

Compiled Once. Published Once. Mounted Everywhere.

How scientific software actually reaches the machines that run it - CernVM-FS, the Compute Canada model, and EESSI.

2026-08-08

Table of contents

1	Where this starts: enterprise Linux is old on purpose	2
1.1	The four requirements, and what fills them	3
2	The obvious objection: “isn’t this what containers are for?”	4
3	The filesystem: software, streamed	5
4	Integrity: one signature, covering everything below it	6
5	Distribution: four tiers, and what each one is actually for	7
5.1	What it scales to	9
6	Performance: faster than the parallel filesystem you already own	9
7	The architecture: four layers, and why the middle one is a role	11
7.1	The observation that makes this worth adopting	13
8	The compatibility layer: a complete userland that is not the host’s	13
8.1	Why Nix was replaced, in their words	14
8.2	What it actually costs to run one	15
9	Build and selection: EasyBuild records it, Lmod makes it choosable	15
10	The Canadian stack: the one that has been in production the longest	18
10.1	The version lineage	19
10.2	How something gets into it	19
11	EESSI: the same architecture, rebuilt for machines nobody had in 2017	20
11.1	The seam: host injections	22

12 Running it: what you actually stand up, and in what order	23
12.1 The four error messages, decoded	24
13 The boundary: what this does not solve	24
14 The consequences: four questions that stop being open	25
15 References	26

Abstract. Almost every research computing estate installs the same scientific software more than once, in more than one place, to more than one result. This brief walks the three production systems that answer that problem properly: **CernVM-FS**, the read-only streaming filesystem built at CERN to give the worldwide LHC computing grid one software tree; the **Compute Canada / Digital Research Alliance of Canada** stack, which has served every national research system in Canada from one catalogue since 2017; and **EESSI**, the European project built on the same four-layer architecture and extended to hardware the original never targeted. It covers what each layer does and why it is there, the toolchain the two stacks share - Gentoo Prefix, EasyBuild, Lmod, archspec - what it costs to operate, and where it stops. Every non-obvious claim carries a numbered citation; section 15 is the bibliography, and nothing is asserted here that is not in it. The live version of this brief, with the same figures, is at <https://what.ndexr.io>.

Part I - The problem nobody funds. Installing scientific software is not hard. Installing it once per team, forever, is - and that is the thing every research computing estate is actually doing.

1 Where this starts: enterprise Linux is old on purpose

Almost every HPC cluster runs an enterprise Linux distribution, for good reasons: the vendor supports the interconnect, the parallel filesystem and the GPU driver against that kernel and no other. The cost of that support is that the userland ships frozen years behind what anyone is writing code against today.¹²

Distribution	Kernel	GCC	glibc	Python
RHEL / Rocky 8	4.18	8.4	2.28	3.9.2
RHEL / Rocky 9	5.14	11.2.1	2.34	3.9.10
Fedora 41	6.11	14.2.1	2.40	3.13.0

¹Oldeman, B. et al. (2023). *Digital Research Alliance of Canada site talk: lessons learned, new developments*. EasyBuild User Meeting 2023. <https://easybuild.io/eum23/> — the guiding principle, the distribution-version comparison, and the Nix post-mortem.

²Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

The first two rows are what your cluster runs. The third is what the documentation your users are reading assumes.³

The user’s move does not work. They follow the upstream install instructions, type `sudo dnf install python3.12-devel` on a login node, and are told they are not in the sudoers file. Nothing about that is their fault - the instructions were written for a laptop.

The administrator’s move does not scale. So the site installs it centrally and writes a modulefile. Then another version. Then the same package against a second compiler, and a third MPI. The modulefiles were written by hand, and so was the software, and neither is written the same way twice.

And the site next door is doing it too. Every site in the country is compiling the same packages, from the same sources, to different results. The effort is real, the people are good, and almost all of it is duplicated.

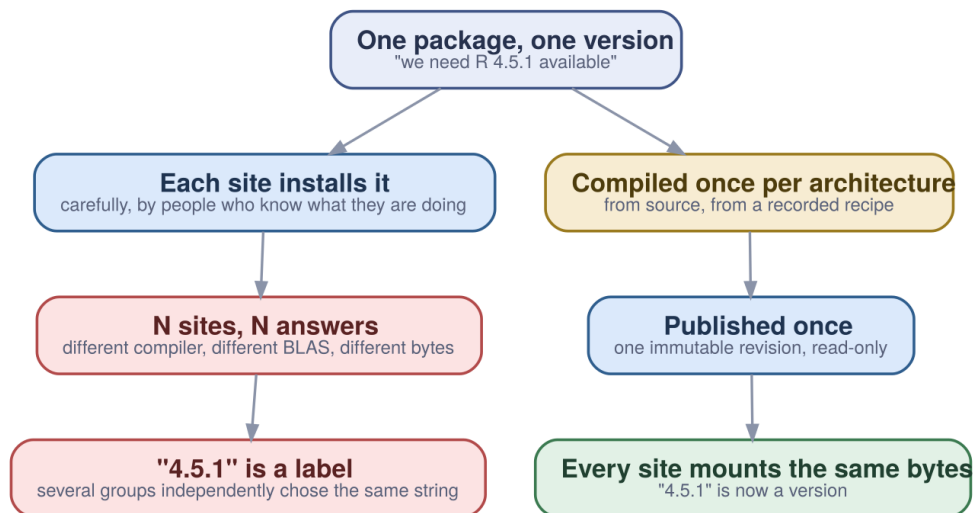


Figure 1: The axis is not build-or-mount - the tree does get built, from source, always. It is how many times. Install per site and “4.5.1” is a string several groups independently chose; install once and it is a version.

1.1 The four requirements, and what fills them

Canada’s national systems hit this first and hardest: five regional consortia, thirty-eight member institutions, around eighteen thousand accounts, and six national systems that had to look the same to a researcher moving between them.⁴⁵ The requirements were written down before the

³Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

⁴Boissonneault, M., Oldeman, B., Taylor, R. (2019). *Providing a Unified Software Environment for Canada’s National Advanced Computing Centers*. PEARC’19. <https://doi.org/10.1145/3332186.3332210> — the peer-reviewed statement of the whole model. If you read one thing on this list, read this.

⁵Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

tools were chosen, which is why the result generalises.

Users should be presented with an interface that is as consistent and as easy to use as possible across all sites. It should also offer optimal performance.

— Digital Research Alliance of Canada, site talk, EasyBuild User Meeting 2023

1. **A way to distribute it.** One tree, published once, reaching every machine without a per-machine installation step. → CernVM-FS.
2. **Independence from the host operating system.** So the same tree runs on the cluster's frozen enterprise Linux and on somebody's current laptop. → a compatibility layer.
3. **Automated installation, because humans are not consistent.** The build has to be a recorded recipe rather than a remembered procedure. → EasyBuild.
4. **An interface that survives ten thousand combinations.** Users choose software, not build permutations. → Lmod, hierarchical.

Why this matters. These four are requirements, not a product. Any organisation that installs scientific software at more than one place is already paying for all four - usually as unbudgeted effort spread across teams that do not know they are solving the same problem.

2 The obvious objection: “isn't this what containers are for?”

Partly, and the part they do they do well. It is worth separating two things the word “reproducible” is doing at once.

Repeatability is running the same bits again. Fix an image, move it, run it, get the same result. Containers are excellent at this and nothing here replaces them.

Accountability is saying what those bits *are*: which compiler, which flags, which source revision, which patch set, on which machine. If the bits were compiled elsewhere, that information never existed on your side of the boundary. Freezing an unknown does not make it known.

A container is a distribution format, not a derivation. That is not a criticism - it is a description of what it is for. The stack inside it still has to be built once, by someone, from something.

