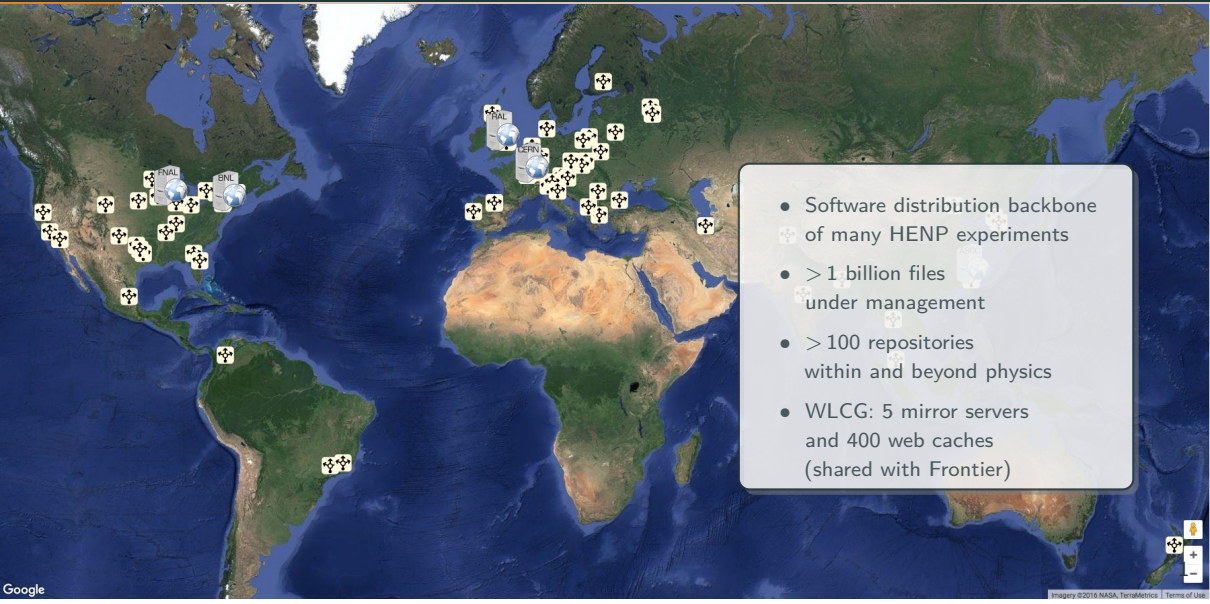
# Towards a Serverless CernVM-FS

Jakob Blomer for the CernVM-FS team
CERN, EP-SFT

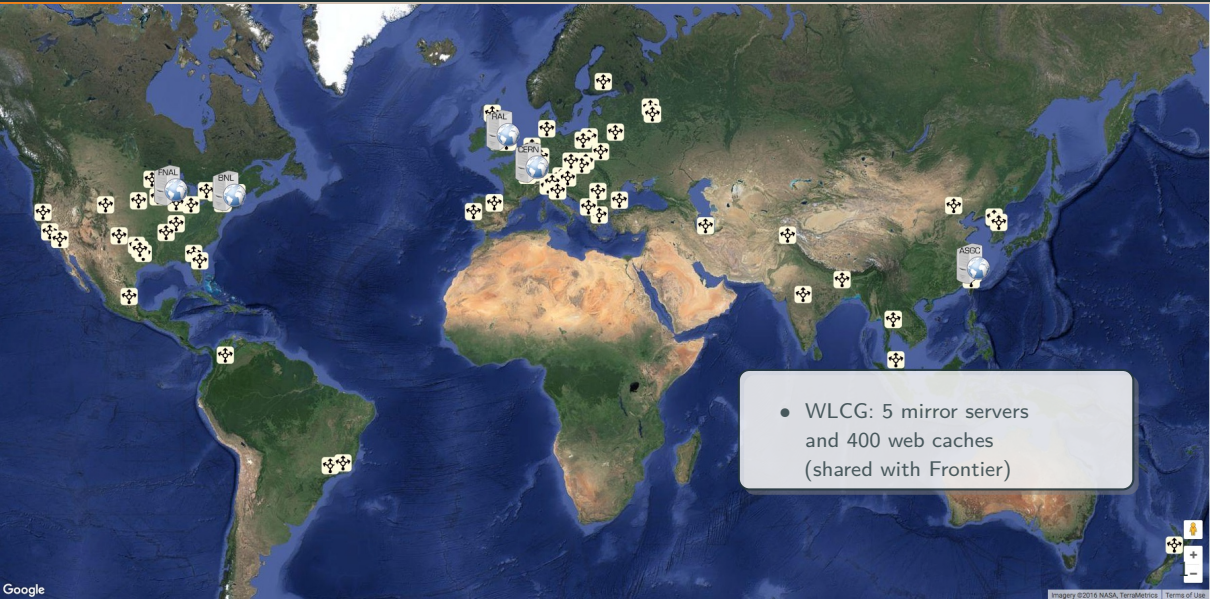
CHEP 2018, Sofia

# CernVM-FS Scale of Deployment




- Software distribution backbone of many HENP experiments
- $>1$ billion files under management
- $>100$ repositories within and beyond physics
- WLCG: 5 mirror servers and 400 web caches (shared with Frontier)

# CernVM-FS Scale of Deployment




- WLCG: 5 mirror servers and 400 web caches (shared with Frontier)

# Project Goals

❶ Provide uniform, consistent, and versioned POSIX file system access to /cvmfs

```
$ ls /cvmfs/cms.cern.ch
slc7_amd64_gcc700  slc7_ppc64le_gcc530  slc7_aarch64_gcc700  slc6_mic_gcc481
...
```

on **grids**, **clouds**, **supercomputers** and **end user laptops**

*read*

