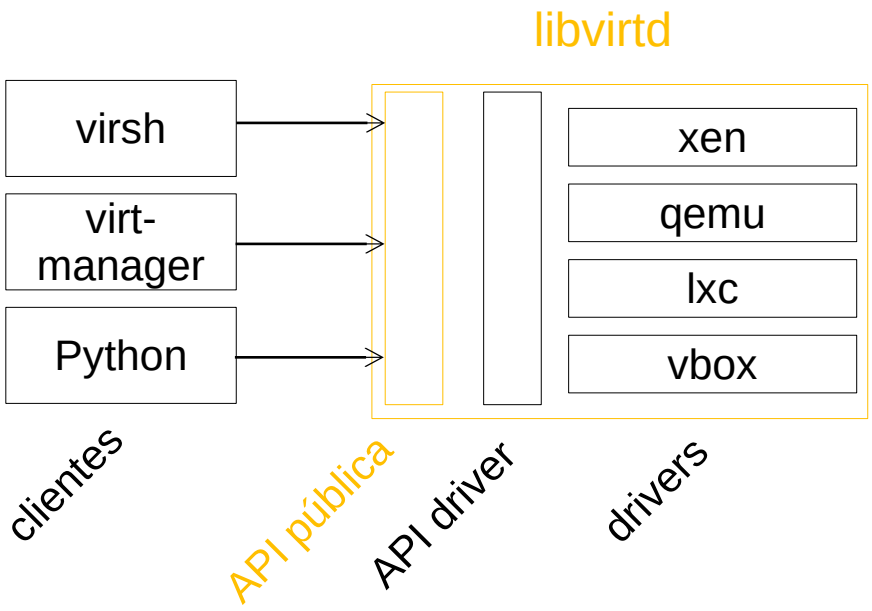


libvirt

Arquitectura

Demonio *libvirtd*

- Publica una API a través de Unix/TCP sockets
- Funciona en dos modos system/session
- Se comunica con los drivers usando una API

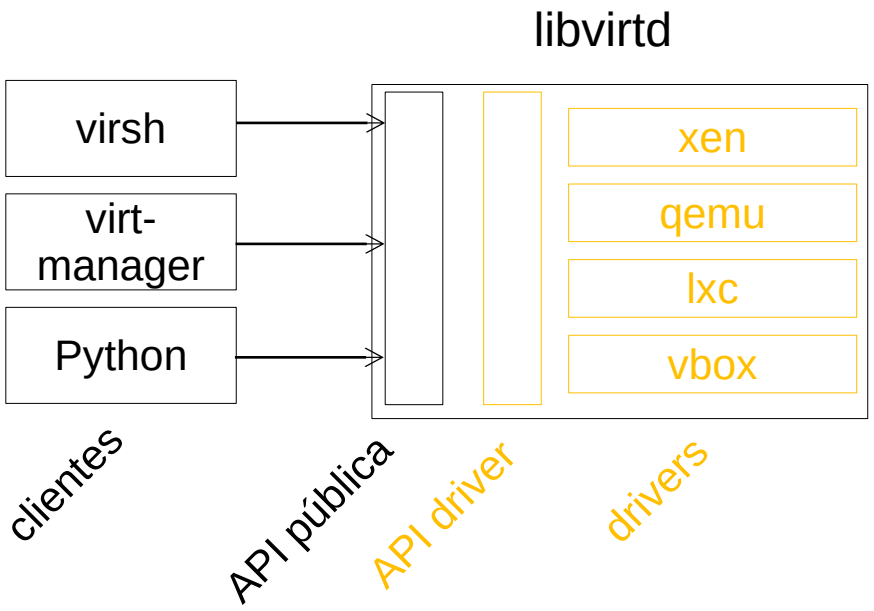


libvirt

Arquitectura

Drivers

- Cada driver traduce a un hipervisor diferente
- Al arrancar *libvirtd* se registran todos los drivers