Even setting provenance aside, shipping the stack inside the image goes wrong at scale in two specific ways. **The image grows with the catalogue:** add a package anyone might want and every image that might want it is rebuilt and redistributed, so the cost of a small addition is set by the size of the collection. And **you ship what might be used, not what is:** importing one Python library touches a few thousand files out of a stack of hundreds of thousands. Pulling an image moves all of them every time; streaming moves the few thousand, once.⁶

Containers stay in the picture. The move is not to abandon them - it is to change what they carry. An image with an operating system and a mount point stays small and stays stable, because the catalogue it reaches is not inside it. CernVM-FS applies the same idea to images themselves:

⁶Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

unpacked.cern.ch publishes container images in extracted, per-file, deduplicated form, so a runtime executes a root filesystem straight off the mount and pulls only the files the run actually opens.⁷⁸

Part II - The delivery substrate. CernVM-FS was built at CERN to give the worldwide LHC computing grid one software tree. It is the layer everything else stands on, and the one most worth understanding on its own terms.

3 The filesystem: software, streamed

CernVM-FS presents a read-only directory under `/cvmfs` that behaves like an on-demand streaming service - for scientific software rather than for video. Nothing is installed and nothing is downloaded up front. Files arrive when a process opens them.⁹¹⁰¹¹

A filesystem, not a package manager. It is implemented in userspace over FUSE, so it installs on every worker node without touching the kernel, and every tool that can open a file can use it - the dynamic loader included.

Content-addressed, so identical bytes are stored once. Files are named by their cryptographic hash rather than by path. Fifty modules sharing a library share one object. Deduplication is not an optimisation pass; it is what naming by content means.

Compressed on the server, transparent on the client. Objects are stored compressed and decompressed on arrival, so the wire cost and the storage cost both fall without anything upstream of the client having to care.

Ordinary HTTP, all the way down. Which is why the caching hierarchy is a Squid proxy and the mirror is an Apache server. There is no bespoke transport to operate.

Why this matters. A vast catalogue costs nothing until it is used. That is the property that makes it affordable to keep every version of everything you have ever built, which in turn is what makes “reproduce the result from 2029” a mount rather than a rebuild.

⁷CernVM-FS project. *CernVM-FS documentation*. <https://cvmfs.readthedocs.io> — manifest fields, catalog structure, cache behaviour, publishing transactions, garbage collection and DUCC. The `cpt-details` chapter is the one to read.

⁸Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

⁹Blomer, J., Buncic, P., Fuhrmann, T. (2011). *CernVM-FS: Delivering Scientific Software to Globally Distributed Computing Resources*. NDM’11. <https://doi.org/10.1145/2110217.2110225> — the original design, and the source of the content-addressed, HTTP-transported model.

¹⁰CernVM-FS project. *CernVM-FS documentation*. <https://cvmfs.readthedocs.io> — manifest fields, catalog structure, cache behaviour, publishing transactions, garbage collection and DUCC. The `cpt-details` chapter is the one to read.

¹¹MultiXscale / EESSI. *Best Practices for CernVM-FS in HPC*. <https://multixscale.github.io/cvmfs-tutorial-hpc-best-practices/> — proxy and Stratum 1 sizing, diskless cache configuration, and which workarounds are no longer recommended.

4 Integrity: one signature, covering everything below it

A globally distributed, aggressively cached, plain-HTTP filesystem sounds like it should be impossible to trust. It is not, because of how little of it is signed: exactly one object, from which everything else is reachable by content hash.¹²

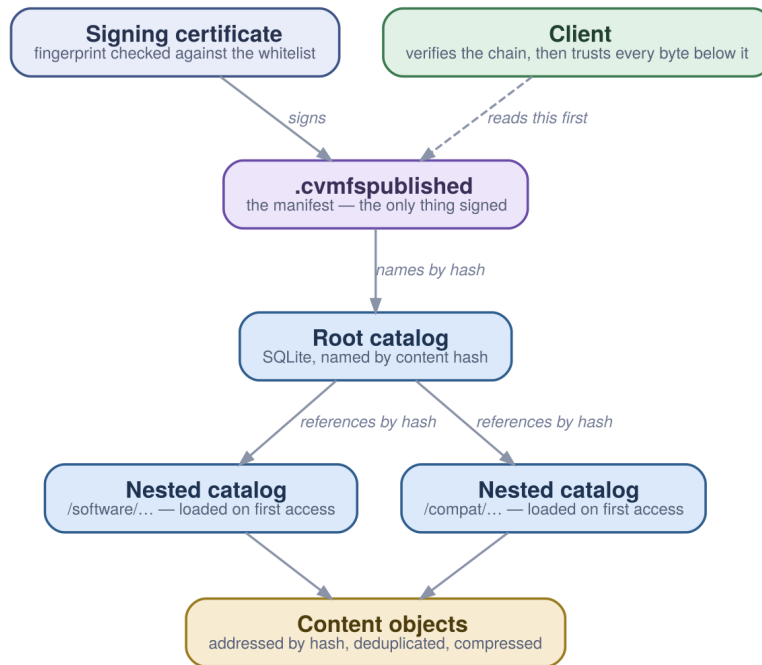


Figure 2: The manifest names the root catalog by hash; the root catalog names its sub-catalogs by hash; each catalog names its files by hash. So a single signature over the manifest covers the whole tree transitively - and a client that verifies it once can accept every byte underneath from any cache, however untrusted.

The repository manifest is the entire entry point - seven fields and a signature:

```
$ curl -s http://<stratum1>/cvmfs/software.eessi.io/.cvmfspublished | head
C7d1e...          # root catalog, by content hash
B1179648          # its size in bytes
Xf3a9...          # hash of the signing certificate
T1747526400       # when this revision was published
D240              # how long a client may cache it, in seconds
S5348             # revision number
Nsoftware.eessi.io # fully qualified repository name
--               # everything below is the signature
```

¹²CernVM-FS project. *CernVM-FS documentation*. <https://cvmfs.readthedocs.io> — manifest fields, catalog structure, cache behaviour, publishing transactions, garbage collection and DUCC. The `cpt-details` chapter is the one to read.

Catalogs are SQLite, and nested. Metadata lives in ordinary database files, split at chosen boundaries and loaded on first access. A repository with billions of files does not require a client to hold billions of entries.

Trust is a fingerprint, and it is revocable. Clients check the signing certificate against a whitelist signed by a master key. A blacklist can revoke a certificate, and can also refuse every revision below a given number - so a compromised key cannot be used to serve an old, known-bad tree.

Publishing is a transaction, and revisions are immutable. Changes accumulate on a writable overlay and become a new revision atomically. There is no half-published state for a client to catch. Tag the revisions that matter and a rollback is a command rather than a restore.

```
cvmfs_server transaction software.example.org # take the lock
# ... install into /cvmfs/software.example.org ...
cvmfs_server publish -a r-4.5.1 -m 'R 4.5.1, foss/2023a' software.example.org

# an abort throws the whole change away; there is no half-published state
cvmfs_server abort software.example.org

# and a named revision can be returned to
cvmfs_server rollback -t r-4.5.1 software.example.org
```

Why this matters. This is the paragraph a security review actually reads. The claim is not that the transport is trusted - it is that the transport does not need to be. Content is verified against hashes that descend from one signed manifest, so a proxy, a mirror or a disk can all be wrong without the client being wrong.

5 Distribution: four tiers, and what each one is actually for

The tiers are often drawn as a diagram and rarely explained as a set of decisions. Each one exists to solve a different failure, and a site can stop at any depth that matches its size.

