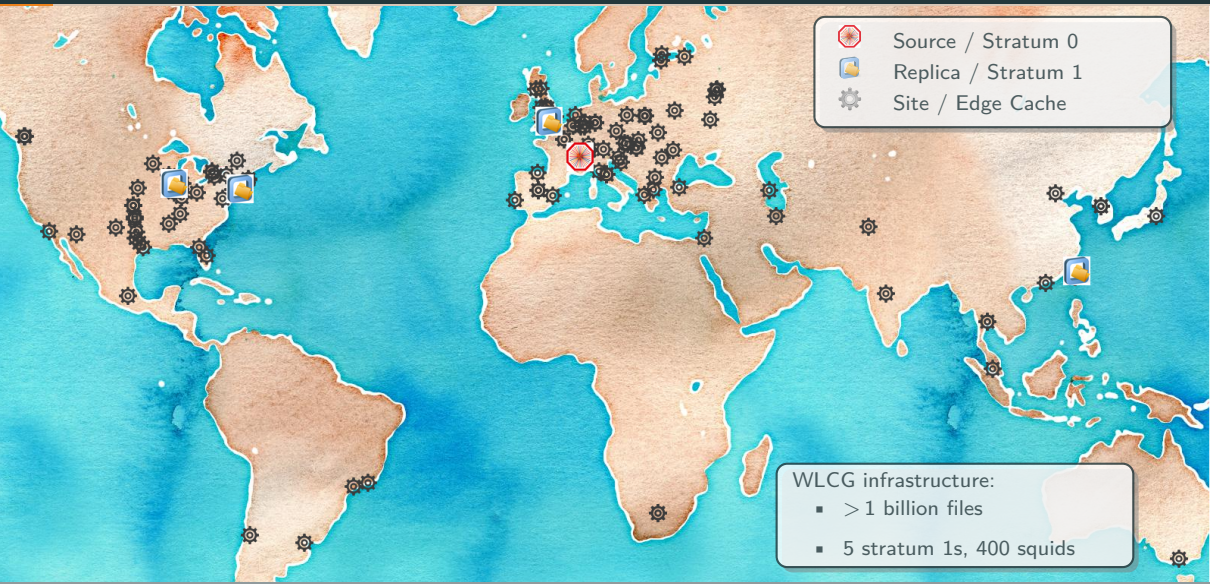
CernVM File System – Status and Plans

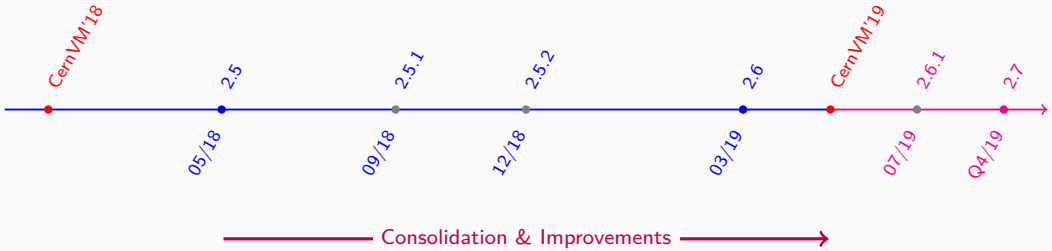
Jakob Blomer

CernVM Workshop 2019

Scale of Deployment



Release Plan



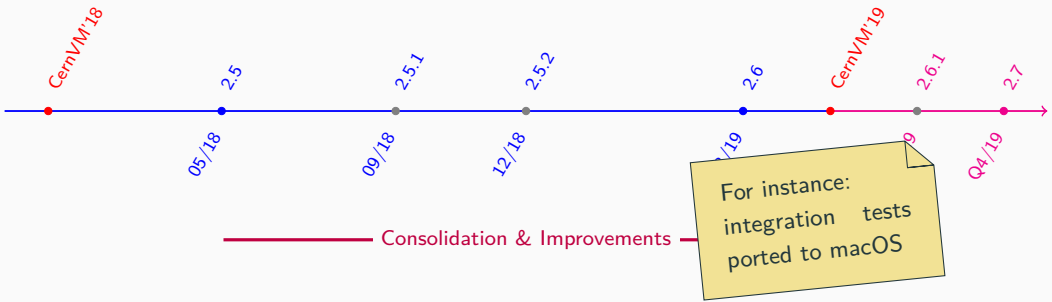
Release 2.5

- Gateway service
- AWSv4 protocol support for S3 backend
- Smart automatic garbage collection
- Automatic handling of DNS server change

Release 2.6

- Shrinkwrap utility for HPC
- Publish metrics
- Direct tarball ingestion
- Container publishing service
- Notification service

Release Plan



Release 2.5

- Gateway service
- AWSv4 protocol support for S3 backend
