KVM/DR

WHITEPAPER | CROSS HYPERVISOR DR SERIES | PART 2 OF 2

How KVMDR Delivers Cross Hypervisor DR

One recovery experience across Proxmox VE, oVirt, OLVM and RHV

KVMDR AI Ltd | kvmdr.ai | November 2026

Abstract

Part 1 showed that Proxmox VE and oVirt share the same KVM virtual machines but are managed in very different ways. This paper explains how KVMDR bridges that gap: a platform neutral replica, platform specific adapters at each end, and the recovery features operators expect from enterprise DR, including point in time ransomware recovery, non disruptive testing and flexible failback.

1. The design principle: neutral in the middle

Most DR tools are built around one hypervisor and bolt others on later. KVMDR was built for KVM from the start, and keeps the protected data in a form that belongs to neither platform. Platform knowledge sits only at the edges, in adapters that know how to read from a source and how to build a VM on a target.

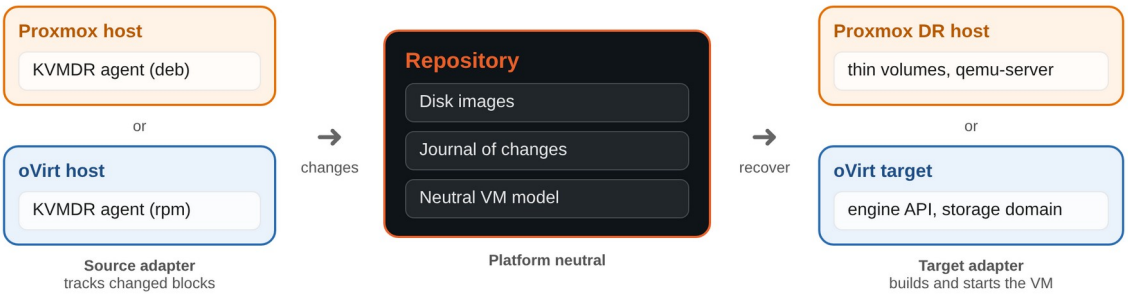

Figure 1. KVMDR cross hypervisor data path

- **Source adapter.** The KVMDR host agent, packaged for both Proxmox (deb) and oVirt (rpm), tracks changed blocks on each VM disk and streams them to the repository. Nothing is installed inside the guest.
- **Platform neutral repository.** Disks are kept as plain images with a journal of changes, plus a neutral description of the VM: CPU, memory, disks, NICs and boot settings.
- **Target adapter.** On Proxmox, KVMDR builds the VM directly on the DR host. On oVirt, it registers the VM and its disks through the engine in the chosen storage domain.

How KVMDR plugs into each stack

Part 1 showed that both platforms end in QEMU, while everything above QEMU differs. KVMDR captures changes at the QEMU layer, which is common to both, and talks to each platform's own management layer only where it has to: to build and start VMs on the target.

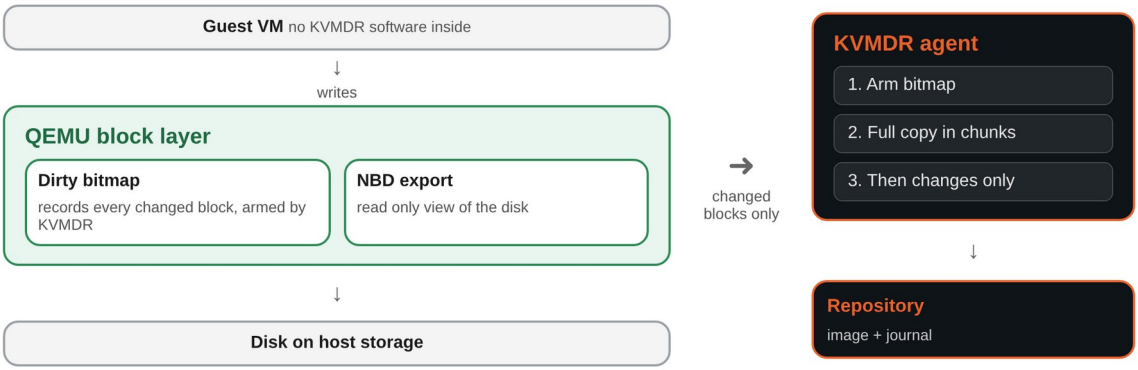

Figure 2. Change capture at the QEMU layer

1. Initial sync: KVMDR arms a dirty bitmap on each disk, so every write from that moment is recorded, then reads the whole disk through a read only NBD export in fixed size chunks. A sync that is interrupted resumes from the last completed chunk.