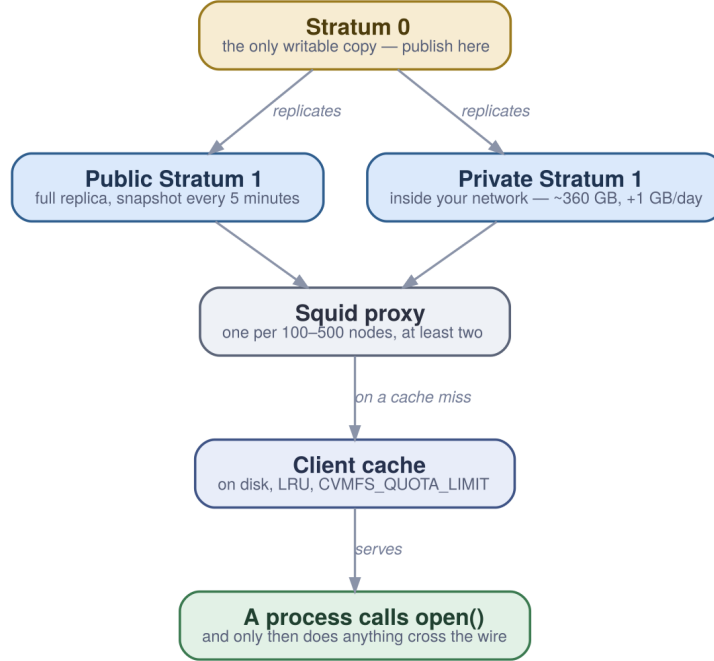


Figure 3: Publish at the top; a process calling `open()` at the bottom. A handful of clients need none of the middle. A cluster needs all of it - and the private Stratum 1 is what makes the whole arrangement survive losing the internet.

Stratum 0 - the only writable copy. One release-manager machine where publishing happens. It is not load-bearing for readers, which is exactly why it can be small and carefully controlled.

Stratum 1 - a full replica, and your disconnect insurance. Public mirrors sit in Europe, the US and Asia. A private one inside your network reduces latency, offloads the public infrastructure, and means a lost upstream link stops updates rather than stopping work. For EESSI that is roughly 360 GB today, growing about a gigabyte a day.¹³¹⁴

Proxy - the tier that makes MPI start-up bearable. A Squid proxy holds a partial cache close to the clients. The rule of thumb is one per 100-500 worker nodes and at least two for redundancy; the reason it matters most on large parallel jobs is that every rank wants the same binaries at the same instant.¹⁵¹⁶

Client cache - where almost every read is answered. On disk, LRU-evicted, bounded by `CVMFS_QUOTA_LIMIT`. Once warm it is the fastest tier by a wide margin, which is why the tuning

¹³Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

¹⁴MultiXscale / EESSI. *Best Practices for CernVM-FS in HPC*. <https://multixscale.github.io/cvmfs-tutorial-hpc-best-practices/> — proxy and Stratum 1 sizing, diskless cache configuration, and which workarounds are no longer recommended.

¹⁵Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

¹⁶MultiXscale / EESSI. *Best Practices for CernVM-FS in HPC*. <https://multixscale.github.io/cvmfs-tutorial-hpc-best-practices/> — proxy and Stratum 1 sizing, diskless cache configuration, and which workarounds are no longer recommended.

that matters most is usually “put it on the SSD and make it big enough.”

The entire client configuration is this:

```
# /etc/cvmfs/default.local
CVMFS_HTTP_PROXY="http://proxy-a.example.com:3128|http://proxy-b.example.com:3128"
CVMFS_QUOTA_LIMIT=10000          # 10 GB of local cache
CVMFS_CACHE_BASE=/ssd/cvmfs      # put it on the fastest disk the node has

# then, once:
sudo cvmfs_config reload
ls /cvmfs/software.eessi.io      # mounted on demand by autofs
```

5.1 What it scales to

For the CernVM-FS installation at CERN, as of May 2026: around **15 Stratum 1** mirror servers, more than **33 billion files** in the `/cvmfs` tree, roughly **2 PB** accessible and proven to 100 PB, and about **290 repositories** including more than 8,000 container images.¹⁷¹⁸

Why this matters. The tiers are separable. A workstation needs none of them; a hundred nodes need a proxy; an air-gapped estate needs a private Stratum 1 and nothing else changes. The operational commitment can be sized to the deployment rather than to the architecture diagram.

6 Performance: faster than the parallel filesystem you already own

This is the result that surprises people. Loading software is a small-file, high-metadata workload, and the large parallel filesystems an HPC site has bought - GPFS, Lustre - are tuned for the opposite. Importing one Python package touches roughly 3,500 files totalling about 1.1 GB, most of them tiny.¹⁹

Where the software lives	Time to import the package	Note
Local SSD or ramdisk	~2 s	The floor. Not a realistic way to run a cluster.

¹⁷Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

¹⁸Blomer, J. et al. (2024). *CernVM-FS at Extreme Scales*. CHEP 2024, EPJ Web of Conferences. <https://doi.org/10.1051/epjconf/202429504012> — what breaks first when a repository grows.

¹⁹Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

Where the software lives	Time to import the package	Note
NFS, lightly loaded	slower than local	Workable at small scale; degrades with concurrency. Even with a hot pagepool. Better when SSD-backed.
GPFS / Lustre	30-60 s	Latency-bound. The configuration to avoid.
CernVM-FS, cold cache, distant mirror, no proxy	worst case	The tiers doing what they exist for.
CernVM-FS, cold cache, near Stratum 1 + proxy	dramatically better	The steady state on any node that has run the job before.
CernVM-FS, warm client cache	approx. local disk	

Measurements from the CernVM-FS training material, May 2026.²⁰ The shape of the result - not the exact seconds - is the transferable part: caching close to the client beats a fast filesystem far from it, for this workload.

²⁰Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

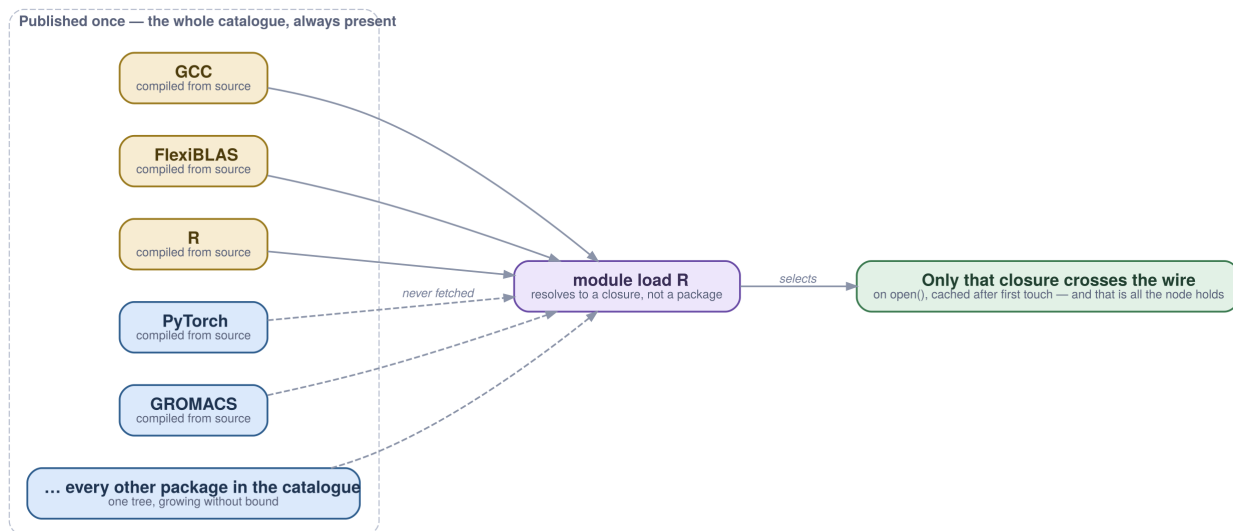


Figure 4: Why it can be both enormous and fast. A module load resolves to a closure, not to a package, and only that closure crosses the wire - lazily, cached after first touch. The catalogue can grow without bound because nothing pays for the parts it does not open.

Why this matters. It removes the trade-off the room is expecting. The usual assumption is that provenance and central control cost you speed, so the argument becomes rigour versus researchers' time. Here the architecture that gives you the audit position is also the one that starts jobs faster than the storage you already paid for.

