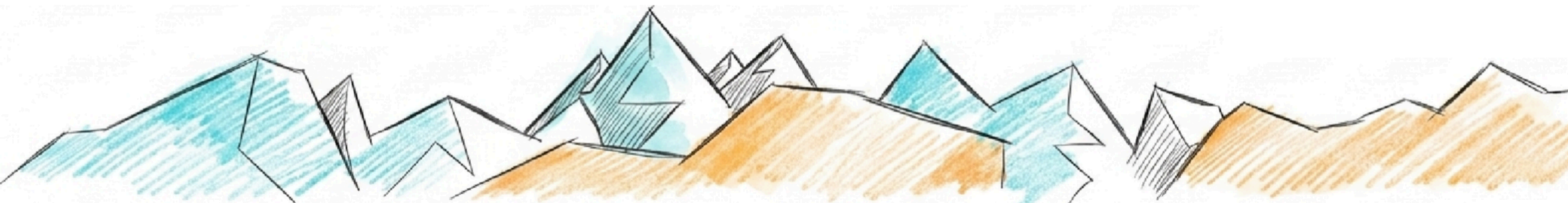


# ISARDVDI

¿UNA PLATAFORMA DE VDI PARA OPTIMIZAR EL USO DE IAS?

HPC ADMINTECH 08/05/2026

ALBERTO LARRAZ DALMASES

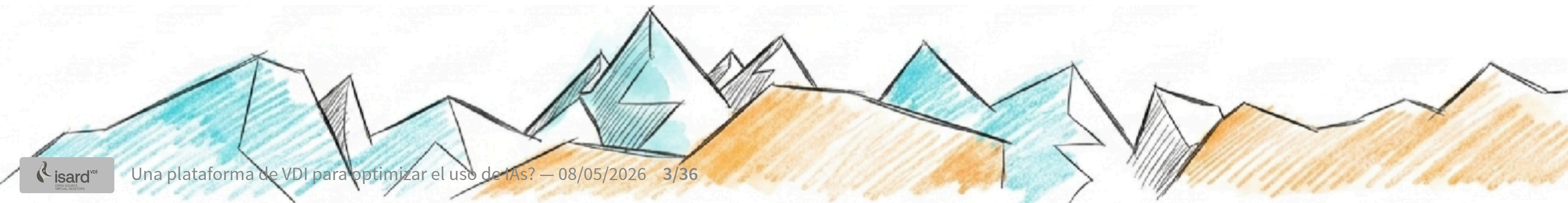


# QUIÉNES SOMOS

- Empresa de Barcelona que desarrolla el proyecto **IsardVDI**
- Desarrollo, sistemas, soporte y consultoría
- Experiencia en **soluciones libres**, sysadmin e integración de servicios
- Promover software libre, soberanía tecnológica y hacer **comunidad**

# 2. QUÉ OFRECEMOS

¿efecto demo?



# UN PRODUCTO DE SOFTWARE LIBRE KMO

- Crece con lo que necesitan nuestros **clientes**
- Free software como modelo de soberanía digital
- Comunidad activa y retroalimentación constante

# HACEMOS VDI... O QUIZÁS OTRA COSA

- La comunidad lo usa para usos que no habíamos previsto
- Un sitio donde **poder probar y desplegar VMs con autonomía y agilidad**
- Crear maquetas, desplegar escritorios, dar autonomía
- Sacarle partido a tu servidor y a tus GPUs

# CASOS DE USO REALES DE LOS USUARIOS

- Diseño de piezas en 3D, Windows con software licenciado
- Muy valorado por quien enseña **redes y ciberseguridad**
- **Laboratorios docentes** con escenarios reproducibles
- Conexión por vpn con robots industriales, redirección USBs locales a vms

