Recent Developments in the CernVM-FS Server Backend

René Meusel

Jakob Blomer, Gerardo Ganis,
Predrag Buncic, Seppo Heikkila

ACAT 2014 - Prague, 4th of September

1 Usage Statistics and Adoption

2 New Challenges and Features

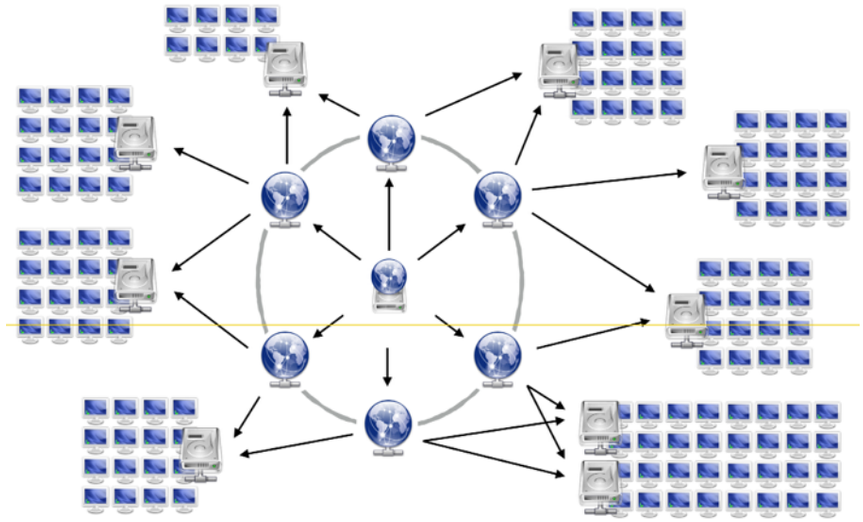
3 File System History

4 Garbage Collection

5 Smart Stratum1 Servers

What is CernVM-FS?

- **Scalable software distribution system**
  - Infrequent atomic updates in a central location
  - Read-only access on the clients
- **HTTP based global data transfer**
  - Minimal protocol requirements
  - Aggressive hierarchical cache strategy
    - Assumption: Coherent working set on physically close nodes (cf. software vs. data distribution)



- **Accessible through a mounted file system (POSIX)**
  - FUSE module, NFS exported FUSE volume or Parrot

Who Uses CernVM-FS?

- **All LHC experiments**
- **CernVM 3**
  - Operating system in CernVM-FS
- **Others beyond the HEP community**
  - Human Brain Project, BioMed, VLEMED, …
- Stratum0s at CERN, RAL, NIKHEF, Fermilab, DESY, …

Repository Statistics

Repository	Files	Stored Objects	Volume	ø File Size
atlas.cern.ch	34'500'000	3'700'000	2.1 TiB	66.2 kiB
cms.cern.ch	30'600'000	4’800'000	0.9 TiB	33.1 kiB
lhcb.cern.ch	13’600'000	4'600'000	0.5 TiB	41.9 kiB
alice.cern.ch	5'900'000	240'000	0.5 TiB	90.7 kiB
ams.cern.ch	2'900'000	1'900'000	1.9 TiB	0.7 MiB
alice-ocdb.cern.ch	700'000	700'000	0.1 TiB	0.2 MiB
atlas-condb.cern.ch	8'000	7'800	0.5 TiB	60.8 MiB

Mainly Software

Software + Conditions Data

Conditions Data

- *Files* and *Volume* as saved in the CernVM-FS catalogs
- Actual number of *Stored Objects* is compressed and de-duplicated
- Based on latest revision - no history involved

(Effective: August 2014)

New Challenges

New Challenges

Large Files (> 200 MiB)

