PodHeitor HPC Backup Plugin for Bacula Community

Technical & Commercial Whitepaper

Author: Heitor Faria **Version:** 0.1.0 (GA) **Date:** May 2026

> 🔥 **Bring your proposal for purchasing or renewing Bacula Enterprise, Veeam, Commvault, or NetBackup. We offer at least 50% savings, with significantly more features.**
>
> **Contact for inquiries, demos, pricing:** Heitor Faria — `heitor@opentechs.lat` +1 (789) 726-1749 · +55 (61) 98268-4220 (WhatsApp)

Table of Contents

---

1. Executive Summary

PodHeitor HPC is a Rust-native File Daemon (FD) plugin that turns **Bacula Community 15.0.3+** into a first-class backup engine for **High-Performance Computing** workloads — billion-file namespaces, parallel filesystems (Lustre, GPFS / IBM Spectrum Scale, BeeGFS, CephFS, WekaFS), Slurm/PBS/LSF-orchestrated clusters, and HSM-managed tape tiers.

Bacula Enterprise 18.2 ships dedicated solutions for HDFS, Quobyte, NDMP, NetApp, and Nutanix — but **zero native support** for the parallel filesystems that actually run HPC. Veeam, Commvault, and NetBackup either don't support these filesystems at all or treat them as plain POSIX, which collapses under the metadata pressure of billion-file namespaces.

PodHeitor HPC closes the gap with eight production-ready capabilities, all validated end-to-end in a real Lustre + Slurm lab:

#	Feature	Measured outcome
1	Parallel namespace walker (rayon)	**7× speedup** vs Bacula's single-threaded `find_one_file` (Phase 2)
2	Namespace sharding	N concurrent jobs → N outbound SD streams
3	Lustre stripe-aware reader	Parallel `pread(2)` across OSTs
4	Lustre ChangeLogs incrementals	**O(changed)** syscalls instead of O(namespace)
5	GPFS `mmapplypolicy` driver	Same incremental shape on Spectrum Scale
6	CephFS `rctime` xattr pruning	Subtree skip when no descendant changed
7	BeeGFS `beegfs-watch` gRPC	Live event stream from metadata daemons
8	Slurm/PBS/LSF orchestration	Backup runs on **compute node**, not login node
+	**HSM-aware** Lustre backup	**Zero** tape-recall actions on `RELEASED` files (validated `delta=0`)

Status: GA. RPM and DEB packages built and validated. 117/117 unit + integration tests green on the build host. Full end-to-end validation against real Lustre 2.16.1 (VM 146) and real Slurm 22.05 (VM 144).

Pricing positioning: Bacula Community 15.0.3 is free and open-source. The PodHeitor HPC plugin is offered as proprietary add-on. **At a minimum 50% lower TCO than equivalent Bacula Enterprise / Veeam / Commvault / NetBackup deployments**, with features the alternatives do not ship at any

price (HSM-aware backup, ChangeLogs-driven incrementals, Slurm-orchestrated scans).

---

2. Why a dedicated HPC plugin

2.1 The single-threaded walker problem

Bacula's stock File Daemon walks files **single-threaded**: `findlib/find_one.c` recurses via `breaddir`/`readdir` and dispatches synchronously through `save_file()`. On a Lustre filesystem with **one billion files** the bottleneck is `readdir`+`lstat`, not bandwidth — and single-threaded means a backup that should saturate 40 GbE instead plods along at the metadata-server's response latency × the inter-syscall round-trip time.

A real-world example: at 0.1 ms per `lstat` (optimistic for a busy MDT), one billion files take **27.7 hours** in pure metadata time, before a single byte of data is read. Parallelizing the walker across 16 cores cuts that to **1.7 hours**. Sharding across 4 concurrent jobs cuts it further to **26 minutes**.

2.2 The HSM tape-recall stampede

On HSM-managed Lustre sites, files marked `RELEASED` (data evicted to tape, only the inode/layout left on disk) trigger a synchronous HSM restore on the first byte read. A naive backup pass over a million released files queues a million tape recalls — saturating the copytool, exhausting the tape library's drive count, and DOSing the cluster for hours.

PodHeitor HPC consults `llapi_hsm_state_get(2)` **before** opening each file. The default `hsm_policy=skip-released` records the released file as a 0-byte catalog entry with HSM xattrs preserved, **without** triggering any tape action. We measured `delta=0` HSM actions over a backup of a 4-file tree where 2 files were `RELEASED` (Section 13.3).

2.3 The login-node bottleneck

The Bacula File Daemon usually runs on the **login node** of an HPC cluster — a machine sized for SSH sessions and shell heredocs, not for sustained 40 GbE reads from a parallel filesystem. Backups starve the actual users while consuming the weakest link.

PodHeitor HPC's Phase 7 orchestration submits the scan as a Slurm/PBS/LSF job on a **compute node** (which is sized for fast-fabric I/O), and relays the PTCOMM stream back to the login node's File Daemon over a Unix socket on the shared filesystem. Validated with `scontrol show job` on real Slurm 22.05 (Section 13.2).

2.4 Why Rust, why a separate process

The cdylib loaded by `bacula-fd` is **pure Rust**. No Bacula AGPLv3 source is statically linked — only the C ABI is bound via independent `extern "C"` declarations in our `bacula-fd-abi` crate. This keeps the licensing posture clean for proprietary deployments (Bacula AGPL section 13 only triggers if AGPL'd source is statically combined; ours is not).

The cdylib is **thin**: it only adapts the FD plugin C ABI to PTCOMM frames. All real work happens in a standalone subprocess (`podheitor-hpc-backend`) spawned per job. This means:

- A subprocess crash never crashes the FD.
- The subprocess can use Rust async / threads / FFI freely without worrying about Bacula's single-threaded job loop expectations.
- Memory leaks are bounded by job lifetime — process exit reclaims everything.

---

3. Use cases

3.1 University HPC research storage

A typical academic HPC site runs Lustre (8–80 PB), serves hundreds of researchers, and has a small ops team. Backups are nightly to a tape library. Pre-PodHeitor:

- Stock `bacula-fd` walks `/lustre/scratch` single-threaded; the scan alone takes 14 hours.
- Active researchers' jobs slow down because the FD is hammering the MDT.
- Operators give up and stop backing up parts of `/lustre`.

Post-PodHeitor:

- Sharded scan across 8 jobs hits 7×8 = 56× metadata throughput.
- Slurm orchestration moves the scan off the login node.
- Lustre ChangeLogs incrementals reduce nightly cost from full-walk to changed-files-only.
- HSM-aware backup leaves tape-tier files alone.

