News from the CernVM-Filesystem

2.14 + plans for 2.15

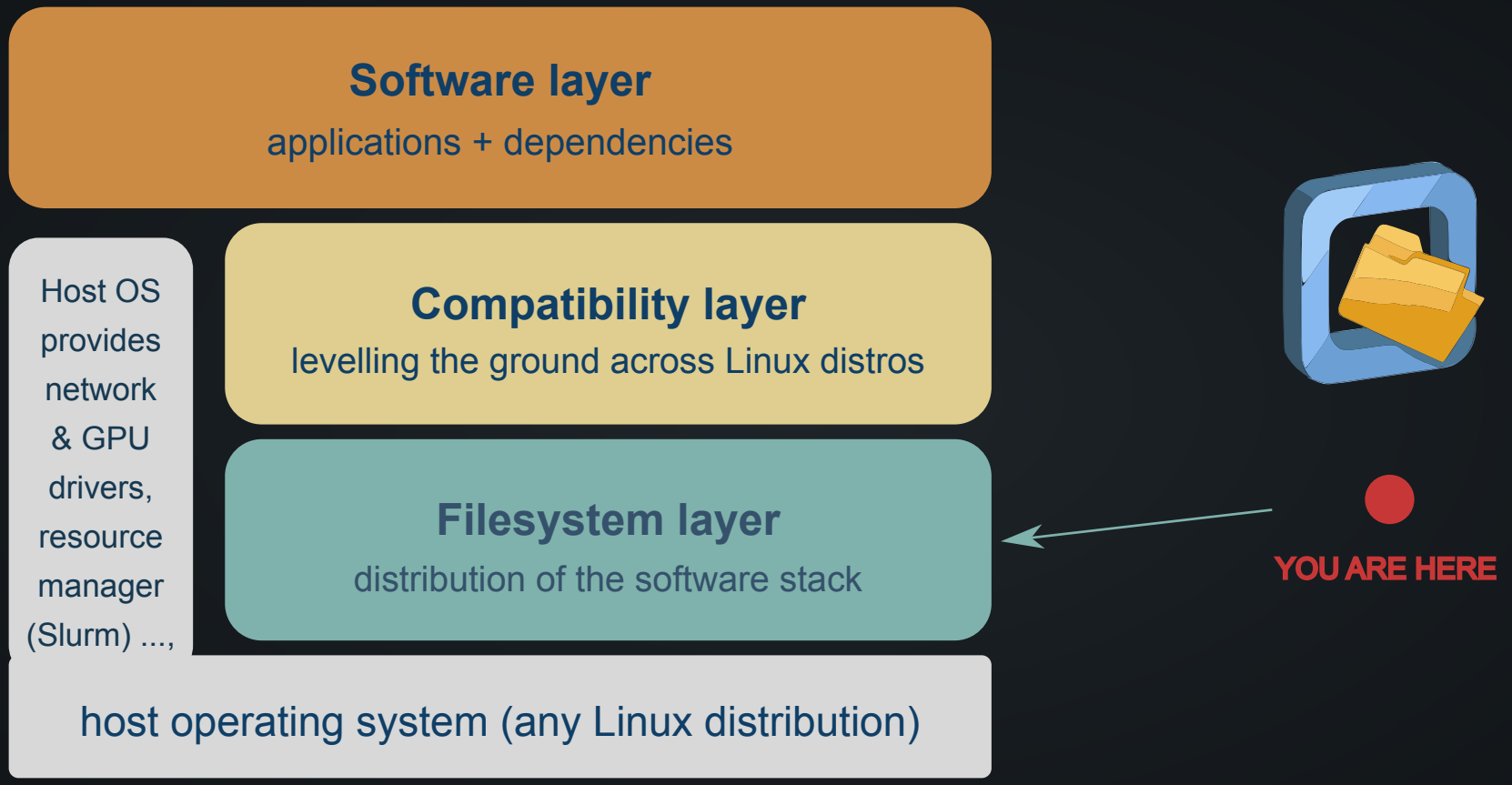
Partial Replication, K8s deployments, and more