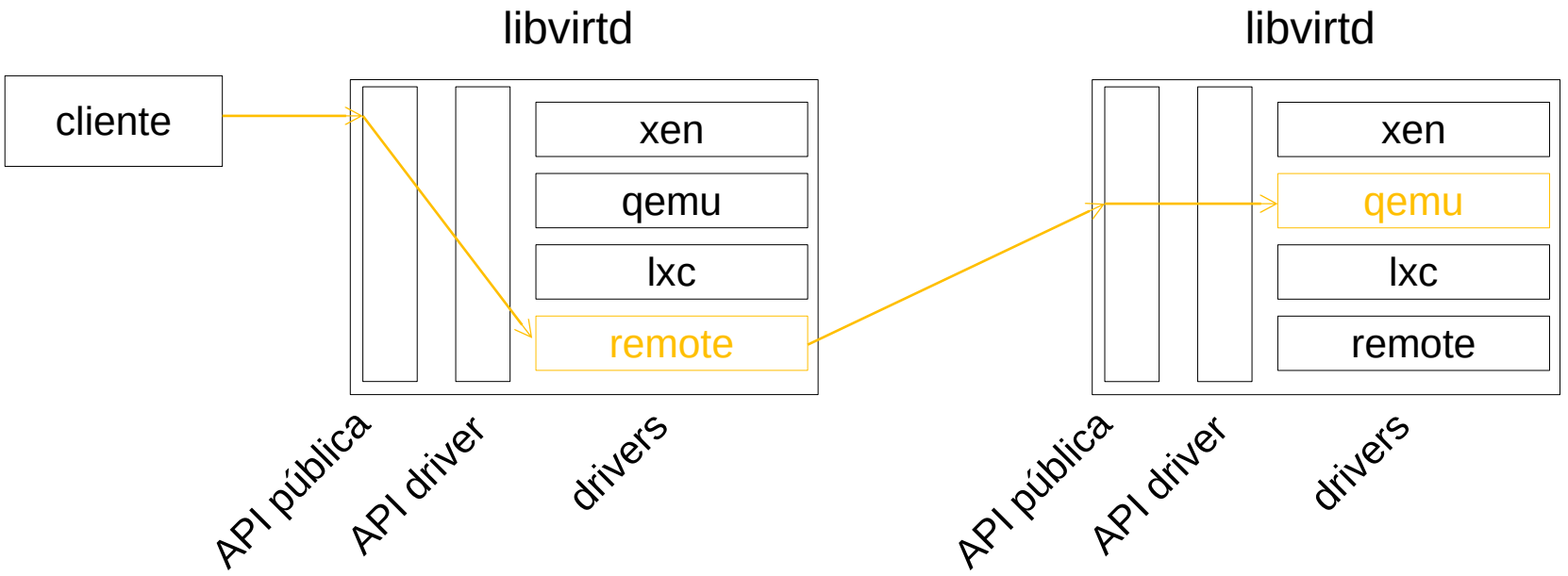


libvirt

Arquitectura

Drivers

- Si la comunicación es remota, el demonio *libvirtd* local se comunica con el demonio *libvirtd* remoto a través del driver remote



libvirt

Instalación

- Demonio *libvirtd* sólo en Linux

```
> sudo apt install libvirt-daemon-system libvirt-clients
```