# DE UN ISARDVDI EN MI PC...

- <https://isard.gitlab.io/isardvdi-docs/install/#quick-start>

```
git clone https://gitlab.com/isard/isardvdi
cd isardvdi
cp isardvdi.cfg.example isardvdi.cfg
./build.sh
docker compose pull
docker compose up -d
```

# ... A MULTITENANT EN UNA INSTALACIÓN

Da servicio al Departament d'Educació de Catalunya como servicio del CTTI:

- 71.887 escritorios persistentes activos
- 21.200 usuarios
- 5.847 plantillas (maquetas)
- 2.899 despliegues (aulas virtuales)
- 110.000 arranques y 13.000 escritorios diferentes por semana
- Picos de 1.500 escritorios concurrentes
- 135 institutos públicos de FP (multitenant)
- 99,58% disponibilidad del servicio
- Todo en Oracle Cloud

# ¿QUÉ MONTAMOS DEBAJO?

- **No somos brokers de VDI:** debajo no hay Proxmox, OpenNebula, OpenStack ni KubeVirt
- Todo con **24 contenedores Docker** (algunos privilegiados)
- El hipervisor es un Docker privilegiado con **libvirt/qemu**
- Gestión de redes con **OpenVSwitch**
- Proxys de visores: **RDP, VNC, Spice, websockets, Guacamole**
- Bastiones SSH, TCP 80/443, DNS configurables · WireGuard VPN
- **Grafana, Prometheus**

# ¿QUÉ MÁS NECESITAMOS PARA QUE FUNCIONE?

- WAF y ciberseguridad
- **Almacenamiento** con IOps y capacidad:
  - DRBD para réplica de bloques con RDMA 100 Gbps
  - Cache NVMe, ramdisk, deduplicación VDO
  - NFS
- HA: Pacemaker
- *Flavours* de Isard: web, hipervisor, monitorización, storage...
- Hardware: RAM, NVMe, cores rápidos, redes rápidas, muchos TB

# 3. QUÉ SERVICIOS OFRECEMOS

# DAAS — VDI COMO SERVICIO

- Sin licencias por usuario
- No cobramos por usuarios, escritorios ni horas
- Por **conurrencia contratada**:
  - vCPUs, memoria, memoria de GPU, espacio de disco para qcow incremental
- En TIER III en CPD Barcelona
- Disponible hoy: **8 TB RAM, servidores con RTX 6000** y podemos crecer

# ¿QUIERES UNA DEMO?

# ON-PREMISE... QUE PUEDE CRECER

- All in One (con GPUs A16 y RTX 6000 Blackwell)
- 2 o 3 nodos en hiperconvergencia en clúster
- NAS + hipervisores + hipervisores con GPU

Nos encargamos de gestionarlos y monitorizarlos

Te damos soporte

# SOPORTE Y SISTEMAS

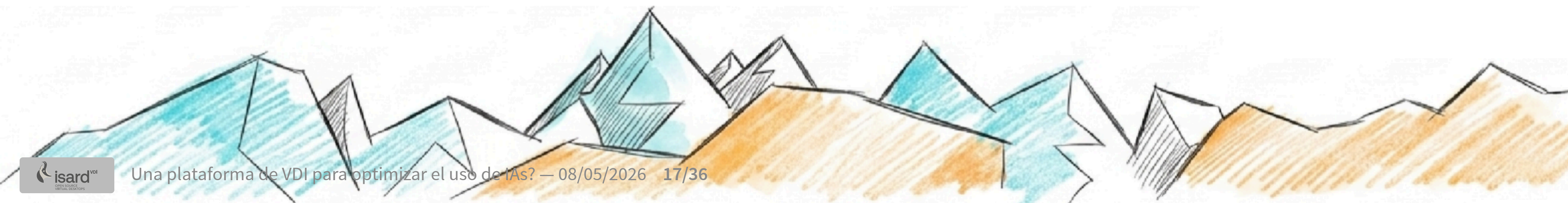
- Soporte N1, N2, N3
- Soporte para las distintas formas de implementar IsardVDI
- Implementaciones y continuidad de servicio en **infraestructura de terceros**