April 23rd

EasyBuild Users Meeting 2026

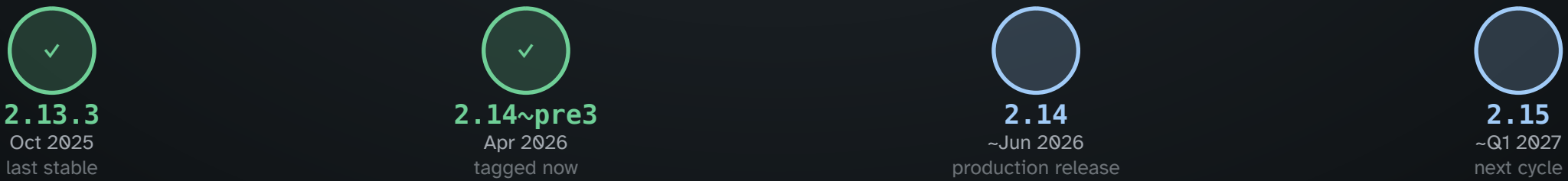
Valentin Volkl (valentin.volkl@cern.ch)

EP-SFT
Software Frameworks And Tools


Software layer
applications + dependencies
Host OS provides network & GPU drivers, resource manager (Slurm) ...,
Compatibility layer
levelling the ground across Linux distros
Filesystem layer
distribution of the software stack
host operating system (any Linux distribution)
YOU ARE HERE

ToC

Release timeline



CernVM-Filesystem

```
[user@host ~]$ python hello.py
-bash: python: command not found
Did you mean to install 'python'
from package 'python'?
```

```
[user@host ~]$ source /cvmfs/
software.eessi.io/versions/2023.06/
init/bash
{EESSI 2023.06} [user@host ~]$
python hello.py
Hello, EESSI!
```

The **CernVM-Filesystem** is an **on-demand**, **read-only** filesystem for **software distribution**.

Software is fetched transparently from a content-addressed server — only the files actually opened are downloaded.

Client vs. Server

CVMFS Client

The **cvmfs2** process — a **filesystem in user space** (FUSE).

Mounts a repository under `/cvmfs/<repo>` and fetches files on-demand over HTTP.

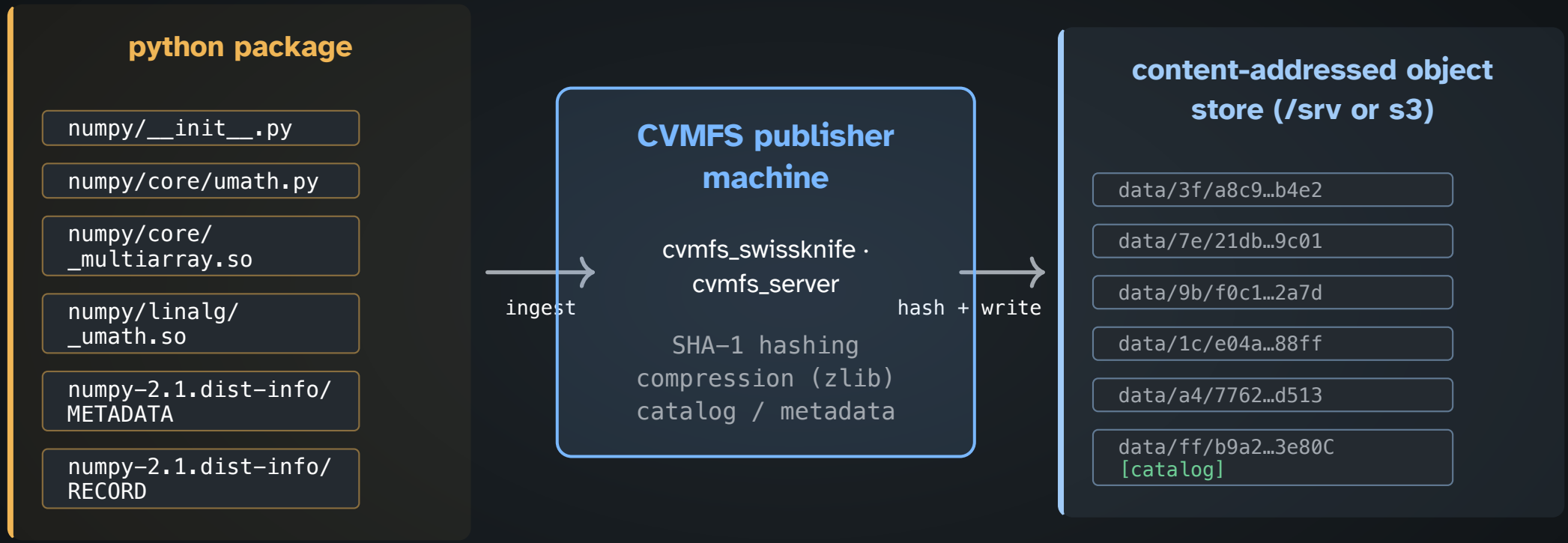
Runs on every worker node that needs the software.