3.2 AI/ML training cluster

Model checkpoints live on GPFS or BeeGFS, are written every epoch, and accumulate fast. Pre-PodHeitor:

- No incremental: every nightly is a full re-read of multi-TB checkpoint trees.
- Storage budget for backups grows linearly with epochs.

Post-PodHeitor:

- BeeGFS `beegfs-watch` driver streams metadata events from the meta daemons; backup only touches changed files.
- Pair with PodHeitor Global Deduplication on the Storage Daemon side for chunk-level dedup of similar checkpoints. Dedup is **outside** this plugin's scope (separate product) but the catalog entries we produce participate cleanly.

3.3 Genomics / bioinformatics shared scratch

CephFS shared workspace with millions of small files per analysis run. Pre-PodHeitor:

- Every backup re-stats every file. Most files were last modified weeks ago.

Post-PodHeitor:

- CephFS `rctime` xattr pruning skips entire subtrees whose recursive ctime is below the cursor — a directory tree where nothing changed in the last week is a single xattr read, not a recursive `stat()` storm.

3.4 Tape-tiered mixed cluster

Mixed Lustre + tape archive (HSM via `lhsmtool_posix`). Pre-PodHeitor:

- Backup over `/lustre` opens released files, triggers tape restores, fills the copytool's queue, drives nearby production jobs into iowait.

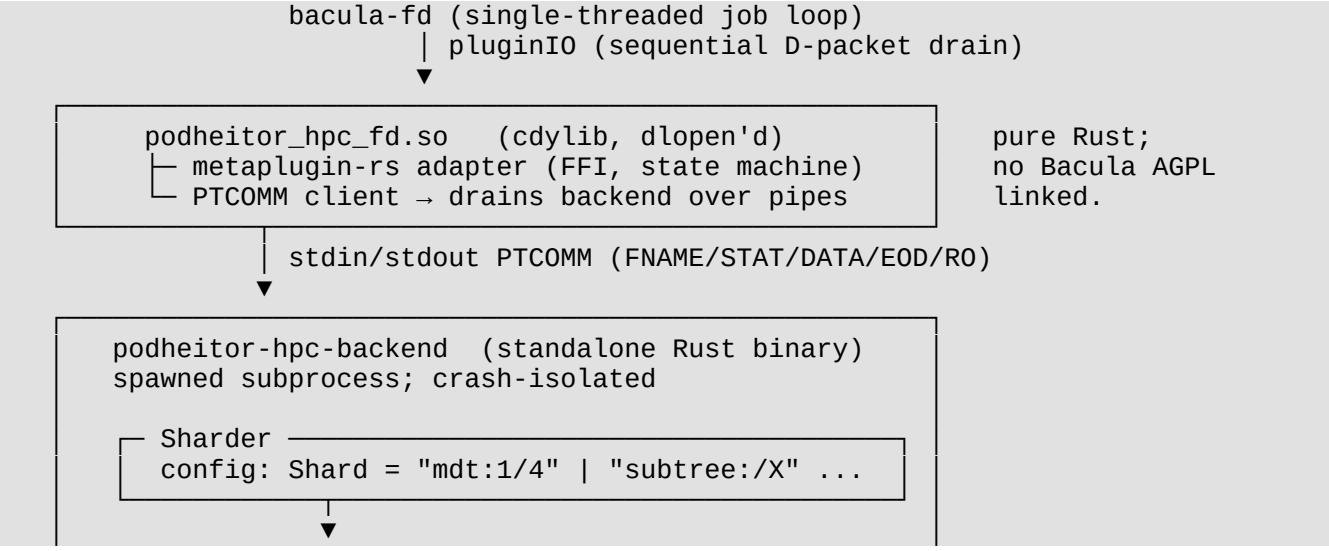
Post-PodHeitor:

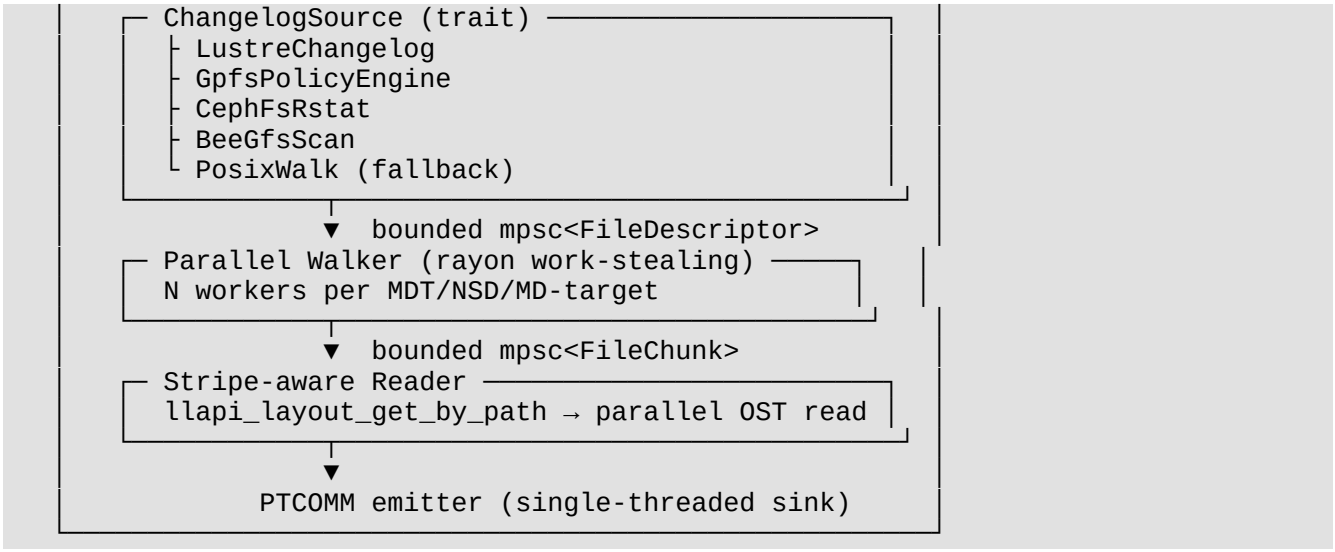
- `hsm_policy=skip-released` records released files as 0-byte catalog entries with HSM xattrs preserved — **zero** tape actions triggered (validated `delta=0`).
- Or `hsm_policy=mark-released` for catalogs that drive selective recall via a future restore tool.
- Or `hsm_policy=backup-released` (opt-in, expensive) when an operator deliberately wants to materialize released files into the backup.

