KVM/DR

WHITEPAPER | CROSS HYPERVISOR DR SERIES | PART 1 OF 2

Proxmox VE and oVirt: Same Engine, Different Worlds

What cross hypervisor disaster recovery can and cannot do, and why recovery behaves differently on each platform

KVMDR AI Ltd | kvmdr.ai | November 2026

Abstract

Proxmox VE and oVirt both run virtual machines on KVM and QEMU, so a VM can move between them with its data intact. Yet the two platforms are managed in very different ways, and those differences decide what disaster recovery can achieve on each side. This paper compares their architectures, sets out what works in each direction, explains why recovery to an oVirt target needs more planning than recovery to Proxmox, and gives a practical best practice checklist. Part 2 describes how KVMDR delivers cross hypervisor DR.

1. Why cross hypervisor DR matters

Many organisations now run more than one KVM platform. Some are moving from oVirt or Red Hat Virtualization to Proxmox VE. Others keep oVirt or Oracle Linux Virtualization Manager in the data centre and use Proxmox at the edge or in a secondary site. In both cases the same question comes up: can one platform protect the other?

The answer is yes, because the virtual machine itself is the same on both sides: a KVM guest with QEMU devices and VirtIO drivers. What is not the same is everything around the VM, and that is where planning matters.

> **The short version**
>
> Proxmox is a set of hosts that each manage themselves. oVirt is a central engine that manages its hosts. Recovery to Proxmox depends only on the DR host. Recovery to oVirt also depends on the oVirt engine being up.

2. Two architectures

Proxmox VE: every host is in charge of itself

Each Proxmox node has its own API, web interface and command line tools. Nodes can form a cluster that shares configuration, but there is no central manager. Any node can create, start and stop virtual machines and containers on its own, using storage it can reach: LVM thin, ZFS, directories, Ceph and more.

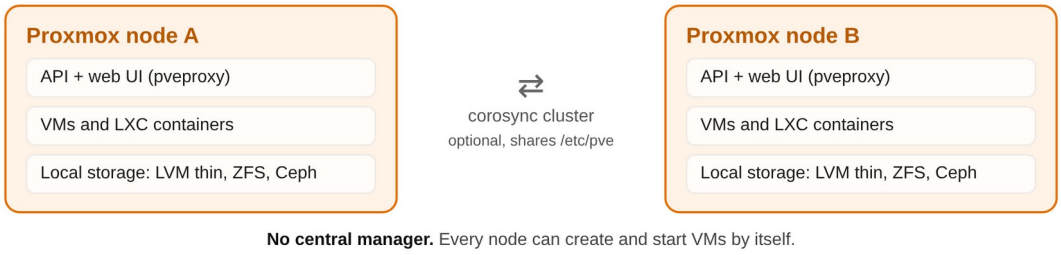
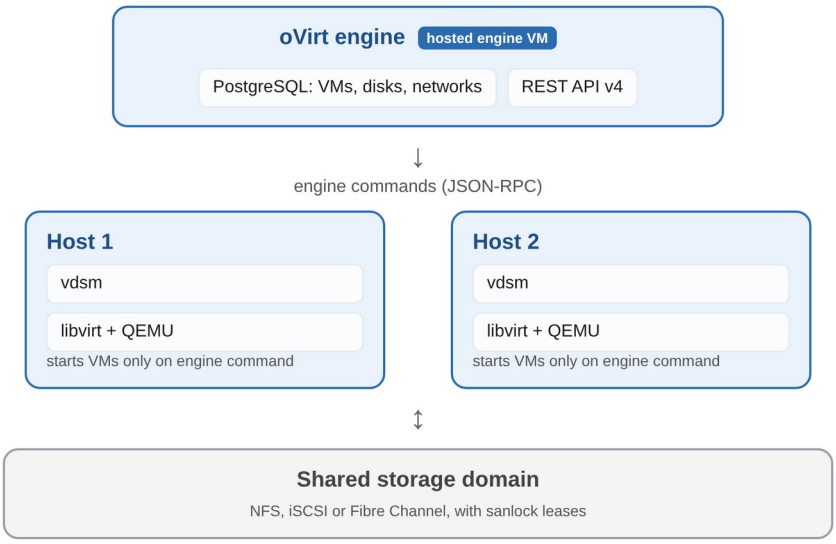


Figure 1. Proxmox VE: every node manages itself

oVirt, OLVM and RHV: a central engine runs the show

In oVirt the engine holds a database of every VM, disk, network and storage domain. Hosts run the vdsm service and start VMs only when the engine tells them to. Disks are images inside shared storage domains on NFS, iSCSI or Fibre Channel. The engine itself usually runs as a hosted engine VM on the same hosts.