- Cliente gráfico *virt-manager*

```
> sudo apt install virt-manager
```

- Binding Python

```
> sudo apt install python3-libvirt
```

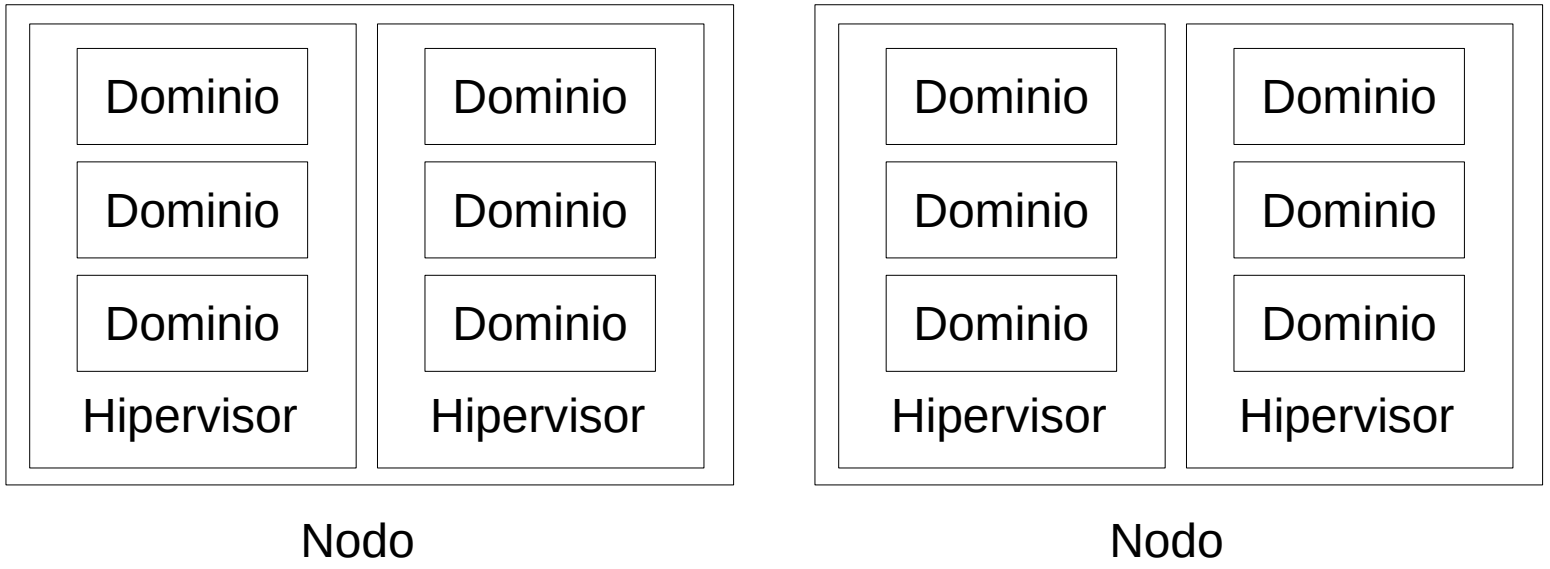
- Añadir usuario al grupo libvirt (ó libvirtd)

```
> sudo usermod -aG libvirt <usuario>
```

libvirt

Modelo libvirt

- Cada máquina física es un nodo
- Cada nodo contiene múltiples hipervisores
- Un hipervisor permite crear múltiples dominios



libvirt

Modelo libvirt > Conexión

- Primero se crea una conexión contra libvirt
- Se proporciona una URI
  - `driver[+unix]:///[system|session]`
  - `driver[+transport]://[username@][hostname][:port]/[path][?params] // remota`
  - `qemu:///system`
  - `vbox:///session`
  - `qemu://node.example.com/system`
  - `xen+ssh://root@node.example.com/`

libvirt

Modelo libvirt > Conexión

- virsh: `virsh --connect=qemu:///system list --all`
- Python:

```
import libvirt
import sys
con = None
try:
  con = libvirt.open("qemu:///system")
except libvirt.libvirtError as e:
  print(repr(e), file=sys.stderr)
  exit(1)
con.close()
exit(0)
```

libvirt

Modelo libvirt > Capacidades

- virsh: `virsh capabilities`
- Python:

```
host = con.getHostname()
print('Hostname:'+host)
nodeinfo = con.getInfo()
print('Model: '+str(nodeinfo[0]))
print('Memory size: '+str(nodeinfo[1])+'MB')
print('Number of CPUs: '+str(nodeinfo[2]))
print('MHz of CPUs: '+str(nodeinfo[3]))
print('Number of NUMA nodes: '+str(nodeinfo[4]))
print('Number of CPU sockets: '+str(nodeinfo[5]))
print('Number of CPU cores per socket: '+str(nodeinfo[6]))
print('Number of CPU threads per core: '+str(nodeinfo[7])
```

libvirt

Modelo libvirt > Dominios

- Máquina virtual con identificadores: ID, name, UUID
- Volátiles (transient) vs persistentes (persistent)
- Listar dominios

```
virsh list --all   |  virsh dominfo <dominio>
```

```
doms = con.listAllDomains()
doms = con.listDefinedDomains()
ids = con.listDomainsID()
dom = con.lookupByID()
dom = con.lookupByName()
dom = con.lookupByUUID()
```

```
dom.ID()
dom.UUIDString()
dom.OSType()
dom.info()
dom.state()
...
```

