

# Gestió d'Infraestructures per al Processament de Dades

## Còmput

Remo Suppi.

Departament Arquitectura d'Ordinadors i Sistemes Operatius

UAB (Remo.Suppi@uab.cat)

**Què  
veurem?**

**Virtualització del processador i del sistema operatiu.**

**Hipervisors (KVM, HyperV, VWmare, VBox, ...),**

**Contenidors (Docker i LXC)**

**Serverless computing.**

# Virtualització

És el gran actor que permet que el cloud existeixi i sigui possible: es podran crear els recursos virtuals i l'usuari «veurà» (eficiència propera 92-96% a un dispositiu físic) la seva infraestructura (màquines, servidors, discos, xarxa) i serà només **la seva** infraestructura i el mateix els N clients de el cloud.

Amb això s'aprofitaran tots els recursos físics maximitzant l'eficiència i amb la consegüent reducció del costos.

Pel que fa a l'estalvi de recursos, per què és millor virtualitzat que físic? Un usuari= 1 Tbyte i 1000 usuaris=1 Pbyte. Valors reals es troba dins d'un 10-15%, si el recurs és virtualitzat i la provisió és dinàmica, el proveïdor no ha de disposar posar de l'1 Pbyte.

Per això la virtualització és la tecnologia base (i essencial) per al **cloud computing** ja que abstruïa a l'usuari del que hi ha «sota»; de fet, l'usuari tampoc sap ni cal saber on estaran els recursos.

**Garanties:** prestacions i un servei -> SLA

**Preu:** acceptable en modalitats de pagament per ús.

<https://calculator.aws/#/>

<https://azure.microsoft.com/en-us/pricing/calculator/>

# Virtualització

Proveïdor serà un negoci amb un compte de resultats positiva, eficiència, reduirà costos que podrà traslladar als seus clients, reduirà espai i consum d'energia, mantindrà els recursos aïllats (privacitat), serà àgil en la gestió i aprovisionament i serà escalable (si «necessito més», puc «afegir més»).

## Ens permet:

- Incrementar la utilització dels recursos físics
- Consolidació de recursos i reducció de l'espai físic utilitzat amb estalvi energètic i major eficiència

Alta disponibilitat, escalabilitat, backups, agilitat en la gestió/administració, reducció de costos i millores del TCO (*total cost of ownership*) i el ROI (*return on investment*).

## Riscos:

- Limitacions del hardware: el proveïdor haurà de disposar del HW requerit i evitar la sobre-utilització. Donar compliment de la QoS acordat en la SLA),
- Plataforma de virtualització (sobre todo de tipo 1 bare-metal) que no son compatibles amb tot el hardware
- Control de fallides: un fallida del hardware del servidor físic (afecta a tots el servidor virtualitzats) s'haurà de respondre amb celeritat i disposar d'alta disponibilitat.
- Altres: personal no adient per a resoldre el problemes, seguretat, obsolescència, ...

# Virtualització

## Hypervisor

Monitor de màquina virtual, VMM, (virtual machine monitor) és la capa d'abstracció per a la virtualització que podrà actuar, conjuntament, com un nucli indivisible amb el sistema operatiu o separat d'ell com una aplicació més (però treballant conjuntament amb ell).

El hipervisor mostrarà a les màquines virtuals (guest) una infraestructura virtualitzada que tindrà la seva contrapartida en la màquina sobre la qual s'executa (Host).

### **Permetrà:**

que el guest «vegi» aquesta infraestructura com si de el maquinari físic es tractés  
gestionar les màquines virtuals, assignar els recursos, permetre l'accés a el maquinari equivalent,  
monitoritzar/comptabilitzar els recursos gastats per cada MV  
gestió i administració de sistema.

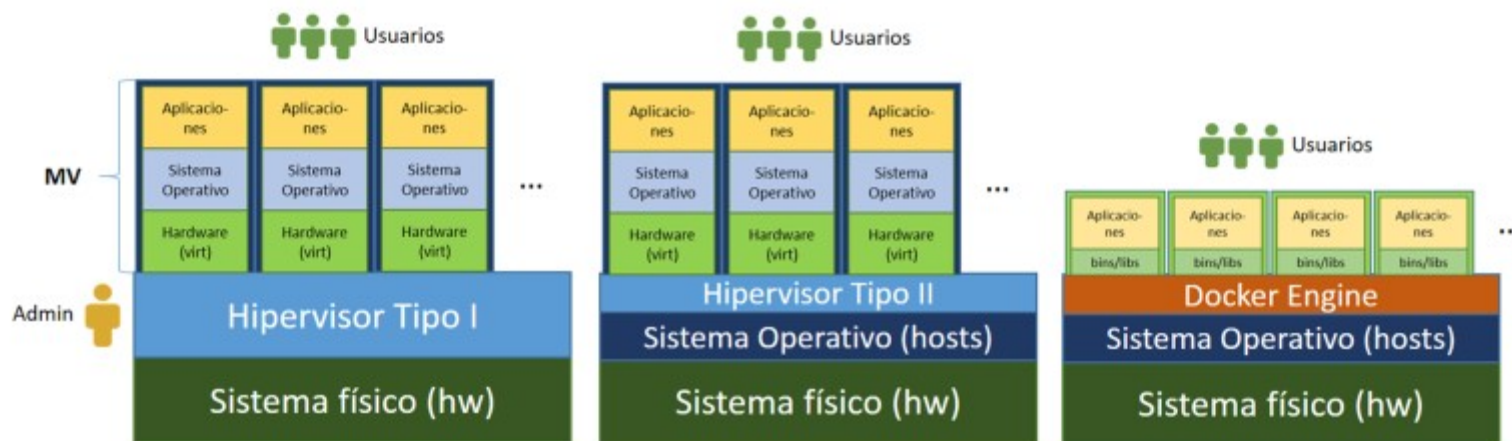
# Virtualització

**Hipervisors tipus 1** (també anomenats nadius, *unhosted* o *bare-metal*), per exemple VMware ESXi, Xenserver, on el hipervisor forma un conjunt indissoluble amb el SO

**Hipervisors tipus 2** (també anomenat *hosted*), per exemple, VMware Workstation, VirtualBox, Qemu, KVM (encara que alguns autors ho consideren de tipus 1), Hyper-V, on l'hipervisor s'executa com una capa que interacciona amb el SO, però no està inclòs en ell (excepte KVM).

**Es necessiten extensions hardware VT-x/AMD-V** (necessàries si les MV han de ser de 64 bits). En el tipus 2 el SO també haurà de ser de 64 bits.

**Contenidors:** virtualització del sistema operatiu.



# Hypervisors: **KVM** (<https://www.linux-kvm.org/>)

**Kernel-based Virtual Machine (KVM):** projecte de codi obert que implementa sobre Linux una infraestructura de virtualització permetent que Linux actuï com hipervisor a través d'un mòdul (kvm-intel/amd.ko) carregable en el nucli i eines en l'espai d'usuari per a la seva gestió.

Aquest hipervisor permet executar màquines virtuals a partir d'imatges de disc que contenen sistemes operatius sense modificar i cada màquina veurà el seu propi maquinari virtualitzat (targeta de xarxa, discs durs, targeta gràfica,..)

Necessita un processador x86/64, amb suport per a virtualització (VT-X / AMD-V) i pot executar diferents SO guest com Linux/Unix/ OSX/Windows, entre d'altres, tant de 32 com de 64 bits. A més, suporta un conjunt de dispositius paravirtualizados per a Linux, OpenBSD i FreeBSD, Windows, utilitzant l'API VirtIO.

Instal·lació sobre Ubuntu 20.04:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

```
sudo apt-get apt -y install qemu-kvm libvirt-daemon-system libvirt-daemon virtinst bridge-utils  
libosinfo-bin libguestfs-tools virt-top
```


```
modprobe vhost_net
```

# Hypervisors: **KVM** (<https://www.linux-kvm.org/>)

Configuració de la Xarxa (hi ha altres possibles):

```
sudo vi /etc/netplan/01-netcfg.yaml
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
      dhcp4: no
      # disable existing configuration for
ethernet
      #addresses: [10.0.0.30/24]
      #gateway4: 10.0.0.1
      #nameservers:
        #addresses: [10.0.0.10]
      dhcp6: no
```



```
# add configuration for bridge interface
bridges:
  br0:
    interfaces: [ens3]
    dhcp4: no
    addresses: [10.0.0.30/24]
    gateway4: 10.0.0.1
    nameservers:
      addresses: [10.0.0.10]
    parameters:
      stp: false
    dhcp6: no
```

```
root@server:~# reboot
root@dlp:~# ip addr
```

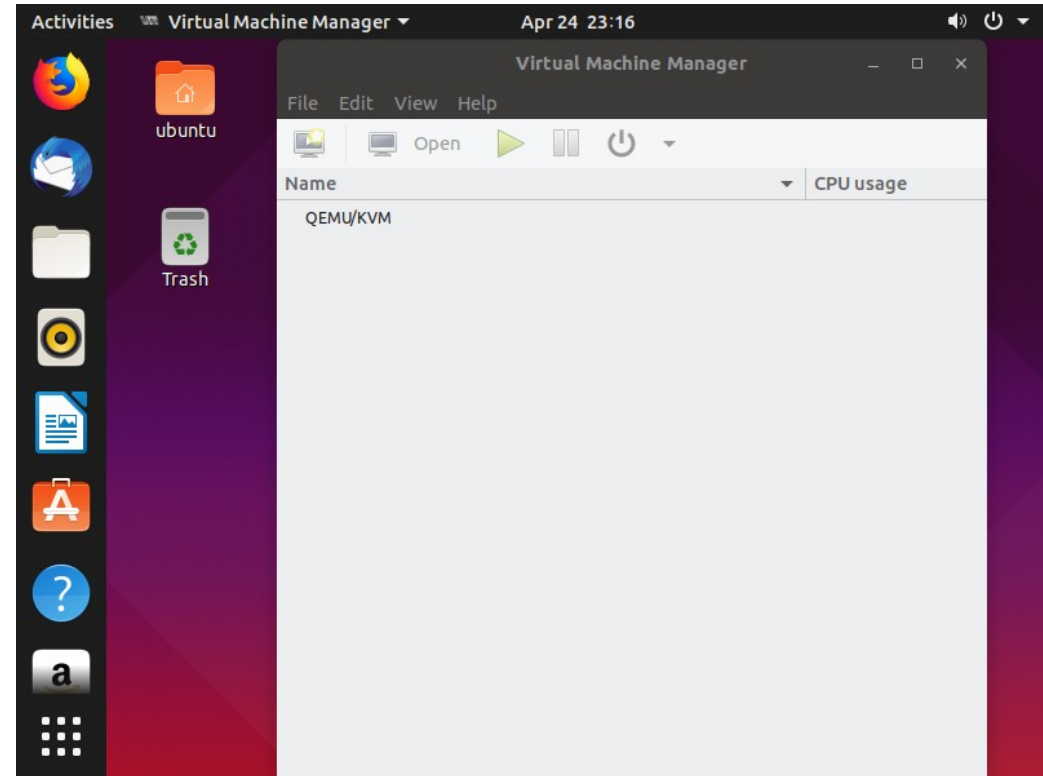


# Hypervisors: **KVM** (<https://www.linux-kvm.org/>)

Creació de una MV:

```
apt -y install virt-manager qemu-system
```

```
virsh list --all
```



# Hipervisors: **VirtualBox** (<https://www.virtualbox.org/>)

**Infraestructura free & open source de virtualització (tipus 2) per a x86/64**, desenvolupada per Innotek GmbH->Sun Microsystems->Oracle

**Admet diversos hosts:** Linux, OS X, Windows, OpenSolaris, FreeBSD ...

Permet **creació i gestió de MV**: Linux, Windows, BSD, OS/2, Solaris, Haiku ...

