

Proxmox:

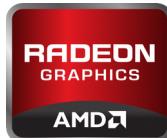
Alta disponibilidad, CEPH, ZFS.

GPU-Passthrough:

Windows, Mac OS.

Sistemas Multipuesto:

Un PC varios usuarios.



.Org Foundation



Índice

Teoría.....	5
Introducción.....	5
Objetivo.....	5
¿Qué es Proxmox?.....	6
Que vamos a hacer mal (A propósito).....	7
Configuración adicional.....	7
El Futuro™	8
Proxmox – Manos a la obra.....	9
Instalación de Proxmox.....	9
Clusterización.....	16
Almacenamiento.....	19
ZFS.....	19
Swap.....	22
CEPH.....	23
GPU-Passthrough.....	32
Introducción.....	32
Limitaciones PCIe / Hardware.....	32
Identificación de compatibilidad.....	33
Configuración.....	35
Máquina Virtual: Nvidia & Windows.....	36
Máquina Virtual: AMD, BIOS temporal & Windows.....	40
Máquina Virtual: Nvidia & Mac Os & AMD.....	41
Máquinas Virtuales y Físicas: Intercambio de drivers.....	51
Introducción.....	51
Configuración.....	51
Sistemas multipuesto.....	53
Introducción.....	53
Configuración.....	53
Windows.....	53
Linux.....	59
Fuentes.....	63

Teoría

Introducción

Este proyecto surge de la curiosidad sobre el futuro de la virtualización así como de la necesidad de encontrar una mejor alternativa al uso de dual boot entre Windows y Linux, ya que es ineficiente reiniciar a Windows cuando quieras jugar y reiniciar a Linux para utilizar el ordenador para el resto de tareas. También aprovecharemos la oportunidad para virtualizar MacOs X.

Este proyecto podría considerarse una continuación de uno de los trabajos hechos con el profesor Javier Puche, donde se nos encomendaba hacer un presupuesto llamativo sobre una plataforma hardware.

En un primer momento se va a montar una alta disponibilidad con 3 nodos, tanto de máquinas virtuales como para almacenamiento de datos (CEPH). También utilizaremos ZFS, un sistema de ficheros orientado para altas prestaciones.

Dedicaremos bastante énfasis en la utilización de tarjetas gráficas físicas en máquinas virtuales Windows y MacOsX. Para ello haremos uso de la tecnología VT-d, que nos permitirá la asignación de dispositivos I/O.

Para realizar este proyecto tenemos disponibles 3 ordenadores:

- 1x Sobremesa I7-5280k (6 núcleos, 12 hilos), Gigabyte X99-UD4, 16GB RAM, 256GB SSD (Sistema), 2TB HDD(1TB para CEPH, /home, /tmp y /var; 1TB para zfs stripe), 2x 1TB zfs stripe (almacenamiento de maquinas y Swap), GPU Nvidia GTX970, GPU Nvidia 610, GPU AMD 7770
- 2x Portátil I5-2410m (2 núcleos, 4 hilos), 4GB RAM, 128GB SSD (CEPH)

Una vez terminado el proyecto, los portátiles y el CEPH serán desactivados, quedando el sobremesa como equipo de uso habitual, ya que cuenta con la suficientes potencia como para tener un pequeño laboratorio para experimentar.

Para la creación del proyecto me basaré en hardware comercial, por lo que no se tendrá en cuenta hardware para entorno empresarial. Por lo tanto, las opciones de NVIDIA Grid y AMD FirePro no serán valoradas.

Las opciones de configuración se basaran para una plataforma Intel sin GPU integrada, por lo que las opciones para AMD o para procesadores Intel con gráfica integrada pueden ser diferentes

Objetivo

Nuestro objetivo es aprender y resolver las dificultades que conlleva administrar un sistema complejo de virtualización, así como aprender a aprovechar las ventajas que nos puede brindar la virtualización como arma de uso diario, las cuales pueden ser bastantes.

También, se aprovechará este trabajo para subirlo a diferentes foros y como parte de la carta de presentación para ciertas empresas españolas que utilizan Proxmox para ofrecer sus servicios. Ya que no hay nada mejor que trabajar de algo que realmente te motiva.

¿Qué es Proxmox?

En primer lugar, podemos considerar Proxmox como un frontend web de KVM (Virtualización completa) y LXC (Contenedores) con añadidos interesantes como utilidades propia de backups, gestión de usuarios, firewall, etc...

KVM es un proyecto de código abierto el cual se distribuye como módulo para el núcleo de Linux. se encarga de proveer las extensiones de virtualización (VTx, AMDv). Trabaja conjunto con QEMU, que se encarga de simular el resto el hardware para la máquina virtual.

Ambos son proyectos hermanos, compartiendo la mayoría de código.

LXC es un proyecto de código abierto el cual se distribuye como paquete. Es un método de virtualización para equipos derivados de unix que permite aislar instancias independientes de diferentes distribuciones pero compartiendo kernel y recursos del sistema. Funciona de una manera similar a chroot pero ofrece un nivel de aislamiento mayor.

Para montar Proxmox los desarrolladores use basan en Debian stable para ofrecer la paquetería y la base del sistema.

Para el kernel utilizan el publicado por Ubuntu, pero adaptado a las necesidades de Proxmox.

A pesar de que en muchos sitios explican que Proxmox es un hypervisor de nivel 1, este no lo es, ya que como veremos más adelante, podremos añadir repositorios de Debian a Proxmox e instalar cualquier paquete, como por ejemplo un entorno gráfico.

Realmente la "magia" no la hace Proxmox, si no los desarrolladores que contribuyen a los proyectos de software libre de kvm,lxc,ceph,zfs... lo que no le quita el merito a Proxmox de crear un buen entorno web para una rápida y sencilla administración.

Otros frontend para kvm pueden ser: Ovirt, Virt-Manager....

Proxmox se distribuye en 3 repositorios:

- Testing: En el cual se van probando funcionalidades a añadir en futuras versiones.
- No subscription: Repositorio gratuito con paquetes estables pero no recomendados para ambientes de producción.
- Enterprise: Repositorio de pago, con paquetes estables y testeados. Su precio varia desde 70€ por año y por socket de CPU hasta 796€ por año y por socket de CPU, dependiendo del nivel de soporte técnico que queramos contratar.

Dentro de lo bueno/malo que nos puede aportar Proxmox es que nos aísla en parte de las tecnologías utilizadas en segundo plano, como puede ser heartbeat, corosync, Qdevice, comandos propios de KVM,LXC... Bajo la filosofía Windows esto es bueno, bajo la filosofía Linux es "No sabes lo que estas haciendo" y en parte, tiene razón, dado que si tuviésemos que configurar todo lo que vamos a hacer fuera de Proxmox, requeriría mucho más esfuerzo y conocimiento.

Como desventajas que nos podemos encontrar con Proxmox con por ejemplo VmWare, es que Proxmox hace uso intensivo del disco duro del sistema, lo que hace inviable instalarlo en una memoria USB, cosa que VmWare si admite. Esto nos hace tener que invertir un presupuesto mayor en el disco duro para el sistema. Tampoco hay proyectos funcionando para implementar clusters de tipo fault tolerance.

Una ventaja que ahora mismo esta en desarrollo y que nos puede dar otro punto de vista sobre la creación de una alta disponibilidad es la migración en vivo de maquinas virtuales almacenadas localmente en un nodo (fichero de disco, archivos de configuración, y memoria de la maquina) hacia otro nodo con la suficiente capacidad para recibir la maquina. Esto lo que hace es ofrecernos una especie de "falsa alta disponibilidad" a lo barato, pero que para servicios no críticos nos ofrece una versatilidad y una media de coste/características muy sugerente.

Para hacernos una idea de hasta donde llega kvm: a finales de 2017 Amazon migró su nube AWS del hypervisor xen a kvm, utilizando además la tecnología SR-IOV (que explicaremos más adelante) para red y almacenamiento.

Que vamos a hacer mal (A propósito)

Vamos hacer una serie de acciones que están totalmente desaconsejadas para un ambiente de producción, pero dado que no tenemos nada crítico, nos vamos a permitir la concesión de hacerlas, pues no tenemos los medios suficientes para hacerlas correctamente:

-CPU's diferentes, se desaconseja en mayor medida para mejor funcionamiento del HA.

-Mezclar diferentes tipos de almacenamientos, tanto de tecnología (SSD con HDD) como de utilizar imágenes montadas como discos duros. Esto esta totalmente desaconsejado, sería literalmente de muy pocas luces hacer esto en un ambiente de producción.

-Misma red para el CEPH que para el HA, y encima de solo 1GB. Si ya es una locura correr la sincronización en vivo de grandes cantidades de datos en una red de solo 1GB, imagine hacerlo también con la HA.

-Utilizar memoria ram No-ECC. Totalmente recomendado utilizar memoria ECC, sobretodo para el almacenamiento, ya que al estar totalmente desaconsejado utilizar tecnologías hardware por raid, el trabajo de que la integridad de los datos es de la ram del sistema.

-Habilitar repositorios de Debian y ponerte a toquetear con backports, drivers gráficos... A no ser que quieras una gran bola de sistema operativo con mitad de paquetes testing y mitad stable.

Configuración adicional

Dado que el equipo de sobremesa estaba previamente con Fedora como sistema principal en el ssd y Proxmox en un hdd a modo de testing, hemos utilizado Clonezilla para hacer una imagen de Fedora por si hay que recuperar algo y acto seguido copiamos la partición de Proxmox al ssd.

Tras comprobar que el arranque es correcto, borramos la partición del hdd en el que estaba instalado Proxmox para prepararnos para el ZFS.

Conectaremos los equipos a un switch Netgear Prosafe GS108E.

Una vez instalado el sistema operativo en los portátiles, editaremos el archivo **/etc/systemd/logind.conf** y bajo la sección login añadiremos las siguientes líneas:

```
HandlePowerKey=suspend  
HandleLidSwitch=ignore
```

Para suspender los equipos utilizaremos el comando: “**systemctl hybrid-sleep**”. Despues, para levantar los equipos de la suspensión con Wake-On-Lan, usaremos el paquete “etherwake” con el siguiente comando: “**etherwake MAC -i INTERFAZ_RED**”.

Dado que contamos con 16GB de RAM, en un primer momento no se iba a utilizar SWAP, pero dado que contamos con suficiente espacio y que ZFS suele utilizar bastante memoria RAM, vamos a montar un SWAP de al menos 4GB y vamos a reducir el swapiness en torno al 10 añadiendo al fichero **/etc/sysctl.conf** la linea **“vm.swappiness = 10”**

La ayuda para implementar CEPH en Proxmox se basa en discos duros físicos, pero la implementación original nos da la opción de poder crearla en particiones o bien, simulando un disco duro (*.img) (Véase apartado "que vamos a hacer mal").

Para la creación del archivo ejecutaremos: `fallocate -l 25G`

`/var/lib/vz/images/cephvirtual.img`

Para montarlo como si fuese un disco: `losetup /dev/loop0`

`/var/lib/vz/images/cephvirtual.img`

Para crear el ceph: `ceph-disk prepare --zap-disk --cluster`

`ceph --cluster-uuid UID --bluestore /dev/loop0`

Para levantar ceph con ese disco: `ceph-disk activate /dev/loop0p1`

El uuid del cluster se puede comprobar en “`/etc/ceph/ceph.conf`”

En cada reinicio del sistema tendremos que ejecutar los siguientes comandos para arrancar el disco virtual.

Para montar la imagen como disco:

`losetup /dev/loop0 /var/lib/vz/images/cephvirtual.img`

Para detectar las particiones: `partprobe /dev/loop0`

Para levantar ceph con ese disco: `ceph-disk activate /dev/loop0p1`

Una vez las máquinas estén creadas, se añadirán los repositorios sin suscripción:

`echo deb http://download.proxmox.com/debian/pve stretch pve-no-subscription >> /etc/apt/sources.list`

Se agregará la clave GPG del repositorio a la carpeta de apt de claves confiadas con el comando:

`wget http://download.proxmox.com/debian/proxmox-ve-release-5.x.gpg -O /etc/apt/trusted.gpg.d/proxmox-ve-release-5.x.gpg`

Actualizaremos los nodos para que todos tengan las últimas versiones publicadas.

El Futuro™

El futuro que nos espera para estas técnicas pasa por la implementación de SR-IOV.

Haciendo un símil del funcionamiento, es como un switch de red, que solo teniendo una tarjeta de red instalada, podemos acceder a otros puertos del switch simultáneamente, en vez de tener varias tarjetas de red con cables directos al resto de nodos.

La tecnología SR-IOV funciona de una manera parecida, teniendo, por ejemplo, una sola tarjeta gráfica, podremos presentarle la misma tarjeta gráfica a diferentes máquinas virtuales, sin tener que dedicar la tarjeta gráfica a una sola máquina.

Imaginemos que tenemos una controladora de discos física y que en vez de virtualizar una controladora y un disco duro virtual que la máquina virtual reconocerá, podremos dedicar un canal de nuestra controladora para que la máquina virtual acceda directamente a los discos.

O una tarjeta de red, que en vez de virtualizar un switch y luego diferentes tarjetas de red virtuales, le pasamos un canal hacia nuestra tarjeta de red física para que acceda a ella directamente.

Y dado que estamos accediendo físicamente al hardware, utilizaríamos sus propios drivers, dejando de tener que utilizar hardware virtualizado intermedio, por lo que con toda probabilidad, esta tecnología acortará enormemente la típica diferencia del 10% de perdida de rendimiento en máquinas virtuales vs máquinas físicas, dejando a los equipos host con más recursos de CPU disponibles.

Respecto a el futuro de Proxmox, actualmente se encuentran implementando lo siguiente:

- Sistema de confianza para la creación de cluster de alta disponibilidad de dos nodos con un tercer quorum externo (por ejemplo, una raspberry Pi).
- Creación de una GUI para la migración en vivo de almacenamiento local.
- Integración de Cloudinit para hacer implementaciones masivas de plantillas de MV.
- Creación de HA desde el panel web.

Proxmox – Manos a la obra

Instalación de Proxmox

Tenemos dos maneras de instalar Proxmox:

1. Instalar un Debian stable y añadir los repositorios de Proxmox e instalar sus propios paquetes que sustituirán algunos de Debian.

2. Descargar la ISO de Proxmox y seguir los pasos de la instalación.

La primera opción fue la manera en la que se instaló en la máquina de escritorio, ya que requería de un entorno de escritorio además de paquetes extra como spotify, telegram, firefox, etc... así como una estructura de ficheros personalizada.

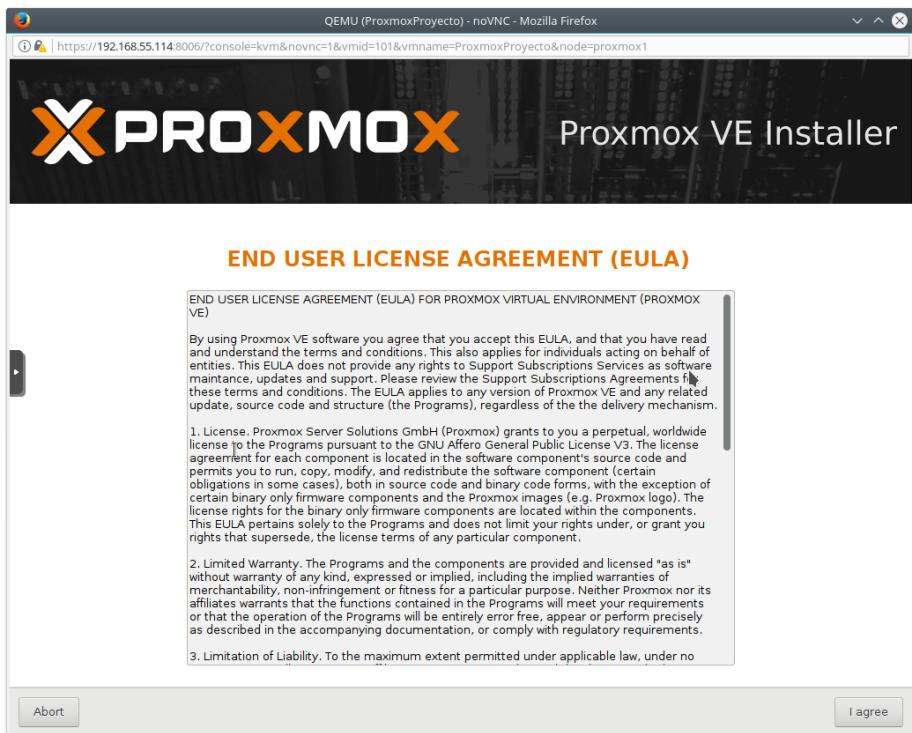
La segunda opción será la que usaremos y que será documentada a continuación.

(Las capturas corresponderán a la instalación de Proxmox en una maquina virtual, para que la visualización de las capturas sea 100% correcta, pero serán utilizadas las mismas opciones de instalación en los portátiles).

Arrancaríamos nuestro equipo desde la ISO quemada en un DVD o USB. Y seleccionamos la opción “Install Proxmox VE”:

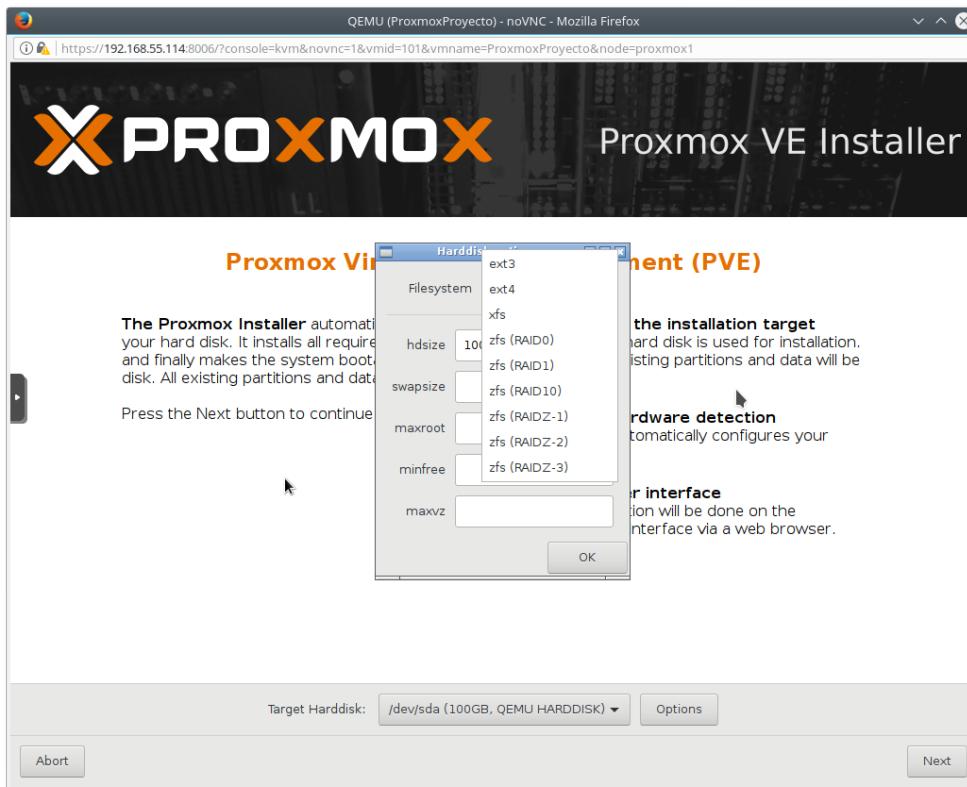


Aceptamos los términos y condiciones de uso:

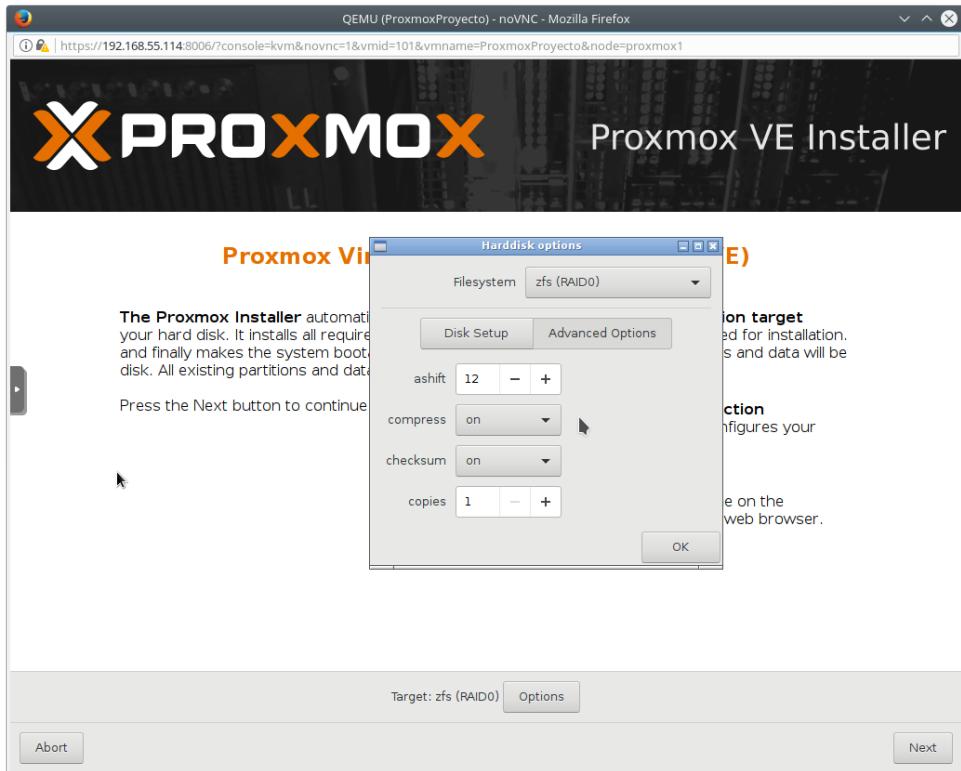


La siguiente ventana nos permite seleccionar los tipos de sistema de ficheros para la instalación. Si escogemos cualquiera de los métodos no replicados se montara un LVM con tres volúmenes (root, data, swap). Podremos controlar el espacio asignado a cada volumen con las opciones de maxroot, maxvz y swapsize, respectivamente.

Desde Proxmox 4.x se utiliza por defecto LVM-thin, para la parte de datos, por lo que la opción de minfree que se utilizaba antiguamente para almacenar los snapshots ya no hace falta, pero se sigue dejando para configuraciones personalizadas:



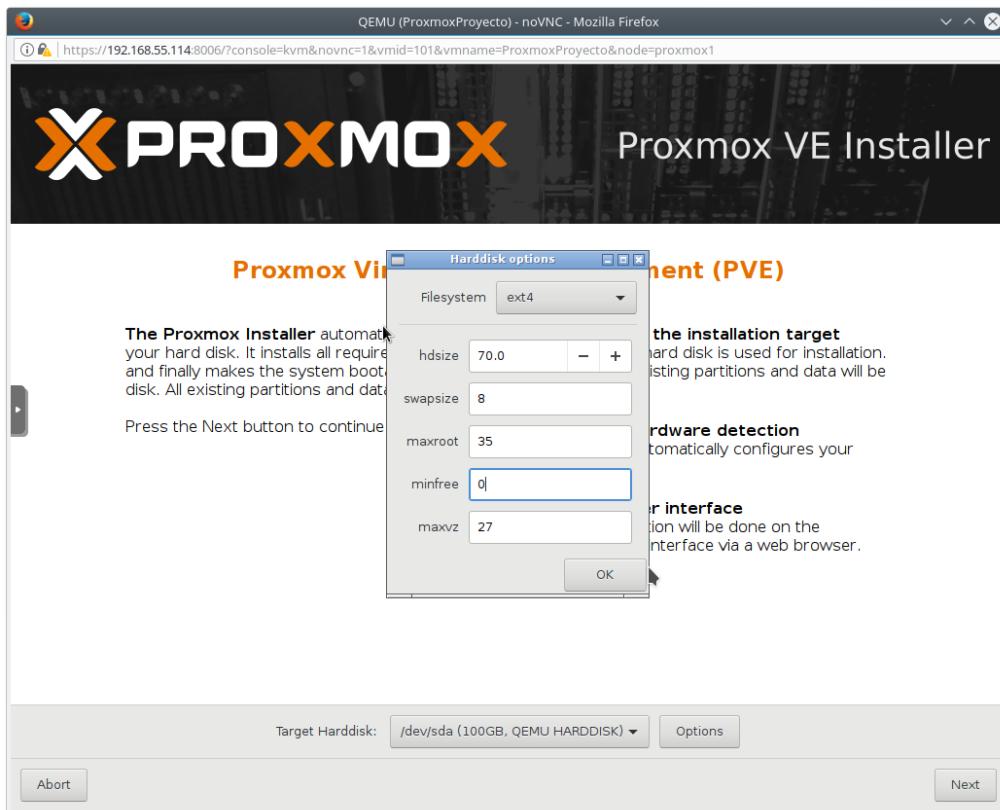
Las opciones de zfs nos permite configurar la compresión, comprobaciones checksum, tamaño de los sectores del disco duro... :



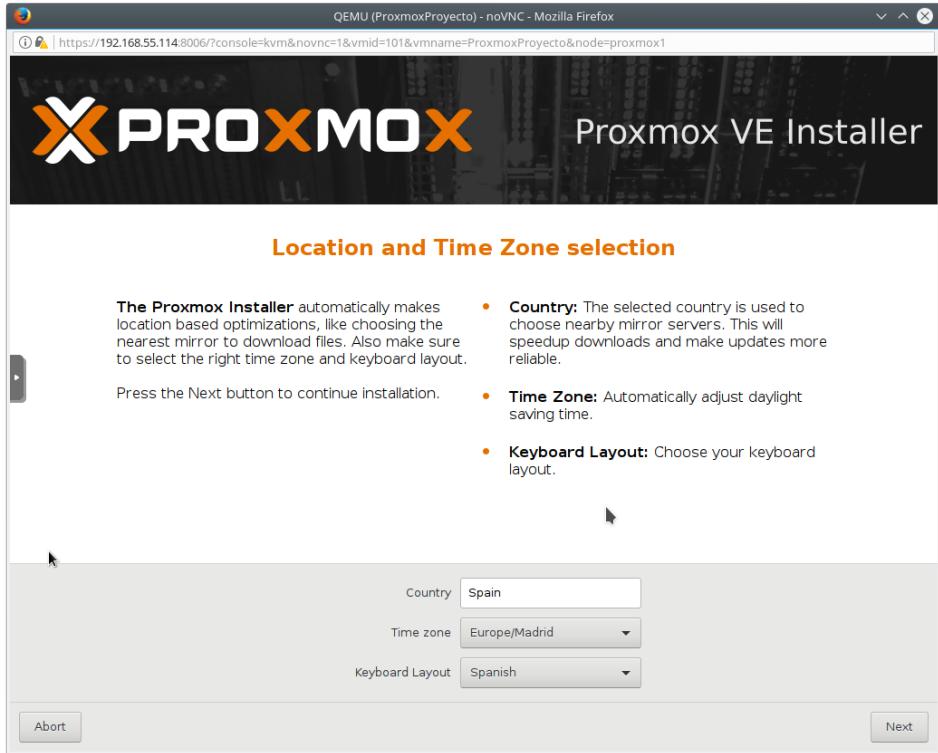
La instalación la vamos a hacer con un sistema de ficheros sin redundancia y con un sistema personalizado para las particiones, pero si dejamos los cuadros de textos por defecto se hará con la siguiente fórmula:

$$\text{Swapsize} = \text{hdsiz}/8 \quad \text{maxroot} = \text{hdsiz}/4 \quad \text{maxvz} = \text{hdsiz} - \text{rootsize} - \text{swapsize} - \text{minfree}$$