- Smart automatic garbage collection
- Automatic handling of DNS server change

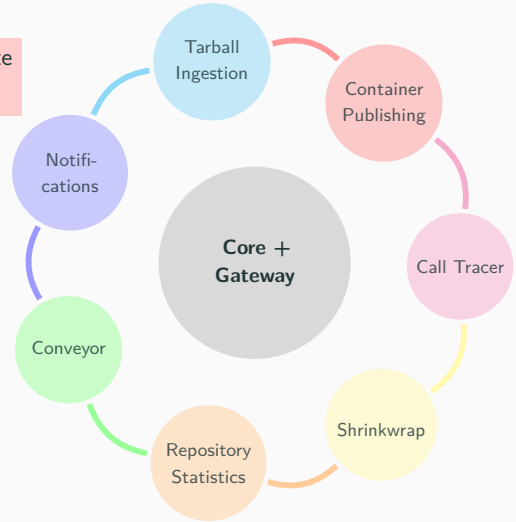
Release 2.6

- Shrinkwrap utility for HPC
- Publish metrics
- Direct tarball ingestion
- Container publishing service
- Notification service

New and Upcoming Satellite Services

New and Upcoming Satellite Services

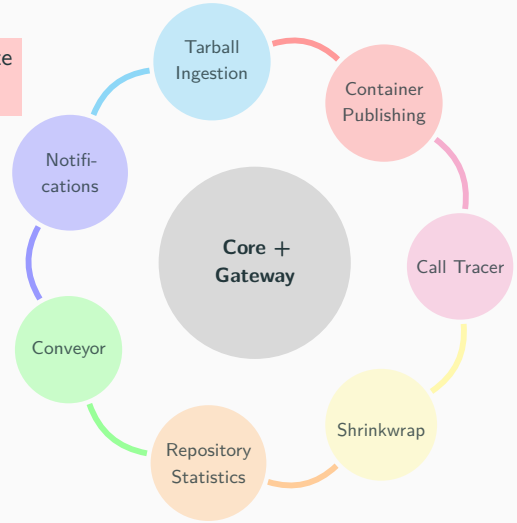
New services will stabilize towards the 2.7 release



New and Upcoming Satellite Services

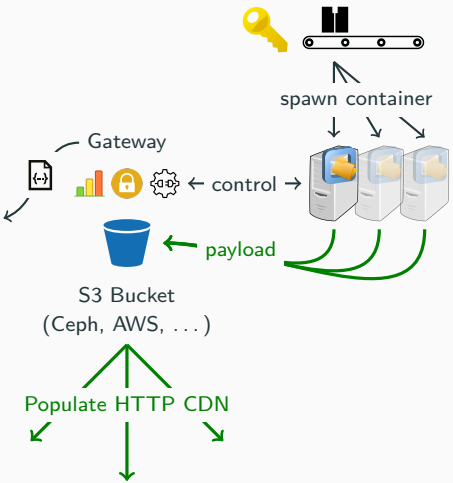
New services will stabilize towards the 2.7 release

Many thanks for your early feedback!