2. Continuous replication: KVMDR reads only the blocks marked in the bitmap and writes them to the repository journal, then starts a fresh bitmap. Because the bitmap is armed before the full copy begins, no write can fall into the gap between the two.
3. Recovery: the target adapter turns the repository image and journal into a disk at the chosen point in time, then hands it to the target platform.

Integration point	Proxmox VE	oVirt, OLVM, RHV
Talk to QEMU	Directly, over the QMP socket that Proxmox itself uses	Alongside vdsm and libvirt on each host
Read VM settings	VM configuration in /etc/pve	Engine REST API
Create DR disks	Thin volumes on the DR host storage	Disk images in the target storage domain, through the engine
Create and start VMs	qemu-server on the DR host	Engine REST API, which tells vdsm and libvirt
Containers	Block filter on LVM thin volumes	Not applicable

> **Why capture at QEMU**
>
> QEMU dirty bitmaps and NBD exports are standard QEMU features, the same ones Proxmox Backup Server and oVirt incremental backup rely on. Working at this layer gives KVMDR one capture method for both platforms, with no software inside the guest and no dependency on either management layer for replication itself.

2. How a cross hypervisor recovery runs

1. The operator picks a VM or consistency group and a point in time: the latest point, or a clean point before a ransomware event.
2. KVMDR checks the target is ready: storage, network mapping and, on oVirt, the engine.
3. The target adapter builds the VM from the neutral model, translating names and networks for that platform.
4. Disks are presented from the chosen point in time.
5. Network rules are applied: new IP addresses (Re IP), or the original MAC address kept.
6. VMs in a consistency group start together, from the same point in time.

> **Same steps, either direction**
>
> Proxmox to oVirt and oVirt to Proxmox use the same workflow and the same screens. The adapters absorb the platform differences described in Part 1.

3. Handling the differences for you

Platform difference (Part 1)	What KVMDR does
oVirt needs its engine	Talks to the engine API to create and start VMs, and checks it as part of recovery

Platform difference (Part 1)	What KVMDR does
Disks live in storage domains	Cross hypervisor profile selects the target cluster and storage domain
Different network models	Network map from source bridges to target networks, with Re IP or MAC kept
Different naming rules	Names translated automatically (for example underscores to hyphens on Proxmox)
Containers only on Proxmox	Containers protected Proxmox to Proxmox with the same features as VMs
Different test mechanics	Simulation adapts per platform: space efficient on Proxmox, baseline on oVirt

4. Sizing the DR site

Part 1 noted that a DR test on an oVirt target needs more space than on Proxmox. The reason is how each platform lets KVMDR make a temporary, writable copy of a VM for testing.

Why Proxmox needs almost no extra space

On a Proxmox target, the DR disks are thin volumes on the DR host that KVMDR controls directly. For a Simulation, KVMDR takes an instant thin snapshot and boots the test VM from it. Only blocks the test VM writes take new space, and they are discarded when the test ends. Replication keeps running to the original disks throughout.

Why oVirt needs a test baseline

On an oVirt target, the DR disks are images inside a storage domain, and only the engine can snapshot or clone them. As Part 1 showed, an oVirt snapshot adds a qcow2 overlay to the disk's volume chain and redirects writes to it. That chain belongs to one VM in the engine database. If KVMDR snapshotted the DR disk for a test, replication and the test VM would be competing for the same chain. Cloning the disk into a new, independent image through the engine is a full copy.

KVMDR therefore keeps a separate test baseline in the storage domain: an independent copy that the test VM boots from, while replication carries on into the DR disks untouched. The baseline is the same size as the protected disks.