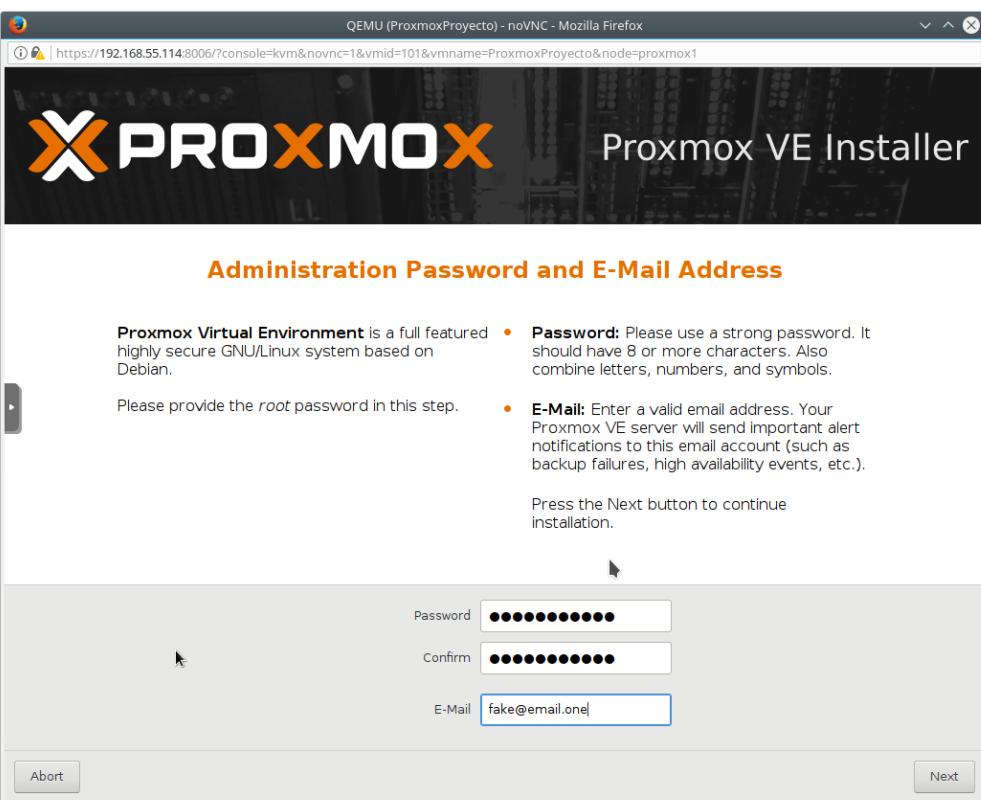
Nosotros lo dejaremos de la siguiente forma:



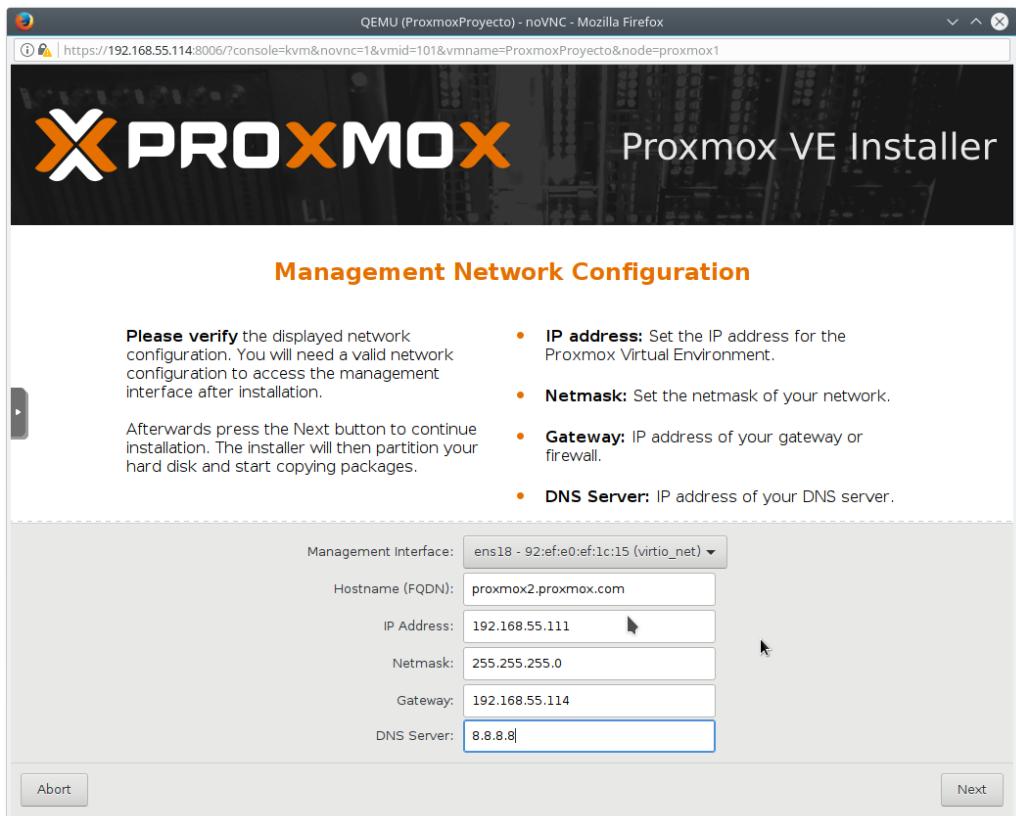
Seleccionamos localización y zona horaria:



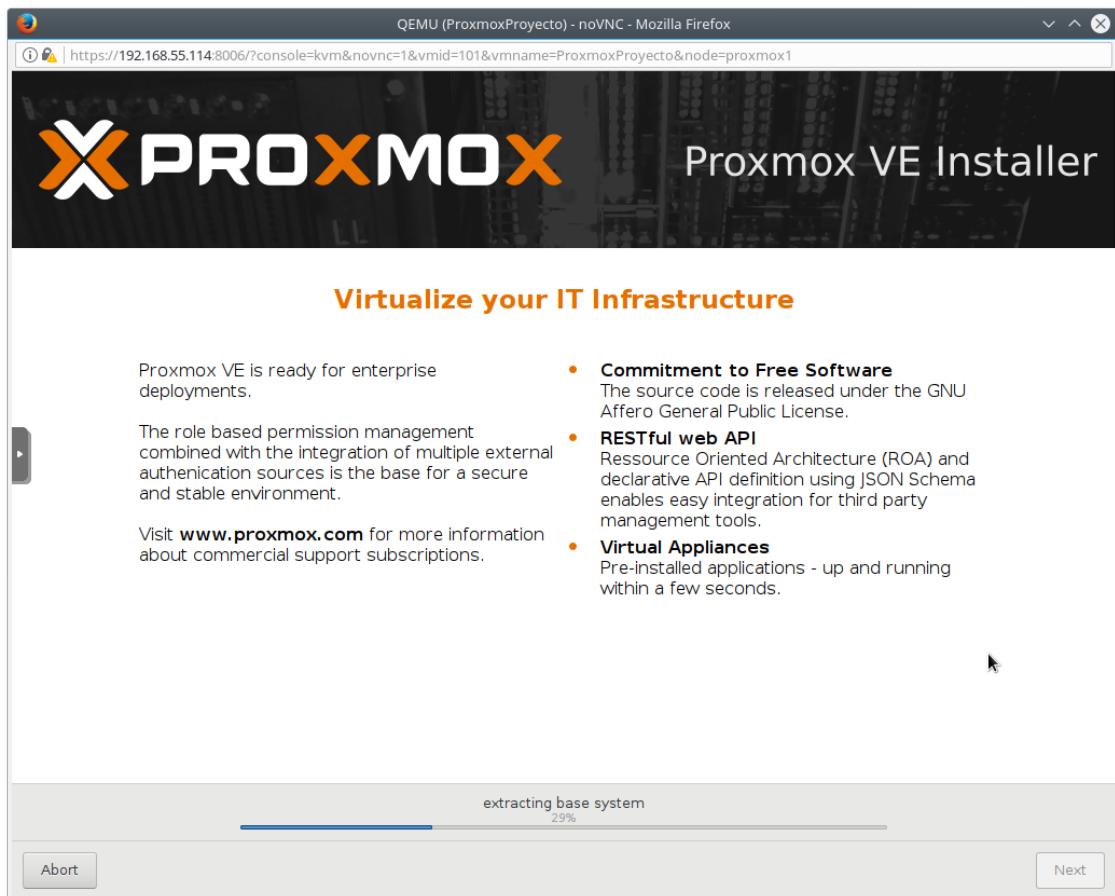
A continuación establecemos la contraseña y el email de administrador para que nos lleguen notificaciones de los servidores:



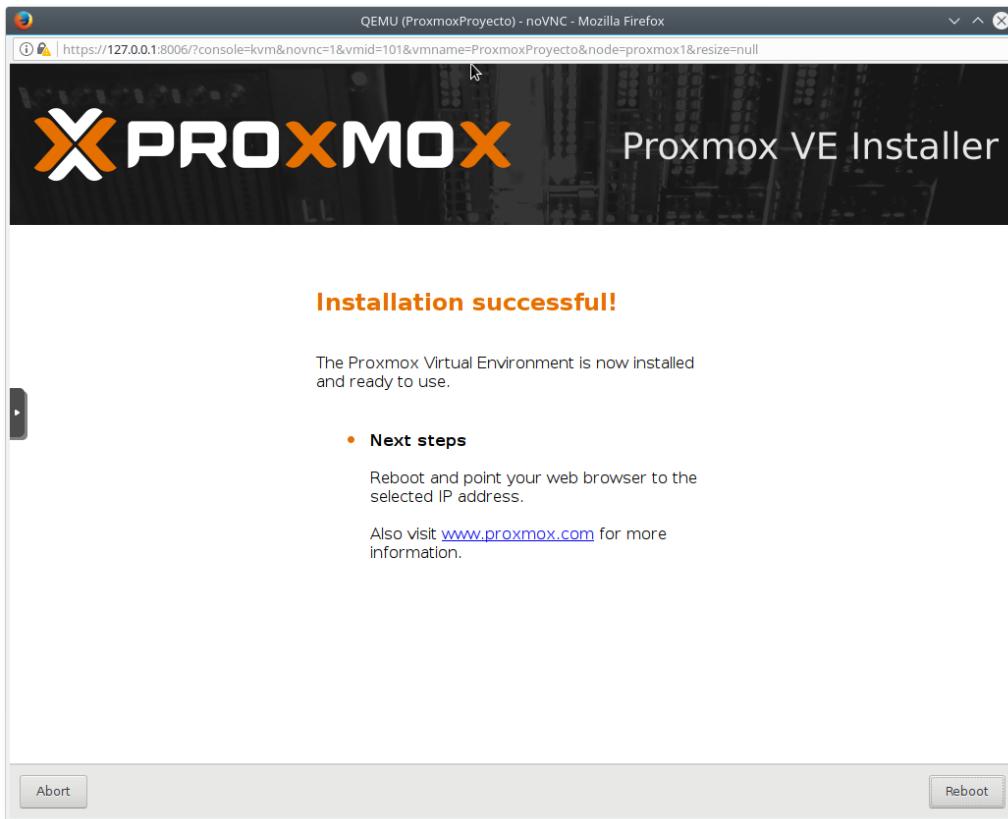
Estableceríamos un nombre para la maquina y su configuración IP:



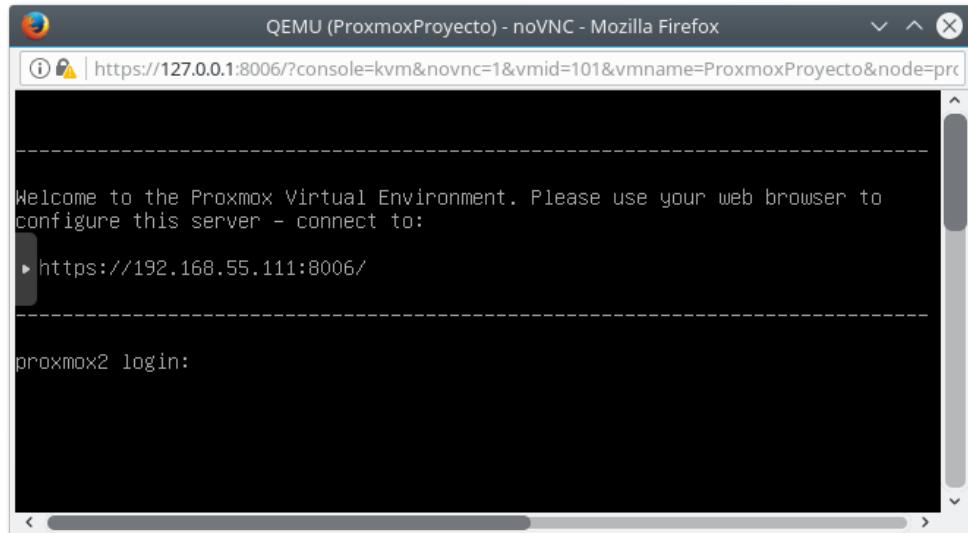
Tras aceptar la configuración empezará la instalación:



Una vez la instalación termine solo tendremos que pulsar reboot y el sistema estará listo para usar:



Podremos acceder por SSH al servidor, o bien accediendo por navegador a la GUI de Proxmox:



Llegados a este punto, realizaremos la misma configuración en los portátiles, llegando al punto donde los tres nodos son accesibles:

The screenshot displays three windows. On the left is a terminal window titled 'zeusinprox : bash — K' showing ping results between nodes. In the center and right are two Firefox browser windows, both titled 'proxmox1 - Proxmox Virtual Environment - Mozilla Firefox' and 'proxmox2 - Proxmox Virtual Environment - Mozilla Firefox'. Both browser windows show the 'Proxmox VE Login' page. The login form includes fields for 'User name', 'Password', 'Realm' (set to 'Linux PAM standard authentication'), and 'Language' (set to 'English'). There is also a 'Save User name:' checkbox and a 'Login' button.

```
zeusinprox : bash — K
Archivo Editar Ver Bookmarks Preferencias Ayuda
bt@proxmox1:/home/zeusinprox# ping 192.168.55.111
PING 192.168.55.111 (192.168.55.111) 56(84) bytes of data.
... (output truncated)
bt@proxmox1:/home/zeusinprox# ping 192.168.55.112
PING 192.168.55.112 (192.168.55.112) 56(84) bytes of data.
... (output truncated)
bt@proxmox1:/home/zeusinprox# ping 192.168.55.114
PING 192.168.55.114 (192.168.55.114) 56(84) bytes of data.
... (output truncated)
bt@proxmox1:/home/zeusinprox# 
```

Clusterización

El siguiente paso para la creación de la alta disponibilidad y CEPH es unir los 3 nodos bajo la misma configuración, para eso vamos a crear un cluster con los tres nodos.

Desde Proxmox1 vamos a ejecutar el comando “**pvecm create NombreDescriptivo**” para la creación del cluster y así mismo, este nodo se considerará el master.

Podemos observar como se crea una clave de autenticación para corosync.

Con el comando “**pvecm status**” podremos ver información relacionada con el cluster, como nodos activos, votos del quorum... :

```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox1:/home/zeusinprox# pvecm create laboratorio
Corosync Cluster Engine Authentication key generator.
Gathering 1024 bits for key from /dev/urandom.
Writing corosync key to /etc/corosync/authkey.
root@proxmox1:/home/zeusinprox# pvecm status
Quorum information
-----
Date: Sat Mar 31 15:59:40 2018
Quorum provider: corosync_votequorum
Nodes: 1
Node ID: 0x00000001
Ring ID: 1/4
Quorate: Yes

Votequorum information
-----
Expected votes: 1
Highest expected: 1
Total votes: 1
Quorum: 1
Flags: Quorate

Membership information
-----
  Nodeid   Votes Name
0x00000001       1 192.168.55.114 (local)
root@proxmox1:/home/zeusinprox#
```

Si ejecutásemos este comando desde un nodo que no pertenece a un cluster obtendríamos:

```
(192.168.55.112) — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox3:~# pvecm status
Corosync config '/etc/pve/corosync.conf' does not exist - is this node part of a cluster?
Cannot initialize CMAP service
root@proxmox3:~#
```

Ahora nos conectaríamos por SSH a los otros host para ejecutar el comando “`pvecm add IpMaestroCluster`” y agregarlos al grupo de servidores. (cluster)

Tras esperar un tiempo prudente, nos mostrará que el nodo ha sido agregado al cluster.

Proxmox3:

```
(() 192.168.55.112 — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox3:~# pvecm add 192.168.55.114
The authenticity of host '192.168.55.114 (192.168.55.114)' can't be established.
ECDSA key fingerprint is SHA256:bfnaAEKF4EKljIjbrb+HsGdetYLHjVDaMJXx+LE1mLLA.
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.55.114's password:
copy corosync auth key
stopping pve-cluster service
backup old database
waiting for quorum...OK
generating node certificates
merge known_hosts file
restart services
successfully added node 'proxmox3' to cluster.
root@proxmox3:~# )
```

Type	Description	Disk usage...	Memory us...	CPU usage
node	proxmox1	3.4 %	20.5 %	3.8% of 12...
node	proxmox3	8.5 %	9.4 %	0.5% of 4C...
qemu	100 (Win10.NVIDIA)			
qemu	101 (ProxmoxProyecto)			
storage	local (proxmox1)	41.6 %		
storage	local (proxmox3)	8.5 %		

Proxmox2:

```
(() 192.168.55.111 — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox2:~# pvecm add 192.168.55.114
The authenticity of host '192.168.55.114 (192.168.55.114)' can't be established.
ECDSA key fingerprint is SHA256:bfnaAEKF4EKljIjbrb+HsGdetYLHjVDaMJXx+LE1mLLA.
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.55.114's password:
copy corosync auth key
stopping pve-cluster service
backup old database
waiting for quorum...OK
generating node certificates
merge known_hosts file
restart services
root@proxmox2:~# )
```

Ahora, conectándonos a cualquiera de la direcciones ip de los equipos host veremos la misma configuración. Los cambios que hagamos se replicaran por igual en todos los host:

Type	Description	Disk usage...	Memory us...	CPU usage	Uptime
node	proxmox1	4.1 %	17.3 %	4.1% of 12...	01:19:5
node	proxmox2	11.4 %	19.7 %	0.3% of 4C...	00:27:0
node	proxmox3	11.4 %	9.5 %	0.4% of 4C...	21:39:3
qemu	100 (Win10.NVIDIA)				-
qemu	101 (ProxmoxProyecto)				-
qemu	102 (prueba)				-
storage	P1-ZFS (proxmox1)	2.7 %			-
storage	local (proxmox1)	74.9 %			-
storage	local (proxmox2)	11.4 %			-
storage	P3-Thin (proxmox3)	0.0 %			-
storage	local (proxmox2)	11.4 %			-

Con esta configuración ya estaría listo el uso de HA, solo necesitaríamos configurar el almacenamiento compartido:

Type	Status
quorum	OK
master	proxmox1 (active, Tue Apr 24 23:00:20 2018)
lrm	proxmox1 (idle, Tue Apr 24 23:00:25 2018)
lrm	proxmox2 (active, Tue Apr 24 23:00:24 2018)
lrm	proxmox3 (idle, Tue Apr 24 23:00:24 2018)

ID	State	Node	Max. Restart	Max. Reloc...	Group ↑	Description

Almacenamiento

De entre las tecnologías soportadas oficialmente, nosotros vamos a utilizar ZFS y CEPH debido a que son los que más características nos ofrecen:

Description	PVE type	Level	Shared	Snapshots	Stable
ZFS	zfspool	file	no	yes	yes
Directory	dir	file	no	No,Solo qcow2	yes
NFS	nfs	file	yes	No,Solo qcow2	yes
GlusterFS	glusterfs	file	yes	No,Solo qcow2	yes
LVM	lvm	block	no	no	yes
LVM-thin	lvmthin	block	no	yes	yes
iSCSI/kernel	iscsi	block	yes	no	yes
Ceph/RBD	rbd	block	yes	yes	yes
Sheepdog	sheepdog	block	yes	yes	beta

ZFS

ZFS es un sistema de ficheros de alto rendimiento, con capacidades de snapshots, replicamiento asíncrono, cache, niveles raid, compresión, deduplicación...

Nuestra idea es configurar un stripe de tres discos: dos disco duros de 1TB y una partición de 1TB en un disco de 2TB.

Primero debemos instalar el paquete para zfs:

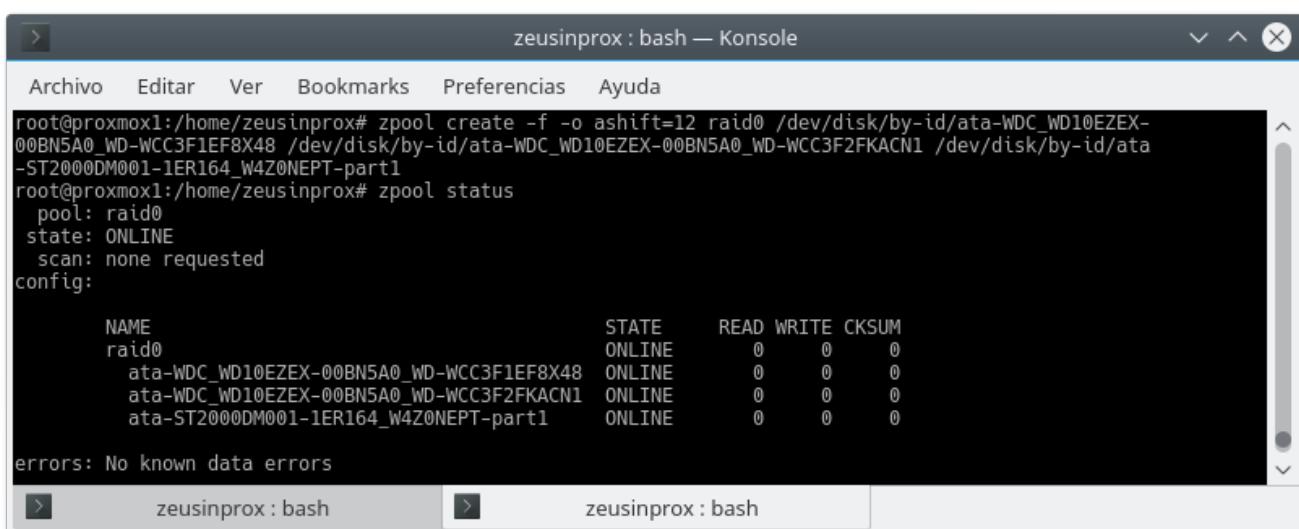
“aptitude install zfsutils-linux zfs-initramfs”

Una de las buenas prácticas es crear el pool de zfs con los id de los discos, no con la unidad de disco (sda,b,c...)

Por lo que el comando para crear el pool de zfs quedaría así:

```
zpool create -f -o ashift=12 raid0 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F1EF8X48 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F2FKACN1 /dev/disk/by-id/ata-ST2000DM001-1ER164_W4Z0NEPT-part1
```

Podemos comprobar la correcta creación con el comando “**zpool status**”.



```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox1:/home/zeusinprox# zpool create -f -o ashift=12 raid0 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F1EF8X48 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F2FKACN1 /dev/disk/by-id/ata-ST2000DM001-1ER164_W4Z0NEPT-part1
root@proxmox1:/home/zeusinprox# zpool status
  pool: raid0
    state: ONLINE
      scan: none requested
config:

  NAME        STATE     READ WRITE CKSUM
  raid0        ONLINE       0      0      0
    ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F1EF8X48  ONLINE       0      0      0
    ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F2FKACN1  ONLINE       0      0      0
    ata-ST2000DM001-1ER164_W4Z0NEPT-part1    ONLINE       0      0      0

errors: No known data errors
```

Si quisiésemos activar compresión, ejecutaríamos el siguiente comando :

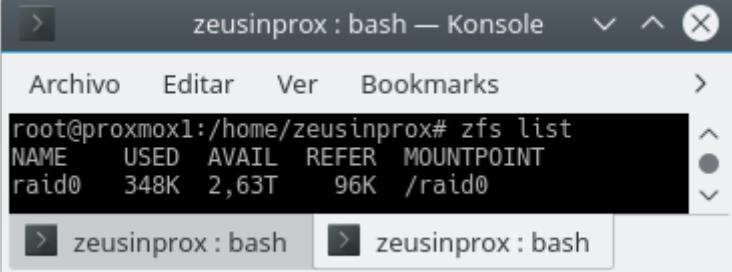
```
"zfs set compression=lz4 NombrePool"
```

Pudiendo ser los formatos de compresión: lzjb | gzip | gzip-[1-9] | zle | lz4

También estuvimos valorando la opción de añadir una parte del ssd para cache, pero se descartó dado que zfs no soporta trim a día de hoy. En caso de querer crearlo con cache, el comando de creación quedaría así:

```
zpool create -f -o ashift=12 raid0 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F1EF8X48 /dev/disk/by-id/ata-WDC_WD10EZEX-00BN5A0_WD-WCC3F2FKACN1 /dev/disk/by-id/ata-ST2000DM001-1ER164_W4Z0NEPT-part1 cache ata-Samsung_SSD_850_EVO_250GB_S2R6NX0H852646N-part3
```

Con el comando “**zfs list**” podemos ver el espacio libre y ocupado, punto de montaje, etc



```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks
root@proxmox1:/home/zeusinprox# zfs list
NAME    USED  AVAIL REFER MOUNTPOINT
raid0   348K  2,63T   96K  /raid0
```

Hay que tener en cuenta que por defecto ZFS se reserva la mitad de la RAM, aunque esto no implica que los vaya usar siempre al máximo.

La regla que debemos aplicar para dimensionar un ambiente de producción es 4GB RAM + 1GB por cada 1TB de almacenamiento.

Dado que nosotros no vamos a tener un grupo muy grande de discos vamos a reducir en primer momento este tamaño a 1GB e iremos realizando benchmarks para monitorizar el uso y encontrar nuestra capacidad idónea.

Para limitar el uso máximo de ram:

```
"echo "1073741824" > /sys/module/zfs/parameters/zfs_arc_max"
```

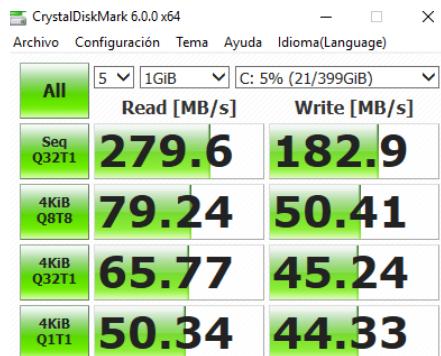
(Resultado en la ram del sistema al limitar la reserva de ram a 2 GB).



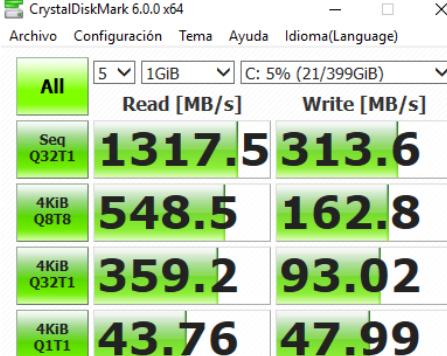
Para hacer este cambio permanente editaríamos algún fichero dentro de “**/etc/modprobe.d/**” y añadiríamos “**options zfs zfs_arc_max=2147483648**”

Los siguientes benchmarks han sido realizados limitando el uso de ram a la capacidad especificada:

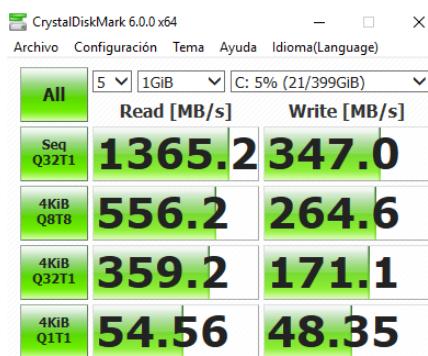
1GB(1073741824)



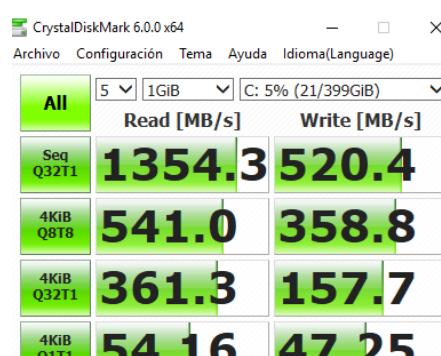
1.5GB (1610612736)



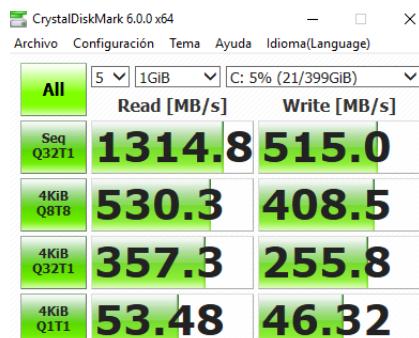
2GB (2147483648)



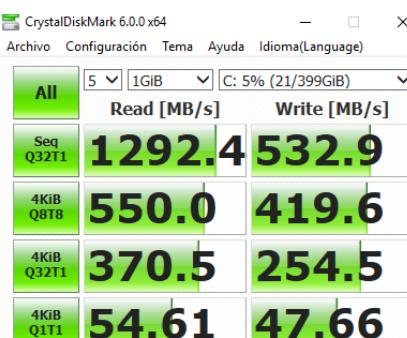
3GB (3221225472)



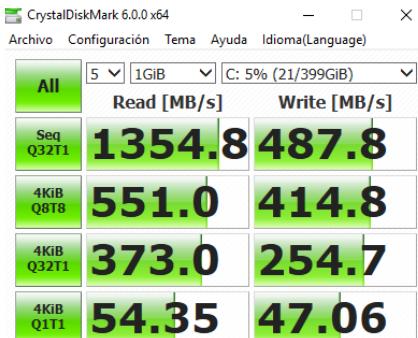
4 (4294967296)



5 (5368709120)



8GB (8589934592)



Podemos comprobar que a partir de 3GB para un volumen pequeño de datos la ganancia de rendimiento es mínima. Si tuviésemos un alto volumen de datos 3GB terminarían resultando inefficientes

Swap

Para añadir el swap, primero haríamos la reserva de espacio en la unidad zfs con el comando “`zfs create raid0/swap -V 5G -b 4K`”.