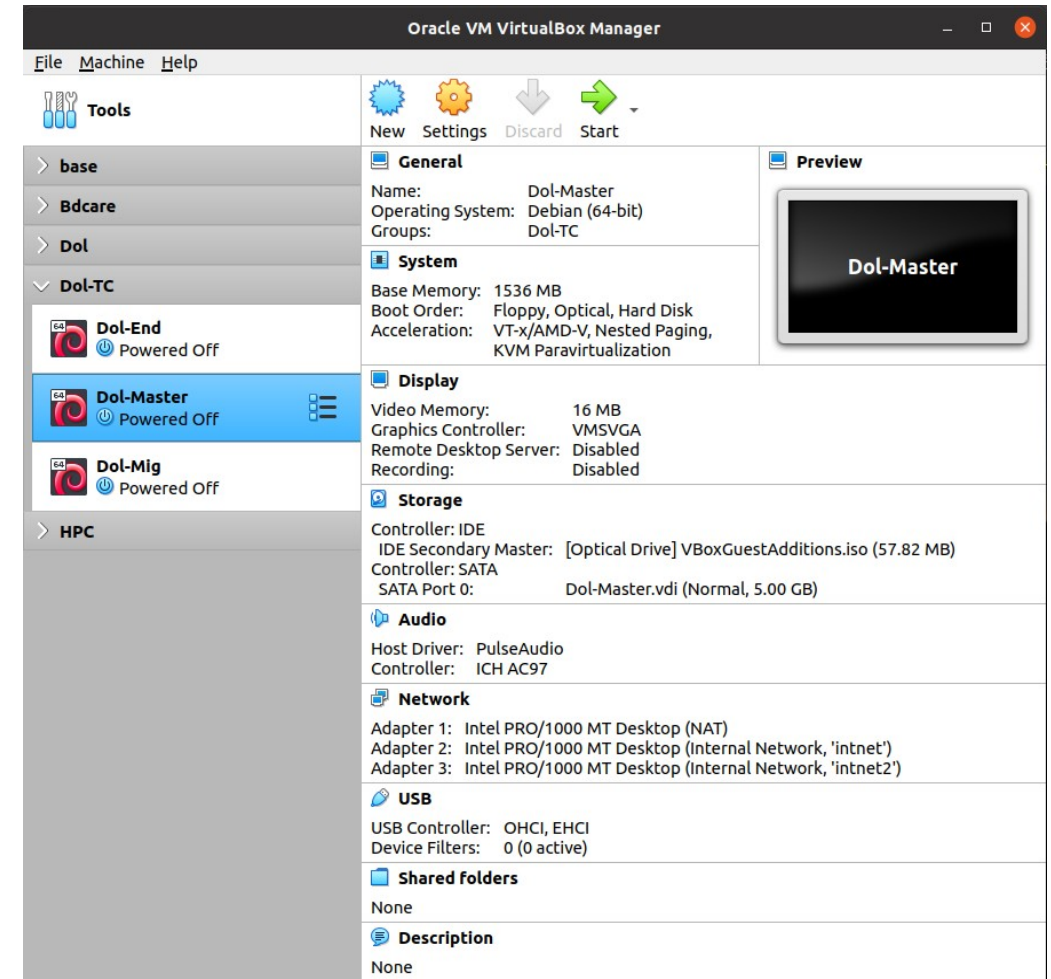
**Guest Additions:** paquet addicional amb controladors que milloren les prestacions, funcionalitats i els gràfics.

VBox és GPLv2, però l'*Oracle VirtualBox extensió pack* (USB 2.0, RDP, PXE) té Personal Use and Evaluation License que permet ús personal, educació ...

**Per a la seva gestió: GUI, i CLI: VBoxManage**

**Format de disc:** VDI, VMDK, VHD, HDD, QDE, QCOW i OVF (per exportar i importar *appliances*)

**Permet:** muntar ISO (virtuals/físiques de CD/DVD), snapshots (congela l'estat de la MV i és possible tornar a la configuració anterior)



# Hypervisors: **VirtualBox** (<https://www.virtualbox.org/>)

## Suporta:

Acceleració 3D, 32 virtualsCPU, dispositius IDE, SATA, SCSI, o connexió a iSCSI, suport ACPI, pantalla completa, 4 NIC d'Ethernet (36 des de CLI), USB, integració amb teclat/ratolí.

Cada MV pot ser configurada amb *programari-based virtualization* (guests de 32 bits) o *maquinari assisted virtualization* (necessita VT-x/AMD-V),

Remote Desktop Extension -VRDE- (connexió remota per RDP i USB over RDP)

Xifrat per AES, i amb el host es pot: compartir carpetes, USB, clipboard, i Drag & Drop.

## Xarxes:

- NAT (*Network Address Translation*): fa servir el NIC del host creant un router, (10.0.2.0/24) però es pot canviar amb CLI.
- NetWork NAT (funciona com un router domèstic, on les màquines que estiguin en aquesta xarxa es veuran entre si mateixes.
- Bridged (IP pròpia en la xarxa del host, la MV serà visible a la xarxa del host.
- Xarxa interna (Xarxa aïllada fent servir un switch virtual),
- Host-only (xarxa interna però que compartirà amb el host).

# Hipervisors: **VirtualBox** (<https://www.virtualbox.org/>)

**Connexió Remota:** es pot activar VirtualBox Remote Desktop Extension (VRDE) (Vbox Extensio) compatible amb Microsoft Remote Desktop Protocol (RDP) i amb això es pot utilitzar qualsevol client RDP estàndard per controlar la MV remota.

**Activació:** menú Display de la MV o amb: VBoxManage modifyvm "nom MV" --vrde on

**Alternativa I (en qualsevol hipervisor):** si la màquina té IP a la xarxa (o bé hi ha una regla de forward a la màquina que fa de gateway): **ssh -X ip\_MV** que fa servir X11 forwarding (per Windows es pot fer servir MobaTerm)

**Alternativa II:** connectar-se i accedir a un escriptori remot és a través de **Virtual Network Computing (VNC)**, aplicació client-servidor que permet prendre el control de l'ordinador servidor remotament a través d'un ordinador client.

**Advertència:** comunicació sense xifrar però es pot canviar i el servidor i client han de suportar el mateix algoritme d'enciptació.

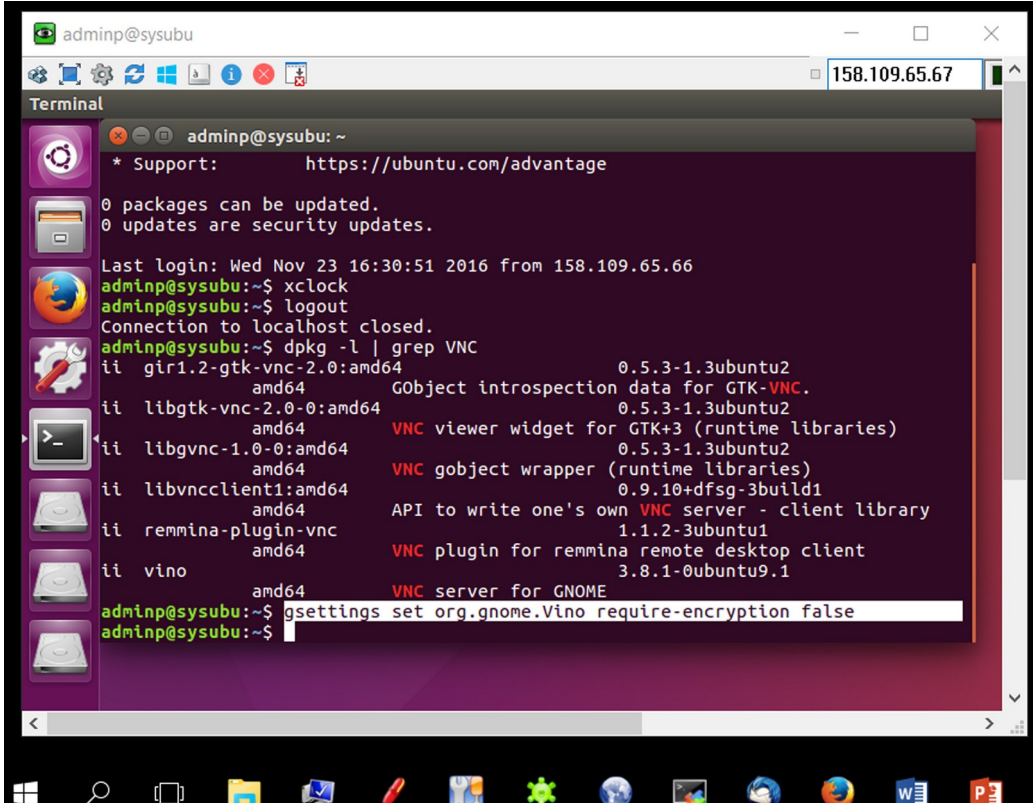
**Cas d'us:** Ubuntu, per exemple, el servidor per defecte és **vino** (i fa servir el port 5900 per defecte), enciptació per defecte (TLS, type 18 que no és suportat pels clients habituals de Windows), però es pot desactivar (des de l'usuari que té oberta la pantalla):

gsettings set org.gnome.Vino require-encryption false

**Compte!** amb aquesta configuració, tota la comunicació va en clar per la xarxa, per la qual cosa es recomana utilitzar un túnel ssh (per exemple, eines com sshvnc) o canviar a un altre servidor.

**Client sobre Linux:** Remmina

**Client sobre W:** UltraVNC



```
adminp@sysubu
* Support: https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

Last login: Wed Nov 23 16:30:51 2016 from 158.109.65.66
adminp@sysubu:~$ xclock
adminp@sysubu:~$ logout
Connection to localhost closed.
adminp@sysubu:~$ dpkg -l | grep vnc
ii gir1.2-gtk-vnc-2.0:amd64      0.5.3-1.3ubuntu2      Object introspection data for GTK-VNC.
ii libgtk-vnc-2.0-0:amd64      0.5.3-1.3ubuntu2      VNC viewer widget for GTK+3 (runtime libraries)
ii libvnc-1.0-0:amd64          0.5.3-1.3ubuntu2      VNC gobject wrapper (runtime libraries)
ii libvncclient1:amd64         0.9.10+dfsg-3build1   API to write one's own VNC server - client library
ii remmina-plugin-vnc          1.1.2-3ubuntu1        VNC plugin for remmina remote desktop client
ii vino                        3.8.1-0ubuntu9.1      VNC server for GNOME

adminp@sysubu:~$ gsettings set org.gnome.Vino require-encryption false
adminp@sysubu:~$
```

# Hypervisors: Proxmox Virtual Environment (<https://www.proxmox.com/en/proxmox-ve>)

Plataforma (Debian) i codi obert (GPL) per a la **gestió de servidors virtualitzats amb QEMU/KVM/LXC** que permet gestionar MV i contenidors, clústers d'alta disponibilitat, emmagatzematge i xarxes amb una interfase web/CLI/API.

Disseny (multi-màster) amb alta disponibilitat (HA), desplegar entorns de virtualització de classe empresarial en un CPD.