CVMFS Server

The tools used to **publish** new software into the **object store** that backs the filesystem.

Typically a few machines per repository

CVMFS server tools — publishing a package

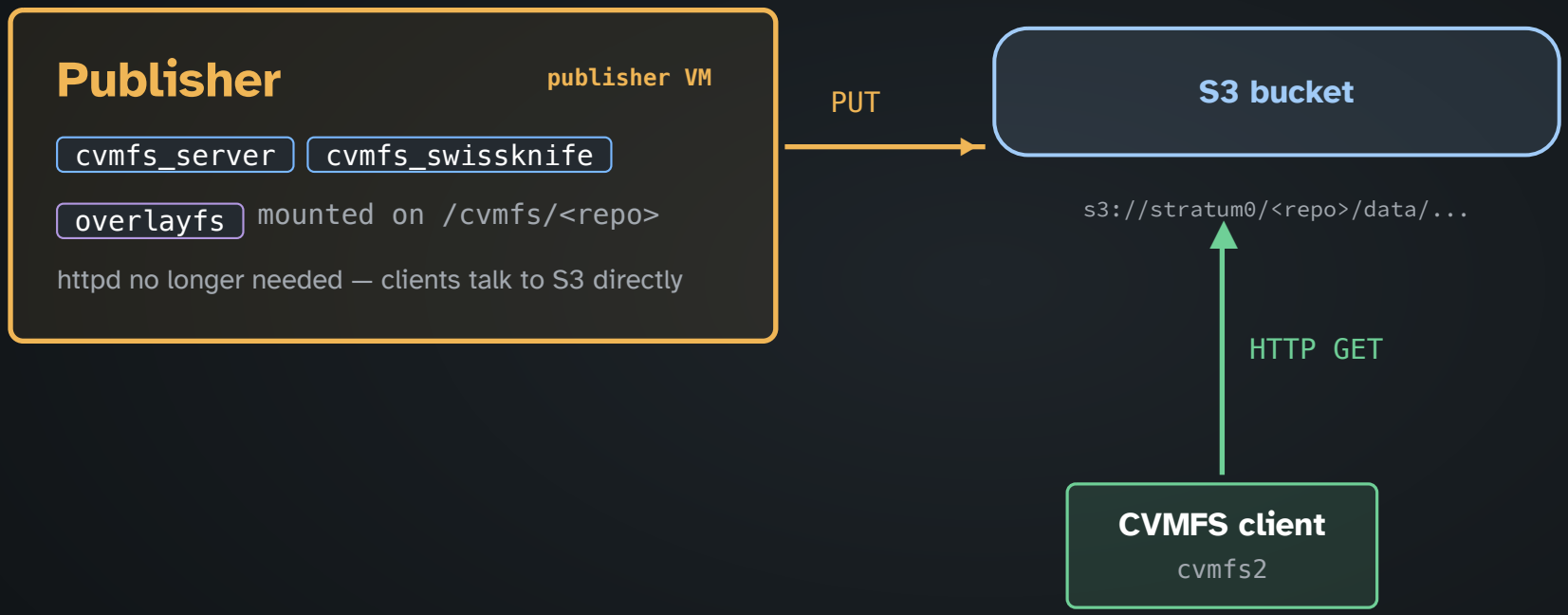


Each file → SHA-1 → stored once under `/data/<xx>/<rest>` the catalog records paths → hashes.