libvirt

Modelo libvirt > Dominios

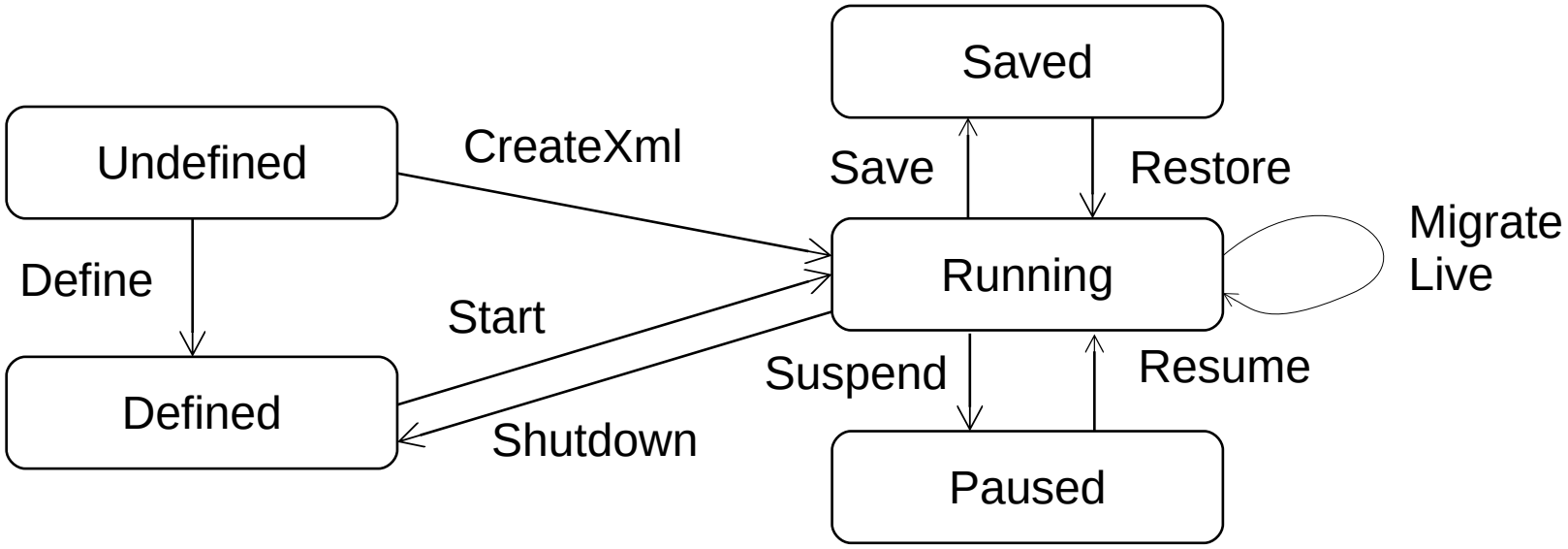
- Crear dominio
  - A partir de XML
  - `virsh dumpxml <domain> > <file.xml>`
  - `virsh create <file.xml>|virsh define <file.xml>`

```
dom = con.lookupByID(5)
xml = dom.XMLDesc(0)
dom = con.createXML(xml)
dom = con.defineXML(xml)
```

libvirt

Modelo libvirt > Dominios

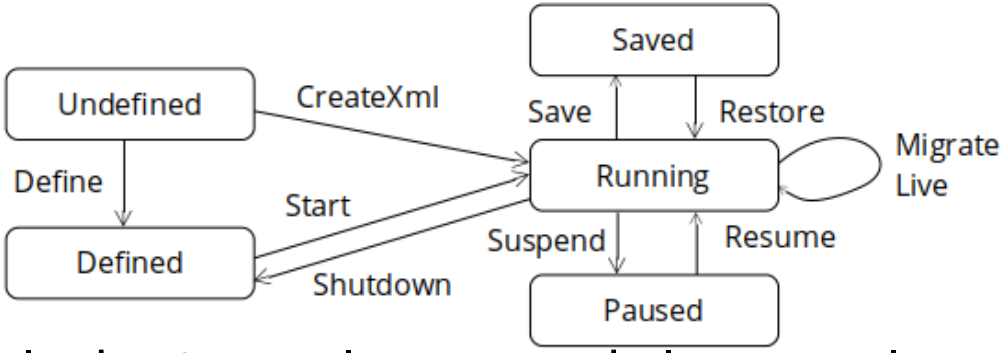
- Ciclo de vida



libvirt

Modelo libvirt > Dominios

- Ciclo de vida
  - virsh

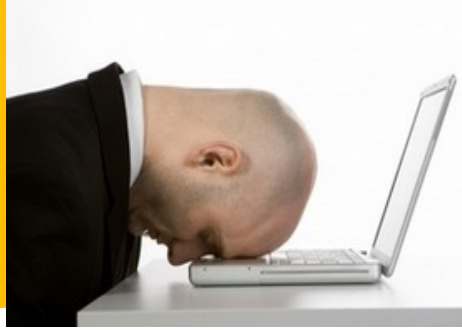



```
virsh [ start | shutdown | destroy | suspend | save | resume | restore ]
```

