Virtualització i hipervisors

PID_00286206

Remo Suppi Boldrito

Temps mínim de dedicació recomanat: 6 hores

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Enginyer de Telecomunicacions. Doctor en Informàtica per la Universitat Autònoma de Barcelona, i professor del Departament d'Arquitectura de Computadors i Sistemes Operatius d'aquesta mateixa universitat.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Josep Jorba Esteve

Primera edició: febrer 2022

Av. Tibidabo, 39-43, 08035 Barcelona
Autoria: Remo Suppi Boldrito
Producció: FUOC

Índex

Objectius

Els objectius principals d'aquest mòdul són els següents:

1. Conèixer els diferents tipus de virtualització i saber quin és el més adequat per a cada infraestructura.

2. Analitzar els diferents hipervisors i conèixer-ne els avantatges i les limitacions.

3. Analitzar les característiques dels contenidors i els seus avantatges en relació amb les màquines virtuals.

A més, l'estudiant haurà de centrar l'atenció en els següents conceptes fonamentals d'aquest mòdul:

- Virtualització i virtualització per maquinari
- Hipervisor
- KVM, VirtualBox, Hyper-V, VMware ESX
- Contenidors
- Docker, LXC

1. Hipervisors, màquines virtuals i contenidors

Com s'ha esmentat anteriorment, la virtualització és el gran actor que permet que el *cloud* existeixi i sigui possible. De la mateixa manera que un usuari no expert pot utilitzar un ordinador, perquè el SO actuarà com una capa d'abstracció que amagarà totes les complexitats dels dispositius físics presentant una interfície gràfica i pròxima a les accions que vol fer l'usuari, la virtualització serà el mateix per al *cloud*.

En certa manera, podem dir que la virtualització és l'equivalent a el SO del *cloud* perquè permetrà disposar de recursos virtualitzats (màquines, servidors, xarxes i emmagatzematge) que faran un treball real però que compartiran recursos físics i amagaran tota aquesta complexitat a l'usuari que els utilitzarà.

La pregunta que pot sorgir és: per a què necessito la virtualització si amb un sistema operatiu executant-se en un maquinari (*bare-metal*) ja tinc això? És evident que en una màquina hi podem posar un servei i podrem utilitzar els recursos per a un usuari, i això suposarà una càrrega determinada sobre el sistema físic, que en la majoria dels casos no arriba al 10 % (es poden analitzar quants recursos, en moments de funcionament estable, està fent servir el nostre ordinador).

Què passa si ara, en funció d'un estalvi de costos i per a deu usuaris més, vull aprofitar els recursos físics i haig de posar deu serveis (que poden ser iguals al que estic donant per al primer usuari, però també poden ser diferents i amb diferents SO i diferents dades)? Podria utilitzar algunes estratègies per compartir recursos i llibreries i crear un espai controlat per a les dades de cadascun d'ells, però no serien mai sistemes aïllats o privats amb les seves configuracions i les seves especificitats, ni tampoc seria àgil ni eficient: seria complicat de mantenir, monitorar i administrar.

Pensem ara en una empresa duent a terme aquesta tasca sobre cent serveis/servidors i mil clients que tenen una dinàmica; serveis que entren i surten; aplicacions que s'inicien i s'acaben; un munt de configuracions, compatibilitats i llibreries, i un llarg etcètera de qüestions específiques. Tot això comportaria que fos extremament difícil o impossible a un cost raonable, amb eficiència i, a més, de manera escalable (que és un dels objectius vitals del *cloud* com a negoci; és a dir, si tinc deu usuaris i n'arriben deu més, hauria de poder ampliar la infraestructura de manera simple i poder donar-los servei en un temps acceptable).

La virtualització és el gran possibilitador tecnològic del *cloud*, perquè es podran crear els recursos virtuals necessaris, i l'usuari «veurà» (amb una eficiència molt semblant a si tingués un dispositiu de maquinari) la seva infraestruc-

tura (màquines, servidors, discos, xarxa) i serà només la seva infraestructura, i passarà el mateix amb els *N* clients del *cloud*. Amb això s'aprofitaran tots els recursos físics existents, maximitzant l'eficiència i amb la reducció consegüent de costos.

Pel que fa a l'estalvi de recursos, per què és millor virtualitzat que físic? Si tenim un requeriment de disc, com per exemple d'1 Tbyte per a un usuari, i el recurs és físic, el proveïdor haurà d'assignar-li, tant si l'usa com si no, 1 Tbyte, perquè l'usuari pagarà per això i el cost estarà en relació amb el cost del recurs; si tinc mil usuaris amb els mateixos requeriments, s'haurà de disposar d'1 Pbyte per a aquesta finalitat. Els valors estadístics d'utilització real indiquen que no supera el 50 % (el client sempre se sobreaprovisiona), i en valors reals està entre un 10-15 %, per la qual cosa si el recurs és virtualitzat i la provisió és dinàmica, el proveïdor no ha de disposar de l'1 Pbyte per donar inici als clients en el servei; d'aquesta manera, si les necessitats van augmentant amb el temps i el sistema és escalable, es podran anar augmentant els recursos en funció de les necessitats reals dels clients, sense tenir «atrapat» 1 Pbyte de disc. Es pot dir el mateix del còmput i la xarxa.

Per aquest motiu, la virtualització és la tecnologia base (i essencial) per al *cloud computing*, perquè abstreu l'usuari de manera total del que hi ha «a sota»; de fet, l'usuari tampoc no sap on s'està executant el seu servidor, i el millor d'això és que tampoc no li cal, perquè haurà demanat unes prestacions i un servei, garantits per un SLA, i els tindrà a un preu acceptable. Per al proveïdor serà un negoci amb un compte de resultats positiu, amb eficiència, amb una reducció de costos que podrà traslladar als clients, amb una disminució de l'espai i el consum d'energia, amb el manteniment dels recursos aïllats (privacitat), amb agilitat en la gestió i l'aprovisionament i amb escalabilitat (premissa que si «necessito més», puc «afegir-hi més»).

Per això, la virtualització ens permet:

- **Incrementar la utilització dels recursos físics**: l'espai de disc és un cas, però també si un servidor s'utilitza poc i un altre molt, puc moure càrrega d'un servidor a un altre, i en alguns casos «en execució», cosa impensable en un servidor físic.

- **Consolidar els recursos i l'escalabilitat**: més enllà dels discos, que és un cas habitual, es podrà fer el mateix amb servidors de tota mena, però també es podrà estar obert al fet que, si es necessita més, es podran afegir més recursos, cosa impossible amb un supercomputador.

- **Utilitzar menys espai**: vinculat al punt anterior, menys servidors és igual a menys espai, i això està relacionat amb l'escalabilitat, perquè si el negoci creix en ampliar un centre de dades, sol ser una inversió molt gran, per la qual cosa tot espai que es pugui optimitzar és bo.

- **Augmentar l'estalvi energètic i l'eficiència**: normalment, als centres de dades, la relació sol ser que per cada kW gastat en còmput es gasta un altre kW en refrigeració, per la qual cosa reduir el nombre de servidors es traduirà directament en una reducció de costos.

- **Aconseguir més agilitat i una reducció de l'administració**: tenir MV preparades per a les diferents opcions o configurar-les en línia redueix notablement les tasques d'instal·lació i administració (perquè serà centralitzada), i això comporta generalment un autoservei per part de l'usuari, amb la conseqüent reducció d'atenció per part del proveïdor, la qual cosa genera satisfacció en el client i reducció de recursos humans en el proveïdor.

- **Obtenir una alta disponibilitat, resguards i recuperació**: en essència, un servidor virtualitzat no és res més que un arxiu, el qual es pot clonar, copiar o, fins i tot en hipervisors moderns, moure en calent, la qual cosa permet estratègies de diferents tipus per a l'alta disponibilitat, QoS i resguards.

- I, finalment, tot això repercuteix en la **reducció dels costos** de gestió, manteniment i escalabilitat i les millores en el TCO (*total cost of ownership*) i el ROI (*rendibilitat de la inversió*).

No obstant, la virtualització no està lliure de riscos, que poden provenir de:

- Les limitacions del maquinari: s'haurà d'analitzar molt bé la càrrega i monitorar per evitar-ne la sobreutilització, ja que s'haurà de complir una QoS acordada en un SLA.

- Les plataformes de virtualització (sobretot de tipus 1 *bare-metal*) que no són compatibles amb tot el maquinari: s'haurà de consultar per a quin maquinari està certificada, per evitar problemes de controladors o funcionalitats no permeses.

- Les necessitats tecnològiques: els sistemes virtualitzats poden necessitar recursos tecnològics nous i la plataforma de virtualització pot no estar preparada per habilitar-los.

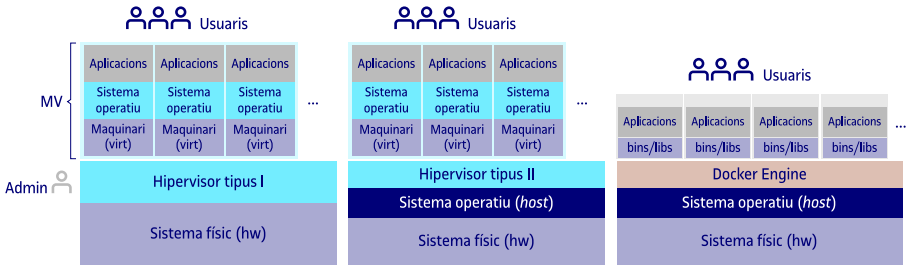
- I, finalment, el més probable i habitual: una fallada del maquinari del servidor físic, que afecta tots els servidors virtualitzats i que s'haurà de solucionar amb una alta disponibilitat dels servidors físics (redundància). En aquest cas, s'incrementarà el cost, però es podrà repercutir al client per mitjà de l'SLA corresponent.

Com ja es va esmentar anteriorment, l'**hipervisor** (o monitor de màquina virtual, VMM o *virtual machine monitor*) és la capa d'abstracció per a la virtualització que podrà actuar com un nucli indivisible amb el sistema operatiu o separat d'aquest com una aplicació més (però treballant-hi conjuntament). L'hipervisor mostrarà a les màquines virtuals (*guest*) una infraestructura virtualitzada que tindrà la contrapartida a la màquina sobre la qual s'executa (*host*), i que permetrà que el *guest* «vegi» aquesta infraestructura com si es tractés del maquinari físic. L'hipervisor tindrà com a tasques, a més, gestionar les màquines virtuals, assignar els recursos, permetre l'accés al maquinari equivalent, monitorar i comptabilitzar els recursos gastats per cada MV i tot un conjunt d'accions que seran necessàries per gestionar i administrar el sistema.

Avui dia podem disposar d'hipervisors de **tipus 1** (també anomenats *nadius*, *unhosted* o *bare-metal*), com per exemple VMware ESXi, XenServer i Xen, entre altres, en què l'hipervisor forma un conjunt indivisible amb el SO, i de **tipus 2** (també anomenats *hosted*), com per exemple VMware Workstation Player, VirtualBox, Qemu, KVM (tot i que alguns autors el consideren de tipus 1) i Hyper-V, en què l'hipervisor s'executa com una capa que interactua amb el SO, però que no hi està inclòs (excepte KVM).

És important recordar que la majoria d'ells només es poden executar en processadors amb extensions de maquinari VT-x/AMD-V, i que aquestes seran necessàries perquè els *guests* puguin ser de 64 bits (a més, en el tipus 2, el SO també haurà de ser de 64 bits). Per altra banda, és important destacar el paper que exerceix la virtualització de el SO, perquè avui dia és una tendència a l'alça (molts PaaS i SaaS ara només treballen amb contenidors), i el predomini de Docker i LXC/LXD en aquest context.

S'ha de tenir en compte que, així com amb els hipervisors de tipus 1 o 2, el sistema operatiu *guest* pot ser de qualsevol altra mena (i podrà ser de 32 bits o 64 bits sempre que el *host*, si és de tipus 2, sigui de 64 bits), en el cas de la virtualització d'SO (contenidors), el *guest* només podrà ser del mateix tipus (excepte en Docker per a Windows), malgrat que podran ser distribucions diferents (és a dir, que no podem tenir un *host* Linux i un *guest* Windows, però sí que podem tenir un *host* Debian i un *guest* Fedora). Com s'ha esmentat, un cas especial de contenidors és el de Docker per a Windows, perquè permet tenir contenidors Windows, i com que W10 inclou *kernel* de Linux, també permet tenir contenidors de Linux. La figura següent mostra les capes de la virtualització.



A continuació es desenvoluparan diferents experiències d'instal·lació i configuració dels hipervisors més habituals, perquè són la base per a qualsevol sistema de *cloud computing*, amb l'objectiu d'analitzar-ne les prestacions i facilitats a l'hora de configurar i gestionar. La virtualització dels SO i els contenidors es deixen per a més endavant, quan es descriuran amb detall.

2. Casos d'ús

2.1. KVM

Kernel-based Virtual Machine (KVM) és un projecte *open source* que implementa sobre Linux (des de la versió 2.6.20/2007) una infraestructura de virtualització perquè Linux actuï com a hipervisor (segons alguns autors, de tipus 2, però quan ja està instal·lat és de tipus 1, segons uns altres), a través d'un mòdul (*kvm-intel/amd.ko*) carregable al *kernel* i eines a l'espai d'usuari per gestionar-lo.

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, discos durs, targeta gràfica, etc.). Només necessita un processador x86/64, compatible amb la virtualització (VT-X/AMD-V), i pot executar diferents SO *guest*, com ara Linux, Unix, OSX, Darwin i Windows, entre altres, tant de 32 com de 64 bits. A més, admet un conjunt de dispositius paravirtualitzats per a Linux, openBSD, FreeBSD i Windows, utilitzant l'API *virtio* [Kvm].

2.1.1. Instal·lació i configuració

KVM es pot instal·lar sobre una màquina física (*bare-metal*) o fins i tot, com a prova funcional, sobre una MV, però cal que l'hipervisor sigui compatible amb la virtualització imbricada, perquè KVM necessita «veure» les extensions de maquinari del processador. Les proves per a aquest apartat s'han dut a terme sobre una màquina amb processador Intel i7 620M (64 bits, 2 *cores* + Hyper-Threading = 4 processos) i sistema operatiu Ubuntu 21.04 LTS. Ubuntu utilitza KVM com a tecnologia per a la virtualització de *back end*, principalment per a servidors no gràfics, i la llibreria libvirt com a *toolkit/API*. libvirt actua com a *front end* per gestionar les màquines virtuals a través de `virt-manager` (`GUI`) o `virsh` (`CLI`).

A continuació es descriuen els passos per verificar si el *host* és compatible amb la virtualització i si disposa de el SO adequat. Per verificar si el processador admet les extensions de maquinari, s'ha d'executar:

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

Si és 0, significa que la CPU no admet la virtualització, i si és 1 o més (4, en el nostre cas), sí que l'admet (en cas que sigui 0, s'ha de verificar que aquestes extensions no estiguin deshabilitades des del BIOS, perquè molts fabricants trien aquesta opció per defecte). També es pot instal·lar el paquet amb l'ordre `apt-get install cpu-checker` i executar `kvm-ok`; amb això ens indica-

rà si és possible o no executar KVM sobre aquesta arquitectura. Cal tenir en compte que una arquitectura que no admeti KVM podrà executar VM, però sense acceleració, solament per emulació.

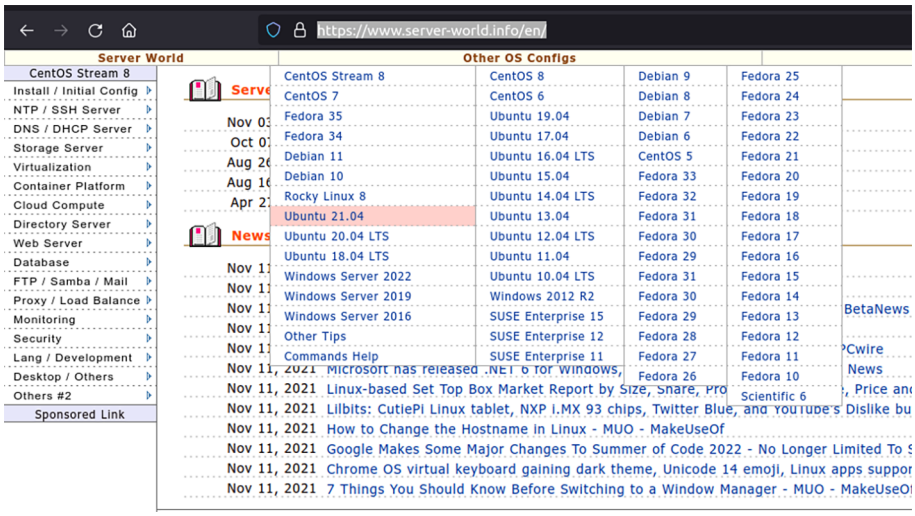
Si es necessita rendiment, és aconsellable tenir un *kernel* de el SO de 64 bits (i si es vol una MV amb més de 2 GB de RAM, el *kernel* haurà de ser de 64 bits). Un *kernel* de 64 bits podrà acollir SO *guests* de 32 o 64 bits, mentre que si el *host* és de 32 bits, el *guest* només podrà ser de 32 bits. Per verificar si el processador és de 64 bits, cal executar:

```
egrep -c ' lm ' /proc/cpuinfo
```

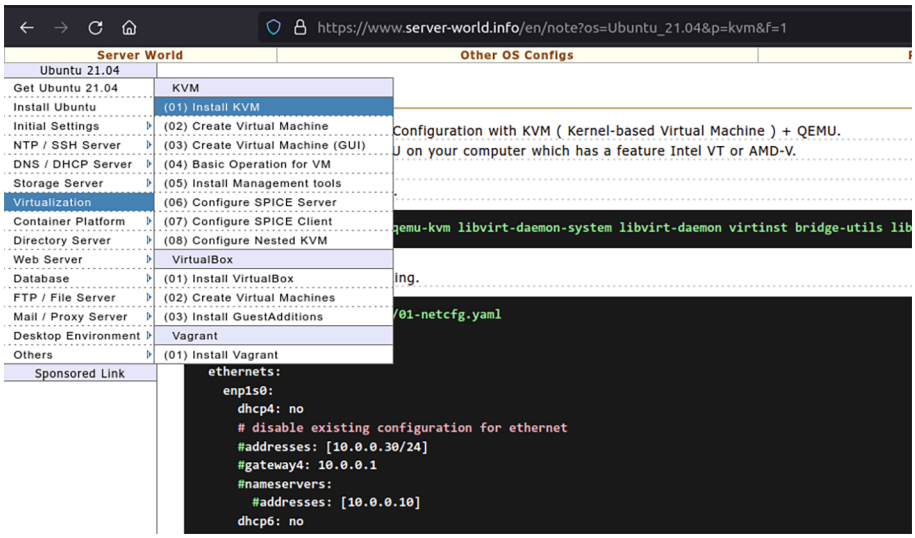
Si el resultat és 0, vol dir que no; si és 1 o més gran, vol dir que sí; lm significa *long mode*, equivalent a CPU de 64 bits si és més gran que 0.

Per verificar si el SO *host* és de 64 bits, cal executar `uname -m`, que indicarà x86_64 (o amd64). Si indica i386, i486, i586 o i686, el *kernel* és de 32 bits, i si el processador és de 64, s'hauria de canviar.

Atès que hi ha diferències entre les instal·lacions de KVM per a les diferents distribucions, recomanem documentar-se sobre el procés, en funció de la distribució i versió de què es disposi de el SO *host*, a Server World. Els autors de Server World mantenen actualitzats, per a les principals distribucions i últimes versions, els procediments d'instal·lació d'una manera detallada i ben documentada, per la qual cosa és recomanable seguir els passos d'instal·lació duts a terme per aquests experts. Quan s'accedeix a la pàgina, es mostren les principals notificacions i SO actualitzats, i des del selector d'*Other* OS *Configs* es pot triar la versió adequada de el SO disponible al nostre *host*, com es mostra a la figura següent.

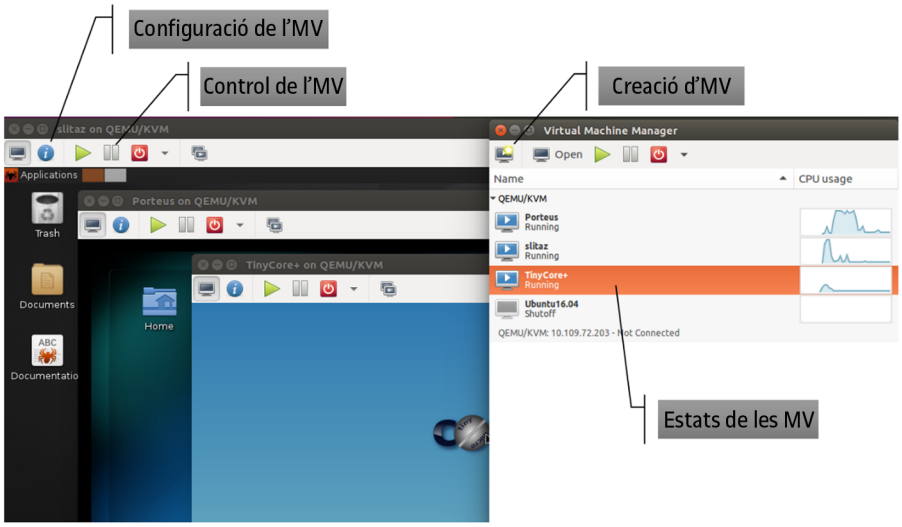


A la figura següent es pot observar, per a Ubuntu 21.04, el selector del menú a *Virtualization* -> *Install* KVM i el procés per seguir a fi d'instal·lar KVM sobre aquest SO.



S'instal·len les diferents llibreries de libvirt i QEMU necessàries per administrar instàncies qemu i kvm, i bridge-utils permet configurar un *bridge* des de la xarxa cap a les VM. A les opcions següents del menú es pot accedir als processos pas per pas a fi de crear una MV des d'un terminal o amb una interfície gràfica, gestionar una MV, instal·lar eines complementàries per gestionar les MV o configurar un servidor-client SPICE per accedir remotament a les MV. Finalment, es pot configurar KVM perquè les MV vegin les extensions de maquinari del processament (el que es coneix com a *virtualització imbricada*).

La figura següent mostra el `virt-manager` executant tres màquines virtuals amb diferents instal·lacions de Linux (TinyCore+, Proteus, SliTaz). A través d'aquesta aplicació o amb `virsh`, es podran crear, gestionar (posar en marxa, reiniciar, aturar, etc.) i administrar totes les opcions de la màquina virtual (CPU, xarxa, pantalla, memòria, discos, etc.).



A més de les opcions abans esmentades de SPICE, el `virt-manager` permet gestionar l'MV tant en local com **en remot** (és a dir, des d'una màquina diferent d'on s'estan executant les MV). Per a això, a la màquina que es vol connectar a l'MV s'hi ha d'instal·lar:

```
apt-get install virt-manager ssh-askpass-gnome --no-install-recommends
```

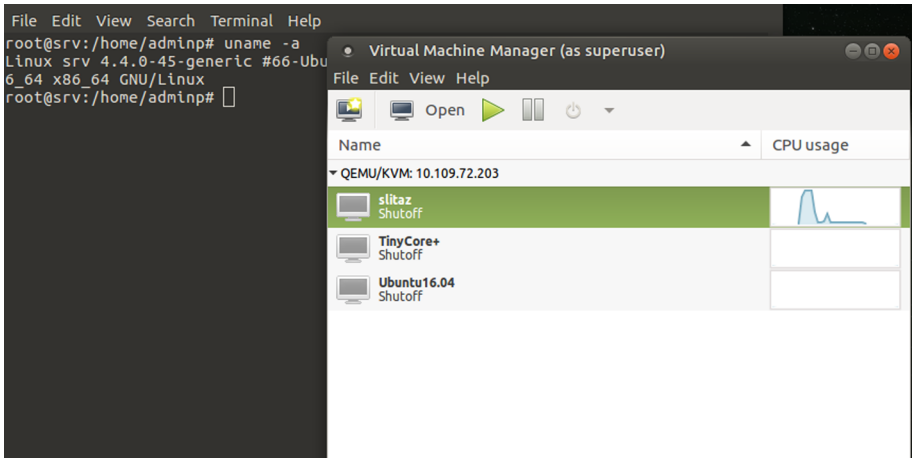
És important el paquet `ssh-askpass-gnome`, o qualsevol dels `ssh_askpass*`, perquè la connexió amb el servidor serà a través de `ssh` i es necessita validar la sessió amb l'usuari i *passwd* del servidor libvirt, i això es farà per mitjà d'aquest paquet.

S'inicia el `virt-manager` i al menú *File* → *Add Connection* → *Hypervisor=QEMU/KVM, Username* = usuari remot dins del grup de libvirt en *host* remot, *Hostname* = IP del *host* remot.

Amb això, es veurà la connexió, se sol·licitarà el *passwd* per a l'usuari indicat i ja es podrà treballar exactament com si fos en local, però des del *host* remot. La figura següent mostra la connexió remota, que té el format:

```
quemu+ssh://user_libvirt@ip_host_remoto/system
```

S'aplica a una màquina diferent de la que està executant les màquines virtuals.



Això mateix es pot fer en la CLI executant en un terminal:

```
virsh -c quemu+ssh://user_libvirt@ip_host_remoto/system
```

Demanarà el *passwd* i podrem accedir a l'administració remota (mostrarà el *prompt virsh #*, en què l'ordre *help* ens donarà totes les possibilitats de gestió del `virsh`).

2.1.2. *Networking*

Hi ha dues maneres diferents de permetre que una MV accedeixi a una xarxa externa. En la manera per defecte, coneguda com a *usermode networking*, fa un NAT a través de la interfície del *host* cap a la xarxa externa. Si es vol accedir a serveis sobre SO *guest* d'una MV, cal configurar una *bridged networking*.

Al ***usermode networking*** (per defecte), les màquines es configuren perquè puguin sortir cap a la xarxa externa (internet), els SO *guest* obtindran IP en el rang 192.168.122.0/24 i el SO *host* tindrà 192.168.122.1. Des de el SO *guest* es pot accedir al *host* (per exemple, per compartir arxius o qualsevol altra acció) per mitjà de ssh 198.168.122.1, o des del *host* a la IP assignada dins de 192.168.122.0/24 al *guest*. Si els SO *guest* no tenen accés al *host* o la xarxa externa, cal verificar amb `sudo iptables -n -t nat -L` si s'està fent el NAT, en què s'haurà de veure (entre altres):

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
MASQUERADE all -- 192.168.122.0/24 !192.168.122.0/24

Si aquesta regla no existeix, s'hauran de parar totes les MV i recrear les regles amb

```
virsh net-destroy default
virsh net-start default
```

Si es perd la connectivitat durant grans transferències (per exemple, durant un `rsync`), una possible solució és habilitar el *driver* de xarxa *virtio* (*driver* paravirtualitzat i necessari per a Windows).

Es pot obtenir més informació sobre aquest mode i com funciona a la pàgina de libvirt [Lib].