Long term data preservation

Rapidly changing repository content

Increasing configuration distribution effort

Instant repository update propagation

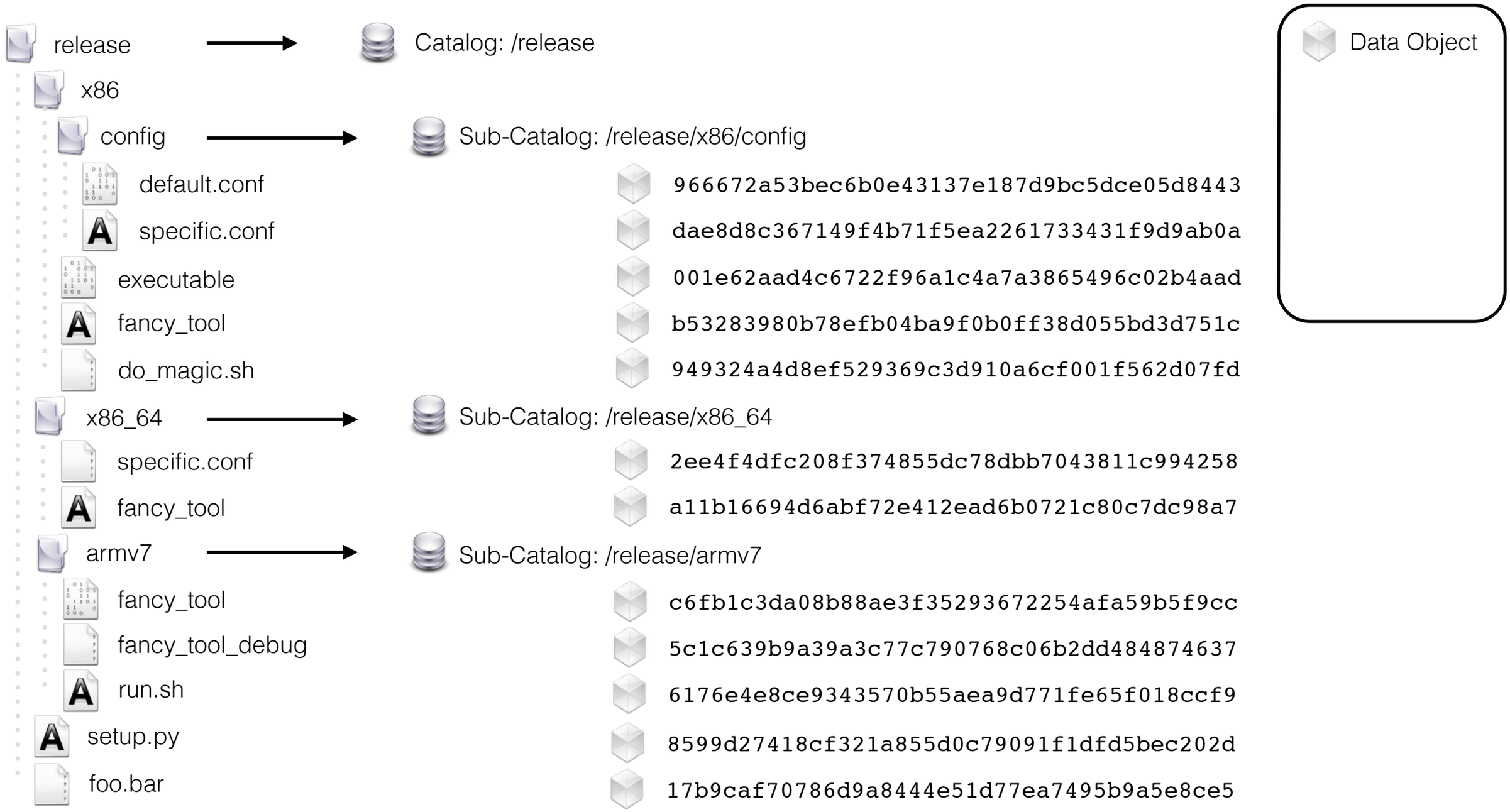