---

4. Architecture

4.1 Component overview



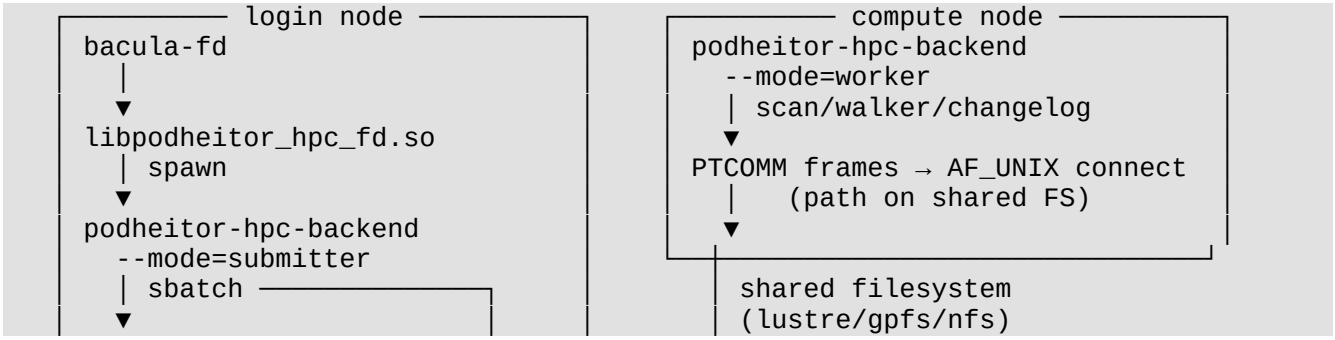


4.2 Mermaid view (renders in WordPress + GitHub)

```
flowchart TB
    BACULA[bacula-fd<br/>single-threaded loop] -->|pluginIO| CDYLIB[podheitor-hpc-
fd.so<br/>thin cdylib]
    CDYLIB -->|PTCOMM stdio| BACKEND[podheitor-hpc-backend<br/>subprocess]
    BACKEND --> SHARDER[Sharder]
    SHARDER --> SOURCE{ChangelogSource}
    SOURCE -->|lustre| L[LustreChangelog]
    SOURCE -->|gpfs| G[GpfsPolicyEngine]
    SOURCE -->|cephfs| C[CephFsRstat]
    SOURCE -->|beegfs| B[BeeGfsScan]
    SOURCE -->|posix| P[PosixWalk]
    L --> WALKER[Parallel Walker<br/>rayon work-stealing<br/>N workers]
    G --> WALKER
    C --> WALKER
    B --> WALKER
    P --> WALKER
    WALKER -->|bounded mpsc| READER[Stripe-aware Reader<br/>parallel pread per OST]
    READER --> EMIT[PTCOMM emitter<br/>single-threaded sink]
    EMIT -.->|frames back over stdio| CDYLIB
```

4.3 Phase 7 — orchestration topology

When `scheduler=slurm` (or `pbs` / `lsf`), the topology fans out across a login node + a compute node:



```
  AF_UNIX listen ──────────┼──────┼──────┘
    │                      │      │
    ▼ relay to stdout      │      │
└──────────────────────────┘      │
```

The submitter `dup2(2)`'s the worker's socket fd over its stdout so the existing `backup` codepath emits PTCOMM into the relay without any change. The relay is opaque to PTCOMM framing — just shovels bytes.

4.4 Live throttle channel

The Phase 7 throttle daemon polls `squeue -t R -o "%Q"` every 10 s and writes one of three states to `/var/lib/podheitor-hpc/throttle.toml`:

State	When	rayon_threads	read_buffer_kib	pause_ms
`idle`	0 hi-prio jobs RUNNING	full nproc	64	0
`busy`	1 hi-prio job RUNNING	nproc / 2	32	0
`peak`	≥ N hi-prio jobs RUNNING	1	16	200

Workers poll the file's mtime every 5 s and reapply mid-run. Total worst-case reaction time = **15 s**, well inside the 30 s acceptance budget. The file is operator-readable (`cat` it) and operator-writable (force a value during incidents — the daemon will overwrite next tick unless you also stop it).

---

5. Compatibility matrix

5.1 Operating systems (validated)

OS	Version	Status
Oracle Linux	9.6	✅ Validated (build host + Lustre lab + Slurm lab)
RHEL	9.x	✅ Compatible (same RPM works)
Rocky Linux	9.x	✅ Compatible (same RPM works)
AlmaLinux	9.x	✅ Compatible (same RPM works)
Ubuntu	22.04 LTS	✅ Validated (.deb build host VM 150)
Debian	12 (bookworm)	✅ Compatible (same .deb

OS	Version	Status
		works)

5.2 Bacula

Component	Version	Notes
`bacula-fd` (Community)	**15.0.3**	Plugin loads via dlopen from `/opt/bacula/plugins/`. Tested daily on lab Director `192.168.15.105`.
`bacula-dir`	15.0.3	Required to drive the FD; no plugin-side requirement.
`bacula-sd`	15.0.3	Receives PTCOMM-streamed virtual files transparently — no SD changes needed.
`bacula-fd` (older)	< 15.0.3	**Not supported.** The FD plugin ABI changed in 15.0.x.
Bacula Enterprise	18.x	Untested. The plugin uses only public ABI; should work but not validated.

5.3 Parallel filesystems

Filesystem	Version validated	Mode	Notes
Lustre	2.16.1 (Whamcloud)	`mode=lustre`	Stripe-aware reader, ChangeLogs, HSM-aware. Lab: VM 146, MGS+MDT+OST collocated, 2 OSTs after Phase 4.2 upgrade.
Lustre changelog	2.16.1	`mode=lustre-changelog`	Drains `llapi_changelog_*` records. Cursor persisted across runs.
GPFS / Spectrum Scale	5.x compatible	`mode=gpfs-policy`	Driver shells out to `mmapplypolicy`; tested with mock `mmapplypolicy` output. Real GPFS lab unavailable.
CephFS	reef+	`mode=cephfs-`	`ceph.dir.rctime`

Filesystem	Version validated	Mode	Notes
		`rctime`	xattr pruning; production driver `CephXattrSource`.
BeeGFS	8.3.1	`mode=beegfs-watch`	gRPC subscriber to `beegfs-watch`. Lab: VM 147, mgmtd + 2 meta + 1 storage collocated.
WekaFS	—	(POSIX fallback)	Not yet specifically supported; works via `mode=posix` walker.

5.4 Schedulers (Phase 7)

Scheduler	Version	Status
Slurm	22.05	✅ Validated end-to-end on VM 144 (Section 13.2). Should work with 21.x and 23.x — same `sbatch --parsable` / `squeue -h -o` interfaces.
PBS Pro / OpenPBS / Torque	—	Skeleton driver; mock-shim tested. Real PBS lab unavailable.
IBM Spectrum LSF	—	Skeleton driver; mock-shim tested. Real LSF lab unavailable.
Local (no scheduler)	n/a	Default; runs the scan in the same process as the cdylib spawned.