La ***bridged networking*** permet connectar-se, a través de les interfícies virtuals, a la interfície física i, per tant, ser visibles des de fora com si es tractés d'una màquina habitual. S'ha d'anar amb compte, perquè si el dispositiu físic sobre el qual es fa el *bridge* és sense fil (wifi), no funcionarà (molts dispositius *wireless* no admeten el *bridging*). En les distribucions que executen el NetworkManager, és necessari deshabilitar aquest procediment per evitar que interfereixi en la configuració (si cal, es pot fer amb `systemctl stop NetworkManager; systemctl disable NetworkManager`). Si s'executa `brctl show`, veurem:

bridge name bridge id STP enabled interfaces
virbr0 8000.5254002bc4ce yes virbr0-nic

Si dona errors, cal verificar que es disposa del paquet bridge-utils (i si no, s'haurà d'instal·lar). L'execució d'`ip address` donarà:

1: el:<LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1 inet 127.0.0.1/8 scope host lo ...
2: enp0s25:<BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master br0 state UP group default qlen 1000 ...
4: virbr0:<NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000 link/ether 52:54:00:2b:c4:ce brd ff:ff:ff:ff:ff:ff inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
5: virbr0-nic:<BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN group default qlen 1000 link/ether 52:54:00:2b:c4:ce brd ff:ff:ff:ff:ff:ff

On virbr0 és la interfície NAT de KVM i enp0s25, la primera interfície física del *host*. Per instal·lar el *bridge* manualment (recordeu que en les instal·lacions de Server World es configura el Netplan habitual en la distribució Ubuntu, però es pot mirar a Debian 11 per exemple, on la instal·lació és manual), es poden trobar més detalls a [Lib,Kne]. En el cas de la configuració manual, primer es baixarà la interfície de maquinari (*enp0s25*, en aquest cas d'estudi) `ifdown enp0s25`, i a continuació es modificarà l'arxiu */etc/network/interfaces* com:

```
auto lo auto br0
iface lo inet loopback

iface enp0s25 inet manual

iface br0 inet static
    address 158.109.65.67
```

```
    netmask 255.255.255.0
    gateway 158.109.64.1
    dns-nameservers 158.109.0.1 158.109.0.9
    bridge_ports enp0s25
    bridge_stp on
    bridge_fd 0
    bridge_maxwait 0
```

En aquest cas, s'ha utilitzat una xarxa amb IP públiques, però pot ser perfectament una xarxa amb IP privades i que després hi hagi una màquina que faci NAT cap a la xarxa pública. Els paràmetres de *bridge* indicats són els següents: *bridge_stp off|on* és la configuració per a l'arbre d'expansió (malgrat que pot causar errors en l'assignació de DHCP als *guests*; si passa això, cal posar-lo en *off* o ometre'l), *bridge_fd 0* indica *turns off all forwarding delay* i *bridge_maxwait 0* és el temps que el sistema esperarà perquè els ports estiguin disponibles (0 = no espera).

A continuació executem `ifup br0`, i amb `brctl show` veurem la nova interfície (que també podrem verificar amb `ip address`):

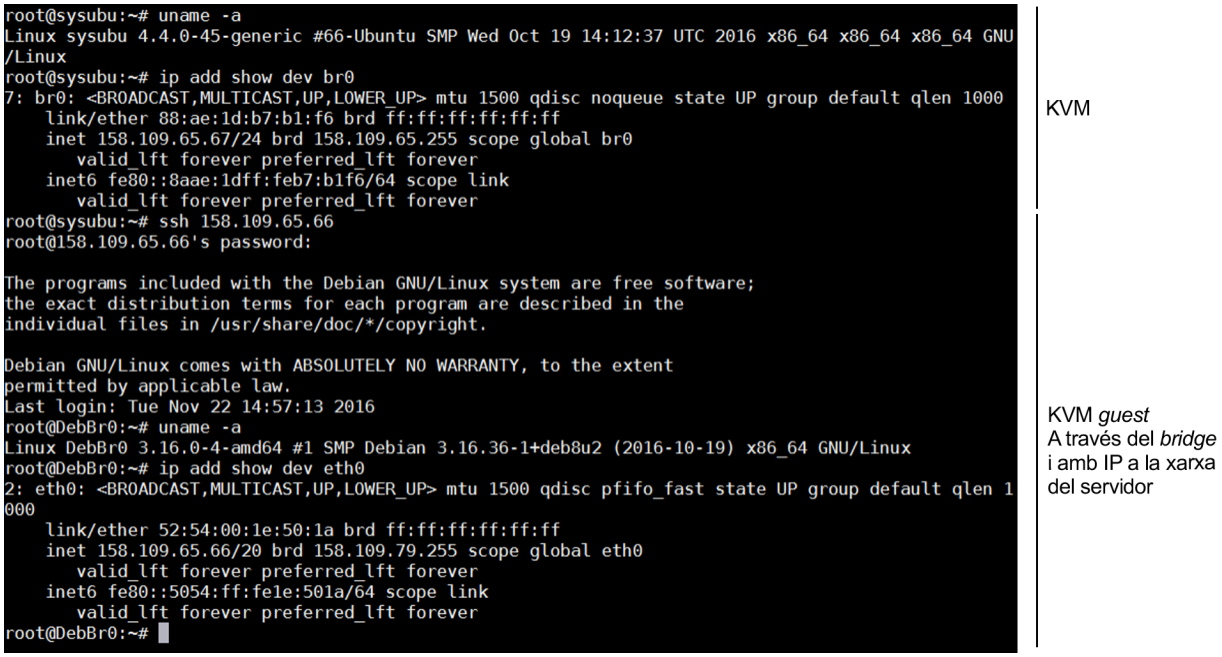
bridge name bridge id STP enabled interfícies
br0 8000.88ae1db7b1f6 yes enp0s25
virbr0 8000.5254002bc4ce yes virbr0-nic

A continuació es pot verificar la connectivitat i començar a crear MV utilitzant aquesta interfície, tant en el mode gràfic com en el mode CLI (si no es disposa de DHCP al servidor, l'assignació d'IP i d'altres paràmetres al *guest* haurà de ser manual). Si es vol modificar una màquina que ja s'ha creat, es pot fer amb el `virsh edit nom-MV`; per exemple, per indicar (dins de dispositius, i la mac es pot ometre):

```
<interface type='bridge'>
   <mac address='52:54:00:1e:50:1a'/>
   <source bridge='br0'/>
   <model type='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
  </interface>
```

libvirt disposa de moltes opcions per configurar la xarxa, a més de les que s'han mostrat, com ara assignar una targeta directament a un *guest* i fer una xarxa encaminada, una xarxa NAT específica o múltiples xarxes, com es pot consultar a [Vnh].

La figura següent mostra la connexió des d'una màquina externa al servidor (Ubuntu sysubu) i després al *guest* (Debian DebBr0).



2.1.3. Creació de màquines virtuals (*guests*)

És un procediment que es du a terme bàsicament amb la GUI, com s'ha vist anteriorment, o amb CLI (`virt-install`).

Com a prova de concepte, s'instal·larà una distribució Windows, per la qual cosa s'ha de comptar amb l'ISO o el DVD de la distribució. En aquest exemple, es disposa del DVD de Windows 8.1 i s'utilitzarà aquest, però és el mateix procediment si es disposa del DVD de Windows 10. Per això, es crea la imatge amb:

```
dd if=/dev/dvd of=/var/lib/libvirt/images/win.iso
```

Després s'utilitza l'ordre `virsh-install` per instal·lar la màquina virtual (és molt potent i s'han de consultar les opcions de què disposa) introduint en una única línia (abans de cada opció, cal indicar '- -'):

```
virt-install --name=win --ram=2000 --cpu=host --vcpus=2
--os-type=windows
--os-variant=win8.1
--disk
/var/lib/libvirt/images/w8.qcow2,bus=sata,size=20,format=qcow2
--disk /var/lib/libvirt/images/win.iso,device=cdrom,bus=ide --network bridge=virbr0
--graphics vnc,listen=0.0.0.0
```

Es poden utilitzar les opcions - *-acelerate*, i millora molt buscant els *virtio drivers* que es poden baixar i compilar o obtenint l'ISO per a la distribució volguda.

A Ubuntu, es baixaria l'última (virtio-win-drivers-xxxx-yyyy.iso) i es canviaria en la creació l'opció del disc, posant *bus=virtio* i continuant amb la instal·lació. Quan la instal·lació s'aturi a la pàgina d'emmagatzematge, hem de carregar els *drivers* SCSI del DVD. Dins de `virsh` s'executa:

S'obté el domini ID:

```
virsh# list
```

Unitats carregades:

```
virsh# domblklist ID
```

S'extreu el DVD:

```
virsh# change-media win81 hdc --eject
```

S'insereix el nou:

```
virsh# change-media win81 hdc /var/lib/libvirt/images/virtio-win-0.1-81.iso --insert
```

Es veuran tres *drives* (*network*, *scsi* i *balloon driver*): cal triar scsi driver.

Ara s'haurà de tornar a posar el disc d'instal·lació:

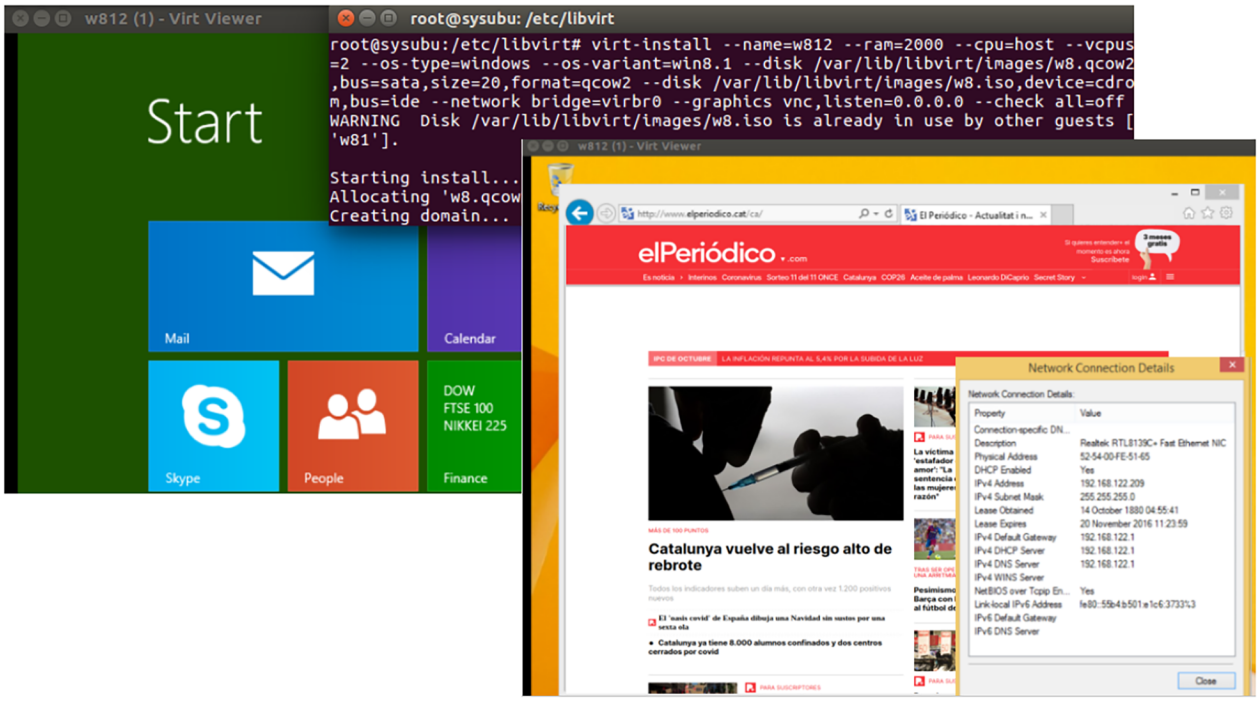
```
virsh# change-media win81 hdc /var/lib/libvirt/images/virtio-win-0.1-81.iso --eject
```

S'insereix el de Windows:

```
virsh # change-media win81 hdc /var/lib/libvirt/images/win8.iso --insert
```

S'hauria de repetir el mateix per actualitzar el *driver* de xarxa.

Les dues figures següents mostren l'execució de W8.1 sobre Ubuntu i les configuracions de xarxa (en aquest cas, s'ha utilitzat NAT).



Altres **ordres útils amb virsh** (CLI):

Connectar-se:

```
virsh --connect qemu:///system
```

Fer la llista de les màquines:

```
virsh# list --all
```

Iniciar la màquina Windows (també es pot utilitzar *suspend, resume, shutdown, destroy* –apaga l'MV per la força– per gestionar les MV):

```
virsh# start win81
```

Des d'una altra terminal, es pot accedir a la seva interfície amb:

```
virt-viewer --connect qemu:///system win81
```

Des d'una altra màquina:

```
virt-viewer --connect qemu+ssh://ip_servidor/system win81
```

Clonar una MV des d'original a còpia:

```
virt-clone --connect qemu:///system -o original -n copia -f /var/lib/libvirt/copia.qcow2
```

La configuració de cada màquina virtual s'emmagatzema en un arxiu del tipus XML a */etc/libvirt/qemu/* amb el nom de la màquina; es pot modificar primer exportant l'arxiu de l'MV (per exemple, nteum):

```
virsh dumpxml nteum > /tmp/nteum.xml
```

S'edita l'arxiu modificant-ne les opcions (CPU, RAM, discos, etc.) i després s'importa.

```
virsh define /tmp/nteum.xml
```

Altres **ordres útils**
(paquets `apt-get -i install libguestfs-tools virt-top`) [Ksw]:

```
virt-ls -l -a /var/lib/libvirt/images/debina8.qcow2
virt-ls -l -d Ubuntu20.04
virt-cat -a /var/lib/libvirt/images/debina8.qcow2 /etc/passwd
virt-edit -d Ubuntu20.04 /etc/fstab (amb la màquina apagada per evitar incoherències en el disc)
virt-df -d Ubuntu20.04
virt-top
guestmount -d ubuntu -i /mnt; ll /mnt (montar el disco de la VM)
```

Per fer una **migració «en viu»** des d'un servidor KVM a un altre, es requereix que tots dos comparteixin un disc amb les imatges de l'MV (per exemple, en aquest cas, per NFS, però pot ser iSCSI o GlusterFS). Els servidors en aquest exemple són *syskvm1.nteum.org* i *syskvm2.nteum.org*, i, a més, *nfs.nteum.org*, que està muntat sobre els dos anteriors a */var/lib/libvirt/images*. En aquest exemple es migrarà la màquina Ubuntu20.04 des de *syskvm1* a *syskvm2*. Per a això, s'ha d'executar:

```
virsh migrate --live Ubuntu20.04 qemu+ssh://syskvm2.nteum.org/system
```

L'execució demanarà el *passwd* de *root* de *syskvm2* i mourà la màquina d'un entorn a un altre, la qual cosa es podrà verificar amb `virsh list` en cada màquina (sobre *syskvm1* es veurà el missatge *just migrated* i sobre *syskvm2*, la màquina en execució).

KVM també pot fer **migració de l'emmagatzematge** quan migra una màquina virtual, amb l'avantatge que no cal tenir un disc compartit com en el cas anterior. Només amb els dos servidors (*syskvm1* i *syskvm2*) n'hi ha prou. Per a això, primer s'ha de generar el mateix espai a la destinació i a l'origen executant sobre *syskvm1*.

```
ll /var/lib/libvirt/images/deb*
-rw------- 1 root 3221946368 dic 2 12:28 debian8.img
```

Sobre *syskvm2* es crea l'espai per allotjar aquesta imatge:

```
fallocate -l 3221946368 /var/lib/libvirt/images/debian8.img
```

Es verifica sobre *syskvm2* que s'ha creat:

```
ll /var/lib/libvirt/images/
-rw------- 1 root root 3221946368 dic 2 12:34 debian8.img
```

Ja es pot fer la migració des de *syskvm1*:

```
virsh migrate --live --copy-storage-all debian8 qemu+ssh://syskvm2.nteum.org/system
```

Demanarà el *passwd* de *root* sobre syskvm2 i es podrà comprovar la migració. Si es vol capgirar i tornar la màquina a *syskm1*, només cal executar una migració «en viu»:

```
virsh migrate --live debian8 qemu+ssh://syskvm1.nteum.org/system
```

2.1.4. Virtualització imbricada (*nested KVM*)

En la virtualització, el *host* Linux físic (*bare-metal*) té instal·lat KVM com a hipervisor i executa diversos sistemes operatius. Si s'analitzen les màquines virtuals per defecte, no disposen de les extensions de maquinari del processador, per la qual cosa no es podria instal·lar un altre hipervisor que ho requerís sobre aquestes MV; es pot comprovar executant:

```
grep --color vmx /proc/cpuinfo
```

Sobre una màquina que en disposi es veurà la paraula **vmx** de color, mentre que si no estan actives no es veurà res. De vegades, és útil disposar d'aquestes extensions sobre les màquines virtualitzades, i KVM permet el que s'anomena **virtualització imbricada** (*nested virtualization*), la qual cosa habilita per executar un *guest* dins d'un hipervisor virtualitzat que s'està executant sobre l'hipervisor base (*bare-metal*). Això és útil quan en un servidor es presta servei a diferents usuaris que, al seu torn, volen tenir diferents MV (per exemple, en el *cloud*); quan es vol provar o donar el servei de diferents hipervisors sobre un mateix maquinari, o quan es volen depurar o analitzar diferents configuracions sobre un determinat hipervisor. En les últimes versions de KVM aquest suport està activat per defecte, però es pot comprovar amb:

```
cat /sys/module/kvm_intel/parameters/nested
```

Es veurà si la resposta és **Y** (està activat), i en */etc/modprobe.d/qemu-system-x86.conf* es podrà observar que hi ha una opció *options kvm_intel nested=1*. Si la resposta és **N** (desactivat), es pot activar amb:

```
echo 'options kvm_intel nested=1' >> /etc/modprobe.d/qemu-system-x86.conf
```

I després es reinicia el sistema. Per modificar una MV (per exemple, la màquina Ubuntu20.04) perquè vegi les extensions de maquinari, s'executa (al directori */etc/libvirt/qemu* es poden consultar els noms de les màquines, perquè és on hi ha els arxius amb els seus noms que contenen les definicions XML de cadascuna d'elles):

```
virsh edit Ubuntu20.04
```

I es canvia *cpu mode='host-passthrough'* (per defecte, es configuren les MV amb *cpu mode='custom'*). Un cop que hagi arrencat, es podrà veure que la màquina ara disposa de l'extensió **vmx** quan es mira */proc/cpuinfo*.

2.1.5. Administració remota

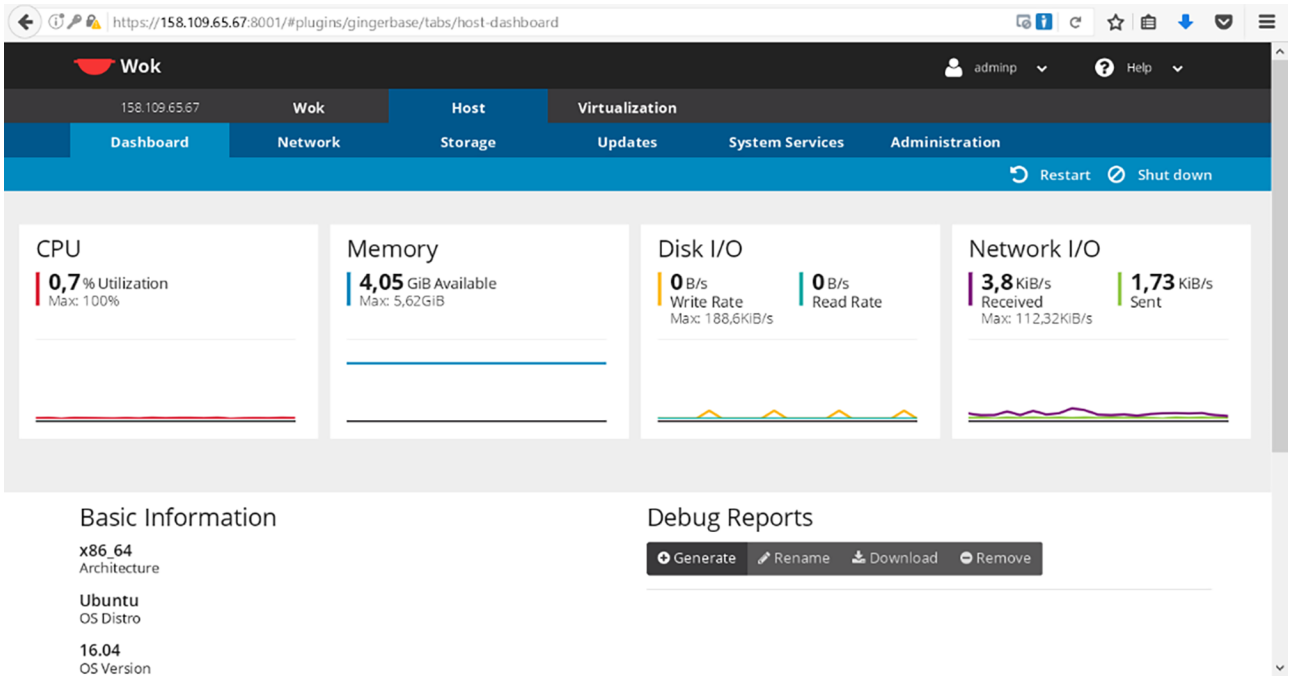
Hi ha una gran quantitat d'eines (a més de la CLI i `virt-manager`) per gestionar arquitectures basades en KVM (si bé moltes d'elles també poden gestionar la infraestructura IaaS completa, altres ja estan obsoletes). Els usuaris que vulguin provar aquesta mena d'eines poden experimentar amb Kimchi.

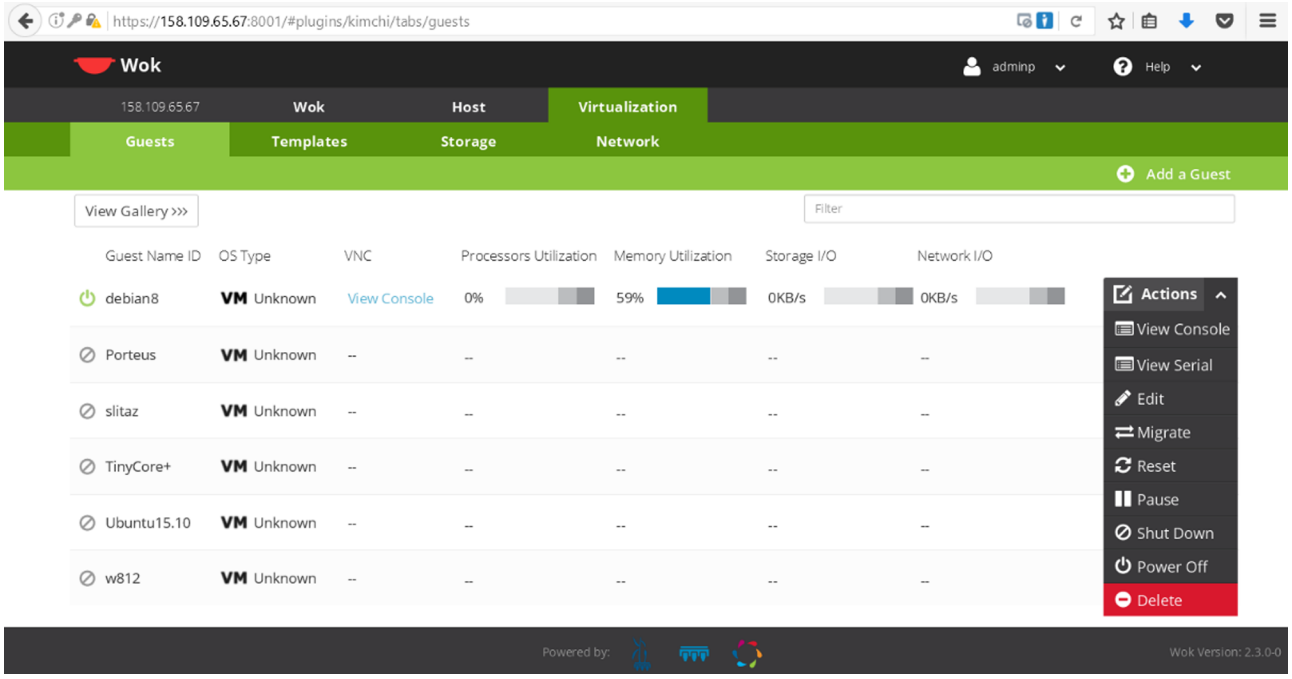
Kimchi és una aplicació excel·lent per a l'administració i el monitoratge remots de KVM a HTML5 (així com del mateix *host*). Els objectius de disseny són que la gestió tant del *host* com de KVM siguin simples i fàcils i des de qualsevol lloc. Kimchi s'executa com un *plugin* de Wok i controla els *guests* KVM a través de libvirt. A la interfície de gestió s'hi accedeix mitjançant un navegador compatible amb HTML5. Instal·lar-la requereix una mica d'atenció, però és factible seguint els passos indicats a la pàgina web del projecte, que disposa dels paquets en diferents distribucions ja compilats (primer s'instal·la Wok i després Kimchi).

Si el sistema sobre el qual s'està instal·lant no té totes les dependències necessàries, fallarà; s'haurà d'executar (també es poden baixar les dependències per a cadascun dels paquets executant `apt-get install -f`). També s'ha de recordar que, després d'instal·lar, s'haurà de reiniciar el servei amb `systemctl restart wokd.service`.

Per accedir a l'entorn, s'haurà d'escriure al navegador com a URL *<https://machine-ip o machine.name.domain:8001>*, on s'haurà d'introduir l'usuari Linux i el *passwd* de Linux per a aquest usuari. Si aquesta operació es fa una vegada instal·lat Wok, només mostrarà les propietats del *host* i permetrà administrar-ne tots els recursos, i quan s'hagi instal·lat Kimchi (i reiniciat el servei), es podrà veure tant la part del *host* com la part de KVM i gestionar totes les màquines que estiguin sota el seu domini. Aquest paquet és de molt alta qualitat; la seva visualització és molt bona, igual que la seva estabilitat, i és recomanable en entorns tant mitjans com grans. Els paquets estan en desenvolupament actiu (hi ha actualitzacions del 2020), i se'n pot trobar més informació al web

del projecte. A les figures següents es mostra primer el *dashboard* (càrrega, recursos, configuració del *host*) i després l'entorn de virtualització amb les seves càrregues i recursos sobre KVM+libvirt.