  - Python

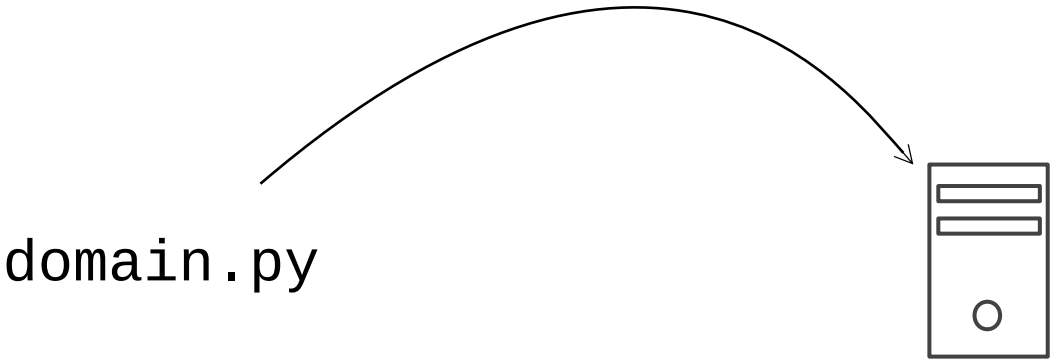
```
dom = con.lookupByID(5)
dom.create(), dom.shutdown(), dom.destroy()
dom.suspend(), dom.save(), dom.resume(), dom.restore()
```

libvirt



Ejercicio 2

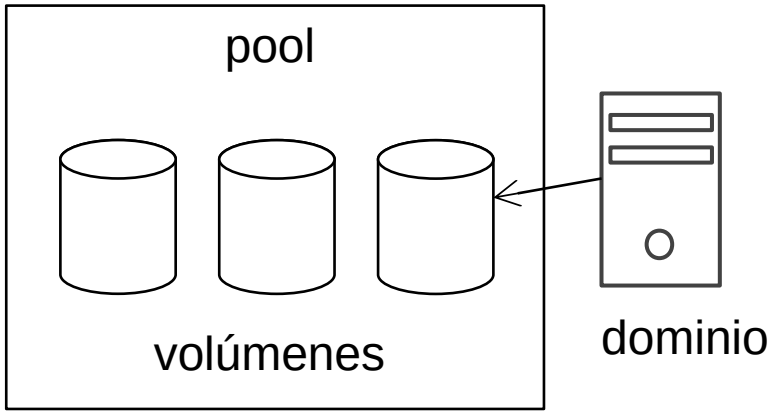
- CLI Python que gestione dominios
  - add
  - remove
  - start
  - stop
  - list



libvirt

Modelo libvirt > Pools

- Espacio de almacenamiento gestionado por sysadmin
- Directorio, share NFS, dispositivo iSCSI, etc.
- Contiene múltiples volúmenes
- Volumen es usado por dominio como disp. bloques



libvirt

Modelo libvirt > Pools

- virsh

```
    virsh [ pool-list | pool-info | pool-create | pool-define
| pool-start | pool-destroy | pool-undefine | … ]

    virsh [ vol-list | vol-info | vol-create | vol-delete |
… ]
```

- Python

```
con.listAllStoragePools(),con.storagePoolLookupByName()
con.storagePool[CreateXML()|DefineXML()]
pool.start(), pool.destroy(), pool.undefine()
pool.listVolumes(), pool.storageVolLookupByName(),
pool.createXML(), vol.delete(), ...
```

libvirt

Modelo libvirt > Pools

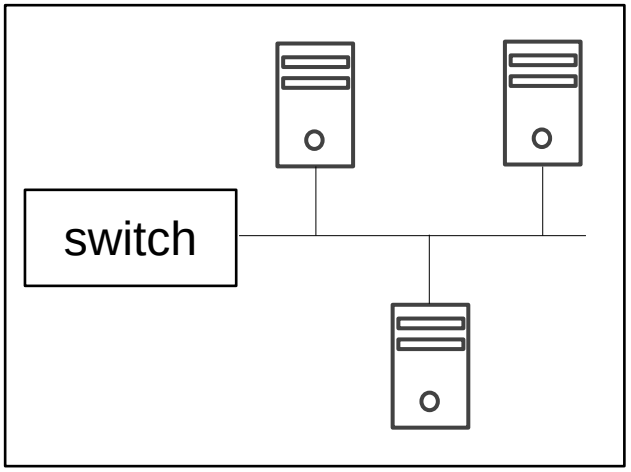
- Ejemplo

```
pools = conn.listAllStoragePools(0)
for pool in pools:
  info = pool.info()
  print('Pool: '+pool.name())
  print('  Num volumes: '+str(pool.numOfVolumes()))
  print('  Pool state: '+str(info[0]))
  print('  Capacity: '+str(info[1]))
  print('  Allocation: '+str(info[2]))
  print('  Available: '+str(info[3]))
```