Deployment — classic single VM

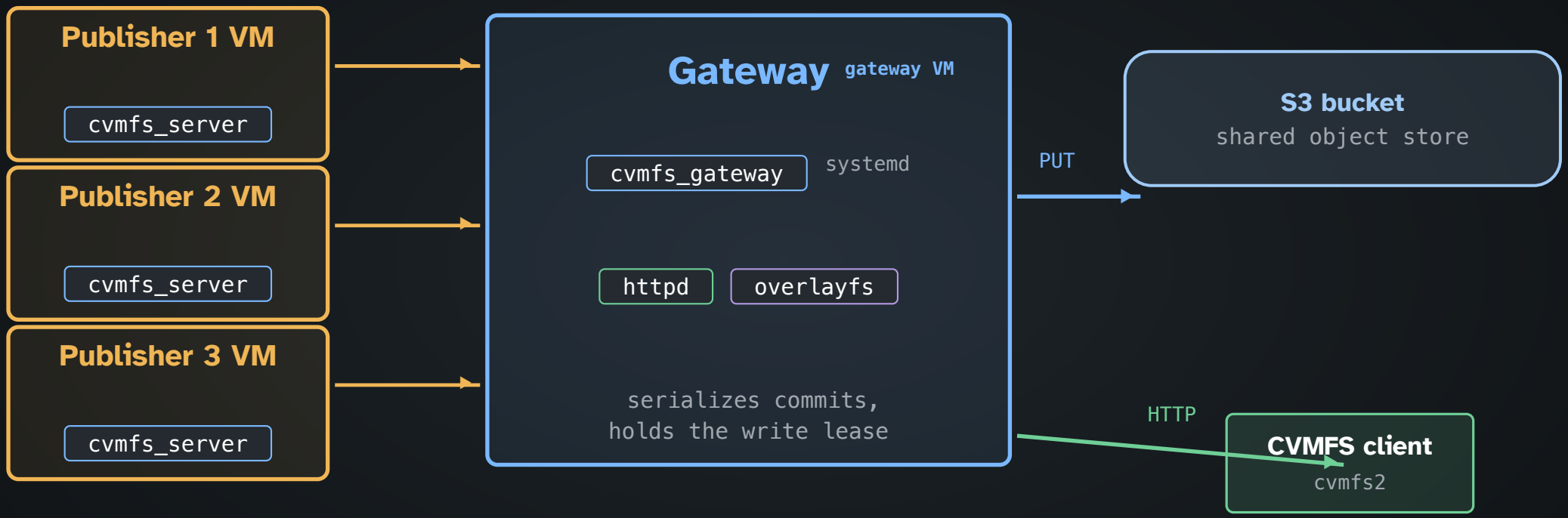
Publisher, web server and storage all on one host. Simple, but scale = that one machine.

Deployment — external S3



Storage is now external — scales independently, and clients bypass the publisher.