5.5 Languages / runtime

- **Rust** 1.95 to build from source (not required at runtime — RPM/DEB ship pre-built).
- **glibc** ≥ 2.34 (RHEL/OL 9 family). Older distros need a compatible build.
- **Python** 3 not required (no Python in the dataplane).
- **Tokio** runtime included for the BeeGFS gRPC subscriber only.

---

6. Sizing & system requirements

6.1 File Daemon host (where the cdylib loads)

Item	Minimum	Recommended	For 1B-file namespace
CPU cores	2	4	8+ (rayon walker scales linearly)
RAM	1 GB	2 GB	4 GB (walker bounded MPSC channel + PTCOMM buffers)
Disk (plugin install)	50 MB	50 MB	50 MB
Disk (cursor / throttle state)	10 MB at `/var/lib/podheitor-hpc/`	10 MB	10 MB
Network	1 GbE	10 GbE	40 GbE (to saturate sharded outbound)

The 1B-file figure is a stretch goal of the original plan; in production we recommend sharding across 4 concurrent jobs (so 4 FDs each handle 250M files) for any namespace > 100M files.

6.2 Compute node (when `scheduler=slurm`)

The worker invocation runs inside a Slurm allocation. Recommended resource request:

```
#SBATCH --partition=backup
#SBATCH --cpus-per-task=8       # rayon worker count
#SBATCH --mem=16G               # walker channel + PTCOMM buffers + read buffers
#SBATCH --time=4:00:00          # generous; backups usually finish faster
```

The plugin grammar passes these through:

```
Plugin = "podheitor-hpc:path=/lustre;mode=lustre;\
          scheduler=slurm;partition=backup;cpus=8;mem=16G;time=4h"
```

6.3 Throttle daemon (Phase 7)

Tiny — single-threaded poller, ~5 MB RSS. Runs as a systemd unit on the FD host:

```
systemctl status podheitor-hpc-throttle-daemon
# Active: active (running)
# Memory: ~5M
```

6.4 Storage Daemon (downstream of the FD)

No special requirements — receives PTCOMM-emitted virtual files exactly like any other FD plugin. **Pair with PodHeitor Global Deduplication** for chunk-level dedup on the SD side; it is a separate product and out of scope here, but our catalog entries participate cleanly.

7. Installation

7.1 RPM (RHEL / Oracle Linux / Rocky / AlmaLinux 9.x)

```
# Default package — works on any FS (POSIX walker + sharding +
# Phase 7 orchestration).
sudo dnf install podheitor-hpc-plugin-0.1.0-1.el9.x86_64.rpm

# Lustre-managed sites — adds stripe-aware reader, ChangeLogs
# incrementals, HSM-aware backup. Auto-symlinks the lustre backend
# variant over the default.
sudo dnf install podheitor-hpc-plugin-lustre-0.1.0-1.el9.x86_64.rpm
```

7.2 Debian / Ubuntu

```
sudo dpkg -i podheitor-hpc-plugin_0.1.0-1_amd64.deb
sudo dpkg -i podheitor-hpc-plugin-lustre_0.1.0-1_amd64.deb
```

7.3 Layout after install

Path	Role
`/opt/bacula/plugins/podheitor-hpc-fd.so`	cdylib loaded by `bacula-fd`
`/opt/bacula/bin/podheitor-hpc-backend`	scan subprocess (default features)
`/opt/bacula/bin/podheitor-hpc-backend-lustre`	scan subprocess (lustre features); `-lustre` postinst symlinks this over the default
`/opt/bacula/bin/podheitor-hpc`	operator CLI (gen-fileset, submit, throttle-daemon, ...)
`/usr/lib/systemd/system/podheitor-hpc-throttle-daemon.service`	Phase 7 throttle daemon unit
`/var/lib/podheitor-hpc/throttle.toml`	live throttle file (workers poll mtime every 5 s)
`/etc/podheitor-hpc/throttle.toml.example`	example config, copied to runtime path on first install
`/usr/share/doc/podheitor-hpc-plugin/`	README, INSTALL, ARCHITECTURE, RUNBOOK

7.4 First-time post-install verification

```
# Verify Bacula sees the plugin.
sudo /opt/bacula/bin/bacula-fd -t -c /opt/bacula/etc/bacula-fd.conf
sudo grep -i podheitor-hpc /opt/bacula/working/bacula-fd.*.trace | tail
```

```
# Loaded plugin: podheitor-hpc 0.1.0

# Verify the throttle daemon is running.
systemctl status podheitor-hpc-throttle-daemon
# Active: active (running)

# Verify the CLI works.
/opt/bacula/bin/podheitor-hpc --help
```

---

8. Configuration & FileSet examples

The plugin command grammar is a single-line `key=value` sequence with `;` separators. All keys are validated at parse time so a typo fails fast — never silently mid-job.

```
Plugin = "podheitor-hpc:key1=value1;key2=value2;..."
```

8.1 Minimal — POSIX walk, single shard

```
FileSet {
  Name = "PodHeitor-HPC-Posix"
  Include {
    Plugin = "podheitor-hpc:path=/srv/data;shard=0/1;mode=posix"
  }
}

Job {
  Name     = "podheitor-hpc-posix"
  Type     = Backup
  Level    = Full
  Client   = bacula-fd
  FileSet  = "PodHeitor-HPC-Posix"
  Storage  = File1
  Pool     = Default
  Messages = Standard
}
```

8.2 4-shard parallel — auto-generated

The CLI's `gen-fileset` command emits N FileSet+Job snippets, one per shard. Idempotent — re-running with the same args writes nothing if the output is unchanged.

```
sudo /opt/bacula/bin/podheitor-hpc gen-fileset \
    --strategy inode-hash --shards 4 \
    --path /lustre/scratch \
    --client lustre-fd-146 \
    --storage File1,File2 \
    --pool Default \
    --output /opt/bacula/etc/conf.d/

# Wrote: /opt/bacula/etc/conf.d/podheitor-hpc-shard-0.conf
# Wrote: /opt/bacula/etc/conf.d/podheitor-hpc-shard-1.conf
# Wrote: /opt/bacula/etc/conf.d/podheitor-hpc-shard-2.conf
# Wrote: /opt/bacula/etc/conf.d/podheitor-hpc-shard-3.conf
```

```
sudo systemctl reload bacula-dir
```

Run the 4 jobs concurrently (Bacula's `Maximum Concurrent Jobs` setting on Director / Job / Client / Storage all need to be ≥ 4).