Para formatearlo como swap ejecutaríamos el comando “`mkswap -f /dev/raid0/swap`”.

Para modificar el swapiness utilizaremos el comando “`sysctl vm.swapiness=10`”

```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox1:/home/zeusinprox# zfs create raid0/swap -V 5G -b 4K
root@proxmox1:/home/zeusinprox# zfs list
NAME      USED  AVAIL REFER MOUNTPOINT
raid0    14,3G  2,62T  9,00G  /raid0
raid0/swap  5,31G  2,62T   56K  -
root@proxmox1:/home/zeusinprox# mkswap -f /dev/raid0/swap
Configurando espacio de intercambio versión 1, tamaño = 5 GiB (5368705024 bytes)
sin etiqueta, UUID=ab372481-01d2-4f82-8844-0b742c5c81e4
root@proxmox1:/home/zeusinprox# swapon /dev/raid0/swap
root@proxmox1:/home/zeusinprox# free
      total        used        free      shared  buff/cache   available
Mem:   16390296    11663520    2696528    293212    2030248    4983400
Swap:  5242876          0    5242876
root@proxmox1:/home/zeusinprox# cat /proc/sys/vm/swappiness
60
root@proxmox1:/home/zeusinprox# sysctl vm.swapiness=10
vm.swapiness = 10
root@proxmox1:/home/zeusinprox# cat /proc/sys/vm/swappiness
10
```

Para montarlo automáticamente al inicio del sistema, agregaríamos en el fichero “`/etc/fstab`” la siguiente linea: “`/dev/raid0/swap none swap defaults 0 0`”

```
GNU nano 2.7.4          Fichero: /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>  <type>  <options>      <dump>  <pass>
##Añadido swap zfs
/dev/raid0/swap none swap defaults 0 0
##-----
```

Para hacer permanente el swapiness editaremos el fichero: “`/etc/sysctl.d/99-sysctl.conf`” añadiendo el siguiente parámetro: “`vm.swapiness=10`”

```
GNU nano 2.7.4          Fichero: /etc/sysctl.d/99-sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#
## Añadido swapiness 10
vm.swapiness=10
```

Antes

```
zeusinprox@proxmox1:~$ free
      total        used        free      shared  buff/cache   available
Mem:   16390296    11663520    2696528    293212    2030248    4983400
Swap:  5242876          0    5242876
```

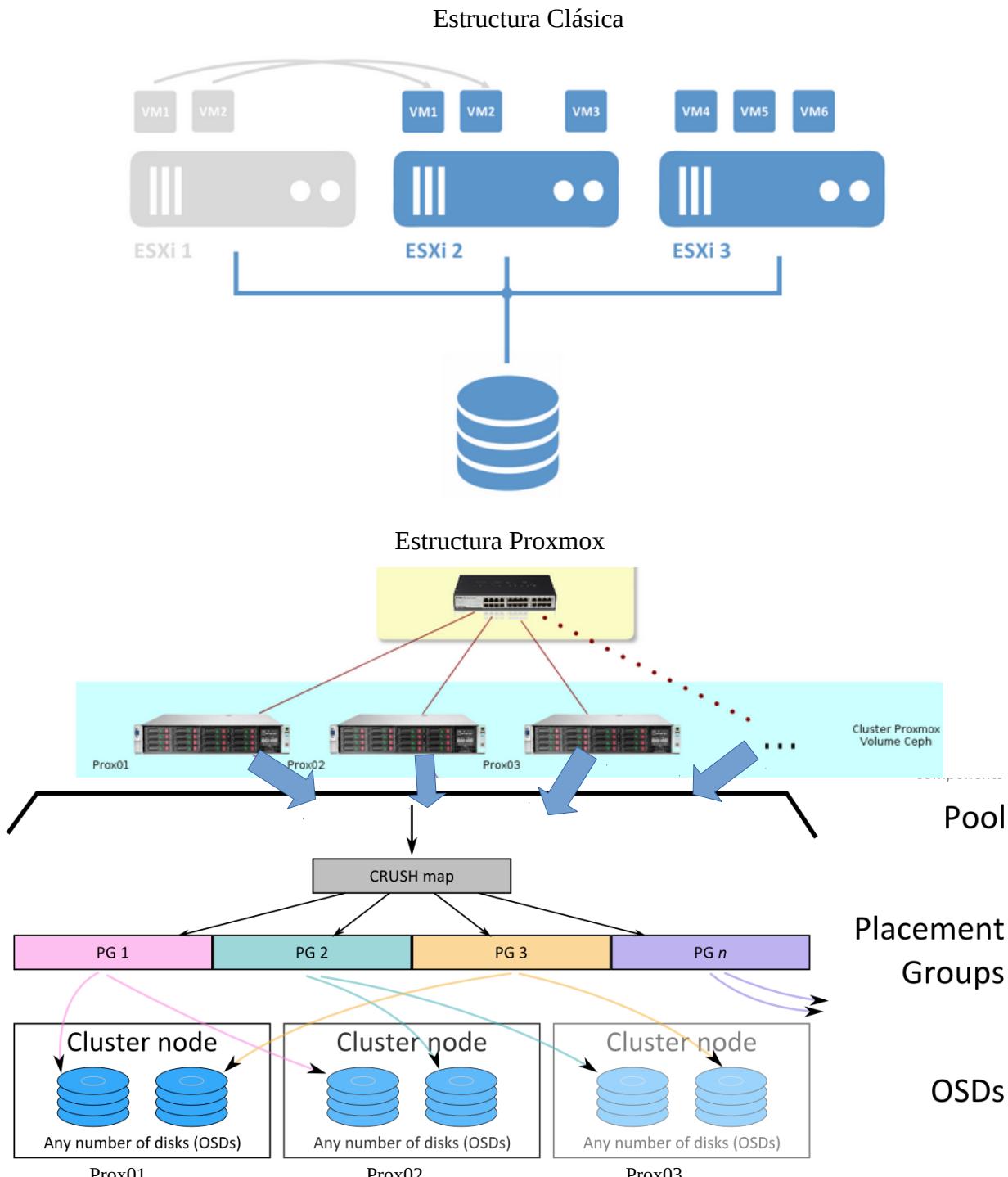
Despues

```
root@proxmox1:/home/zeusinprox# free
      total        used        free      shared  buff/cache   available
Mem:   16390288    5943420    7366132    0         0    5242876
Swap:  5242876          0    5242876
```

CEPH

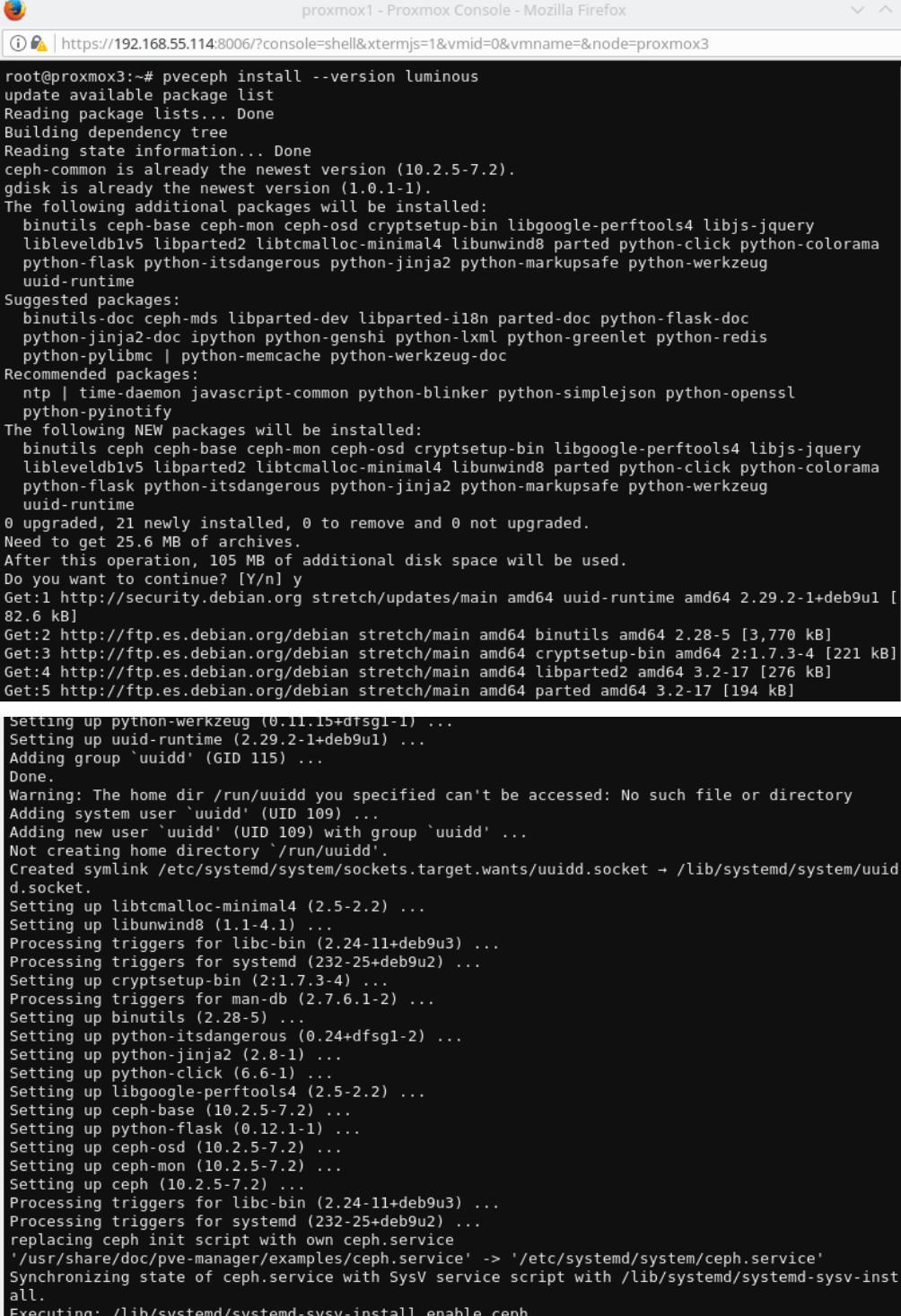
CEPH es un sistema de almacenamiento distribuido. Desde la versión 12 utiliza su propio sistema de ficheros (Bluefs). Anteriormente se delegaba por defecto en el sistema de ficheros XFS.

Una de las cosas que resuelve CEPH, es que en una típica instalación barata de Alta Disponibilidad, se montan dos o tres nodos para que se balanceen las máquinas virtuales y que si un nodo cae no se note en el servicio que prestamos, pero todo recae sobre una misma cabina de discos, que como caiga, no hay ni alta disponibilidad ni niveles 6 de raid que valgan.



En entornos de producción donde por presupuesto no se pueda adquirir un buen switch 10G o superior, se recomienda conectar los nodos con estructura de malla completa, evitando así la utilización de un switch.

Vamos a proceder con la instalación de CEPH en los tres nodos ejecutando en cada nodo el comando: “**pveceph install --version luminous**” que es un pequeño script de Proxmox para instalar los paquetes por apt-get, después añadiendo servicios propios de Proxmox sustituyendo a los de CEPH.



The screenshot shows a Mozilla Firefox browser window with the title "proxmox1 - Proxmox Console - Mozilla Firefox". The address bar shows the URL "https://192.168.55.114:8006/?console=shell&xtermjs=1&vmid=0&vmname=&node=proxmox3". The main content area displays the terminal output of the "pveceph install --version luminous" command:

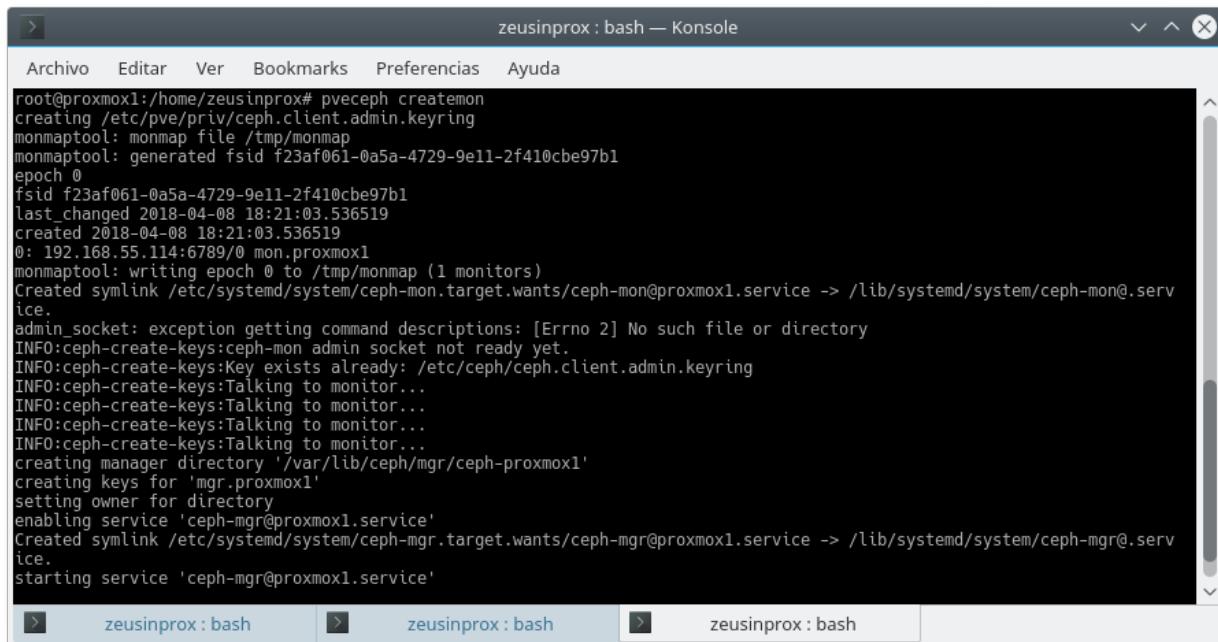
```
root@proxmox3:~# pveceph install --version luminous
update available package list
Reading package lists... Done
Building dependency tree
Reading state information... Done
ceph-common is already the newest version (10.2.5-7.2).
gdisk is already the newest version (1.0.1-1).
The following additional packages will be installed:
  binutils ceph-base ceph-mon ceph-osd cryptsetup-bin libgoogle-perf-tools4 libjs-jquery
  liblleveldb1v5 libparted2 libtcmalloc-minimal4 libunwind8 parted python-click python-colorama
  python-flask python-itsdangerous python-jinja2 python-markupsafe python-werkzeug
  uuid-runtime
Suggested packages:
  binutils-doc ceph-mds libparted-dev libparted-i18n parted-doc python-flask-doc
  python-jinja2-doc ipython python-genshi python-lxml python-greenlet python-redis
  python-pylibmc | python-memcache python-werkzeug-doc
Recommended packages:
  ntp | time-daemon javascript-common python-blinker python-simplejson python-openssl
  python-pynotify
The following NEW packages will be installed:
  binutils ceph ceph-base ceph-mon ceph-osd cryptsetup-bin libgoogle-perf-tools4 libjs-jquery
  liblleveldb1v5 libparted2 libtcmalloc-minimal4 libunwind8 parted python-click python-colorama
  python-flask python-itsdangerous python-jinja2 python-markupsafe python-werkzeug
  uuid-runtime
0 upgraded, 21 newly installed, 0 to remove and 0 not upgraded.
Need to get 25.6 MB of archives.
After this operation, 105 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.debian.org stretch/updates/main amd64 uuid-runtime amd64 2.29.2-1+deb9u1 [82.6 kB]
Get:2 http://ftp.es.debian.org/debian stretch/main amd64 binutils amd64 2.28-5 [3,770 kB]
Get:3 http://ftp.es.debian.org/debian stretch/main amd64 cryptsetup-bin amd64 2:1.7.3-4 [221 kB]
Get:4 http://ftp.es.debian.org/debian stretch/main amd64 libparted2 amd64 3.2-17 [276 kB]
Get:5 http://ftp.es.debian.org/debian stretch/main amd64 parted amd64 3.2-17 [194 kB]

Setting up python-werkzeug (0.11.15+dfsg1-1) ...
Setting up uuid-runtime (2.29.2-1+deb9u1) ...
Adding group `uuidd' (GID 115) ...
Done.
Warning: The home dir /run/uuidd you specified can't be accessed: No such file or directory
Adding system user `uuidd' (UID 109) ...
Adding new user `uuidd' (UID 109) with group `uuidd' ...
Not creating home directory `/run/uuidd'.
Created symlink /etc/systemd/system/sockets.target.wants/uuidd.socket → /lib/systemd/system/uuidd.socket.
Setting up libtcmalloc-minimal4 (2.5-2.2) ...
Setting up libunwind8 (1.1-4.1) ...
Processing triggers for libc-bin (2.24-11+deb9u3) ...
Processing triggers for systemd (232-25+deb9u2) ...
Setting up cryptsetup-bin (2:1.7.3-4) ...
Processing triggers for man-db (2.7.6.1-2) ...
Setting up binutils (2.28-5) ...
Setting up python-itsdangerous (0.24+dfsg1-2) ...
Setting up python-jinja2 (2.8-1) ...
Setting up python-click (6.6-1) ...
Setting up libgoogle-perf-tools4 (2.5-2.2) ...
Setting up ceph-base (10.2.5-7.2) ...
Setting up python-flask (0.12.1-1) ...
Setting up ceph-osd (10.2.5-7.2) ...
Setting up ceph-mon (10.2.5-7.2) ...
Setting up ceph (10.2.5-7.2) ...
Processing triggers for libc-bin (2.24-11+deb9u3) ...
Processing triggers for systemd (232-25+deb9u2) ...
replacing ceph init script with own ceph.service
'/usr/share/doc/pve-manager/examples/ceph.service' -> '/etc/systemd/system/ceph.service'
Synchronizing state of ceph.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ceph
```

Una vez instalado en los tres nodos, desde el nodo que utilicemos como principal ejecutaremos el comando: “**pveceph init --network 192.168.55.0/24**” que nos creará el archivo de configuración en la carpeta compartida por los tres nodos.

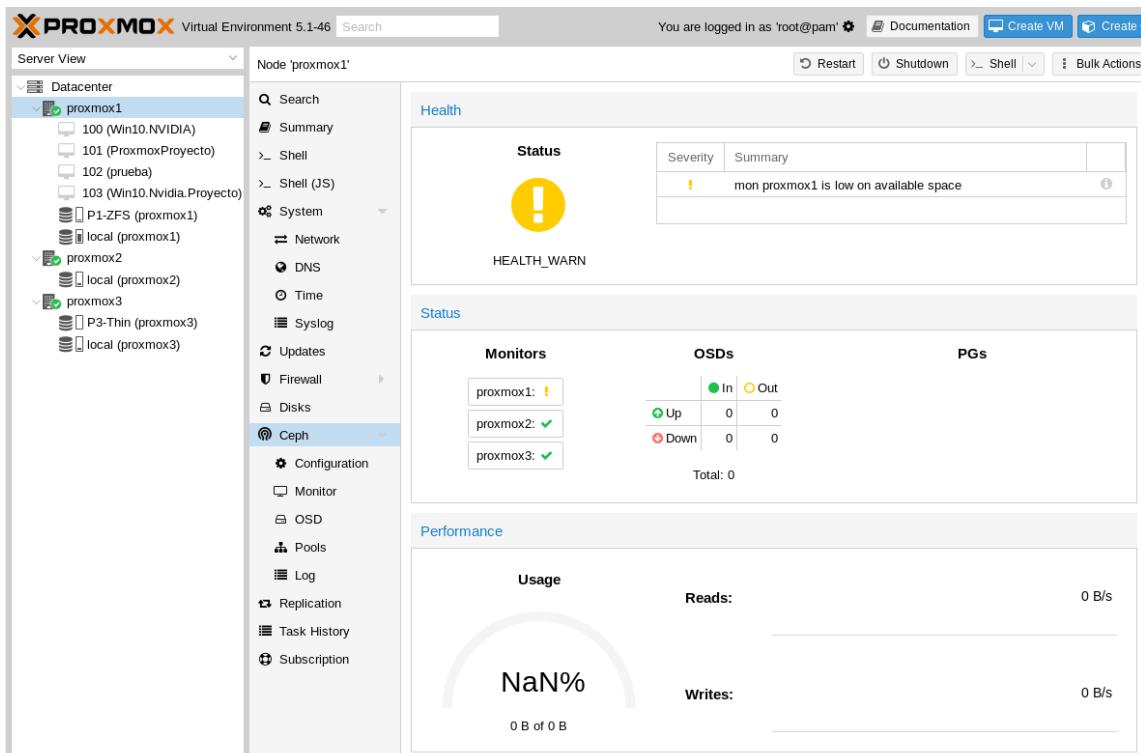
El siguiente paso sería instalar los monitores en cada uno de los nodos. Estos son unos procesos muy ligeros que se encargan de mantener la salud y la lógica del clúster funcionando, similar al quorum.

Para realizar la instalación usaremos el comando: “`pveceph createmon`” en cada nodo



```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxmox1:/home/zeusinprox# pveceph createmon
creating /etc/pve/priv/ceph.client.admin.keyring
monmaptool: monmap file /tmp/monmap
monmaptool: generated fsid f23af061-0a5a-4729-9e11-2f410cbe97b1
epoch 0
fsid f23af061-0a5a-4729-9e11-2f410cbe97b1
last_changed 2018-04-08 18:21:03.536519
created 2018-04-08 18:21:03.536519
0: 192.168.55.114:6789/0 mon.proxmox1
monmaptool: writing epoch 0 to /tmp/monmap (1 monitors)
Created symlink /etc/systemd/system/ceph-mon.target.wants/ceph-mon@proxmox1.service -> /lib/systemd/system/ceph-mon@.service.
admin_socket: exception getting command descriptions: [Errno 2] No such file or directory
INFO:ceph-create-keys:ceph-mon admin socket not ready yet.
INFO:ceph-create-keys:Key exists already: /etc/ceph/ceph.client.admin.keyring
INFO:ceph-create-keys:Talking to monitor...
INFO:ceph-create-keys:Talking to monitor...
INFO:ceph-create-keys:Talking to monitor...
INFO:ceph-create-keys:Talking to monitor...
INFO:ceph-create-keys:Creating manager directory '/var/lib/ceph/mgr/ceph-proxmox1'
INFO:ceph-create-keys:Creating keys for 'mgr.proxmox1'
INFO:ceph-create-keys:Setting owner for directory
INFO:ceph-create-keys:Enabling service 'ceph-mgr@proxmox1.service'
INFO:ceph-create-keys:Created symlink /etc/systemd/system/ceph-mgr.target.wants/ceph-mgr@proxmox1.service -> /lib/systemd/system/ceph-mgr@.service.
INFO:ceph-create-keys:Starting service 'ceph-mgr@proxmox1.service'
```

Si ahora fuésemos a la GUI y entrásemos en la opción para CEPH, podríamos ver estadísticas:



The screenshot shows the Proxmox VE 5.1-46 interface. On the left, the Server View sidebar is open, showing the Datacenter tree with nodes proxmox1, proxmox2, and proxmox3. The proxmox1 node is selected. In the main pane, under the 'Ceph' section of the sidebar, the 'Health' tab is active. The 'Status' section displays a yellow warning icon and a message: "mon proxmox1 is low on available space". Below this, the 'Monitors' section lists proxmox1 (warning), proxmox2 (up), and proxmox3 (up). The 'OSDs' section shows 0 up and 0 down OSDs. The 'PGs' section shows a total of 0. The 'Performance' section includes a gauge for 'Usage' (NaN%) and metrics for 'Reads' (0 B/s) and 'Writes' (0 B/s).

Ahora tendríamos que añadir el almacenamiento en cada uno de los tres nodos, para ello utilizaremos el comando:

“`pveceph createosd /dev/sdX`” indicando en cada nodo los discos que vayamos a usar

```
( ) 192.168.55.111 — Konsole

File Edit View Bookmarks Settings Help
110+0 records out
115343360 bytes (115 MB, 110 MiB) copied, 0.50558 s, 228 MB/s
/dev/loop0p1: 4 bytes were erased at offset 0x00000000 (xfs): 58 46 53 42
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.236148 s, 444 MB/s
Caution: invalid backup GPT header, but valid main header; regenerating
backup header from main header.

Warning! Main and backup partition tables differ! Use the 'c' and 'e' options
on the recovery & transformation menu to examine the two tables.

Warning! One or more CRCs don't match. You should repair the disk!

*****
Caution: Found protective or hybrid MBR and corrupt GPT. Using GPT, but disk
verification and recovery are STRONGLY recommended.
*****
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
GPT data structures destroyed! You may now partition the disk using fdisk or
other utilities.
Creating new GPT entries.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
Setting name!
partNum is 0
REALLY setting name!
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
Setting name!
partNum is 1
REALLY setting name!
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
meta-data=/dev/loop0p1      isize=2048    agcount=4, agsize=6400 blks
                           =          sectsz=512   attr=2, projid32bit=1
                           =          crc=1       finobt=1, sparse=0, rmapbt=0, reflink=0
data        =          bsize=4096   blocks=25600, imaxpct=25
                           =          sunit=0     swidth=0 blks
naming      =version 2      bsize=4096   ascii-ci=0 ftype=1
log         =internal log    bsize=4096   blocks=864, version=2
                           =          sectsz=512   sunit=0 blks, lazy-count=1
realtime    =none           extsz=4096   blocks=0, rtextents=0
Warning: The kernel is still using the old partition table.
```

o bien desde el entorno gráfico, seleccionando CEPH → OSD → CREATE OSD



Una vez hecho esto, veríamos en cada nodo el almacenaje añadido:

Name	Ty...	Class	OSD...	Status	weight	reweigh...	Used		Latency (ms)	
							%	Total	A...	C...
default	root									
proxmox3	host									
osd.2	osd	hdd	bluestore	up / in	0.019394	1	5.28	19.90 GiB	0	0
proxmox2	host									
osd.1	osd	hdd	bluestore	up / in	0.019394	1	5.28	19.90 GiB	0	0
proxmox1	host									
osd.0	osd	hdd	bluestore	up / in	0.019394	1	5.28	19.90 GiB	0	0

Ahora deberíamos crear un pool de discos con la configuración de defecto, marcando la opción “add storages”:

The screenshot shows the Proxmox VE web interface. On the left, there's a sidebar with various system management options like Shell, Network, DNS, Time, Syslog, Updates, Firewall, Disks, Ceph, Configuration, Monitor, OSD, Pools, Log, Replication, Task History, and Subscription. The 'Pools' option is currently selected.