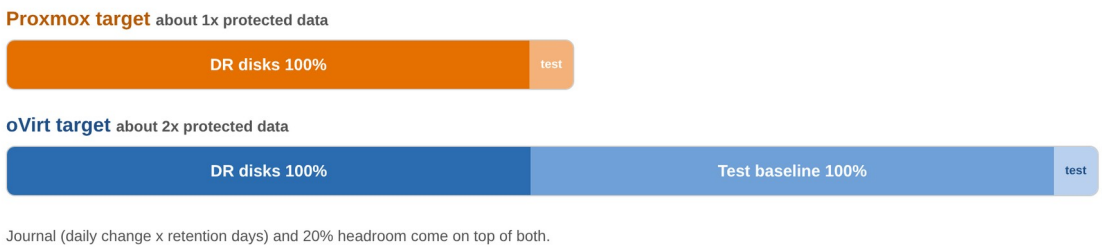

Figure 3. DR target space during a Simulation

Sizing guide

Item	Proxmox target	oVirt target
DR disks	1 x protected data	1 x protected data
Test baseline	None	1 x protected data
Test writes	Small, temporary	Small, temporary
Recovery history (journal)	Daily change x retention days	Daily change x retention days
Headroom	At least 20%	At least 20%

> **Example: 2 TB of protected VMs**
>
> Proxmox target: 2 TB of DR disks plus journal plus 20% headroom. oVirt target: 2 TB of DR disks plus a 2 TB test baseline plus journal plus 20% headroom. Plan roughly twice the DR disk space on oVirt. The journal size depends on how much data changes each day and how long history is kept.

5. Enterprise DR features, across platforms

- **Near synchronous replication.** Changes are captured continuously, with recovery points as fresh as one second.
- **Point in time recovery.** Roll back to the moment before ransomware hit, not last night's backup.
- **AI ransomware detection.** Abnormal write patterns are flagged as they happen, and the last clean point is highlighted.
- **Simulation.** Full recovery tests on an isolated network while protection keeps running, with no impact on production.
- **Consistency groups.** Linked VMs, such as an application and its database, recover to the same point in time.
- **Flexible failback.** Return home by sending only the changes made since failover, or recover the entire VM or container when required.
- **Highly available management.** An active and standby manager pair, so DR control survives the loss of a site.
- **No agents in the guest.** Protection runs from the hypervisor host; guests only need the standard QEMU guest agent for file restore.

6. Why KVMDR is good at this

- **Built for KVM, not adapted from VMware.** KVMDR was designed around KVM and QEMU from day one, which is why both platforms map cleanly onto one neutral model.
- **Both platforms, both directions, one product.** One licence and one workflow cover Proxmox VE, oVirt, OLVM and RHV, as source or target.
- **Recovery, not just copies.** Point in time recovery, Re IP, consistency groups and Simulation make a replica usable in a real disaster.
- **Ransomware aware.** Detection and clean point recovery are part of the replication path, not a separate product.

- **Enterprise DR experience.** The team behind KVMDR has decades of experience building and selling enterprise disaster recovery, and brings that discipline to KVM.

7. Typical uses

- **Mixed estates.** oVirt or OLVM in the main data centre, Proxmox at a secondary site, each protecting the other.
- **Moving platforms.** Teams moving from oVirt or RHV to Proxmox can replicate, test with Simulation, then cut over, with the source kept for rollback.
- **Lower cost DR sites.** Protect an oVirt or OLVM production estate to a Proxmox DR site on commodity hardware.

8. Conclusion

Proxmox VE and oVirt are managed in very different ways, but the virtual machines they run are the same. KVMDR uses that common ground: it keeps protected data neutral, puts platform knowledge in adapters at each end, and wraps it in enterprise DR features. The result is one recovery experience across every KVM platform, in either direction.

See it working: start a 7 day free trial or book a demo at kvmdr.ai.

About KVMDR

KVMDR AI Ltd is a UK company building disaster recovery for KVM based hypervisors: Proxmox VE, oVirt, Oracle Linux Virtualization Manager and Red Hat Virtualization. KVMDR combines near synchronous replication, point in time ransomware recovery, non disruptive DR testing and AI driven threat detection in one platform. Start a 7 day free trial at kvmdr.ai.