Publishing

Publishing: The Big Picture

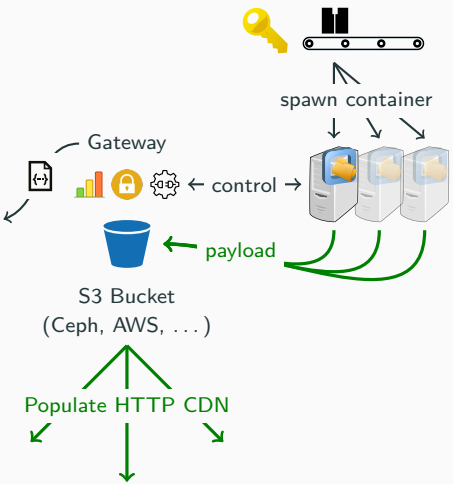


- On demand publish container **or** submit installation job to Conveyor
- Gateway services:
  - Provides **API** for publishing
  - Issues **leases** for sub paths
  - Issues **pre-signed S3 URLs**
  - **Notifies** registered subscribers

Release Status

S3 backend	2.1.20 ff
Gateway service	2.5
Repository statistics	2.6
Notification service	2.6
Conveyor service	under development
Publish container	under development
Presigned S3 URLs	under development

Publishing: The Big Picture



- On demand publish container **or** submit installation job to Conveyor
- Gateway services:
  - Provides **API** for publishing
  - Issues **leases** for sub paths
  - Issues **pre-signed S3 URLs**
  - **Notifies** registered subscribers

Release Status

S3 backend	2.1.20 ff
Gateway service	2.5

Excellent weak scaling up to 4 publishers when publishing ATLAS releases ▸ Report

Reference Deployments

Idea: showcase latest developments,
learn for the adoption of new features into production environment

- **sw.hsf.org, sw-nightlies.hsf.org**
  Send publish jobs from Jenkins to CernVM-FS (CernVM-FS Queue Service), *use multiple concurrent release manager machines* (Gateway Service) *to publish into S3 storage* (S3 backend) *and monitor the change sets over time* (Publisher Metrics)

- **unpacked.cern.ch** (in collaboration with WLCG container working group)
  Publish unpacked container images (Tarball Ingestion) *automatically based on a declaration of interest of images* (Docker Registry Connector) *and use them with Singularity, Docker, or containerd* (Docker graph driver)

- Conditions data repository (depending on experiment interest):
  Start validation jobs immediately after publishing conditions data (Notification Service)

Repository Statistics

Better tooling for maintainers of heavy-duty repositories, requested by LHCb

- New feature: every transaction logs key metrics, e. g. # files, upload volume, etc.
- Stored in SQlite database, accessible to ROOT

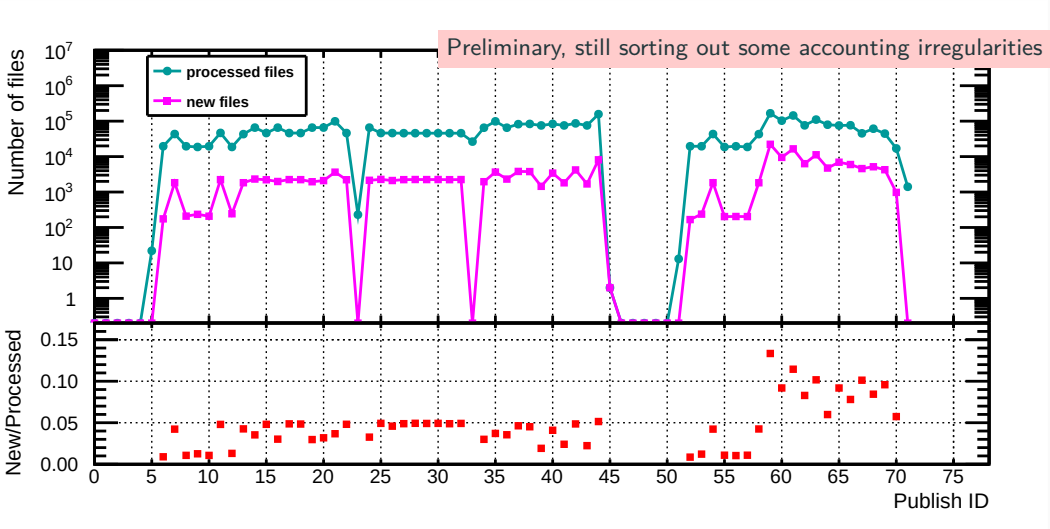
```
auto rdf = ROOT::RDF::MakeSqliteDataFrame(
   "/var/spool/cvmfs/sft-nightlies.cern.ch/stats.db",
   "SELECT * FROM publish_statistics;");

// ...
```

