Netframe

OPERATIONS

Backup Strategies for Netframe

Planning principles for backup on Netframe: scheduling and retention, OS-consistent quiescence, offsite copies, and recovery testing

May 2026 · Version 1.0

Contents

Overview

Netframe Backup is the supported backup component of the Netframe platform. It provides incremental VM-level backups with OS-consistent quiescence, VM-level instant restore, file-level restore, and cross-cluster restore.

This document is a short, practical overview of how to plan backups on Netframe. The audience is operators standing up a new Netframe environment, or revisiting backup configuration on an existing one. It is intentionally light on detail; the goal is to make sure the right decisions are in place before you configure schedules and policies.

The five things that matter most are workload categorisation, backup frequency and retention, OS-consistent quiescence, off-site copies, and recovery testing. Each is covered briefly below.

Workload categorisation

Not every VM needs the same backup posture. Categorising workloads by criticality before you build the schedule lets the backup configuration match the operational value of each workload.

A useful starting framework:

Critical. Customer-facing services, transactional databases, anything whose unavailability has direct business impact. These workloads warrant the most frequent backups, the longest retention, and the most thorough restore testing.

Important. Internal applications, reporting platforms, secondary services. Less frequent backups than critical, shorter retention.

Standard. General-purpose VMs: file servers, internal tooling, development environments. Daily backup is usually sufficient.

Ephemeral. Build agents, cache nodes, anything reconstructible from configuration. Light or no backup; the source of truth lives elsewhere.

Categorising forces a conversation with workload owners that surfaces useful information about which systems matter most. Once categorisation is done, the rest of the backup configuration follows mechanically.

Backup frequency and retention

For each category, decide the backup frequency (how often a fresh backup is taken) and the retention (how long old backups are kept).

A defensible starting point:

- **Critical:** backup every 4 hours, retain daily for 30 days, weekly for 12 weeks, monthly for 12 months.
- **Important:** backup daily, retain daily for 14 days, weekly for 8 weeks, monthly for 6 months.
- **Standard:** backup daily, retain daily for 14 days, weekly for 4 weeks.
- **Ephemeral:** weekly or as-needed.

These numbers are a starting point; the right values depend on the data change rate, regulatory requirements, and the storage budget. Netframe Backup reports actual storage consumption per workload, so the retention can be tuned empirically after a few weeks of operation.

OS-consistent quiescence

A backup of a running VM that captures the disk state mid-write is not a useful backup; it may not restore cleanly, and any restore that does work may leave the filesystem in an inconsistent state. Quiescing the guest before the snapshot is taken solves this.

Netframe Backup supports OS-consistent quiescence on both Linux and Windows guests. The mechanism varies by guest:

- **Linux guests:** the QEMU Guest Agent triggers filesystem flush and freeze before the snapshot is taken.
- **Windows guests:** Volume Shadow Copy Service (VSS) is invoked to quiesce the filesystem.

For most workloads, OS-consistent quiescence is the right default. It produces backups that are crash-consistent at the filesystem layer, which is sufficient for the great majority of restore scenarios.

For workloads that require deeper consistency (databases, transactional applications), application-level coordination should be layered on top: take application-level dumps or backups inside the guest, scheduled to run shortly before the platform-level snapshot, and let the platform snapshot capture both the live database state and the most recent application backup.

Off-site copies

The traditional 3-2-1 backup pattern (three copies of the data, on two different media, with one copy off-site) predates virtualisation by decades and still applies.

In a Netframe environment:

- **First copy:** the production VM disks on the primary storage appliance.
- **Second copy:** local backup, written to a backup repository on the same site as production. Typically on a separate appliance or at least a separate filesystem from production storage.
- **Third copy:** off-site backup, replicated from the local backup repository to a remote location. This is the copy that protects against site-level events.

For organisations without a second physical site, replicating the local backup repository to an S3-compatible object storage target in a public cloud is a reasonable pattern.

The off-site copy is the one most often broken when needed. Monitoring off-site replication lag as a first-class metric is the most useful operational discipline for catching this early.

Recovery testing

The single most important operational discipline in backup is regular recovery testing. A backup that has not been restored has not been validated.

A pragmatic cadence:

- **Critical workloads:** restore validation monthly, full recovery exercise quarterly.
- **Important workloads:** restore validation quarterly.
- **Standard workloads:** restore validation semi-annually.

The exercise should restore the workload to a test environment, validate that it functions, and then dispose of the restored copy. The point is to discover problems (missing dependencies, stale documentation, backup corruption) under controlled conditions rather than during an actual incident.

A test that finds and fixes a problem is more valuable than a test that simply passes. The objective is to discover gaps, not to confirm success.

Common failure modes

A short list of failure modes we see most often, with the prevention for each.

Backup taken but never tested. Restore on a defined cadence to a test environment. Validate that the restored VM boots and the application functions.

Backup retention too short. Define retention by category. Document it. Audit periodically.

Quiescence not configured. Use OS-consistent quiescence for all workloads above the ephemeral category. The marginal overhead is small.

Off-site copy lagged or broken. Monitor off-site replication lag as a first-class metric. Alert on lag exceeding the configured frequency.

Single point of failure in the backup path. The backup repository is itself a system that can fail. It should have its own redundancy, monitoring, and replication.

No documented runbook. When the restore is needed, the on-call engineer needs to know what to do. A short runbook per category of restore (VM-level, file-level, cross-cluster) keeps recovery predictable.

Capacity planning

Backup storage capacity is a recurring planning task. The dominant drivers are: the total data set being protected, the change rate (which drives incremental backup size), the retention policy (which sets how many incrementals are kept), and the deduplication ratio achievable by the backup repository.

A practical starting point for sizing:

- Full backups equal to the protected data set, kept according to long-term retention.
- Daily incrementals at 5–10% of the data set, kept for the short-term retention window.
- 25–50% headroom for growth and ad-hoc snapshots.

Netframe Backup reports actual deduplication ratios achieved against your data, which is the more accurate way to refine the estimate after the first weeks of operation.

Next steps

If you are designing backup for a new Netframe deployment, the practical next steps are:

1. Categorise workloads and assign backup frequency and retention to each category.
2. Configure Netframe Backup with the corresponding schedules. Start with the free tier (up to 15 VMs) for evaluation; move to a licensed configuration when you're ready to scale out and bring in support.
3. Schedule the first restore exercise within 30 days of go-live. Document the runbook for each restore type.

A platform that backs up reliably and restores quickly is a platform that makes operational incidents recoverable. The discipline behind the configuration is what makes that the default outcome.