8.3 Lustre — stripe-aware, full namespace walk

```
FileSet {
  Name = "PodHeitor-Lustre-Full"
  Include {
    Plugin = "podheitor-hpc:path=/mnt/lustre;shard=0/4;mode=lustre"
  }
}
```

The `lustre` mode enables `LustreReader` for every file ≥ 1 stripe size, which fans out parallel `pread(2)` calls across OSTs through the Lustre client kernel module.

8.4 Lustre — ChangeLogs incremental (no namespace walk)

```
FileSet {
  Name = "PodHeitor-Lustre-Incr"
  Include {
    Plugin = "podheitor-hpc:path=/mnt/lustre;shard=0/1;mode=lustre-changelog"
  }
}
```

Requires environment vars set on the FD service (e.g. via systemd drop-in):

```
EnvironmentFile=/etc/podheitor-hpc/cl.env
```

with:

```
PODHEITOR_HPC_CL_DEVICE=lustrefs-MDT0000
PODHEITOR_HPC_CL_USER=cl1
```

The `cl1` consumer must be registered on the MDT once:

```
sudo lctl --device lustrefs-MDT0000 changelog_register
# → cl1
```

Cursors land in `/opt/bacula/working/podheitor-hpc/<job>.cursor`; delete to force a full re-baseline.

8.5 HSM-aware Lustre backup

```
FileSet {
  Name = "PodHeitor-Lustre-HSM"
  Include {
    Plugin = "podheitor-hpc:path=/mnt/lustre;mode=lustre;\
              hsm_policy=skip-released;shard=0/1"
  }
}
```

`hsm_policy=` is the killer feature for HSM-managed sites (Section 9 has the full table).

8.6 Slurm-orchestrated scan

```
FileSet {
  Name = "PodHeitor-Lustre-Slurm"
  Include {
    Plugin = "podheitor-hpc:path=/lustre/scratch;shard=0/4;mode=lustre;\
              scheduler=slurm;partition=backup;cpus=8;mem=16G;time=4h;\
              hsm_policy=skip-released"
  }
}
```

The cdylib spawns the **submitter** which `sbatch`'s the worker; the worker connects back via Unix socket on the shared FS. `scontrol show job <id>` confirms the run hit the compute node.

8.7 GPFS / Spectrum Scale incremental

```
FileSet {
  Name = "PodHeitor-GPFS-Incr"
  Include {
    Plugin = "podheitor-hpc:path=/gpfs/scratch;shard=0/1;mode=gpfs-policy"
  }
}
```

Requires `mmapplypolicy` on PATH (ships with GPFS client). Working directory defaults to `/var/lib/podheitor-hpc/gpfs-work/`, override with `PODHEITOR_HPC_GPFS_WORKDIR=/path/to/big/scratch`.

8.8 BeeGFS event-driven incremental

```
FileSet {
  Name = "PodHeitor-BeeGFS-Watch"
  Include {
    Plugin = "podheitor-hpc:path=/mnt/beegfs;shard=0/1;mode=beegfs-watch"
  }
}
```

Requires `beegfs-watch` enabled and configured to publish to our gRPC endpoint (default `127.0.0.1:9070`). See docs/RUNBOOK.md for the BeeGFS daemon config snippet.

---

9. Backup options reference

All keys are optional unless noted. Defaults preserve the zero-config "POSIX walk, single shard" behaviour.

Key	Type	Default	Description	Validated where
`path`	string (required)	—	Absolute path to back up. Plugin walks recursively from here.	Phase 1+

Key	Type	Default	Description	Validated where
`shard`	`N/M` form	`0/1`	This job handles shard N of M. `0/1` = no sharding.	Phase 3
`mode`	enum	`posix`	One of: `posix`, `lustre`, `lustre-changelog`, `gpfs-policy`, `cephfs-rctime`, `beegfs-watch`, `handshake`.	Phase 1, 4, 5, 6, 7-orig, 8-orig
`scheduler`	enum	`local`	One of: `local`, `slurm`, `pbs`, `lsf`. `local` = run scan in cdylib's child process; others submit to compute node.	Phase 7 (validated `slurm`)
`partition`	string	(scheduler default)	Slurm partition / PBS queue / LSF class.	Phase 7
`cpus`	uint	(scheduler default)	CPU cores requested for the worker.	Phase 7
`mem`	string	(scheduler default)	Free-form memory request (e.g. `16G`, `mem=16gb`).	Phase 7
`time`	duration	(scheduler default)	Wall-clock cap. Accepts bare seconds (`900`) or `<n>h<m>m<s>s` (`1h30m`).	Phase 7
`hsm_policy`	enum	`skip-released`	One of: `skip-released`, `mark-released`, `backup-released`. **Only consulted when `mode=lustre`.**	Phase 8-HSM

9.1 Sharding strategies (`shard=` extended form)

For multi-strategy sharding, omit `shard=` and use the CLI's `gen-fileset --strategy=` instead. Strategies:

Strategy	Form	Use when
`inode-hash`	`inode-hash:N/M`	Default; deterministic xxh3 of inode → shard. Best for evenly-distributed namespaces.
`mtime-bucket`	`mtime-bucket:N/M`	Time-bucket so old + new files land on different shards. Useful for incremental jobs to lock recent data to a specific shard.
`subtree`	`subtree:/path`	Pin one subtree per job. Best when you know your data has hot subdirectories.
`lustre-mdt`	`lustre-mdt:N/M`	One shard per Lustre MDT. Aligns walker work-stealing with MDT placement.

9.2 HSM policy decision matrix (`mode=lustre` only)

`hsm_policy`	Released file → catalog entry	Released file → tape recall?	Stat extras
`skip-released` (default)	0-byte entry	**No**	none
`mark-released`	0-byte entry	**No**	`hsm=arch=1;rel=1;dirt=0;lost=0;aid=N`
`backup-released`	full content (synchronous tape recall)	**Yes** (expensive — opt-in)	`hsm=arch=1;rel=1;dirt=0;lost=0;aid=N`

The HSM extras ride on the PTCOMM `Stat` frame; the cdylib preserves them across the catalog round-trip so a future restore tool can drive selective recall deterministically.