Part III - One architecture, two implementations. *Two national programmes, a decade apart, built the same four-layer stack and filled the middle slot with different software. Both work.*

7 The architecture: four layers, and why the middle one is a role

Host operating system, filesystem layer, compatibility layer, software layer. The decomposition is stated the same way by both programmes, which is itself worth noticing - it was arrived at twice.²¹²²²³

²¹Dröge, B., Holanda Rusu, V., Hoste, K. et al. (2023). *EESSI: A cross-platform ready-to-use optimised scientific software stack*. Software: Practice and Experience 53(1), 176-210. <https://doi.org/10.1002/spe.3075> — the full architectural argument with the performance evaluation attached.

²²EESSI. *Project documentation*. <https://www.eessi.io/docs/> — paths, versions, CPU targets, host injections, GPU support and the test suite.

²³O'Cais, A., Vela, H. (2025). *EESSI layer decomposition*. HPCKP 2025. <https://www.hpckp.org/> — a compact statement of the four-layer decomposition.

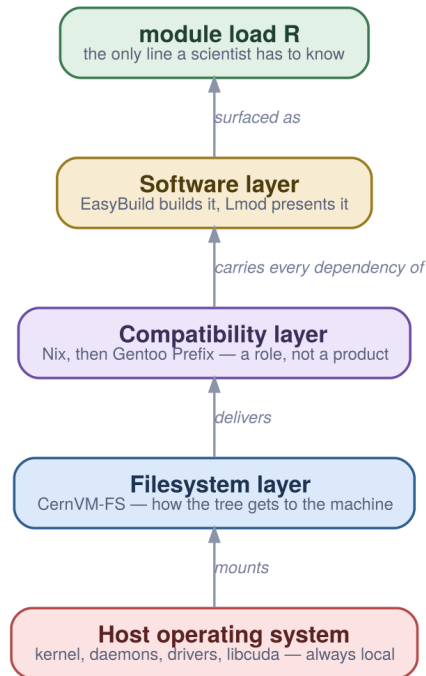


Figure 5: Read bottom to top: the host keeps the kernel and the drivers and gives up nothing else; CernVM-FS delivers; the compatibility layer removes the host’s userland from the picture; the software layer is what a person selects from.

Host OS - smaller than you expect. Kernel, daemons, drivers, libcuda, anything privileged. That is the whole list. It is short deliberately: every item on it is something that has to be true per machine and can therefore differ between machines.

Filesystem layer - how the tree arrives. CernVM-FS in both implementations. This is the layer Part II was about, and the one neither programme has ever swapped.

Compatibility layer - the interesting one. A complete userland - its own loader, its own glibc, its own coreutils - installed in a non-standard prefix. Everything above depends on it and nothing above depends on the host’s /usr. This is the layer that makes “runs on RHEL 8 and on a current laptop” a property rather than an aspiration.

Software layer - the part with the science in it. Every scientific application, built against the compatibility layer by EasyBuild, presented by Lmod, and built more than once - one subtree per CPU microarchitecture.

7.1 The observation that makes this worth adopting

Canada filled the compatibility slot with Nix. Later they replaced it with Gentoo Prefix. EESSI chose Gentoo Prefix from the start.²⁴²⁵²⁶ Three configurations, two programmes, one working architecture each time - and critically, the layers above and below the swapped one did not change when it was swapped.

Why this matters. It converts a technology bet into an architectural choice. Adopting this model does not require being right about which prefix implementation wins, because the slot has already been refilled once in production without disturbing anything else. That is a materially different risk profile from picking a vendor.

8 The compatibility layer: a complete userland that is not the host's

A prefix is a full operating system userland installed somewhere other than / - glibc, coreutils, bash, awk, grep, make, autotools, binutils, OpenSSL, and a compiler - built from source, without root.²⁷ Software above it links against it and never against the host, so the host's age stops being a constraint on anything except the kernel.

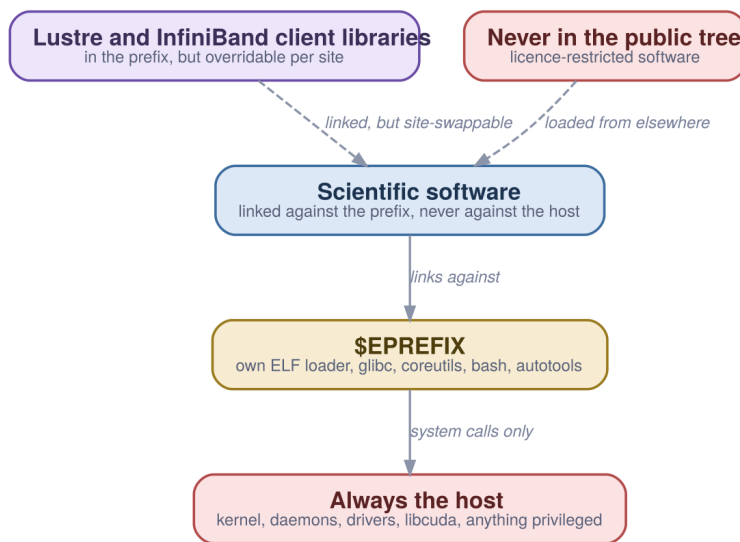


Figure 6: Where the boundary actually falls. The two red boxes are what can never come off the tree; the purple one is the honest grey area - libraries that live in the prefix but are properties of the site's hardware, and so have to stay swappable at runtime.

²⁴Compute Canada (2018). *Combining CVMFS, Nix, Lmod and EasyBuild*. FOSDEM 2018. <https://archive.fosdem.org/2018/> — the same architecture with a different compatibility layer.

²⁵Oldeman, B. et al. (2023). *Digital Research Alliance of Canada site talk: lessons learned, new developments*. EasyBuild User Meeting 2023. <https://easybuild.io/eum23/> — the guiding principle, the distribution-version comparison, and the Nix post-mortem.

²⁶EESSI. *Project documentation*. <https://www.eessi.io/docs/> — paths, versions, CPU targets, host injections, GPU support and the test suite.

²⁷Gentoo Project. *Gentoo Prefix*. <https://wiki.gentoo.org/wiki/Project:Prefix> — the compatibility layer implementation, including the bootstrap.

The mechanism is a custom ELF interpreter. The prefix ships its own dynamic loader and every binary in it is linked to use that one instead of the host's. This is the single technical fact the whole layer rests on, and it is also the source of most of the friction below.

It carries dependencies, not science. Bash, Git, Vim, X11, TeX Live - the things nobody publishes a paper about and everything breaks without. Newer than the enterprise distribution ships, which is the point.²⁸

The grey area is named, not hidden. Lustre client libraries and InfiniBand or OmniPath libraries are dependencies of the MPI in the tree, but they are also properties of the site's hardware. They live in the prefix and can be overridden per site at runtime.²⁹

Anything privileged stays on the host. The kernel, daemons, drivers, `libcuda`, `sudo`. Also anything legally restricted - licensed software sits in a separate, restricted repository rather than in the public tree.³⁰

8.1 Why Nix was replaced, in their words

Canada ran a single read-only Nix profile for the 2016 and 2018 environments and moved to Gentoo Prefix for 2020. The post-mortem is worth reading because it is a failure of interaction rather than of the tool.³¹³²

Store hashes leaked upward. Despite wrapping the linker, Nix store paths found their way into EasyBuild-compiled software through CMake, qmake and Python virtualenvs. A path that encodes a hash is only immutable if nothing copies it somewhere that outlives it.

And garbage collection was destructive. Upgrading a component changes its store hash, which is the feature; collecting the old one out from under something that had recorded the old path is the consequence. The verdict was that Nix is better used as a top layer with a writable store, and that HPC users know environment modules rather than Nix's tools.

Gentoo Prefix: no symlinks, no store leak, minimal solution.

— Digital Research Alliance of Canada, on the 2020 migration

²⁸Oldeman, B. et al. (2023). *Digital Research Alliance of Canada site talk: lessons learned, new developments*. EasyBuild User Meeting 2023. <https://easybuild.io/eum23/> — the guiding principle, the distribution-version comparison, and the Nix post-mortem.

²⁹Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

³⁰Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

³¹Oldeman, B. et al. (2023). *Digital Research Alliance of Canada site talk: lessons learned, new developments*. EasyBuild User Meeting 2023. <https://easybuild.io/eum23/> — the guiding principle, the distribution-version comparison, and the Nix post-mortem.