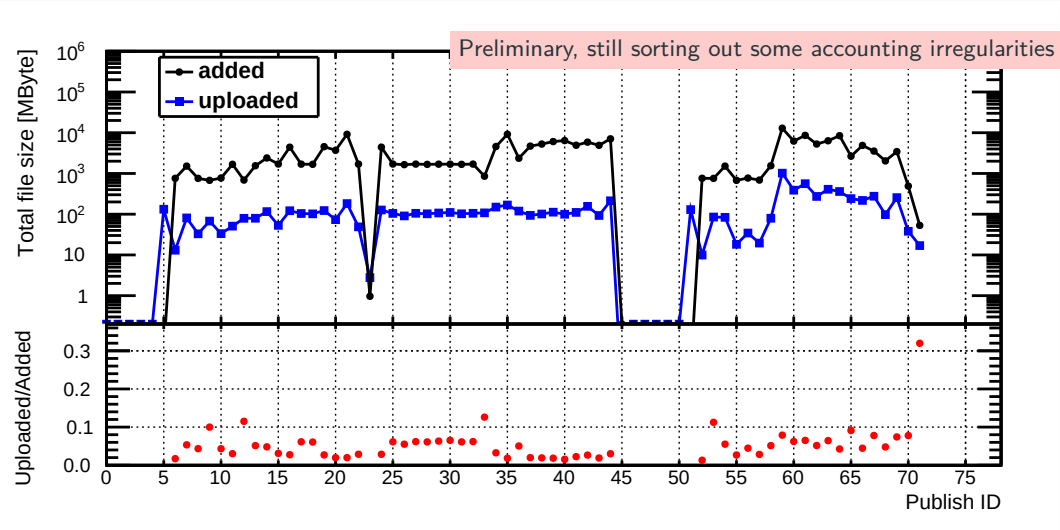
→ **Enables repository insights and quality monitoring**

Repository Statistics: File De-Duplication

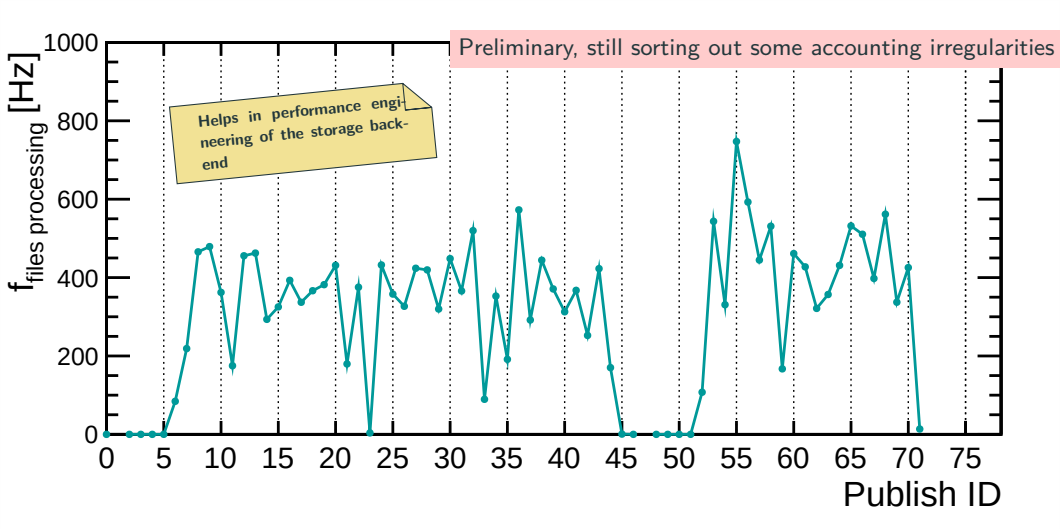
sft-nightlies.cern.ch, 2019-04-10 – 2019-04-12