Una altra opció simple és accedir a la màquina remotament (i si no es volen instal·lar clients VNC) o s'hi pot accedir des d'un dispositiu mòbil; és possible accedir-hi des d'un simple navegador utilitzant noVNC. noVNC és un client VNC basat en un navegador desenvolupat sobre HTML5-Canvas/WebSockets, i funciona amb un servidor VNC compatible amb WebSockets (com ara `x11vnc/libvncserver`) o pot utilitzar el mateix *websockify* que farà de *bridge*

entre el navegador i el servidor de VNC. noVNC és compatible amb els navegadors habituals (incloent-hi els d'iOS i Android), diferents codificacions de VNC (*raw, copyrect, rre, hextile, tight, tightPNG*), WebSocket SSL/TLS encryption (per exemple, wss://), el redimensionament de la pantalla d'acord amb la grandària del client (navegador), el cursor remot o local, copiar i enganxar i els desplaçaments verticals i horitzontals, entre altres característiques. Per instal·larlo simplement en la màquina, cal baixar-ne l'última versió (sobre Ubuntu està com a paquet, però no inclou l'última versió):

```
git clone https://github.com/novnc/noVNC.git
```

En aquest cas, s'utilitzarà el servidor per defecte d'Ubuntu, que és `vino`, i que no admet WebSockets, per la qual cosa s'utilitzarà el propi (*websockify*), però abans hem de deshabilitar l'encriptació de `vino` (les propietats es poden modificar amb `vino-preferences`, i per iniciar-lo sobre Ubuntu, simplement usem `/usr/lib/vino-server`), perquè l'algorisme que implementa no és compatible amb el noVNC; per a això s'ha d'executar des d'un terminal (de l'usuari que té l'entorn):

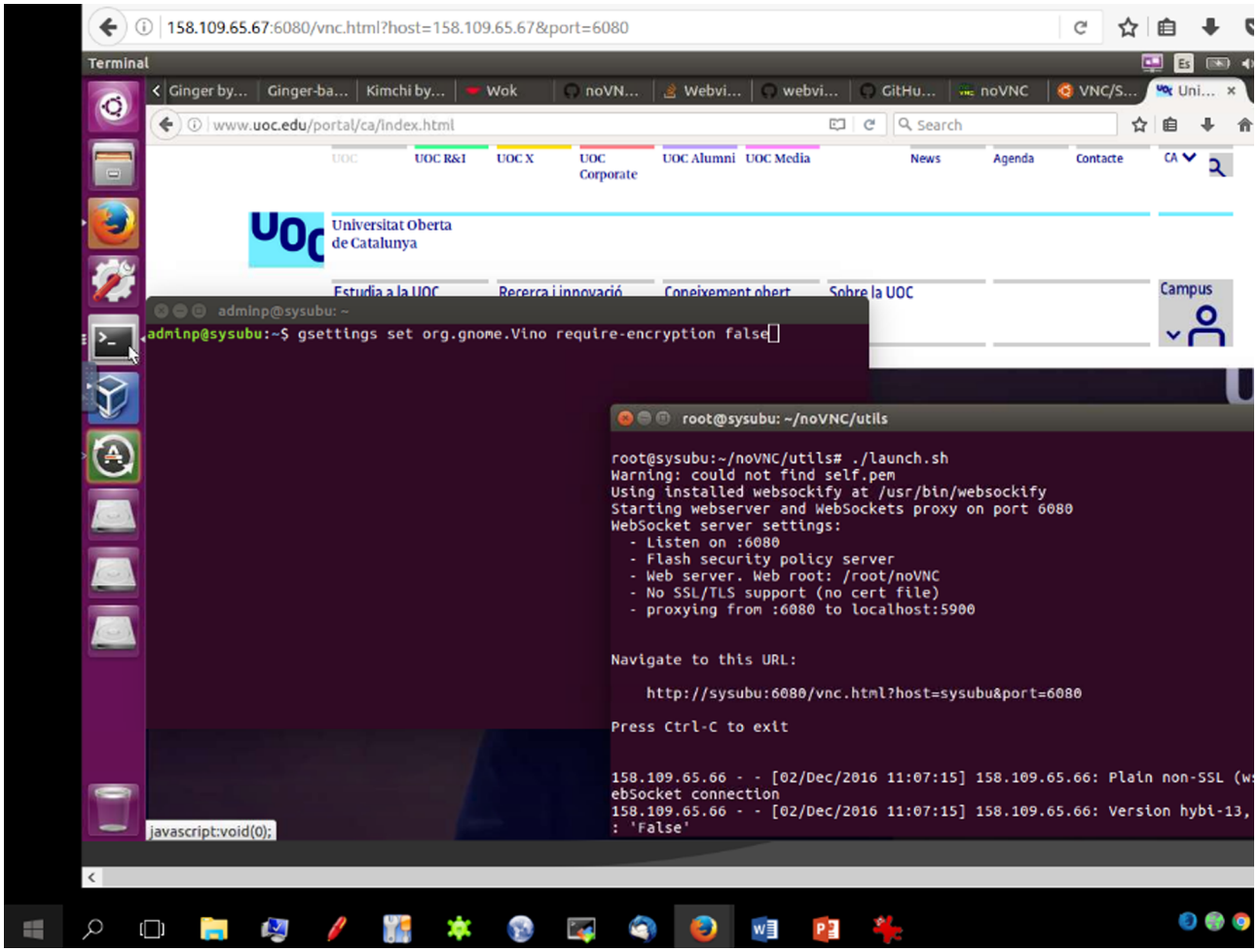
```
gsettings set org.gnome.Vino require-encryption false
```

Després s'executa un *script* que posa en marxa el servidor (i proveeix la URL de connexió; cal mirar els paràmetres de l'*script*, perquè es poden canviar els ports i altres configuracions).

```
cd ./noVNC/utils
./launch.sh
```

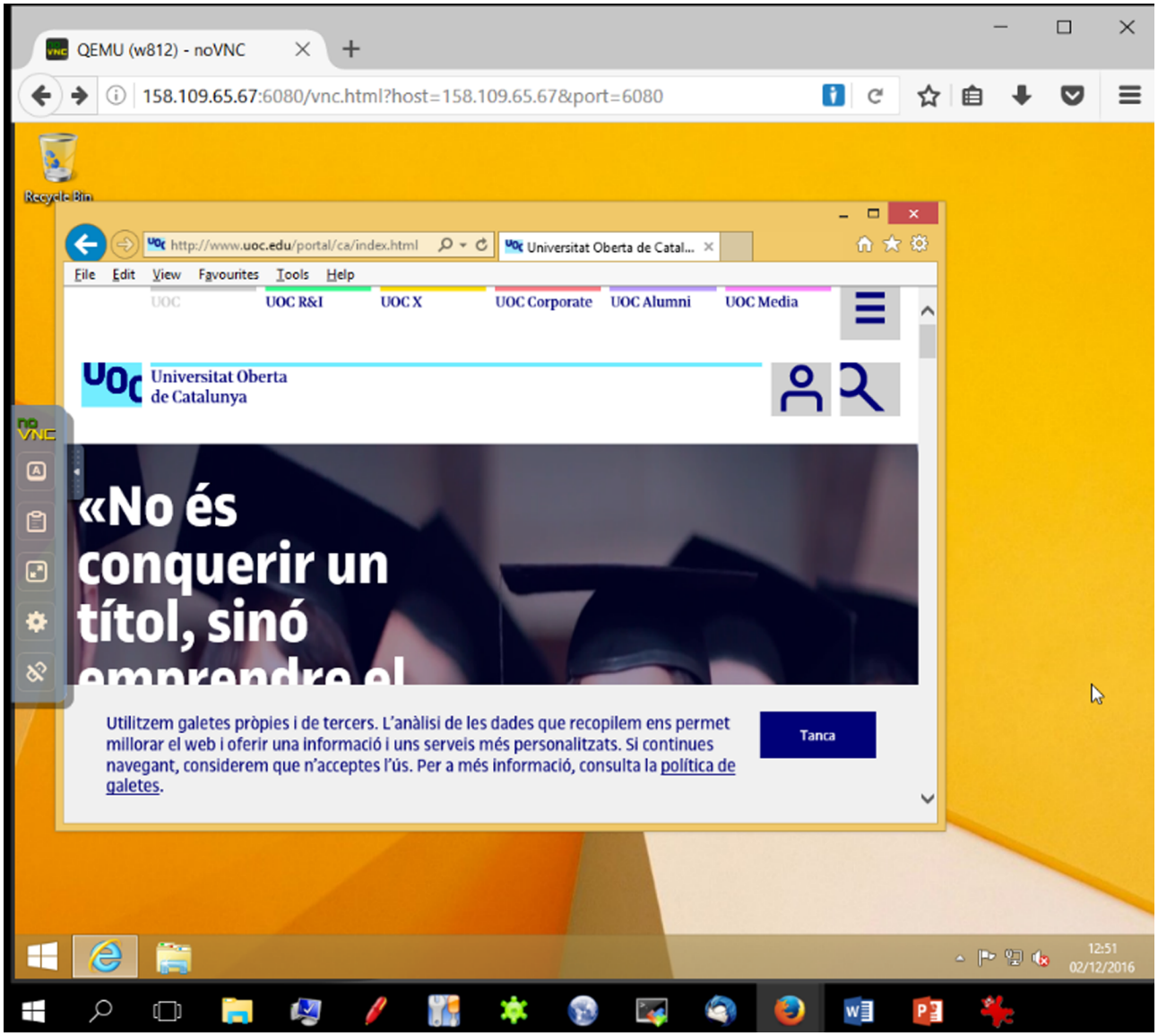
Atès que ens interessa accedir-hi remotament, executem (vegeu la figura següent, en què es mostra una connexió entre una màquina W10 i el servidor Ubuntu):

http://158.109.65.67:6080/vnc.html?host=158.109.65.67&port=6080



També es pot executar una MV amb KVM i activar el servidor de VNC al *guest*, tenint en compte que, si es vol connectar una màquina remota, cal canviar les configuracions de */etc/libvirt/qemu.conf*: `vnc_listen = "0.0.0.0"`

S'ha d'anar amb compte amb això, però, ja que qualsevol client es podria connectar al *guest* remotament i és necessari considerar aspectes de seguretat de `libvirt`. Finalment, reiniciant `libvirt` i posant en marxa el *guest*, es podrà accedir com abans a través de noVNC. La figura següent mostra la connexió amb un *guest* W8.1 des d'un W10 utilitzant noVNC sobre un navegador Firefox.



També és possible connectar-se als *guest* a través de l'entorn Spice [Spi]: seleccionant-lo a l'MV i, mitjançant l'ordre `remote-viewer`, indicant-li la URL *spice://localhost:5900*. Si no es disposa d'aquesta ordre, s'ha d'instal·lar el paquet `virt-viewer` (`apt-get install virt-viewer`). Igual que per a VNC, si es vol connectar remotament un *guest* per SPICE, cal modificar la configuració de */etc/libvirt/qemu.conf* per treure el comentari a spice_listen = *0.0.0.0*, i reiniciar el servei i la connexió.

A [Ksw] s'hi poden trobar referències addicionals sobre la instal·lació de Spice servidor-clients.

2.2. VirtualBox

Oracle VM VirtualBox és una infraestructura *free & open source* de virtualització (tipus 2) per a x86/64, creada per l'empresa Innotek GmbH (adquirida el 2008 per Sun Microsystems, la qual va ser adquirida per Oracle el 2010) i que es pot instal·lar sobre diversos *hosts* (Linux, OS X, Windows, OpenSolaris, FreeBSD

i Genode, entre altres). Admet la creació i gestió d'MV, pot utilitzar com a *guest* diferents SO (Linux, Windows, BSD, OS/2, Solaris, Haiku i OSx86, entre altres) i fa servir per a alguns SO un paquet addicional (en moltes distribucions de Linux ja està inclòs en el repositori, com per exemple en el de *non-free* i en altres en el *contribution*) anomenat *Guest Additions*, amb controladors que milloren les prestacions, les funcionalitats i els gràfics.

A partir de la versió 4.0, el nucli de VBox és GPLv2, però l'*Oracle VM VirtualBox Extension Pack* (compatible amb USB 2.0/3.0, RDP i PXE) té una llicència propietària *Personal Use and Evaluation License* (PUEL), que permet que el programari es faci servir per a ús personal, educatiu o avaluatiu sense càrrec. Per gestionar-lo, disposa d'una GUI, però tot es pot fer des de CLI per mitjà de l'ordre `VBoxManage` (fins i tot algunes opcions o configuracions només es poden aplicar des d'aquest).

Com a formats de disc, utilitza VDI (VBox Disk Image), VMDK (Virtual Machine Disk), VHD (Virtual HD), HDD (Parallels HD), QDE (QEMU Enhanced Disk), QCOW (QEMU Copy-on-write) i OVF (per exportar i importar *appliances*), i disposa d'ordres per canviar entre formats; permet muntar ISO (tant d'unitats virtuals òptiques com físiques de CD/DVD); és compatible amb l'acceleració en 3D, 32 vCPU (CPU virtuals), dispositius IDE, SATA, SCSI, connexió a iSCSI, ACPI, pantalla completa, quatre targetes d'Ethernet (o 36 si s'utilitza CLI), USB i la integració amb teclat i ratolí; admet que cada màquina virtual es pugui configurar mitjançant *software-based virtualization* (en aquest mode és compatible amb *guests* de 32 bits executant-se en *rings* 0 i 3 de l'arquitectura *ring* d'Intel) o *hardware assisted virtualization* (si es disposa de les extensions de maquinari), i des de la versió 6 admet la virtualització imbricada (és a dir, les MV poden veure les extensions de maquinari del processador).

Respecte a la gestió de màquines, permet fer *snapshots* (congela l'estat de l'MV i és possible tornar a la configuració anterior); és compatible amb Remote Desktop Extension o VRDE (per a connexió remota per RDP amb mapatge de l'USB local *USB over RDP*), configuració de les seqüències de tecles sobre el *guest* (per exemple, Ctrl + Alt + Del), exportació i importació d'MV en format OVF1.0/2.0, agrupacions per aplicar ordres a totes les MV del grup (iniciar, posar en pausa, reiniciar, salvar estat, enviar senyal d'apagat, apagar, descartar estat salvat, mostrar en sistema d'arxiu, ordenar) i xifratge mitjançant AES; disposa d'un administrador de mitjans virtuals, i amb el *host* pot compartir carpetes, USB i porta-retalls, i arrossegar i deixar anar.

Pel que fa als aspectes de xarxa, admet NAT o *Network Address Translation* (permet utilitzar el NIC del *host* creant un encaminador, per defecte en 10.0.2.0/24, però es pot canviar amb CLI, i reenvia els paquets pel *host*, per la qual cosa si el *host* té connexió, l'MV també ho tindrà), NetWork NAT (funciona com un encaminador domèstic, en què les màquines que estan en aquesta xarxa

es poden connectar entre si), *bridged* (IP pròpia a la xarxa del *host*, la màquina serà visible igual que el *host*), xarxa interna (xarxa aïllada a través d'un *switch* virtual) i *host-only* (xarxa interna però que compartirà amb el *host*).

També disposa d'un mode Generic Driver, que inclou altres *drivers* que vinguin amb VirtualBox o en un paquet d'extensió, i amb dos modes disponibles: UDP Tunnel (permet interconnectar les màquines virtuals que s'executen en diferents *hosts* de manera directa, fàcil i transparent, sobre la infraestructura de xarxa existent) i VDE o *Virtual Distributed Ethernet* (per connectar-se a un *switch* Virtual Distributed Ethernet en un *host* Linux o FreeBSD) [Vir].

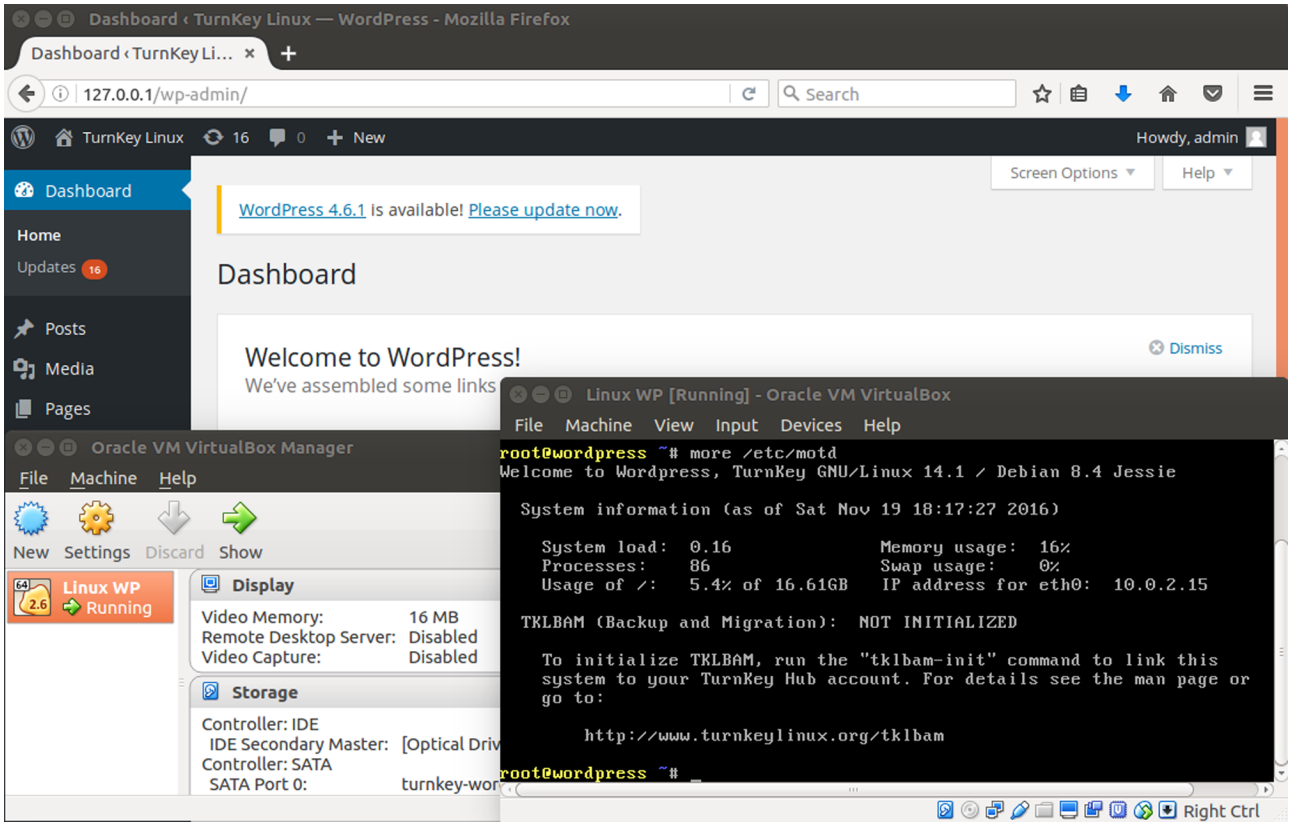
Tingueu en compte que, si es vol instal·lar VirtualBox mantenint KVM, es recomana treure (temporalment) els mòduls kvm i kvm_intel, ja que podrien interactuar entre ells i generar algun tipus d'error (tot i que en les distribucions actuals, com per exemple Ubuntu20.04, és possible executar VirtualBox tenint els mòduls KVM instal·lats sense problemes).

2.2.1. Instal·lació i configuració

La instal·lació de VirtualBox es durà a terme sobre una màquina amb processador Intel i7 620M (64 bits, 2 *cores* + Hyper-Threading = 4 processos) i sobre el SO Ubuntu. El repositori d'Ubuntu disposa d'una versió (6.1.26 per a Ubuntu 21.10) que es pot veure en qualsevol distribució mitjançant la següent ordre `apt-get search virtualbox | grep ^virtualbox`, i es podria instal·lar directament amb `apt-get install virtualbox`. Però si es vol l'última versió (6.1.28 en el moment d'elaborar aquest material), es pot baixar del repositori seleccionant la distribució, versió i arquitectura que es desitgi. Per instal·lar-ho, s'han de seguir les indicacions dels desenvolupadors, però bàsicament és:

```
dpkg -i virtualbox-xxx-yyy_amd64.deb
```

En cas que hi hagi problemes de dependències, es poden resoldre fàcilment amb `apt install -f` o, si no és possible, instal·lant els paquets indicats. La figura següent mostra una imatge de la interfície del VBoxManager (a baix a l'esquerra), l'MV (a sota a la dreta) executant una *appliance* de WordPress (disc en format VMDK) i un navegador sobre el *host* visualitzant una sessió de WordPress (per a això, l'MV té NAT a l'adaptador de xarxa i s'ha fet un *port forwarding* del port 80 del *host* al port 80 del *guest*).



2.2.2. Administració i execució remotes

Una característica de VirtualBox és que inclou una API que permet gestionar-lo i administrar-lo remotament. Hi ha diversos paquets i es comentaran a continuació algunes d'aquestes opcions.

RemoteBox és un client de VirtualBox que pot administrar de manera remota (és a dir, a través de la xarxa) una instal·lació de VirtualBox en un servidor, incloent-hi les màquines virtuals (*guests*), i interactuar-hi com si s'estiguessin executant localment. Funciona en mode client-servidor, en què VirtualBox està instal·lat a la màquina «servidor» i RemoteBox s'executa a la màquina «client». Proporciona una interfície gràfica completa (basada en GTK2) amb una aparença molt similar a la de la GUI nativa de VirtualBox i que permet controlar tot l'entorn de manera remota.

El funcionament és segur, perquè s'accedeix a través de l'API de VirtualBox (mitjançant un protocol SOAP), i això proporciona un accés estàndard i segur independentment de la versió de VirtualBox. Per accedir a les MV, es pot utilitzar el protocol d'escriptori remot (RDP) i RemoteBox usa un client RDP per mostrar la pantalla d'un convidat, localment a la màquina client i de manera completament interactiva. RemoteBox s'executa a Linux, Solaris, Mac OS X, Windows i versions actuals de BSD, i es pot connectar amb VirtualBox en qualsevol dels SO en què s'executi (Linux, Mac OS X, Windows...), per la qual cosa es recomana com a eina de gestió remota de VirtualBox.

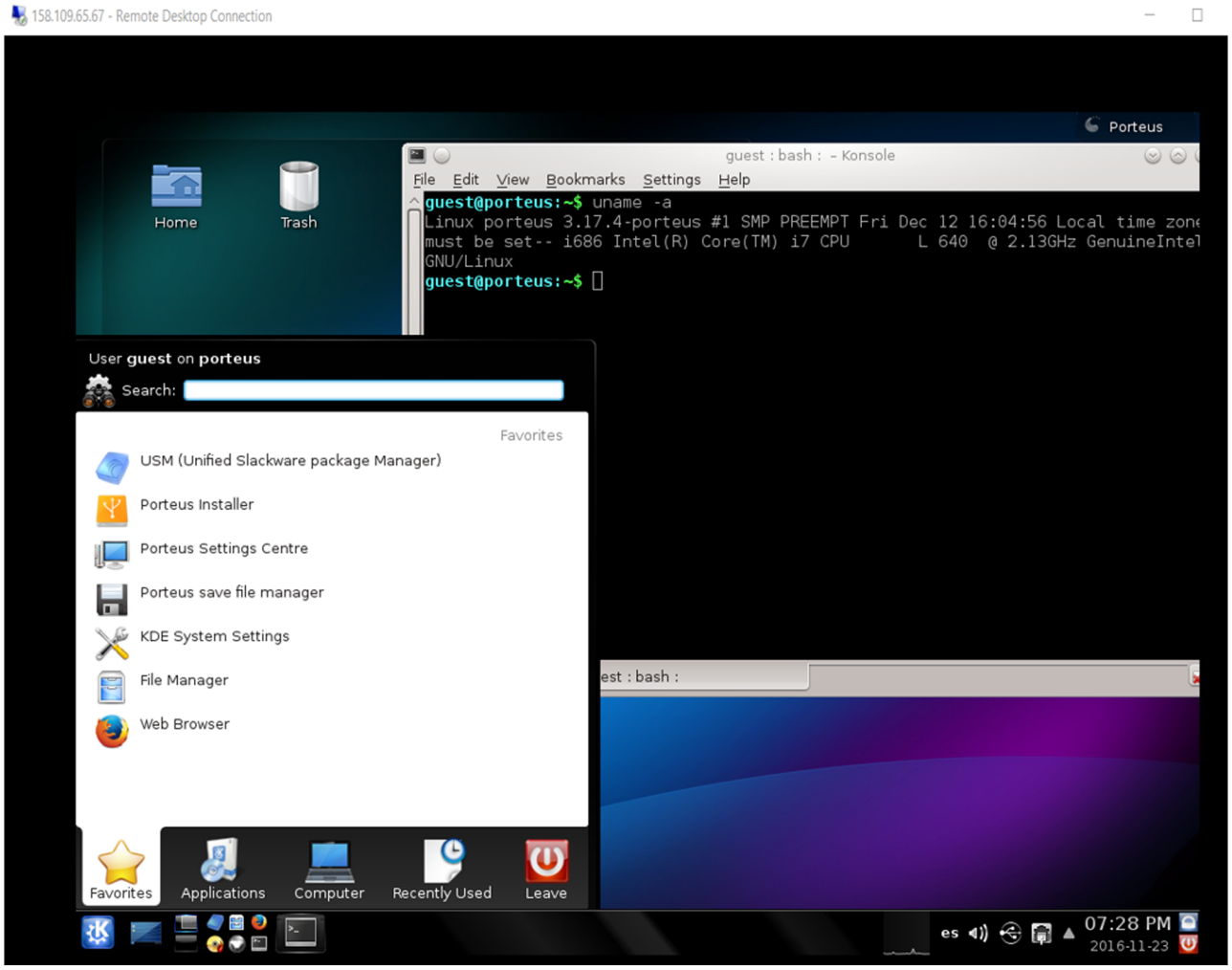
Un altre paquet interessant com a gestor d'infraestructura és Hyperbox, que és una alternativa *open-source* a productes comercials com ara *VMware vCenter/ESXi* i *Citrix XenCenter/XenServer,* basat en una arquitectura de client-servidor i que s'integra molt bé amb VirtualBox (i amb altres hipervisors segons l'autor).

Dins d'aquest apartat no es pot deixar d'esmentar, si bé avui ja és obsolet, phpVirtualbox, que va ser una implementació molt popular d'*open-source* AJAX de la interfície d'usuari de VirtualBox en PHP, per accedir i controlar remotament les instàncies de VirtualBox. Si bé ha estat un programari molt popular, en aquest moment està molt desactualitzat i només s'hauria d'utilitzar en casos molt específics, ja que depèn molt de la distribució de VirtualBox.

Si només es vol accedir a les MV remotament, la manera més simple és a través d'una interfície de *VirtualBox Remote Desktop Extension* (VRDE). Inclosa a les extensions, Oracle proporciona una implementació de *VirtualBox Remote Display Protocol* (VRDP) en què, atès que no és *open source* (tot i que és de franc), s'han d'acceptar les llicències quan s'instal·la. VRDP és una extensió compatible amb Microsoft *Remote Desktop Protocol* (RDP), i amb això es pot utilitzar qualsevol client RDP estàndard per controlar l'MV remota. Per instal·lar les extensions, simplement cal accedir a la pàgina web de Virtualbox i baixar l'arxiu d'extensions per a la versió instal·lada (VirtualBox x.y.zz Oracle VM VirtualBox *Extension Pack*), i VirtualBox ja ho obrirà i ho instal·larà (si no, s'ha de baixar l'arxiu i afegir-lo: *File → Preferences → Extensions*). Quan es crea a l'MV, el servidor VRDP està desactivat i es pot activar al menú de Display de l'MV o amb:

```
VBoxManage modifyvm "nom de la màquina virtual" --vrde on
```

De manera predeterminada, el servidor VRDP utilitza el port TCP 3389, i si es vol usar més d'un servidor, s'hauran de canviar els ports, ja que el port només es pot utilitzar per part d'un servidor alhora (a la CLI amb *--vrdeport* o al menú de Display), o també es poden utilitzar rangs de ports (consulteu el manual). La figura següent mostra l'execució d'una MV amb Linux Proteus per VRDP utilitzant el client *Remote Desktop Display* sobre W10.



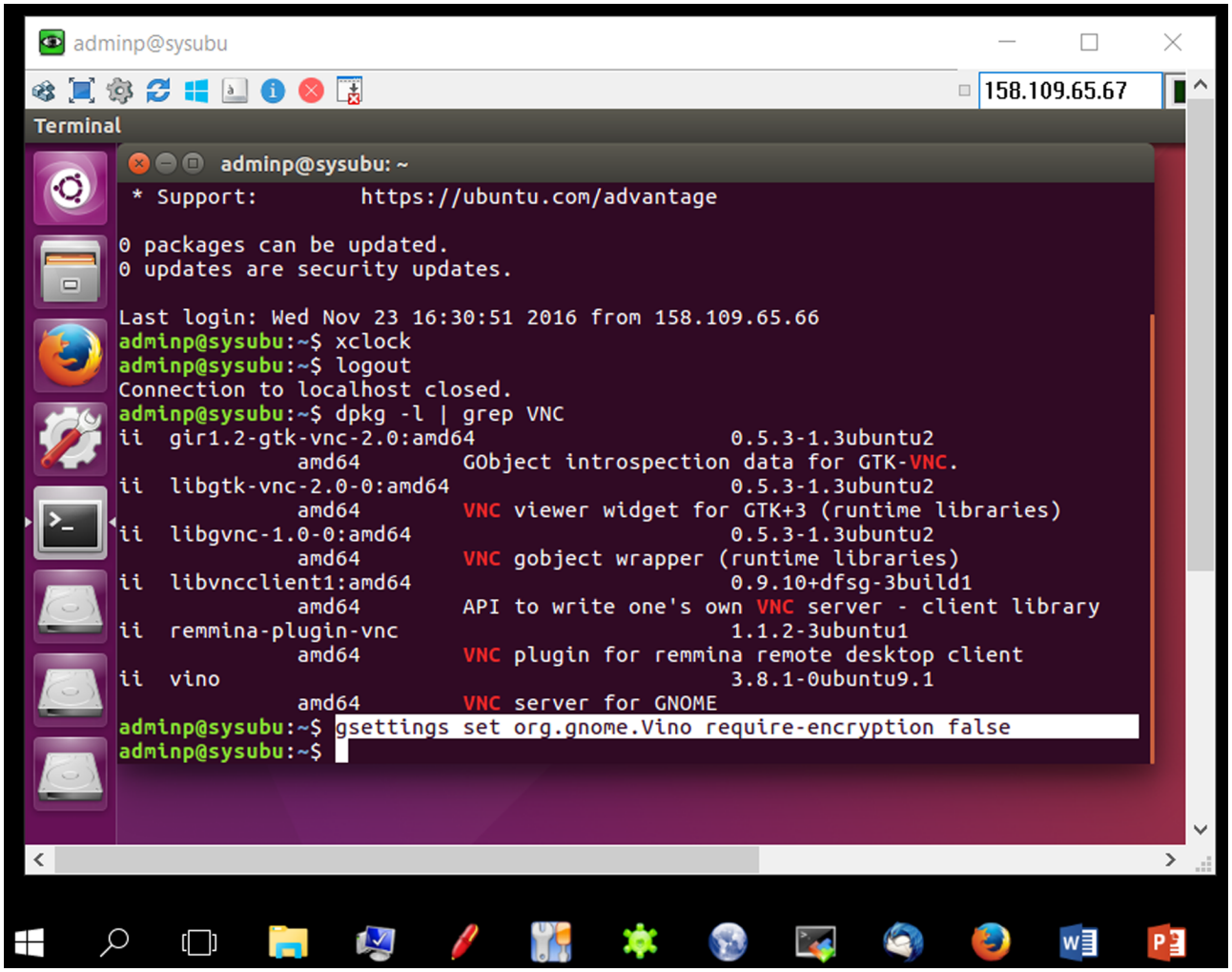
No obstant això, i és vàlid en qualsevol hipervisor, si la màquina té IP a la xarxa (o bé hi ha una regla de *forward* a la màquina que fa de passarel·la), la manera més simple de connectar-se a una MV és a través de ssh `-X ip_MV`, amb la qual cosa s'habilita amb el paràmetre -X un X11 *forwarding* mitjançant el qual es pot accedir al *display* remot (consulteu la documentació del `ssh` sobre qüestions de seguretat i la diferència amb el paràmetre -Y), i les aplicacions que s'executin remotament es visualitzaran al *display* local (haurà de ser un *display* compatible amb X11). Per a Windows es pot utilitzar l'aplicació (gratuïta) MobaTerm, que ja inclou un servidor d'X11.