libvirt

Modelo libvirt > Redes

- Los dominios se conectan a redes virtuales
- Una red virtual se crea por medio de un switch virtual
- En Linux, se utiliza el bridge de Linux



libvirt

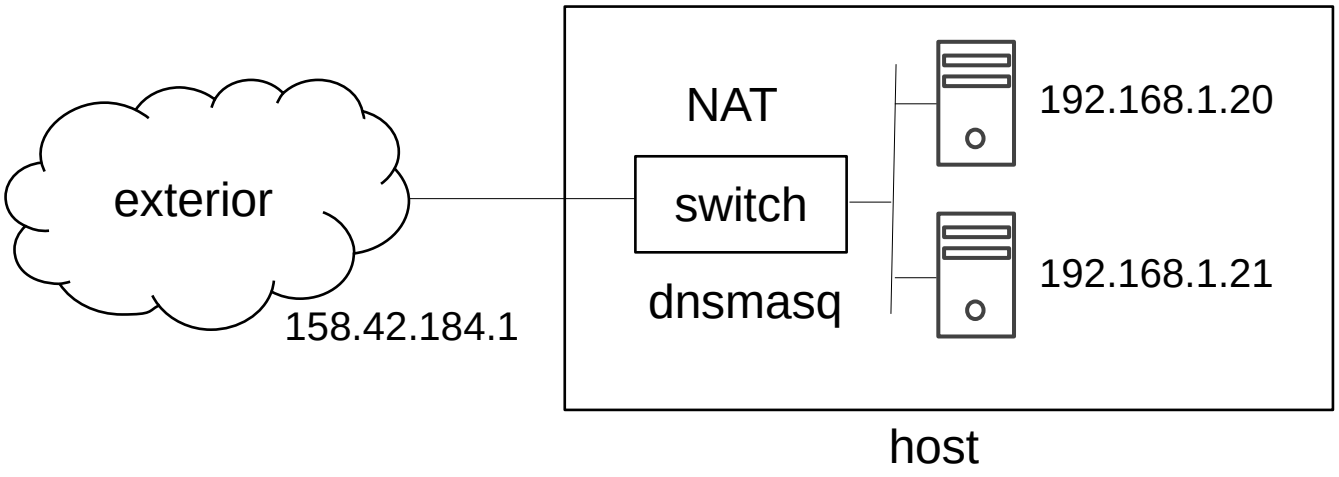
Modelo libvirt > Redes

- 3 modos de funcionamiento del switch virtual
- Modo NAT
- Modo routed
- Modo isolated

