BEST PRACTICES GUIDE

Protecting Containers on Proxmox VE and oVirt

How to protect LXC containers and Kubernetes clusters on Proxmox VE, oVirt and Oracle Linux Virtualization Manager, and recover them cleanly at a second site.

EXECUTIVE SUMMARY

Containers do not fail over the way virtual machines do. The word covers two different workloads, and each needs its own protection plan.

- **System containers (LXC).** A full Linux system that shares the host kernel. It is persistent by default and is run and patched like a small server. Proxmox VE runs these natively, next to its VMs.
- **Application containers (Docker, Kubernetes).** One application per image, replaced rather than patched, with its state kept outside the container in volumes.

The platforms differ too. Proxmox VE runs LXC containers beside virtual machines. oVirt and Oracle Linux Virtualization Manager (OLVM) manage virtual machines only, so on those platforms every container runs inside a VM. This guide sets out how to protect both kinds of container on all three platforms.

How to protect containers in five steps

1. **Map where each workload keeps its state.** State you have not mapped is state you are not protecting.
2. **Choose the protection layer for each workload.** Protection at the VM or container level recovers whole systems quickly. Kubernetes level backups add history and let you restore a single namespace. Most clusters need both.
3. **Group dependent workloads.** An application and its database belong in one group that is recovered to one point in time.
4. **Prepare the recovery site.** Containers depend on their hosts more than VMs do, and Kubernetes depends on its IP addresses.
5. **Test in an isolated network, on a schedule.** Recovery that has never been exercised cannot be relied on.

START HERE

Know where the state lives

Every recovery starts with knowing what has to come back. Map each workload against this table before choosing how to protect it.

WORKLOAD	WHERE ITS STATE LIVES	WHAT TO PROTECT
LXC container	Root disk, volume mount points and the container configuration	All volumes and the configuration, together
LXC bind or device mount	A host path outside the Proxmox storage library	The host path, separately, or move the data onto a volume
Docker in a VM	The VM disks: images, volumes and compose files	The VM
Kubernetes cluster	etcd (cluster state), the node VM disks and persistent volumes	Cluster state and volume data, recovered to the same point
Volumes on external storage	NFS, Ceph or SAN storage outside the node VMs	The storage itself, or a Kubernetes level backup

LXC ON PROXMOX VE

Protecting system containers

1. Keep container volumes on snapshot capable storage

Snapshot mode backups need every included volume on storage with snapshot support, such as LVM thin, ZFS or Ceph RBD. Without it, the remaining backup modes suspend the container or stop it for the length of the backup.

2. Do not keep data you must recover on bind or device mounts

Proxmox never backs these up, because their content sits outside its storage library. Additional volume mount points are also left out by default unless their Backup option is enabled.

3. Protect the configuration with the data

The container configuration holds the mount points, ID mapping, features, network settings and resource limits. Disks recovered without it are hard to bring back as the same container.

4. Match the hosts at the recovery site

Containers rely on host settings that VMs do not:

- the same `/etc/subuid` and `/etc/subgid` ranges for containers with custom ID mapping, or restores fail with ownership errors
- the same storage IDs, bridges and VLANs
- the same Proxmox VE major version

5. Plan for the shared kernel

A container runs on its host's kernel, so a host upgrade changes what the container runs on. Proxmox VE 9 removed cgroup v1 entirely, and containers with old systemd releases, for example CentOS 7 or Ubuntu 16.04, do not start on it. Move those workloads into VMs before you rely on failing them over.

6. Know what built in replication covers

Proxmox storage replication works only on local ZFS, copies to other nodes of the same cluster on a schedule (every 15 minutes by default) and keeps only the latest state. High availability can restart a guest from that copy,

losing changes made since the last sync. It shortens recovery from a failed node, but it is not a second site and not a backup history. `pve-zsync` can send ZFS data to hosts outside the cluster.

7. Keep copies with history at another site

Back up containers to a backup server, such as Proxmox Backup Server, and sync those backups to a second server at another site. A deleted or encrypted container can then go back to a clean point.

8. Recover dependent workloads together

When an application container depends on a database in another container or VM, protect them as one group and recover them to the same point in time.

APPLICATION CONTAINERS

Docker and Kubernetes on Proxmox VE

Run application containers inside VMs.

Proxmox recommends a QEMU VM for application containers such as Docker images. OCI image support for containers, added in Proxmox VE 9.1, is still a technology preview, and Docker inside LXC is possible but fragile across updates.

Inside a VM, the container runtime, images and volumes all live on the VM disks, so the VM is the unit you protect and recover. For Kubernetes, protect the node VMs as described in the Kubernetes section of this guide.

OVIRT AND OLVM

Containers on oVirt and OLVM

oVirt and OLVM manage virtual machines only. Plans to run containers beside VMs date from the oVirt 4.0 era and survive only as historical design notes; libvirt LXC was ruled out after Red Hat deprecated it. On these platforms Kubernetes always runs in VMs.