Una altra opció, que ja s'ha esmentat al subapartat de KVM, per connectar-se i accedir a un escriptori remot és per mitjà de *Virtual Network Computing* (VNC). Igual que amb KVM, s'ha d'anar amb compte amb el servidor-client que s'utilitzi i les opcions, perquè en alguns casos la comunicació anirà sense encriptar o tots dos (servidor i client) han d'admetre els algorismes d'encriptació per poder connectar-se. Com ja s'ha indicat, a Ubuntu el servidor per defecte és `vino` (i corre en el port 5900 per defecte), el qual utilitza un algorisme

d'encriptació per defecte (TLS, *type 18*) que no és compatible amb els clients habituals de Windows, però es pot deshabilitar l'encriptació mitjançant (des de l'usuari que té oberta la pantalla)

```
gsettings set org.gnome.Vino require-encryption false
```

S'ha de tenir en compte que, amb aquesta configuració, tota la comunicació va en clar per la xarxa, per la qual cosa es recomana utilitzar un túnel `ssh` o canviar a un altre servidor. Hi ha una gran quantitat de clients per a Linux (per exemple, Ubuntu), i un que s'utilitza habitualment és Remmina, que està per a la majoria de distribucions. La figura següent mostra UltraVNC sobre W10 i servidor vino-Ubuntu1 6.04 amb l'encriptació desactivada (és una opció només per a xarxes segures).



Algunes vegades es vol tenir a la mateixa màquina VirtualBox i KVM, però, segons la versió del *kernel* (si és +5.8 no passa), hi pot haver un error (similar a *can't operate in VXM root mode*) si es vol executar Virtualbox amb KVM

instal·lat. Per aquest motiu, s'han de desinstal·lar (temporalment) els mòduls kvm i kvm_intel perquè es puguin arrencar les MV en Virtualbox. Per a això es comproven primer els mòduls:

```
lsmod | grep kvm
kvm_intel       172032 0
kvm          540672 1 kvm_intel
...
```

Després es desinstal·len amb:

```
/sbin/rmmod kvm_intel
/sbin/rmmod kvm
```

I ja es podran executar les màquines virtuals. Quan es vulgui tornar a iniciar les màquines en KVM, s'han de carregar els mòduls novament amb:

```
cd /lib/modules/'uname -r'/kernel/arch/x86/kvm/
insmod kvm.ko; insmod kvm-intel.ko
```

2.3. VMware Workstation Player

És un dels paquets de virtualització produïts per VMware (empresa que té com a accionista majoritària EMC, amb un 83 % de les accions, que al seu torn va ser adquirida per Dell el 2016) per a equips x64, que utilitza com a *host* Windows o Linux; és gratuït per a ús personal, domèstic i no comercial. VMWPlayer pot executar *appliances* existents i crear i gestionar les seves pròpies màquines virtuals utilitzant el mateix nucli de virtualització que VMware Workstation Pro (no gratuït), però amb algunes limitacions (probablement la més important és que en la versió Player només deixa executar una màquina virtual alhora, restricció que no existeix en la Pro) [Vwp].

Entre altres característiques, és compatible amb *multiple-monitor display*, USB 3.0, fins a 16 vCPU× 64 GB RAM, discos de fins a 8 TB, gràfics amb acceleració i execució de MV xifrades.

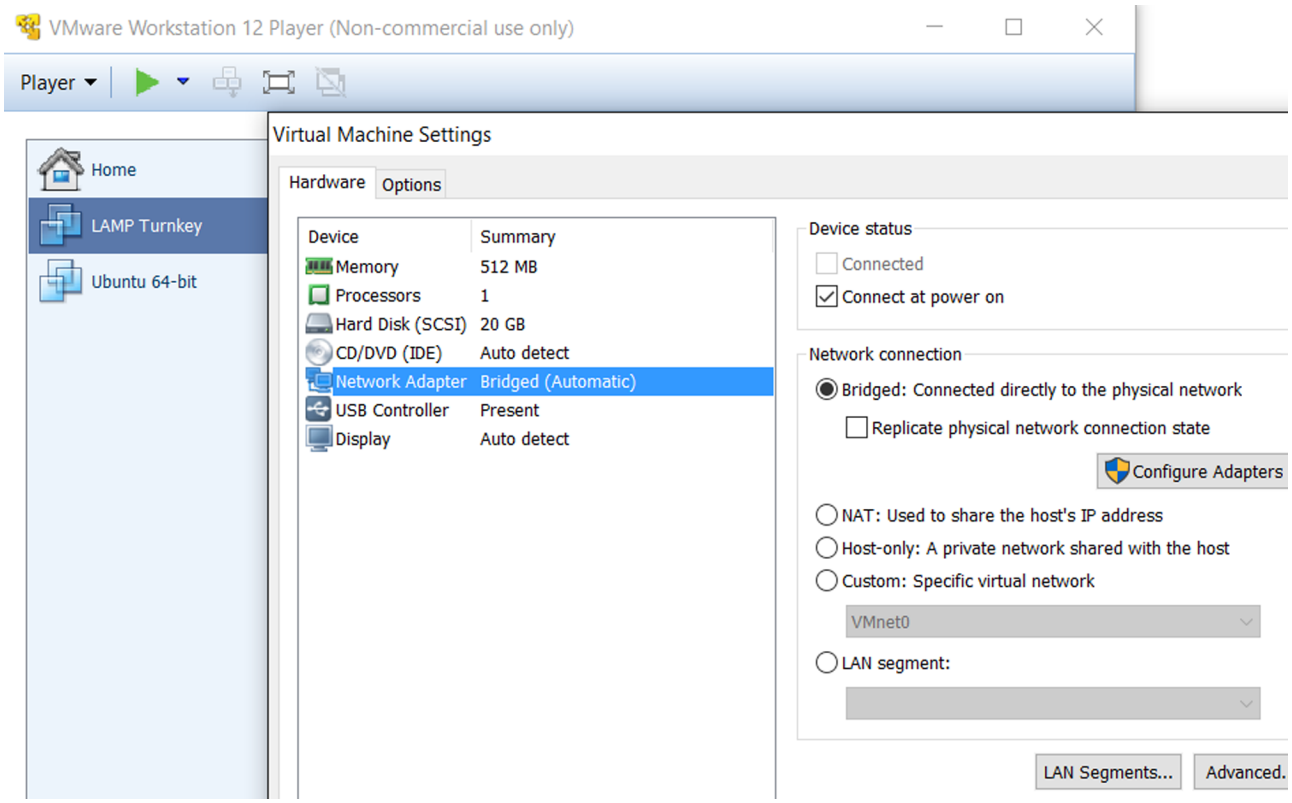
S'instal·la molt fàcilment tant per a Windows com per a Linux (instal·lació), i els passos per instal·lar una MV són els mateixos que en qualsevol altra aplicació, ja que es pot fer des del CD/DVD o des de l'ISO (també utilitza Ctrl + Alt per passar del *guest* al *host*). Quan detecta que el sistema que s'instal·larà és Linux, baixa un conjunt d'eines (VMware Tools) per millorar el rendiment, el moviment del ratolí i els gràfics. Per a Linux (el paquet té extensió *.bundle*) es pot instal·lar sobre una consola o amb un terminal gràfic (en aquest cas, amb un doble clic sobre l'arxiu, és el gestor d'arxius). El programa d'instal·lació a Linux es pot posar en marxa amb:

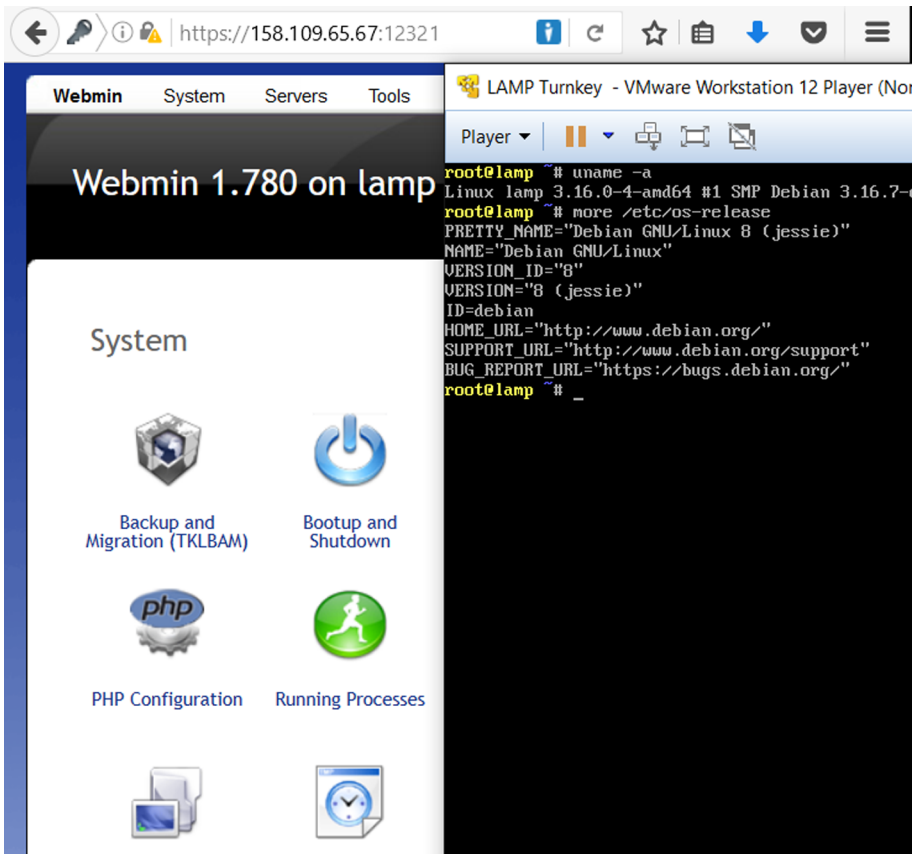
```
sh VMware-Player-xxx-yyy.x86_64.bundle --opción
```

On s'ha de reemplaçar *xxx-yyy* per la versió adequada (l'arquitectura només pot ser de 64 bits) i les possibilitats són *gtk* (interfície gràfica), *console* (text de terminal) i *custom* (permet triar diferents opcions de directori o límits).

Després d'instal·lar l'MV (Linux en aquest cas) i VMware Tools, indicarà que es faci clic en un botó d'Install tools, la qual cosa muntarà el CD/DVD (virtualitzat) amb el programari. Després de muntar el dispositiu (amb la següent ordre `sudo mount /dev/cdrom /media`) trobarem el paquet VMware-Tools-xyz.tar.gz, que es podrà copiar al /tmp per exemple, extractar (`tar xzvf file.tgz`) i, dins del directori, executar `vmware-install.pl`. S'ha de tenir en compte que algunes distribucions (Ubuntu, per exemple) ja disposen d'aquestes eines com `open-vm-tools` i que només s'han d'instal·lar.

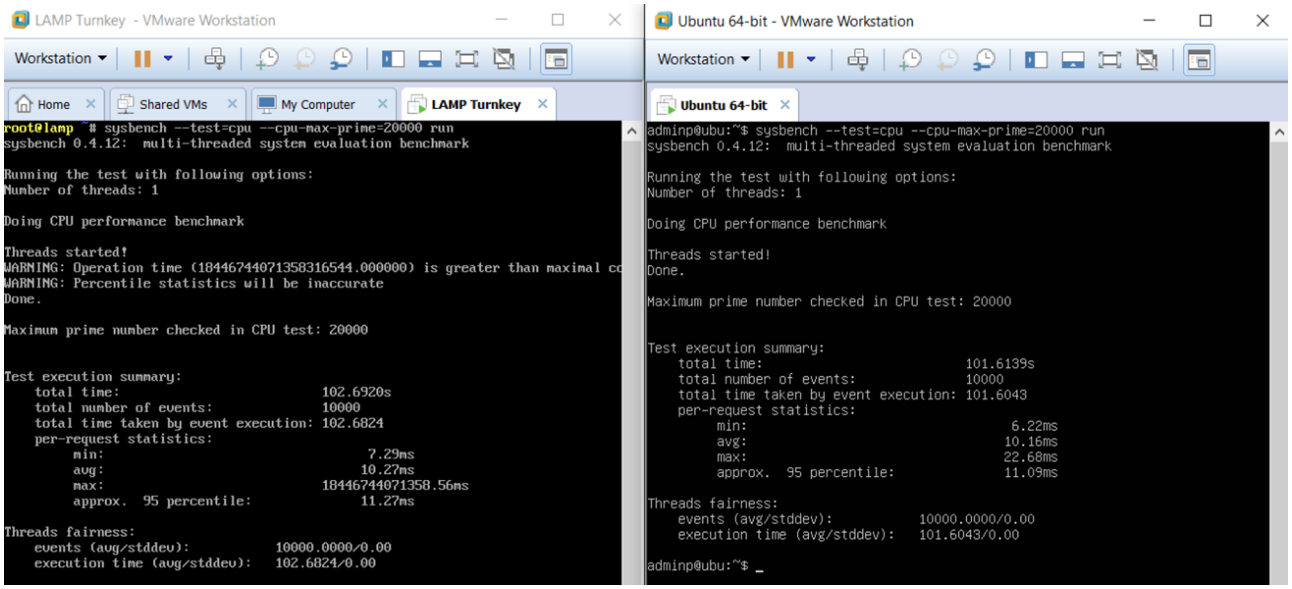
Les figures següents mostren la interfície de VMware WP amb dues MV instal·lades (i amb les opcions habituals per configurar els recursos) i l'execució d'un *appliance* (VMDK) LAM (Turnkey) amb la interfície de Webmin.





A la part de la dreta es pot veure l'MV sobre la consola del VMware WP, i a l'esquerra, l'accés a l'MV des d'una màquina remota i a l'aplicació Webmin per administrar l'*appliance*.

La instal·lació de VMware Workstation Pro (versió de pagament, però que permet una versió de prova de 30 dies) inclou una sèrie d'eines addicionals, com ara editor de xarxes; permet executar totes les MV que es vulgui, i es connecta amb altres productes VMware, entre altres opcions. La figura següent mostra dues MV executant el *benchmark sysbench*.



Més enllà del fet que VirtualBox és *open source*, com s'ha pogut observar, les opcions i possibilitats de VirtualBox i VMware Workstation Player en les seves últimes versions són semblants, excepte que amb el segon només es pot obrir una MV (limitació que no tenen els altres productes de VMware, però són de pagament). Pel que fa a les prestacions, són similars, i es recomana, sobre el maquinari disponible, executar diversos *benchmarks* (`sysbench` o també Phoronix, per exemple) per veure quin aconsegueix millors prestacions (en molts aspectes dependrà dels *drivers* usats i de si s'utilitzen *drivers* paravirtualitzats *virtio*); en aspectes de facilitat i flexibilitat, segons molts experts, VirtualBox proveeix prestacions i opcions semblants als productes de VMware, amb l'avantatge de ser *open-source* (malgrat que no disposa de suport empresarial).

2.4. Proxmox

Proxmox *Virtual Environment* és una solució (basada en Debian) *open source* (GPL) per gestionar servidors virtualitzats amb QEMU/KVM i LXC que permet gestionar màquines virtuals, contenidors, clústers d'alta disponibilitat, emmagatzematge i xarxes amb una interfície web molt simple o a través de la CLI. Un aspecte important del seu disseny (*multi-master*) és que no és necessari un servidor d'administració addicional, estalviant recursos i permetent l'alta disponibilitat (HA), la qual cosa possibilita desplegar entorns de virtualització de classe empresarial en un centre de dades. Permet, a més, múltiples fonts d'autenticació combinades amb la gestió de rols i permisos d'usuari, donant un control total a l'administrador del clúster virtualitzat d'HA, i disposa, a més, d'una API web RESTful per integrar eines de gestió de tercers [Pve].

Les característiques principals de Proxmox són les següents:

- **Tecnologia**: és compatible amb els *guests* Linux i Windows (32/64 bits) i incorpora les últimes especificacions d'Intel/AMD per millorar les prestacions de les MV, a fi de suportar càrregues de treball dins d'una empresa.

- **Administració**: conté totes les eines necessàries per gestionar una infraestructura virtualitzada en un únic entorn amb API RESTful i amb verificació automàtica de paràmetres, per reduir errors en la definició i la gestió.

- **Arquitectura**: basada en ROA (*resource oriented architecture*), que permet *high availability cluster with no SPOF* (*no single point of failure*), i amb *multi-master cluster* (per garantir la disponibilitat), i tot gestionat des de la interfície GUI (basada en AJAX).

- **Sistema d'arxius**: basat en pmxcfs (*Proxmox VE Cluster File System*) orientat a una base de dades per a l'emmagatzematge dels arxius de configuració, que són replicats sobre tots els nodes utilitzant Corosync.

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

- **Agents**: suport per a KVM i Linux Contenidors (LXC).

- **Seguretat**: MV i contenidors aïllats amb suport segur (SSL) per a consola VNC en HTML5, i amb gestió basada en rols per al tractament de permisos de tots els objectes (MV, contenidors, emmagatzematge...) i autenticació multimode (per exemple, local, MS ADS, LDAP...). A més, incorpora un tallafoc integrat que permet filtrar paquets sobre qualsevol interfície, tant d'una MV com d'un contenidor, i aplicar les regles per grups anomenats *security groups*.

- **Migració**: «en viu», que permet moure MV des d'una màquina física a una altra sense que calgui un temps d'apagament i recuperació.

- **Còpia de seguretat i recuperació**: incorpora una eina (`vzdump`) per crear *snapshots* dels contenidors o MV, que permet salvar-los (i posteriorment recuperar-los) en diferents tipus d'emmagatzematge (com ara NFS, iSCSI LUN, Ceph RBD o Sheepdog) i en un format optimitzat i efectiu.

- ***Bridged networking***: totes les MV comparteixen un *bridge*; la connectivitat entre les MV i l'exterior es dona a través d'una interfície de xarxa, i es poden generar VLAN (IEEE 802.1q) i el *network bonding/aggregation*, la qual cosa permet construir xarxes complexes adequades a les necessitats de connexió dels *guests*.

- **Emmagatzematge**: és flexible. Les MV es poden emmagatzemar en local o compartir-se per NFS o SAN sense grans restriccions i permetent la migració «en viu» si estan en sistemes compartits. Els sistemes d'arxius en xarxa

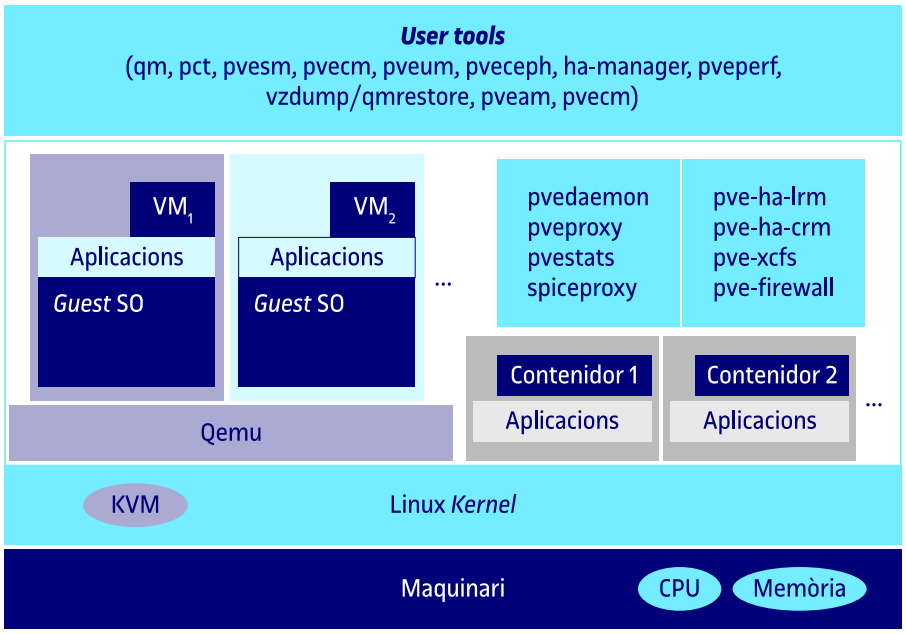
compatibles són els habituals en Linux LVM sobre iSCSI targets, iSCSI target, NFS, Ceph RBD, Direct to iSCSI LUN o GlusterFS, i com a locals LVM Group (sobre dispositius de blocs, DRBD...), Local File System o ZFS.

Proxmox VE té com a característiques similars a VMware Sphere, Hyper-V i Citrix-XenServer que és un hipervisor *bare-metal*, i que és compatible amb la migració «en viu», HA, *snapshots* i còpies de seguretat d'MV i contenidors, però, a més de ser *open source*, és l'únic que disposa de suport per a contenidors i d'una gestió centralitzada integrada. Pel que fa a les seves limitacions, són les menys restrictives i iguals que les de VMWare, que estan en 160 CPU / 2 TB RAM per *host*.

Com es pot observar a la documentació [Pve], aquest projecte és recent (comença el 2007 i la seva primera versió estable és del 2008). A la primera versió es va utilitzar OpenVZ per a contenidors i KVM per a màquines virtuals, i a les versions successives, s'hi afegeixen Corosync i *pmxcfs* (nou sistema d'arxius de clúster), amb la qual cosa s'aconsegueix un gran avenç en la gestió de clústers, perquè administrar múltiples nodes requeria la mateixa complexitat que ferho amb un. També s'hi afegeix una API REST (escrita en JSON-Schema) per integrar Proxmox VE amb altres eines, i que després va servir per reemplaçar la interfície d'usuari per una moderna aplicació HTML5 + JavaScript, i la consola original VNC (basada en Java) per noVNC a fi d'administrar les màquines virtuals.

No obstant això, els canvis més importants per als desenvolupadors van ser la compatibilitat amb ZFS (primera distribució a oferir-lo en Linux el 2014), la gestió d'emmagatzematge Ceph als nodes de l'hipervisor, les còpies de seguretat «en viu» de KVM i el canvi d'OpenVZ a LXC, perquè els contenidors estiguin plenament integrats i puguin utilitzar les mateixes funcions d'emmagatzematge i xarxa que les màquines virtuals.

La figura següent mostra l'arquitectura de Proxmox VE.



A la capa d'usuari s'hi poden apreciar les eines següents:

- **qm** (QEMU/KVM *Virtual Machine Manager*): executa ordres en QEMU (*guest agent*).
- **pct** (*Container Toolkit*): crea o clona un contenidor.
- **pvesm** (*Storage Manager*): crea un nou espai d'emmagatzematge.
- **pveum** (*User Manager*): gestiona els usuaris.
- **pveceph** (*Manage CEPH Services on Proxmox VE Nodes*): crea un monitor de Ceph.
- **ha-manager** (HA): gestiona la infraestructura d'alta disponibilitat (HA).
- **pveperf** (*Benchmark Script*): analitza les prestacions.
- **vzdump/qmrestore** (*Backup Utility for VMs and Containers / Restore Backups*): crea i recupera còpies de resguard.
- **pveam** (*Appliance Manager*): gestiona les *appliances*.
- **pvecm** (*Cluster Manager*): gestiona el clúster.

Entre els principals *daemons* o serveis, es poden enumerar els següents:

- **pve-firewall** (*Firewall*)

- **pvedaemon** (API)
- **pveproxy** (*Proxy*)
- **pvestatd** (*Status*)
- **spiceproxy** (SPICE *Proxy Service*).
- **pmxcfs** (*Cluster File System*)
- **pve-ha-crm** (*Cluster Resource Manager*)
- **pve-ha-lrm** (*Local Resource Manager*)

2.4.1. Instal·lació i creació de màquines virtuals i contenidors

Per instal·lar aquest hipervisor, s'utilitzarà, com a prova de concepte només per veure'n la potencialitat (no per a una màquina de serveis), una màquina virtual de KVM. Per a això, caldrà activar la virtualització imbricada, com s'ha esmentat al subapartat de KVM (perquè si no les MV que posi en marxa només s'iniciaran en mode emulació, amb la conseqüent pèrdua de prestacions). Aquest hipervisor està basat en Debian i la imatge inclou tots els paquets necessaris: només s'ha de crear la màquina virtual en KVM, inserir aquesta imatge en la unitat de CD-ROM i iniciar-la, seleccionar *Install Proxmox VE* al menú d'arrencada i respondre a les preguntes de configuració que formularà:

- Discos per utilitzar: particions, que en aquest cas són simples, perquè és tot el disc virtual i els formats *ext3/4*, *xfs* i *ZFS*.
- Paquets: tots els paquets per gestionar l'entorn, incloent-hi les màquines virtuals KVM i els contenidors LXC, a més de l'entorn d'administració web.
- Configuració bàsica de la xarxa: en aquest cas, és necessari donar-li un IP del nostre segment de xarxa KVM (no obstant això, després es podrà reconfigurar).

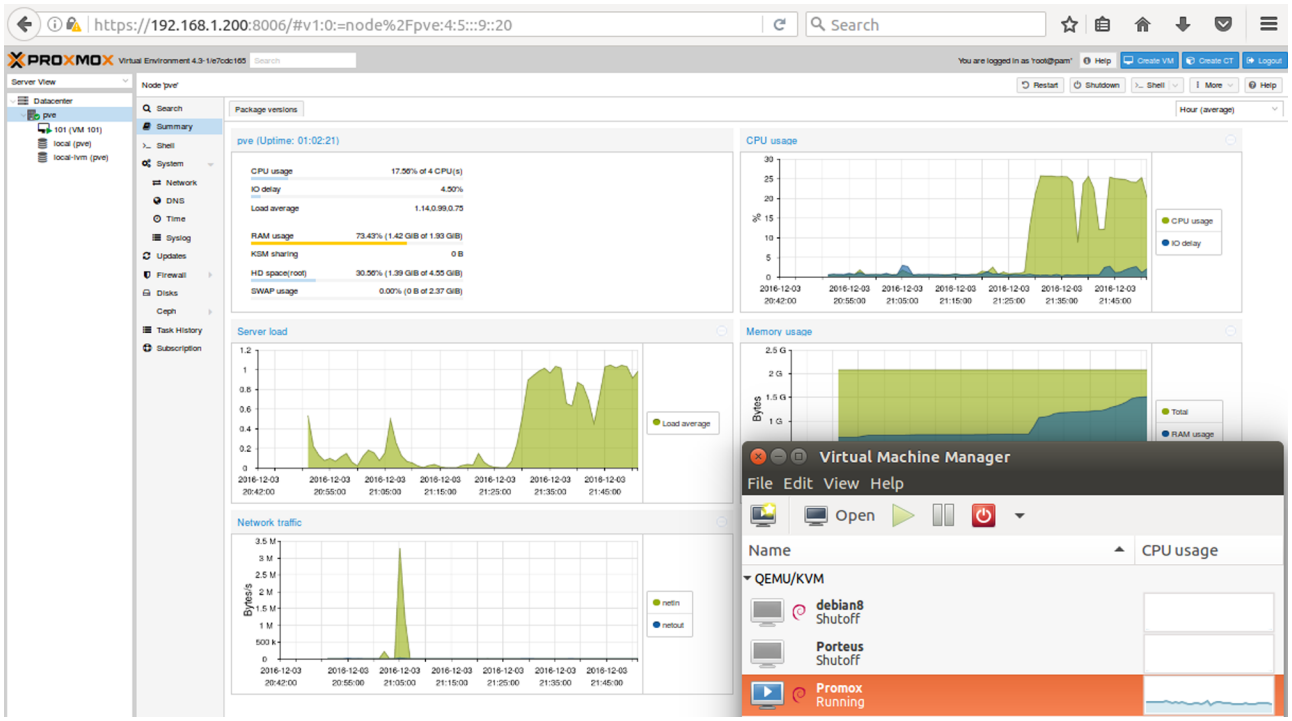
Al cap d'uns quants minuts tindrem el *login* de PVE i es podrà accedir a la consola. Una alternativa (consulteu la documentació) és instal·lar-lo sobre un Debian instal·lat prèviament (sigui virtual o sobre *bare-metal*), però només es recomana a usuaris avançats de Debian. Cal recordar que si s'està sobre un sistema físic, el disc seleccionat per a la instal·lació serà formatat i, per tant, perdrà tot el contingut.