libvirt

Modelo libvirt > Redes

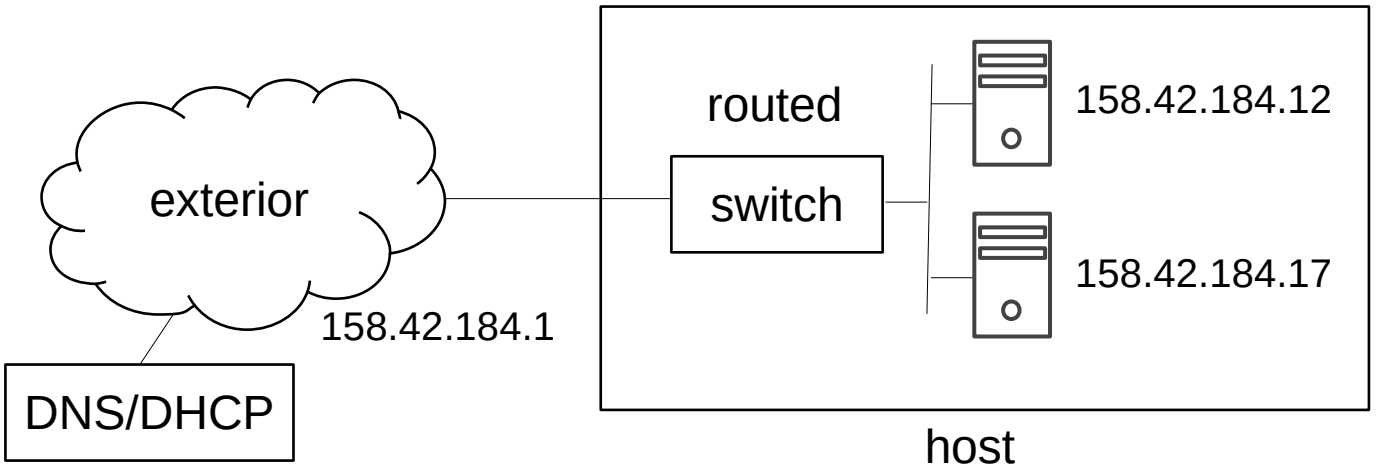
- 3 modos de funcionamiento del switch virtual
- Modo NAT
  - Dominios usan IP del nodo hacia el exterior (no visibles)
  - Traducción de direcciones (iptables). Switch actúa de DNS/ DHCP (dnsmasq)



libvirt

Modelo libvirt > Redes

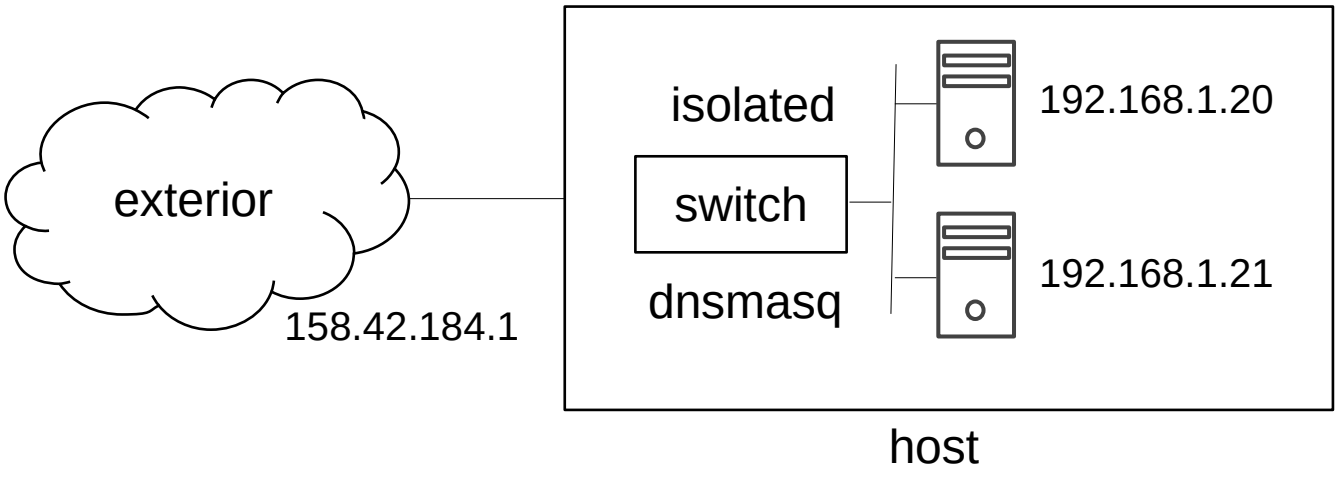
- 3 modos de funcionamiento del switch virtual
- Modo routed
  - Switch se conecta a la red del host, enruta directamente
  - Dominios forman parte de la subred enrutada por switch



libvirt

Modelo libvirt > Redes

- 3 modos de funcionamiento del switch virtual
- Modo isolated
  - Dominios sólo se comunican entre sí y con el host (no con exterior)
  - Switch actúa de DNS/DHCP



libvirt

Modelo libvirt > Redes

- Una red posee nombre y UUID
- Puede ser volátil o persistente (fichero XML)

libvirt

Modelo libvirt > Redes

- virsh

```
virsh [ net-list | net-info | net-create | net-define | net-start | net-destroy | net-undefine | … ]
```

- Python

```
con.listNetworks(),con.networkLookupByName()
con.network[CreateXML()|DefineXML()]
net.start(), net.destroy(), net.undefine(), ...
```


TO BE
CONTINUED...