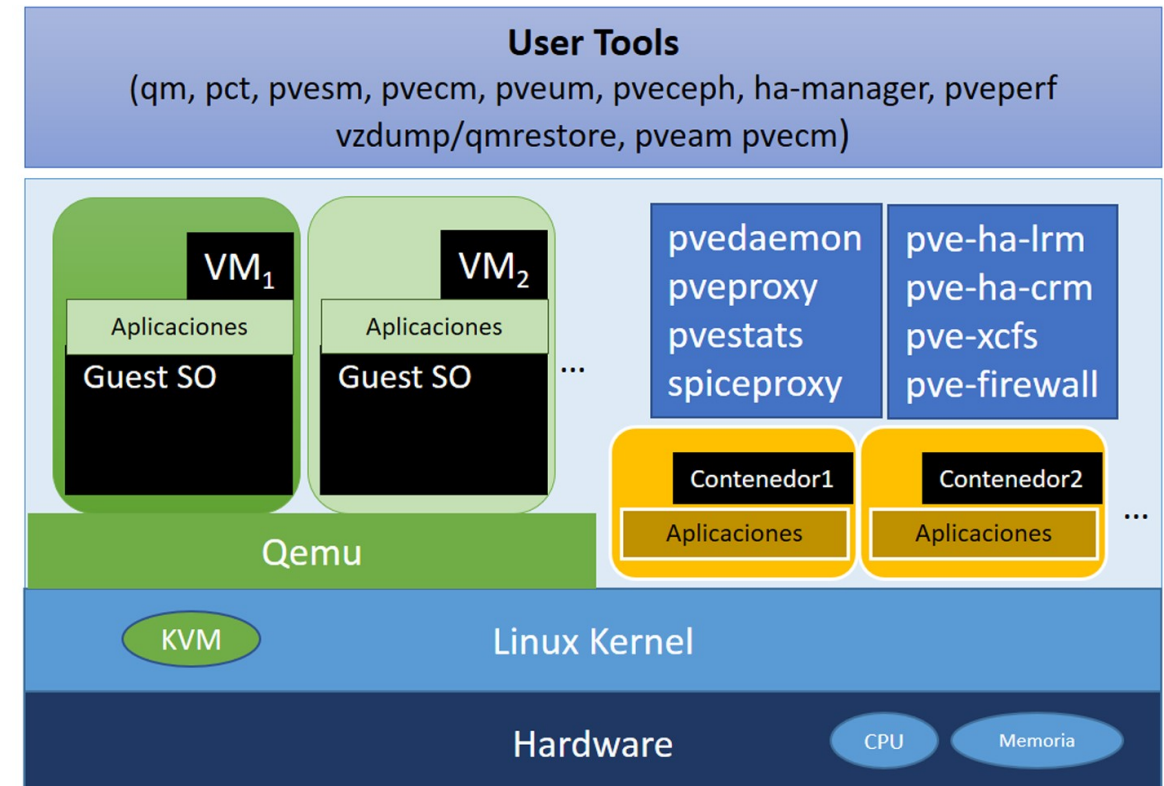
Permet: múltiples fonts d'autenticació, gestió de rols i permisos d'usuari que dóna control total a l'administrador del clúster virtualitzat de HA i que junt a l'API web RESTful permet la integració d'eines de gestió externes.

## Característiques:

**Guest:** Linux i Windows (32/64 bits), incorpora les últimes especificacions de Intel/AMD per millorar les prestacions de les MV, suportant càrregues de treball dins d'una empresa.

**Administració:** eines, web, cli API RESTful

**Arquitectura:** basada en ROA (*Resource Oriented Architecture*) que permet *High Availability* amb no SPOF (*no single point of failure*) i amb multi-màster (per garantir la disponibilitat) i tot gestionat des de la GUI (web-AJAX).



# Hypervisors: Proxmox Virtual Environment (<https://www.proxmox.com/en/proxmox-ve>)

## Característiques:

**Sistema d'arxius:** basat Proxmox VE Clúster File System, orientat a base de dades per a l'emmagatzematge dels arxius de configuració i que són replicats sobre tots els nodes utilitzant Corosync.

**HA:** basat en Linux HA per proveir un sistema fiable amb alta disponibilitat.

**Agents:** KVM i Linux Containers (LXC).

**Seguretat:** MV / contenidors aïllats amb suport segur (SSL) per aconseguir una sola VNC en HTML5 i amb gestió basada en rols per permisos per a tots els objectes (MV, contenidors, emmagatzematge ...) i autenticació multimode (local, MS ADS, LDAP ...). Tallafocs integrat que permet filtrar paquets sobre qualsevol interfície tant d'una MV com d'un contenidor, i aplicar les regles per grups denominats security groups.

**Migració:** «en viu» que permet moure MV des d'una màquina física a una altra sense temps d'apagada / recuperació.

# Hypervisors: Proxmox Virtual Environment (<https://www.proxmox.com/en/proxmox-ve>)

## Característiques:

**Backup/Recuperació:** eina (vzdump) per a la creació de snapshots dels contenidors o MV que permet salvar (i recuperar-los) aquests en diferents tipus d'emmagatzematge (com NFS, iSCSI LUN, CEPH RBD or Sheepdog).

**BridgedNetworking:** comparteixen un bridge donant la connectivitat tant entre les MV i l'exterior a través d'una interfície de xarxa i es permet la generació de VLAN (IEEE 802.1Q) i el network bonding per construir xarxes complexes adequades a les necessitats de connexió dels guests.

**Emmagatzematge:** flexible i permet que les MV puguin ser emmagatzemades en local o compartit per NFS o SAN sense grans restriccions i permetent la migració en viu si estan en sistemes compartits. Suport per a Linux LVM sobre iSCSI targets, iSCSI target, NFS, CEPH RBD, Direct to iSCSI LUN o GlusterFS i com a locals LVM Group sobre ZFS.

**Restriccions:** Proxmox VE té característiques similars a VMware Sphere, Hyper-V o Citrix-XenServer similars VMWare, que estan en 160 CPU / 2 TB RAM per host.

Cas d'ús: Es farà servir una MV en VirtualBox que conté instal·lada la plataforma ([Instruccions](#))



# Hypervisors: HyperV

(<https://docs.microsoft.com/en-us/windows-server/virtualization/virtualization>)

Plataforma per crear i administrar un entorn virtualitzat en la tecnologia de virtualització integrada en Windows.

Arquitectura: hipervisor, el servei d'administració, control d'instrumentació (WMI), bus de màquina virtual (VMbus), proveïdor de serveis de virtualització (VSP) i el controlador d'infraestructura virtual (VID).

**Administració:** GUI (un complement Microsoft Management Console -MMC-) i la connexió a màquina virtual, que dóna accés a la sortida d'una MV per poder interactuar externament. També es pot interactuar mitjançant ordres específics (cmdlets) que es despleguen sobre CLI i PowerShell.

**Objectiu:** proveir un entorn de cloud privat i adaptar-los a l'ús en funció dels canvis en la demanda, amb la finalitat de prestar uns serveis de TI més flexibles, amb la consegüent reducció del maquinari.

**Usos:** IaaS, PaaS, infraestructura d'escriptori virtual (VDI)

**Requisits:** processador de 64 bits amb suport Intel VT-x o AMD-V i amb suport maquinari per DEP (Data Execution Prevention) habilitada.

**Guests:** MV 32/64 bits a Windows (10 i 8 fins a 32VCPU, W7 -4VCPU) i WServercom guest, Linux (CentOS / RHEL, Debian, SUSE, Oracle Linux, Ubuntu i FreeBSD).

Hi ha diferències en el model de memòria i algunes característiques solament estan habilitades per Wserver.

**Plataforma d'Avaluació/Proves** (180 dies o *unlimited*)



# Hypervisors: HyperV

(<https://docs.microsoft.com/en-us/windows-server/virtualization/virtualization>)

## Elements principals:

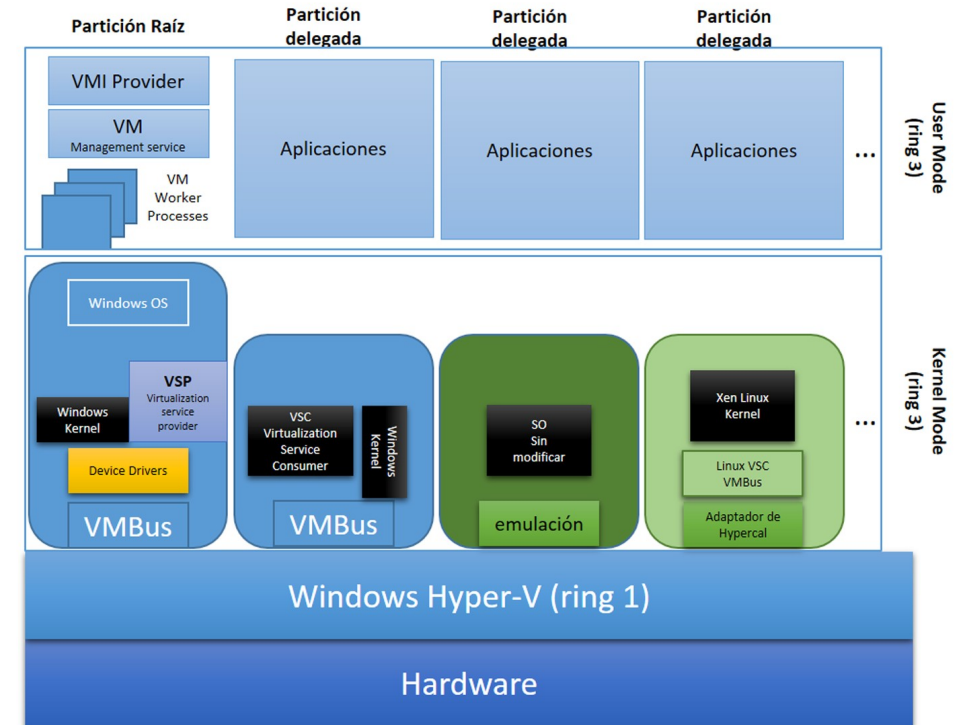
**VMBus** mecanisme de comunicació lògica entre les particions i l'hypervisor.

**VSC (Virtualization Service Client/Consumer)** dispositiu sintètic a les particions delegades/filles que fa servir recursos de **Virtualization Service Providers (VSP)**

i que es comunica amb **VMBus** per satisfer les peticions d'I E/S;

**VMMS (Virtual Machine Management Service)** responsable de gestionar l'estat de la MV;

**VMWP (Virtual Machine Worker Process)** component a l'espai d'usuari i que proveeix els serveis de gestió de la MV a l'SO de la partició arrel (un per cada MV en una partició delegada).



**Plataforma d'Avaluació/Proves** (180 dies o *unlimited*)

# Hypervisors: VMware ESXi (vSphere Hypervisor)

(<https://www.vmware.com/products/esxi-and-esx.html>)

**Hypervisor de classe empresarial de tipus 1** desenvolupat per VMware (uns dels líders empresarial en el **quadrant de Gartner** en virtualització, filial de EMC Corporation que és propietat de Dell Inc). Inclou el seu propi sistema operatiu i és una de les peces fonamentals de tota la infraestructura/productes de VMware.

Dos components principals de vSphere Hypervisor: **ESXi** i **vCenter Server**. ESXi = plataforma de virtualització per a MV i *appliances* virtuals. VCenter Server = servei administrador dels hosts ESXi connectats en una xarxa, permetent agrupar i administrar els recursos de múltiples hosts. Pot ser instal·lat: MV Windows o servidor físic, o desplegar VCenter Server Appliance (vCenter preconfigurat sobre una MV Linux) sobre el propi ESXi (5.5 o posterior).

**Principals característiques** (versió 6.x/7):

**Escalabilitat:** nombre de cores per CPU física / CPU per host = sense límits, nombre de vCPU = 480, màxim de vCPU per MV = 8.

**VMware vCenter Server® Appliance:** centre de control únic i eix central de vSphere.

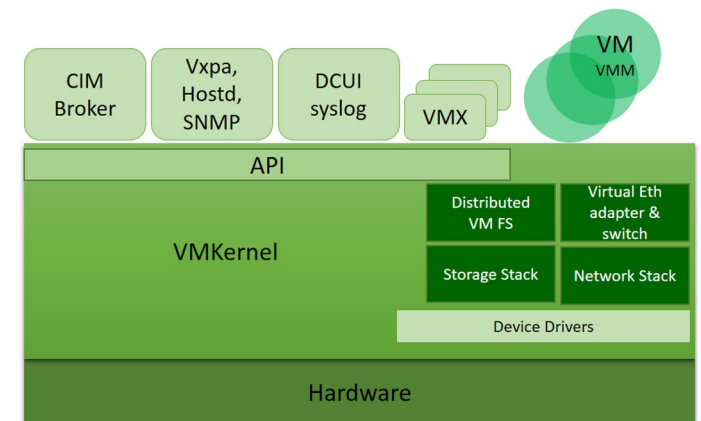
**vCenter Server® Alta disponibilitat:** solució de HA per a centres crítics.