Hem de fer una observació important. Quan es tria el format dels discos, és, per defecte, *ext4*, però es poden seleccionar els altres disponibles, i s'ofereixen opcions addicionals per configurar l'LVM. L'instal·lador crearà un *volume group*

(VG), anomenat **pve**, i *logical volumes* (LV), anomenats ***root***, ***data*** i ***swap***, en què el volum es podrà controlar amb els paràmetres *hdsize* (defineix la grandària total del disc que es farà servir, la qual cosa permet reservar espai per a accions futures, com un PV/VG addicional, i es podrà afegir a l'LVM posteriorment), *swapsize* (per defecte, la mateixa grandària de la RAM i hdsize/8 com a màxim), *maxroot* (grandària màxima per al volum de *root*), *maxvz* (grandària del volum de dades) i *minfree* (defineix la quantitat mínima d'espai lliure en el volum *pve*, que serà de 16 GB si la partició és superior a 128 GB, o *hdsize*/8 en un altre cas).

Si es disposa de més d'un disc, es pot utilitzar ZFS (*Zettabyte File System*) com a sistema d'arxiu, que destaca perquè és compatible amb arxius de gran volum, uneix dos conceptes (separats per altres sistemes d'arxius) com sistema de fitxers i administrador de volums *lògics* i té una nova estructura optimitzada sobre el disc que permet arxius lleugers i una gestió de l'espai simple. Per tenir una idea dels volums amb què treballa, ZFS permet 2 10^{14} entrades en un directori (2^{48}), 16 exbibytes (2^{64} bytes) com a volum màxim d'un únic arxiu i de qualsevol atribut i 256 zebibytes (2^{78} bytes) com a volum màxim d'un *zpool* (3 × 10^{23} petabytes), dels quals se n'admeten 2^{64}. Una comparació que utilitzen sovint els seus desenvolupadors (Sun Microsystems el 2004, que es va integrar a OpenSolaris el 2005, el qual va interrompre la seva activitat el 2010, tot i que hi ha diferents *forks* com Illumos que continuen amb els seus avenços) és que, si un usuari creés mil fitxers per segon, trigaria més de nou mil anys a arribar al límit imposat pel nombre de fitxers. A Proxmox VE, ZFS és compatible amb diversos nivells de RAID, la qual cosa permet adaptar-se si no es disposa d'una controladora RAID de maquinari, però cal tenir en compte que ZFS utilitza una quantitat molt gran de memòria; es necessita memòria addicional si se selecciona aquesta opció: per exemple, 4 GB més 1 GB RAM per cada terabyte d'espai de disc per a ZFS (*raw*), és a dir, si es disposa de dos discos d'1 TB a RAID 1 sobre ZFS, es recomanen 6 GB de RAM.

Tota la configuració de PVE la durà a terme la interfície web, a la qual s'accedirà per la IP del servidor (la indicada durant la instal·lació) posant en un navegador ***https://ip-servidor:8006*** (cal tenir en compte que s'haurà d'acceptar el certificat propi de PVE abans d'accedir, i entrarem amb l'usuari *root* i *passwd* indicats durant la instal·lació). La imatge següent ens mostra una pantalla de l'administració de PVE executant-se com a MV de KVM amb l'opció de virtualització imbricada.



Per instal·lar una MV o un contenidor, cal disposar dels ISO o dels arxius dels contenidors que es poden gestionar des de la mateixa interfície web. Per a això s'ha de seleccionar el *datastore* local i, a *Content*, accedir a *Templates* (per als contenidors); sortirà una llista de tots els disponibles tant del sistema com dels contenidors que proveeix Turnkey (els del sistema també estan a http://download.proxmox.com/appliances/system/), que es poden descarregar directament. Per als ISO es pot utilitzar l'opció *Upload* des del mateix menú i penjar un ISO baixat prèviament. Aquestes operacions també es poden fer des de la CLI i s'hauran de guardar al *datastore* (*/var/lib/vz/template/iso* per als ISO i */var/lib/vz/template/cache* per als contenidors). Per a un ISO de Debian, per exemple, es pot fer:

```
cd /var/lib/vz/template/iso
wget http://cdimage.debian.org/debian-cd/8.6.0/amd64/iso-cd/debian-8.6.0-amd64-netinst.iso
```

Els quals ja es veuran a la interfície web quan s'actualitzi. Per als contenidors, en format *xz*, una manera simple és a través de la CLI amb l'ordre *pveam* (*appliance manager*).

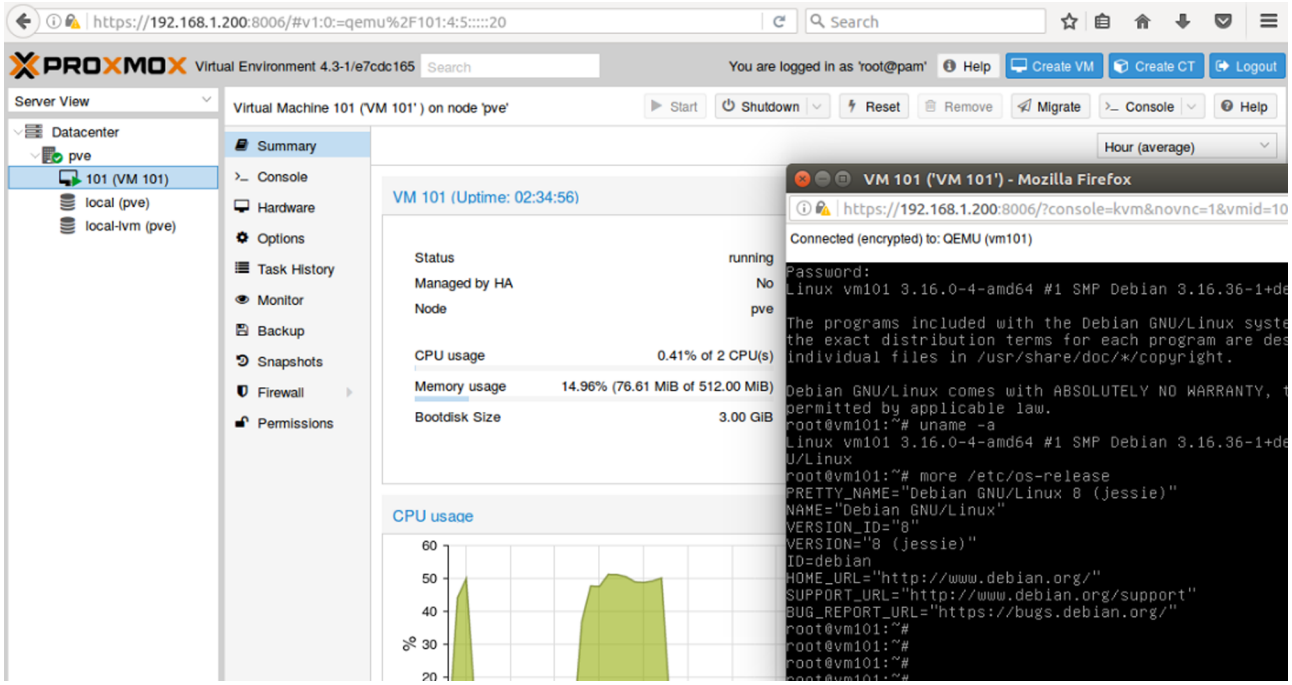
```
pvme update
pveam available
...
system     centos-x-default_xxxx_amd64.tar.xz
...
turnkeylinux  debian-x-turnkey-lamp_xx.yy_amd64.tar.gz
...
pveam download local debian-x-turnkey-lamp_xx.yy_amd64.tar.gz
```

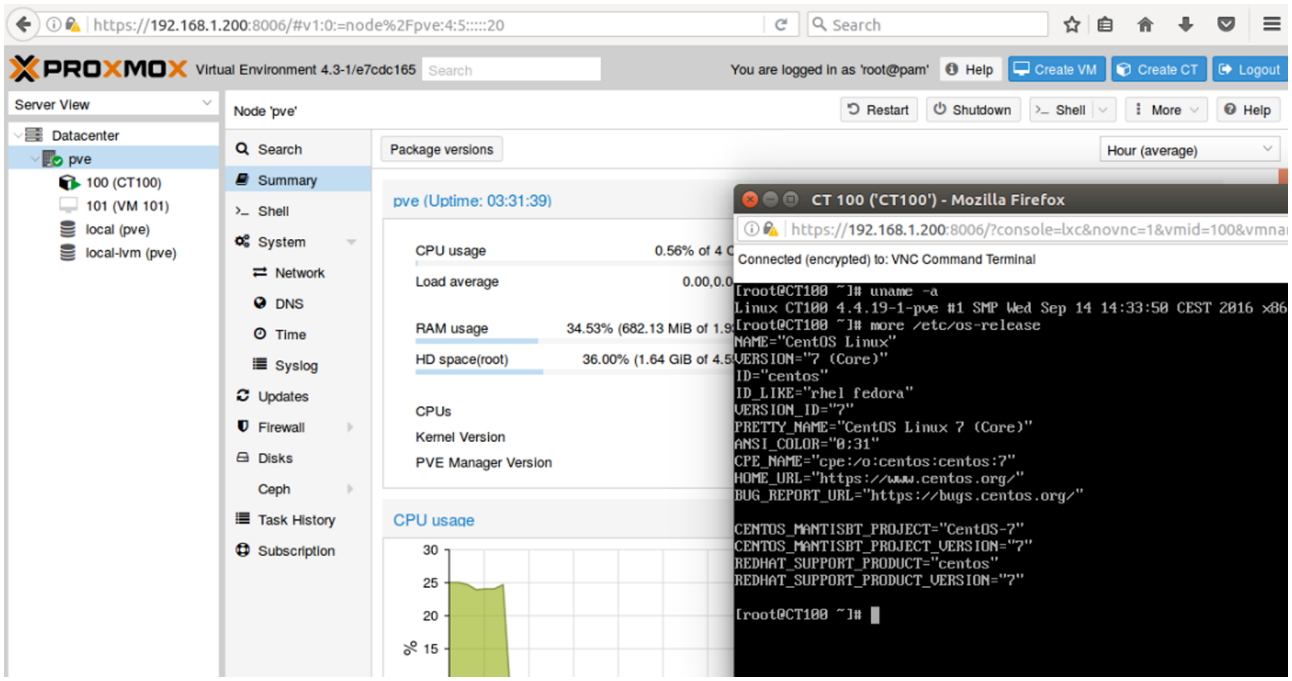
```
pveam download local centos-x-default_xxxx_amd64.tar.xz
```

Amb això, ja s'està en disposició de crear tant una màquina virtual com un contenidor, seleccionant les opcions indicades a la dreta de la interfície.

És important tenir en compte que en aquesta prova s'està utilitzant un servidor Proxmox virtualitzat sobre una màquina KVM, per la qual cosa les màquines virtuals dins d'aquest, si no es disposa de la virtualització imbricada, s'hauran d'emular a través de QEMU. Per això, no s'ha de seleccionar aquesta opció a l'hora de crear o s'ha de deshabilitar al menú *Options KVM software Virtualization* de la màquina virtual, perquè, si no, aquesta no arrencarà (*error: no accelerator found!*). Si es disposa de la virtualització imbricada (vegeu l'apartat KVM) a l'MV de Promox, les MV sobre aquest es podran executar amb l'opció KVM activada. A les màquines s'hi podrà accedir a través de la consola noVNC o a través de ssh.

Les figures següents mostren l'execució d'una MV Debian 8 instal·lada a partir de l'ISO i d'un contenidor CentOS7.





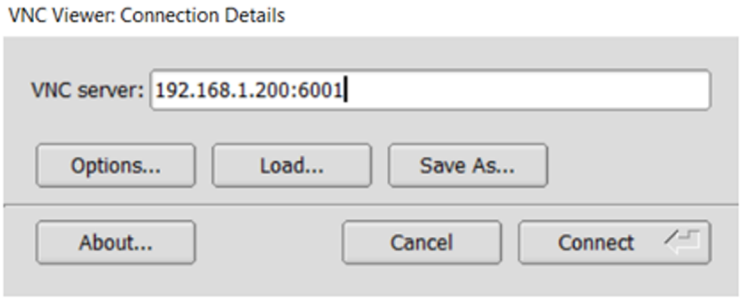
És important destacar la reducció significativa en el desplegament d'un contenidor i la reducció en l'ús de recursos (com es pot apreciar a la interfície gràfica de cadascun) en relació amb les màquines virtuals.

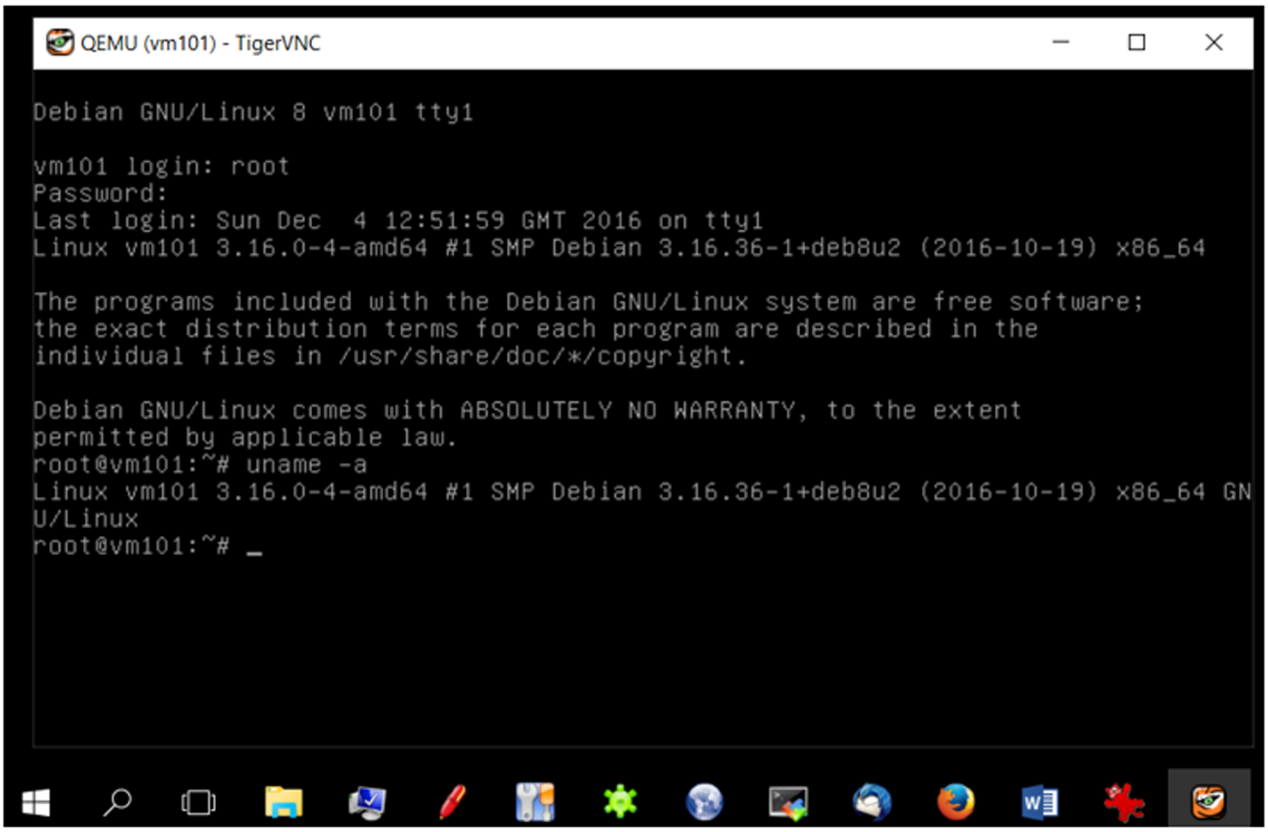
A [Pwi] s'hi poden trobar diversos manuals (*HowTo*) sobre com s'ajusten paràmetres determinats o s'instal·len i administren aspectes determinats de l'entorn.

Un aspecte interessant és poder-se connectar a les MV externament, i la manera més simple de fer-ho a través de VNC (o per SPICE). Per a això, simplement s'accedeix a l'opció de monitor a la màquina virtual i se li afegeix:

```
change vnc 0.0.0.0:101
```

On 101 és el nombre que s'ha d'afegir al port base de VNC (5900) per a la connexió a aquesta màquina, i en aquest cas queda 6001. Després, per exemple, amb TigerVNC (inclou el client i el servidor i és compatible amb diferents algorismes d'encriptació, però en aquest cas només s'ha descarregat el client de VNC), s'executa:





És important recordar que en aquest cas l'MV 101 només té NAT a la seva xarxa i que en aquesta prova el servidor Proxmox s'està executant com a MV de KVM, però amb *bridge* de la seva connexió (IP 192.168.1.200). A la documentació s'indica com s'introdueix *passwd* i com es fa aquesta configuració definitiva (perquè amb Monitor només és temporal).

També hi ha altres formes de connexió, a través d'un servei al mateix servidor Proxmox. Per a això, s'haurà d'activar el `vnc proxy` al servidor instal·lant `openbsd-inetd` (o també es pot fer amb `xinetd`) i habilitar-lo per a la màquina en qüestió. Sobre el servidor s'executarà:

```
apt-get update
apt-get install openbsd-inetd
```

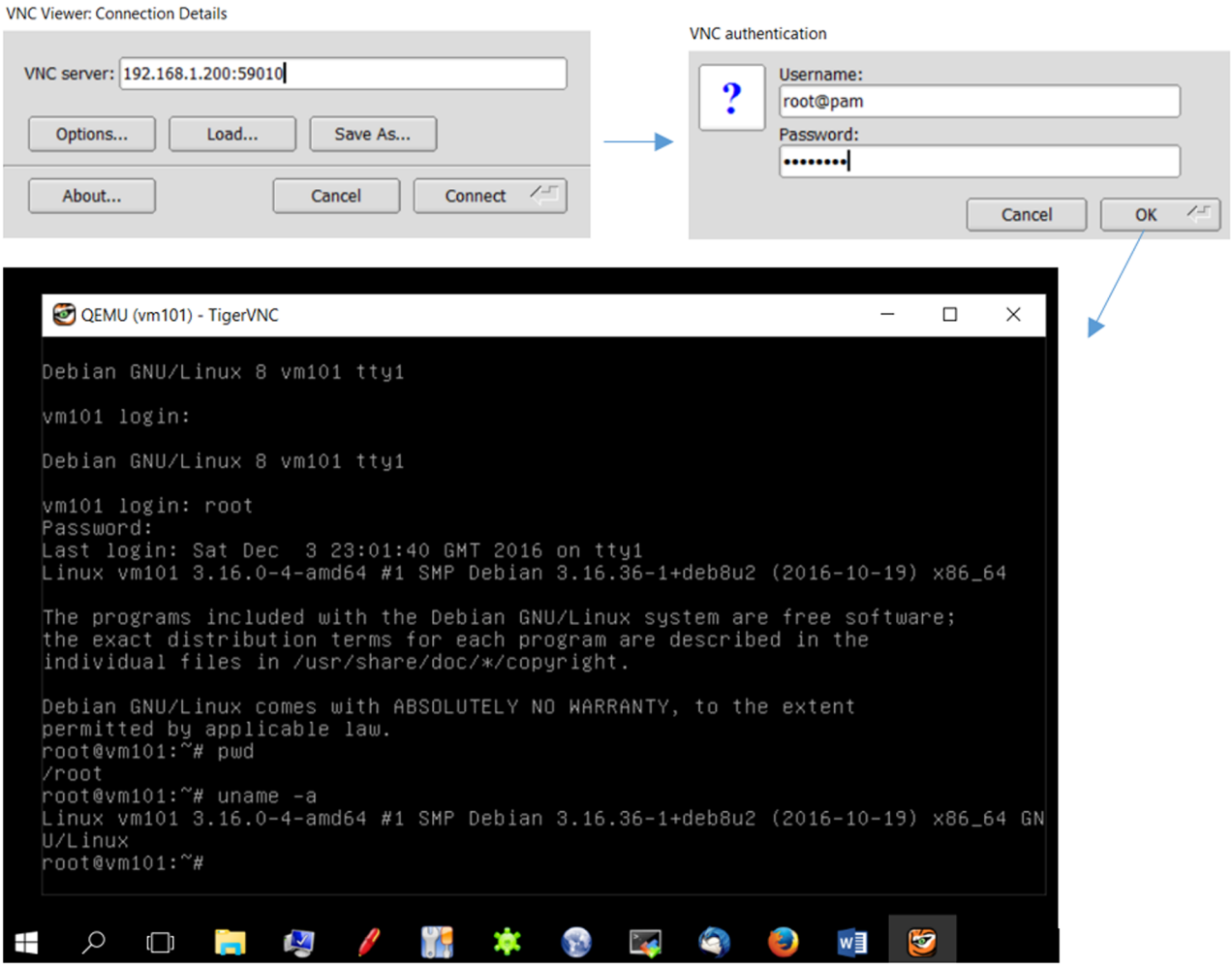
S'edita */etc/inetd.conf* i s'hi afegeix una línia amb:

```
59010 stream tcp nowait root /usr/sbin/qm qm vncproxy 101
```

On s'indica a *QEMU/KVM Virtual Machine Manager* (`qm`) que faci un *proxy* de la comunicació VNC (`vncproxy`) per a l'MV 101 (aquest és el nom de l'MV a Proxmox i no té res a veure amb el 101 utilitzat en el port en l'exemple anterior). Després es reinicia el servei:

```
service inetd restart
```

Per connectar-se amb una màquina externa, s'utilitza el mateix Tiger VNCviewer, malgrat que en aquest cas s'ha d'autenticar amb un usuari definit en el servidor.



Finalment, s'ha de tenir en compte que aquí només s'han mostrat les possibilitats bàsiques de l'hipervisor, però és important considerar que disposa de característiques de nivell avançat, com per exemple muntar una arquitectura d'alta disponibilitat amb clústers de servidors, connectar-se a diferents serveis

d'emmagatzematge, gestionar la seguretat mitjançant el tallafoc intern, fer còpies de seguretat i restaurar de diferents maneres o migrar «en calent», entre altres, que escapen a l'objectiu d'aquest subapartat.

2.4.2. Connexió *bridged* sobre una wifi

Perquè les màquines virtuals tinguin adreça pròpia dins de la xarxa, és necessari poder fer un *bridge* {Dbr] perquè diferents IP surtin pel mateix NIC físic: per exemple, el nostre servidor (KVM) i les màquines virtuals (per exemple, Promox); però si es vol muntar sobre una connexió wifi, hi ha diversos problemes (consulteu la documentació de *bridged connections*, que si bé s'indica per a Ubuntu16.04, és la mateixa per a versions posteriors). El problema principal és que la majoria de punts d'accés rebutgen, per seguretat, els *frames* que tenen una adreça d'origen diferent de la que s'ha autenticat. Com que Linux fa el *bridging* de manera transparent (no modifica ni els *frames* de sortida ni d'entrada), cal modificar això perquè cada paquet pugui provenir des de la mateixa IP d'origen.