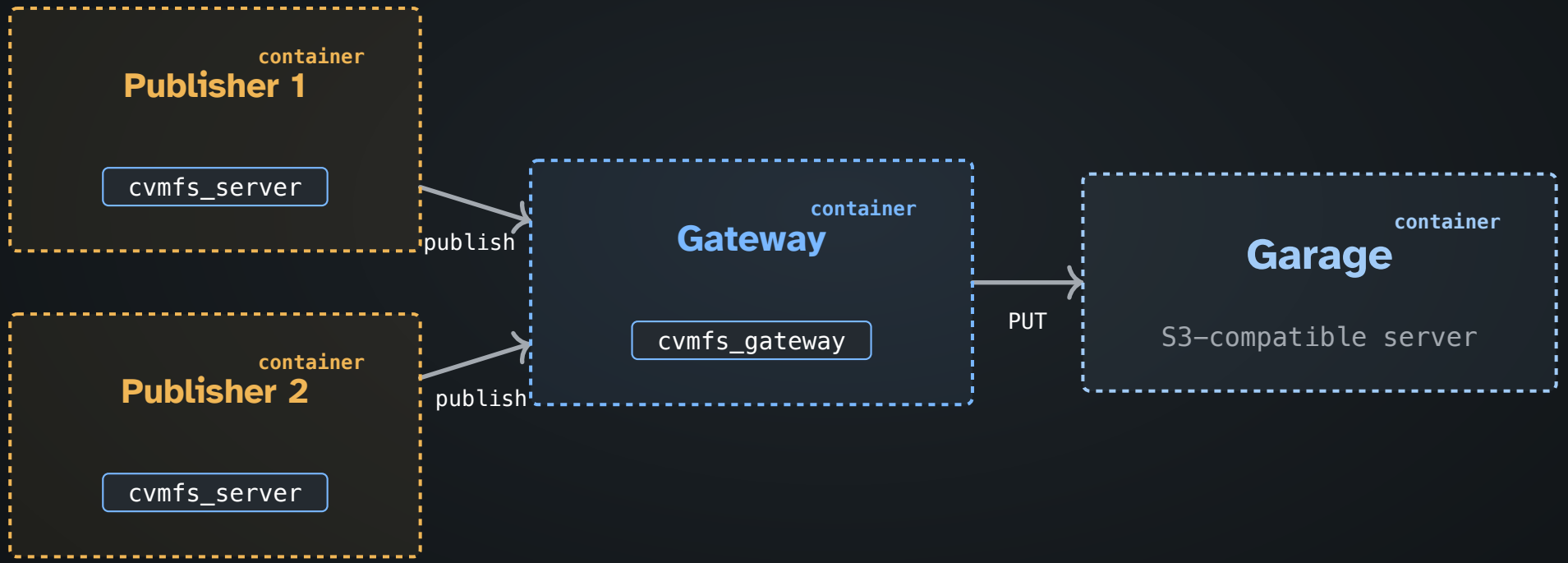
Deployment — gateway + many publishers



Many publishers commit through a single gateway — the gateway holds the write lease and fans out.

Deployment — self-hosted storage with Garage

No external S3 — **Garage** (lightweight S3-compatible server) runs as another container.



Everything — publishers, gateway, storage — runs as plain containers. No cloud S3 required.

docker-compose — before & after

Before — privileged + systemd init

```yaml
services:
 cvmfs-dev:
  image: cvmfs-dev:latest
  privileged: true
  cgroup: host # cgroups v2
  volumes:
   - /sys/fs/cgroup:/sys/fs/cgroup
   - var_spool_cvmfs:/var/spool/cvmfs
   - ../../../:/home/sftnight/cvmfs
```

- needs the host kernel cgroups mounted in
- container runs `sbin/init` — full systemd inside
- **privileged** → access to everything on the host
- awkward under Kubernetes / rootless engines

After — plain, unprivileged

```yaml
services:
 publisher:
  image: cvmfs-publisher:latest
  command: cvmfs_server publish
 httpd:
  image: nginx:alpine
 garage:
  image: dxflrs/garage:latest

# no privileged, no cgroup: host, no init
```

- one concern per container, no `sbin/init`
- no host mounts, no capabilities escalation
- works unchanged on Kubernetes
- reproducible dev + prod from the same compose

Source: `test/common/container/docker-compose.yml` — stripped of the systemd scaffolding.

Server changes for the containerized setup

No systemd, no fuse, no overlayfs — and a friendlier CLI.

Mountless gateway `mkfs -P`

Skips the fuse mount — no overlayfs, no privileged container.

```
$ cvmfs_server mkfs -P
   -o root test.repo.org
```

Mountless ingest `cvmfs_server ingest`

Publish a tarball straight into the repo — no transaction, no mount.

```
$ cvmfs_server ingest
   -t data.tar.gz -b / test.repo.org
```

Landed in 2.14 — PRs #4182 / #4202

One-shot publisher `connect-gw`

Connect to a running gateway — keys fetched from its `/keys` endpoint.

```
$ cvmfs_server connect-gw
   -u gateway.cern.ch test.cern.ch
```

Replaces an 80-char `mkfs -w … -u … -k …` invocation.