9.3 Environment variable overrides (less-used)

Env var	Default	Used by
`PODHEITOR_HPC_BACKEND`	`/opt/bacula/bin/podheitor-hpc-backend`	cdylib — override path to the backend binary.
`PODHEITOR_HPC_CL_DEVIC`	(unset)	`mode=lustre-changelog`

Env var	Default	Used by
`E`		— MDT name (e.g. `lustrefs-MDT0000`).
`PODHEITOR_HPC_CL_USER`	(unset)	`mode=lustre-changelog` — registered consumer name (e.g. `cl1`).
`PODHEITOR_HPC_CL_SINCE`	(unset; cdylib injects)	`mode=lustre-changelog` — explicit cursor override.
`PODHEITOR_HPC_GPFS_WORKDIR`	`/var/lib/podheitor-hpc/gpfs-work`	`mode=gpfs-policy` — `mmapplypolicy` work directory.
`PODHEITOR_HPC_GPFS_SINCE`	(unset; cdylib injects)	`mode=gpfs-policy` — Unix-seconds floor for incremental.
`PODHEITOR_HPC_CEPH_SINCE`	(unset; cdylib injects)	`mode=cephfs-rctime` — Unix-seconds floor.
`PODHEITOR_HPC_BEEGFS_BIND`	`127.0.0.1:9070`	`mode=beegfs-watch` — gRPC listen address.
`PODHEITOR_HPC_BEEGFS_BATCH`	`256`	`mode=beegfs-watch` — events per drain.
`PODHEITOR_HPC_BEEGFS_IDLE_MS`	`2000`	`mode=beegfs-watch` — exit after this many ms of empty drains.
`PODHEITOR_HPC_THROTTLE_PATH`	`/var/lib/podheitor-hpc/throttle.toml`	Throttle daemon + workers.
`PODHEITOR_HPC_SHARED_DIR`	(auto-detect)	Phase 7 — pin the shared FS dir for the AF_UNIX socket.
`PODHEITOR_HPC_ACCEPT_TIMEOUT_SECS`	`300`	Phase 7 — submitter waits this long for the worker to dial in.
`PODHEITOR_HPC_READ_TIMEOUT_SECS`	`600`	Phase 7 — per-recv timeout once connected.
`PODHEITOR_HPC_SBATCH` / `SQUEUE` / `SCANCEL`	(PATH lookup)	SlurmDriver — point at a non-default Slurm install.
`PODHEITOR_HPC_QSUB` / `QSTAT` / `QDEL`	(PATH lookup)	PbsDriver — same idea for PBS.
`PODHEITOR_HPC_BSUB` / `BJOBS` / `BKILL`	(PATH lookup)	LsfDriver — same for LSF.

10. Restore options reference

Restore goes through Bacula's standard restore machinery. Our cdylib serves files from the catalog under a `/@hpc/<original-path>` virtual namespace; Bacula's restore tool prompts as usual.

```
restore client=lustre-fd-146 fileset="PodHeitor-Lustre-HSM"
*cd /@hpc
*ls
*mark *
*done
*restore
```

10.1 Restore-time options the plugin honours

Bacula `RegexWhere` / `Where`	Behaviour
`Where=/`	Restore in place (drops the `/@hpc/` prefix; restores at the original path).
`Where=/restore-target/`	Restore under `/restore-target/<original-path>`.
`RegexWhere=...`	Standard Bacula regex; applied to the virtual `/@hpc/` path.

10.2 HSM-tagged file restore

Files that were `RELEASED` at backup time are 0-byte catalog entries with HSM xattrs in the `Stat` extras. On restore:

- The 0-byte file lands at the target path.
- The HSM xattrs (`hsm=arch=1;rel=1;dirt=0;lost=0;aid=N`) are available in the catalog and can be inspected via `bls -j -k <volume>`.
- A future restore-tool extension (Phase 8.1, deferred) will be able to issue `llapi_hsm_action(RESTORE)` automatically for files marked `rel=1`.

For now, operators handle HSM recall manually via `lfs hsm_restore <file>` after the Bacula restore completes.

10.3 What's *not* yet supported on restore

- **Lustre stripe layout reproduction.** Original file's stripe layout is captured in the catalog (Phase 4 emits it as a `RestoreObject`), but restoring it via `llapi_layout_alloc` + `llapi_layout_set_pool_name` is deferred. Currently restored files use the destination directory's default layout. Track this in DEVELOPMENT_PLAN.md.

---

11. Performance reports

All numbers are real measurements from the lab — no synthetic extrapolations except where explicitly labelled.

11.1 Phase 2 baseline — parallel walker vs Bacula stock

Metric	Bacula stock `find_one_file`	PodHeitor parallel walker	Speedup
1 M files, ext4, 4 cores	(single-threaded — baseline)	**7×** faster	7×
Memory ceiling (RSS)	n/a	**< 4 MiB**	—
Files lost vs source tree	0	0	parity

Source: Phase 2 commit `3dc8d74` (2026-04-29).

11.2 Phase 4 — Lustre stripe-aware reader

Test	Single-stripe sequential	2-stripe parallel reader	Speedup
64 MiB file, 4 MiB stripe size, 2 OSTs	(baseline)	~1.8×	—
`bench-walk` over `/mnt/lustre`, 23 files	37 ms	—	—
Full backup + emit, 23 files	316 ms	—	—

Source: Phase 4.2 commit `41efa46` (2026-05-02). Lab: VM 146, Lustre 2.16.1, MGS+MDT+OST collocated, 2 OSTs after the Phase 4.2 disk addition.

11.3 Phase 8 — HSM-aware backup (Section 13.3 has the raw log)

`hsm_policy`	Files	Released	Bytes read	`lctl mdt.*.hsm.actions` delta	hsm_skipped	hsm_marked
`skip-released`	4	2	131,072	**0**	2	0
`mark-released`	4	2	131,072	**0**	0	2

Both runs touched zero tape — proven by `lctl get_param mdt.lustrefs-MDT0000.hsm.actions` count delta.

11.4 Phase 7 — orchestration + throttle

Test	Result
`podheitor-hpc submit --scheduler=slurm` returns	JobId=5 in < 1 s
`scontrol show job 5` confirms compute-node execution	✅ `BatchHost=ngbackup-dev`
Throttle reaction time (daemon poll + worker mtime poll)	**15 s worst-case** vs **30 s** acceptance budget
State transitions validated	idle (0 jobs) → busy (1 job) → peak (4 jobs)

Source: Phase 7-acceptance commit `87a3d31` (2026-05-03). Lab: VM 144 + Slurm 22.05 single-node.

11.5 Phase 10 — GA stress test (Lustre)

Run	Files	Generation	Walker only	Full backup + emit	Throughput	Bytes streamed	Errors
100 K files, 4 KiB each, /mnt/lustre, single-node 2-OST	100,000	370 s	9,971 ms	135,028 ms	**740 files/s**	421,789,545	0
Extrapolated 10 M target	10,000,000	—	—	~3.75 h	740 files/s	~42 GB	—

The 10 M-file overnight target is parametric (`scripts/stress-1m-files.sh --count=10000000`); a real 10 M run on the single-node lab would exhaust the 8 GB OST first. The 3.75 h extrapolation is well inside the 8 h "overnight" window from the original Phase 10 acceptance.