*publish*

❷ Populate and propagate new and updated content

- Support an increasing number of /cvmfs/. . . writers

Propagation delay down to 5–10 min

# Project Goals

❶ Provide uniform, consistent, and versioned POSIX file system access to /cvmfs

```
$ ls /cvmfs/cms.cern.ch
slc7_amd64_gcc700  slc7_ppc64le_gcc530  slc7_aarch64_gcc700  slc6_mic_gcc481
...
```

on **grids**, **clouds**, **supercomputers** and **end user laptops**

*read*

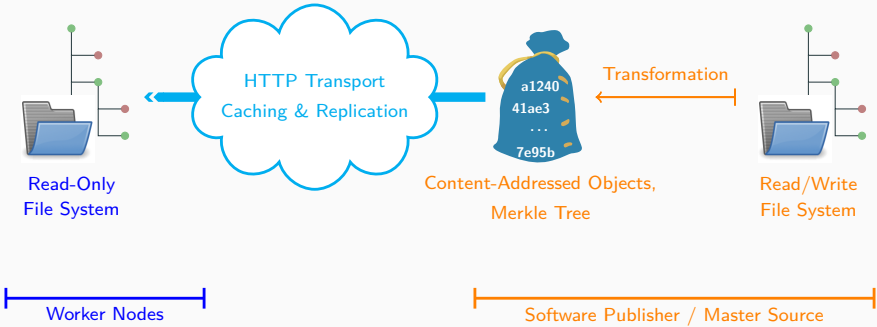
*publish*

❷ Populate and propagate new and updated content

- Support an increasing number of /cvmfs/. . . writers

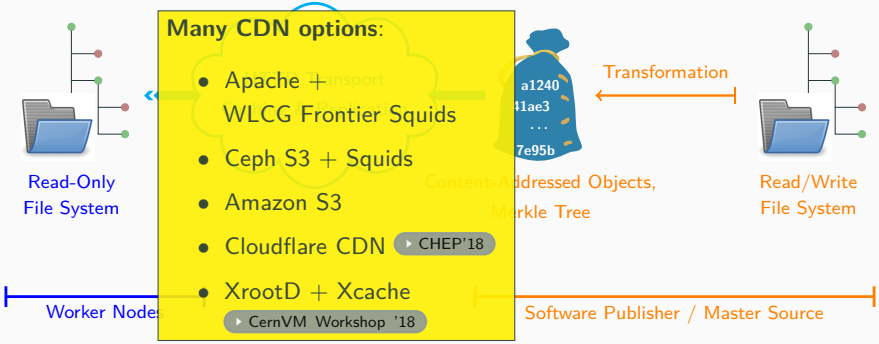
Propagation delay down to 5–10 min

# Key Design Choices




- Reading and writing treated asymmetrically
- Immutable objects, stateless services
- HTTP transport
- Aggressive caching