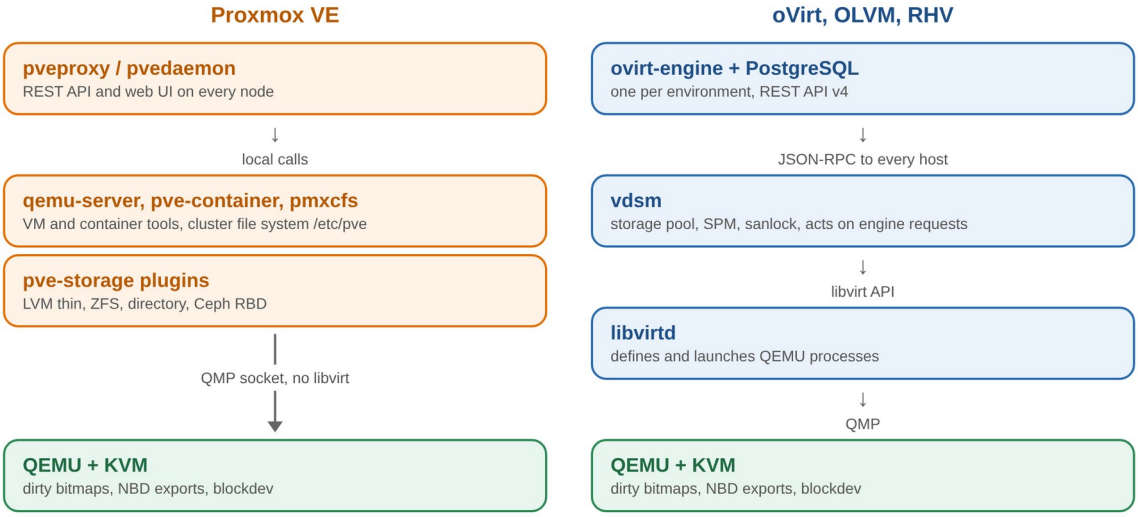


VMs exist only in the engine database. No engine, no new VMs.

Figure 2. oVirt, OLVM and RHV: a central engine manages the hosts

Inside the software stack

Both platforms end at the same place: a QEMU process running a KVM guest. The layers above QEMU are where they part ways, and those layers decide how replication, snapshots and recovery can work.



Same bottom layer, different control above it. Proxmox drives QEMU directly; oVirt goes engine, vdsm, libvirt, QEMU.

Figure 3. Software stack, top to bottom

Layer	Proxmox VE	oVirt, OLVM, RHV
Management API	pveproxy and pvedaemon on every node	ovirt-engine, one per environment
VM definition	Text file in /etc/pve/qemu-server, replicated by pmxcfs over corosync	Rows in the engine PostgreSQL database

Layer	Proxmox VE	oVirt, OLVM, RHV
Host agent	qemu-server and pve-container, part of each node	vdsm, which only acts on engine requests
QEMU control	Direct, over the QMP socket. Proxmox does not use libvirt	Through libvirt, which vdsm drives
Storage layer	pve-storage plugins: LVM thin, ZFS, directory, Ceph RBD	Storage pool of storage domains, coordinated by the SPM host
Locking	pmxcfs cluster locks	sanlock leases on shared storage

How storage works underneath

Proxmox hands each VM disk to a storage plugin. On LVM thin, a disk is a device mapper thin volume; on ZFS, a zvol. Snapshots are taken by the storage itself (a dm thin snapshot or a ZFS snapshot), are near instant, and produce a separate block device that can be used on its own.

oVirt groups storage into a storage pool of storage domains. On a file domain (NFS), each disk is an image directory holding one or more volume files, each with a metadata file and a sanlock lease. On a block domain (iSCSI, FC), each storage domain is an LVM volume group and each volume a logical volume. A snapshot in oVirt adds a new qcow2 volume on top of the existing one, forming a chain. Creating volumes changes shared metadata, so it is done by one elected host, the Storage Pool Manager (SPM), on the engine's instruction.

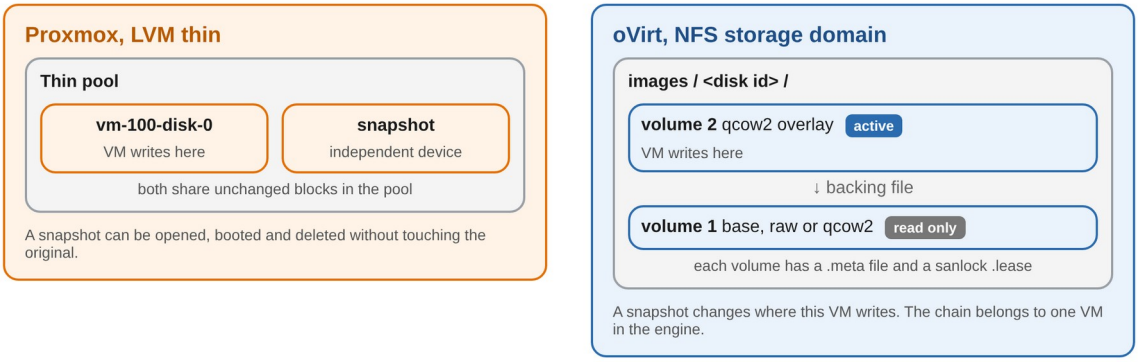


Figure 4. A disk with one snapshot

This difference matters for disaster recovery. On Proxmox a snapshot is an independent device that a DR tool can open, boot and throw away without touching the original. On oVirt a snapshot changes which volume the VM writes to, and the chain belongs to one VM in the engine database.