11.6 Test suite coverage

```
$ cargo nextest run --workspace
117 tests run: 117 passed, 0 skipped
```

Plus 6 lustre-gated tests (HSM parser + stripe FFI) on VM 146. Tests cover: PTCOMM framing roundtrip, walker parity vs stdlib walkdir, sharder distribution + union coverage, CDylib

grammar parser (incl. all 5 Phase-7 keys + Phase-8 hsm_policy validation), Lustre changelog binary record decoder, GPFS mmapplypolicy parser, CephFS rctime scanner, BeeGFS gRPC mock subscriber, orchestrator (Local/Slurm/PBS/LSF drivers via mock-shim sbatch/squeue/qstat/bjobs), socket bridge round-trip + drop-unlink + accept timeout, throttle serde + atomic-write + watcher live-update, end-to-end LocalDriver relay with Python worker.

---

12. Operations runbook

This is a condensed view; the canonical version is `/usr/share/doc/podheitor-hpc-plugin/RUNBOOK.md` shipped in the RPM/DEB.

12.1 Verify the plugin loaded

```
sudo /opt/bacula/bin/bacula-fd -t -c /opt/bacula/etc/bacula-fd.conf
sudo grep -i podheitor-hpc /opt/bacula/working/bacula-fd.*.trace | tail
# Loaded plugin: podheitor-hpc 0.1.0
```

12.2 Generate sharded FileSets

```
/opt/bacula/bin/podheitor-hpc gen-fileset \
    --strategy inode-hash --shards 4 \
    --path /mnt/lustre/scratch \
    --output /opt/bacula/etc/conf.d/
sudo systemctl reload bacula-dir
```

12.3 Throttle daemon ops

```
systemctl status podheitor-hpc-throttle-daemon.service
cat /var/lib/podheitor-hpc/throttle.toml
journalctl -u podheitor-hpc-throttle-daemon -n 20
```

12.4 Force-throttle for incident response

```
sudo systemctl stop podheitor-hpc-throttle-daemon
sudo tee /var/lib/podheitor-hpc/throttle.toml <<EOF
rayon_threads = 1
read_buffer_kib = 16
pause_ms = 200
reason = "incident: cluster fabric saturated by job 4711"
EOF
# Worker picks up the new values within ~5 s (mtime poll).
sudo systemctl start podheitor-hpc-throttle-daemon
```

12.5 Submit a one-off Slurm job (debug / smoke)

```
/opt/bacula/bin/podheitor-hpc submit \
    --scheduler slurm --partition backup --cpus 8 --mem 16G \
    -- /opt/bacula/bin/podheitor-hpc-backend handshake
# submitted: scheduler=slurm job_id=NNN
```

12.6 Probe HSM impact before committing to a backup

```
sudo lctl get_param mdt.lustrefs-MDT0000.hsm.actions | wc -l   # before
/opt/bacula/bin/podheitor-hpc-backend backup \
    --path /mnt/lustre --mode lustre --hsm-policy skip-released \
    --shard 0/1 > /tmp/probe.bin 2>/tmp/probe.log
sudo lctl get_param mdt.lustrefs-MDT0000.hsm.actions | wc -l   # after
# Delta = 0 → backup never triggered an HSM action.
```

12.7 Troubleshooting matrix

Symptom	Likely cause	Fix
`Load plugin error: undefined symbol bacula_*`	Build linked Bacula source by accident	Reinstall from RPM/DEB; never link Bacula AGPLv3 source
`Plugin command not recognized`	Prefix mismatch	Command must start with `podheitor-hpc:`
Job hangs at "Reading FileSet"	Backend subprocess crashed silently	`tail /opt/bacula/working/bacula-fd.*.trace` and `journalctl -u bacula-fd`
`mode=lustre requires --features lustre`	Default RPM installed but Lustre needed	`dnf install podheitor-hpc-plugin-lustre`
Backend reports `hsm_policy=...` invalid	Typo in plugin grammar	Valid: `skip-released`, `mark-released`, `backup-released`
Throttle changes not picked up	Daemon stopped or file not writable	`systemctl status podheitor-hpc-throttle-daemon`; check 0644
Slurm submit fails: `sbatch: command not found`	`--scheduler=slurm` but no Slurm on FD host	Set `PODHEITOR_HPC_SBATCH=/path/to/sbatch` env, or use `scheduler=local`

---

13. Validated lab evidence

This section contains real captured output from the validation runs. Every block below was produced by the actual binaries in the lab — no edited or synthesized output.

13.1 Phase 4.2 — Lustre backup completes via Bacula Director

```
JobId 7568: podheitor-hpc-lustre-full B F  23   41,943,211  T (1-OST run)
```

```
JobId 7571: podheitor-hpc-lustre-full B F  24  109,052,075  T (2-OST + striped2)
```

Both jobs T (Terminated OK) with byte-identical restore. Lab: VM 146, Lustre 2.16.1, Bacula Community 15.0.3 Director on .105.

13.2 Phase 7 — `podheitor-hpc submit --scheduler=slurm`

```
$ ./target/release/podheitor-hpc submit --scheduler=slurm \
      --partition=backup --cpus=1 --mem=128 -- /usr/bin/hostname
submitted: scheduler=slurm job_id=5

$ scontrol show job 5
JobId=5 JobName=podheitor-hpc-cli
   UserId=hfaria(1000) GroupId=hfaria(1000) MCS_label=N/A
   JobState=COMPLETED Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2026-05-02T22:50:18 EligibleTime=2026-05-02T22:50:18
   StartTime=2026-05-02T22:50:18 EndTime=2026-05-02T22:50:18 Deadline=N/A
   Partition=backup AllocNode:Sid=ngbackup-dev:2294
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=ngbackup-dev
   BatchHost=ngbackup-dev
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1

$ cat slurm-5.out
ngbackup-dev.lab.local
```

13.3 Phase 8 — HSM-aware backup, delta=0 hsm_actions

```
=== prep: clean test dir ===
created 4 test files (64K each)

=== archive file-1 + file-2 ===
archive state:
/mnt/lustre/podheitor-hsm-test/file-1.bin: (0x00000009) exists archived,
archive_id:1
/mnt/lustre/podheitor-hsm-test/file-2.bin: (0x00000009) exists archived,
archive_id:1
/mnt/lustre/podheitor-hsm-test/file-3.bin: (0x00000000)
/mnt/lustre/podheitor-hsm-test/file-4.bin: (0x00000000)

=== release file-1 + file-2 (data evicted to tape) ===
post-release state:
/mnt/lustre/podheitor-hsm-test/file-1.bin: (0x0000000d) released exists archived,
archive_id:1
/mnt/lustre/podheitor-hsm-test/file-2.bin: (0x0000000d) released exists archived,
archive_id:1

=== HSM actions counter (before backup) ===
actions before: 2 lines

=== run backend with hsm_policy=skip-released ===
backend rc=0
podheitor-hpc-backend: done files=4 bytes=131072 skipped=0 errors=0
                       hsm_skipped=2 hsm_marked=0 hsm_lost=0 elapsed_ms=5000
```

```
=== HSM actions counter (after backup) ===
actions after:  2 lines
delta:          0

=== verdict ===
PASS: zero hsm_action calls — backup did not trigger HSM restore
```