Repository Statistics: Data Compression and De-Duplication

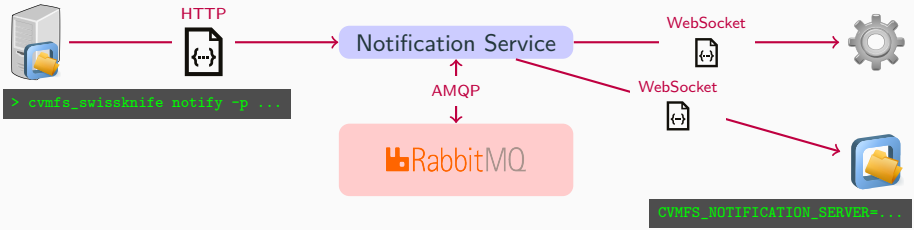


Repository Statistics: Publish Performance



Notification Service

Fast distribution channel for repository manifest: useful for CI pipelines, data QA



- Optional service supporting a regular repository
- Publish/subscribe utility in `cvmfs_swissknife`
- Subscribe component integrated with the client, automatic reload on changes

→ **CernVM-FS writing remains asynchronous but with fast response time in $\mathcal{O}$(seconds)**

Conveyor

A high-level abstraction of writing based on interdependent publication jobs.

```
$ ssh cvmfs-sft.cern.ch
$ cvmfs_server transaction sft.cern.ch /lcg/ROOT
$ tar -xf ROOT-6.18.tar.gz
$ post-install.sh
$ cvmfs_server publish
```

Current approach

```
{
  "repository": "sft.cern.ch",
  "path": "/lcg/ROOT",
  "payload": "https://root.cern.ch/ROOT-6.18.tar.gz",
  "script": "https://spi.cern.ch/post-install.sh",
  "uuid": "e7b67a2...",
  "dependencies": ["f61d...", "a00e...", "..."]
}
```

- Send jobs to Conveyor API
- Conveyor distributes work to multiple publisher nodes

Goal: liberate CI pipeline from handling `cvmfs_server` intrinsics.
Draws heavily from LHCb nightly build publishing system.

Container Integration

CernVM-FS and Containers

❶ CernVM-FS in containers

- Bind mount:

```
docker run -v /cvmfs:/cvmfs:shared ...
singularity exec -B /cvmfs ...
```

- CSI driver ▸ Github repository
  "behind the scenes" bind mount, integrates with kubernetes
- Mount inside privileged container,
  no sharing of the cache

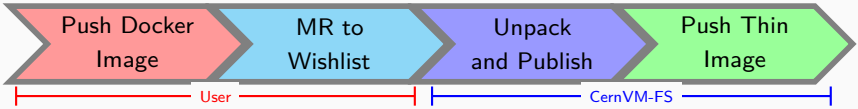
Some issues around autofs peculiarities

❷ Container images in CernVM-FS

- **Unpacked** images on /cvmfs in order to benefit from de-duplication and on-demand caching
- Out of the box support for container engines that support unpacked root file systems (*Singularity*)
- Storage plug-in required for layer based engines
  - Docker graph driver plugin available
  - In touch with containerd developers to upstream plug-in functionality
- Publisher side: requires convenient way to publish images