# Y ESTÁBAMOS EN ESTO...

CUANDO LLEGÓ LA IA

# LABORATORIOS DE IA?

## VMS CON GPU PASSTHROUGH



# VM + PASSTHROUGH VS CONTENEDORES

- Los contenedores (Docker, LXC, cgroups) **comparten kernel** con el host
- Un CVE de kernel = riesgo para todos los tenants del nodo
- 2025–2026: ya **cuatro CVEs documentados de runc** con escapes
- Para labs IA con drivers propietarios y kernels custom, "comparto kernel" y hay riesgos
- La VM aporta una frontera **enforced en hardware** (IOMMU + hypervisor)

# VM + PASSTHROUGH VS CONTENEDORES

Necesidad	cgroups / LXC / Docker	VM + passthrough
Driver NVIDIA distinto al del host	✗	✓
Kernel propio (RT, custom)	✗	✓
Snapshot de estado completo	✗	✓
Reset limpio de GPU tras crash	⚠ requiere reboot host	✓ por VM
Aislamiento ante CVE de kernel	✗	✓

# QUÉ OFRECE UNA TARJETA Y QUÉ PODEMOS APROVECHAR:

RTX PRO 6000 Blackwell Server Edition:

- **96 GB GDDR7 ECC** y 1,8 TB/s de ancho de banda en una sola tarjeta
- **24 064 CUDA cores**, 752 Tensor Cores 5ª gen, PCIe Gen 5
- Soporta los **tres modelos de virtualización**: passthrough, MIG y vGPU
- MIG: partir la tarjeta en hasta **4 instancias aisladas** por hardware
- **FP4 nativo**: ~2× throughput vs FP8 en inferencia compatible
- Modelos de hasta **70B parámetros** en una sola GPU sin sharding agresivo

# UN LABORATORIO, VARIAS VMS, CERO FRICCIÓN

- Necesito **levantar un modelo como API** para consumir tokens desde el código
- Quiero un **escritorio gráfico** para revisar imágenes, secuencias, vídeo
- Editar ficheros con **VSCo** sin salir del entorno, con los datos siempre dentro
- Una sola GPU física → **N VMs especializadas**: API, VDI gráfica, dev box
- **Datos y pesos** del modelo nunca salen del perímetro del laboratorio

# VDI SOBRE GPU VIRTUALIZADA: LA PIEZA QUE FALTA

Rol VM	Recursos GPU	Para qué
Inference API	MIG 1g.24gb / passthrough	vLLM, Ollama, TGI sirviendo tokens
VDI gráfica (vWS)	vGPU profile (p.ej. 16Q)	Imágenes, vídeo, ComfyUI
Dev workstation	vGPU pequeño (4Q–8Q)	VSCoode remoto, notebooks, debugging
Render / batch	MIG 2g.48gb	Stable Diffusion, generación masiva

*Server Edition admite hasta 48 vGPUs concurrentes y mezcla cómputo + gráficos en MIG*

# MULTI-GPU: CUÁNDO IMPORTA EL XML DE LIBVIRT

- Cuando una sola GPU no llega: **tensor parallelism, modelos grandes, training**
- Sin tuning, QEMU reparte vCPUs y memoria por nodos NUMA "como puede"
- Problema clásico: **GPU en NUMA node 1, vCPUs en node 0** → tráfico cross-socket
- En cargas memory-bound (LLM inference) se nota en **tokens/s reales**
- Tres palancas: `<cputune>`, `<numatune>` y `<cpu><numa>`
- Buenas prácticas validadas por **NVIDIA DGX KVM** y **Red Hat Virt Tuning Guide**