**Còpia de seguretat i restauració en «calent».**

**Migració d'un sol pas** i actualitzacions sense interrupció del servei

**API REST:** per integració amb altres eines.

**VSphere Client:** canvi de l'antiga aplicació client a una GUI sobre HTML5



# Hypervisors: VMware ESXi (vSphere Hypervisor)

(<https://www.vmware.com/products/esxi-and-esx.html>)

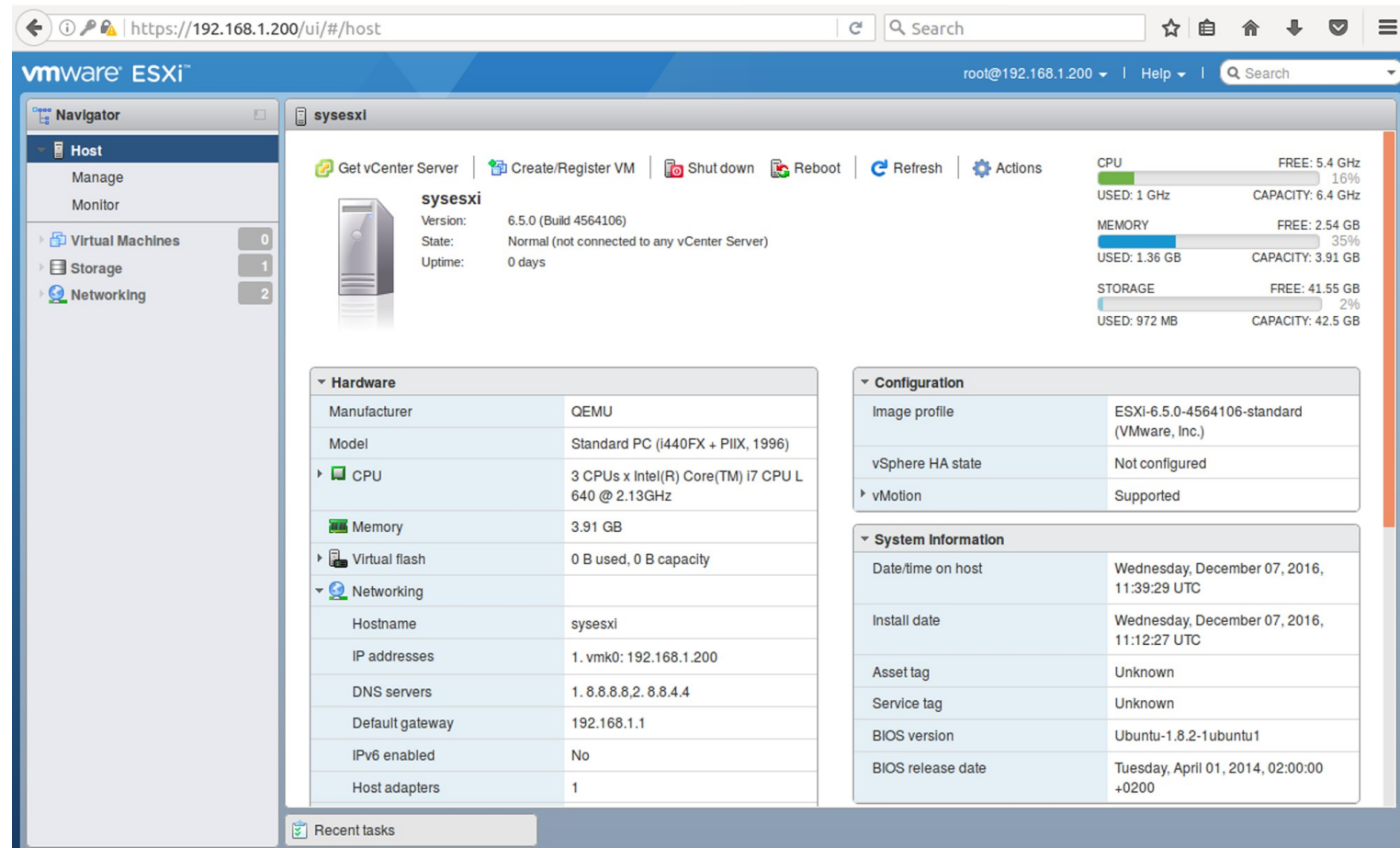
**Seguretat escalable:** polítiques de seguretat aplicables a MV, grups, individus, **Xifrat** a nivell de MV per a la protecció contra accessos no autoritzats. **Auditoria activa** per a un control total dels usuaris i accions tant actives com passives, incloent informació per a l'anàlisi forense. **Secure boot** evita injeccions o modificacions *on-fly*. **Replicació de volums virtuals**.

**Suport per MV i contenidors** en forma eficient, integrada i sense modificacions sobre el guest.

**Arquitectura:** sistema operatiu (**VM-kernel**), Interfície d'usuari de consola directa (**DCUI**, Monitor de màquina virtual (**VMM**, entorn de execució per a cada MV, inclou procés auxiliar **VMX**), Agents d'administració, Sistema d'informació comú (**CIM**).

**Cas d'us:** Downloads->Free Products->vSphere Hypervisor,

**Hands on Lab:** sense instal·lació



# Hypervisors: VMware Workstation Player (<https://www.vmware.com/products/workstation-player.html>)

Hypervisor per x64 (Windows o Linux), gratuït per a ús personal, domèstic i no comercial, pot executar appliances existents i crear i gestionar MV utilitzant el mateix nucli de virtualització que VMware Workstation Pro (no gratuït) però amb algunes limitacions (una MV per vegada, restricció que no existeix en la Pro)

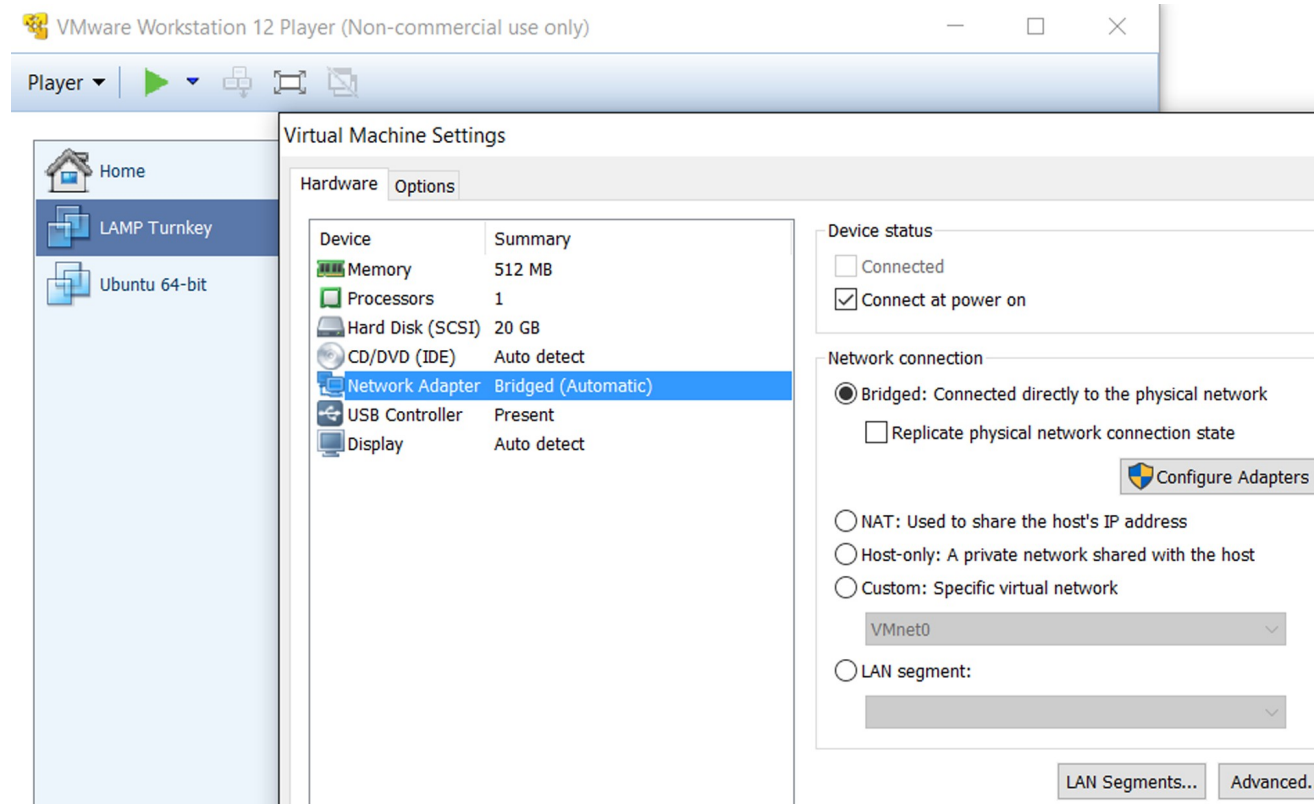
Característiques: multiple-monitor display, USB 3.0, 16 vCPU × 64 GB RAM, discs de fins a 8 TB, gràfics amb acceleració, execució de MV xifrades. , entre altres característiques, instal·lació fàcil i passos per a instal·lar una MV són els equivalents a VirtualBox.

**Instal·lació en Linux (.bundle):** doble clic o en un terminal

`sh Vmware-Player-16.x.x.x.x86_64.bundle --opció`

Opcions: gtk (interfície gràfica), console (terminal text) i custom (permet escollir diferents opcions de directori / límits).

**Cas d'ús:** Downloads



# Gestor de Cloud (+hipervisor): OpenNebula

(<https://opennebula.io/>)

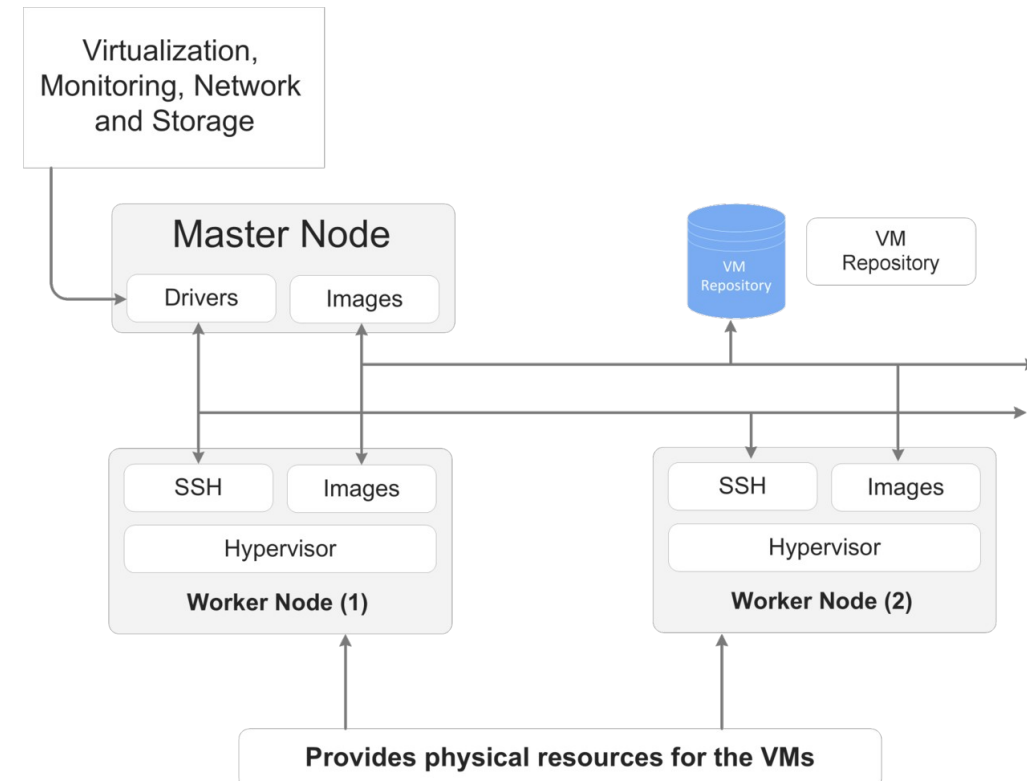
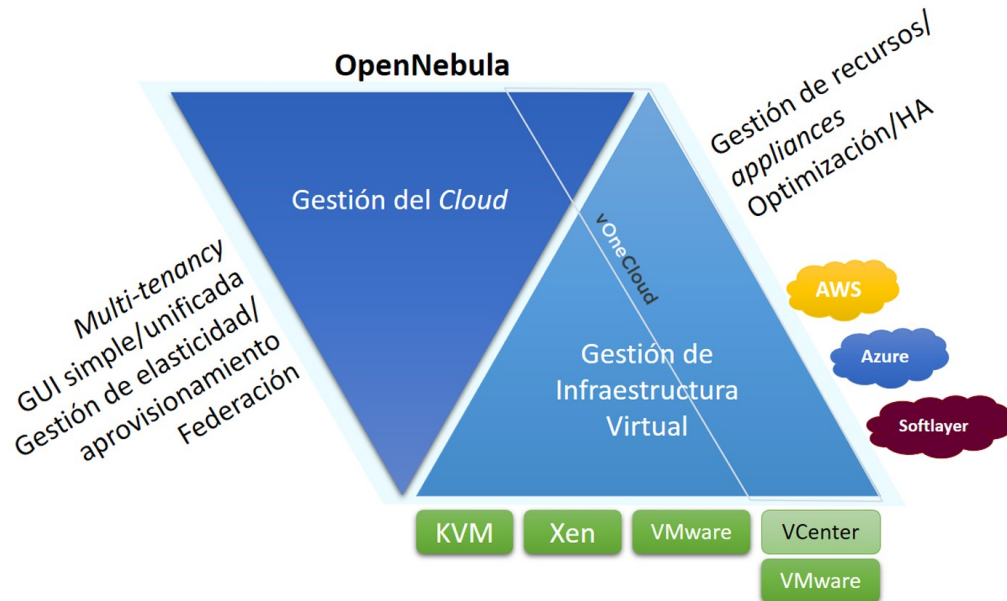
Plataforma *Cloud Computing* per administrar infraestructures centre de dades heterogènies distribuïdes.