³²Dolstra, E., de Jonge, M., Visser, E. (2004). *Nix: A Safe and Policy-Free System for Software Deployment*. Proc. LISA XVIII, USENIX. — what the store model guarantees, which is what makes the leak described in section 8 legible.

8.2 What it actually costs to run one

The prefix is the layer that generates support tickets, and a page that omits them is not useful to anyone who has to operate this. All five of these are documented by the people running it in production.³³

1. **Downloaded binaries have to be patched.** A vendor binary is linked against the host's loader and will not run. A script walks them and rewrites the interpreter with `patchelf --set-interpreter`. It works, and it is a step that has to exist.
2. **A stale line in a user's .bashrc breaks everything.** `LD_LIBRARY_PATH=/usr/lib64`, usually set years ago by accident, puts the host's libraries ahead of the prefix's and most tools stop working. The diagnosis is easy once and expensive the first fifty times.
3. **Anaconda and Julia ship binaries that do not always work.** Both distribute pre-built components that assume a standard layout. The Canadian answer is a wheelhouse - more than seventeen thousand Python wheels compiled against the stack - and actively discouraging Anaconda, because a user who installs it can end up with their own unoptimised Open MPI.
4. **The host writes to /var and the prefix reads from \$EPREFIX/var.** So who and last read an empty file. Fixed by patching `paths.h` rather than by symlinking - a small thing, and a good example of the class.
5. **There is one runtime C++ library, centrally.** `libstdc++` comes from the compatibility layer rather than from each compiler build. That is slightly off-piste from upstream, and it is what lets builds against different GCC versions collapse to one module level instead of multiplying.

Why this matters. This is the layer to evaluate honestly before adopting, because it is where the work is. The upside is total: the host's age stops constraining anything above the kernel. The cost is a small, well-documented set of sharp edges that every site hits in the same order - which is a much better position than a set nobody has written down.

9 Build and selection: EasyBuild records it, Lmod makes it choosable

These two are usually named together and do entirely different jobs. EasyBuild turns a build into a recorded, repeatable specification. Lmod turns ten thousand build permutations into something a person can navigate.³⁴³⁵³⁶ Neither idea is new - environment modules date to 1991³⁷ and Lmod is the Lua re-implementation that made a hierarchy practical³⁸ - which is worth knowing, because it means the interface a researcher learns here is one they have probably already met.

³³Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

³⁴Geimer, M., Hoste, K., McLay, R. (2014). *Modern Scientific Software Management Using EasyBuild and Lmod*. Proc. HUST'14, IEEE, 41-51. <https://doi.org/10.1109/HUST.2014.8> — why the two are always named together.

³⁵EasyBuild community. *EasyBuild documentation*. <https://docs.easybuild.io> — easyconfigs, easyblocks, toolchains and the easyconfig archive.

³⁶TACC. *Lmod documentation*. <https://lmod.readthedocs.io> — the hierarchy, MODULEPATH manipulation, and spider versus avail.

³⁷Furlani, J. L. (1991). *Modules: Providing a Flexible User Environment*. Proc. LISA V, USENIX, 141-152. — the original environment-modules concept the whole selection layer still rests on.

³⁸McLay, R., Schulz, K. W., Barth, W. L., Minyard, T. (2011). *Best Practices for the Deployment and Management of Production HPC Clusters*. SC'11, ACM. <https://doi.org/10.1145/2063348.2063360> — the origin of Lmod.

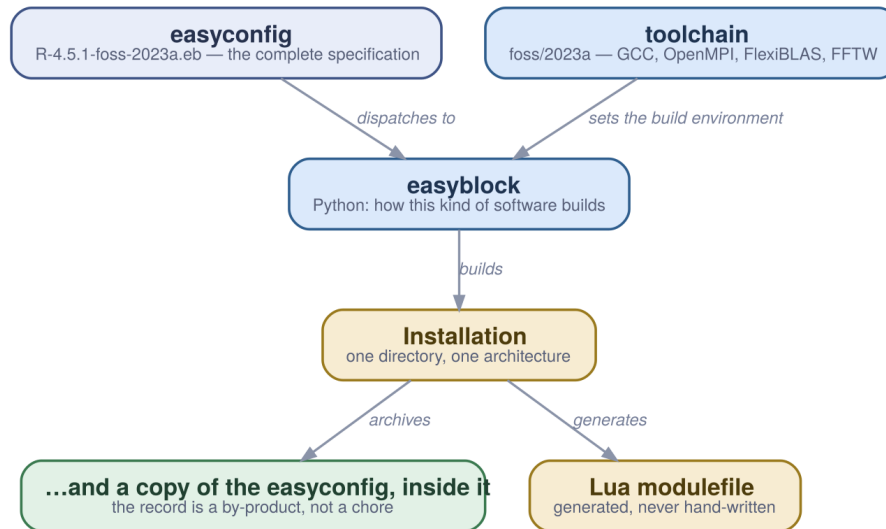


Figure 7: Three inputs, one installation - and the detail that carries the reproducibility claim: the easyconfig is copied into the install directory and into an archive on every successful build. The record is a by-product of installing, not something anyone has to remember to write down.

easyconfig - the complete specification. A plain text file of key-value assignments naming the sources, the checksums, the toolchain, the options and the dependencies. One easyconfig yields one installation and one computed module name.

easyblock - how this kind of software builds. Python that implements the build procedure. Generic ones cover CMake or autotools; software-specific ones exist where a package insists on being unusual. The unusualness lives here, once, instead of in everyone’s notes.

toolchain - the compiler and libraries as one versioned thing. foss/2023a is GCC, Open MPI, a BLAS and FFTW, pinned together and released as a generation. Naming the generation is what makes “built the same way” a checkable statement rather than a recollection.

And the easyconfig is kept, inside the installation. This is the part worth insisting on. Reproducing a build is reading a file that shipped with it, not reconstructing what someone did. Provenance is the mechanism’s output rather than anyone’s discipline.

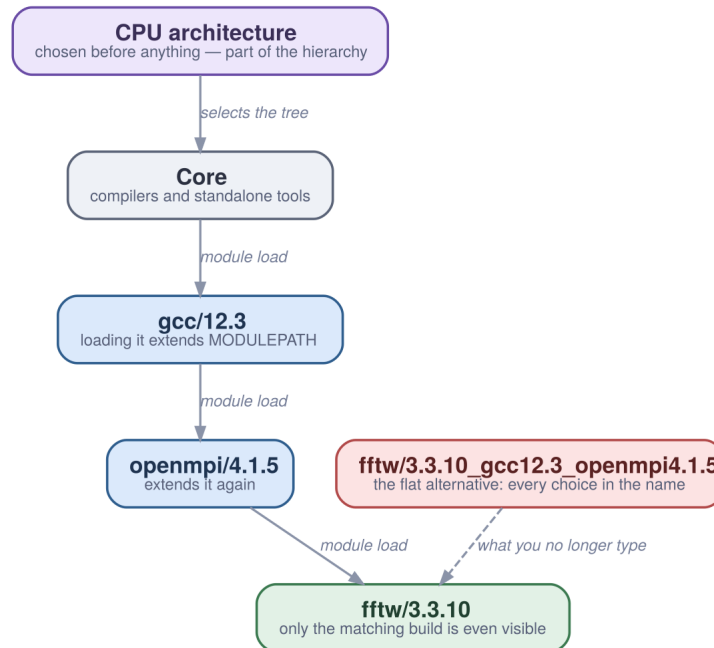


Figure 8: Why the hierarchy is not decoration. Loading a compiler extends MODULEPATH, which is what makes the matching builds visible and the mismatched ones unreachable. The red box is the name you would otherwise have to type - and get right.

```

$ module load fftw
Lmod has detected the following error: These module(s) exist but cannot be
loaded as requested: "fftw"
Try: "module spider fftw" to see how to load the module(s).

$ module spider fftw/3.3.10
You will need to load all module(s) on one of the lines below first:
gcc/12.3 openmpi/4.1.5

$ module load gcc/12.3 openmpi/4.1.5 fftw/3.3.10 # and now it is visible
  
```

Incompatible combinations are unreachable, not discouraged. In a flat layout, conflicts are declared and become fragile as compilers are added. In a hierarchy, loading a compiler is what makes the builds against it appear at all - so a user cannot assemble a combination that was never built.