In the main content area, there's a summary table for storage pools. Below it, a modal dialog titled 'Create: Ceph Pool' is open. The form fields are as follows:

- Name: ceph
- Size: 3
- Min. Size: 2
- Crush Rule: replicated_rule
- pg_num: 64
- Add Storages:

At the bottom of the dialog are 'Help' and 'Create' buttons.

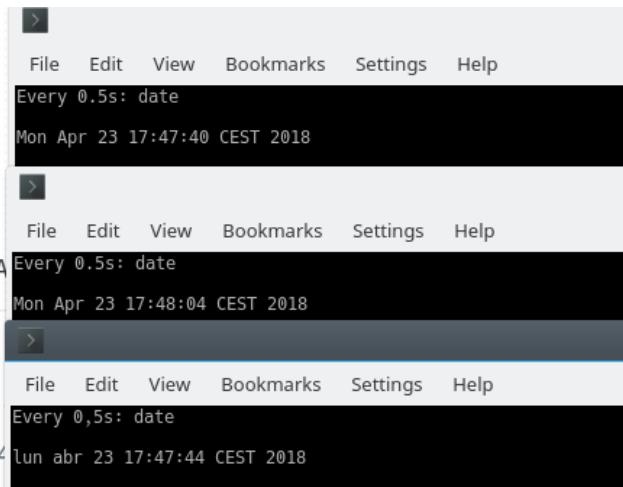
Aquí podemos ver la opción de almacenamiento en ceph creada:



Revisando la configuración me di cuenta de que estaba recibiendo unos errores “clock skew”:

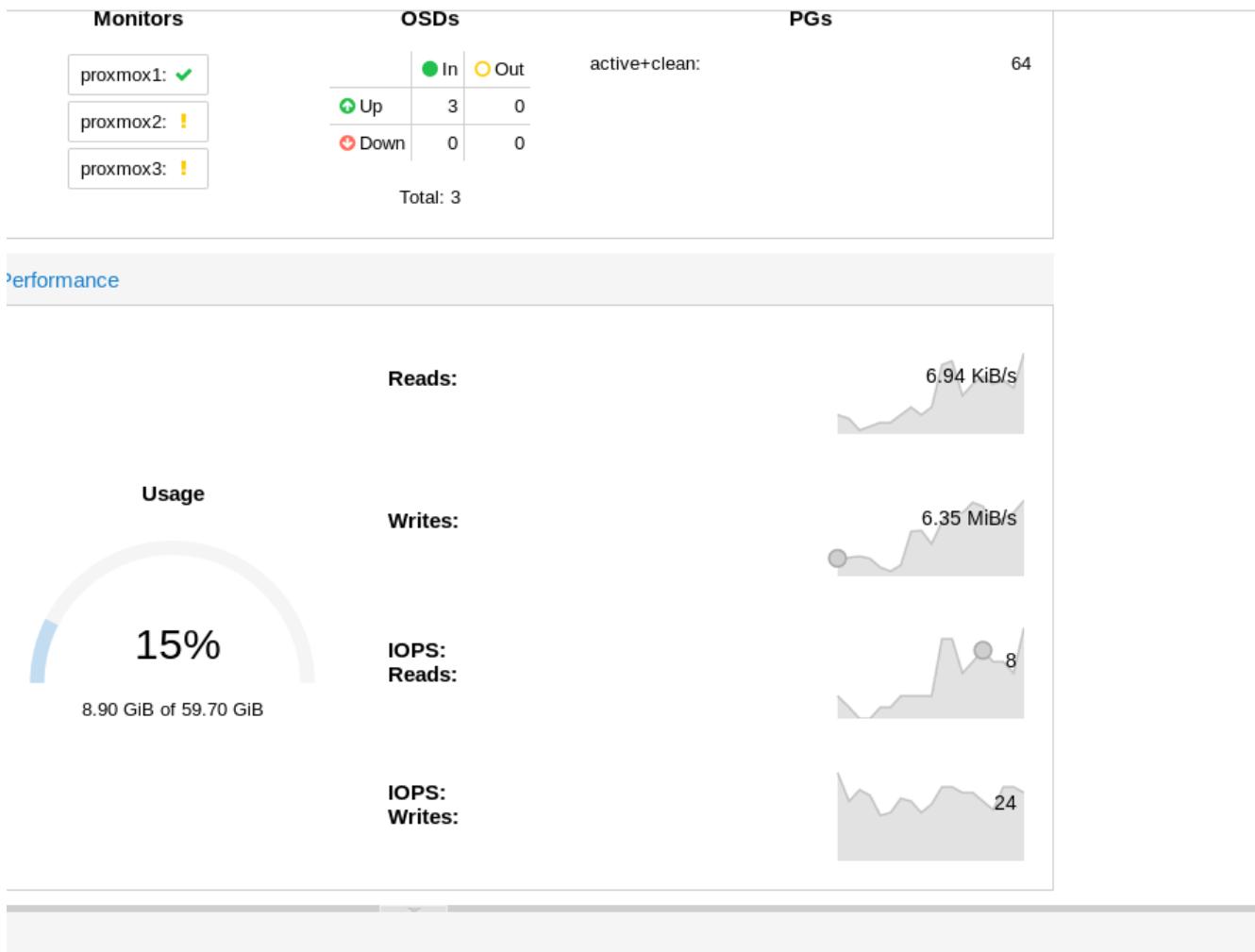
```
Detail X
clock skew detected on mon.proxmox3, mon.proxmox1
mon.proxmox3 addr 192.168.55.112:6789/0 clock skew 23.7719s > max 0.05s (latency 0.00138866s)
mon.proxmox1 addr 192.168.55.114:6789/0 clock skew 4.07695s > max 0.05s (latency 0.009787s)
```

Buscando en internet, di con que el problema para este error es que los equipos tienen distinto hora establecida... y efectivamente así era:

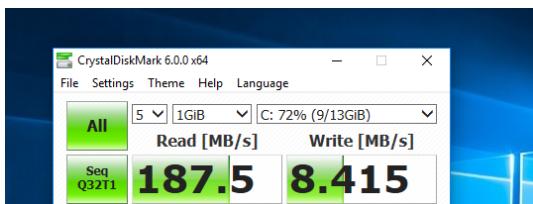


Tras dar acceso a internet a los nodos, el cliente NTP sincronizó contra el servidor que viene configurado por defecto y el mensaje de error desapareció.

Aquí podemos ver la ventana principal de estadísticas mientras se instala un sistema operativo en el almacenamiento compartido:



Con todo el sistema operativo ya instalado al hacer un test de velocidad del disco duro, obtenemos unos valores muy bajos, incluso para una red 1GB, por ello, volver a remarcar la necesidad de seguir las buenas prácticas propuestas por los desarrolladores.



Con el almacenamiento compartido creado ya podemos habilitar la HA y ser tolerantes a fallos de nodos.

Para añadirlos máquinas virtuales a la HA, haríamos lo siguiente: "Datacenter → HA → Resources → Add" y elegiríamos la máquina a añadir:

The screenshot shows the "Add: Resource: Container/Virtual Machine" dialog. It lists available virtual machines (VMs) in the Datacenter: Win10.NVIDIA (proxmox1), MacOs (proxmox1), and WinHA (proxmox2). The "Add" button is visible at the bottom right.

VM:	ID	Name	Node	Status	Type
Max. Restart:	100	Win10.NVIDIA	proxmox1	stopped	Virtual Machine
Max. Relocate:	101	MacOs	proxmox1	stopped	Virtual Machine
Comment:	103	WinHA	proxmox2	stopped	Virtual Machine

Configuración por defecto:

Add: Resource: Container/Virtual Machine

VM:	103	Group:	
Max. Restart:	1	Request State:	started
Max. Relocate:	1		
Comment:			
Help		Add	

Y ya estaría configurada para balancearse en caso de fallo de los equipos host:

Datacenter

Search Summary Options Storage Backup Replication Permissions Users Groups Pools Roles Authentication HA Groups Fencing	Status <table border="1"> <thead> <tr> <th>Type</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>quorum</td> <td>OK</td> </tr> <tr> <td>master</td> <td>proxmox1 (active, Tue Apr 24 23:02:40 2018)</td> </tr> <tr> <td>lrm</td> <td>proxmox1 (idle, Tue Apr 24 23:02:40 2018)</td> </tr> <tr> <td>lrm</td> <td>proxmox2 (active, Tue Apr 24 23:02:34 2018)</td> </tr> <tr> <td>lrm</td> <td>proxmox3 (idle, Tue Apr 24 23:02:39 2018)</td> </tr> </tbody> </table> Resources <table border="1"> <thead> <tr> <th>ID</th> <th>State</th> <th>Node</th> <th>Max. Restart</th> <th>Max. Reloc...</th> <th>Group ↑</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vm:103</td> <td>queued</td> <td>proxmox2</td> <td>1</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	Type	Status	quorum	OK	master	proxmox1 (active, Tue Apr 24 23:02:40 2018)	lrm	proxmox1 (idle, Tue Apr 24 23:02:40 2018)	lrm	proxmox2 (active, Tue Apr 24 23:02:34 2018)	lrm	proxmox3 (idle, Tue Apr 24 23:02:39 2018)	ID	State	Node	Max. Restart	Max. Reloc...	Group ↑	Description	vm:103	queued	proxmox2	1	1		
Type	Status																										
quorum	OK																										
master	proxmox1 (active, Tue Apr 24 23:02:40 2018)																										
lrm	proxmox1 (idle, Tue Apr 24 23:02:40 2018)																										
lrm	proxmox2 (active, Tue Apr 24 23:02:34 2018)																										
lrm	proxmox3 (idle, Tue Apr 24 23:02:39 2018)																										
ID	State	Node	Max. Restart	Max. Reloc...	Group ↑	Description																					
vm:103	queued	proxmox2	1	1																							

Para migrar un nodo que esta online pincharíamos en la máquina que queramos migrar y pulsaríamos en “Migrate”. Una vez hecho esto, nos saldría del desplegable para elegir el nodo al que queremos migrar:

PROXMOX Virtual Environment 5.1-40

Virtual Machine 103 (WinHA) on node 'proxmox2'

Start Shutdown

Máquina en Nodo "Proxmox2"

er View	Summary	Edit	Revert
Datacenter	Console		
proxmox1	Hardware		
proxmox2	Options		
103 (WinHA)	Task History	OS Type	Linux 4.X/3.X/2.6 Kernel
ceph_ct (proxmox2)	Monitor	Boot Order	Disk 'virtio0', CD-ROM, Network
ceph_vm (proxmox2)	Backup	Use tablet for pointer	Yes
local (proxmox2)	Replication	Hotplug	Disk, Network, USB
proxmox3	Snapshots	ACPI support	Yes
	Firewall	SCSI Controller	VirtIO SCSI
	Permissions	BIOS	Default (Seabios)
		KVM hardware virtualization	Yes
		Freeze CPU at startup	
		Use local time for RTC	
		RTC start date	
		SMBIOS settings (type1)	
		Qemu Agent	
		Protection	

Migrate VM 103

Target node: proxmox1

Mode: Online

[Help](#) [Migrate](#)

Una vez elegido, nos saldría el log del proceso de migración, donde se migrará la configuración de la máquina, memoria ram... :

Task viewer: VM 103 - Migrate

Output Status

Stop

```
task started by HA resource agent
2018-04-24 23:13:14 starting migration of VM 103 to node 'proxmox1' (192.168.55.114)
2018-04-24 23:13:15 copying disk images
2018-04-24 23:13:15 starting VM 103 on remote node 'proxmox1'
2018-04-24 23:13:17 start remote tunnel
2018-04-24 23:13:17 ssh tunnel ver 1
2018-04-24 23:13:17 starting online/live migration on unix:/run/qemu-server/103.migrate
2018-04-24 23:13:17 migrate_set_speed: 8589934592
2018-04-24 23:13:17 migrate_set_downtime: 0.1
2018-04-24 23:13:17 set migration_caps
2018-04-24 23:13:17 set cachesize: 272000614
2018-04-24 23:13:17 start migrate command to unix:/run/qemu-server/103.migrate
2018-04-24 23:13:19 migration status: active (transferred 105415649, remaining 2608128000), total 2737643520
2018-04-24 23:13:19 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
2018-04-24 23:13:20 migration status: active (transferred 227349375, remaining 2481012736), total 2737643520
2018-04-24 23:13:20 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
2018-04-24 23:13:21 migration status: active (transferred 345172135, remaining 2359300096), total 2737643520
2018-04-24 23:13:21 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
```

y al terminar veremos la migración en el otro nodo:

proxmox1

- 100 (Win10.NVIDIA)
- 101 (MacOs)
- 103 (WinHA)** (highlighted)
- P1-SSD (proxmox1)
- P1-ZFS (proxmox1)
- ceph_ct (proxmox1)
- ceph_vm (proxmox1)
- local (proxmox1)

proxmox2

- ceph_ct (proxmox2)
- ceph_vm (proxmox2)
- local (proxmox2)

proxmox3

Máquina migrada a Nodo "Proxmox1"

OS Type: WinHA
Boot Order: No
order=any
Linux 4.X/3.X/2.6 Kernel
Disk 'Virtio0', CD-ROM, Network
Use tablet for pointer: Yes

Task viewer: VM 103 - Migrate

Output Status

Stop

```
2018-04-24 23:13:35 migration status: active (transferred 1273710536, remaining 355679400), total 2737643520
2018-04-24 23:13:33 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
2018-04-24 23:13:34 migration status: active (transferred 1345496044, remaining 214450176), total 2737643520
2018-04-24 23:13:34 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
2018-04-24 23:13:35 migration status: active (transferred 1463865194, remaining 90075136), total 2737643520
2018-04-24 23:13:35 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 0 overflow 0
2018-04-24 23:13:36 migration status: active (transferred 1569602316, remaining 49229824), total 2737643520
2018-04-24 23:13:36 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 7466 overflow 0
2018-04-24 23:13:36 migration status: active (transferred 1582170049, remaining 34242560), total 2737643520
2018-04-24 23:13:36 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 10528 overflow 0
2018-04-24 23:13:36 migration status: active (transferred 1593965141, remaining 22392832), total 2737643520
2018-04-24 23:13:36 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 13401 overflow 0
2018-04-24 23:13:36 migration status: active (transferred 1607151417, remaining 13942784), total 2737643520
2018-04-24 23:13:36 migration xbzrle cachesize: 268435456 transferred 0 pages 0 cachemiss 16614 overflow 0
2018-04-24 23:13:36 migration speed: 136.53 MB/s - downtime 174 ms
2018-04-24 23:13:36 migration status: completed
2018-04-24 23:13:40 migration finished successfully (duration 00:00:29)
TASK OK
```

GPU-Passthrough

Introducción

Se conoce como “PCI Passthrough” a la técnica empleada para dedicar dispositivos PCIe a máquinas virtuales.

En el caso de las tarjetas gráficas, Intel va un paso por delante y desde la versión 2.12 de QEMU ofrece soporte estable para la virtualización de hasta 7 gráficas virtuales por gráfica física.

Limitaciones PCIe / Hardware

Una de las limitaciones a la que nos enfrentamos con la asignación en el caso de las tarjetas gráficas son los impedimentos a nivel de software que nvidia hace en sus drivers, estableciendo una serie de parámetros para comprobar si donde está siendo instalada la tarjeta gráfica es una máquina virtual o no y en caso de serlo, inutiliza los drivers para que no puedan ser usados (Código 43).

En primer lugar, para sobreponer esta limitación se desactivaron las llamadas "HyperV Enlightments" que mejoraban el funcionamiento de máquinas virtuales Windows en KVM, más tarde, se descubrió que deshabilitando la exposición de los metadatos de máquina virtual con la opción "`kvm=off`" y estableciendo el parámetro `hv_vendor_id="Inventado"` se pueden pasar las comprobaciones hechas por nvidia a día de hoy. Posiblemente esto deje de funcionar en un futuro.

La excusa de nvidia para limitar esta funcionalidad es que ya ofrecen productos orientados para la virtualización con sus gráficas empresariales GRID y que sus gráficas de escritorio no están preparadas para esta función a pesar de ser totalmente capaces e incluso, dar menos problemas que las gráficas AMD.

En el caso de AMD no nos encontramos con impedimentos, pero sí nos encontramos con falta de soporte oficial. Por eso si das con una tarjeta gráfica que funcione bien, no vas a tener ningún problema, pero si encuentras una gráfica conflictiva (GPU's Vega hasta kernel 4.16), tienes que esperar un largo tiempo hasta que la comunidad encuentre una solución o amd destine recursos para solucionarlo.

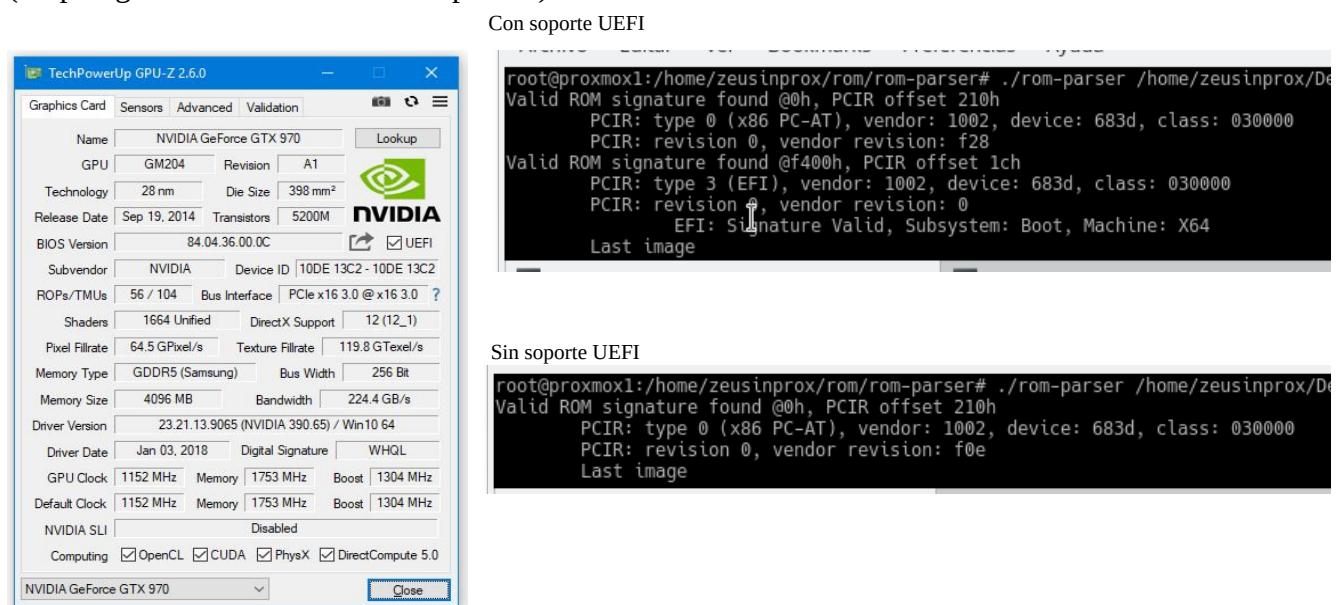
Dado que esta no es una funcionalidad que se requiera habitualmente la mayoría de placas bases para el consumidor básico vienen con esta funcionalidad incompleta, con implementaciones a medias... siendo una lotería que funcione o no, pudiendo dejar de funcionar en versiones nuevas de bios.

En cuanto a cpus, Intel lleva la bandera en cuanto al correcto funcionamiento.

AMD, desde principios del año 2018 se encuentra con un funcionamiento estable, pero gracias al esfuerzo de la comunidad que se ha dedicado a arreglar fallos que hacían inviable usar AMD como plataforma para estos menesteres.

Identificación de compatibilidad

Ahora deberíamos asegurarnos de que las tarjetas gráficas que estemos utilizando soporte el modo UEFI. Probablemente las gráficas de los últimos 4-5 años lo soporten. Para comprobarlo podemos utilizar el software GPU-Z en Windows o el script rom-parser en Linux:
(<https://github.com/awilliam/rom-parser>)



Para ciertos modelos de gráficas, como por ejemplo la serie 7000 de AMD, ciertos fabricantes ofrecieron soporte UEFI en sus bios y otros no... (Ejemplo de la capturas de las dos terminales ejecutando el script).

Si obtenemos “**EFI: Signature Valid**” es que nuestra gráfica es compatible con UEFI.

También puede darse el caso de que simplemente puede esta deshabilitada la comprobación que indica que la tarjeta soporta UEFI, a pesar de que la gráfica sí lo soporta, por lo que podemos buscar una bios de otro fabricante para nuestro modelo que sí soporte UEFI, y probar a configurarla temporalmente con esa bios.

Para poder integrar la tarjeta gráfica a la máquina virtual primero tenemos que saber en que puerto pcie esta insertada, para ello nos ayudaremos del comando “**lspci**”:

“**lspci | egrep "VGA compatible|Audio device"**”

En mi caso podemos ver en el puerto 02:00 podemos ver dos entradas .00 para la gráfica y 01 para la tarjeta de sonido integrada (hdmi).

En el puerto 03:00 tenemos la tarjeta nvidia del sistema y subpuerto para su tarjeta de sonido(hdmi).

En el puerto 05:00 la tarjeta nvidia potente y su tarjeta de sonido (hdmi)

```
root@proxmox1:/home/zeusinprox# lspci | egrep "VGA compatible|Audio device"
00:1b.0 Audio device: Intel Corporation C610/X99 series chipset HD Audio Controller (rev 05)
02:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Cape Verde XT [Radeon HD 7770/8760 / R7 250X]
02:00.1 Audio device: Advanced Micro Devices, Inc. [AMD/ATI] Cape Verde/Pitcairn HDMI Audio [Radeon HD 7700/7800 Series]
03:00.0 VGA compatible controller: NVIDIA Corporation GF119 [GeForce GT 610] (rev a1)
03:00.1 Audio device: NVIDIA Corporation GF119 HDMI Audio Controller (rev a1)
05:00.0 VGA compatible controller: NVIDIA Corporation GM204 [GeForce GTX 970] (rev a1)
05:00.1 Audio device: NVIDIA Corporation GM204 High Definition Audio Controller (rev a1)
```

Ahora ejecutaríamos el comando “`find /sys/kernel/iommu groups/ -type l`” que nos mostrará los grupos IOMMU.

En nuestro caso podemos ver que las gráficas que vamos a usar, se encuentran cada una un diferente grupo, y que ningún dispositivo ocupa estos grupos.

Por ejemplo, podría darse el caso en el que la controladora SATA que tiene el disco con el sistema operativo del host esté en el mismo grupo que la gráfica, lo que nos imposibilitaría la configuración, ya que todos los dispositivos que estén en un grupo IOMMU se desconectaran del sistema cuando se dediquen a una máquina virtual.

Si por ejemplo tuviésemos un controlador USB en el mismo grupo que la tarjeta gráfica, debemos también aplicar el driver para virtualizar en ese controlador si no queremos que se quede sin uso.

Configuración

Una vez comprobamos que nuestros periféricos son compatibles, empezaríamos a configurar la máquina.

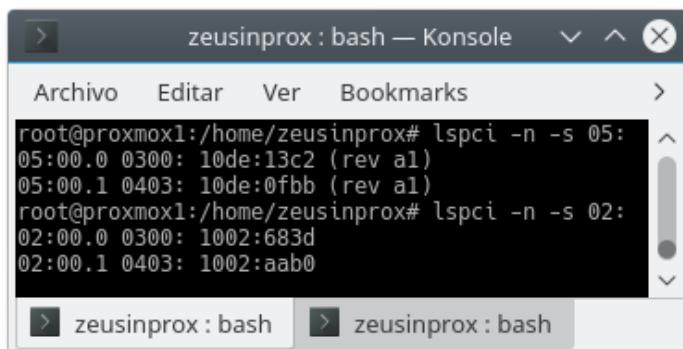
Debemos editar el fichero “`/etc/default/grub`”. En la linea que por defecto es:

`GRUB_CMDLINE_LINUX_DEFAULT="quiet"`

y la editaríamos, añadiendo s lo siguiente: “`iommu=pt intel_iommu=on rd.driver.pre=vfio-pci`”, quedando tal que así:

`GRUB_CMDLINE_LINUX_DEFAULT="quiet iommu=pt intel_iommu=on rd.driver.pre=vfio-pci"`

Ahora ejecutaríamos el comando “`lspci -n -s 05:`” (en 05 escribimos el puerto de la grafica), que nos devolverá los ids de los dispositivos:



```
zeusinprox : bash — Konsole
Archivo Editar Ver Bookmarks
root@proxmox1:/home/zeusinprox# lspci -n -s 05:
05:00.0 0300: 10de:13c2 (rev a1)
05:00.1 0403: 10de:0fbb (rev a1)
root@proxmox1:/home/zeusinprox# lspci -n -s 02:
02:00.0 0300: 1002:683d
02:00.1 0403: 1002:aab0
```

Una vez obtenidos nuestros ids, editaríamos el fichero “`/etc/modprobe.d/vfio.conf`” y añadiríamos la siguiente linea:

“`options vfio-pci ids=10de:13c2,10de:0fbb,1002:683d,1002:aab0`”

Una vez hecho esto, realizaríamos un “`update grub`” y reiniciaríamos la máquina.

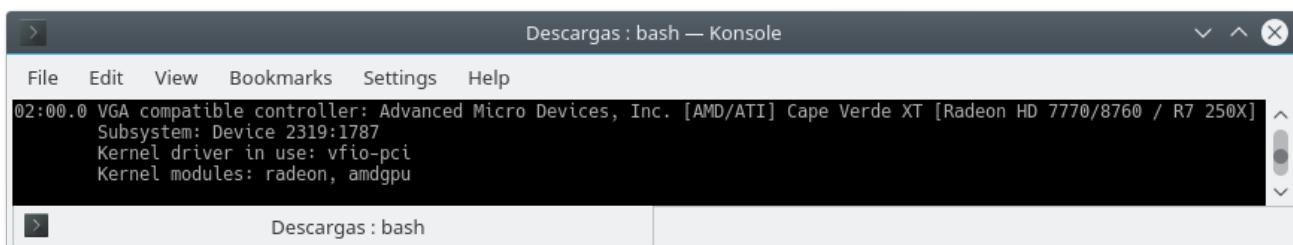
Una vez reiniciado, si ejecutamos el comando:

“`dmesg | grep -e DMAR -e IOMMU`”

Si la terminal no nos devuelve ningún tipo de información como:

```
[0.000000] DMAR: IOMMU enabled
[0.038406] DMAR-IR: Enabled IRQ remapping in xapic mode
[8.761127] DMAR: Hardware identity mapping for device 0000:05:00.0
[8.761137] DMAR: Intel(R) Virtualization Technology for Directed I/O
```

O si ejecutamos el comando “`lspci -k -s IDPCI:`” y en las opciones de la vga no nos sale “`Kernel driver in use: vfio-pci`” es que el sistema no está bien configurado y quizás deberíamos revisar la configuración de la bios de nuestro equipo.



```
Descargas : bash — Konsole
File Edit View Bookmarks Settings Help
02:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Cape Verde XT [Radeon HD 7770/8760 / R7 250X]
    Subsystem: Device 2319:1787
    Kernel driver in use: vfio-pci
    Kernel modules: radeon, amdgpu
```

Máquina Virtual: Nvidia & Windows

Vamos a configurar la tarjeta grafica nvidia 970 para una maquina virtual.

Crearíamos una máquina virtual en Proxmox y realizaríamos una instalación de Windows 10 con todos los periféricos por defecto y añadiendo soporte UEFI.