On OLVM, Oracle Cloud Native Environment includes an `olvm` provider that builds Kubernetes clusters from OLVM VMs through the Kubernetes Cluster API, together with an oVirt CSI driver for persistent volumes on OLVM disks. Node VMs take their addresses from a fixed range rather than DHCP, with a virtual IP in front of the control plane, so those addresses must carry over to the recovery site.

oVirt's own disaster recovery comes in two forms, both requiring replicated storage:

- **Active/active.** One stretch cluster across two sites, with VMs moving to hosts at the second site when the first fails. It must meet latency and network requirements.
- **Active/passive.** Two separate environments, with failover and failback run manually through Ansible. Local storage domains are not supported.

Two points affect Kubernetes on this stack. The oVirt CSI driver documented for OpenShift has no snapshot support, so check the version you run before relying on CSI snapshots. And OpenShift dropped support for RHV as a platform in release 4.14, so OpenShift clusters still on RHV are legacy and need a migration plan.

KUBERNETES

Kubernetes on either platform

A Kubernetes cluster has three things to protect: cluster state in etcd, the node VMs, and application data in persistent volumes.

1. Protect the cluster as one unit

Put every control plane and worker VM in one group and recover them all to the same point. Members of etcd cannot be mixed from different points: a running cluster refuses a member restored from an older copy, and an etcd restore brings every member back from the same snapshot.

2. Keep node IP addresses unchanged

Node addresses are written into cluster certificates and the etcd configuration. The kubeadm tool has no supported way to change them, and the documented fix after a change is to restore the original addresses. Give nodes static addresses or DHCP reservations, carry the same subnets to the recovery site, and do not change node addresses during failover. A DNS name for the control plane endpoint helps clients but does not remove this dependency.

3. Back up etcd on its own schedule

All Kubernetes objects live in etcd, including Secrets. Take periodic snapshots, encrypt them and store them outside the cluster. A restore creates a new logical etcd cluster, with every member restored from the same snapshot file.

4. Back up volume data separately

An etcd snapshot holds no persistent volume contents. Use a Kubernetes aware backup tool, such as Velero, for namespace level history: CSI snapshots where the driver supports them, copied off the cluster's own storage so they survive a site loss, and file system backup where it does not. File system backup reads live files over time, so it is less consistent than a snapshot, and it cannot back up hostPath volumes.

5. Know where your CSI driver puts volumes

The Proxmox CSI plugin creates persistent volumes as VM disks and moves them between node VMs when pods move within a Proxmox node; it also supports snapshots and ZFS replication. The oVirt CSI driver backs volumes with oVirt disks. With either driver, protection at the VM level must follow disks that are added and removed, or those volumes need a Kubernetes level backup.

6. Make stateful applications consistent

Crash consistent copies suit stateless workloads. For databases, use the database's own replication or backups, or quiesce related volumes together: snapshots taken one volume at a time do not share a common point.

7. Keep the cluster definition in Git

Manifests, Helm values and cluster configuration under version control let you rebuild a cluster from scratch and restore its data into it. This is the fallback when the cluster itself cannot be recovered.

8. Pause Cluster API during a failover

Where a management cluster owns the node VMs, as with Oracle Cloud Native Environment on OLVM, include it in the plan and pause the workload cluster (`spec.paused` on its Cluster object) before a failover or test. While a cluster is paused, its health checks do not remediate machines, so recovered nodes are not replaced. Unpause once the nodes are healthy.

RECOVERY

Recovery order and testing

Recovery order

1. **Shared services:** DNS, NTP, directory services, storage and the networks the workloads expect.
2. **Stateful dependencies:** databases and other services the applications rely on.
3. **Kubernetes control plane VMs.** Wait for etcd to form quorum before starting workers.
4. **Worker VMs,** then the applications.
5. **Checks from the outside in:** cluster health, then workloads, then the application itself.

Testing

- **Use an isolated network.** Recovered systems keep their original addresses, so a test beside running production causes address conflicts. Run tests in a network cut off from production.
- **Test on a schedule and after major changes,** such as a Proxmox or oVirt upgrade, a Kubernetes upgrade or a new CSI driver.
- **Measure every test** against the recovery point and recovery time objectives set for each workload tier.

> *A recovery plan that has not been tested is a guess.*

CHECKLIST

Container protection checklist

- ✓ Every workload's state is mapped: root disks, mount points, bind mounts and persistent volumes
- ✓ Container volumes are on snapshot capable storage
- ✓ No data you must recover sits on bind or device mounts
- ✓ Recovery hosts match: subuid and subgid ranges, storage IDs, bridges, VLANs and Proxmox version
- ✓ No container depends on cgroup v1
- ✓ Application containers run inside VMs
- ✓ Backups with history are kept at another site
- ✓ Kubernetes node VMs are protected and recovered as one group
- ✓ Node IP addresses carry over to the recovery site
- ✓ etcd snapshots are taken, encrypted and stored outside the cluster
- ✓ Persistent volume data is backed up off the cluster's own storage
- ✓ The cluster definition is kept in Git
- ✓ Cluster API is paused during failover where a management cluster is used
- ✓ Recovery is tested in an isolated network on a schedule

© 2026 KVMDR AI Limited · Registered in England & Wales · Company No. 17289429 · This paper is provided for informational purposes.