**Gestiona la infraestructura virtual** d'un centre de dades per construir implementacions privades, públiques i híbrides d'infraestructura com a servei (IaaS).

Dos usos principals: solucions de virtualització de centres de dades i les solucions d'infraestructura en el núvol.

També és capaç d'oferir la infraestructura de núvol necessària per operar un núvol sobre les solucions d'administració d'infraestructura existents.

OpenNebula és de codi obert, sota llicència ApacheV2.



# Virtualització: containers

La virtualització a nivell de sistema operatiu fa referència a un paradigma del sistema operatiu en què el nucli permet l'existència de diverses instàncies aïllades de l'espai de l'usuari.

Aquests casos, anomenats contenidors (LXC, Docker), Zones (Solaris), servidors privats virtuals (OpenVZ), particions, entorns virtuals (VEs), nucli virtual (DragonFly BSD) o Gàbies (jail FreeBSD o presó chroot) on el programari d'aplicació 'veu' un sistema HW/SW complet però solament veuran els dispositius assignats al contenidor.

En Linux es equivalent a canviar (en una implementació avançada) del mecanisme estàndard chroot, que canvia la carpeta arrel aparent del procés en curs i els seus fills. A més dels mecanismes d'aïllament, el nucli sovint proporciona funcions de gestió de recursos per limitar l'impacte de les activitats d'un contenidor sobre altres contenidors.

Implementations [\[ edit \] https://en.wikipedia.org/wiki/OS-level\\_virtualisation](https://en.wikipedia.org/wiki/OS-level_virtualisation)

Mechanism ⇅	Operating system ⇅	License ⇅	Available since or between ⇅	Features									
				File system isolation ⇅	Copy on Write ⇅	Disk quotas ⇅	I/O rate limiting ⇅	Memory limits ⇅	CPU quotas ⇅	Network isolation ⇅	Nested virtualization ⇅	Partition checkpointing and live migration ⇅	Root privilege isolation ⇅
chroot	Most UNIX-like operating systems	Varies by operating system	1982	Partial <sup>[a]</sup>	No	No	No	No	No	No	Yes	No	No
Docker	Linux, <sup>[7]</sup> FreeBSD, <sup>[8]</sup> Windows x64 (Pro, Enterprise and Education) <sup>[9]</sup> macOS <sup>[10]</sup>	Apache License 2.0	2013	Yes	Yes	Not directly	Yes (since 1.10)	Yes	Yes	Yes	Yes	Only in Experimental Mode with CRIO <sup>[1]</sup>	Yes (since 1.10)
LXC	Linux	GNU GPLv2	2008	Yes <sup>[12]</sup>	Yes	Partial <sup>[e]</sup>	Partial <sup>[f]</sup>	Yes	Yes	Yes	Yes	Yes	Yes <sup>[12]</sup>

# Virtualització: cgroups

cgroups (control groups) és una característica del nucli Linux que limita, compta i aïlla l'ús de recursos (CPU, memòria, E/S de disc, xarxa, etc.) d'una col·lecció de processos.

Enginyers de Google (principalment Paul Menage i Rohit Seth) van iniciar el treball en aquesta funció el 2006 amb el nom de "contenidors de procés" es va canviar a "grups de control" al 2007 per evitar confusions causades per diversos significats del terme "contenedor" i es va fusionar al nucli principal del nucli Linux 2.6.24 (2008).

Des de llavors, els desenvolupadors han afegit moltes funcions i controladors nous, i un nou redisseny en 2013.

Cgrups es poden fer servir de moltes formes:

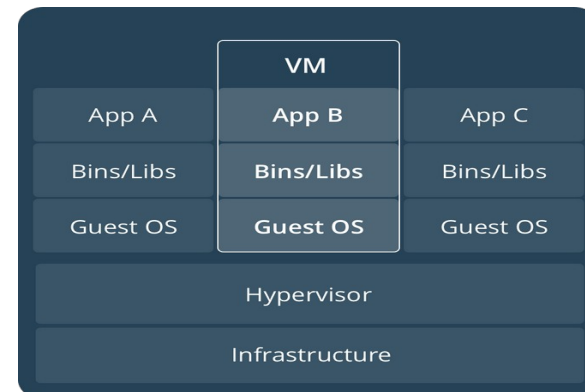
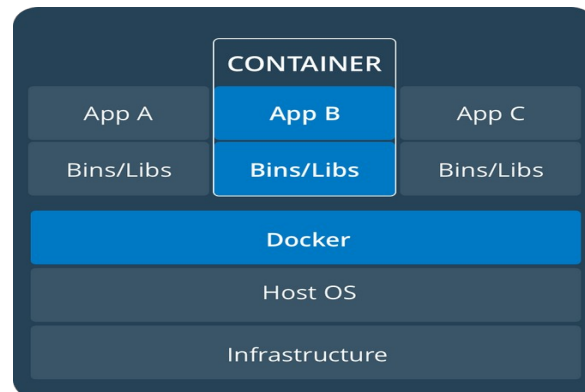
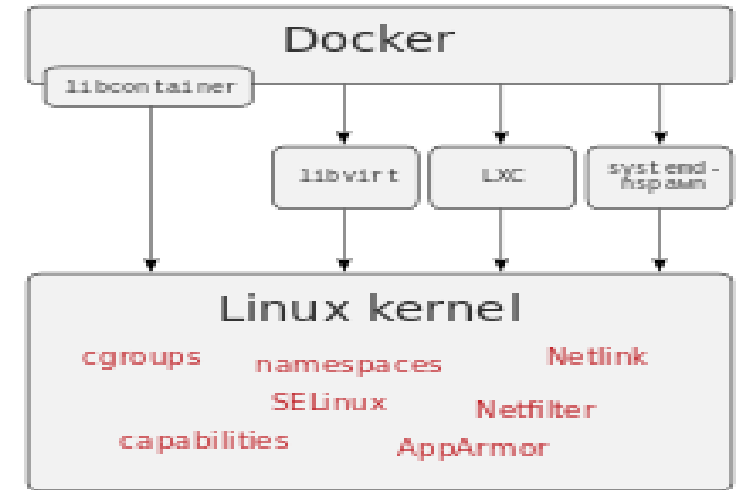
- Accedint a un cgroup virtual file system manualment.
- Creant i gestionat grups on the fly amb la llibreria libcgroup.
- Utilitzant "rules engine daemon" que automàticament pot moure processos a cgroups com estigui especificat en la seva configuració.
- Indirectament a través de altre software que faci servir cgroups: Docker, Firejail, LXC, libvirt, systemd, etc.

`systemd-cgls` ens mostrarà un arbre dels grups i les seves dependències



# Virtualització: docker

- Arquitectura client/servidor
- Imatges
- Docker Hub, permet compartir imatges
- Arxiu Docker i UFS (Union File System)
- Docker Compose; eina per definir i desplegar multi-containers
- Docker (CLI): client per interactuar amb la API
- Docker Swarm: funcionalitat nativa de clustering per contenidors que permet tractar a un grup de *Docker engines* en una *single virtual Docker engine*.
- Open Source i free: Docker Engine – Community
- Install: debian <https://docs.docker.com/install/linux/docker-ce/debian/>

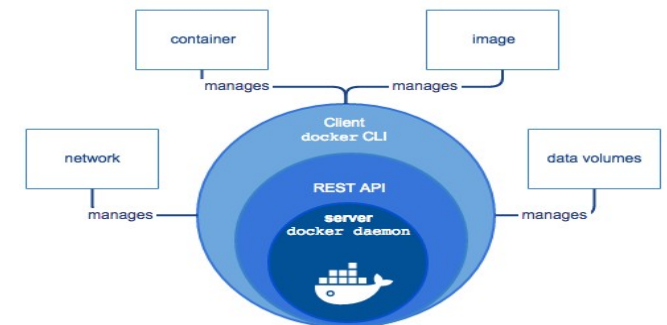
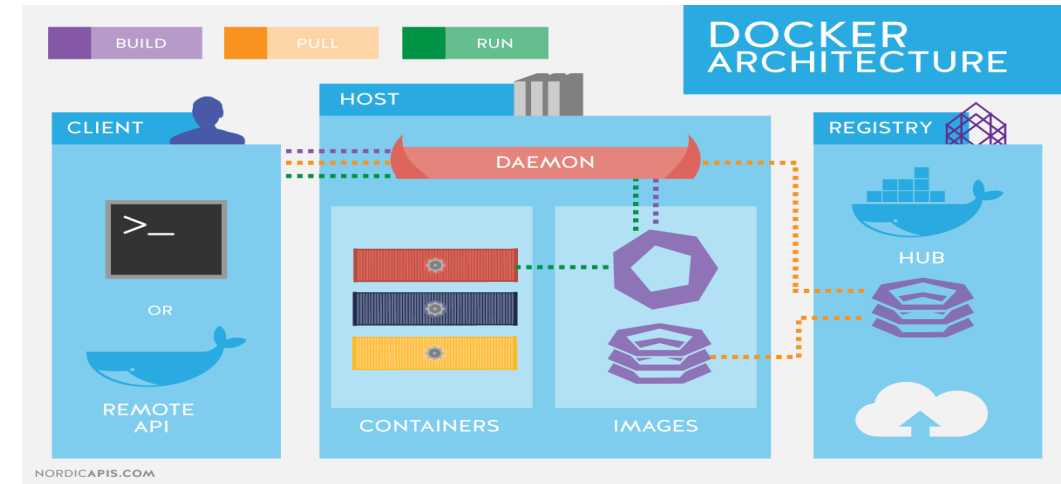