Easy-mount dev repos `mkfs -D`

Publishes the public key + a mount helper via the repo's HTTP endpoint.

```
$ cvmfs_server mkfs -D
```

Status of the containerized deployment

✓ **Running in production**

as a `docker-compose` stack

! **Needs major code renovation**

the `cvmfs_server` bash layer is showing its age

? **Helm charts next**

hopefully a small step — but still prototype

Partial replication (prototype in 2.14)

Different sites often only need to serve **one architecture** locally, but still want to keep a persistant copy

Example software tree

```
software.example.io/
├─ software/linux/x86_64 ← keep
├─ software/linux/aarch64 ← exclude
├─ software/linux/ppc64le ← exclude
├─ software/linux/riscv64 ← exclude
└─ …
```

Exclusion spec format

Example .cvmfs_partial_replication

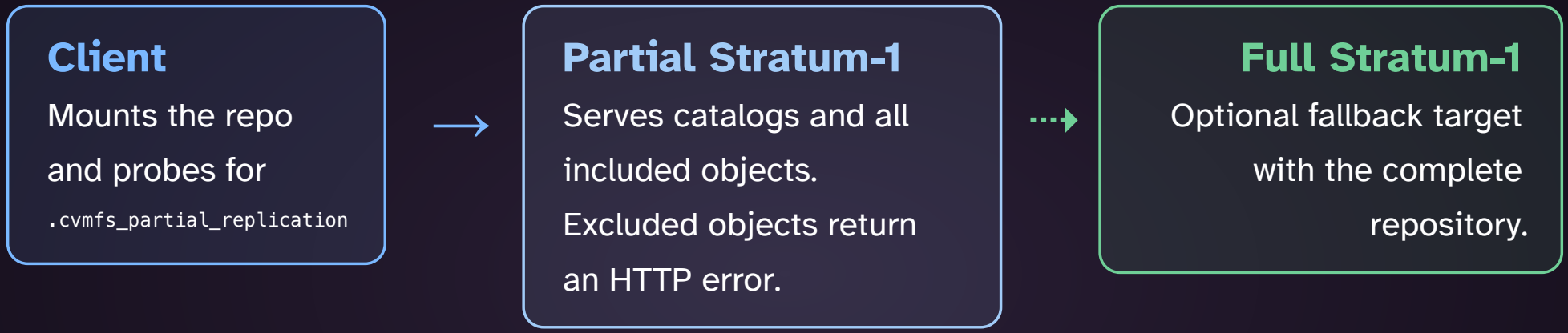
```
version 1
# exclude arch not served locally
/software/linux/aarch64
/software/linux/ppc64le
/software/linux/riscv64
# inclusion inside an excluded tree
!/software/linux/aarch64/generic
```

The server snapshot command passes the spec as **-E** and publishes it as **http://<stratum-one-url>/cvmfs/software.example.repo/.cvmfs_partial_replication** so clients can discover the policy.

Format rules

- First non-comment line must be **version N**
- A normal path means: **exclude this subtree from object replication**
- A path prefixed with **!** means: **re-include** a subtree inside an excluded parent

Client behavior for missing objects



Option 1: failover (default)

Set `CVMFS_PARTIAL_REPLICA_MODE=failover` and provide `CVMFS_FULL_STRATUM1_URL`. If the partial replica returns a missing-object HTTP error, the client retries the fetch from the full Stratum-1.

Option 2: hard fail

Set `CVMFS_PARTIAL_REPLICA_MODE=fail`. There is **no fallback** reads of excluded content end in a hard error (`EIO`). The client can also present excluded entries as unreadable to make the policy obvious.

File bundles — Prototype in 2.14

Today — one GET per file

python

```
import
tensorflow
```

⇓ open() → wait → open() → wait …

GET	tensorflow/__init__.py	→ 1 RTT
GET	tensorflow/keras/__init__.py	→ 1 RTT
GET	tensorflow/python/eager/context.py	→ 1 RTT
GET	numpy/core/_multiarray_umath.so	→ 1 RTT
GET	numpy/core/umath.py	→ 1 RTT

… × hundreds more

import stalls on each round-trip

With `.cvmfsbundle` — async prefetch

python

```
import
tensorflow
```

⇓ open(trigger) → bundle manifest

.cvmfsbundle.trigger

async	tensorflow/__init__.py
async	tensorflow/keras/__init__.py
async	tensorflow/python/eager/context.py
async	numpy/core/_multiarray_umath.so
async	numpy/core/umath.py

fetched in parallel, in the background

Monitoring

There’s now a centrally maintained Prometheus exporter for CVMFS clients.