# Key Design Choices




- Reading and writing treated asymmetrically
- Immutable objects, stateless services
- HTTP transport
- Aggressive caching

# From R&D to Community Software


- R&D character attracted excellent engineers; some went on to Mesosphere, Amazon, Pivotal, etc.
- Contributions by OpenLab, CERN KT fund, CERN IT, FNAL, U Nebraska, U Notre Dame, GSoC, industry
- User workshops at RAL and CERN
- Active user, operations, and code community

| | |
|---|---|
| 2008 | CernVM R&D: 1st prototype. |
| 2010 | CernVM-FS @ CHEP Taipei. |
| 2011 | Code moves to Github. |
| 2012 | Picked up by PIC, RAL; grid deployment task force. |
| 2014 | DPHEP / software preservation. |
| 2016 | Growing outside interest: LIGO, EUCLID, Mesosphere, . . . . |
| 2017 | Deployment on HPC. |

# Strategic Application Areas

## ❶ Production Software

Example: */cvmfs/ligo.egi.eu*

- ✓ Most mature use case
- Continuous effort on HPC systems

## ❷ Integration Builds

Example: */cvmfs/lhcbdev.cern.ch*

- ✓ High churn, requires regular garbage collection
- Instant update propagation to file system clients ▸ CHEP'18

## ❸ Extracted Container Images

Example: */cvmfs/singularity.opensciencegrid.org*

- ✓ Works out of the box with Singularity
- ✓ CernVM-FS driver for Docker ▸ ACAT'17
- Easy ingestion of images ▸ Prototype

## ❹ Auxiliary data

Example: */cvmfs/alice-ocdb.cern.ch*

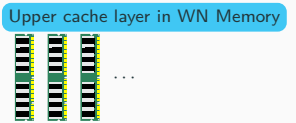
- ✓ Benefits from internal versioning
- Depending on volume requires more planning for the CDN components

Current focus of developments

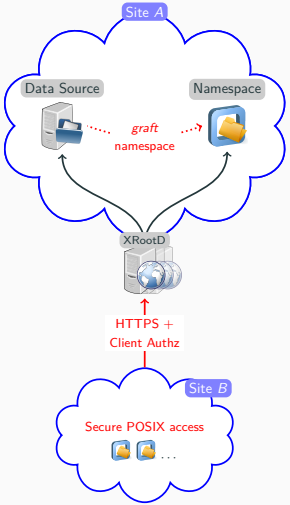
# CernVM-FS à la Carte

## ❶ HPC Client Deployment

- Piz Daint: Europe's fastest supercomputer
- Runs ATLAS, CMS, and LHCb jobs from native Fuse client




## ❷ Data Namespace: */cvmfs/*.osgstorage.org*

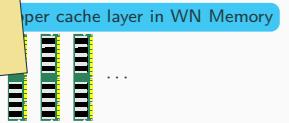


# CernVM-FS à la Carte

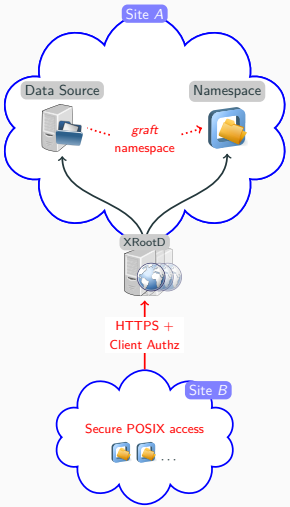
## ❶ HPC Client Deployment

Custom client cache:
hierarchical cache
+ RAM disk plugin

- Runs ATLAS, CMS, and LHCb jobs from native Fuse client




## ❷ Data Namespace: */cvmfs/*.osgstorage.org*

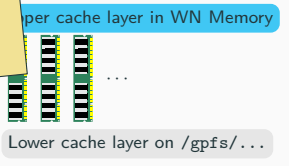


# CernVM-FS à la Carte

## ❶ HPC Client Deployment

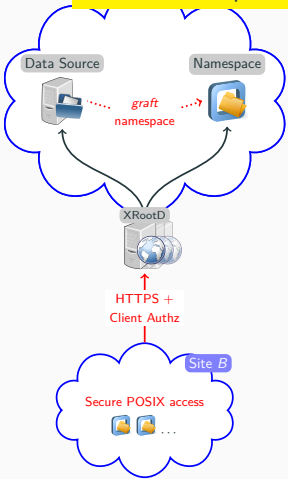
Custom client cache:
hierarchical cache
+ RAM disk plugin