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Node: proxmox1 Resource Pool:

VM ID: 103

Name: Win10.Nvidia.Proyecto

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Use CD/DVD disc image file (iso) Guest OS:

Storage: local Type: Microsoft Windows

ISO image: 14393.0.160715-1616.RS1_RE Version: 10/2016

Use physical CD/DVD Drive

Do not use any media

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Bus/Device: SATA Cache: Default (No cache)

Storage: P1-ZFS No backup:

Disk size (GiB): 32 Discard:

Format: Raw disk image (raw) IO thread:

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Sockets: 1 Type: host

Cores: 4 Total cores: 4

Enable NUMA:

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Use fixed size memory

Memory (MiB): 2048 Ballooning:

Automatically allocate memory within this range

Maximum memory (MiB): 1024

Minimum memory (MiB): 512

Shares: Default (1000)

Create: Virtual Machine

General OS Hard Disk CPU Memory Network Confirm

Bridged mode Model: Intel E1000

VLAN Tag: no VLAN MAC address: auto

Bridge: vmbr666 Rate limit (MB/s): unlimited

Firewall: Multiqueue:

NAT mode Disconnect:

No network device

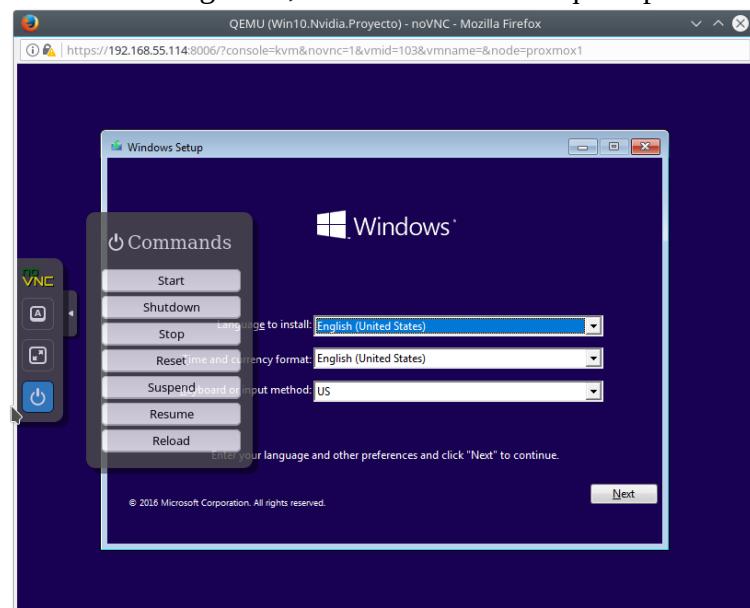
Para añadir soporte EFI: (Options → BIOS → OVMF (UEFI)):

The screenshot shows the Proxmox VE interface. On the left, the 'Server View' sidebar lists Datacenter, proxmox1, proxmox2, and proxmox3. Under proxmox1, there are several virtual machines: 100 (Win10.NVIDIA), 101 (ProxmoxProyecto), 102 (prueba), and 103 (Win10.Nvidia.Proyecto). The 'Edit' button for VM 103 is highlighted. The main window displays the configuration for VM 103, specifically the 'Edit' screen for the BIOS tab. A modal window titled 'Edit: BIOS' is open, showing the 'BIOS:' dropdown set to 'OVMF (UEFI)'. Below the dropdown, a message says: 'You need to add a Default (SeaBIOS) settings. See the SeaBIOS OVMF (UEFI)'. There are 'OK' and 'Reset' buttons at the bottom of the modal.

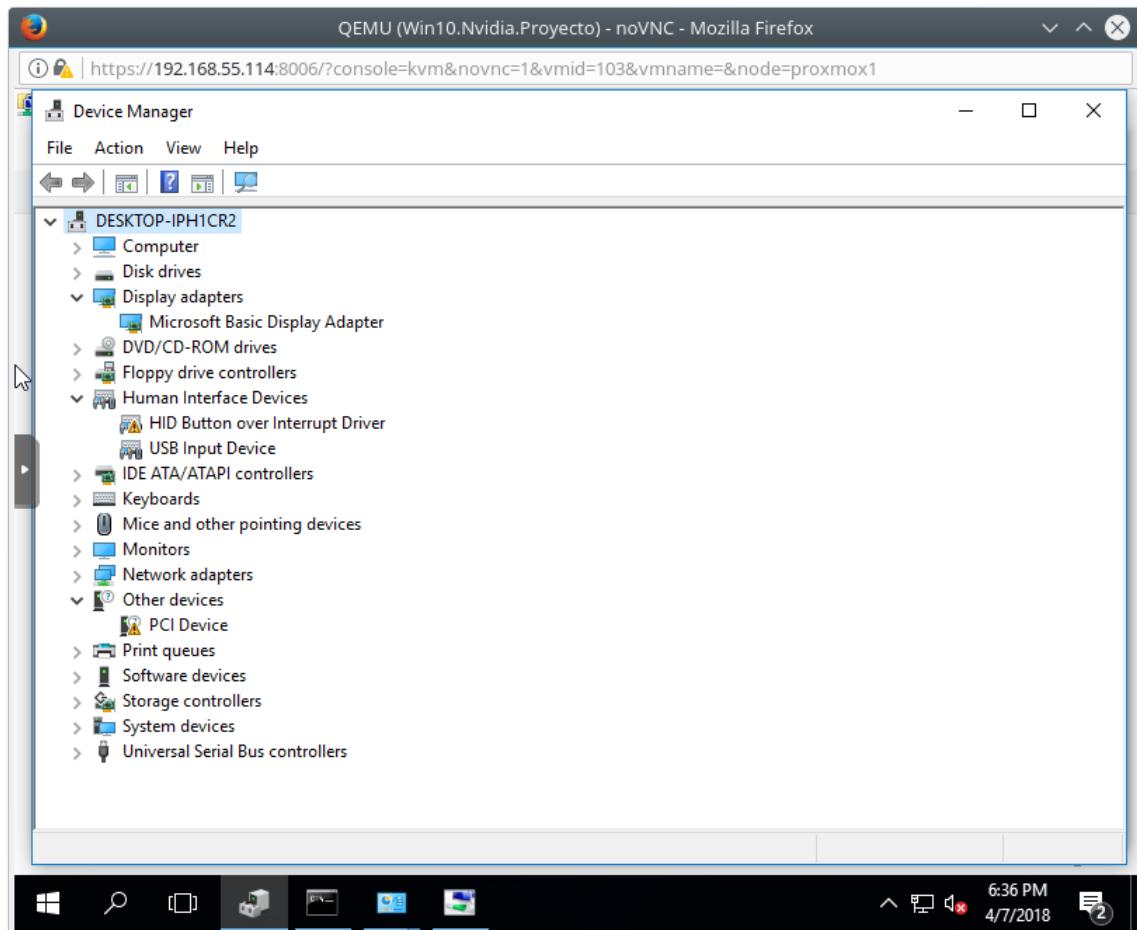
A continuación agregaríamos el disco EFI:

The screenshot shows the Proxmox VE interface. The left sidebar shows proxmox1 with VM 103 selected. The main window shows the 'Edit' screen for VM 103 under the 'Hardware' tab. A modal window titled 'Edit: EFI Disk' is open, showing the 'Storage:' dropdown set to 'P1-ZFS'. Below it, the 'Format:' dropdown is set to 'Raw disk image (raw)'. There are 'OK' and 'Reset' buttons at the bottom of the modal.

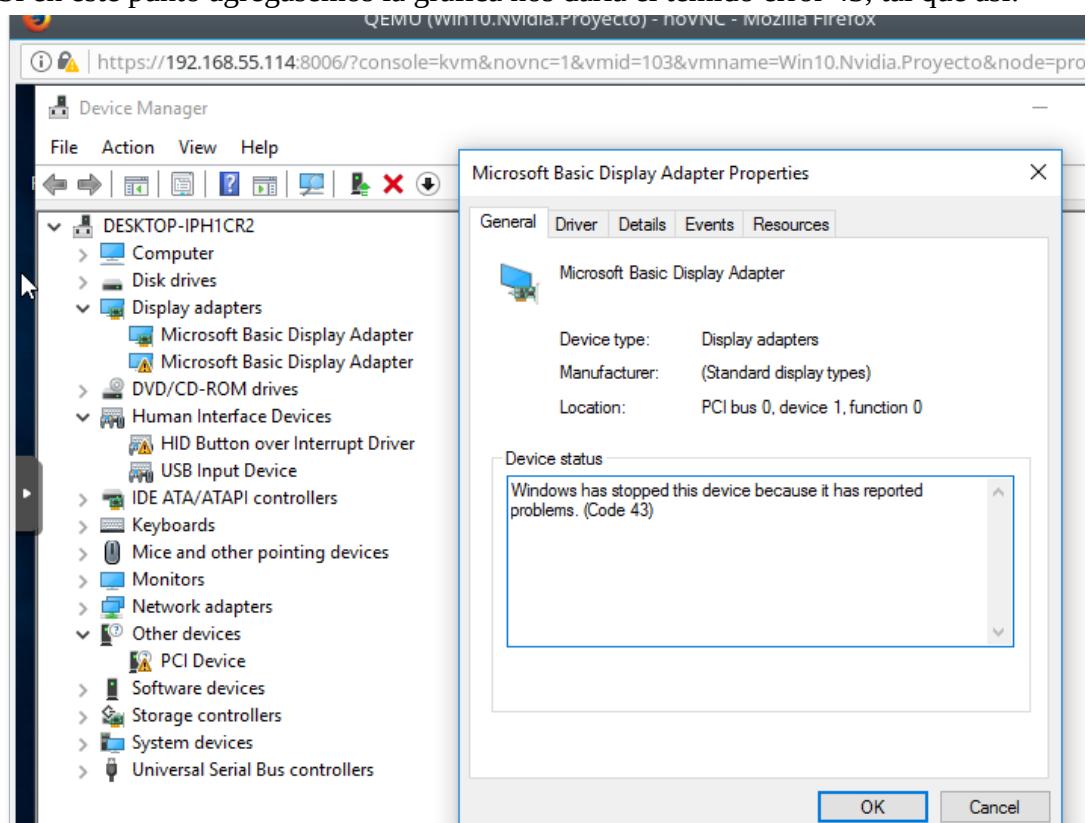
Con todo configurado, arrancaríamos la máquina para realizar la instalación:



Una vez acabe la instalación, los dispositivos de la máquina virtual estarán tal que así:



Si en este punto agregásemos la gráfica nos daría el temido error 43, tal que así:



Para evitar que esto pase, vamos a agregar una pequeña configuración a la configuración de la máquina virtual.

Editamos el fichero “**/etc/pve/qemu-server/IDMAQUINA.conf**”

Viendo una configuración tal que así:

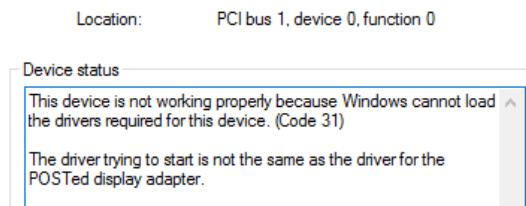
```
GNU nano 2.7.4                                     Fichero: /etc/pve/qemu-server/103.conf

bios: ovmf
bootdisk: sata0
cores: 4
cpu: host
efidisk0: P1-ZFS:vm-103-disk-2,size=128K
ide2: local:iso/14393.0.160715-1616.RS1_RELEASE_CLIENTENTERPRISE_S_EVAL_X64FRE_EN-US.ISO,media=cdrom
memory: 2048
name: Win10.Nvidia.Proyecto
net0: e1000=5A:B9:44:EA:2E:0E,bridge=vmbr666
numa: 0
ostype: win10
sata0: P1-ZFS:vm-103-disk-1,size=32G
scsihw: virtio-scsi-pci
smbios1: uuid=f3658778-dea4-43e4-811c-afc74fd0d71c
sockets: 1
```

Añadiríamos las siguientes lineas, las cuales son unos parámetros para la cpu, tipo de chipset a emular y puerto PCIe donde esté conectada la gráfica a utilizar:

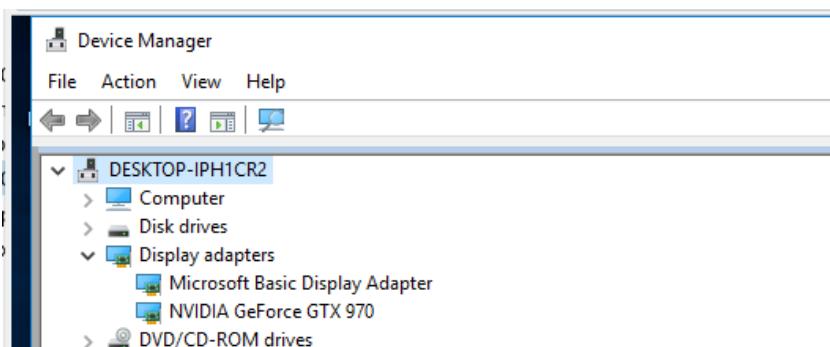
```
args: -cpu'host,kvm_pv_unhalt +kvm_pv_eoi,hv_vendor_id=NvidiaFix,kvm=off'
machine: q35
hostpci0: 05:00.0,pcie=1
```

Al arrancar la máquina recibiremos un error 31, cosa totalmente normal. Este nos indica que no puede cargar los drivers debido a que son distintos a los usados por la otra GPU.



Ahora nos conectaríamos al sitio de nvidia o bien con Windows update para descargar los drivers de la tarjeta gráfica.

Una vez instalados y tras un reinicio de la maquina virtual, nos desaparecerían de desaparecer los errores:



Comprobado que todo funciona bien, desactivaríamos la grafica VNC/SPICE dejando solamente la NVIDIA como gráfica principal, accediendo a ella mediante un puerto gráfico que tenga disponible (hdmi, dvi, displayport...)

Para hacer esto, debemos añadir a la linea “**hostpci0: 05:00.0,pcie=1**”, previamente configurada, el parámetro “**x-vga=on**” tal que así:

```
“hostpci0: 05:00.0,pcie=1,x-vga=on”
```

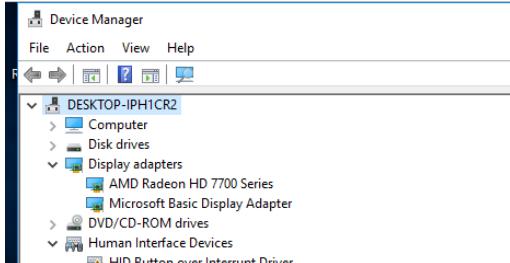
Máquina Virtual: AMD, BIOS temporal & Windows

En el caso de AMD es mas sencillo aún.

Tras realizar los mismos pasos de instalación para la máquina virtual de NVIDIA, apagaríamos la máquina y editaríamos el fichero

“`/etc/pve/qemu-server/IDMAQUINA.conf`”, añadiendo solamente las lineas:

```
machine: q35  
hostpci0: 02:00.0,pcie=1
```



Tras iniciar la maquina e instalar los controladores y que todo sea reconocido sin errores, podríamos añadir la opción “`x -vga=on`” que como hemos dicho previamente, desactiva la grafica VNC o SPICE dejando únicamente la tarjeta grafica dedicada.

Dado que mi tarjeta AMD es vieja y de un fabricante secundario, funciona con UEFI pero el bit de comprobación del soporte de uefi no lo tiene activado, por lo que voy a indicar como cargar otra bios que si lo soporte, simplemente por el hecho de que salga marcado, ya que en mi caso, funcionar, funciona sin el.

Nuestro modelo es este: Club3D HD 7770 Bios 015.014.000.001.000746

<https://www.techpowerup.com/vgabios/120999/club3d-hd7770-1024-120130>

Y vamos a cargar la BIOS de esta tarjeta: Gigabyte HD7700 Bios 015.040.000.003.000000

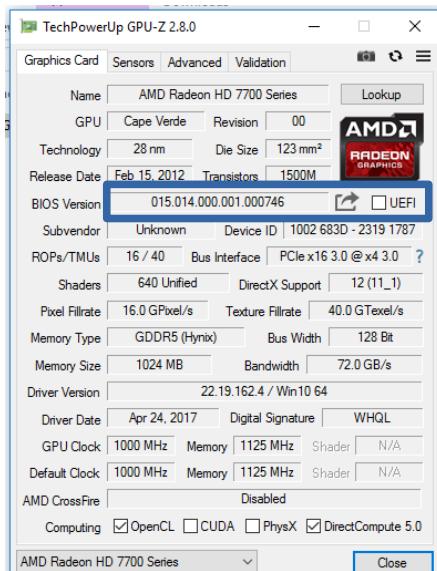
<https://www.techpowerup.com/vgabios/155238/gigabyte-hd7770-1024-131021>

Para modificarlo, nos descargariamos la bios y la guardariamos en “`/usr/share/kvm`”, yo lo he llamado “`7770UEFI.bin`”

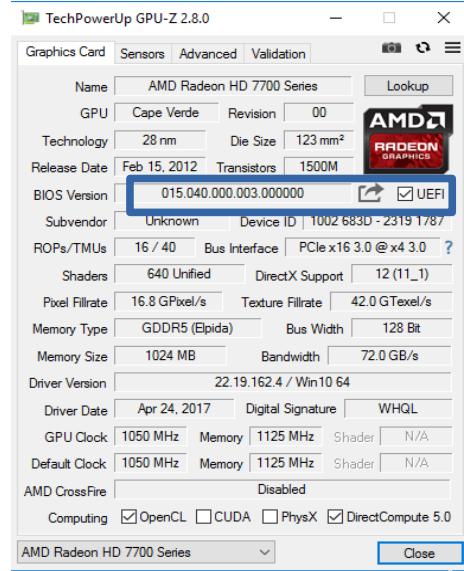
Para indicarle la BIOS a cargar, añadiríamos en el archivo de configuración:

```
“hostpci0: 02:00.0,pcie=1,romfile=7770UEFI.bin”
```

Antes, por defecto:



Después:



Máquina Virtual: Nvidia & Mac Os & AMD

Nuestra instalación se basa en la versión “High Sierra” de Mac Os X.

Dado el estado “alegalidad” que conlleva instalar Mac fuera de su “mundo burbuja”, no voy a explicar los pasos para conseguir los ficheros.

El primer paso sería conseguir la ISO del sistema operativo. En nuestro caso usamos un MacOs real para generarla. También deberemos de integrar en el disco “Clover.qcow2” el fichero “FakeSMC.kext”. Clover actuará como gestor de arranque.

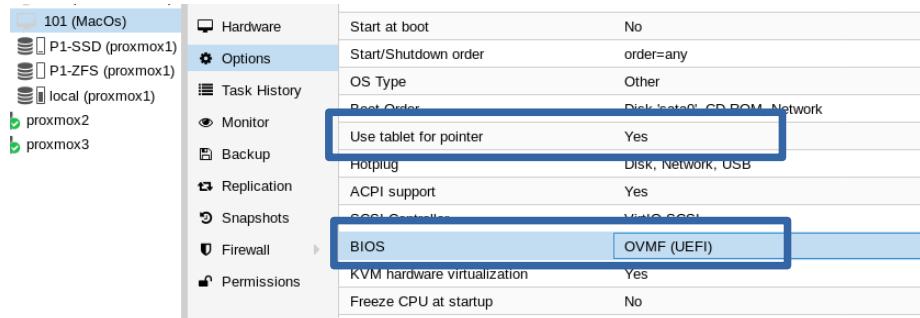
El fichero FakeSMC.kext se encargará de sobrepasar la key común que utiliza apple para comprobar que es un hardware original, a partir de la versión 2.12 de qemu en proxmox no debería hacer falta utilizar este fichero, puesto que existe la opción para utilizar la clave común de apple.

Creamos la maquina virtual con las siguientes características:

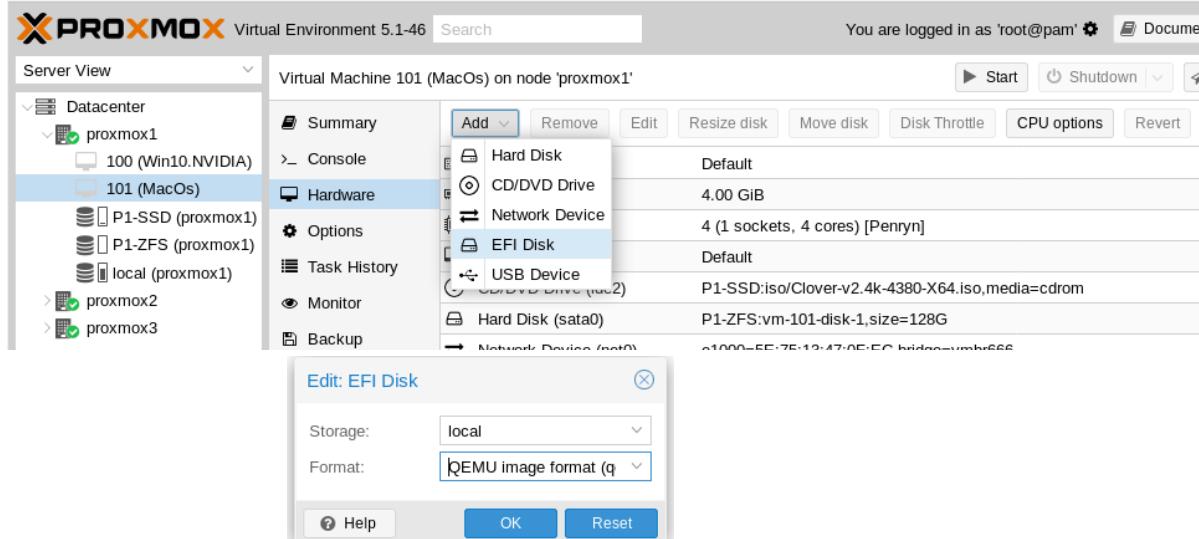
The screenshot shows the 'Create: Virtual Machine' interface in Proxmox VE, displaying five configuration tabs:

- General:** Node: proxmox1, VM ID: 101, Name: MacOs.
- OS:** Guest OS: Other, Storage: P1-SSD, ISO image: macos.iso.
- Hard Disk:** Bus/Device: SATA, Storage: P1-ZFS, Disk size (GiB): 128, Format: Raw disk image (raw).
- CPU:** Sockets: 1, Cores: 4, Type: Penryn.
- Memory:** Memory (MiB): 4096, Ballooning: (unchecked).
- Network:** Mode: Bridged mode, Model: Intel E1000, MAC address: auto, Rate limit (MB/s): unlimited, Multiqueue: 1, Disconnect: .

En el menú de opciones de la máquina virtual comprobaremos que el parámetro “Use table for pointer” esta en “yes” (enabled). Cambiaríamos la BIOS a modo UEFI:



Y agregaríamos un disco efi:



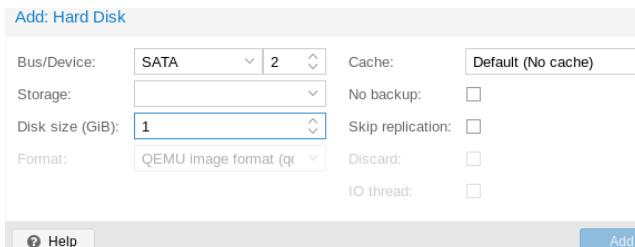
Editaríamos el fichero “/etc/pve/qemu-server/IDMAQUINA.conf” de nuestra maquina MacOs para añadir los siguientes parámetros:

```
machine:pc-q35-2.9
args: -smbios type=2 -cpu Penryn,kvm=on,vendor=GenuineIntel,
+invtsc,vmware-cpuid-freq=on
```

También modificaríamos la linea “net0: e1000=MAC,bridge=SWITCHVIRTUAL” para dejarla en mi caso tal que así: “net0: e1000-82545em=5E:75:13:47:0F:EC,bridge=vmbr666” ya que MacOs no soporta la intel e1000 estándar

A parte, añadiríamos otro disco duro ide que contendrá Clover, por lo que deberemos copiar el fichero a la carpeta images del id de nuestra máquina (/var/lib/vz/images/ID/) y añadirlo al fichero de configuración. En mi caso : “ide0: local:101/clover.qcow2”.

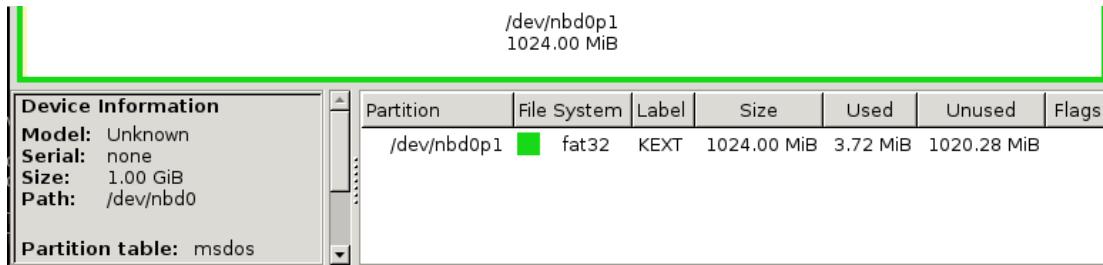
Ahora, mediante la gui, crearíamos un disco duro qcow2 de 1GB.



El cual debemos montar ejecutando los comandos:

```
modprobe nbd
qemu-nbd -c /dev/nbd0 /var/lib/vz/images/101/vm-101-disk-2.qcow2
```

Una vez montado, crearíamos una tabla de particiones ms-dos y la formatearíamos con formato fat32, bien por comandos o bien por gui ejecutando el comando “`gparted /dev/nbd0`” :



Hecho esto montaríamos la partición formateada con el comando ‘`mount /dev/nbd0p1 /mnt/2`’ y después copiaríamos el kext modificado.

Sin este kext modificado, MacOs no arrancaría. “`cp -r ./FakeSMC.kext /mnt/2/`”

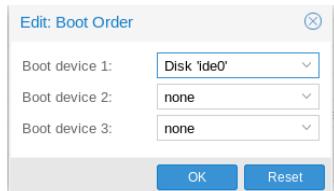
Añadiríamos ese disco a la configuración bien por gui o añadiéndolo a mano:

“`sata1: local:101/vm-101-disk-2.qcow2, size=1G`”

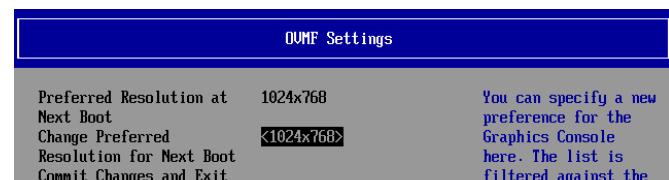
Como último paso, habilitaremos el teclado virtual por usb, ya que macos no soporta ps2. Para hacerlo añadiríamos al final del fichero “`/usr/share/qemu-server/pve-q35.cfg`” la linea:

```
[device "keyboard1"]
driver = "usb-kbd"
bus = "ehci.0"
port = "2"
```

Ahora en el apartado “Options” de nuestra máquina virtual editaríamos el “Boot Order” y seleccionaríamos como “Boot device 1” el id de disco que hemos puesto a clover, en mi caso ide0:



Iniciaríamos la máquina y entraríamos en la bios con la tecla “Esc”. Iríamos a “Device Manager” > “OVMF Platform Configuration” y modificariámos la opción “Change Preferred Resolution for Next Boot” a “1024x768”:

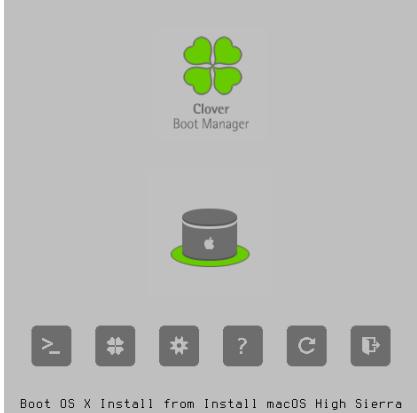


Salvando luego los cambios con “Commit Changes and Exit”.