Per a això, seguint la documentació [Ufk], s'ha construït un *script* que permet fer-ho sobre una màquina que disposa de KVM i que es connecta a través de la wifi NIC (wlo1). L'arxiu */etc/network/interfaces* només té la configuració de connexió a la wifi, com es mostra a continuació (és important desactivar o desinstal·lar el *NetworkManager* per evitar que hi hagi conflictes):

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
auto wlo1
iface lo inet loopback
#adequar si es disposa d'un altre mètode de connexió WEP p. ex.
iface wlo1 inet dhcp
wpa-ssid SSID-de-la-xarxa-Wifi
    wpa-psk PASSWD-de-la-xarxa-Wifi
    dns-nameserver 8.8.8.8
    dns-nameserver 8.8.4.4
#no s'utilitzarà ethernet wired
iface enp0s25 inet manual
```

Després es crea un *script* amb el contingut següent (en aquesta configuració s'utilitza una connexió *tap*, que és un dispositiu de xarxa virtual que simula un dispositiu en la capa de *link* i opera amb els paquets igual que si fossin *ethernet frames*, i s'utilitzen normalment per crear un *network bridge*):

```
#!/bin/bash
# Create br0 device
brctl addbr br0
# Create the tap device:
tunctl -t tap0
```

```
# Add tap0 to the bridge:
brctl addif br0 tap0
# IP to Br0
ip addr add 172.16.1.1/32 dev br0
ip link set br0 up
#Use parprouted to perform voodoo magic on the routing tables.
parprouted wlo1 br0
#start bcrelay - this will make sure that broadcast traffic
#(like the dhcp stuff) gets pushed through the tap as well.
bcrelay -d -i br0 -o wlo1
# It is necessary to add a routing rule between the ip assigned to the VM (in this case 10.0.0.100)
#for the packets can back to br0. Replace with the ip of the virtual machine (which is an IP
#inside the wifi network).
route add -host 10.0.0.100 dev br0
```

Una vegada que s'hagi executat l'*script* i s'hagin assignat les IP a les màquines virtuals, es podrà verificar que tenen connexió a la xarxa, i si la wifi està connectada a internet, també ho estaran les màquines virtuals i s'hi podrà accedir per mitjà d'una IP a la mateixa xarxa.

2.5. Hyper-V

Aquesta plataforma permet crear i administrar un entorn informàtic virtualitzat mitjançant la tecnologia de virtualització integrada a Windows. L'arquitectura del programari està formada pel mateix hipervisor, el servei d'administració d'MV, el control d'instrumentació (WMI), el bus de màquina virtual (*VMbus*), el proveïdor de serveis de virtualització (VSP) i el controlador d'infraestructura virtual (VID). L'administració es du a terme mitjançant una eina (GUI) administradora de Hyper-V, un complement Microsoft Management Console (MMC) i la connexió a màquina virtual, que dona accés a la sortida d'una MV per poder interactuar externament. A més, és possible interactuar mitjançant ordres específiques (anomenats *cmdlets*), que es despleguen sobre CLI i PowerShell.

La tecnologia inclosa a Hyper-V virtualitza el maquinari per proporcionar un entorn que permet executar diferents sistemes operatius alhora en un equip físic i administrar les MV, així com els seus recursos. L'objectiu de Hyper-V és proveir un entorn de *cloud* privat i adaptar-lo a l'ús en funció dels canvis en la demanda, amb la finalitat de prestar uns serveis de TIC més flexibles, amb la conseqüent reducció del maquinari (per la consolidació dels servidors i les càrregues de treball en un nombre inferior d'equips físics més potents), que disminueix globalment el consum de recursos com ara l'energia i l'espai físic. Amb aquesta infraestructura és possible disposar d'una infraestructura d'escriptori virtual (VDI) que contribueix a augmentar l'agilitat empresarial i la seguretat de les dades, i alhora simplifica l'administració del sistema operatiu i les aplicacions de l'escriptori.

Com a requisits de maquinari, requereix un processador de 64 bits amb maquinari compatible amb la virtualització (Intel VT-x o AMD-V) i amb DEP (*Data Execution Prevention*). Com a SO *guests*, Hyper-V és compatible amb màquines de 32/64 bits a Windows. Amb *guests* Linux i FreeBSD, admet (vegeu l'enllaç anterior per a consideracions particulars) CentOS/RHEL, Debian, SUSE, Oracle Linux, Ubuntu i FreeBSD. Hyper-V sobre Windows 10 Enterprise/Professional substitueix Microsoft Virtual PC, que va acabar el seu cicle l'abril del 2017.

Entre les raons per utilitzar Hyper-V amb entorns Windows, es poden enumerar les següents:

- Executar múltiples SO, i configuracions de programari i maquinari diverses amb diferents versions de el SO sobre la mateixa màquina física, amb el consegüent aïllament en entorns de prova o desenvolupament i amb una considerable reducció de la inversió, energia i espai.

- Desplegar un entorn de proves sobre un equip de desenvolupament (portàtil i d'escriptori) i després exportar-lo a sistemes de producció sobre els sistemes reals o sobre Azure. També és possible fer l'operació inversa sobre una MV en producció sobre la qual es detecten problemes: exportar-la i importar-la a l'entorn local per analitzar-los en un entorn de depuració.

- Analitzar les possibilitats del *virtual networking* per generar desenvolupaments semblants als entorns reals, però sense afectar-ne la continuïtat ni inserint màquines que puguin interferir en la comunicació.

Hi ha diferències que depenen de si Hyper-V s'executant en versions WServer o Windows 10. Una d'elles és que tenen un model de memòria diferent (sobre Server, només MV; sobre W10, comparteix espai amb altres aplicacions) i l'altra és que hi ha una sèrie de característiques que només són compatibles amb WServer (*Virtualizing GPUs Using RemoteFX*, *Live Migration*, *Hyper-V Replica*, *Virtual Fiber Channel*, SR-IOV *Networking*, *Shared* VHDX).

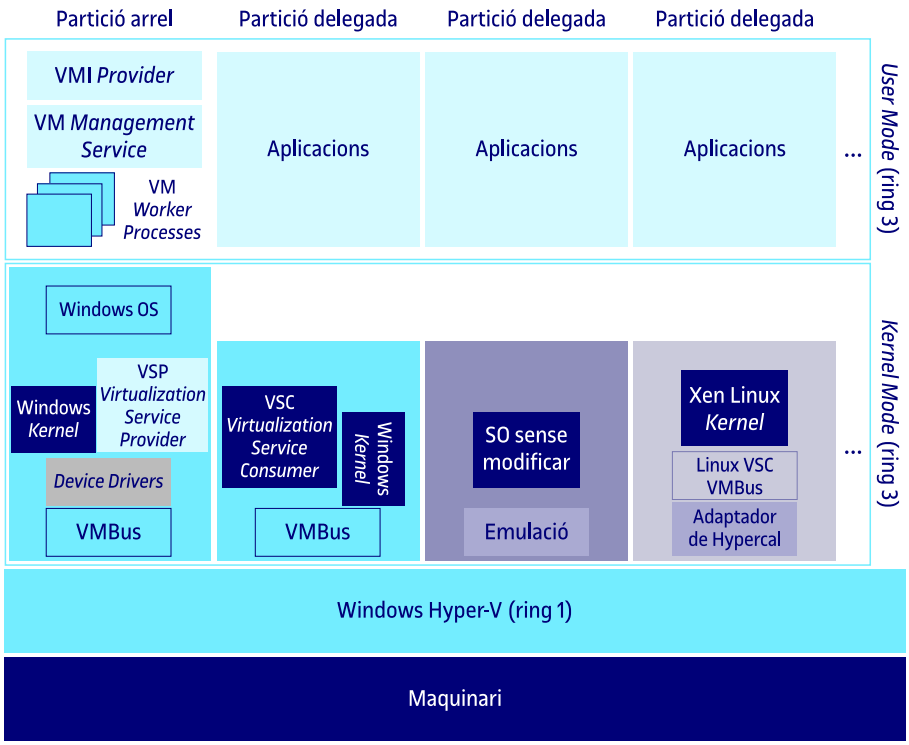
És convenient tenir en compte que hi ha algunes limitacions a la virtualització: per exemple, en aplicacions que depenen d'un maquinari específic i que no tindran un bon rendiment sobre una MV (jocs, aplicacions que requereixin un ús intens de GPU o temporitzadors que treballin per sota dels 10 ms, com ara *live music mixing*). Així mateix, les aplicacions que s'executin sobre el *host* i que siguin sensibles a la latència es veuran afectades quan s'executin sobre el *host*, perquè el mateix SO *host* s'executa sobre la capa de virtualització de Hyper-V.

Atesa la rellevància dels sistemes Windows en l'àmbit empresarial, Microsoft ha desenvolupat una estratègia enfront de la competència per cobrir una part important del mercat. De fet, el 2016 apareix al *Quadrant màgic* de Gartner com un dels proveïdors líders en solucions de virtualització per a x86 (juntament amb VMware) i ha mantingut les seves posicions en entorns empresarials, bàsicament pel domini de Windows Server. Dins d'aquesta estratègia, Microsoft

ha llançat (sense restriccions temporals) una plataforma anomenada Hyper-V Server (l'última versió disponible en aquest moment és del 2019). Hyper-V Server és una plataforma autònoma (*stand-alone*) que conté l'hipervisor, el Windows *Server Driver Model*, el suport per a la virtualització i components per a la clusterització d'alta disponibilitat, però no conté les altres característiques d'un SO Windows Server.

Aquesta plataforma és una solució important, ja que produeix una empremta molt petita i requereix uns recursos mínims; això la fa adequada per a empreses o institucions que necessiten consolidar servidors sense noves llicències o han d'executar uns altres SO juntament amb Windows. L'última versió inclou un conjunt de característiques addicionals que han estat afegides per donar suport a la virtualització (de categoria empresarial) i per integrar-la amb el *cloud* híbrid, així com millores en l'escalat i adaptació a diferents tipus de càrrega per ajustar-se a les necessitats de prestacions de sistemes crítics. A més, Hyper-V es pot activar a Windows 10 (Desktop) i a les versions *Windows Server*.

La figura següent mostra l'arquitectura de Hyper-V [Hay].



VMBus és un mecanisme dins de l'arquitectura Hyper-V que permet la comunicació lògica entre les particions i l'hipervisor, és a dir, és el canal de comunicacions intern per redirigir les peticions dels dispositius virtuals al maquinari, i viceversa; **VSC** (*Virtualization Service Client/Consumer*) és un dispositiu sintètic que està a les particions delegades o filles, utilitza els recursos proveïts pels *Virtualization Service Providers* (VSP) a la partició arrel (*parent*) i es comunica amb VMBus per satisfer les peticions d'E/S; **VMMS** (*Virtual Machine Management Service*) és responsable de manejar l'estat de l'MV a les particions delega-

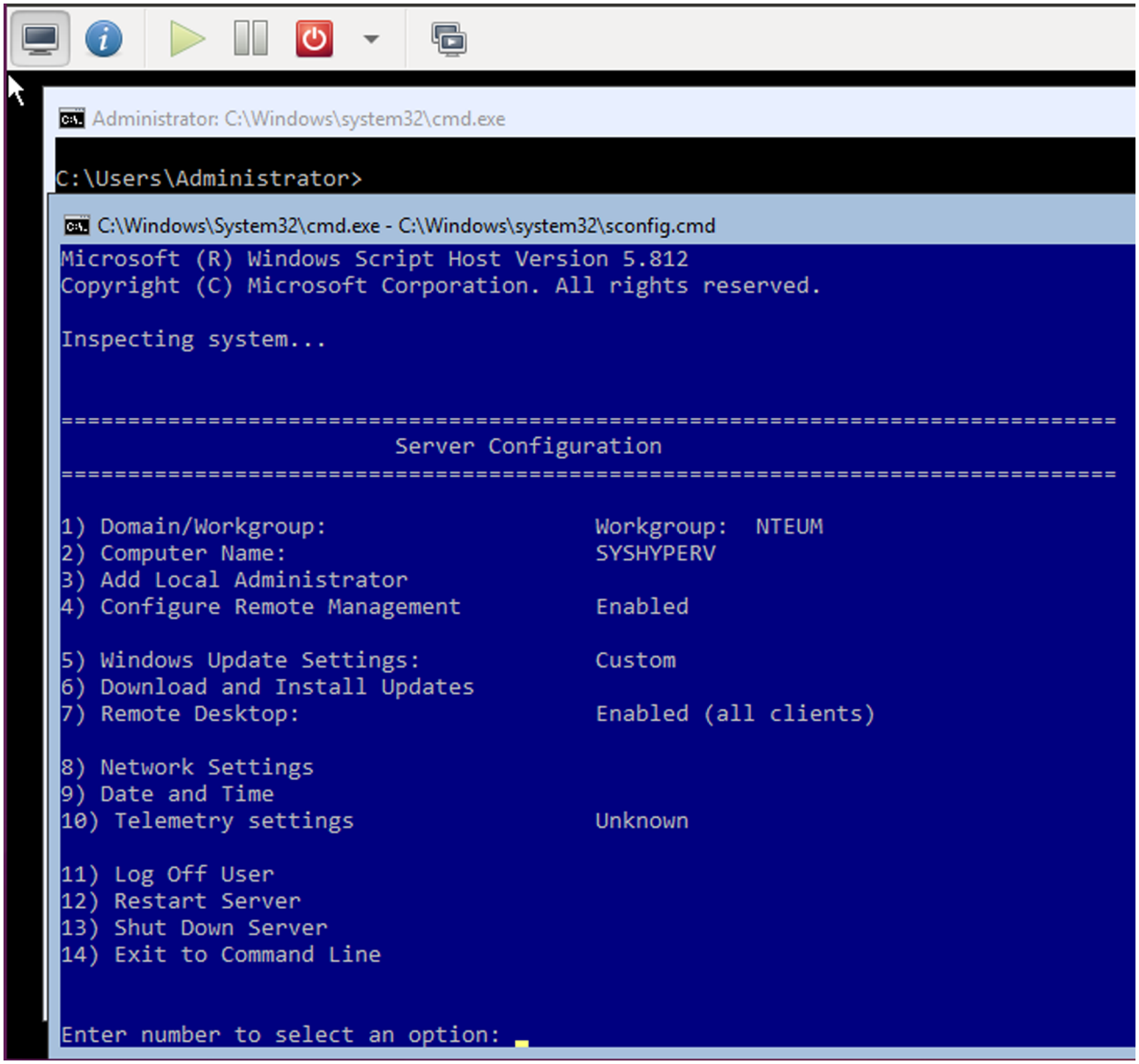
des o filles, i **VMWP** (*Virtual Machine Worker Process*) és un component que s'executa dins de l'espai d'usuari i que proveeix els serveis de gestió de l'MV al SO de la partició arrel (un per cada MV en una partició delegada).

2.5.1. Instal·lació de Hyper-V Server

Per instal·lar Hyper-V Server [Hyv], es descarregarà l'ISO del repositori de Microsoft, i en el nostre cas, com a prova de concepte, s'utilitzarà una màquina virtual de KVM amb 3 vCPU (però n'hi pot haver menys), 4 GB de RAM, 40 GB de disc i un dispositiu de xarxa de tipus compartit apuntant a un dispositiu de *bridge* de KVM (br0 en el nostre cas; cal recordar que s'ha de configurar el *bridged*, com s'ha indicat al subapartat de Promox, si es vol treballar amb un wifi com a connexió *bridged*). Després de crear la màquina i abans de la instal·lació, cal modificar sobre la configuració creada (*/etc/libvirt/qemu/*) el *CPU mode*:

```
virsh edit nom_màquina_hyperV
Modificar la línia de cpu mode a: <cpu mode='host-passthrough'/>
```

Després ja es pot arrencar la màquina i instal·lar Hyper-V responent a les preguntes que formularà (disc, *passwd*, que haurà de ser de tipus segur amb majúscules, minúscules, dígits...). Finalment, abans de reiniciar el sistema, cal treure l'ISO i iniciar l'MV, que podrà arrencar des del disc dur. Durant l'arrencada sol·licitarà que s'introdueixi Ctrl + Alt + Supr per accedir a la consola, que quedarà com la de la figura:



S'hi podran ajustar els paràmetres del *Workgroup* i el *Computer Name*, habilitar el *Remote Management* i el *Remote Display*, modificar els *Network Settings* per donar-li una IP dins de la nostra xarxa (a més de *netmask*, passarel·la i DNS), configurar *Date/Time* i, finalment, reiniciar el Hyper-V Server. Com que és un servidor complex, sobretot en els apartats d'administració i gestió remota, es recomana consultar la documentació [Hyv].

Per gestionar remotament el servidor, és necessari disposar de Windows 10 (o d'un altre servidor de Windows); en el nostre cas es farà sobre W10: s'ha d'activar a *Programs & features* → *Turn Windows features on-off* → *Hyper-V* → *Hyper-V management tools*. Cal seguir els passos indicats a [Mrs] per activar els permisos, segons si es té domini o no.

En el nostre cas no es disposa ni de domini ni de DNS per al segment, per la qual cosa el primer serà afegir a *C:/Windows/System32/drivers/etc/hosts* la màquina remota i el seu FDQN. Per a això, s'ha de desactivar l'antivirus si se'n

té (normalment eviten que aquest arxiu pugui ser modificat), i després obrir una finestra de `cmd` a través de «cercar» (Cortana, per exemple), però obrir-la amb el botó dret seleccionant l'execució amb permisos d'administrador. Després s'ha d'executar:

```
notepad C:/Windows/System32/drivers/etc/hosts
```

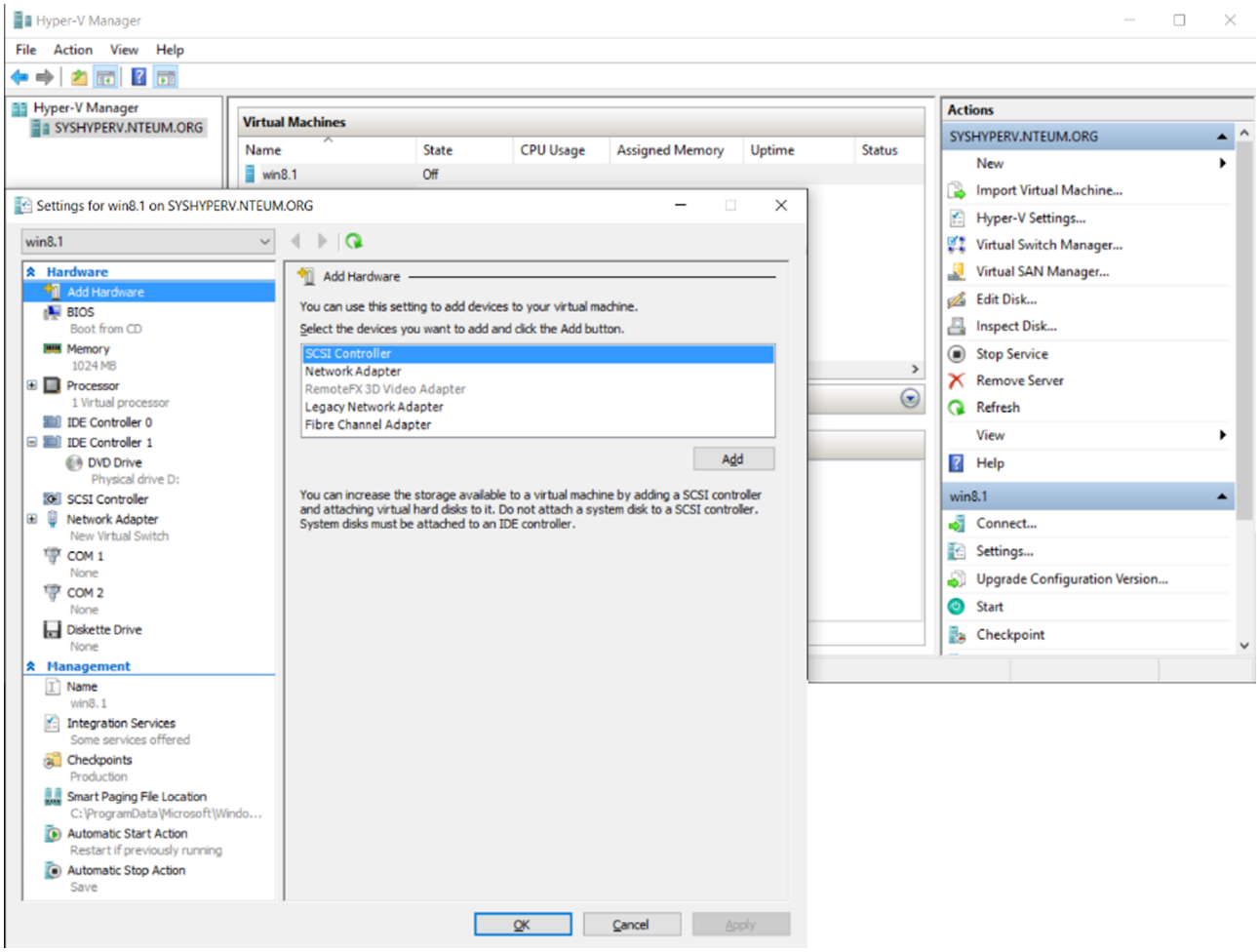
I sobre aquest arxiu s'ha d'incloure la IP i l'FDQN del servidor Hyper-V.

Després, sobre el servidor Hyper-V, s'ha d'activar l'opció d'administració remota i la màquina des de la qual es vol administrar el servidor remot; en una finestra de PowerShell, s'ha d'executar (reemplaçant *fqdn-of-hyper-v-host* pel nom FDQN de la màquina remota).

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "fqdn-of-hyper-v-host"
Enable-WSManCredSSP -Role client -DelegateComputer "fqdn-of-hyper-v-host"
```

Addicionalment, és necessari configurar el *group policy*: se'l crida des d'una finestra de `cmd`, s'executa l'ordre `gpedit` i es va a *Computer configuration* → *Administrative templates* → *System* → *Credentials delegation* → *Allow delegating fresh credentials with NTLM-only server authentication*, s'habilita i s'insereix `wsman/fqdn-of-hyper-v-host` (en el nostre cas, `wsman/syshyperv.nteum.org`). Cal recordar que s'haurà de disposar d'un usuari amb el mateix nom i *passwd* a les dues màquines i que la màquina gestora serà la que inicïi el Hyper-V Manager.

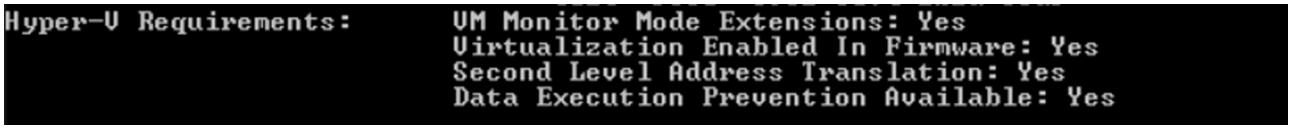
Després es podrà executar el Hyper-V Manager [Mrs]; afegint-hi un nou servidor, ja podrem tenir la connexió al servidor Hyper-V i fer totes les tasques d'administració i gestió de la infraestructura. La figura següent mostra la interfície d'aquesta eina:



És important comentar (com ja s'ha fet amb Promox) que Hyper-V és un entorn complex i amb moltes opcions, escenaris i configuracions possibles. Cal temps i dedicació per conèixer la infraestructura, els models de gestió, la seguretat, els permisos, els atributs, etc., que són necessaris per tenir-ho tot correctament configurat i executant-se. En aquest apartat només s'ha fet una prova funcional sobre l'hipervisor per mostrar-ne les possibilitats (que en són moltes i variades).

2.5.2. Instal·lació de Hyper-V sobre W10

Instal·lar Hyper-V sobre una màquina local Windows10 no té cap dificultat. Els requisits són un processador de 64 bits amb traducció de direccions de segon nivell (SLAT), extensió de maquinari (VT-x) i un mínim de 4 GB de memòria i des del BIOS; a més de les extensions de virtualització, ha d'estar habilitada la prevenció d'execució de dades (*Hardware Enforced Data Execution Prevention*). Aquesta informació es pot obtenir executant el PowerShell o consola (`cmd.exe`) i a continuació `systeminfo.exe`, en què, si tots els requisits que s'enumeren a Hyper-V tenen un valor de Yes, el sistema pot executar el rol de Hyper-V. Si algun element retorna No, s'hauran de comprovar i dur a terme els ajustos necessaris.



Per instal·lar Hyper-V, s'ha d'anar a *Programs & features → Turn Windows features on-off → Hyper-V*, activar-les totes i marcar després OK. Després de reiniciar la màquina en les eines d'administració, podrem trobar *Hyper-v Management* i iniciar la consola d'administració, la qual detectarà la màquina local i l'afegirà a la llista de servidors (en cas que no la detecti, s'ha de clicar a *add new server*).

Com a pas previ a crear una MV de Hyper-V, cal que es pugui connectar a la xarxa, i per això es crearà un *switch* virtual. Hyper-V té tres tipus de *switches*:

- **Extern**: connectivitat entre la xarxa virtual i un determinat NIC físic, que pot ser el del *host* o un altre NIC, si es vol aïllar el trànsit de les MV.

- **Intern**: la xarxa no està connectada a un NIC, però sí que hi ha comunicació entre el *host* i l'MV connectats al *switch*.

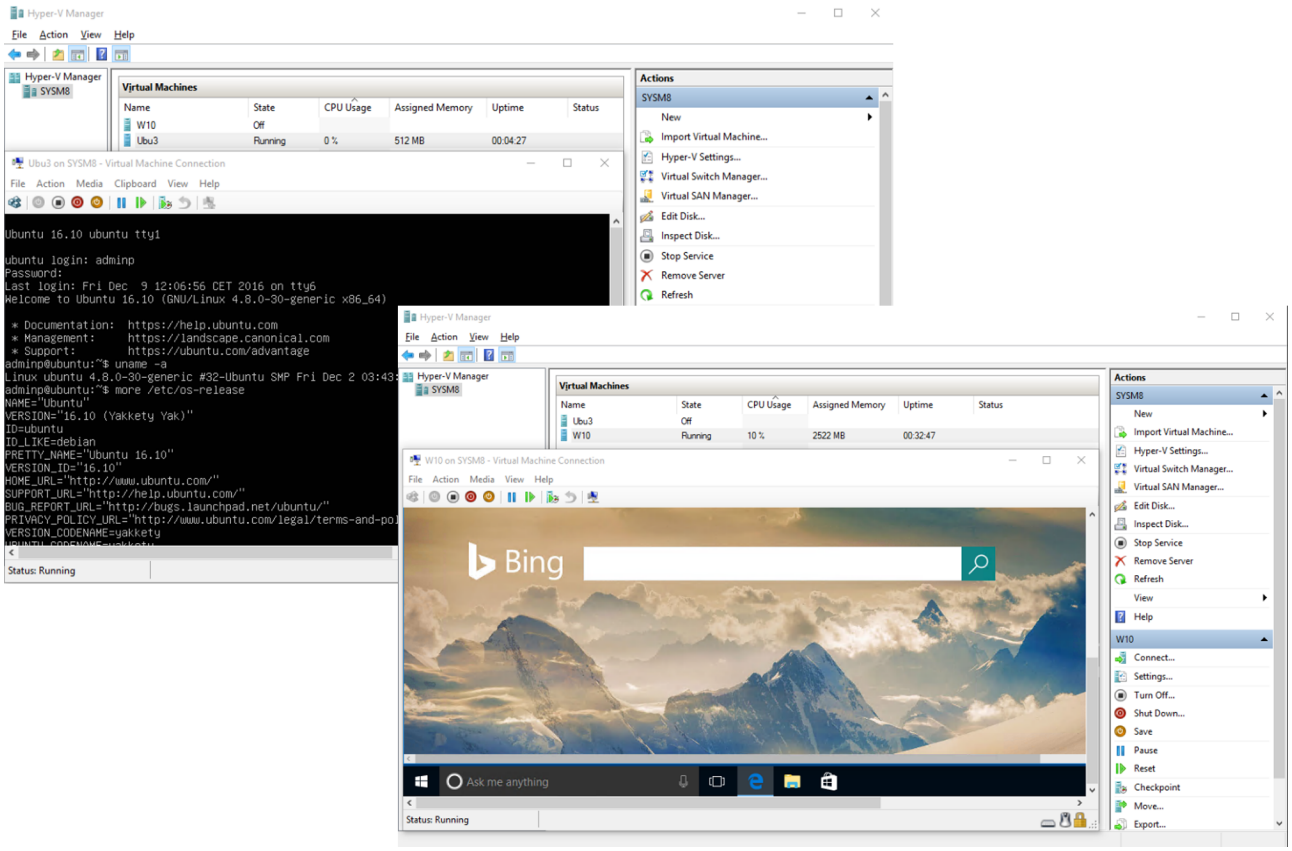
- **Privat**: no està connectat a un NIC ni hi ha connectivitat amb el *host* ni amb les MV connectades a aquest *switch*).

Per a això, s'ha d'accedir a *Virtual switches → New virtual network switch → Type of virtual switch → seleccionar External → Create virtual switch*, indicar un nom i verificar les propietats seleccionant el NIC sobre el qual es vol connectar i finalment *OK*.

Per crear una MV, es pot fer de moltes maneres, mitjançant Windows *Deployment Services*, amb un disc dur virtual preparat o, manualment, mitjançant un ISO, per exemple. En aquest cas s'instal·larà Ubuntu16.10 (Yakkety), del qual prèviament s'haurà baixat l'ISO ISO, i un W10. En aquest últim cas es poden baixar els ISO de prova del TechNet Evaluation Center, però s'ha optat per cercar una MV ja instal·lada de W10 i s'utilitzarà una de preparada per a Hyper-V des del lloc de prova Microsoft Edge (per a això, s'ha de seleccionar la versió i el tipus de màquina virtual Hyper-V i baixar el Zip que s'haurà de descomprimir).

A l'hora de crear una MV, cal seleccionar *New → Virtual Machine* i respondre a totes les preguntes per configurar la màquina virtual. Es recomana seleccionar màquines de Generation 1 i les quantitats de memòria i disc adequades (4 GB per a Linux), connectar-les al *switch* creat anteriorment i determinar on s'emmagatzemaran les MV i les seves propietats. En el cas de Linux, cal indicar que en la instal·lació es farà un ISO (indicant-li on està), i en el cas de W10, s'ha de seleccionar que s'utilitzarà un disc amb format *vhdx* (que es trobarà on anteriorment s'ha descomprimit l'arxiu Zip baixat).

Després es podran posar en marxa les MV, obrir una finestra per accedir a l'MV (a Connect, si s'utilitzen les màquines de prova Microsoft Edge, l'usuari és **IEUser** i el *passwd* **Passw0rd!**) i entrar-hi a través d'RDP des d'una altra màquina (per a això, en la màquina client, s'haurà de permetre la connexió a l'escriptori remot dins de les opcions del sistema). Les dues imatges a continuació mostren les pantalles de l'execució d'Ubuntu Yakkety i de W10.



Cal tenir en compte que W10 és molt exigent amb els recursos; s'ha de considerar una bona quantitat de memòria per als clients W10 en MV (2 GB mínim per a un funcionament acceptable).

La gestió i administració de la plataforma i els detalls avançats es poden dur a terme per mitjà de PowerShell. L'ordre que donarà una llista de tots els *cmdlets*, la funcionalitat dels quals es pot consultar a les Referències de Microsoft, és `Get-Command -Module hyper-v | Out-GridView`.

2.6. ESXi

VMware ESXi és un hipervisor de classe empresarial de tipus 1 desenvolupat i mantingut per VMware; el producte que el conté (VMware vSphere) figura com a líder empresarial al Server Virtualization Software Reviews and Ratings

de Gartner. Com tots els supervisors de tipus 1, inclou el seu propi sistema operatiu: aquesta és una de les peces principals de tota la infraestructura i productes de VMware [Ves].