- Runs ATLAS, CMS, and LHCb jobs from native Fuse client




## ❷ Data Namespace: */cvmfs/*.osgstorage.org*

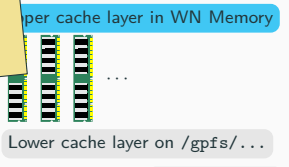
access to multiple PB of data



# CernVM-FS à la Carte

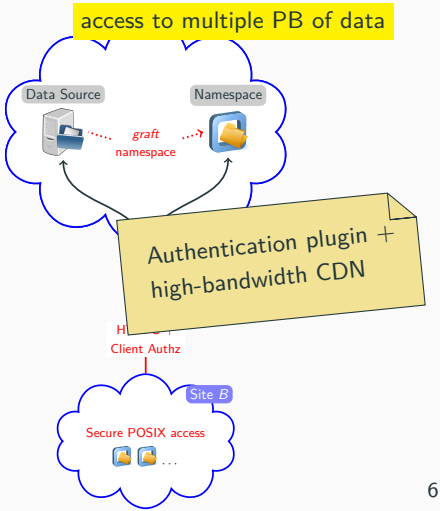
## ❶ HPC Client Deployment

Custom client cache:
hierarchical cache
+ RAM disk plugin




- Runs ATLAS, CMS, and LHCb jobs from native Fuse client

## ❷ Data Namespace: */cvmfs/*.osgstorage.org*

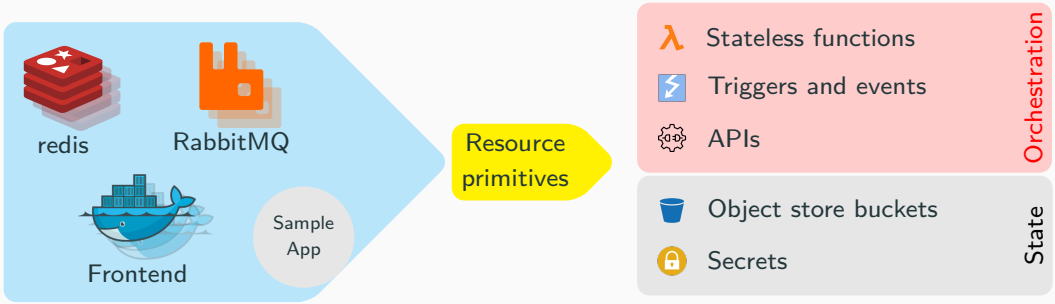


# Serverless Publishing

# “Serverless” – Looking inside the Box

**A reactive model of cloud computing**

Replace a set of always-on components by **triggered functions** and **explicit state**



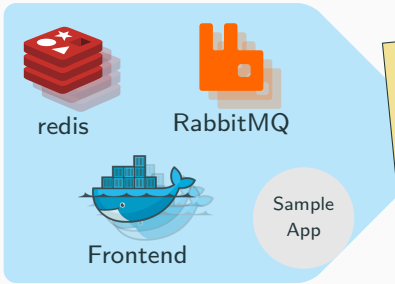

Commercial offers from the big cloud platforms available.

Many open source frameworks emerging, core functionality spawns **ephemeral containers on demand**

# “Serverless” – Looking inside the Box

**A reactive model of cloud computing**

Replace a set of always-on components by **triggered functions** and **explicit state**




⊕ No special nodes to care about
⊕ Seamless scaling

⊖ More moving parts
⊖ Soft infrastructure lock-in

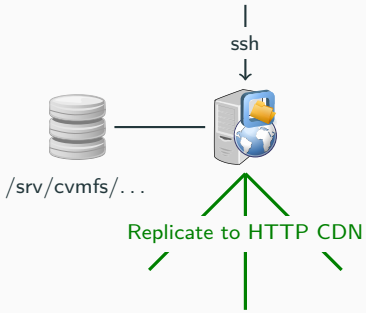
Orchestration
- Stateless functions
- Triggers and events
- APIs

State
- Object store buckets
- Secrets

Commercial offers from the big cloud platforms available.