Architecture belongs in the hierarchy, not in the name. This is the Canadian design decision worth copying. Around a thousand applications across four CPU generations and four GPU generations produce ten thousand or more permutations. Put the architecture in the tree and a user sees `fftw/3.3.10`. Put it in the name and they see the permutation count.³⁹⁴⁰

³⁹Boissonneault, M., Oldeman, B., Taylor, R. (2019). *Providing a Unified Software Environment for Canada's National Advanced Computing Centers*. PEARC'19. <https://doi.org/10.1145/3332186.3332210> — the peer-reviewed statement of the whole model. If you read one thing on this list, read this.

⁴⁰Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://>

EasyBuild is not the only credible answer in this layer. **Spack** solves the same problem with a different model - a dependency solver and per-installation hashes rather than named toolchain generations - and is worth evaluating on its own terms.⁴¹ Both stacks described here chose EasyBuild, and the architecture around it does not depend on that choice: the build layer is as replaceable as the compatibility layer turned out to be.

Why this matters. Together these are what stop the catalogue's complexity from reaching the person using it. The build side records everything; the selection side hides everything that is not a choice. Get either half wrong and the stack is technically sound and practically unusable.

10 The Canadian stack: the one that has been in production the longest

Canada's national research systems have run a single shared software stack across every site since 2017: one catalogue, one scheduler, one way of installing things, administered nationally and mounted everywhere.⁴²⁴³ It is the reference implementation, and it is public - you can mount it from anywhere today.

```
# the Alliance stack - public, mountable anywhere
source /cvmfs/soft.computecanada.ca/config/profile/bash.sh
module load StdEnv/2023
```

```
# EESSI - in the default CernVM-FS configuration since November 2023
source /cvmfs/software.eessi.io/versions/2023.06/init/bash
module load R/4.3.2-gfbf-2023a
```

```
# in both cases the next line is the whole user interface
which R
```

Two layers, two paths. The EasyBuild layer under `easybuild/{modules,software}/2023` and the compatibility layer under `gentoo/2023/x86-64-v3`. Loading the `gentoo/2023` module sets `$EPREFIX` and everything above resolves from there.

Architecture is a directory, not a build flag. The tree carries `x86-64-v3` and `x86-64-v4` subtrees today, and carried `sse3`, `avx`, `avx2` and `avx512` historically.

`//users.ugent.be/~kehoste/eum25/` — the current path layout, `StdEnv/2023`, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

⁴¹Gamblin, T. et al. (2015). *The Spack Package Manager: Bringing Order to HPC Software Chaos*. Proc. SC'15, ACM. <https://doi.org/10.1145/2807591.2807623> — same problem, different answer, and a real one.

⁴²Boissonneault, M., Oldeman, B., Taylor, R. (2019). *Providing a Unified Software Environment for Canada's National Advanced Computing Centers*. PEARC'19. <https://doi.org/10.1145/3332186.3332210> — the peer-reviewed statement of the whole model. If you read one thing on this list, read this.

⁴³Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, `StdEnv/2023`, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

Site variables, not site forks. `CC_CLUSTER`, `RSNT_ARCH`, `RSNT_INTERCONNECT` and `RSNT_CUDA_DRIVER_VERSION` are how a mount is adapted to a site. What varies between destinations is the configuration, never the software.⁴⁴

Python is served as wheels, not as an environment manager. More than seventeen thousand wheels compiled against the stack, with an `avail_wheels` command to search them. It is the pragmatic answer to the single largest source of unreproducible scientific software.⁴⁵

10.1 The version lineage

Standard environments are versioned and the old ones stay mountable. That is what lets a result from 2018 be re-run without arguing about what 2018 meant.⁴⁶

Environment	Compiler	MPI	What changed
StdEnv/2016.4	GCC 5.4 / Intel 2016.4	Open MPI 2.1.1	The first unified environment.
StdEnv/2018.3	GCC 7.3 / Intel 2018.3	Open MPI 3.1.2	First AVX-512 support.
StdEnv/2020	GCC 9.3 / Intel 2020.1	Open MPI 4.0.3	Compatibility layer moved from Nix to Gentoo Prefix.
StdEnv/2023	GCC 12.3 / Intel 2023.1	Open MPI 4.1.5	GCC becomes the default compiler; minimum AVX2; FlexiBLAS; CUDA 12. Default since April 2024.

10.2 How something gets into it

The promotion path is the part most worth copying, because it is already a qualification pipeline whether or not anyone calls it one. Nothing is promoted by being edited - each stage publishes into the next and is tested there.⁴⁷

⁴⁴Digital Research Alliance of Canada. *Technical documentation: Accessing CVMFS, Standard software environments, Using modules*. <https://docs.alliancecan.ca> — the StdEnv lineage and the site variables, described from the user's end.

⁴⁵Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

⁴⁶Digital Research Alliance of Canada. *Technical documentation: Accessing CVMFS, Standard software environments, Using modules*. <https://docs.alliancecan.ca> — the StdEnv lineage and the site variables, described from the user's end.

⁴⁷Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

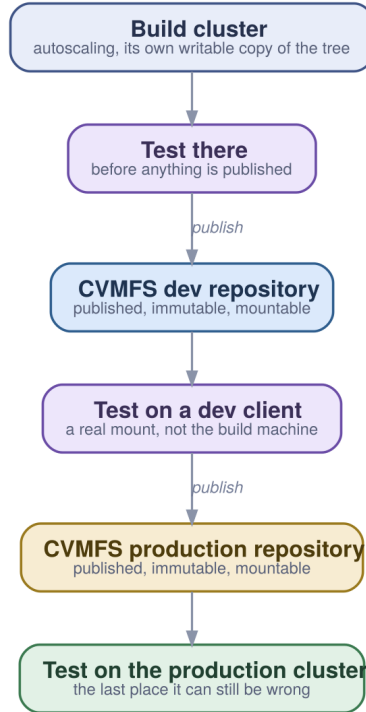


Figure 9: A one-way street. The build cluster is disposable and rebuildable; the dev repository is a real, immutable, mountable publication that happens not to be the production one. The last test is on the production cluster, because that is the last place it can still be wrong.

Why this matters. Everything here has been operating at national scale for years, across systems that do not share administrators, for around eighteen thousand researchers. Adopting it is not a research project - and the parts that are hard are hard in ways that have already been written down.

11 EESSI: the same architecture, rebuilt for machines nobody had in 2017

The European Environment for Scientific Software Installations is explicitly modelled on the Canadian stack and re-implemented for targets the original never had: Arm, RISC-V, cloud instances, laptops.⁴⁸⁴⁹⁵⁰ Its repository has been part of the default CernVM-FS configuration since November 2023, which means a client with no site-specific setup can already see it.

One repository, versioned as a whole. `versions/2023.06` is the current default (foss 2022b through 2023b); `versions/2025.06` is available and becoming default. Releases are yearly, and old

⁴⁸Dröge, B., Holanda Rusu, V., Hoste, K. et al. (2023). *EESSI: A cross-platform ready-to-use optimised scientific software stack*. Software: Practice and Experience 53(1), 176-210. <https://doi.org/10.1002/spe.3075> — the full architectural argument with the performance evaluation attached.

⁴⁹EESSI. *Project documentation*. <https://www.eessi.io/docs/> — paths, versions, CPU targets, host injections, GPU support and the test suite.

⁵⁰O’Cais, A., Vela, H. (2025). *EESSI layer decomposition*. HPCKP 2025. <https://www.hpckp.org/> — a compact statement of the four-layer decomposition.

versions stay mounted.