Els dos components principals de vSphere Hypervisor són ESXi i vCenter Server. ESXi és la plataforma de virtualització que pot crear i executar màquines virtuals i *appliances* virtuals. vCenter Server és un servei que actua com a administrador central dels *hosts* ESXi connectats a una xarxa, permetent agrupar i administrar els recursos de múltiples *hosts*, i es pot instal·lar en una màquina virtual Windows o servidor físic, o es pot desplegar vCenter Server Appliance (vCenter preconfigurat sobre una MV Linux) sobre el mateix ESXi (5.5 o posterior).

Entre les característiques principals d'aquest hipervisor (en la versió 6.5) i de tota la infraestructura que l'envolta, es poden enumerar, entre altres [Wne]:

- Escalabilitat provada: nombre de *cores* per CPU física, o de 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 d'HA per a centres crítics.
- Còpia de seguretat i restauració en «calent».
- Migració d'un sol pas i actualitzacions sense interrupció del servei, de manera dirigida o automàtica, segons les condicions de QoS.
- API REST: per integrar-se amb altres eines.
- VSphere Client: canvi de l'antiga aplicació client a una GUI basada en HTML5, que assegura millors prestacions i serveis homogenis entre les plataformes.
- Seguretat escalable: polítiques de seguretat dissenyades per proveir el nivell de protecció volgut, sense complicar-ne la gestió o administració.
- Xifratge a escala d'MV per protegir contra accessos no autoritzats.
- Auditoria activa: per controlar totalment els usuaris i les accions tant actives com passives, incloent-hi informació per a l'anàlisi forense.
- Replicació de volums virtuals.
- *Secure boot*: per prevenir injeccions de components o modificacions *on-fly*.

- Suport per a MV i contenidors: de manera eficient, integrada i sense modificacions sobre el *guest*.

L'arquitectura VMware [EAr] ESXi inclou el sistema operatiu, anomenat VMkernel, i els processos que s'hi executen a la part superior. VMkernel proporciona els mitjans per executar tots els processos del sistema, incloent-hi les aplicacions i agents de gestió, així com les màquines virtuals i contenidors; controla tots els dispositius de maquinari al servidor, i administra recursos per a les aplicacions. Els principals processos que s'executen a la part superior de VMkernel són:

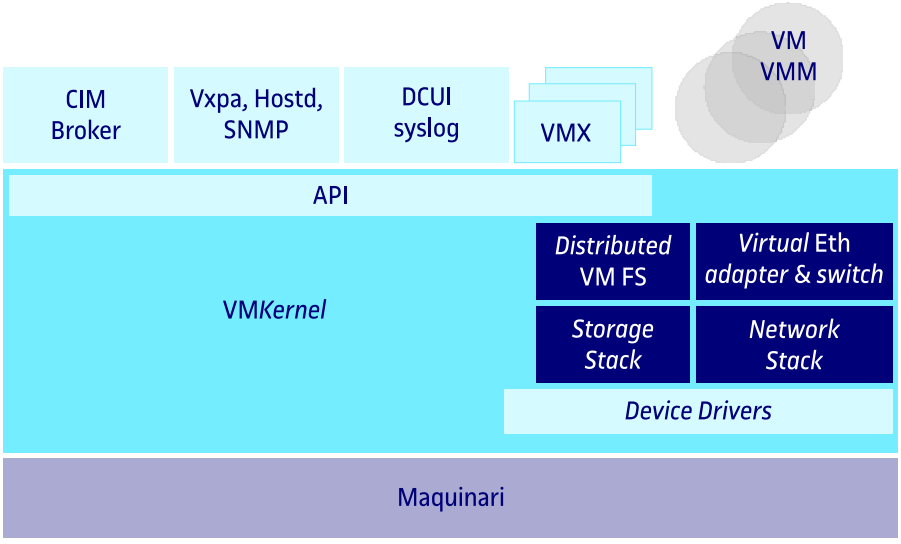
- Interfície d'usuari de consola directa (DCUI): interfície de configuració i administració de nivell baix, accessible a través de la consola del servidor, utilitzada principalment per a la configuració bàsica inicial.

- Monitor de màquina virtual: procés que proporciona l'entorn d'execució per a cada màquina virtual, així com un procés auxiliar conegut com a VMX. Cada màquina virtual en execució té els seus propis processos VMM i VMX.

- Agents d'administració: utilitzats per a l'administració i el monitoratge de nivell alt i des d'aplicacions remotes.

- Sistema comú d'informació (CIM): interfície que permet l'administració a escala del maquinari des d'aplicacions remotes a través d'un conjunt d'API estàndard.

La figura següent mostra els principals components de l'arquitectura ESXi.



Un dels aspectes, molt esperat pels usuaris, a partir de la versió 6.0u2, és una nova eina (gratuïta) de VMware que permet administrar els *hosts* ESXi a través d'un client web sense necessitat de servidor vCenter (com era necessari

per a les versions anteriors). La interfície de client d'ESXi Free Web (basada en HTML5) permet administrar un *host* sense necessitat del client de Windows, i s'incorpora amb les actualitzacions o s'instal·la fàcilment. Amb aquesta eina es pot gestionar tot el *host* tant en tasques bàsiques com avançades; crear MV (des de zero o des d'OVF/OVA); configurar els paràmetres del *host*; visualitzar resums, esdeveniments, tasques, notificacions i alertes; accedir a una consola a l'MV, i configurar la xarxa, entre altres; segons els experts, aporta més flexibilitat i facilitat a l'hora de gestionar el *host* que la seva predecessora. Aquesta opció és l'alternativa al vSphere Web Client basat en tecnologia Flash/Flex que hi havia a les versions anteriors, el qual s'havia d'instal·lar en una altra màquina (Windows o Linux) que tingués accés a la xarxa on es trobava el servidor ESXi.

Un dels dubtes habituals és en relació amb els noms dels productes i què integren: ESX i ESXi són en essència el mateix hipervisor, amb diferents serveis i mòduls que han evolucionat cap a la línia d'ESXi (tot i que van coexistir i VMware continua donant suport per a ESX). A més, VMware té una línia de productes anomenada vSphere, on s'integren diferents productes de virtualització, i de *cloud*, que des de la versió de vSphere 4.1 (actualment, VSphere 7.0) ja no conté la versió «clàssica» d'ESX, sinó la nova versió anomenada ESXi (així com totes les versions següents). Una de les grans diferències entre ESX i ESXi és que ESX contenia un entorn d'usuari anomenat *Console Operating System* (COS), o també *Service Console*, que es basava en una distribució de Linux i que s'utilitzava com a interfície d'administració i interacció amb el *vmkernel*. En el model ESXi desapareix el COS i la seva funcionalitat, ara sobre *vmkernel*, la proveeix un nou mòdul anomenat *PowerCLI + vCLI + ESXi Shell*, per la qual cosa l'hipervisor redueix la seva petjada, millora les prestacions, incrementa la seva seguretat i millora la interacció amb una nova API. Si es consulta en els productes per baixar i se selecciona l'ESXi de VMware, quan s'accedeix realment a la baixada, apareix com a *VMware vSphere Hypervisor* 7.0 U3b. Hi ha altres productes gratuïts de VMware, com ara VCenter Converter, que permet transformar màquines físiques (basades en Windows i Linux) i els formats d'imatges de tercers en màquines virtuals de VMware; la versió de *Workstation Player* esmentada en subapartats anteriors, i una aplicació de gestió de programari de VMware anomenada *Software Manager*.

Quan es baixa v7.x o v6.x (disponibles en el moment d'elaborar aquest material), demana una estimació de servidors físics (limitats a cent) i una llicència per provar tots els productes (HA, vMotion...) i la seva integració amb altres (per exemple, vCenter) durant seixanta dies, però en el moment de fer la baixada VMware proveeix la llicència definitiva, que s'haurà d'incloure sobre ESXi perquè passi a estat *Expiration date: Never*.

2.6.1. Instal·lació d'ESXi 6.5

Pel que fa als requeriments, ESXi necessita un mínim de 2 *cores* en un *socket* (es recomana *dualsocket* i 4 *cores* o més) i 6 GB (es recomanen 8 GB) de RAM, perquè si no, no s'instal·larà. A més, calen un adaptador de xarxa 1 Gbit/s

(es recomanen 2 Gbit/s) i un mínim de 35 GB de disc (es recomanen discos redundants). També s'aconsella (encara que no és necessari en primera instància) un espai compartit pels servidors per NFS, iSCSI o Fiber Channel a fi d'emmagatzemar les MV. ESXi no té límits per als *cores* per CPU física ni per als CPU per *host*, si bé hi ha, depenent de les versions, límits, de vCPU per *host* i límits de vCPU per MV, però són valors prou grans per instal·lar-lo, i fins i tot són d'alt rendiment.

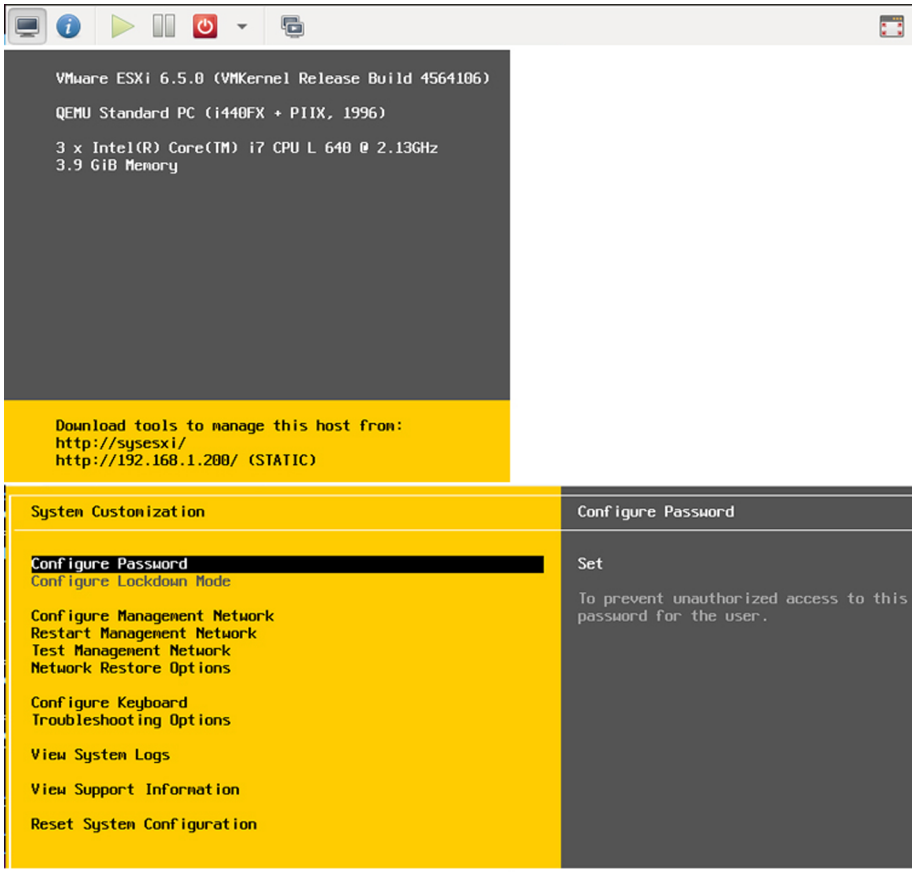
A la prova de concepte següent s'instal·larà ESXi (només per analitzar-ne la funcionalitat i no serà apte per a producció) i se'n provaran les característiques com una MV de KVM amb 4 GB de RAM, 3 vCPU, amb virtualització imbricada (*host-passthrough* en el *CPU-mode* de l'MV KVM), 50 GB d'espai de disc (perquè no s'utilitzarà un espai compartit i s'emmagatzemaran les MV al mateix disc) i un dispositiu de xarxa en mode *bridged* i de tipus *vmxnet3* (perquè si no, durant la instal·lació donarà un error, ja que el dispositiu no és físic). Per canviar aquest dispositiu, abans d'instal·lar la màquina, cal editar-ne la definició (no es pot fer des del `virt-manager`, per la qual cosa se'n pot triar un qualsevol, com per exemple e1000, i després reemplaçar-lo en la definició de la màquina):

```
virsh edit nom_maquina_esxi
```

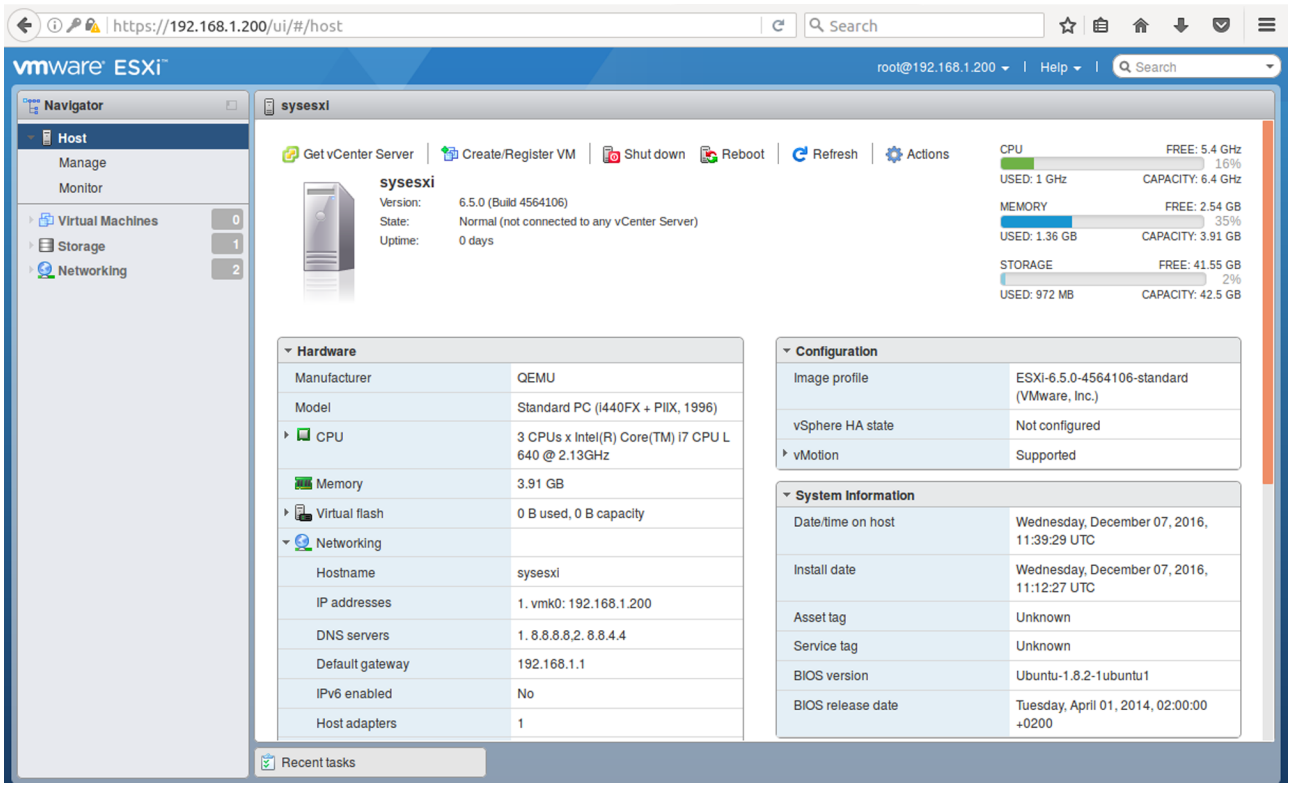
S'ha de canviar la interfície de xarxa on indica:

```
<interface type='bridged'>
   <mac address='aa:bb:cc:dd:ee:ff'/>
   <source bridge='dispositiu_bridge'/>
   <model type='vmxnet3'/>
   ...
</interface>
```

Amb això, ja es podrà inserir la imatge ISO baixada de VMware (després de crear un compte, verificar-lo i registrar-se per baixar la versió desitjada) i iniciar la instal·lació. Després d'algunes preguntes (disc, *passwd*...), tindrem la versió instal·lada amb l'habitual text de consola (gris i groc de VMware) i la URL per connectar-se a la interfície d'administració. En primera instància, la IP es gestionarà per DHCP, però si no hi ha un servei disponible n'assignarà una d'interna, per la qual cosa serà necessari accedir a la consola de l'ESXi i configurar-la. Per a això, s'ha de prémer F2, autenticar-se i ja es disposarà de la consola per fer els canvis necessaris (i altres ordres de gestió mínima del servidor). Les figures següents mostren la interfície d'ESXi i la de consola després de prémer F2 i autenticar-se.

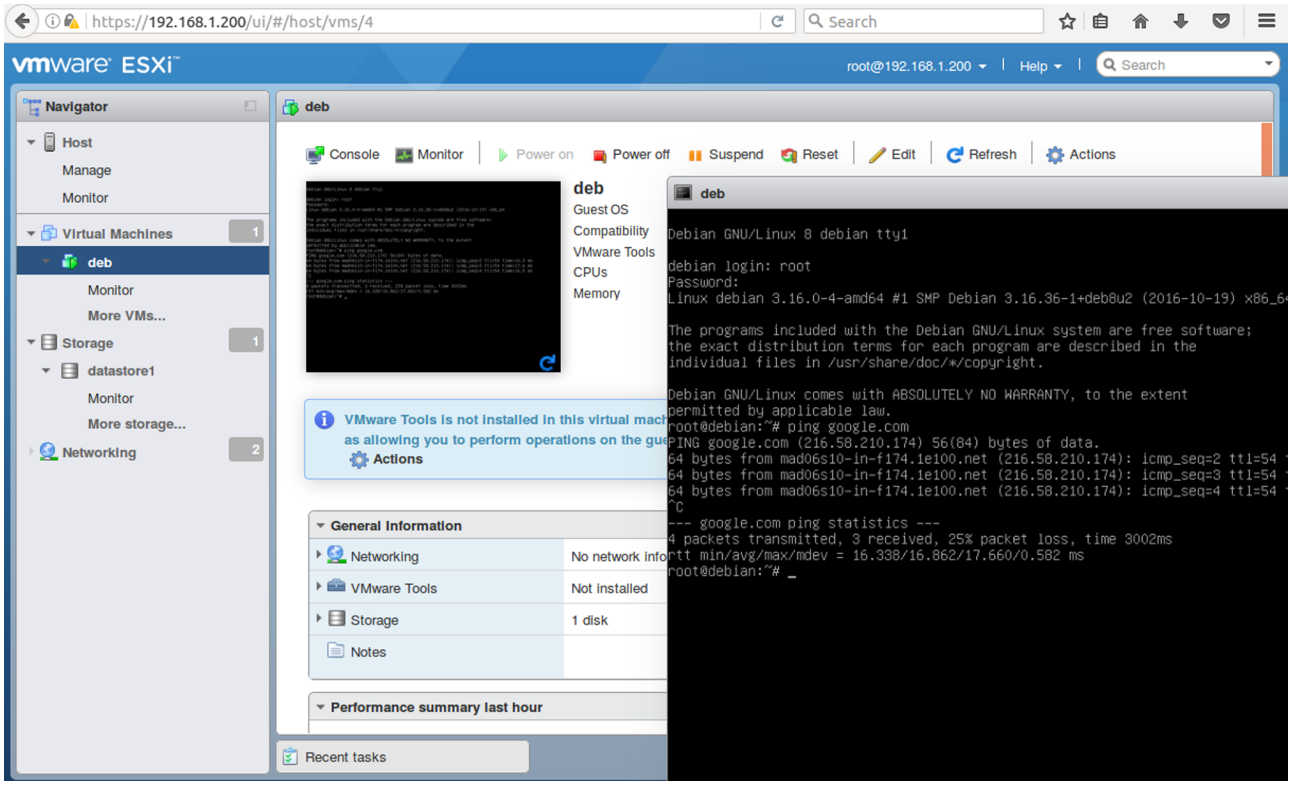


Una vegada configurat, s'hi pot accedir des de la URL indicada (després d'acceptar el certificat HTTPS i autenticar-se), i es mostra la interfície de gestió i administració d'ESXi, com es veu a la figura següent:



És important tenir present la inclusió de la llicència definitiva abans dels seixanta dies, la qual s'obté de la mateixa pàgina des d'on s'ha fet la baixada. Per a això, s'ha d'accedir a *Host* → *Manage* → *pestanya Licensing* i introduir la llicència obtinguda a la pàgina de baixada, i s'observarà que l'estat passa a *Expiration date: Never* i amb les *n* vCPU per MV (*up to X-way virtual SMP*). Cal tenir en compte que ja no és necessari l'antic client vSphere per a Windows per connectar-se al servidor ESXi, perquè ha estat reemplaçat pel nou, que funciona sobre la interfície web i és molt més àgil i flexible.

Una vegada disponible la interfície web, el funcionament és similar al dels altres hipervisors vistos: es podrà crear una xarxa associada al NIC i crear una MV carregant l'ISO des del disc local al servidor, definir-ne tots els paràmetres i configuració i posar-la en marxa. Les possibilitats de connexió seran a través del navegador (fins i tot des d'una màquina remota), `ssh` o utilitzant un producte VMware *Remote Console* (VMRC és de pagament per a Windows, Linux i MacOS). Sobre l'MV instal·lada, es rebrà un missatge per avisar que les VMware Tools no estan instal·lades, i es recomana fer-ho per millorar la interacció ESXi-MV (bàsicament, informació, encesa, apagament i reinici, entre altres). La figura següent mostra una imatge d'ESXi executant una Debian amb xarxa en mode *bridged* i sortint pel dispositiu de xarxa (i a la qual es pot accedir externament mitjançant la URL indicada, en aquest cas https://192.168.1.200/ui/#/console/4).



Una màquina virtual en VMware ESXi, com es pot apreciar al *datastore*, és bàsicament un directori amb el nom de la màquina i un conjunt de fitxers (text i binaris) que componen l'MV, incloent-hi el sistema operatiu i les aplicacions. Les extensions dels fitxers més importants que componen una màquina virtual són les següents: vmx (fitxer de configuració), vmdk (fitxer de disc virtual), nvram (fitxer BIOS) i log (fitxer de registre d'esdeveniments i errors de l'MV). En relació amb el maquinari disponible, es poden configurar d'1 a 8 vCPU, 1 TB de RAM, controlador SCSI, IDE, SSD (NVMe), CD/DVD, USB, controlador gràfic, controlador de xarxa, ports en sèrie i paral·lels, controlador de so, ratolí i teclat (per a tots ells, utilitzant els diferents controladors inclosos o podent, a més, afegir-hi controladors de tercers compatibles).

Una altra manera d'interacció amb la plataforma és a través de la línia d'ordres (CLI), utilitzant les ordres ESXCLI (`esxcli`), `vmk*` i Datacenter CLI (DCLI, per a VCenter), entre altres, que permeten dur a terme totes les tasques d'administració necessàries (sobre ESXi: `excli i vmk*`) amb gran detall i eficiència per a administradors experts [Cli].

És important tenir en compte que en aquesta prova tot és virtualitzat, ja que la Debian accedeix al NIC virtual d'ESXi (en mode *bridged*), que, al seu torn, accedeix al NIC virtual de KVM (en mode *bridged*), el qual està associat al dispositiu de xarxa real; no obstant això, com es pot veure, la seva funcionalitat està provada i no hi haurà cap diferència entre aquesta instal·lació i una que es faci sobre *bare-metal*. Com es podrà apreciar, no se li poden demanar gaires

prestacions; no obstant això, la fluïdesa i l'execució són més que acceptables (malgrat totes les virtualitzacions que es duen a terme) i demostra que és una plataforma consolidada, eficient i simple d'administrar i gestionar.

Com ja s'ha afirmat amb altres hipervisors, les potencialitats d'ESXi són molt grans, i en aquest subapartat només s'ha fet una petita demostració de les possibilitats de virtualització que proporciona, però cal consultar la documentació [Ves] (i la mateixa interfície d'administració) per tenir en compte les prestacions i característiques que pot oferir la plataforma (fins i tot la gratuïta), que la converteixen en una candidata d'excepció per a diferents escenaris i configuracions, siguin per a petites, mitjanes o grans instal·lacions.

3. Contenidors

Quan la virtualització es du a terme a escala del sistema operatiu, s'obtenen els contenidors (o virtualització basada en contenidors), que són el resultat d'aplicar una capa de virtualització sobre el nucli del sistema operatiu que permet que hi hagi múltiples instàncies aïllades d'espais d'usuari, en lloc de només una. Aquestes instàncies s'anomenen *contenidors* (terme utilitzat per LXC i Docker), però també reben altres noms, com ara *zones* (Solaris), *servidors privats virtuals* (OpenVZ), *particions* o *entorns virtuals* (VEs), *nucli virtual* (DragonFly BSD) i *gàbies* (*jail* FreeBSD).

La primera referència que es té de la virtualització a escala del sistema operatiu és la utilització del *chroot*, el qual rep els arguments següents `chroot [OPTION] NEWROOT [COMMAND [ARG]...` i permet executar l'ordre passat com a paràmetre impedint que aquell i els seus processos fills accedeixin a qualsevol ruta fora del directori passat com a argument. Aquesta manera d'executar-se és coneguda com a *gàbia chroot*. Basant-se en aquestes idees, FreeBSD, a principis dels anys noranta, introdueix gàbies (*jails*) amb l'objectiu de solucionar els problemes i limitacions del *chroot*, perquè aquest només limita el sistema d'arxius a què poden accedir els processos, però no la resta de recursos. Les gàbies BSD estenen aquest model virtualitzant, a més de l'accés al sistema de fitxers, la resta de recursos (usuaris, subsistema de xarxa, processos, etc.), fins i tot el rol del mateix usuari *root* dins i fora de la gàbia, la qual cosa permet crear un entorn virtual igual a la màquina real (o màquina virtual real), amb l'única limitació que, a diferència de les màquines virtuals, executen la mateixa versió del nucli de el SO.

Aquestes idees generen nous desenvolupaments, com els *namespaces* (2002), que permeten partir els recursos del *kernel*, de manera que un grup de processos vegi un conjunt de recursos diferent del que veu un altre grup de processos, i el 2006 Google publica per a el SO Linux *Process Containers*, que permet limitar i aïllar els accessos a recursos de la màquina (CPU, memòria, I/O de disc, xarxa, etc.) per a un grup de processos (el 2007, per qüestions de noms, els *Process Containers* es van passar a anomenar Control Groups o, simplement, *cgroups*), integrant-se amb el *kernel* de Linux. Si s'executa `systemd-cgls` sobre Linux, mostrarà l'arbre de grups i les seves dependències.

El 2008 s'ofereix, per a sistemes Linux, Linux Containers (LXC), que, utilitzant *namespaces* i *cgroups* del *kernel*, crea entorns de gestió de contenidors, i el 2013, després d'alguns anys de desenvolupament, es llança Docker, que permet gestionar el cicle de vida dels contenidors de manera senzilla en comparació amb les eines anteriors; ha tingut una gran acceptació per part dels desenvolupadors i un creixement exponencial en l'ús, ja que permet, dins dels contenidors, empaquetar aplicacions aïllades entre si amb capacitat total d'execució

i, a més, que es puguin integrar als fluxos d'integració o distribució contínua, generant un únic flux des del desenvolupament d'aplicacions fins a la posada en producció.

Docker va usar com a entorn d'execució per defecte LXC. No obstant això, més tard el va reemplaçar per libcontainer que permet utilitzar altres tecnologies d'aïllament (diferents de les d'LXC) i poder accedir directament a les API del *kernel* del sistema operatiu, la qual cosa redueix les dependències de llibreries i augmenta l'eficiència del conjunt.

No són els únics sistemes basats en la virtualització de el SO. Apache Mesos abstreu la CPU, la memòria, l'emmagatzematge i altres recursos informàtics de les màquines (físiques o virtuals), la qual cosa permet que es pugui construir un sistema distribuït elàstic i tolerant a les fallades, i que s'executi fàcilment de manera efectiva. Aquest sistema està especialment orientat a entorns de dades massives.

Windows també ha desenvolupat el seu propi ecosistema (*Windows Server Container / Hyper V Container*). No obstant això, la versió de Docker per a Windows, que utilitza la infraestructura pròpia de Windows, és la referència actual en el SO Windows.