Container Publishing Service: Workflow

→ Simone's presentation



Wishlist https://gitlab.cern.ch/unpacked/sync

```
version: 1
user: cvmfsunpacker
cvmfs_repo: 'unpacked.cern.ch'
output_format: >
  https://gitlab-registry.cern.ch/unpacked/sync/$(image)
input:
   - 'https://registry.hub.docker.com/library/fedora:latest'
   - 'https://registry.hub.docker.com/library/debian:stable'
   - 'https://registry.hub.docker.com/library/centos:latest'
```

/cvmfs/unpacked.cern.ch

```
# Singularity
/registry.hub.docker.com/fedora:latest -> \
  /cvmfs/unpacked.cern.ch/.flat/d0/d0932...
# Docker with thin image
/.layers/f0/1af7...
```

Currently ∼25 test images available for ATLAS and CMS

Compared to experiment repositories: expected increase of scale by an order of magnitude

- Expect 1 final image per analysis → 1000 – 10 000 images / year
- 250 M to 2.5 B files per year, 5 TB to 50 TB / year [250 k files and 5 GB per image]
- Garbage collection required for image development phase

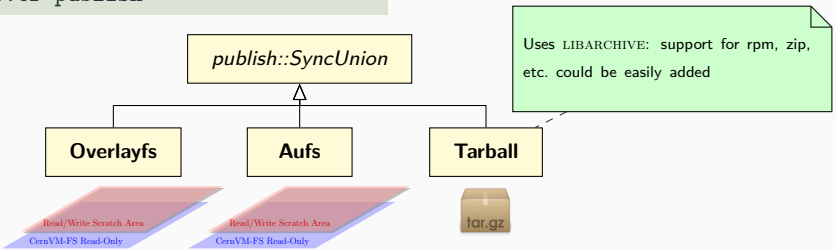
Enabling Feature for Container Publishing: Tarball Ingestion

Direct path for the common pattern of publishing tarball contents

→ Simone's presentation

```
$ cvmfs_server transaction
$ tar -xf ubuntu.tar.gz
$ cvmfs_server publish
```

```
$ zcat ubuntu.tar.gz | \
    cvmfs_server ingest -t -
```



Performance Example

Ubuntu 18.04 container – 4 GB in 250 k files: **56 s untar + 1 min publish** vs. **74s ingest**

HPC Support

HPC Menu & Options

Core issues: network connectivity, missing local harddisk cache, missing fuse

Requirement	**Standard/Grid**		⟶	⟶		**Cumbersome**
`/cvmfs` on WNs	pre-installed	on-demand mount[1]	parrot sandbox[2]	shrinkwrap'd container[3]	NFS/DVS[7]	rsync to cluster fs[4]
WN local cache	local disk	loop-back mounted file	tiered cache	shrinkwrap'd container[3]		
Internet access	site squids	private stratum 1[5]	Pre-loaded cluster cache[6]	shrinkwrap'd container[3]		

[1] e. g. SLURM plug-in, in the future Singularity plugin
[2] needs testing with application workflow
[3] maintenance burden of 100GB+ images
[4] scalability issues, maintenance burden
[5] rather used for online farms
[6] often requires tiered setup for scalability
[7] scalability issues, experience at NERSC

HPC Support: Shrinkwrap

→ Tim's presentation

Official UNCVMFS: export bulky `/cvmfs` subtrees into "fat containers".
Requested by ATLAS and CMS for US HPCs, also used by IT/HEPiX benchmark working group.

```
cvmfs_shrinkwrap -r sft.cern.ch \
  -t sft.cern.ch.spec \
  -z /export/cvmfs ...
```

sft.cern.ch.spec

```
/lcg/releases/ROOT/6.16.00-fcdd1/*
/lcg/releases/gcc/*
...
```

```
/export/cvmfs/.provenance/...
/export/cvmfs/.data/...
/export/cvmfs/sft.cern.ch/...
```

Compared to `rsync`:

- Faster: 50 MB/s vs. 30 MB/s
- Data de-duplication through hardlinks
- Efficient synchronization and GC
- Aware of CernVM-FS specifics

Shrinkwrapping is a rather heavy-weight process, dedicated "bridge nodes" recommended.