The compatibility layer is visible in the path. `compat/linux/x86_64/` carries its own `libc.so.6` and its own dynamic loader. That is not an implementation detail you have to take on trust - it is a directory you can list.

The software layer is one subtree per microarchitecture. `software/linux/x86_64/amd/zen4/` and its siblings, selected at login by `archspec` or the pure-bash `archdetect`.⁵¹ Each carries both the binaries and the modulefiles, so selecting an architecture selects a whole consistent world rather than a flag.

Validated by a test suite, not by assertion. The EESSI test suite is built on ReFrame and exercises real applications - GROMACS, LAMMPS, OpenFOAM, PyTorch - plus the underlying BLAS. A site runs it against its own mount to confirm the deployment works and performs.⁵²⁵³

Architecture	Targets
x86_64 / AMD	zen2, zen3, zen4, zen5
x86_64 / Intel	haswell, skylake_avx512, cascadelake, icelake, sapphirerapids
aarch64	neoverse_n1, neoverse_v1, a64fx, nvidia/grace
generic	x86_64 and aarch64 fallbacks, for anything unrecognised
riscv64	in a separate development repository

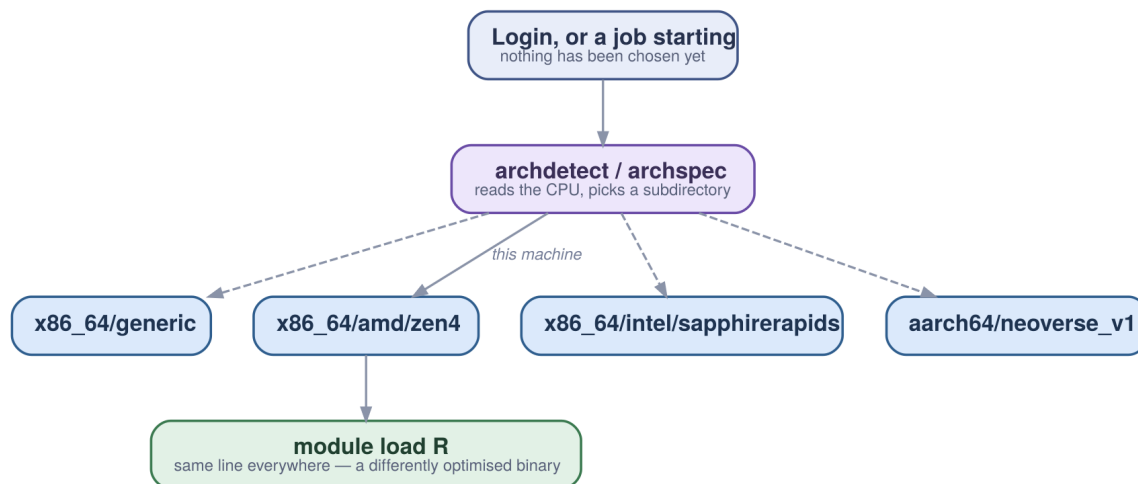


Figure 10: The tree carries all of them and the machine picks. `archspec` or `archdetect` reads the CPU at login and selects a subdirectory, so “we moved to Arm” stops being a recompile and a revalidation and becomes the same module load resolving somewhere else.

⁵¹Archspec contributors. *archspec*. <https://github.com/archspec/archspec> — microarchitecture detection driving the subdirectory choice.

⁵²EESSI. *Project documentation*. <https://www.eessi.io/docs/> — paths, versions, CPU targets, host injections, GPU support and the test suite.

⁵³Karakasis, V. et al. *ReFrame*. <https://reframe-hpc.readthedocs.io> — the validation framework behind the EESSI test suite.

11.1 The seam: host injections

A read-only global tree cannot contain a GPU driver. The driver belongs to one machine, and NVIDIA’s licence does not permit redistributing it anyway. EESSI names this seam explicitly rather than working around it quietly.⁵⁴

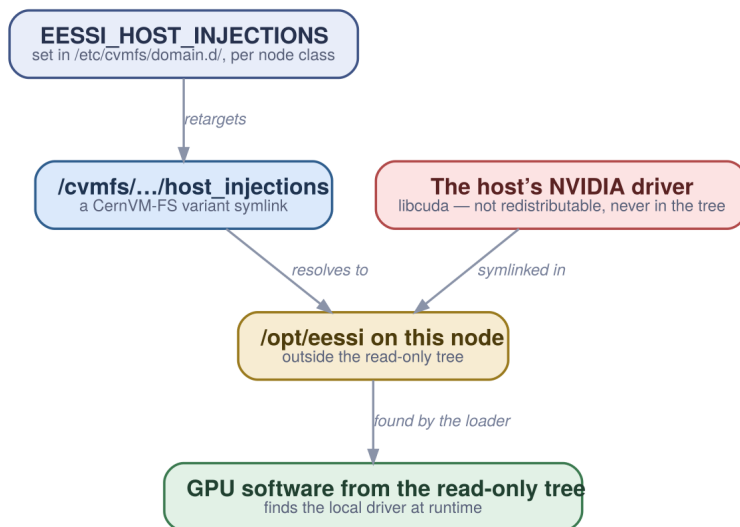


Figure 11: A CernVM-FS variant symlink: a path inside the read-only repository that resolves to a location on the client. Point it at a per-node-class directory and one tree serves nodes with different drivers without the tree knowing anything about them.

Runtime libraries ship; the toolkit does not. The CUDA licence permits redistributing runtime libraries only, so the CUDA and cuDNN installations in the tree are stripped to those. `nvcc` is a symlink into a locally installed SDK - you can run GPU software with nothing installed, and compile it with an SDK installed once.

And it now flows back the other way. The Alliance’s Gentoo Prefix bootstrap is based on EESSI’s Ansible implementation, and the Alliance has adopted the variant-symlink approach for CUDA driver libraries. The influence is no longer one direction.⁵⁵

Why this matters. The method has transferred once already, between programmes that do not share staff, funding or hardware - and is now being transferred back. That is the difference between a method and one team’s preference.

Part IV - Running it, and what it does not do.

⁵⁴EESSI. *Project documentation*. <https://www.eessi.io/docs/> — paths, versions, CPU targets, host injections, GPU support and the test suite.

⁵⁵Oldeman, B. et al. (2025). *Digital Research Alliance of Canada site talk*. EasyBuild User Meeting 2025. <https://users.ugent.be/~kehoste/eum25/> — the current path layout, StdEnv/2023, the wheelhouse, the build cluster, the promotion ladder, and the prefix-friction list.

12 Running it: what you actually stand up, and in what order

The commitment scales with the deployment. A workstation is one package and one config file. A cluster is that plus two proxies. An estate that cannot lose access when a link goes down is that plus one replica.⁵⁶⁵⁷⁵⁸

1. **Install the client and point it somewhere.** A package, a public key, and a config file naming the proxy and the cache. Repositories are mounted on demand by autofs; there is no service to start per repository.
2. **Put two proxies in front of it, once you have nodes.** One per 100-500 workers, at least two for redundancy, on SSD, on a fast link to the clients. The client config takes a failover list, so losing one is not an outage.
3. **Add a private Stratum 1 if a lost link would stop work.** One replica, one Apache, one cron job. Disable the Geo API - it is for public mirrors. Expect hours for the first synchronisation and minutes thereafter.
4. **Deal with the nodes that have no local disk.** A loopback file on the shared filesystem is the recommended answer, sized about 15% above the cache so metadata operations stay inside it. RAM cache works when there is memory to spare. Re-exporting a single client over NFS does not - it can be made to work and it runs into operational problems.
5. **Monitor the two things that actually predict trouble.** Cache cleanup frequency - if evictions are more frequent than a typical job is long, the cache is too small - and the client's current revision against the Stratum 1's, which is how you see replication lag before a user does.

```
# a private Stratum 1: one replica, inside your own network
sudo dnf install -y cvmfs-server python3-mod_wsgi
echo 'CVMFS_GEO_DB_FILE=NONE' | sudo tee -a /etc/cvmfs/server.local

sudo cvmfs_server add-replica -o $USER \
    http://aws-eu-west-s1-sync.eessi.science/cvmfs/software.eessi.io \
    /etc/cvmfs/keys/eessi.io

cvmfs_server snapshot software.eessi.io          # first sync: hours
*/5 * * * * cvmfs_server snapshot -a -i         # thereafter: minutes

# no local disk? a loopback file on the shared filesystem, per node -
# sized ~15% above the cache, so metadata stays inside the loopback
CVMFS_CACHE_BASE=/var/lib/cvmfs
```

⁵⁶CernVM-FS project. *CernVM-FS documentation*. <https://cvmfs.readthedocs.io> — manifest fields, catalog structure, cache behaviour, publishing transactions, garbage collection and DUCC. The `cpt-details` chapter is the one to read.