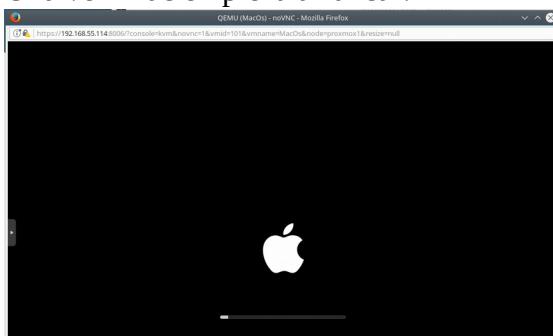
A partir de Proxmox 5.1 necesitamos una version de UEFI para MacOs diferente a la que viene con proxmox. Para ello nos descargaremos el paquete desde aquí:

http://s3.nicksherlock.com/forumposts/2018/pve-edk2-firmware_1.20180316-1_all.deb y lo instalaremos con el comando: “`dpkg -i pve-edk2-firmware_1.20180316-1_all.deb`”

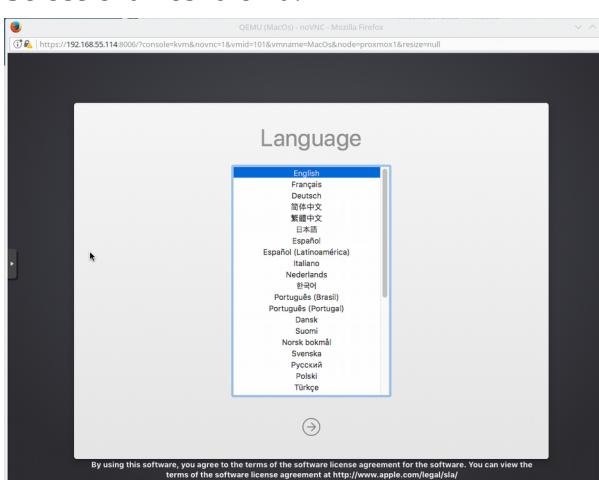
Una vez arranque clover, nos tendría que salir la opción “Boot OS X Install from Install macOS High Sierra:



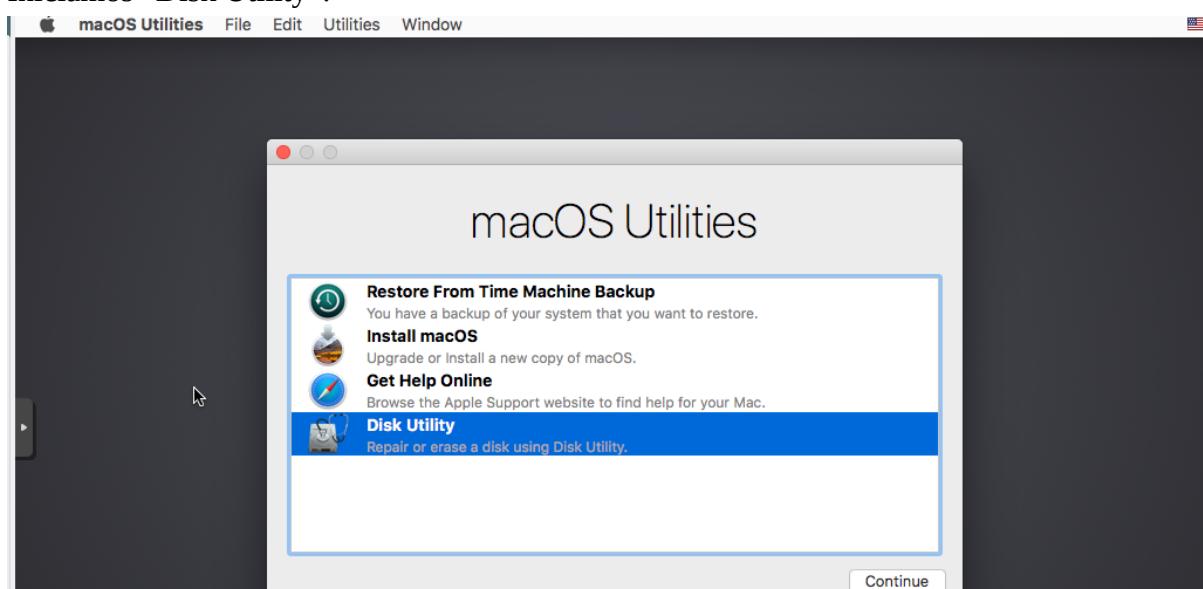
Una vez Mac empieza arrancar:



Seleccionamos idioma:

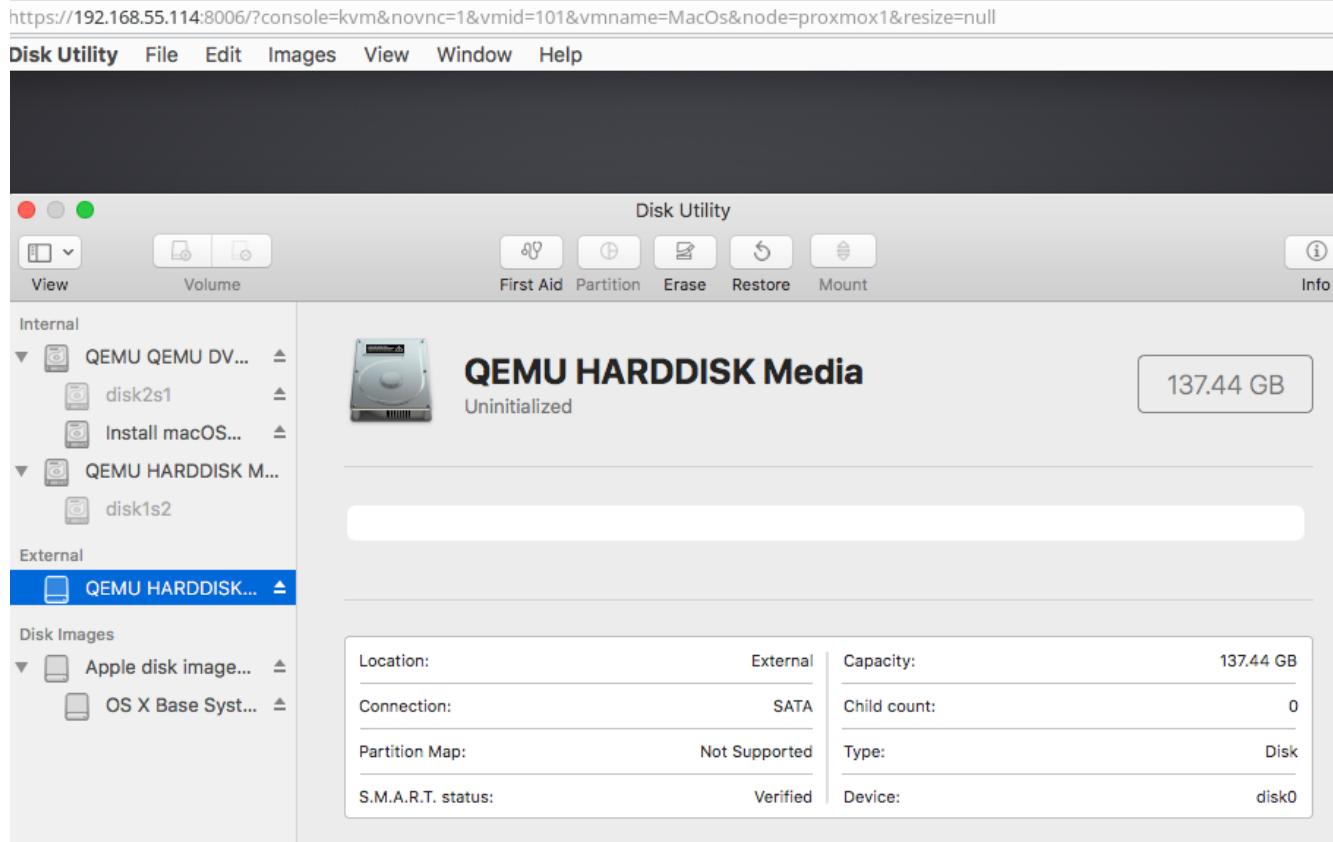


Iniciamos “Disk Utility”:

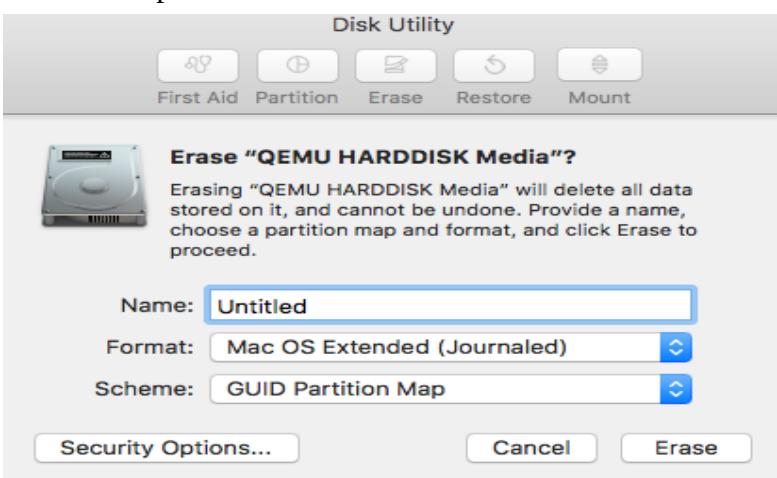


Clickamos en “View” > “Show all devices” y cerramos la utilidad “Disk Utility”, volviendo a la pantalla de la imagen anterior. Ahora volveremos a abrir Disk Utility y nos aparecerá nuestro disco en el apartado “External”

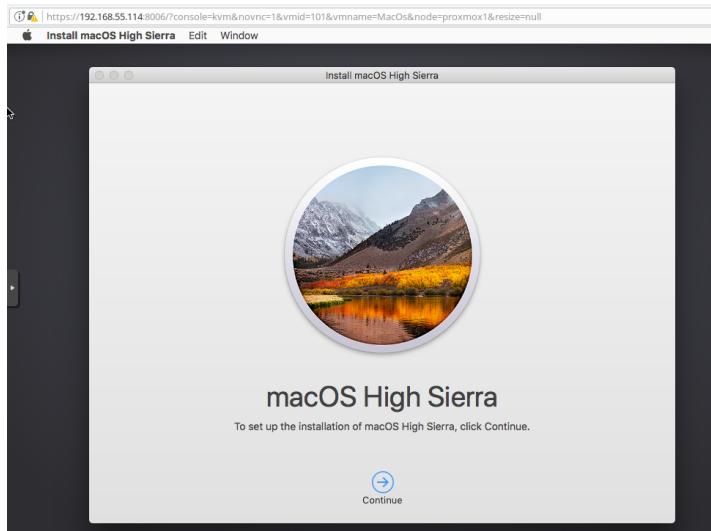
Formateamos nuestro disco sata, fijándonos en el tamaño para no equivocarnos de unidad:



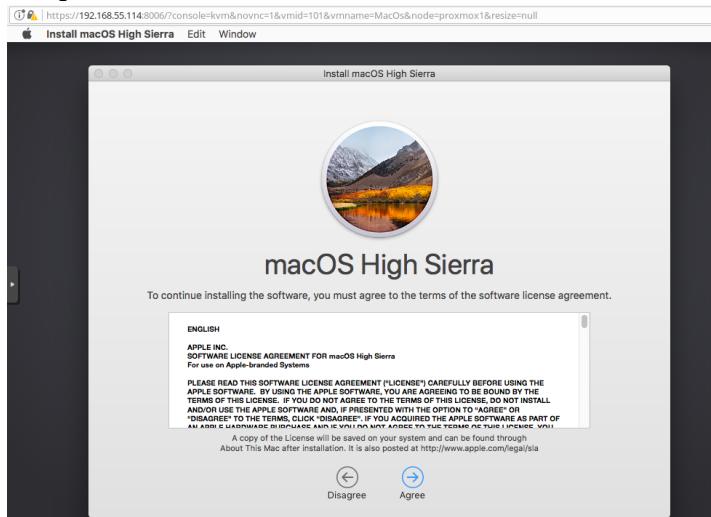
En la segunda opción, debemos marcar “Mac OS Extendend (Journaled), una vez hecho, clickaríamos en “Erase” para darle formato:



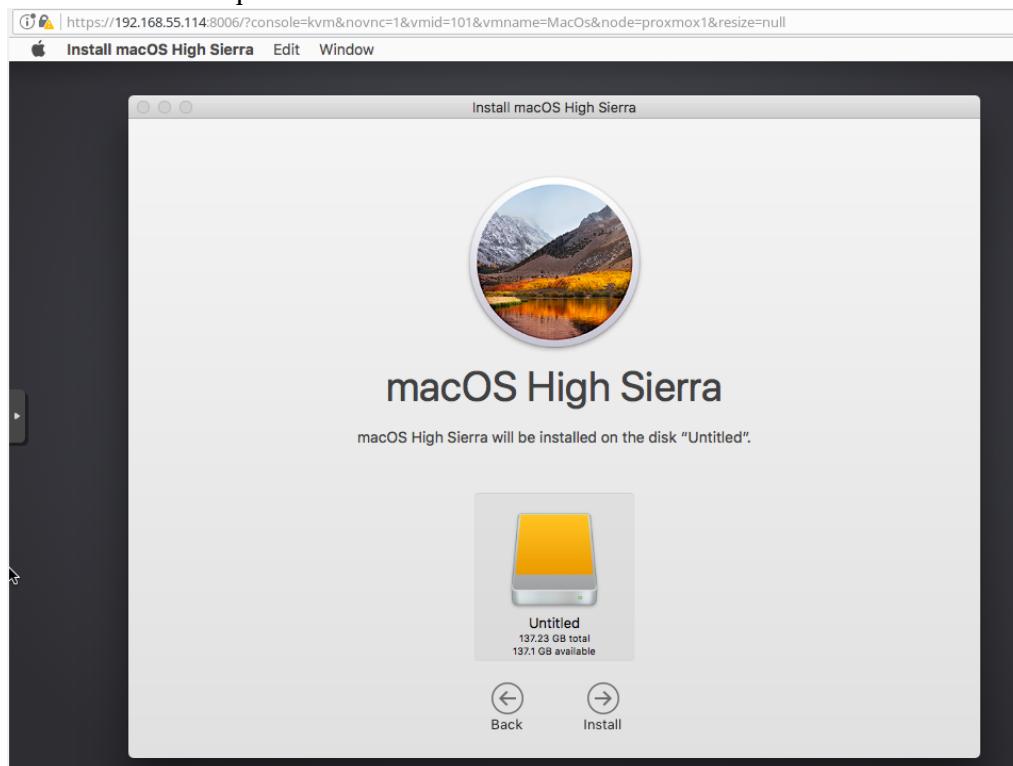
Una vez hecho esto cerramos la utilidad de discos y continuaremos con la opción “install macOS”:



Aceptaríamos su licencia:



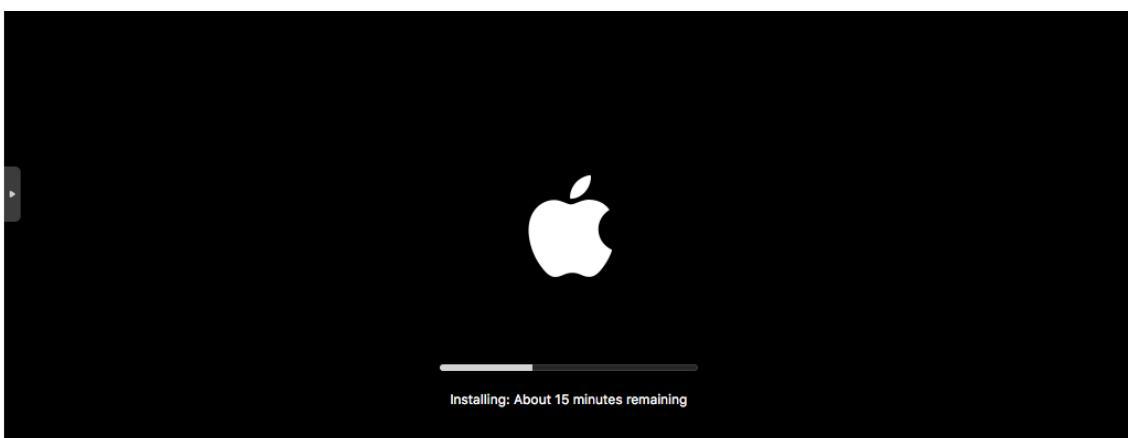
Le indicamos en que disco instalar:



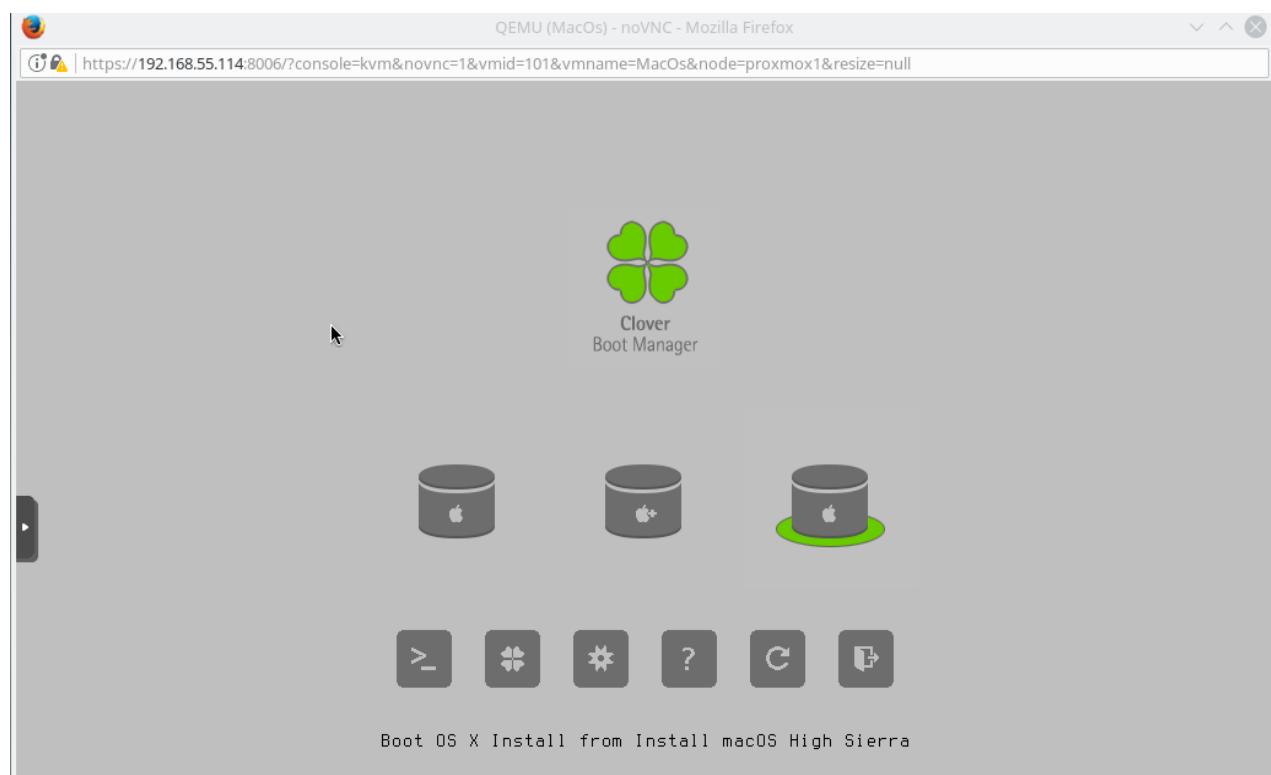
Comenzaría la copia de datos al disco:



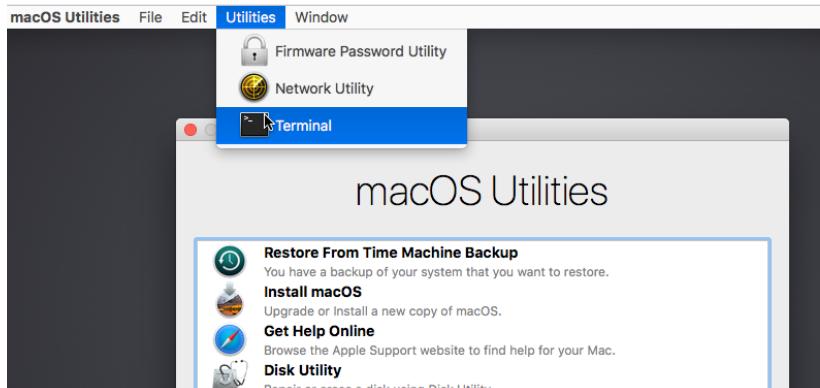
Después de copiar los datos, el equipo se reiniciara y empezará la instalación:



Una vez finalice la pantalla anterior el equipo se volverá a reiniciar y esta vez veremos más opciones, pero arrancaremos desde el cd de nuevo. Opción “Boot OS x Install from Install macOS High Sierra”:

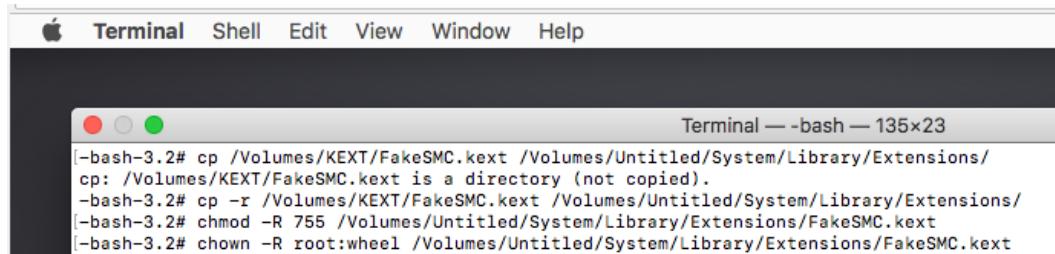


Una vez esté arrancado, abriremos la terminal:

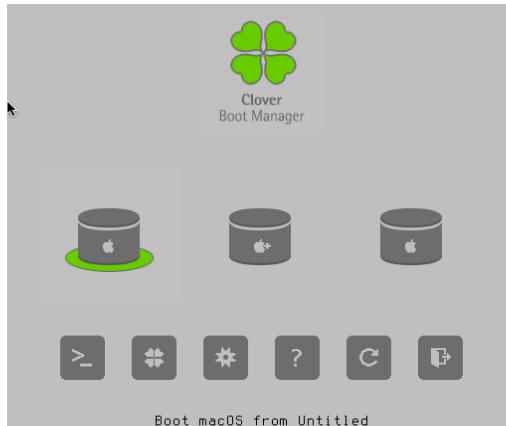


y copiaremos los kext al volumen de instalación de Mac en la carpeta “/System/Library/Extensions”:

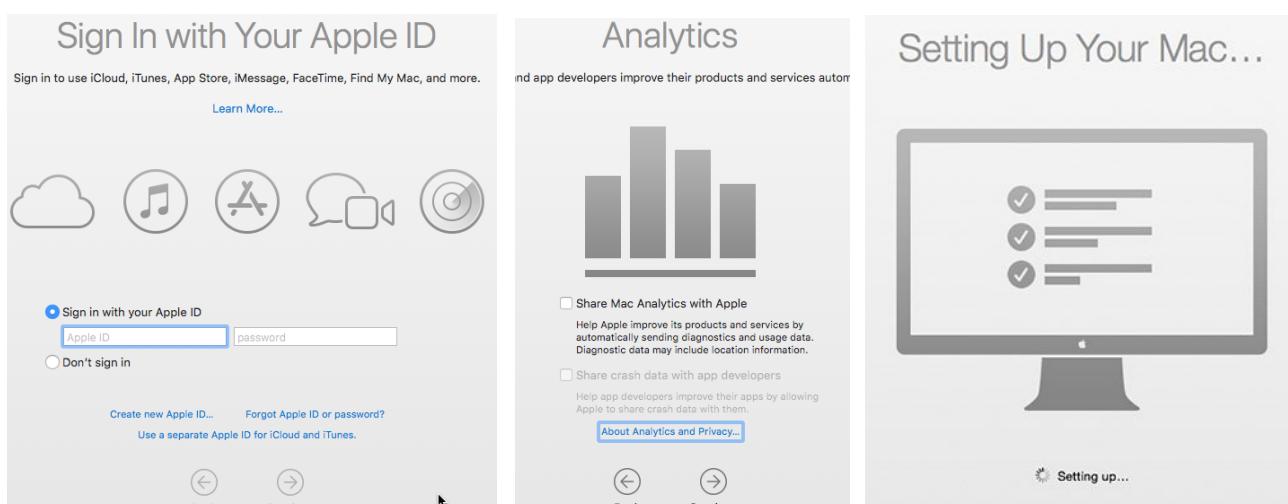
```
cp -r /Volumes/KEXT/FakeSMC.kext /Volumes/Untitled/System/Library/Extensions  
chmod -R 755 /Volumes/Untitled/System/Library/Extensions/FakeSMC.kext  
chown -R root:wheel /Volumes/Untitled/System/Library/Extensions/FakeSMC.kext
```



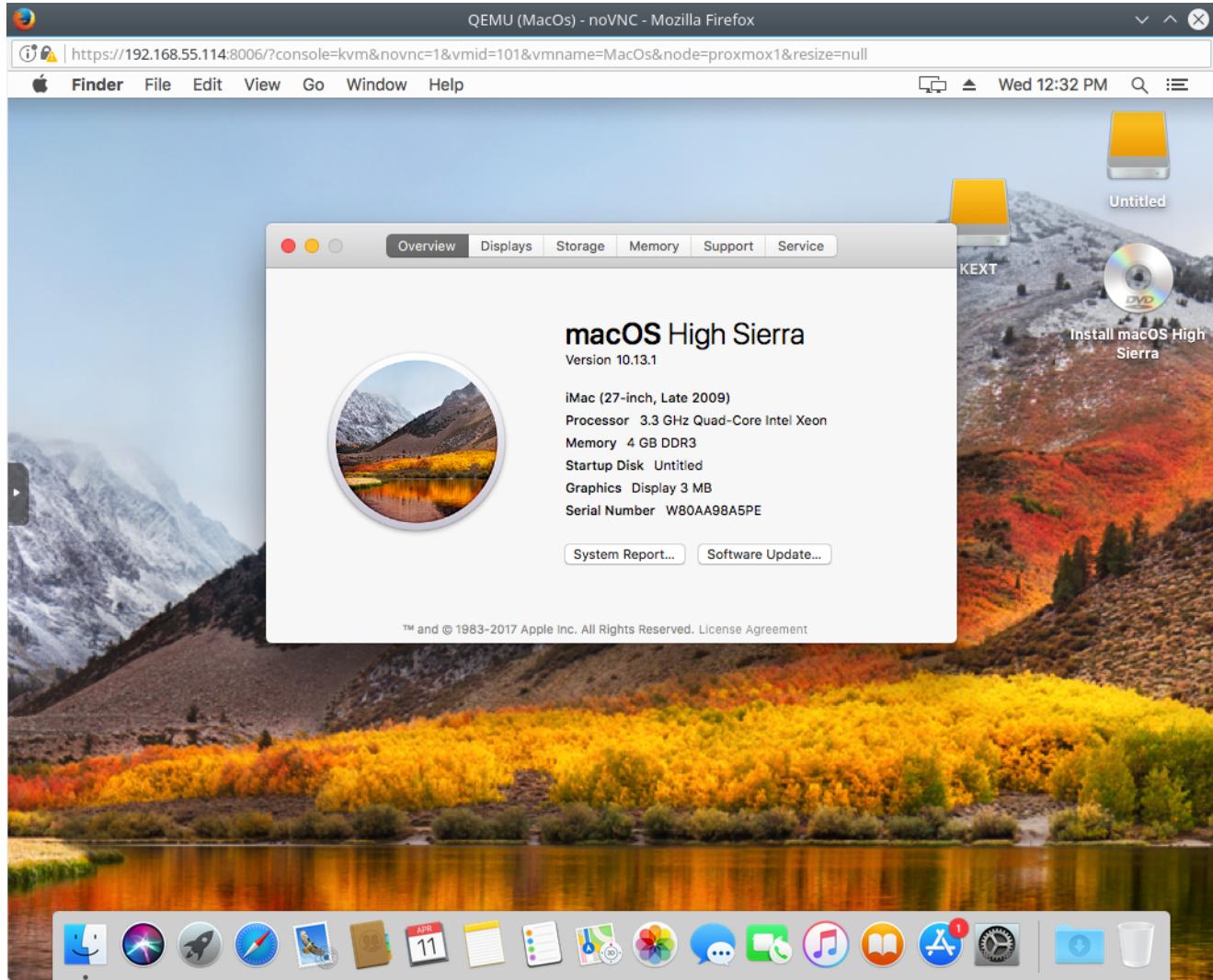
Ahora arrancaríamos desde la opción “Boot macOS from NombreDisco”



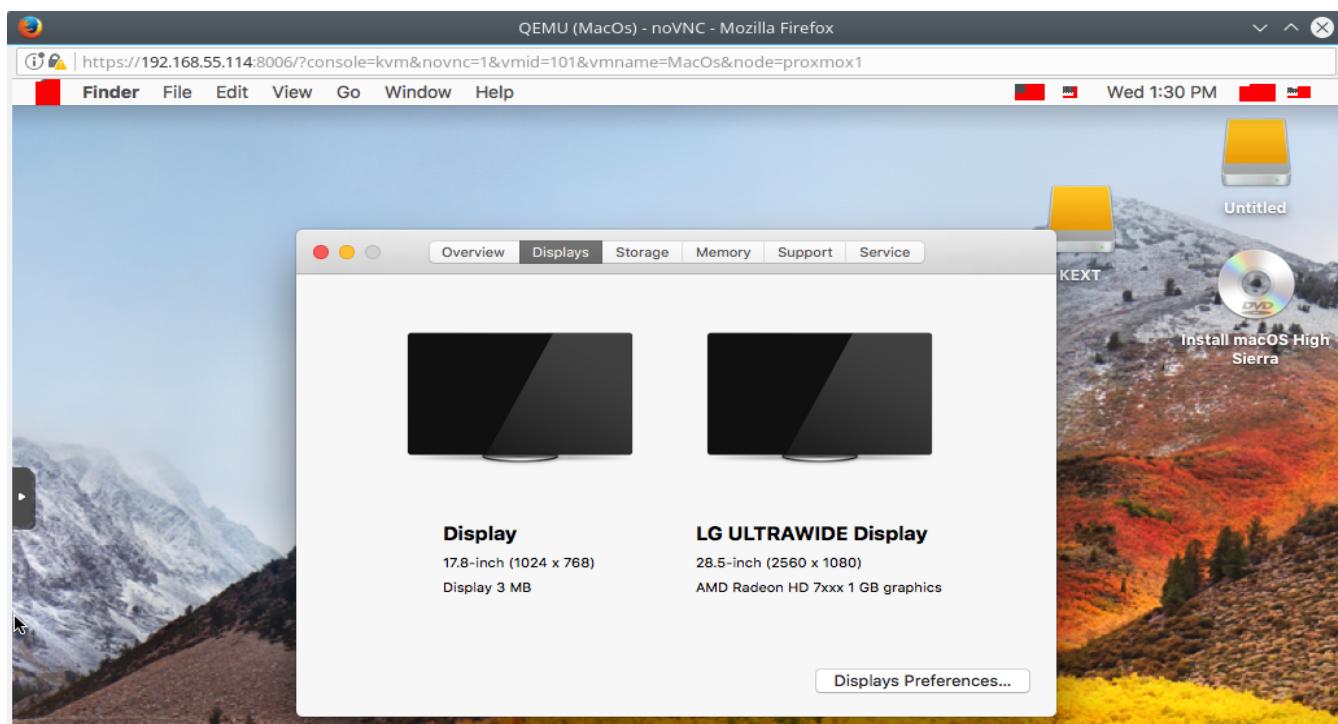
El sistema arrancaría mostrándonos el asistente de configuración de macOS:



Una vez hecho esto tendremos el sistema listo para usar:



Una vez tengamos el sistema instalado, con añadir la misma configuración de “hostpci0” que en las máquinas de Windows macOS debería ser capaz de detectar las gráficas:



Pero llegados a este punto, deberíamos empezar a investigar hasta dar con un driver adecuado, ya que es posible encontrar fallos gráficos, como es mi caso:



En el caso de la AMD, la forma más rápida de hacerla funcionar fue añadiéndola como segunda tarjeta, evitando poner la linea “x-vga=on”.