Instrumentation: Call Tracer

→ Tim's presentation

Precise, file-system level trace of /cvmfs accesses

1. Specification input for `cvmfs_shrinkwrap`
2. Instrumentation tool for benchmark analysis

```
$ echo "CVMFS_TRACEFILE=/tmp/trace.@fqrn@.csv" > /etc/cvmfs/default.local
$ mount -t cvmfs repo.cvmfs.io /cvmfs/repo.cvmfs.io
$ # Run testee from /cvmfs/repo.cvmfs.io
$ sudo cvmfs_talk -i repo.cvmfs.io tracebuffer flush
```

CSV

```
"1555099772803.948","-1","Tracer","Trace buffer created"
"1555099776596.462","6","","getattr()"
"1555099776596.700","2","","opendir()"
"1555099776599.053","4","/lcg","lookup()"
"1555099777187.145","2","/lcg","opendir()"
"1555099777351.414","4","/lcg/app","lookup()"
```

Development and Support Plan

Code Works

- Revisit and refurbish publisher tools
  - Drop requirement of exclusively owning a release manager machine, instead: use ephemeral publish container
- Future-proof client:
  investigate implications of root-less FUSE and latest Linux kernel namespaces features
- Consolidation of satellite services in Go → Radu's talk

- Incremental (faster) garbage collection *(Jan's project)*
- cernvm-monitor.cern.ch: transition to client-side monitoring based on JavaScript client *(summer student project)*
- Deep performance analysis using modern Linux tools (eBPF) *(openlab student project)*

Server Refurbishment

Current state

A set of tools targeted for dedicated release manager machines, but also, partially, used by gateway ("receiver"), container publishing, tarball ingestion, ...

Where we want to be

A common base library providing repository transformation primitives, on top of which higher-level publish abstractions can be built

Supported Platforms

- *A* Platforms:
  - EL 6–7 AMD64 (soon: EL8)
  - Ubuntu 16.04, 18.04 AMD64
  - macOS, latest two versions
- *B* Platforms
  - SLES 11 – 12
  - Fedora, latest two versions
  - Debian 8–9
  - Ubuntu 12.04 and 14.04
  - EL7 AArch64
  - IA32 architecture
- Experimental: POWER8, Raspberry Pi, RISC-V
- Blue sky idea: Windows port based on ProjFS ▸ Microsoft CernVM'18 spresentation

Based on the current needs.
Any platform with Fuse support should be straight-forward to address.

Outlook to CernVM-FS 2.7/2.8

Ephemeral Publish Container

proof of concept

Eliminate the need for dedicated publisher nodes

```
$ cvmfs enter hsf.cvmfs.io /users/joe
...Opens a shell in an ephemeral container
   with write access to the repository
$ cvmfs publish
...Back to read-only mode
```

- Requires the gateway service
- Will require major renovation in the `cvmfs_server` tool chain
- **Will enable cvmfs publisher clusters (e. g. “lxcvmfs”)**

Unprivileged Fuse Client

merged

Leap in support of opportunistic resources

- Only privileged operation required: `mount()`
  - Currently handled by fuse suid binary
  - Reason why cvmfs needs to be “installed”
- As of RHEL8 with new kernel and libfuse3:
  - Limitations on `mount()` lifted

  → **Possibility of a “super-pilot” comprising cvmfs and singularity**

Demo of ephemeral publish container

Summary

- CernVM-FS 2.6 released several new satellite services supporting **HPC sites**, **container-based workflows**, and the **publishing process**
- Intend to gain experience with new services through reference deployments
- New functionality will stabilize in patch releases during the upcoming months

- CernVM-FS 2.7/2.8: revisit client and server internals in order to better exploit **opportunistic resources** and to provide **on-demand publishing** workflows