⁵⁷Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

⁵⁸MultiXscale / EESSI. *Best Practices for CernVM-FS in HPC*. <https://multixscale.github.io/cvmfs-tutorial-hpc-best-practices/> — proxy and Stratum 1 sizing, diskless cache configuration, and which workarounds are no longer recommended.

```
# or, if there is memory to spare and the workload is small:
CVMFS_CACHE_BASE=/dev/shm/cvmfs-cache
CVMFS_QUOTA_LIMIT=4000

# no root at all? mount it unprivileged, per process:
git clone https://github.com/cvmfs/cvmfsexec && cd cvmfsexec && ./makedist default
./cvmfsexec software.eessi.io -- /bin/bash -l
```

12.1 The four error messages, decoded

It is a filesystem, so it can only report errors the way a filesystem can. These four cover most of what a site will actually see, and none of them says what it means.⁵⁹

What it says	What it means
Too many levels of symbolic links	A mount point reached inside a namespace that is not shared. Almost always a container - mount <code>/cvmfs</code> with the shared propagation flag.
No such file or directory	On a path that should exist: a catalog could not be loaded into the cache. The real reason is in the syslog.
Transport endpoint is not connected	The client process died and the watchdog did not restart it. Unmount and remount.
Software caused connection abort	The kernel cut the FUSE connection, usually through administrator action. Remount.

```
cvmfs_config probe software.eessi.io      # is it reachable at all
cvmfs_config stat -v software.eessi.io    # revision, cache use, hit rate
cvmfs_config showconfig software.eessi.io # what it thinks it was told
cvmfs_talk -i software.eessi.io ncleanup24 # cache evictions in 24h
sudo cvmfs_config wipecache               # last resort, not first
```

Why this matters. None of this is exotic. It is a package, a config file, a Squid instance and an Apache instance - components a platform team already operates, arranged in a way that has been documented by people running it at national scale.

13 The boundary: what this does not solve

Everything above is worth adopting and none of it is complete. These are the limits a reviewer will find anyway, and finding them here first is cheaper than finding them in month three.

⁵⁹Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

Updates are visible within minutes, not instantly. Clients re-check on the catalog time-to-live. That is fast enough that no rollout is needed and slow enough that “immediately” is the wrong word to use in a change record.

The signing whitelist is an availability dependency. It expires and has to be refreshed. That is a good property - it is what bounds the damage from a compromised key - but it is one more thing that can stop a mount working, and it belongs on the runbook.

Unprivileged mounting is possible, not free. `cvmfsexec` needs user namespaces or a suitable Apptainer, and gives each process its own cache rather than a shared one.⁶⁰

EasyBuild does not sandbox the build. Which means bit-for-bit reproducibility is a target rather than a guarantee. What you get is a complete, archived record of how the build was specified - strictly stronger than a lock file, strictly weaker than a hermetic build system.

Licensed software does not fit the model. Anything that cannot be redistributed stays in a restricted repository or on the host. GPU drivers are the common case and are handled by variant symlinks; commercial applications are handled by access control, which is a different kind of answer.

Derivation proves construction, never meaning. Knowing exactly how a binary was built tells you nothing about whether the analysis it ran was correct. This model closes one class of doubt completely and leaves the scientific ones exactly where they were.

The one that catches people. If your nodes have no local disk and you are tempted to run a single client and export it over NFS: it can be made to work, it is slow, and it is the configuration that generates the most operational problems. Use a loopback cache file on the shared filesystem instead.⁶¹⁶²

Part V - What it settles.

14 The consequences: four questions that stop being open

Not benefits - consequences. Each of these follows from the arrangement rather than from anyone remembering to do something, which is the property that makes it survive the people who built it.

⁶⁰Dykstra, D. *cvmfsexec*. <https://github.com/cvmfs/cvmfsexec> — unprivileged mounting, with its four modes and their trade-offs.

⁶¹Völkl, V., Christodoulis, G., Blomer, J., with Hoste, K. et al. (2026). *Introduction to CernVM-FS*. EESSI webinar, 18 May 2026. <https://www.eessi.io/docs/training-events/> — current scale figures, the software start-up measurements, and the error-message decoding.

⁶²MultiXscale / EESSI. *Best Practices for CernVM-FS in HPC*. <https://multixscale.github.io/cvmfs-tutorial-hpc-best-practices/> — proxy and Stratum 1 sizing, diskless cache configuration, and which workarounds are no longer recommended.

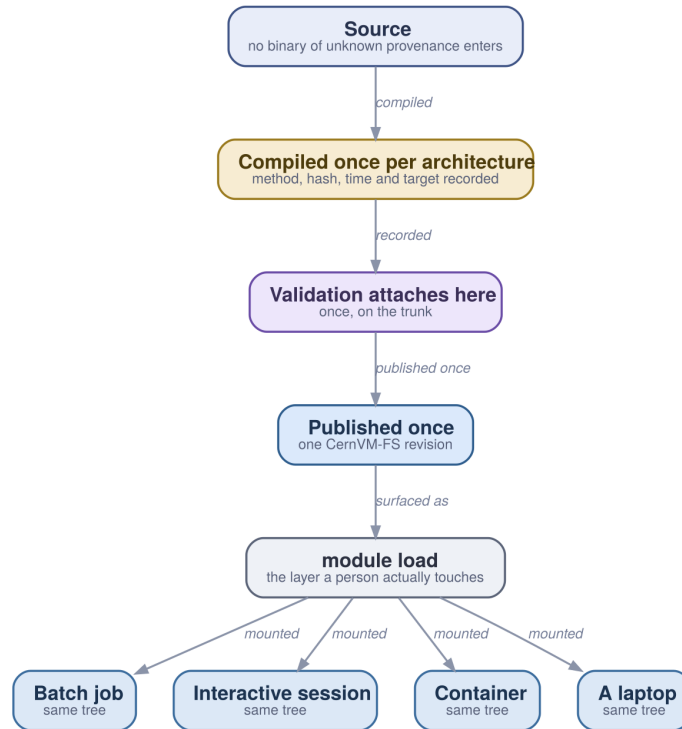


Figure 12: Validation attaches to the trunk, before the fan-out. Every destination downstream inherits one qualification rather than negotiating its own, and what differs between them is the mount, never the software.

“Which version is this?” has one answer. A named environment resolves to a specific published path on a specific revision. Ask every place in the estate whether they are offering the same R 4.5.1 - same source, same compiler, same BLAS, same bytes - and there is now a way for the answer to be yes.

Approval attaches once, on the trunk. Where a validation process exists, it applies to the published tree rather than to each destination. A second approval path is not redundancy; it is two things to keep in agreement forever.

Reproducing an old result is a mount, not a rebuild. Old revisions stay published and old environments stay mountable. The difference between minutes and a week, and between evidence and an assertion.

Changing architecture stops being a project. Arm instances are materially cheaper for a lot of analytical work, and moving normally means a recompile, a revalidation and a second environment to defend. Here it is the same line resolving to a differently optimised build of the same thing.

None of this is novel and all of it is in production. The useful conversation is not whether the model works - two national programmes have answered that - but *which parts of it fit the estate you already have*.

15 References