Where the engine itself lives

In most oVirt estates the engine runs as a hosted engine VM on the same hosts. Two services on each host, ovirt-ha-agent and ovirt-ha-broker, keep it running, using a sanlock lockspace on shared storage. If the broker stops or the lockspace cannot be acquired, the hosted engine stays down, and with it every engine operation, including starting recovered VMs.

Aspect	Proxmox VE	oVirt, OLVM, RHV
Management	Each node has its own API and UI	One central engine for all hosts
Where a VM is defined	Configuration file on the node	Engine database
Who starts a VM	Any node, by itself	The engine, through vdsm
Where disks live	Volumes on node storage	Images in a shared storage domain
Containers	Yes (LXC)	No
VM names	Must be valid host names, no underscores	Underscores allowed
Component to protect	None, each node is independent	The engine

3. What works in each direction

Capability	Proxmox to oVirt	oVirt to Proxmox
VM replication	Yes	Yes
Containers (LXC)	No, oVirt has no containers. Protect them Proxmox to Proxmox	Not applicable
Target storage	oVirt file storage domains (NFS)	Thin provisioned storage on the DR host
Non disruptive DR test	Yes, with extra space for a test baseline	Yes, space efficient
Failover	Yes, with the oVirt engine running	Yes, needs only the DR host
Re IP on failover	Yes	Yes
VM name on target	Kept	Underscores become hyphens
File restore into a running VM	Yes, with the QEMU guest agent	Yes, with the QEMU guest agent

Current limits

- **Containers stay on Proxmox.** oVirt has no container support, so LXC containers are protected Proxmox to Proxmox.
- **File storage domains only on oVirt.** iSCSI and Fibre Channel storage domains are not yet supported as a target.
- **No engine, no failover to oVirt.** oVirt hosts cannot start new VMs without the engine.

4. Why recovery is different on an oVirt target

The engine must be up

On Proxmox, recovery talks straight to the DR host, which can create and start VMs by itself. On oVirt, every VM and disk must be registered with the engine, and only the engine can start a VM. If the hosted engine is stopped or unhealthy, recovery waits. This is the single biggest difference between the two targets.

Disks must live in a storage domain

On Proxmox, DR disks sit on the DR host and can be created and swapped quickly. On oVirt, disks are images inside a storage domain, created through the engine. The target storage domain has to be chosen up front and sized for the protected data.

Testing needs more space on oVirt

A DR test on Proxmox needs almost no extra space. On oVirt a test baseline copy is kept, so the storage domain must hold the protected data plus that baseline. Part 2 of this series explains why, and shows how to size each target.

Names and networks are translated

Proxmox VM names cannot contain underscores. Networks are modelled differently: Proxmox uses Linux bridges such as vmbr0, oVirt uses logical networks and vNIC profiles. Every source network needs a clear target network.

5. Best practice checklist

Before protecting

1. Decide which platform is primary for each VM and keep it consistent.
2. If oVirt is the target, run the engine as a highly available hosted engine on at least two hosts.
3. If oVirt is the target, create an NFS data domain sized for protected data plus the test baseline.
4. Keep containers on Proxmox to Proxmox protection.
5. Use VirtIO disks and NICs, and install the QEMU guest agent in every VM.
6. Avoid underscores in names of VMs that may run on Proxmox.

Networks

1. Map every source bridge or network to a target network.
2. Decide per VM whether to change IP addresses on failover or keep them.

Testing

1. Test each VM after protecting it, then at least monthly.
2. Check that the VM boots, gets the expected address, applications respond and the guest agent answers.
3. If oVirt is the target, include engine health in every test.

Failover and failback

1. Trigger failover only on a real event; never schedule it.
2. Confirm users can reach recovered VMs on the target network.
3. Schedule failback in a maintenance window.

6. Conclusion

Cross hypervisor DR between Proxmox VE and oVirt is practical today, because both run the same KVM virtual machines. Success depends on respecting how each platform is

managed: Proxmox recovers on its own, oVirt recovers through its engine. Plan storage, networks and engine availability up front, test regularly, and either platform can protect the other.

Part 2 of this series explains how KVMDR handles these differences, so cross hypervisor recovery works the same way from the operator's point of view.

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.