Probablement, en el món dels contenidors, el fet més significatiu després de la seva creació va ser, el 2014, l'obertura per part de Google del projecte Kubernetes (també conegut com a K8s): una plataforma portàtil, extensible i de codi obert que permet administrar càrregues de treball i serveis en contenidors, i facilita una configuració declarativa, com l'automatització. Aquesta plataforma s'ha convertit en el referent a la indústria per al desplegament basat en contenidors, la gestió de càrregues i l'escalabilitat horitzontal, i és un dels entorns més utilitzats en els processos d'integració contínua del desenvolupament de programari.

Els experts classifiquen els sistemes de contenidors de la manera següent:

- **Contenidors d'infraestructura**, com LXC/LXD: poden executar múltiples instàncies de sistemes operatius de manera aïllada i permeten accedir als recursos (CPU, xarxa, I/O) de forma similar a una màquina virtual, però més ràpida i lleugera.

- **Contenidors de processos o aplicacions**, com `systemd-nspawn` o Docker (per defecte, a tot Linux): ofereixen un sistema d'encapsulament d'aplicacions; en permeten accelerar el desenvolupament i la distribució, eliminant les diferències d'entorn entre producció i desenvolupament, i faciliten la migració de l'entorn.

- **Contenidors de proves**, com FireJail, Bubblewrap, nsroot i nsjail: permeten l'aïllament mitjançant un entorn on es té accés restringit a recursos del sistema operatiu o dades de l'usuari (*sandbox*).

3.1. Linux Containers (LXC)

LXC (Linux Containers) és un mètode de virtualització en el nivell del sistema operatiu per executar múltiples sistemes Linux aïllats (anomenats *contenidors*) sobre un únic *host*. El *kernel* de Linux utilitza `cgroups` per poder aïllar els recursos (CPU, memòria, E/S, xarxa, etc.), la qual cosa no requereix iniciar cap màquina virtual. Els `cgroups` també proveeixen aïllament dels espais de noms, a fi d'aïllar del tot l'aplicació del sistema operatiu, incloent-hi l'arbre de processos, la xarxa, els ID d'usuaris i els sistemes d'arxius muntats. Mitjançant una API molt potent i eines simples, permet crear i gestionar contenidors de sistema o aplicacions.

LXC utilitza diferents mòduls del *kernel (ipc, uts, mount, pid, network, user)* i d'aplicacions (per exemple, Apparmor, SELinux Profiles, Seccomp Policies, *chroots* –pivot_root– i *control groups* –*cgroups*–) per crear i gestionar els contenidors. Es pot considerar que LXC està a mig camí entre un «potent» *chroot* i una màquina virtual, i ofereix un entorn molt pròxim a un Linux estàndard, però sense necessitat de tenir un *kernel* separat. Això és més eficient que utilitzar la virtualització amb un hipervisor (KVM, VirtualBox) i més ràpid de reiniciar, sobretot si s'estan fent desenvolupaments i cal reiniciar sovint, i té un impacte en el rendiment molt baix (el contenidor no ocupa recursos) i tots es dedicaran als processos que s'estiguin executant.

Com que LXC és compatible amb Ubuntu, la instal·lació i configuració és molt simple, com es pot observar a la *Getting Started*.

Les ordres són molt simples, i per crear un contenidor Debian, simplement s'executa: `sudo lxc-create -t download -n u1 -- --dist ubuntu --release DISTRO-SHORT-CODENAME --arch amd64`

Es podrà observar que, al cap d'uns instants, al directori */var/lib/xlc* hi ha un directori amb el nom del contenidor i no més de 300 MB de volum (dependrà de la mena de contenidor, però són valors habituals). És important observar la informació que es dona del contenidor quan es crea, ja que proporcionarà informació útil sobre usuaris i *passwd*. Les ordres més útils per treballar amb contenidors són els següents:

- `lxc-ls` per mostrar els contenidors i `lxc-info` per obtenir-ne informació (els que s'estan executant s'han d'afegir --active).

- `lxc-start -n debian8 -d` per posar en marxa el contenidor en segon pla (`-d = daemon`).

- `lxc-console -n debian8` per connectar-se al contenidor (amb el *passwd* generat per al *root* durant la creació). Per tornar a el SO del *host*, cal prémer les tecles Ctrl + a i després q.

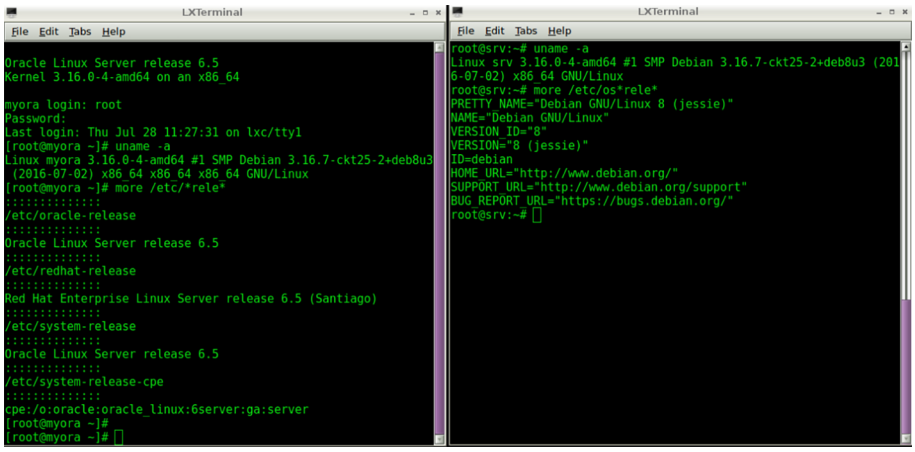
- `lxc-stop -n debian8` atura l'execució del contenidor.

- `lxc-destroy -n debian8` elimina el contenidor.

- `lxc-cloni debian8 debian8V2` clona un contenidor.

- Per muntar un directori del *host* en un contenidor, s'afegeix a */var/lib/lxc/containername/config* la sentència *lxc.mount.entry = /usr/bin /mnt none ro, bind 0.0*, que muntarà /usr/bin del *host* al directori /mnt del contenidor.

La figura següent mostra el *host* a la dreta (Debian) i el contenidor a l'esquerra (Oracle) funcionant sobre el mateix nucli (ordre `uname -a` des de cadascun).



És interessant instal·lar el paquet `lxctl` (`apt-get install lxctl`), ja que permet gestionar de manera més simple els contenidors LXC.

3.2. Docker

Docker és una plataforma oberta per desenvolupar, empaquetar i executar aplicacions, de manera que es puguin posar en producció o compartir més ràpidament, i que les separa de la infraestructura. Per això, té un cost molt inferior en recursos (bàsicament, espai de disc, CPU i arrencada) i permet que els desenvolupadors puguin disposar de tot tal com ho ha deixat el desenvolupador anterior sobre la infraestructura. Això significarà menys temps per provar i accelerarà el desplegament escurçant de manera significativa el cicle entre l'escriptura del codi i el pas a producció. Docker empra una plataforma (contenidor) de virtualització lleugera amb fluxos de treball i eines que l'ajuden

a administrar i implementar les aplicacions, i proporciona una forma en què gairebé qualsevol aplicació es podrà executar de manera segura en un contenidor aïllat.

Aquest aïllament i seguretat permeten executar molts contenidors de manera simultània al *host*, i atès el caràcter (lleuger) dels contenidors, tot això s'executa sense la càrrega addicional d'un hipervisor (que seria l'altra manera de gestionar aquestes necessitats compartint l'MV), la qual cosa significa que es poden obtenir millors prestacions i utilitzar més bé els recursos. És a dir, amb una MV, cada aplicació virtualitzada inclou no sols l'aplicació, que pot ser de desenes de Mbytes (binaris i biblioteques), sinó també el sistema operatiu *guest*, que pot ser d'uns quants Gbytes; en canvi, a Docker només hi haurà l'aplicació i les seves dependències, que s'executen com un procés aïllat a l'espai d'usuari de el SO *host*, compartint el *kernel* amb altres contenidors. Per tant, té el benefici de l'aïllament i l'assignació de recursos de les màquines virtuals, però és molt més portàtil i eficient, i així es transforma en l'entorn perfecte per donar suport al cicle de vida del desenvolupament de programari, test de plataformes, entorns, etc. [d1].

L'entorn està format per dos grans components: Docker [d2] (la plataforma de virtualització, o contenidor) i Docker Hub [d4] (una plataforma SaaS que permet obtenir, publicar i gestionar contenidors ja configurats). En la seva arquitectura, Docker utilitza una estructura client-servidor en què el client (CLI Docker) interactua amb el *daemon* de Docker, que fa el treball de construir, executar i distribuir els contenidors de Docker. Tant el client com el *daemon* es poden executar al mateix *host*, o un client es pot connectar a un *daemon* de Docker remot. El client Docker i el servei es comuniquen mitjançant *sockets* o una API RESTful. Per tenir més detalls sobre l'arquitectura, es pot consultar https://docs.docker.com/get-started/overview/.

És important no confondre una aplicació anomenada docker a Debian amb la plataforma Docker (per evitar confusions, a Debian es diu docker.io).

S'instal·la molt fàcilment, perquè Docker manté els binaris per a cada distribució, i el procés d'instal·lació es pot consultar a la pàgina web del desenvolupador sigui per a Linux, MacOS o Windows.

La manera més simple de verificar-ne el funcionament és executar un contenidor anomenat hello-world: simplement es posa `docker run hello-world`, es baixa una imatge del contenidor del Docker Hub, el posarà en marxa i mostrarà una pantalla de benvinguda. També es pot fer una prova amb el contenidor Ubuntu `docker run -it ubuntu bash` (com que el contenidor no existeix, el baixarà i l'executarà, per la qual cosa veurem el *prompt* del *bash* d'Ubuntu). Si fem `more /etc/os-release`, es podrà apreciar la versió de l'Ubuntu instal·lada. Amb `exit` (o Ctrl-D) se surt al *host*.

Són importants els conceptes *imatge d'un contenidor* i *contenidor* pròpiament dit. La imatge conté tots els elements i configuracions per crear el contenidor, però no és el contenidor. El contenidor es crea a partir de la imatge i es pot modificar, reiniciar o aturar, però els canvis que s'hi facin no es reflectiran en la imatge, que només és de lectura. Si es volen desar els canvis en un contenidor, s'haurà de fer una nova imatge amb el nou contenidor (docker commit)

Les ordres inicials per treballar amb contenidors són els següents:

- `docker`: mostra totes les opcions.
- `docker images`: enumera les imatges localment.
- `docker search patró`: cerca contenidors o imatges que tinguin aquest patró.
- `docker pull nom`: obté imatges del *hub*. El nom pot ser `<username>/<repository>` per als particulars.
- `docker run name cmd`: executarà el contenidor '*name*' i, a dins, l'ordre indicada per cmd.
- `docker ps -l`: obté la informació d'un contenidor (--all de tots els contenidors, fins i tot els que estan aturats).
- `docker commit ID name`: crea una imatge del contenidor amb totes les modificacions que s'hi han fet (amb 3-4 xifres de l'ID n'hi ha prou).
- `docker inspect`: permet veure la configuració exhaustiva d'un contenidor o una imatge.
- `docker push`: penja una còpia de la imatge al Docker Hub (s'ha de tenir un usuari, que és de franc) i estarà disponible per a altres usuaris o *hosts* que vulguin instal·lar-la.
- `docker cp`: copia arxius i directoris des del contenidor al *host*.
- `docker export/import`: exporta o importa el contenidor en un arxiu tar.
- `docker history`: mostra la història d'una imatge.
- `docker info`: mostra informació general (imatges, directoris, etc.).
- `docker restart`: reinicia un contenidor.
- `docker rm ID`: elimina un contenidor.
- `docker rmi ID`: elimina imatges.
- `docker start/stop`: inicia o atura un contenidor.

Com a prova funcional, s'instal·larà un servidor Apache sobre un contenidor Docker. Per a això, començarem amb un contenidor base Ubuntu, s'hi instal·larà Apache2 i, després, hi accedirem des del *host* fent un mapatge dels ports del contenidor. Per a això:

1) S'executa `docker run -it ubuntu /bin/bash` per accedir al contenidor. Si no està instal·lat, es baixa i s'instal·la.

2) Una vegada dins del contenidor, executem `apt-get update` i, a continuació, s'hi instal·la Apache2 (també s'hi instal·larà l'editor Vi) amb la següent ordre `apt-get install apache2 vim`. Es pot provar que Apache2 arrenca

sense problemes amb `systemctl restart apache2` (normalment només mostrarà un advertiment: que no té el nom de servidor configurat). També es pot veure amb les ordres `ps -edaf` o `systemctl status apache2`.

3) Es modifica la pàgina inicial per posar-hi un element relatiu a Docker i, d'aquesta manera, poder identificar la pàgina a /var/www/html/index.html.

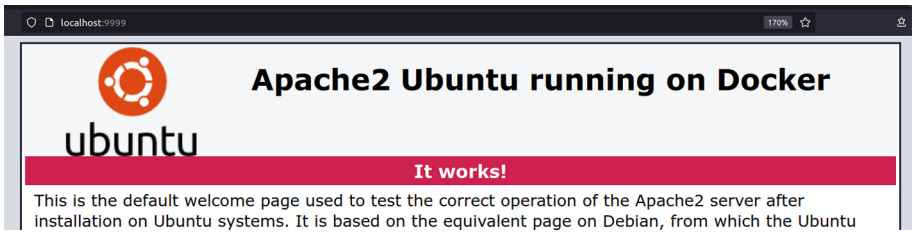
4) És important no sortir del contenidor, sinó fer Ctrl + P i, després, Ctrl + Q per sortir al *host*, perquè, si es fa Ctrl + D, el contenidor acabarà d'executar-se (també es pot fer des d'un altre terminal).

5) Amb el contenidor en marxa, verifiquem que està en UP amb `docker ps` i s'executa `docker commit ID_container apache`, en què `ID_container` és el nombre que mostra `docker ps`, i només s'han de posar les 3-4 primeres xifres. Aquesta ordre salvarà una nova imatge anomenada apache, que podem verificar amb `docker images`.

6) Ara ja és possible sortir del contenidor (Ctrl + D o exit) o apagar-lo amb l'ordre `docker stop ID_container`; `docker ps` ens haurà de mostrar que no hi ha cap contenidor Ubuntu en execució.

7) Per iniciar un contenidor a partir de la imatge creada, s'executa: `docker run -d -p 9999:80 apache /usr/sbin/apache2ctl -D FOREGROUND`, on amb -d s'indica que s'executi en mode *daemon* i amb -p s'assigna el port del contenidor (80) al del *host* (9999); i després s'executa l'ordre `apache2ctl` indicant-li que posi en marxa Apache2 en mode *Daemon* i en *FOREGROUND*.

8) Finalment, des del *host* es pot obrir un navegador amb localhost: 9999 o amb el nom de domini del *host* (si s'ha actualitzat /etc/hosts, com per exemple srv.nteum.org:9999), i veurem la pàgina modificada del contenidor.



Glossari

API web RESTful Manera de proveir interoperabilitat entre serveis i clients a internet.

appliance Aplicació de programari que es pot combinar amb JeOS (*just enough operating system*) per executar-se sobre un servidor o una màquina virtual.

bare-metal Màquina de maquinari «nua», és a dir, sense nucli (SO) instal·lat.

benchmark Programa o conjunt de programes que permet avaluar prestacions determinades d'un dispositiu.

bridged networking Entorn de xarxa configurat perquè els múltiples clients puguin tenir un IP dins de la xarxa sobre un mateix dispositiu físic.

CLI (*command line interface*) Línia d'ordres o de text de consola.

cloud computing També conegut com a *serveis en núvol*, *informàtica en núvol* o *núvol de còmput*, és un paradigma que permet oferir serveis TIC per mitjà d'una xarxa, que habitualment és internet.

cmdlet Ordre utilitzada a Hyper-V.

contenidor Entorn de programari virtualitzat a escala de el SO.

control d'instrumentació (**WMI**), **bus de màquina virtual** (**VMbus**), **proveïdor de serveis de virtualització** (**VSP**), **controlador d'infraestructura virtual** (**VID**), **Microsoft Management Console** (**MMC**): mòduls que componen la infraestructura de l'hipervisor Hyper-V.

dashboard Panell de control o instruments d'un entorn.

DHCP Servei que permet assignar els paràmetres de xarxa i informació relativa en un esquema de client-servidor.

Docker i LXC/LXD Plataformes que permeten la virtualització a escala de el SO.

ESXCLI `(esxcli)`, **`vmk*`** i **Datacenter CLI** (DCLI, per a VCenter) Diferents maneres d'interacció entre l'administrador i un *host* ESXi.

ESXi i vCenter Server Hipervisor i entorn de gestió de VMware.

extensions de maquinari VT-x/AMD-V Conjunt d'instruccions introduïdes per Intel i AMD per donar suport a la virtualització, que permeten accedir als recursos del processador guanyant en prestacions i eficiència.

front end Entorn de programari que interacciona amb els usuaris.

GlusterFS Sistema d'arxiu distribuït escalable.

guest Client (SO) d'un sistema virtualitzat.

hardware assisted virtualization Tècnica de virtualització que utilitza les extensions de maquinari del processador per millorar les prestacions.

high availability cluster Conjunt de servidors que poden prestar un servei ininterromput i sincronitzat.

hipervisor (*hypervisor*) Plataforma que permet aplicar diverses tècniques de control de virtualització per utilitzar, alhora, diferents sistemes operatius.

host Màquina (SO) que integra l'hipervisor i ofereix un servei de virtualització.

host-passthrough Paràmetre utilitzat per libvirt a fi de permetre la virtualització imbricada.

Hyper-Threading (H Technology, SMT)®, de l'empresa Intel, que permet a programes preparats per executar múltiples fils (*multi-threaded*) processar-los en paral·lel dins d'un únic processador, incrementant l'ús de les unitats d'execució del processador.

Hyper-V Hipervisor sobre el SO Windows.

interfície d'usuari de consola directa (DCUI) Interacció entre l'usuari i el SO/hipervisor.

IP address Número o adreça (lògica) que identifica un dispositiu en una xarxa.

iSCSI (internet SCSI) Estàndard que permet l'ús del protocol SCSI sobre xarxes TCP/IP, definit dins de la capa de transport.

ISO Arxiu que conté una còpia o imatge exacta d'un sistema d'arxius. Utilitzat per instal·lar programari o SO.

JSON-Schema Proveeix les regles per a un arxiu en format JSON requerit per una aplicació o servei i defineix com es pot modificar.

kernel Nucli de el SO.

Kernel-based Virtual Machine (KVM) Infraestructura de virtualització que permet transformar Linux en un hipervisor d'altes prestacions.

kW (quilowatt) Unitat de mesura de potència.

micronucli (*microkernel*) Codi de programari essencial (mínim) que pot proporcionar els mecanismes necessaris per implementar un sistema operatiu (SO).

Microsoft Remote Desktop Protocol (RDP) Protocol de Microsoft per comunicar-se quan s'executa una aplicació entre un terminal i un servidor.

monitor de màquina virtual (VMM, VMX) Procés que proporciona l'entorn d'execució per a cada màquina virtual, així com un procés auxiliar conegut com a VMX sobre ESXi.

multi-master cluster (per garantir la disponibilitat) Sistema multiservidor per garantir una alta disponibilitat.

NAT (*network address translation*) Mecanisme que permet que un dispositiu en una xarxa es pugui connectar a una altra xarxa (mecanisme implementat pels encaminadors).

nested virtualization Virtualització imbricada que permet a un hipervisor poder «traspassar» la interfície de maquinari a la seva MV, per permetre que, per exemple, es pugui executar un altre hipervisor.

networking Entorn de xarxa en un SO.

NFS, iSCSI LUN, Ceph RBD, Sheepdog Tecnologia d'accés a dispositius d'emmagatzematge distribuït.

NIC Network Interface Card.

no SPOF (*no single point of failure*) Evitar que quan una part del sistema falli, aquesta faci que tot el sistema falli.

open source Programari distribuït i desenvolupat lliurement que se centra en els beneficis pràctics de l'accés al codi font, més que en qüestions ètiques o de llibertat que són l'essència del programari lliure.

overhead Combinació de temps de càlcul excessiu o indirecte, memòria, amplada de banda o altres recursos que es requereixen per aconseguir una meta en particular.

paravirtualització Tècnica que permet mostrar una interfície virtualitzada a un SO o a part d'aquest, semblant al maquinari virtualitzat, però no idèntica.

password Paraula clau.

personal use and evaluation license (PUEL) Llicència d'ús lliure per a uns productes determinats.

plugin Complement que permet afegir una funcionalitat a un programari o a un SO.

PowerShell Intèrpret d'ordres de Windows, semblant als intèrprets d'ordres de Linux.

QEMU Emulador de processadors basat en la traducció dinàmica de binaris (conversió del codi binari de l'arquitectura font en codi comprensible per a l'arquitectura del *host*) i amb possibilitats de virtualització.

QoS (*quality of service*) Índex que mesura la qualitat d'un servei.

ROI (rendibilitat de la inversió) Relació que compara el benefici o la utilitat obtinguts amb la inversió que s'ha fet.

root Usuari administrador a Unix. Directori principal en els sistemes *Nix, identificat pel símbol '/'.

serveis TIC Conjunt d'activitats que responen a necessitats d'un client i que utilitzen sistemes informàtics. Alguns autors utilitzen *servei TIC* o simplement *TIC*.

sistema comú d'informació (CIM) Mòdul de comunicació entre diferents mòduls en ESXi.

sistema virtualitzat És el que s'executa sobre un hipervisor.

SLA (*service level agreement*) Contracte escrit entre un proveïdor de servei i el client per fixar el nivell acordat per a la qualitat d'aquest servei.

snapshot Estat o imatge congelat d'una MV i que és possible recuperar en el punt en què es va fer.

SSD (NVMe) Non-Volatile Memory Host Controller Interface Specification (NVMHCI) Especificació per a l'accés a les unitats d'estat sòlid (SSD) connectades pel bus PCI Express (PCIe).

TCO (*total cost of ownership*) Mètode de càlcul per determinar els costos directes i indirectes, així com els beneficis, relacionats amb la compra d'equipaments o programes informàtics.

Turnkey repositori d'*appliances*.

vCPU CPU virtual.

VirtIO *Driver* paravirtualitzat per a KVM Linux.

Virtual Network Computing (VNC) Programa basat en una estructura client-servidor que permet prendre el control de l'ordinador servidor remotament per mitjà d'un ordenador client.

VirtualBox Hipervisor *open source* desenvolupat i mantingut per Oracle (abans, Sun Microsystems).

VirtualBox *Remote Display Protocol* (VRDP) Implementació en Virtualbox de Remote Desktop Protocol (RDP).

virtualització Capa de programari que ofereix una versió virtual dels recursos de maquinari i programari subjacents.

VMware ESXi Hipervisor de VMware integrat dins de la línia de productes VSphere.

VMware *Remote Console* Aplicació de Windows que proveeix accés a la consola de les màquines virtuals sobre un *host* remot.

VMware Workstation Player Hipervisor per instal·lar sobre un SO (*hosted*).

Volume Group / Logical Volumes Part del Logical Volume Manager (LVM), que és un mòdul per gestionar volums sobre el *kernel* de Linux.

xarxa *host-only* Xarxa que permet comunicar un *guest* i el *host* per mitjà d'una interfície virtual.

Xen Projecte de virtualització orientat a l'eficiència mitjançant una tècnica anomenada *paravirtualització*.

XenServer Hipervisor *open source* desenvolupat i mantingut per Citrix.

WebSocket Canal de comunicació bidireccional sobre un *socket* TCP.

wireless Tecnologia de xarxa sense fil.

ZFS Sistema que permet emmagatzemar molts arxius i de gran volum.

Bibliografia

Bibliografia bàsica

Gavanda, M. et al. (2019). *Mastering VMware vSphere 6.7: Effectively deploy, manage, and monitor your virtual datacenter with VMware vSphere 6.7*. Packt Publishing

Ivanov, K. (2017). *KVM Virtualization Cookbook*. Packt Publishing.

Nickoloff, J.; Kuenzli, S. (2019). *Docker in Action*. Manning.

Portnoy, M. (2016). *Virtualization Essentials*. Sybex (Wiley).

Webgrafia

Tots els enllaços s'han visitat el novembre del 2021.

[ApM] Apache Mesos. <http://mesos.apache.org/>

[Cli] VSphere CLI. <https://pubs.vmware.com/vsphere-60/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-60-command-line-interface-concepts-examples-guide.pdf>

[Cpv] Comparison of platform virtualization software. <https://en.wikipedia.org/wiki/Comparison_of_platform_virtualization_software>

[Cwp] Computing with a price tag: VM cost calculation guide. S. Bigelow. 2016. Techtarget.com. <http://searchservervirtualization.techtarget.com/feature/Computing-with-a-price-tag-VM-cost-calculation-guide>

[Dbr] Bridging Network Connections. Debian. <https://wiki.debian.org/BridgeNetworkConnections>

[EAr] Architecture of VMware ESXi. <http://www.vmware.com/techpapers/2007/architecture-of-vmware-esxi-1009.html>

[Hvs] Hyper-V Server Evaluations. <https://www.microsoft.com/en-in/evalcenter/evaluate-hyper-v-server-2016 https://technet.microsoft.com/library/mt169373.aspx>

[Hya] Arquitectura de Hyper-V. Microsoft. <https://msdn.microsoft.com/en-us/library/cc768520(v=bts.10).aspx https://www.microsoft.com/en-us/download/details.aspx?id=29189>

[Hyp] Virtualization using Hyper-V on Windows 10. <https://msdn.microsoft.com/virtualization/hyperv_on_windows/index>

[Hyv] Hyper-V. <https://technet.microsoft.com/library/mt169373.aspx>

[Iar] Icones amb llicència d'ús lliure. <http://www.iconarchive.com> <http://www.customicondesign.com> <http://icons8.com>

[Kne] KVM Networking Ubuntu. <https://help.ubuntu.com/community/KVM/Networking>

[Ksw] Kvm on Server World. <https://www.server-world.info/en/note?os=Ubuntu_16.04&p=kvm&f=5>

[Kvm] Kernel Virtual Machine. 2016. <http://www.linux-kvm.org/>

[Lib] Libvirt Networking. 2016. <https://wiki.libvirt.org/page/Networking>

[Lvh] Libvirt Networking Handbook. Version 1.0.1. Jamie Nguyen. 2015. <https://jamielinux.com/docs/libvirt-networking-handbook/>

[Mrs] Manage Remote Hyper-V Hosts with Hyper-V Manager. <https://msdn.microsoft.com/en-us/virtualization/hyperv_on_windows/user_guide/remote_host_management>

[NoV] NoVNC. <https://kanaka.github.io/noVNC/>

[Pve] Proxmox Virtual Environment. <http://pve.proxmox.com/pve-docs/pve-admin-guide.html>

[Pwi] Proxmox Wiki (HOWTOs). <https://pve.proxmox.com/wiki/Main_Page>

[Sht] Spice html5 client. <https://www.spice-space.org/page/Html5>

[Spi] Spice Server & Clients. <https://www.spice-space.org/>

[Ufk] Virt-Manager Bridged Wireless Network. Ubuntu Forums. <https://ubuntuforums.org/showthread.php?t=1766674>

[Ves] VMware vSphere 6.5 Documentation Center. <https://pubs.vmware.com/vsphere-65/index.jsp#com.vmware.vsphere.doc/GUID-1B959D6B-41CA-4E23-A7DB-E9165D5A0E80.html>

[Vir] VirtualBox. <https://www.virtualbox.org/>

[VSO] Virtualització a escala del sistema operatiu. <https://es.wikipedia.org/wiki/Virtualización_a_nivel_de_sistema_operativo>

[Vwp] VMware Workstation Player. <http://www.vmware.com/products/workstation.html>

[Xag] Citrix Xen Server 7.0. Administration Guide. <http://docs.citrix.com/content/dam/docs/en-us/xenserver/xenserver-7-0/downloads/xenserver-7-0-administrators-guide.pdf>

[Wne] What's New in VMware vSphere®.6.5. <http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/vsphere/vmw-white-paper-vsphr-whats-new-6-5.pdf>