# Virtualització: docker: get started

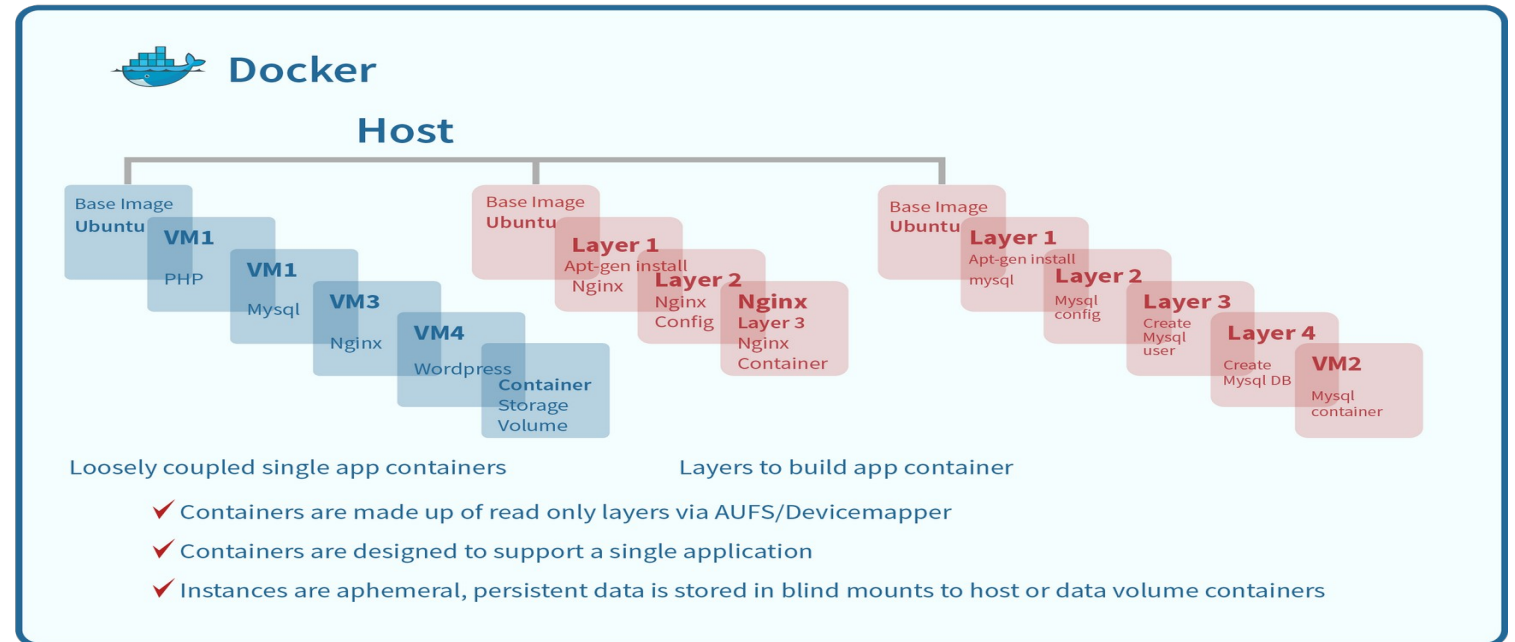
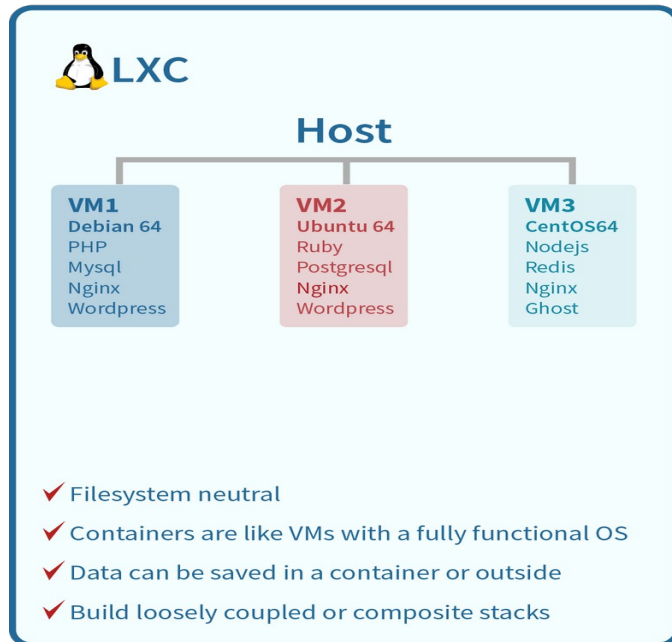
- 1) `docker -version`
- 2) `docker info`
- 3) `docker run hello-world`
- 4) `docker image ls`
- 5) `docker container ls -all`
- 6) `docker search ubuntu`
- 7) `docker pull ubuntu`
- 8) `docker run ubuntu ls -l`
- 9) `docker run -a, --attach | -d, --detach | -i, --interactive | --name`
- 10) `docker run -it alpine /bin/sh --> dintre exit`
- 11) `docker ps -a`
- 12) `docker run -d dockersamples/static-site`
- 13) `docker ps`
- 14) `docker stop containerID`
- 15) `docker rm containerID`
- 16) `docker run --name static-site -d -P dockersamples/static-site`
- 17) `docker port static-site URL -> localhost:port`
- 18) `docker-machine ip default`
- 19) `docker run --name static-site -d -p 8888:80 dockersamples/static-site`

URL -> IP:8888



# Virtualització: LXC vs Docker

## Key differences between LXC and Docker



# Virtualització: Dockerfile

Amb un Dockerfile es pot definir el que passa a l'entorn dins del vostre contenidor. L'accés a recursos com les interfícies de xarxa i les unitats de disc virtualitzat dins, que estan aïllats de la resta del sistema, es poden transportar cap a fora o 'copiar' fitxers des de fora al entorn. Això dona garanties la aplicació definida en aquest document Dockerfile es comportarà exactament igual allà on s'execute. Crear en un directori buit l'arxiu **Dockerfile**, **app.py** i **requirements.txt**

## Arxiu: Dockerfile

```
# Dockerfile: Use an official Python runtime as a parent image
FROM python:3.6-slim
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
ADD . /app
# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt
# Make port 80 available to the world outside this container
# Define environment variable
# ENV NAME World
# Run app.py when the container launches
CMD ["python", "app.py"]
```

## Arxiu: requirements.txt

```
flask
redis
```

# Virtualització: Dockerfile

## Arxiu: app.py

```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2, port=6379)
app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"
    html = "<h3>Hello {name}!</h3>" \
        "<b>Hostname:</b> {hostname}<br/>" \
        "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)
if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

**docker build --tag=app .**

### docker images

REPOSITORY	TAG	IMAGE ID
App	latest	326387cea398

**docker run -d -p 5000:5000 app** → **generarà un error ja que Redis no existeix**

# Solució: entorns multi-contenidors = docker-compose

## Arxiu: **docker-compose.yml**

```
version: '2'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/app
    depends_on:
      - redis
  redis:
    image: redis
```

**Per crear els dos contenidors executar:**

En Ubuntu: **docker compose up**

En Debian:

**apt install docker-compose**

**docker-compose up**

## Virtualització: Docker Compose & app to a Swarm Voting App

Python webapp permet votar entre dues opcions  
Redis queue junta el nous vots  
.NET worker recull els vots i enmagatzema aquest en...  
Postgres database amb suport d'un Docker volume  
Node.js webapp mostra els resultats de la votació en temps real

```
git clone https://github.com/docker/example-voting-app.git
cd example-voting-app
docker swarm init
```

### **docker-stack.yml**

```
version: "3"
services:
  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - frontend
  deploy:
    replicas: 2
    update_config:
      parallelism: 2
      delay: 10s
    restart_policy:
      condition: on-failure
  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - backend
    deploy:
      placement:
        constraints: [node.role == manager]
  environment:
    POSTGRES_HOST_AUTH_METHOD: trust
```

## Virtualització: Docker Compose & app to a Swarm Voting App

vote:  
 image: dockersamples/examplevotingapp\_vote:before  
 ports:  
 - 5000:80  
 networks:  
 - frontend  
 depends\_on:  
 - redis  
 deploy:  
 replicas: 2  
 update\_config:  
 parallelism: 2  
 restart\_policy:  
 condition: on-failure  
result:  
 image: dockersamples/examplevotingapp\_result:before  
 ports:  
 - 5001:80  
 networks:  
 - backend  
 depends\_on:  
 - db  
 deploy:  
 replicas: 1  
 update\_config:  
 parallelism: 2  
 delay: 10s  
 restart\_policy:  
 condition: on-failure  
worker:  
 image: dockersamples/examplevotingapp\_worker  
 networks:  
 - frontend  
 - backend