Many open source frameworks emerging, core functionality spawns **ephemeral containers on demand**

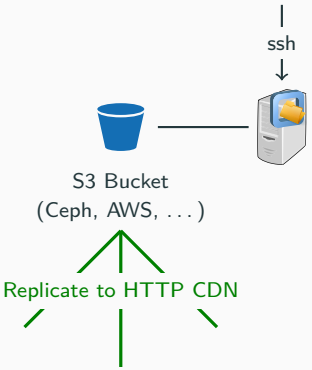
# Evolution of the Publishing Process




**❶ Stand-Alone Release Manager Machine**

- Dedicated web server
- Local storage or attached block volume

# Evolution of the Publishing Process

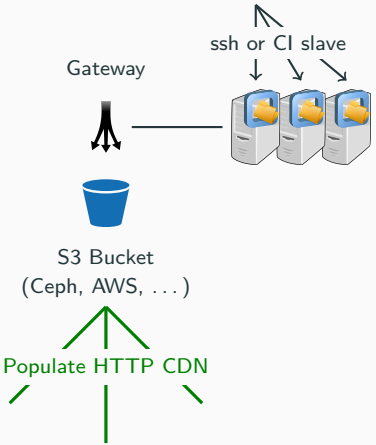



## ❷ S3 Storage

CHEP'15

- Dedicated release manager machine, now stateless
- Cloud storage, also available to replicas

# Evolution of the Publishing Process

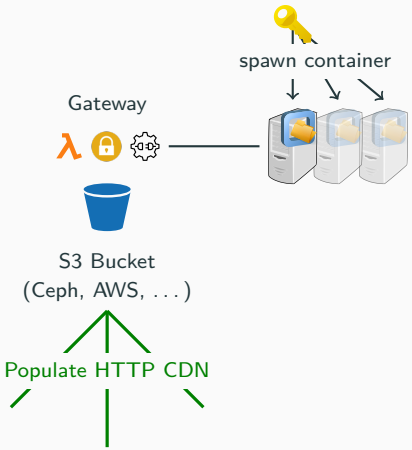



**❸ Multiple Release Manager Machines**

Released 2018 ▸ CHEP'18

- Concurrent release manager access through new gateway services
- Example: /cvmfs/cernvm-prod.cern.ch
- Gateway services powered by Erlang/OTP
  - **API** for publishing
  - Issues **leases** for sub paths
  - Receives change sets as set of **signed object packs**

# Evolution of the Publishing Process




**❹ Future Serverless Architecture**

- On demand release manager container
  - ▸ Prototype Report
- Gateway service primitives

| | |
|---|---|
| **State** | Access keys and active leases |
| **Functions** | Lease management, receiving object packs, committing change sets |

# Vision for Publishing

1. Scale number of writers (users, computers) from tens to hundreds per repository.
2. Fast-path (*"portal"*) to directly push certain payload types.

❶

```
$ cvmfs enter hsf.cvmfs.io /users/joe
...Opens a shell with write access
$ cvmfs publish
...Back to read-only mode
```


❷

```
$ cvmfs push docker://hsf/software \
    hsf.cvmfs.io/containers
$ cvmfs push myanalysis.tar.gz \
    hsf.cvmfs.io/users/joe
```

Development program for $\sim$ 2 years.

# Summary and Outlook

We have a **stable, yet extensible core** which will continue receiving our attention.

**Targeted Out of the Box Support**

Read access to `/cvmfs` on **grids**, **clouds**, **supercomputers**, **end-user devices**
for **production software**, **integration builds**, **container images**, **auxiliary data**

**Future Directions**

Focus on **scaling up publishing workloads** from tens to hundreds of users per repository

```
$ touch /cvmfs
```

https://github.com/cvmfs/cvmfs

Thank you for your time!

# Summary and Outlook

We have a **stable, yet extensible core** which will continue receiving our attention.

**Targeted Out of the Box Support**

Read access to `/cvmfs` on **grids**, **clouds**, **supercomputers**, **end-user devices**
for **production software**, **integration builds**, **container images**, **auxiliary data**

**Future Directions**

Focus on **scaling up publishing workloads** from tens to hundreds of users per repository

```
$ touch /cvmfs
```

https://github.com/cvmfs/cvmfs

Thank you for your time!