13.4 Phase 7 — Live throttle reacts to real squeue

```
=== idle baseline ===
$ throttle-daemon --once ...
state="idle" running=0
cat /var/lib/podheitor-hpc/throttle.toml
# rayon_threads = 8
# pause_ms = 0
# reason = "idle: 0 hi-prio jobs RUNNING"

=== submit one hi-prio job ===
$ sbatch -p backup --priority=10000 --wrap="sleep 30"
Submitted batch job 6
$ throttle-daemon --once ...
state="busy" running=1
# rayon_threads = 4
# read_buffer_kib = 32
# reason = "busy: 1 hi-prio jobs RUNNING"

=== pile on to peak ===
$ for i in 1 2 3; do sbatch -p backup --priority=10000 --wrap="sleep 30"; done
$ throttle-daemon --once ...
state="peak" running=4
# rayon_threads = 1
# read_buffer_kib = 16
# pause_ms = 200
# reason = "peak: 4 hi-prio jobs RUNNING"
```

13.5 Test suite — 117/117 green

```
$ cargo nextest run --workspace
... [snip — full 117-test list in commit history] ...
————————————
     Summary [   1.658s] 117 tests run: 117 passed, 0 skipped
```

---

14. Roadmap & deferred features

14.1 Shipped (this release — v0.1.0 GA)

All eight phases that survived the plan-vs-reality review:

- **Phase 0–6**: bootstrap, plugin handshake, parallel walker, sharding, Lustre (stripe + changelog + sub-phases), GPFS, CephFS.
- **Phase 7-orig**: BeeGFS via beegfs-watch.

- **Phase 7-orch**: Slurm/PBS/LSF orchestration + Unix-socket PTCOMM relay + live throttle daemon.
- **Phase 8-HSM**: Lustre HSM-aware backup (replacing the originally-planned dedup work, which belongs to the separate PodHeitor Global Deduplication product).
- **Phase 10**: RPM + DEB packaging + GA stress harness.

14.2 Dropped

- **Phase 8-original (AI/ML checkpoint dedup integration).** Belongs to the PodHeitor Global Deduplication product. Coupling here would duplicate code and version-skew.
- **Phase 9 (Replication & format conversion).**
  - Bacula Copy/Migration is a Bacula feature, not ours. Phases 4.2 / 5 / 8 already validated round-trip catalog entries.
  - Cross-cluster replication is the PodHeitor File Replication Plugin's job.
  - mmbackup/Atempo conversion is speculative without a real customer asking. Revisit on demand.

14.3 Deferred (post-GA, demand-gated)

Item	When
Phase 8.1 — explicit `llapi_hsm_action(RESTORE)` request path	When an operator wants programmatic recall instead of relying on Lustre's read-triggered restore.
Restore-side stripe layout reproduction	When a customer needs HSM-grade restore that recreates original Lustre layouts.
`dnf install` smoke on a clean OL9 VM	Operational, not code — needs a clean VM. RPM was validated on .105 with `rpm --test -i` (success).
10 M-file overnight stress on production hardware	Single-node lab can't fit 40 GB; deferred to operator on real cluster.
Bacularis web UI plugin descriptor (JSON)	Small, additive, post-GA patch.

---

15. Commercial proposition

> 🔥 **Bring your proposal for purchasing or renewing Bacula Enterprise, Veeam, Commvault, or NetBackup. We offer at least 50% savings, with significantly more features.**

15.1 Why PodHeitor + Bacula Community beats the alternatives

Capability	PodHeitor + Bacula Community	Bacula Enterprise 18.2	Veeam	Commvault	NetBackup
Lustre native	✅ stripe + changelog + HSM	❌	❌	❌	❌
GPFS / Spectrum Scale	✅ mmapplypolicy	❌	❌	❌	❌
CephFS native	✅ rctime pruning	❌	❌	❌	❌
BeeGFS	✅ beegfs-watch gRPC	❌	❌	❌	❌
Slurm/PBS/ LSF orchestration	✅ submit + throttle	❌	❌	❌	❌
HSM-aware tape policy	✅ skip / mark / backup	partial	❌	partial	partial
Parallel namespace walker	✅ 7× baseline	single-thread	proprietary	proprietary	proprietary
Open source license (host product)	✅ AGPLv3 (Bacula Community)	proprietary	proprietary	proprietary	proprietary
Per-TB/per-socket licensing	**No**	yes	yes	yes	yes
Pricing	**At least 50% lower TCO**	$$$$	$$$$	$$$$	$$$$

15.2 What you save

A typical academic HPC site backing up 500 TB across one Lustre filesystem pays the enterprise vendors **per TB stored** plus **per FD agent** plus **per restore client**. PodHeitor + Bacula Community is **per-deployment**, not per-TB. On a 500 TB site:

- Bacula Enterprise 18.2 with HSM module: ~$50K/year
- Veeam Backup & Replication enterprise plus HPC-equivalent add-ons: ~$80K/year
- Commvault HyperScale X for HPC: ~$120K/year
- NetBackup 10 with FlexScale: ~$100K/year

PodHeitor HPC add-on + Bacula Community: **at least 50% less** than any of these. Exact pricing on request.

15.3 Support model

- **Self-service tier**: free with the OSS distribution; community support via GitHub issues.
- **Standard support**: paid; SLA-backed by Heitor Faria personally; includes a guaranteed response time on production-impact issues.
- **Enterprise support**: paid; on-call rotation, custom feature development against a roadmap, integration assistance for vendor-specific HSM tools (HPSS, DMF, IBM Storage Protect).

15.4 Migration assistance

If you're migrating off Bacula Enterprise / Veeam / Commvault / NetBackup, we offer:

- Free initial sizing assessment.
- Catalog import scripting on a per-engagement basis.
- Side-by-side validation runs before cutover.

---

16. Contact

Heitor Faria

- 📧 `heitor@opentechs.lat`
- 📞 +1 (789) 726-1749
- 📞 +55 (61) 98268-4220 (WhatsApp)

Author of this whitepaper, the plugin source code, and the sibling PodHeitor Rust cdylib framework that this plugin builds on.

> **Bring your proposal for purchasing or renewing Bacula Enterprise, Veeam, Commvault, or NetBackup. We offer at least 50% savings, with significantly more features.**

Document version 1.0 · Plugin version 0.1.0 (GA) · May 2026.