En el caso de NVIDIA, la forma más rápida de hacerla funcionar fue dedicándola como tarjeta única, sin embargo, no obtuvimos la resolución nativa, por lo que probablemente esté arrancando con los drivers VESA, por lo que tendríamos que pegarnos buscando un driver de nvidia.



Máquinas Virtuales y Físicas: Intercambio de drivers

Introducción

Dado que para poder hacer la dedicación de un dispositivo PCIe a una maquina virtual se utiliza el driver “vfio-pci”, perdemos el uso de este dispositivo para la máquina física.

Una vez terminemos con el dispositivo, existiría la opción de sustituir ese driver por el válido para usarse de nuevo en el sistema.

Pero como todo, tiene un pero. Esto para dispositivos de audio funciona sin ningún tipo de problemas. Con gráficas, en AMD según que últimas gráficas salidas al mercado puede dar fallos, pero una vez se encuentra porque falla, empieza a funcionar de manera estable.

El mayor problema lo tiene nvidia, y es que en sus drivers privados, cuando intentas cambiar el driver “vfio-pci” el sistema se vuelve muy inestable, realizándose reinicios del entorno X y congelaciones del sistema, lo que lo hace inviable. Además de no poder mezclar aceleracion gráfica entre drivers libres y privativos.

Configuración

Para hacer realizar esta configuración nos basaremos en la tarjeta nvidia 610 con driver nouveau y en una tarjeta AMD, en este caso una 290x que hemos adquirido, con driver amdgpu.

Debemos configurar el fichero “`/etc/X11/xorg.conf`” añadiendo las dos tarjetas graficas, en mi caso ejecuté el comando “`Xorg -configure`” para tener una plantilla y a partir de ahí hice las modificaciones correspondientes para:

1. Solo tener pantallas en la tarjeta nvidia (por lo que si nos detecta en AMD, deberemos eliminarlas).
2. Añadir la opcion `Option "dri3" "1"` en la tarjeta que se usará para procesar los gráficos 3D.
3. Añadir la opción `Option "DRI" "3"` en la tarjeta gráfica menos potente.

```
Section "ServerLayout"
    Identifier "X.org Configured"
    Screen 0 "Screen1" 0 0
    InputDevice "Mouse0" "CorePointer"
    InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

[...]

```
Section "Device"
    Option "dri3" "1"
    Identifier "Card0"
    Driver "amdgpu"
    BusID "PCI:5:0:0"
EndSection
```

```
Section "Device"
    Option "DRI" "3"
    Identifier "Card1"
    Driver "nouveau"
    BusID "PCI:3:0:0"
EndSection
```

[...]

Para hacer el intercambio de drivers podemos utilizar el script aquí dado.
Como primer parámetro deberemos introducir el ID pci y como segundo parámetro el driver que queremos cargar.

```
echo "driver PREVIO"  
lspci -nnk -s $1 | grep use:  
VENDOR=$(cat /sys/bus/pci/devices/0000\:$1/vendor)  
DEVICE=$(cat /sys/bus/pci/devices/0000\:$1/device)  
  
echo "$VENDOR $DEVICE" > /sys/bus/pci/drivers/$2/new_id  
  
echo "0000:$1" > /sys/bus/pci/devices/0000:$1/driver/unbind  
echo "0000:$1" > /sys/bus/pci/drivers/$2/bind  
echo "$VENDOR $DEVICE" > /sys/bus/pci/drivers/$2/remove_id  
  
echo "driver CAMBIADO"  
lspci -nnk -s $1 | grep use:
```

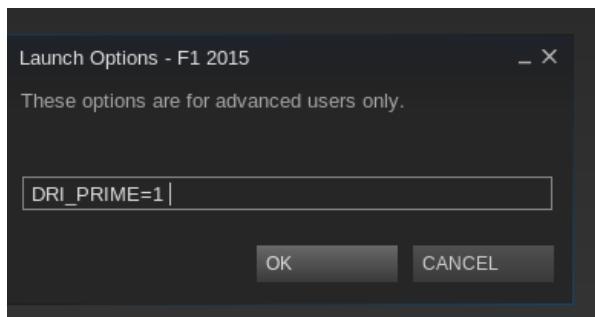
Ejecución:

```
root@proxmox1:/home/zeusinprox# sh scriptgpu2 05:00.0 vfio-pci  
"driver PREVIO"  
    Kernel driver in use: amdgpu  
"driver CAMBIADO"  
    Kernel driver in use: vfio-pci  
root@proxmox1:/home/zeusinprox# sh scriptgpu2 05:00.0 amdgpu  
"driver PREVIO"  
    Kernel driver in use: vfio-pci  
sh: echo: I/O error  
sh: echo: I/O error  
"driver CAMBIADO"  
    Kernel driver in use: amdgpu  
root@proxmox1:/home/zeusinprox# □
```

Siempre que queramos correr algún proceso con la tarjeta gráfica potente, deberemos usar la opción “**DRI_PRIME=1**” precedido del comando, en caso de querer ejecutarlo con la tarjeta gráfica básica, podemos ejecutarlo sin más o bien forzarlo con la opción “**DRI_PRIME=0**”

```
zeusinprox@proxmox1:~$ DRI_PRIME=1 glxinfo | grep "OpenGL render"  
OpenGL renderer string: Gallium 0.4 on AMD HAWAII (DRM 3.18.0 / 4.13.16-2-pve, LLVM 3.9.1)  
zeusinprox@proxmox1:~$ DRI_PRIME=0 glxinfo | grep "OpenGL render"  
OpenGL renderer string: Gallium 0.4 on NVD9  
zeusinprox@proxmox1:~$ □
```

Por ejemplo, para un juego de steam, estableceríamos en las opciones de ejecución del juego así:



Sistemas multipuesto

Introducción

Otro tema muy interesante son los sistemas multipuesto. Partiendo de mi base, que puedo tener hasta 4 tarjetas gráficas instaladas, podría tener cuatro máquinas virtuales con sus gráficas y usb para teclado/ratón y en cada máquina virtual podría estar funcionando una persona trabajando independientemente (Teclado, Pantalla, Ratón).

Pero a pesar de que esta abstracción nos puede brindar bastantes beneficios también tiene su desventajas, como obligar a redimensionar mas en recursos de hardware, debido a la necesidad de aguantar cuatro sistemas operativos completos.

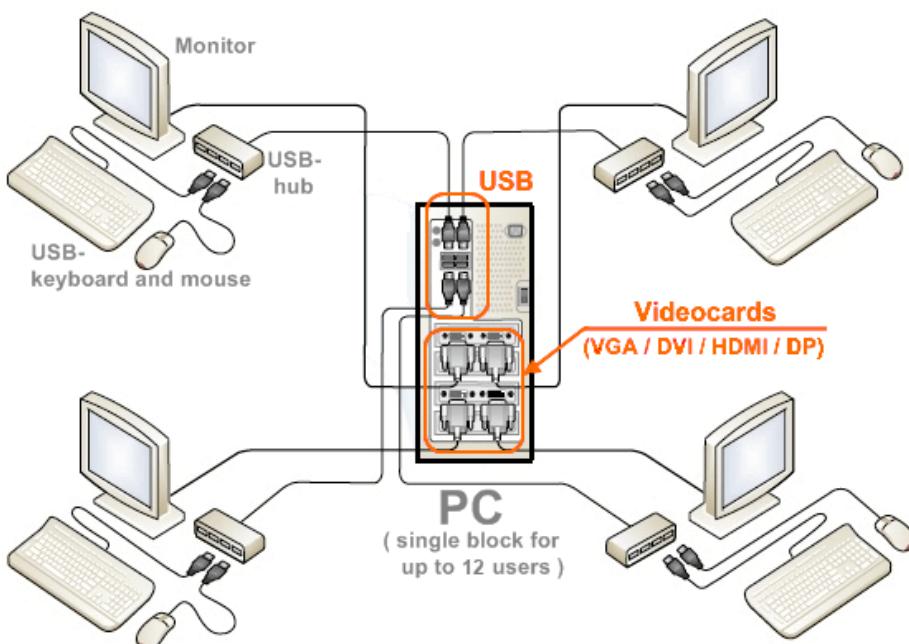
Para evitar este problema existen alternativas tanto a nivel de Windows como de Linux.

Configuración

Windows

Dejando de lado que tenemos Windows Multipoint como rol en Windows Server 2016 y dado que es no es muy apto para correr programas con aceleración 3D tenemos que recurrir a aplicaciones de terceros para esta tarea.

En Windows 10 existe un programa llamado “Ibik Aster” que permite utilizar hasta 12 salidas de video disponibles entre diferentes tarjetas gráficas.

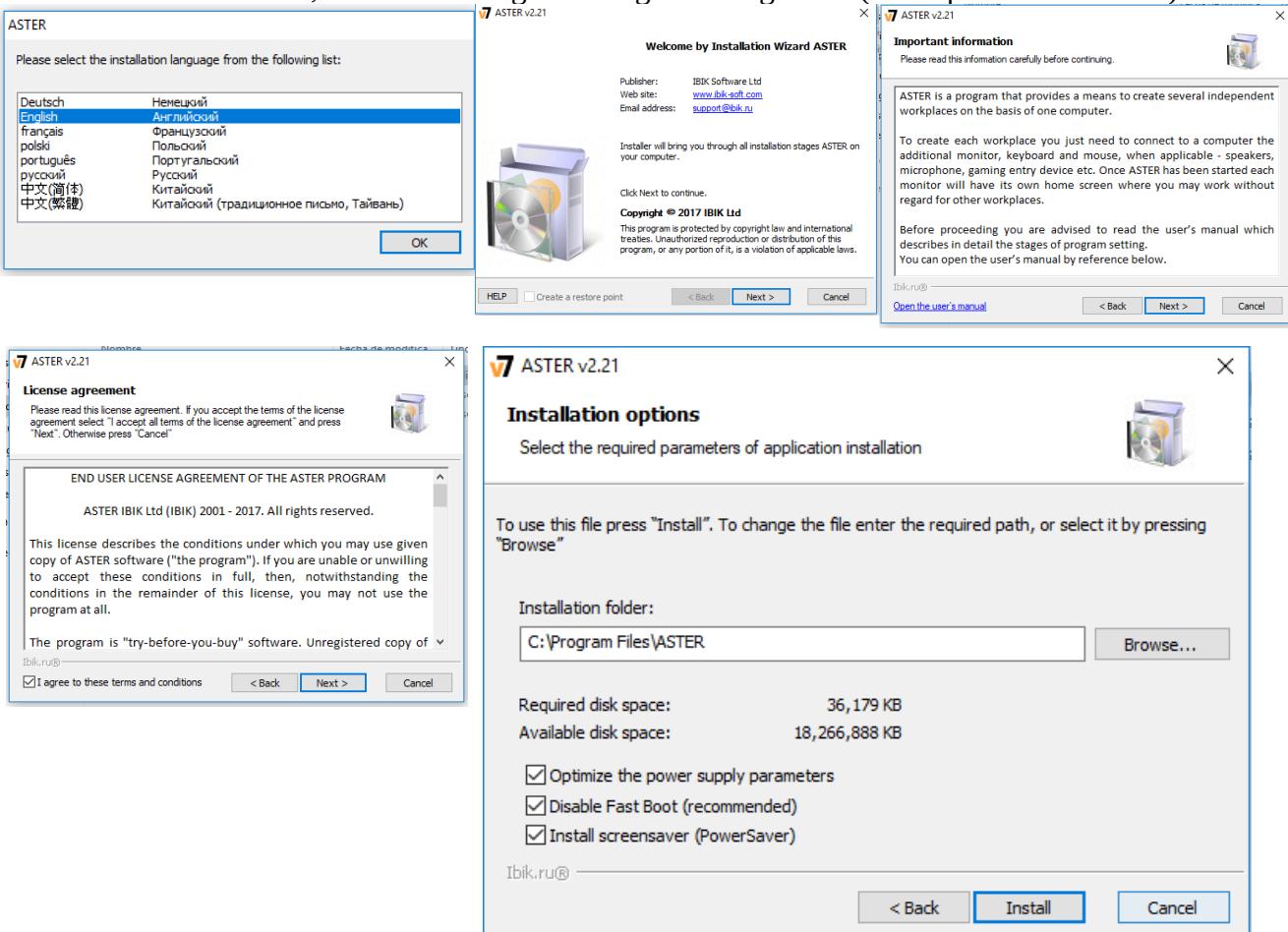


Aster Es un programa de pago y de código cerrado desarrollado por una empresa rusa , por lo cual es un misterio el como funciona a nivel interno y todo en un ejecutable de 30mb.

La comunidad más activa en este programa está en su mayoría en ruso, por lo cual es muy difícil estar al día en cuanto a las noticias de este programa.

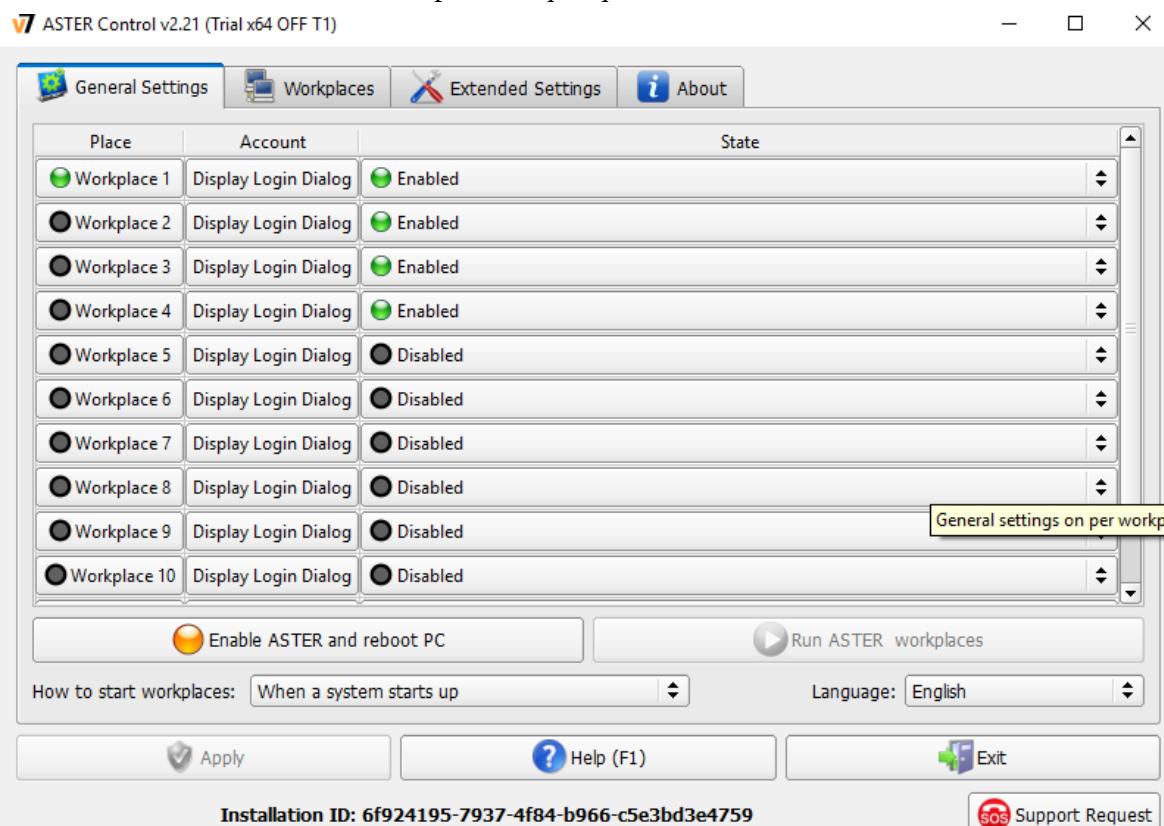
2 Usuarios		6 Usuarios	Usuario adicional (Hasta 12)
De por vida	Por año	De por vida	
54€	18€	108€	30€

La instalación es sencilla, básicamente siguiente-siguiente-siguiente (de izquierda a derecha):

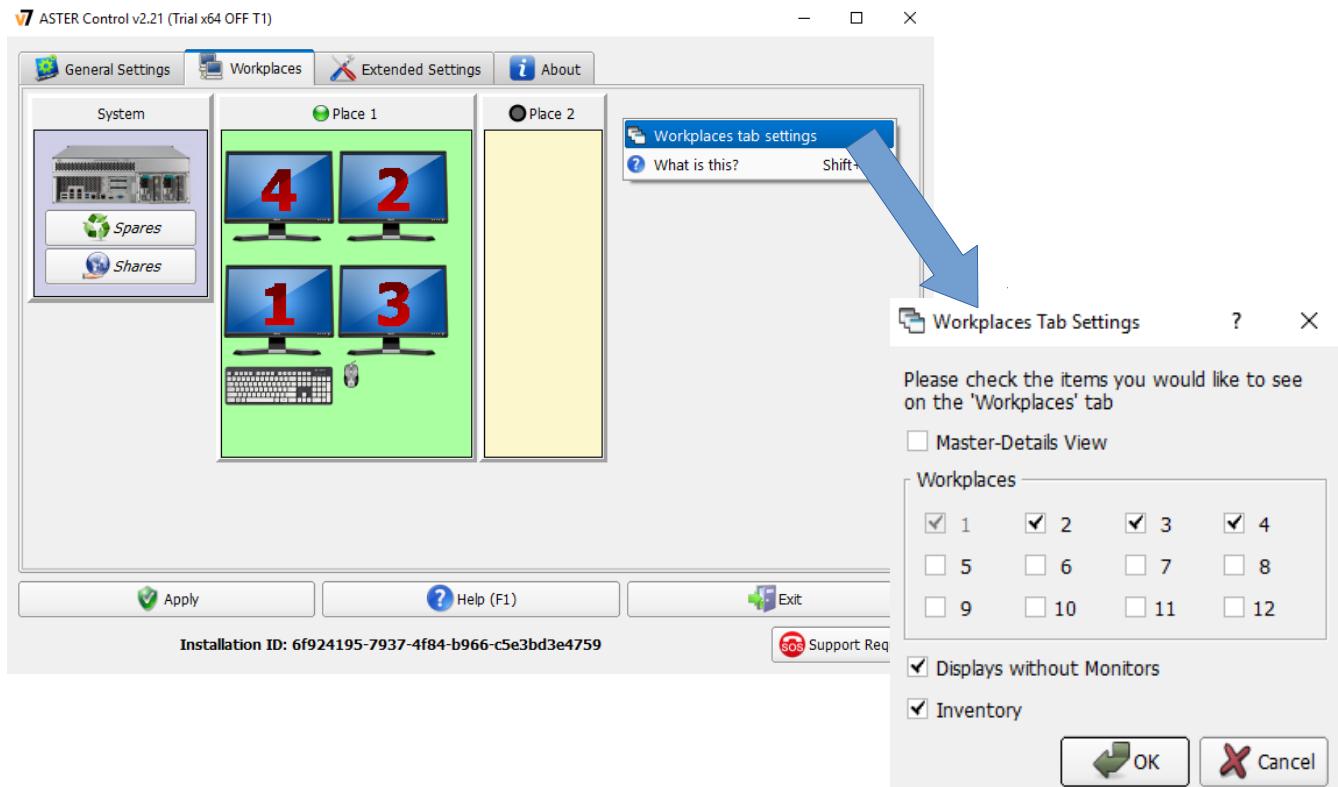


Una vez acabe de instalar, reiniciaremos la máquina.

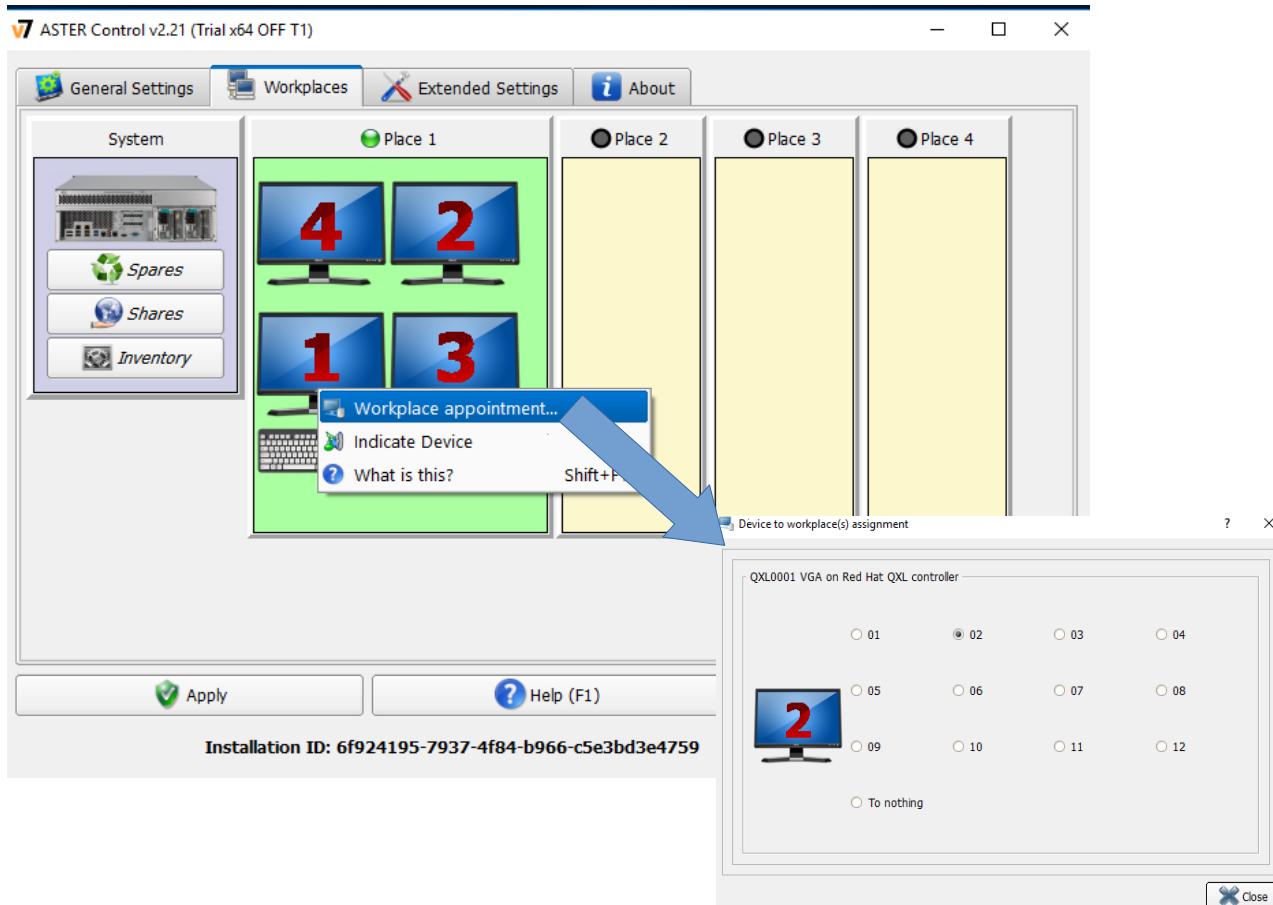
Una vez reiniciado, arrancaríamos el panel de configuración de aster y en la pestaña “General Settings” marcaríamos como “Enabled” los puestos que queramos utilizar:



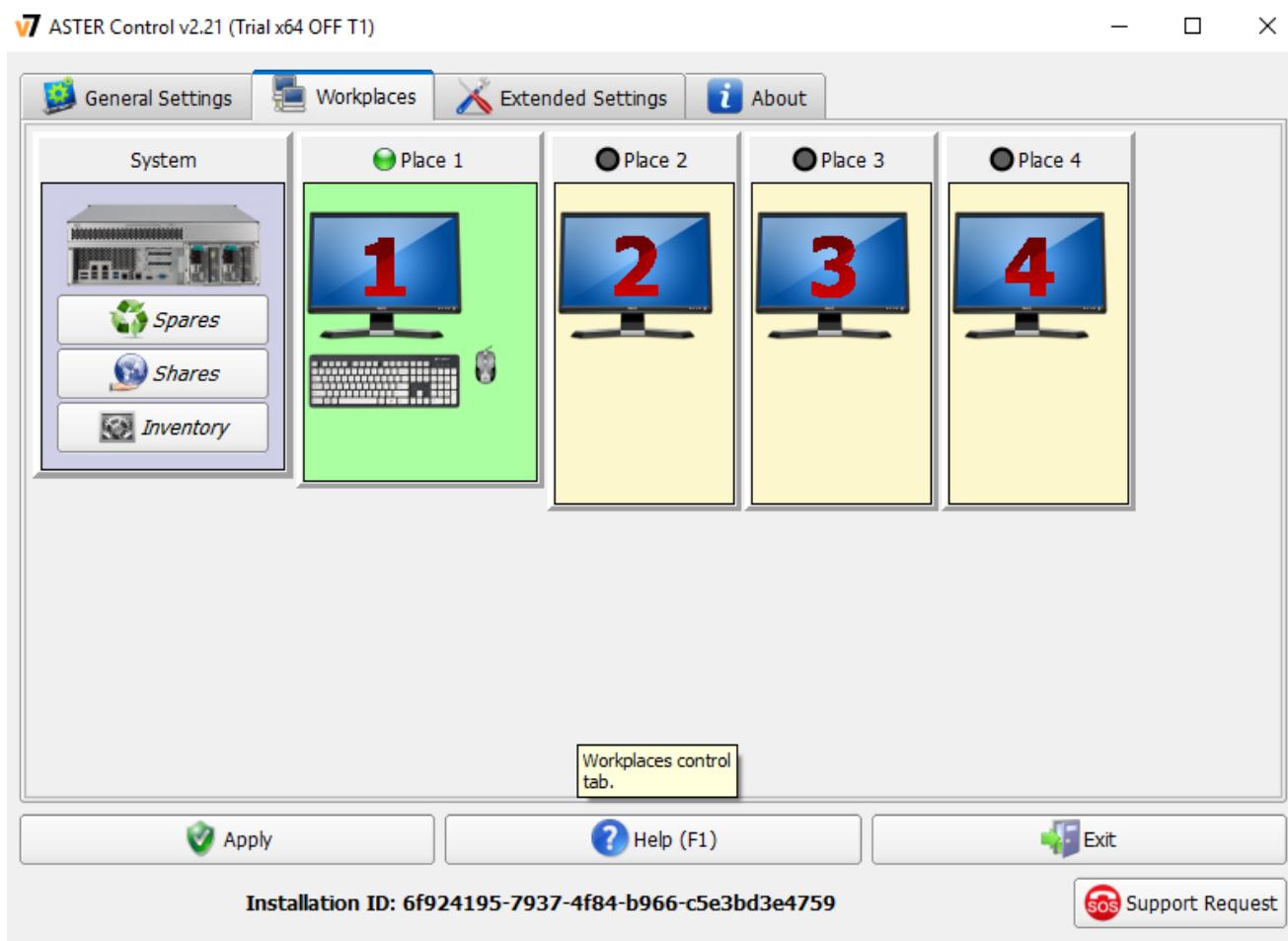
Por defecto todas las pantallas están asignadas al primer puesto, para cambiarlo, iremos a la pestaña “Workplaces” y haríamos click derecho en una zona vacía y seleccionando “Workplaces tab settings” para hacer visibles más puestos:



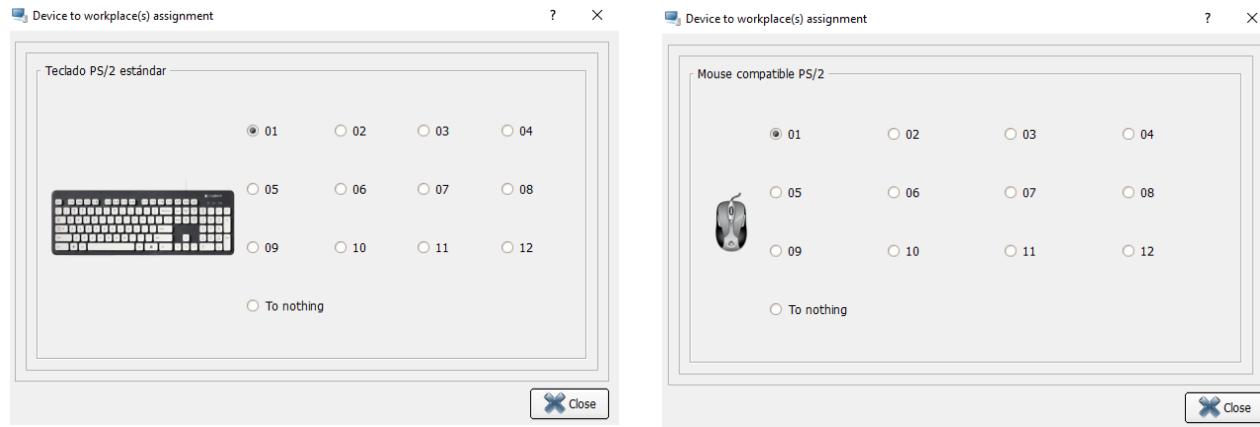
Al hacer click derecho en los dispositivos nos saldrá un desplegable y en “Workplaces Appointment...” podremos seleccionar a que puesto queremos asignarlo:



así hasta dejar todos configurados:

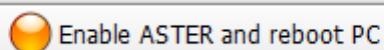


De igual manera haríamos lo mismo con todos los teclados y ratones conectados:



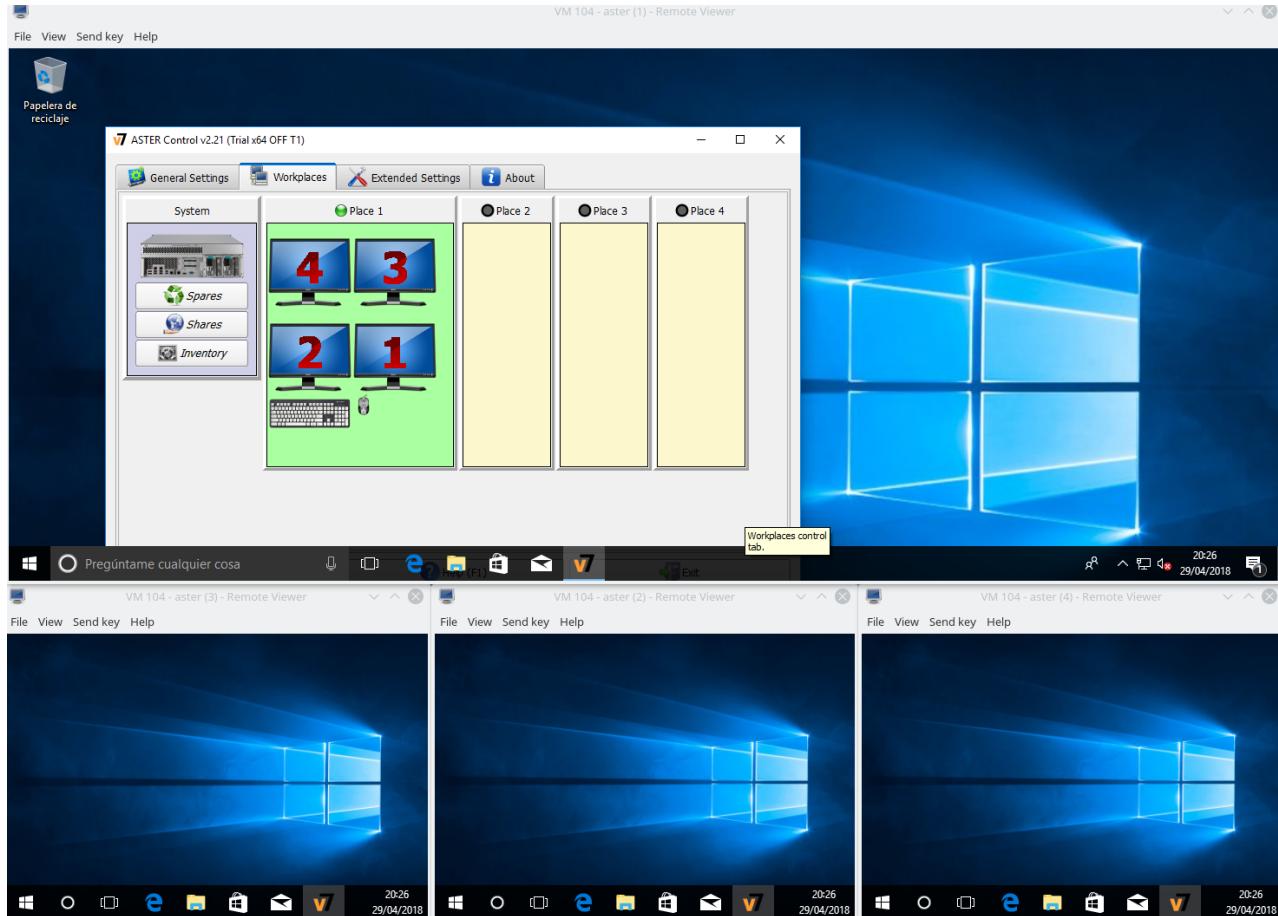
Todo esto también puede ser configurado con “drag and drop” en los dispositivos.

Una vez configurado volveríamos a la pestaña “General Settings” y Clickariamos en “Enable ASTER and reboot PC”

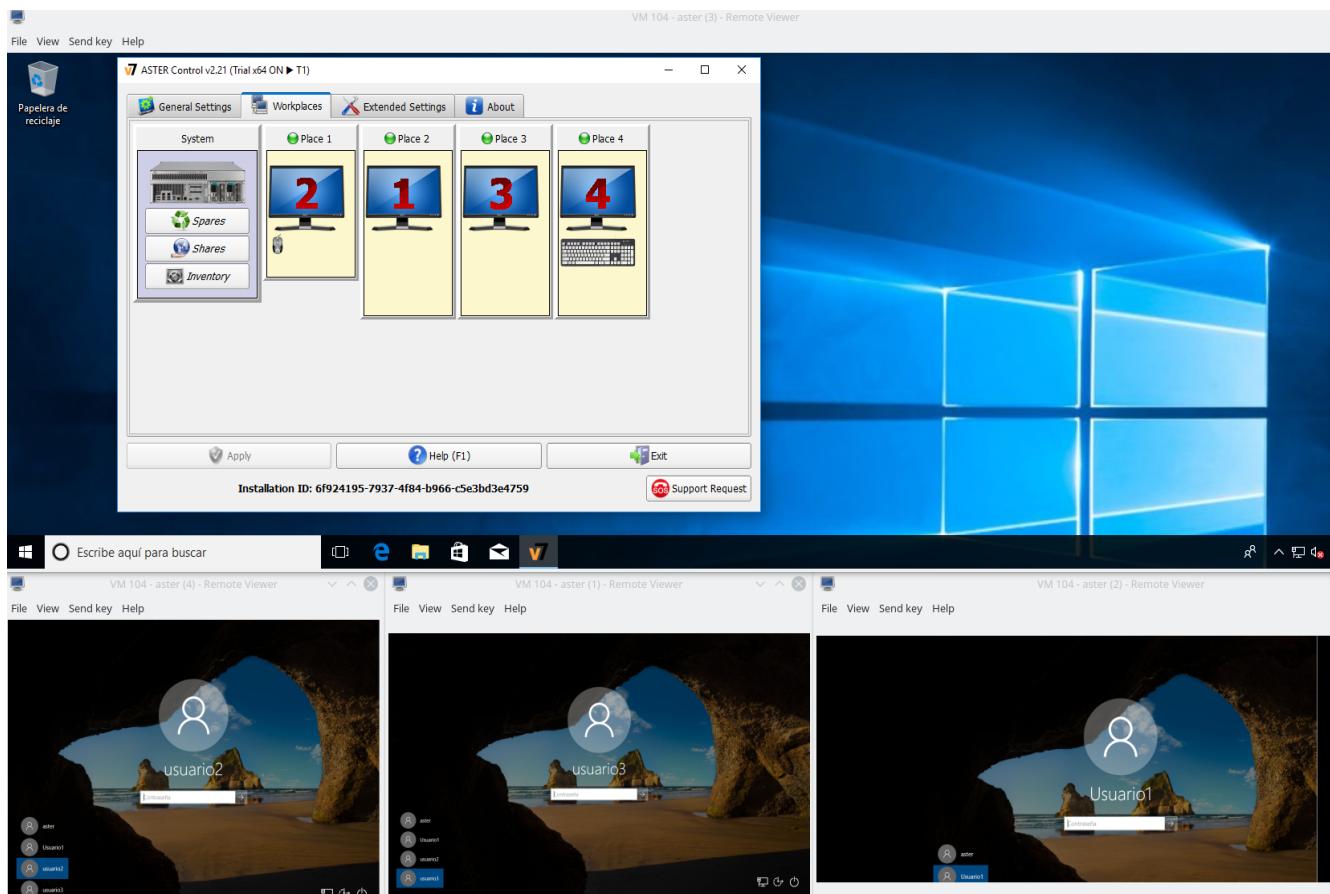


El equipo nos pedirá permisos de administrador y se reiniciara.

Así estaría por defecto, con todas las pantallas asignadas a la primera sesión:



y una vez aplicamos la configuración realizada tendremos las pantallas esperando a poder iniciar sesión:



Por el uso que le he podido dar a esta aplicación, cumple con creces puesto que permite ejecutar varios juegos a la vez, programas, navegadores, etc.

Con lo que si nos podemos encontrar problemas es con ciertos juegos o aplicaciones (Steam, League of Legends) que incorporan métodos para que solo se pueda ejecutar una instancia de aplicación por máquina física. Para intentar solucionar este problema podemos intentar utilizar programas que creen Sandbox de aplicaciones.

(Imagen de internet)



(Fotograma de video de internet)



Linux

Linux a día de hoy esta viviendo su mayor cambio en cuanto a servidores gráficos se refiere por lo que podemos encontrar diferentes maneras de hacerlo, como:

- Configurar el Xorg a mano.
- Comando loginctl de systemd para Wayland y Xorg.
- Comando elogind para Wayland y Xorg para sistemas sin systemd.
- Xephyr, xf86-video-nested o Xnest.

Una de las limitaciones de Linux es que de manera nativa solo se puede ejecutar un servidor gráfico , Xorg,(y por tanto, un solo puesto) por tarjeta gráfica (PCIe o USB) instalada en el ordenador. Existe un pequeño apañó, que es corriendo un Xorg anidado en otro Xorg (Xephyr), pero de hacerlo así nos encontraremos con falta de aceleración 3D... Lo cual puede ser solucionado con otro apañó llamado VirtualGl, que es una especie de cliente-servidor que se encarga de reenviar las peticiones gráficas a una máquina (local o externa) que si cuenta con aceleración gráfica.

Para Ubuntu, gracias al gobierno de Brasil, existe un driver para Xorg llamado xf86-video-nested que funciona como alternativa a Xephyr, resolviendo algunas de las deficiencias de este.

Nosotros nos vamos a basar en la solución nativa de puesto por gráfica.

Con la ayuda del comando “**loginctl seat-status**” podremos ver los dispositivos asignables a cada puesto (Por defecto todos los dispositivos están en seat0)