```
deploy:
  mode: replicated
  replicas: 1
  labels: [APP=VOTING]
  restart_policy:
    condition: on-failure
    delay: 10s
    max_attempts: 3
    window: 120s
  placement:
    constraints: [node.role == manager]
visualizer:
  image: manomarks/visualizer
  ports:
    - "8080:8080"
  stop_grace_period: 1m30s
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock"
  deploy:
    placement:
      constraints: [node.role == manager]
networks:
  frontend:
  backend:
  db-data:
```

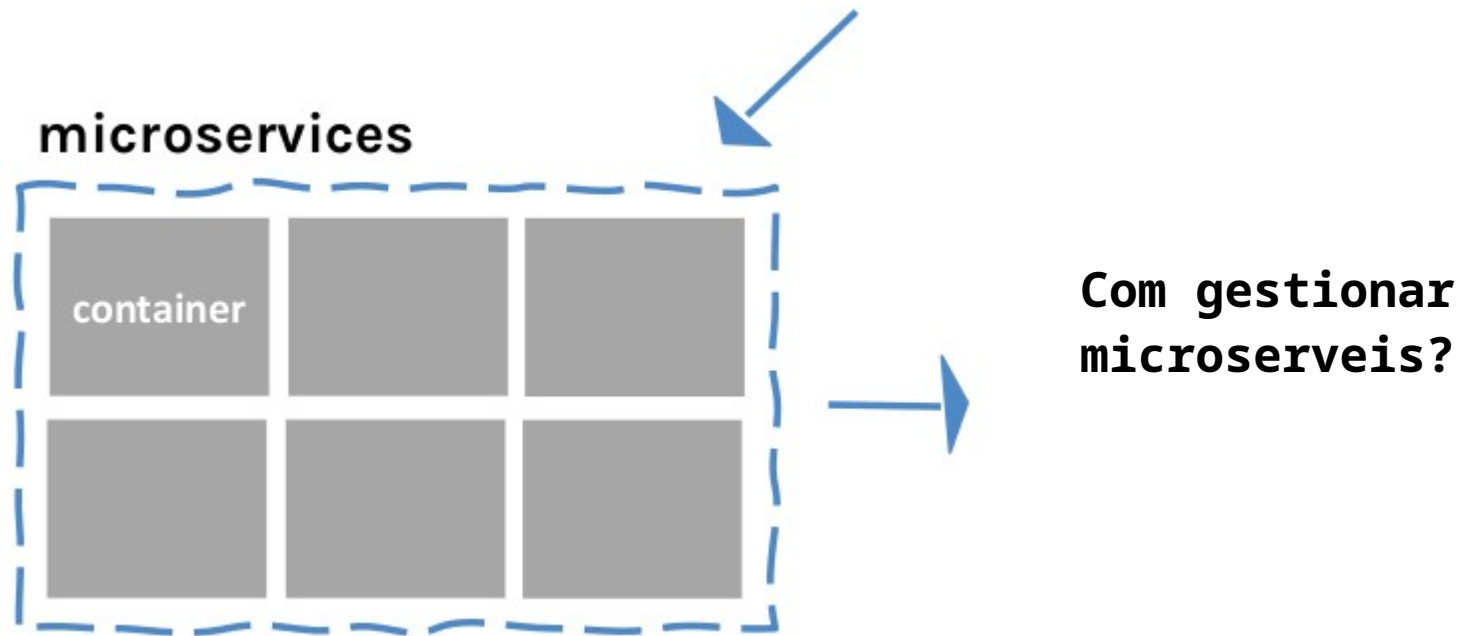
docker stack deploy --compose-file docker-stack.yml vote  
docker stack services vote  
URL interfase: <http://localhost:5000>  
URL results: <http://localhost:5001>  
docker stack rm vote

# Kubernetes (k8s)

**Entorn virtual:** Pros: eliminar la complexitat, Contres: no aïlla del SO

**Màquines virtuals:** Pros: aïlla el convidat del SO de l'amfitrió, Contres: maquinari d'ús intensiu

**Contenidors:** Pros: lleuger Contres: problemes de seguretat, escalabilitat, i control





# Kubernetes (k8s)

Hem parlat dels avantatges/desavantatges dels entorns, màquines virtuals, i contenidors

**Objectiu:** trobar maneres efectives de desplegar les nostres aplicacions (més difícil que el que podria imaginar inicialment) i desglossar una aplicació complexa en serveis més petits (és a dir, microserveis)

Problemes solucionats fins ara:

- sistema operatiu conflictiu/diferent
- diferents dependències
- comportament estrany "inexplicable".



# Kubernetes (k8s)

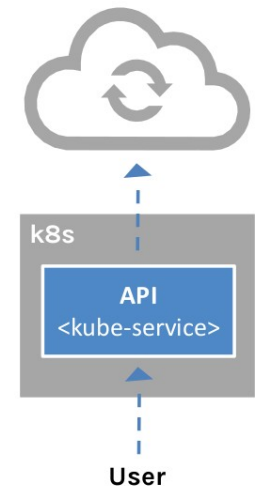
K8s gestiona contenidors. Plataforma de codi obert per a la gestió de contenidors desenvolupat per Google i introduït el 2014.

S'ha convertit en l'API estàndard per crear aplicacions natives del núvol, present a gairebé tots els núvols públics.

Els usuaris de K8s defineixen regles sobre com hauria de gestionar els contenidors i K8s s'encarrega de la resta!

Raons per les quals utilitzar contenidors i una API de contenidors com Kubernetes:

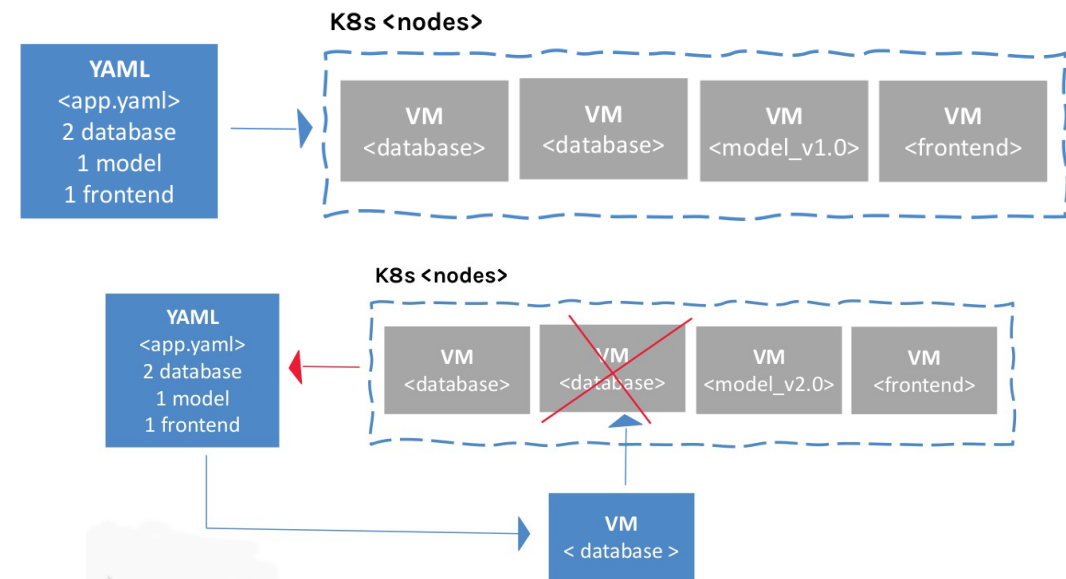
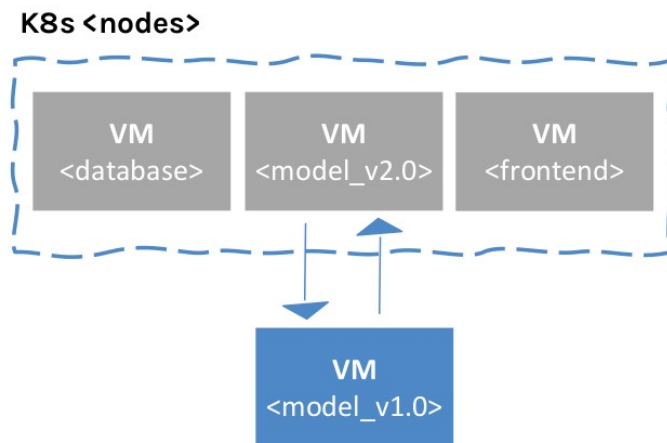
- Velocitat
- Escalat (tant del programari com dels equips)
- Abstracció de la infraestructura
- Eficiència





# Kubernetes (k8s)

**Velocitat:** amb què es pots respondre a les innovacions desenvolupades per d'altres (per exemple, canvi en la indústria del programari -CD- des de l'enviament fins al lliurament). **Sistema immutable:** no es pot canviar el contenidor en execució, però si crear-ne un de nou i substituir-lo en cas de fallada (conservant l'historial i carregar imatges més antigues). **Configuració declarativa:** es pot definir l'estat desitjat del sistema que reafirma l'estat declaratiu anterior per tornar enrere. **Imperatiu:** la configuració es defineix mitjançant l'execució d'una sèrie d'instruccions, però no al revés. **Sistemes d'autoreparació en línia:** k8s pren accions per garantir que el l'estat actual coincideix amb l'estat desitjat.





# Kubernetes (k8s)

**Escalabilitat:** A mesura que el producte creix, és inevitable que hagi d'escalar:

- Programari
- Equip/s que el desenvolupen

Kubernetes ofereix avantatges per abordar l'escalabilitat:

- Arquitectures desacoblades: cada component està separat de l'altre components per API definides i equilibradors de càrrega de servei.
- Escalat fàcil per a aplicacions i clústers: simplement canviant un número en un fitxer de configuració, k8s s'encarrega de la resta (declaratiu).
- Escalar equips de desenvolupament amb microserveis: equip petit és responsable del disseny i la prestació d'un servei que es consumeix per altres petits equips (mida òptima del grup: equip de 2 pizzes).

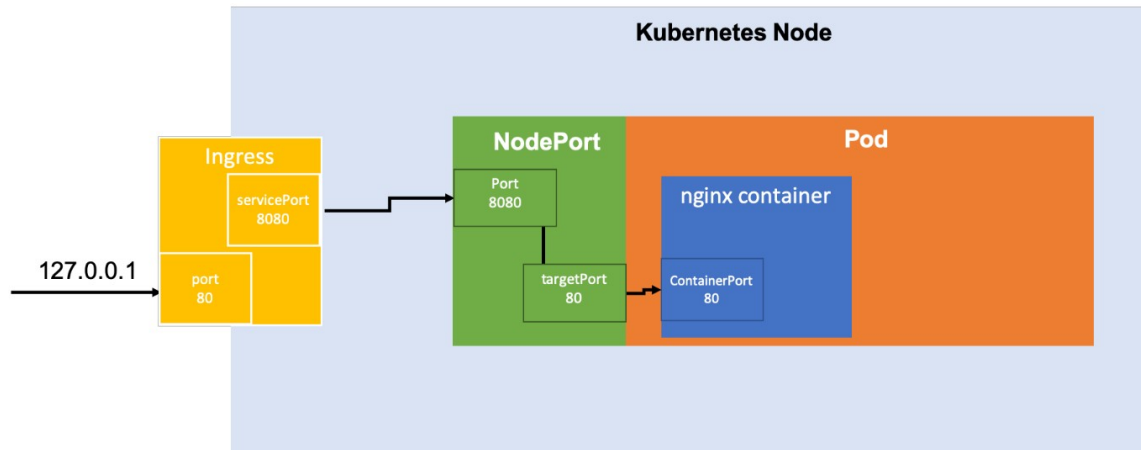
# Kubernetes (k8s)



**Escalabilitat:** Kubernetes ofereix nombroses abstraccions i API que ajuden a construir arquitectures de microservei desacob

- Els **pods** poden agrupar imatges de contenidors desenvolupades per diferents equips en una única unitat desplegable (sir docker-compose)
- **Altres serveis per aïllar** un microservei d'un altre (p. ex. equilibri de càrrega, denominació i descobriment)
- Els **espais de noms** controlen la interacció entre serveis
- **Ingress** combina diversos microserveis en una única API externalitzada (interfície fàcil d'utilitzar)

K8s ofereix solucions entre fer-ho "de la manera més difícil" i un servei totalment gestionat



Un **Pod** és un grup d'un o més contenidors, units per a finalitats d'administració i xarxa. Un desplegament de Kubernetes comprova l'estat del pod i reinicia el contenidor del pod si finalitza. Els desplegaments són la manera recomanada de gestionar la creació i l'escalat de Pods.

De manera predeterminada, només es pot accedir al pod mitjançant la seva adreça IP interna dins del clúster de Kubernetes.

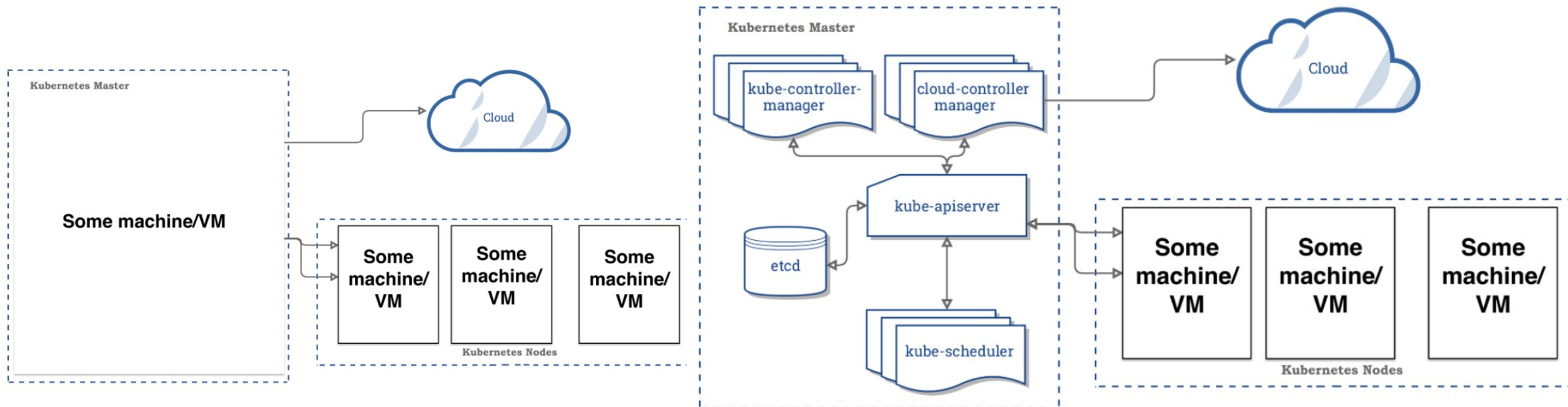
Per fer que un contenidor sigui accessible des de fora de la xarxa virtual de Kubernetes, s'ha d'exposar el pod com a servei de Kubernetes.

# Kubernetes (k8s)



**Eficiència:** Hi ha un benefici econòmic concret a l'abstracció perquè les tasques de diversos usuaris es poden empaquetar en menys màquines:

- Consumeix menys energia (proporció de l'utilitat a la quantitat total)
- Limitar els costos d'execució d'un servidor (ús d'energia, refrigeració requisits, espai del centre de dades i potència informàtica bruta)
- Creeu ràpidament un entorn de prova per a desenvolupadors com a conjunt de contenidors
- Redueix el cost de les instàncies de desenvolupament a la teva pila, alliberant recursos per desenvolupar-ne altres que fossin costosos



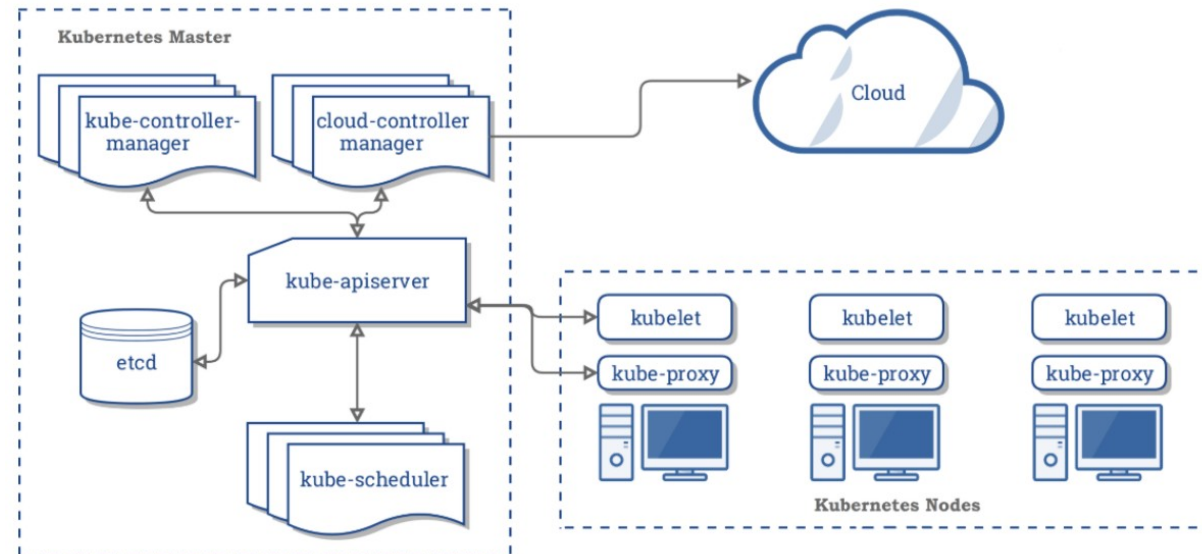
# Kubernetes (k8s)



**Arquitectura:** La tasca principal del node mestre és gestionar els nodes de treball per executar una aplicació

El **node mestre** consta de:

- 1) La **API server** conté diversos mètodes per accedir directament a Kubernetes
- 2) El **planificador** (scheduler) assigna una aplicació a cada node de treball
- 3) **Controlador**: Fa un seguiment dels nodes de treball, Gestiona les fallades del node i es replica si cal, Proporciona els punts finals per accedir a l'aplicació des del món exterior
- 4) El controlador del núvol es comunica amb el subministrament del núvol pel que fa als recursos com ara nodes i adreces IP
- 5) **etcd** funciona com a backend per al descobriment de serveis que emmagatzema l'estat del clúster i la seva configuració



# Kubernetes (k8s)



## Arquitectura:

Un **node de treball** està format per:

- 1) **Contenidor** (*runtime*) que prove d'una imatge de Docker especificada i la desplega en un node de treballador
- 2) **kubelet** parla amb el servidor de l'API i gestiona els contenidors del seu node
- 3) **kube-proxy** equilibra el trànsit de xarxa entre els components de l'aplicació i el món exterior

## Desplegant un clúster Kubernetes

Per implementar un clúster, s'ha d'instal·lar Kubernetes. Una de les formes (no per producció) es pot fer servir **minikube** per desplegar un clúster en mode local.

Després d'instal·lar minikube, es pot fer servir start per començar la sessió creant una MV màquina virtual (stop per aturar-lo i delete per suprimir-lo per eliminar la VM):