# TUNING XML LIBVIRT — REFERENCIA

```
<vcpu placement='static'>16</vcpu>
<cputune>
  <vcpupin vcpu='0' cpuset='8' />    <!-- vCPUs sobre cores del NUMA -->
  <vcpupin vcpu='1' cpuset='9' />    <!-- node donde está la GPU -->
  <emulatorpin cpuset='4-7' />      <!-- threads QEMU fuera de vCPUs -->
  <iothreadpin iothread='1' cpuset='4-5' />
</cputune>
<numatune>
  <memory mode='strict' nodeset='1' />
</numatune>
<cpu mode='host-passthrough'>
  <numa><cell id='0' cpus='0-15' memory='65536' unit='MiB' /></numa>
</cpu>
<memoryBacking><hugepages><page size='1' unit='G' /></hugepages></memoryBacking>
```

Tuneo	Qué resuelve
<code>vcpupin 1:1</code>	Cache hit ratio, evita migración de threads
<code>emulatorpin separado</code>	Threads QEMU/vhost no compiten con vCPUs
<code>numatune strict</code>	Memoria local al socket de la GPU
<code>1G hugepages</code>	~20% menos overhead TLB en cargas grandes
<code>host-passthrough</code>	El guest ve flags reales (AVX-512, AMX...)

# MICROVMS: EL PANORAMA 2026

- **Firecracker** (AWS, Rust): boot ~125 ms, <5 MiB overhead, **sin GPU passthrough**
- **Cloud Hypervisor** (Rust): GPU passthrough vía VFIO, hotplug, live migration
- **QEMU microvm**: machine type minimalista, **mismo ecosistema libvirt/virsh**
- Firecracker y Cloud Hypervisor son rápidos pero **rompen tu tooling**
- QEMU microvm = ~4× más rápido manteniendo libvirt, virsh, hooks, snapshots
- Para un laboratorio HPC, **menos boot time NO compensa** rehacer la operación

# QEMU MICROVM + PASSTHROUGH

```
<domain type='kvm'>
  <name>infer-microvm-01</name>
  <memory unit='GiB'>32</memory>
  <vcpu placement='static'>8</vcpu>
  <os>
    <type arch='x86_64' machine='microvm'>hvm</type>
    <kernel>/var/lib/libvirt/kernels/vmlinux</kernel>
    <cmdline>console=ttyS0 root=/dev/vda rw</cmdline>
  </os>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' />
      <source file='/var/lib/libvirt/images/infer.img' />
      <target dev='vda' bus='virtio' />
      <address type='virtio-mmio' />
    </disk>
    <interface type='bridge'>
      <source bridge='br0' />
      <model type='virtio' />
      <address type='virtio-mmio' />
    </interface>
    <hostdev mode='subsystem' type='pci' managed='yes'>
      <source><address domain='0x0000' bus='0x41' slot='0x00' function='0x0' /></source>
    </hostdev>
  </devices>
```

Devices vía *virtio-mmio* + *<hostdev>* para VFIO de la GPU. PVH boot, sin BIOS/ACPI, footprint mínimo.

# EFICIENCIA Y COSTES: GPU COMPARTIDA CON PRIORIDAD

- Equipo de investigación quiere comprar 2× RTX PRO 6000 para su proyecto
- Propuesta: las pinchan en **nuestro chasis denso** (5U, hasta 8–10 GPUs)
- Garantía de **prioridad de uso** vía VMs dedicadas con passthrough
- Cuando no las usen: **MIG/vGPU** dejan slots disponibles para otros equipos
- 8× RTX PRO 6000 = **768 GB VRAM** agregada, hasta **32 instancias MIG**
- Resultado: el grupo paga 2 tarjetas, la organización rentabiliza 8