```
zeusinprox@proxmox1:~$ loginctl seat-status
seat0
  Sessions: *3
  Devices:
  /sys/devices/LNXSYSTM:00/LNXPWRBN:00/input/input1
    input:input1 "Power Button"
  /sys/devices/LNXSYSTM:00/LNXPWRBN:00/input/input0
    input:input0 "Power Button"
  /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/drm/card1
    [MASTER] drm:card1
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/drm/card1/card1-DVI-I-1
        [MASTER] drm:card1-DVI-I-1
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/drm/card1/card1-HDMI-A-2
        [MASTER] drm:card1-HDMI-A-2
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/drm/card1/card1-VGA-1
        [MASTER] drm:card1-VGA-1
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/drm/renderD129
        drm:renderD129
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/graphics/fb1
        [MASTER] graphics:fb1 "nouveaufb"
      /sys/devices/pci0000:00/0000:02.0/0000:03:00.0/sound/card1
        sound:card1 "NVidia"
          /sys/devices/pci0000:00/0000:02.0/0000:03:00.1/sound/card1/input22
            input:input22 "HDA NVidia HDMI/DP,pcm=3"
          /sys/devices/pci0000:00/0000:02.0/0000:03:00.1/sound/card1/input23
            input:input23 "HDA NVidia HDMI/DP,pcm=7"
          /sys/devices/pci0000:00/0000:02.0/0000:03:00.1/sound/card2
            sound:card2 "HDMI"
              /sys/devices/pci0000:00/0000:02.0/0000:05:00.1/sound/card2/input10
                input:input10 "HDA ATI HDMI HDMI/DP,pcm=9"
              /sys/devices/pci0000:00/0000:03.0/0000:05:00.1/sound/card2/input11
                input:input11 "HDA ATI HDMI HDMI/DP,pcm=10"
              /sys/devices/pci0000:00/0000:03.0/0000:05:00.1/sound/card2/input12
                input:input12 "HDA ATI HDMI HDMI/DP,pcm=11"
              /sys/devices/pci0000:00/0000:03.0/0000:05:00.1/sound/card2/input7
                input:input7 "HDA ATI HDMI HDMI/DP,pcm=3"
              /sys/devices/pci0000:00/0000:03.0/0000:05:00.1/sound/card2/input8
                input:input8 "HDA ATI HDMI HDMI/DP,pcm=7"
              /sys/devices/pci0000:00/0000:03.0/0000:05:00.1/sound/card2/input9
                input:input9 "HDA ATI HDMI HDMI/DP,pcm=8"
            /sys/devices/pci0000:00/0000:01:0/usb3
              usb:usb3
                /sys/devices/pci0000:00/0000:01:0/usb3/3-4/3-4:1.0/input/input20
                  input:input20 "Microsoft X-Box 360 pad"
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-4/3-4:1.0/input/input20/event20
                      input:event20
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-4/3-4:1.0/input/input20/jst0
                      input:jst0
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-7/1.0/0003:046D:C317.0001/input/input2
                      input:input2 "Logitech USB Multimedia Keyboard"
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-7/1.1/0003:046D:C317.0002/input/input3
                      input:input3 "Logitech USB Multimedia Keyboard"
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-8/1.0/0003:04D9:A067.0003/input/input4
                      input:input4 "Holtek USB Gaming Mouse"
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-8/1.1/0003:04D9:A067.0004/input/input5
                      input:input5 "Holtek USB Gaming Mouse"
                    /sys/devices/pci0000:00/0000:01:0/usb3/3-9
                      usb:3-9
                    /sys/devices/pci0000:00/0000:01:0/usb4
                      usb:usb4
                        /sys/devices/pci0000:00/0000:01:0/usb4/4-5
                          usb:4-5
                        /sys/devices/pci0000:00/0000:01:0/usb1
                          usb:usb1
                            /sys/devices/pci0000:00/0000:01:0/usb1/1-1
                              usb:1-1
                            /sys/devices/pci0000:00/0000:01:0/usb0
                              sound:card0 "PCH"
```

Para indicarle a Linux que hardware pertenece a cada puesto deberemos buscar las tarjetas gráficas que tengamos asignadas, junto con sus correspondientes dispositivos fb (framebuffer) y drm (aceleración gráficas) y dispositivos usb que queramos asignar.

En mi caso:

Nvidia:

```
/sys/devices/pci0000:00/0000:00:02.0/0000:03:00.0/drm/card1  
/sys/devices/pci0000:00/0000:00:02.0/0000:03:00.0/drm/renderD129  
/sys/devices/pci0000:00/0000:00:02.0/0000:03:00.0/graphics/fb1
```

AMD :

```
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/drm/card0  
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/drm/renderD128  
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/graphics/fb0  
/sys/devices/pci0000:00/0000:00:14.0/usb3/3-11:1.0/0003:046D:C077.0084
```

Una vez sepamos que dispositivos queremos asignar, podremos asignárselos a los seats con el comando “`logindt attach NºSeat ID`”.

```
logindt attach seat1 /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/drm/card0  
logindt attach seat1 /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/drm/renderD128  
logindt attach seat1 /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/graphics/fb0  
logindt attach seat1 /sys/devices/pci0000:00/0000:00:14.0/usb3/3-11:1.0/0003:046D:C077.0084
```

Una vez hecho esto, en mi caso, tuve que añadir una configuración básica al Xorg para indicarle cada pantalla a cada layout, ya que si no no había manera de hacerlo funcionar.

```
Section "ServerLayout"  
    Identifier      "seat1"  
    Screen          "Screen0" 0 0  
EndSection
```

[...]

```
Section "ServerLayout"  
    Identifier      "seat0"  
    screen          "Screen2" 0 0  
EndSection
```

[...]

En cualquier caso no es necesario hacer la asignación de teclado y ratón en Xorg ya que logindt se encargará de ello.

Una vez llegados a este punto debemos testear que ambos puestos inician sus respectivos servidores, para ello ejecutamos “`startx -- -layout seat0 -seat seat0`” y una vez comprobado que arranca, pararíamos el servidor grafico con CTRL + C y ejecutaríamos “`startx -- -layout seat1 -seat seat1`”. Si todo va bien, podemos pasar a configurar el gestor de sesiones.

Como gestor de sesiones utilizaremos Lightdm ya que otros gestores de sesiones como sddm no soportan multipuesto.

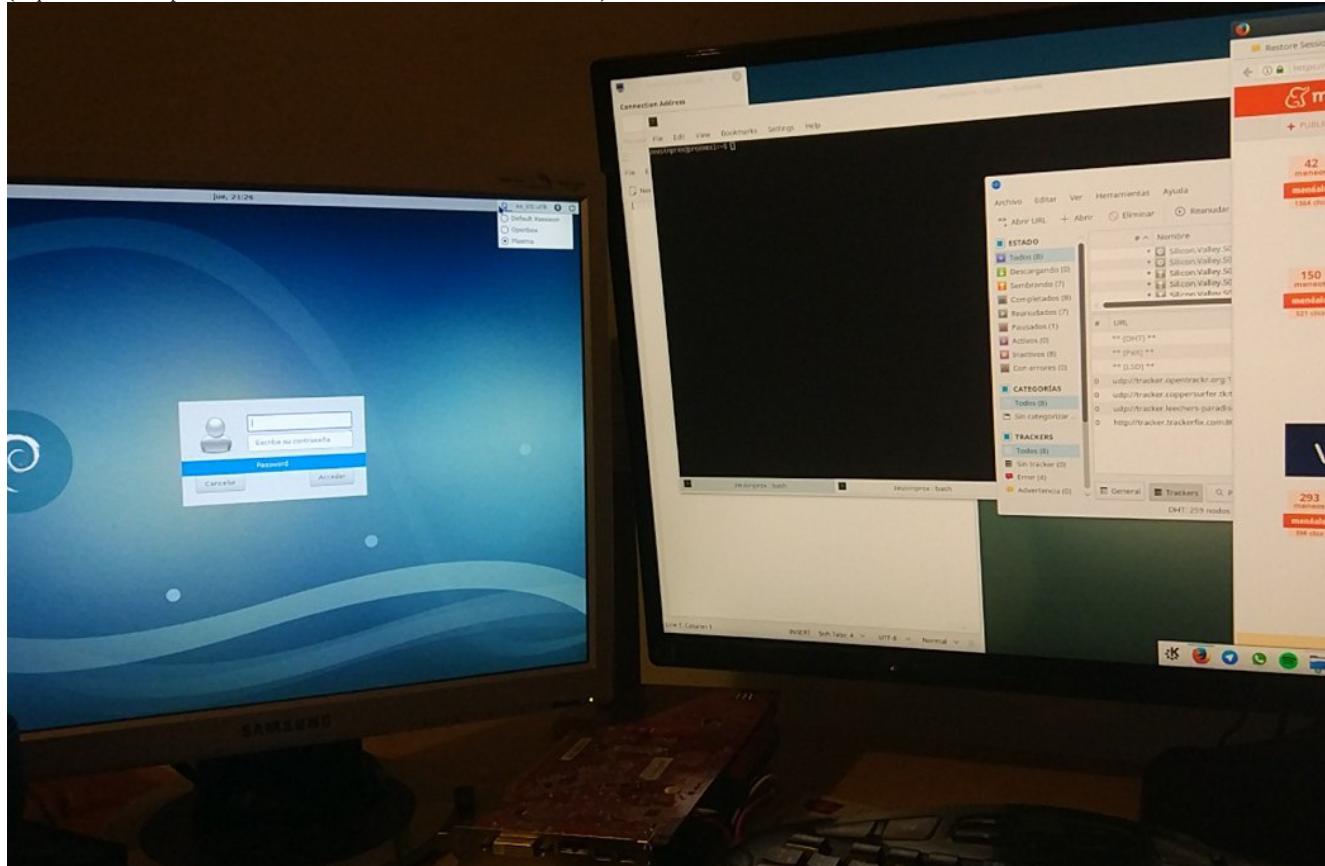
Para realizar la configuración del multipuesto deberemos de editar el fichero “`/etc/lightdm/lightdm.conf`” y añadir las siguientes lineas:

```
[Seat:seat1]
xserver-command=X
xserver-layout=seat1

[Seat:seat0]
xserver-command=X
xserver-layout=seat0
```

Una vez hecho esto podremos reiniciar el servicio de lightdm, pudiéndonos logar independientemente:

(Izquierda Seat1 esperando sesión, Derecha Seat0 con sesión iniciada)



Para comprobar sesiones iniciadas, podemos usar el comando “`loginctl`” que nos devolverá las sesiones, con sus usuarios y los puestos:

```
zeusinprox:bash
File Edit View Bookmarks Settings Help
zeusinprox@proxmox1:~$ su
Password:
root@proxmox1:/home/zeusinprox# nano /etc/lightdm/lightdm.conf
root@proxmox1:/home/zeusinprox# loginctl
  SESSION      UID USER          SEAT      TTY
    8        1004 zeusinprox    seat0
    2        1004 zeusinprox    seat0
   c3        115 lightdm       seat1
    9        1004 zeusinprox    seat0
   c7        115 lightdm       seat1
    6         0 root           seat0      /dev/tty1

6 sessions listed.
root@proxmox1:/home/zeusinprox#
```


Fuentes

Experiencia propia

https://pve.proxmox.com/wiki/Proxmox_VE_Kernel

<http://informatica.gonzalonazareno.org/proyectos/2015-16/Proyecto.%20Cl%C3%BAster%20de%20HA%20en%20Proxmox%204.pdf>

<https://wiki.gentoo.org/wiki/LXC>

https://pve.proxmox.com/wiki/Package_Repositories

<https://www.proxmox.com/en/proxmox-ve/pricing>

<https://pve.proxmox.com/wiki/Storage>

<http://vfio.blogspot.nl/>

<https://forum.proxmox.com/threads/ceph-server-feedback.17909/>

https://wiki.archlinux.org/index.php/Power_management#Suspend_and_hibernate

https://wiki.archlinux.org/index.php/Wake-on-LAN#On_the_same_LAN

https://wiki.archlinux.org/index.php/ZFS#Swap_volume

https://pve.proxmox.com/wiki/ZFS_on_Linux

https://pve.proxmox.com/wiki/Installation#advanced_lvm_options

<https://pve.proxmox.com/wiki/LVM2>

<https://forum.proxmox.com/threads/minfree-for-lvm-snapshots.28977/>

https://pve.proxmox.com/wiki/Installation:_Tips_and_Tricks#Optional:_Reverting_Thin-LVM_to_.22old.22_Behavior_of_.2Fvar.2Flib.2Fvz_.28Proxmox_4.2_and_later.29

https://pve.proxmox.com/wiki/Cluster_Manager

<http://docs.ceph.com/docs/jewel/rados/operations/add-or-rm-mons/>

<http://www.nicksherlock.com/2017/10/installing-macos-high-sierra-on-proxmox-5/>

<http://www.nicksherlock.com/2018/04/patch-ovmf-to-support-macos-in-proxmox-5-1/>

<https://wiki.archlinux.org/index.php/PRIME>

https://www.reddit.com/r/pcmasterace/comments/38sxgb/destroying_the_split_screen_argument_master_race

<https://wiki.ubuntu.com/Multiseat>

<http://code.lexarcana.com/posts/simple-multiseat-setup-on-fedora-17.html>

<https://streamable.com/69vdr>

https://wiki.archlinux.org/index.php/xorg_multiseat

<https://launchpad.net/~ubuntu-multiseat/+archive/ubuntu/xf86-video-nested>

<https://www.anchor.com.au/blog/2012/09/a-crash-course-in-ceph/>

<https://stackoverflow.com/questions/43286518/how-to-set-up-nested-wayland-desktop-environment-with-systemd-nspawn-container>