```
minikube start
```

```
minikube stop
```

```
minikube delete
```

```
kubectl get po -A
```

Per descarregar la versió adequada de kubectl: `minikube kubectl -- get po -A`

```
alias kubectl="minikube kubectl --"
```

```
minikube dashboard
```

```
kubectl get nodes/services/deployments
```

Ver ordres bàsiques: <https://minikube.sigs.k8s.io/docs/handbook/controls/>

<https://kubernetes.io/docs/tasks/tools/>



# Monitorització

Càrrega



¿Què està passant?

Problema de les mesures

- ¿Quina informació?
  - ¿On és aquesta informació?
  - ¿Com es pot extraure i on l'emmagatzemo?
- L'instrument de mesura afecta a la mesura?

¿Quan fer la mesura?

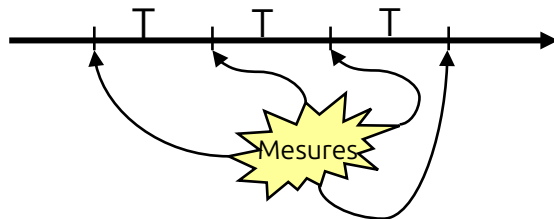
Cada vegada que ocorre un esdeveniment

Cada període fix de temps (mostreig):

Observació a intervals regulars o aleatoris

Anàlisi estadística de dades més fàcil

Volum d'informació recollida i precisió: depenen de T



**Tipus:**

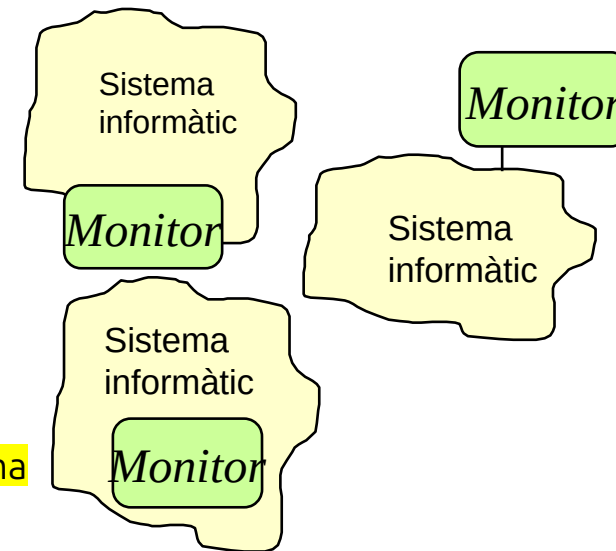
**Per programari:**  
instal·lats en el sistema

**Per Maquinari:**  
Dispositius externs a sistema

**Híbrids**

**Monitor:** Eina dissenyada per observar l'activitat d'un sistema informàtic mentre és utilitzat pels usuaris

**Accions típiques d'un monitor:** Observar el comportament, Recollir dades estadístiques, Analitzar aquestes dades, Mostrar els resultats



**Característiques**

- Interferència o sobrecàrrega (overhead)
- Precisió: Qualitat de la mesura
- Resolució: Freqüència de mesura
- Àmbit o domini de mesura: Què mesura?
- Amplada: Bits d'informació
- Capacitat de síntesi de dades
- Cost
- Facilitat d'instal·lació i ús

# Nagios

Es pot fer servir el docker Nagios [jasonrivers/nagios](https://github.com/JasonRivers/Docker-Nagios) per analitzar el seu funcionament i quines opcions té (usuari i passwd: nagiosadmin / nagios). <https://github.com/JasonRivers/Docker-Nagios>

## Icinga

Fork de Nagios per qüestions ètiques de l'Open Source de Nagios. Inclòs en tots els repositoris (compte que hi ha Icinga i Icinga2 -es recomana el primer per simplicitat-

### **Altres eines de monitorització:**

- Cacti,
- PandoraFMS,
- Zabbix,
- Monit,
- Munin,
- Prometheus,
- ntopng (network),
- Shinken,
- **Netdata**
- MRTG (network),
- Collectd,
- XYMon

# Netdata

Plataforma de monitorització i visualització open-source i altament configurable (tant per 1 màquina com per N)

Visualització totalment automatitzada: Netdata visualitza totes les mètriques d'una manera significativa, correlacionant les dimensions adequades i aplicant la configuració adequada per a cada cas.

Alertes totalment automatitzades: Netdata inclou alertes preconfigurades per a 350 components i aplicacions úniques. Totes aquestes plantilles d'alerta s'apliquen automàticament als components, aplicacions i gràfics adequats.

Interfície d'usuari configurable: permet definir fàcilment les dades de la interfície d'usuari, sense necessitat d'aprendre un llenguatge de consulta.



# Netdata

Per instal·lar-la en una màquina: `apt install netdata` i obrir el navegador i posar com URL: `localhost:19999`

**Per monitoritzar un conjunt de màquines:** s'ha de fer un compte en `https://app.netdata.cloud/` fent el *Sign-up* amb un mail (també es pot fer amb google o github).

**Important:** si s'ha instal·lat prèviament dels repositoris s'ha de eliminar amb `apt remove --purge netdata`

Seleccionar el SO i ens donarà una ordre `wget . . . .` per executar en cada màquina que inclou un hash per afegir totes les màquines al compte durant la instal·lació.

The screenshot shows the Netdata Cloud web interface. The browser address bar displays the URL: `https://app.netdata.cloud/spaces/remo-suppi-space-space/rooms/all-nodes/integrate-anything#expires_at=2025-01-07T06%253A24%253A16.349371434Z&metrics_correlation=false&after=-900&before=0&utc=Europe%2FAndorra&offset=`. The interface features a sidebar on the left with a 'Remo Suppi space' header and a 'Community' tab. Below this, there's a notification to 'Upgrade your plan for unlimited access and Business features.' with an 'Upgrade' button. The sidebar also includes links for 'Integrations', 'Connect Nodes', 'Invite Users', 'Rooms', and 'My rooms'. The main content area has a 'Welcome to Netdata!' message, a search bar for integrations, and a 'Deploy' section. The 'Deploy' section lists various operating systems and data collection methods. A terminal window shows the command to install Netdata using `wget` and `curl`.

```
wget -O /tmp/netdata-kickstart.sh https://get.netdata.cloud/kickstart.sh && sh /tmp/netdata-kickstart.sh --nightly-channel --claim-token j-Hw70MAwbH0bXXWmC9tHDK_EffqwXVjy1V0sMLgQJ5iUw4NzxxNc6K3K9_4zH9Je7r3UHjXrXLcaVPwvyD1k_NKkt6SLQUK12QAwfhpSB87FP_0JCRk_ANPCX0vYSCsCpJjl_k --claim-rooms 2e7368c1-56d9-427a-a948-b98863183a0b --claim-url https://app.netdata.cloud
```

Administració de sistemes GNU/Linux, 2016: <http://openaccess.uoc.edu/webapps/o2/handle/10609/60687>

Adm. Avançada, 2016: <http://openaccess.uoc.edu/webapps/o2/handle/10609/60685>

Docker: <https://docs.docker.com/engine/docker-overview/>

Install Docker: <https://docs.docker.com/install/linux/docker-ce/debian/>

Docker Containers: <https://docs.docker.com/get-started/part2/>

Docker Net overview: <https://docs.docker.com/network/>

WebApps with Docker: <https://medium.com/@Grigorkh/docker-for-beginners-part-3-webapps-with-docker-18f2243c144e>

Docker Swarm Tutorial: <https://github.com/docker/labs/blob/master/swarm-mode/beginner-tutorial/README.md#creating-the-nodes-and-swarm>

Docker Compose & Swarm App: <https://github.com/dockerexamples/example-voting-app>

Tots els materials, enllaços, imatges, formats, protocols i informació utilitzada en aquesta presentació són propietat dels seus respectius autors i es mostren amb finalitat acadèmica i sense ànim de lucre, excepte tots aquells que tenen llicències o distribució d'ús lliure i/o cedides per tal finalitat. (Articles 32-37 de la llei 23/2006, Spain).

Sota cap concepte (en el cas que es mostrin) accions, ordres, exemples o qualsevol altre activitat es poden provar fora de l'àmbit acadèmic i que no sigui proves en màquines virtuals/xarxes internes protegides i amb finalitat d'aprenentatge ja que es podria incorre en activitats delictives i/o punibles.