github.com/cvmfs-contrib/prometheus-cvmfs

Plan to add a Grafana Dashboard template this Summer (GSoC Project together with Brookhaven)

Also — a catalog visualization tool:
github.com/cvmfs-contrib/cvmfs-catalog-visualizations

```
$ curl localhost:9868/metrics
# HELP cvmfs_cached_bytes CVMFS currently
cached bytes.
# TYPE cvmfs_cached_bytes gauge
cvmfs_cached_bytes{repo="cvmfs-
config.cern.ch"} 13823595531
# HELP cvmfs_pinned_bytes CVMFS currently
pinned bytes.
# TYPE cvmfs_pinned_bytes gauge
cvmfs_pinned_bytes{repo="cvmfs-
config.cern.ch"} 281152512
…
```

CVMFS Catalog Visualizer - software.eessi.io

All repositories

Selected: /

Size Legend
- < 10 MB
- 10 - 25 MB
- 25 - 100 MB
- > 100 MB

Selected Catalog

Catalog Size	18 KB
Cumulative Cost	18 KB
Depth	0
Hash	b65506b6d404a6b3…

View directory detail

Largest Catalogs

- 10.57 MB: /versions/2025.06/softwar…
- 10.56 MB: /versions/2025.06/softwar…
- 10.36 MB: /versions/2025.06/softwar…
- 10.24 MB: /versions/2025.06/softwar…
- 10.2 MB: /versions/2025.06/software…
- 10.19 MB: /versions/2025.06/softwar…
- 10.16 MB: /versions/2025.06/softwar…
- 10.14 MB: /versions/2025.06/softwar…
- 10.14 MB: /versions/2025.06/softwar…
- 10.13 MB: /versions/2025.06/softwar…

Instructions:
- Click on a segment to zoom in
- Click center to return to top level
- Hover for details

Containers

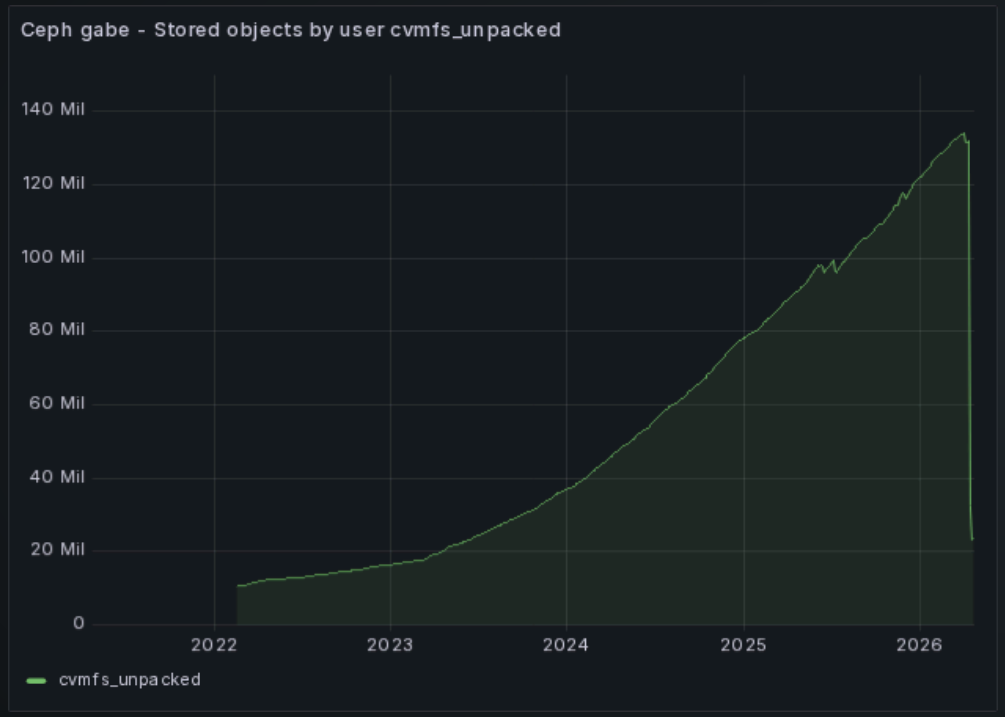
CVMFS provides tooling to unpack, store and distribute containers, with unpacked.cern.ch being the biggest repository:

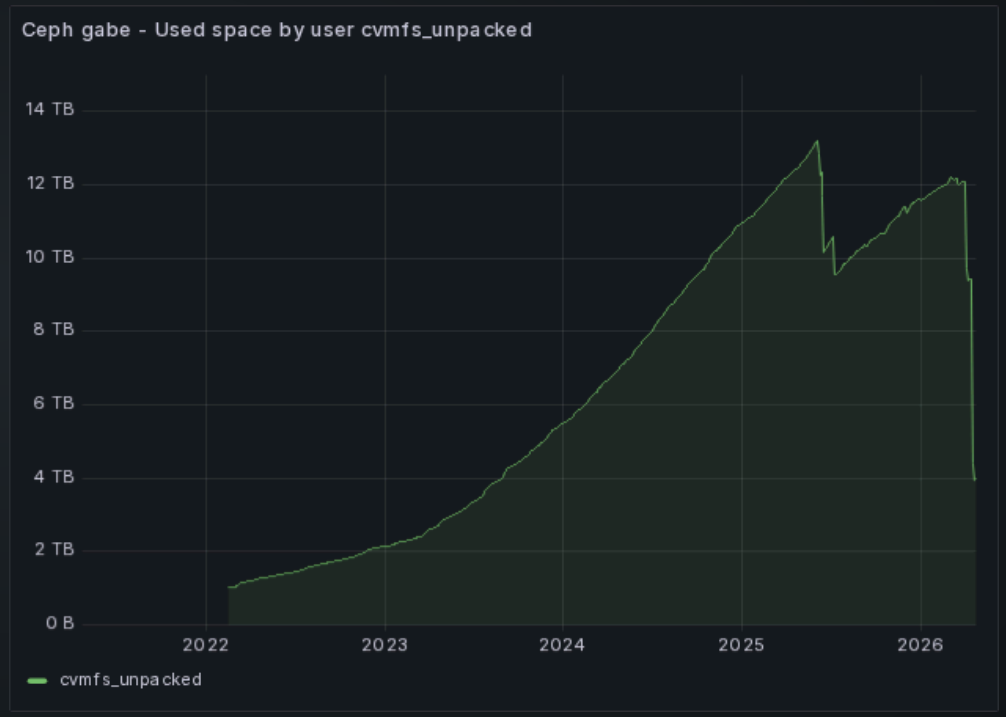
```
user@host:~$ ls /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cmssw/cs8:x86_64-d20211124

afs   build  dev          etc   lib64       mnt   proc  sbin         sys  var
bin   cvmfs  environment  home  lost+found  opt   root  singularity  tmp
boot  data   eos          lib   media       pool  run   srv          usr
```

- Apptainer can directly launch the container from this root file system.
- The same benefits from using CVMFS apply! Leading to:
  - Drastically faster container **startup** times
  - Automatic **cache management** of container images on the worker nodes

Cleanup of unpacked.cern.ch





Stored objects — 131 - 23 M

Used space — 12 -> 4 TB

- Fundamental redesign of unpacker ✓
- Quotas and expiration dates
- Push- instead of pull model

- Partial Replication
- Packing small files
- Asynchronous GC

CernVM-CR

Standalone Container Registry based on CernVM-FS

Further plans (post 2.14)

- Build system and dependencies
- Non-blocking, interruptible garbage collection
- High availability gateway
- Container Distribution

Get in touch

Source code
github.com/cvmfs/cvmfs

Website & documentation
cernvm.cern.ch

Discussion forum
cernvm-forum.cern.ch

CERN Mattermost
mattermost.web.cern.ch/cvmfs

Thanks! Questions?

Valentin Volkl — valentin.volkl@cern.ch

Stay tuned

CernVM Users Workshop

probably February 2027