# MODELOS PEQUEÑOS: EL CASO DEL HARDWARE RECICLADO

- No todo en el laboratorio es un LLM de 70B parámetros
- **Whisper** (audio): tiny ~1 GB, base ~1 GB, small ~2 GB, medium ~5 GB
- **DeepSeek-OCR** y otros OCR de PDFs: 4–8 GB en BF16
- **Imagen médica** (segmentación, clasificación): 2–6 GB
- Modelos **destilados / cuantizados** (Distil-Whisper, Phi-3, TinyLlama): 2–4 GB
- Estos modelos **desperdician** una RTX PRO 6000 entera

# RECICLAR TARJETAS VIEJAS VÍA VGPU

GPU	VRAM	Perfiles vGPU típicos	Uso ideal
V100	16 / 32 GB HBM2	4Q, 8Q, 16Q, 32Q	Inferencia clásica, embeddings, training ligero
A40	48 GB GDDR6	4Q, 8Q, 12Q, 24Q, 48Q	Mix visualización + AI, render + inferencia
L40S	48 GB GDDR6	4Q, 8Q, 12Q, 24Q, 48Q	Inferencia masiva, transformer engine FP8
H100	80 GB HBM3	5gb, 10gb, 20gb, 40gb (MIG)	LLMs grandes, training serio, NVLink

*Una V100 partida en 4 vGPUs de 4 GB = 4 servicios de Whisper-medium concurrentes, sin tirar hardware*

# PLURALIDAD DE FABRICANTES: EL PASSTHROUGH TE SALVA

- Si AMD lanzase una nueva tarjeta con > 120GB
- Probarlas implica drivers ROCm, kernel modules nuevos, posibles conflictos
- Con passthrough: una VM, sus drivers, su problema — el host ni se entera
- Mismo planteamiento para mezclar Hopper, Ada, Blackwell, AMD, Intel Gaudi
- Cada VM ve solo "sus" GPUs → adiós incompatibilidades cruzadas
- Roadmap de adopción: pilotos sin tocar el host de producción

# VIRTIOFS: COMPARTIR EL HOST CON LA VM, SIN RED

- Ejemplo: cada VM se descarga el mismo modelo de Hugging Face (10–50 GB)
- Almacenamiento nvme local compartido entre vms
- **virtiofs**: sistema de ficheros compartido basado en **FUSE + virtio**
- Se ejecuta vía `virtiofsd` en el host; la VM ve un dispositivo virtio como un FS local
- **Semántica de FS local** → soporta DAX, page cache compartida con el host
- Combina con un `qcow2` de SO/drivers + directorios montados del host

# VIRTIOFS — EJEMPLO Y REPARTO DEL CACHE

```
<memoryBacking>  
  <source type='memfd' />  
  <access mode='shared' />  
</memoryBacking>  
<devices>  
  <filesystem type='mount' accessmode='passthrough'>  
    <driver type='virtiofs' queue='1024' />  
    <source dir='/srv/hf-models' />  
    <target dir='hf-cache' />  
    <readonly />  
  </filesystem>  
</devices>
```

# VIRTIOFS

```
# Dentro de cada VM:  
mount -t virtiofs hf-cache /root/.cache/huggingface
```

Caso	Solución
Pesos HF compartidos por N VMs	virtiofs <b>read-only</b> sobre /srv/hf-models
Datasets temporales por VM	virtiofs read-write o disco propio
SO + drivers + CUDA	Dentro del <b>qcow2</b> de cada VM (snapshotable)
Varias VMs pueden montar el mismo <source> simultáneamente	Como NFS, pero con <b>rendimiento de FS local</b>

# RECURSOS

- Repositorio — <https://gitlab.com/isard/isardvdi/>
- Pruébalo — <https://isard.gitlab.io/isardvdi-docs/install/#quick-start>
- Demos y soluciones — [info@isardvdi.com](mailto:info@isardvdi.com)
- Redes:
  - Twitter — [https://twitter.com/isard\\_vdi](https://twitter.com/isard_vdi)
  - Mastodon — <https://fosstodon.org/@isard>

# GRACIAS

## ¿PREGUNTAS?

*IsardVDI · 8 de mayo de 2026*