CernVM-FS Repository

From POSIX File Filesystem to
Content-Addressable Objects

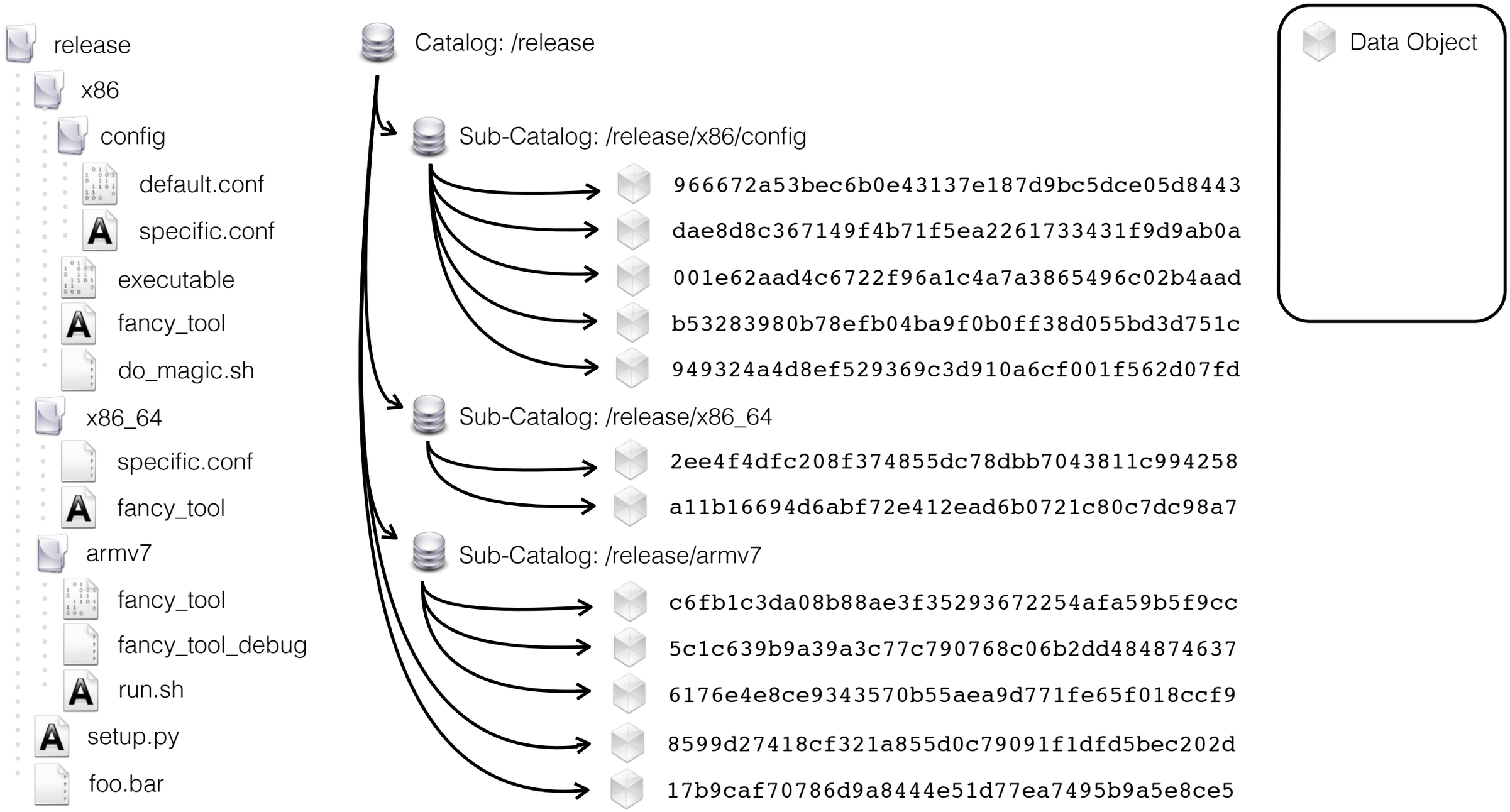
From POSIX to Blob-Objects

From POSIX to Blob-Objects

From POSIX to Blob-Objects

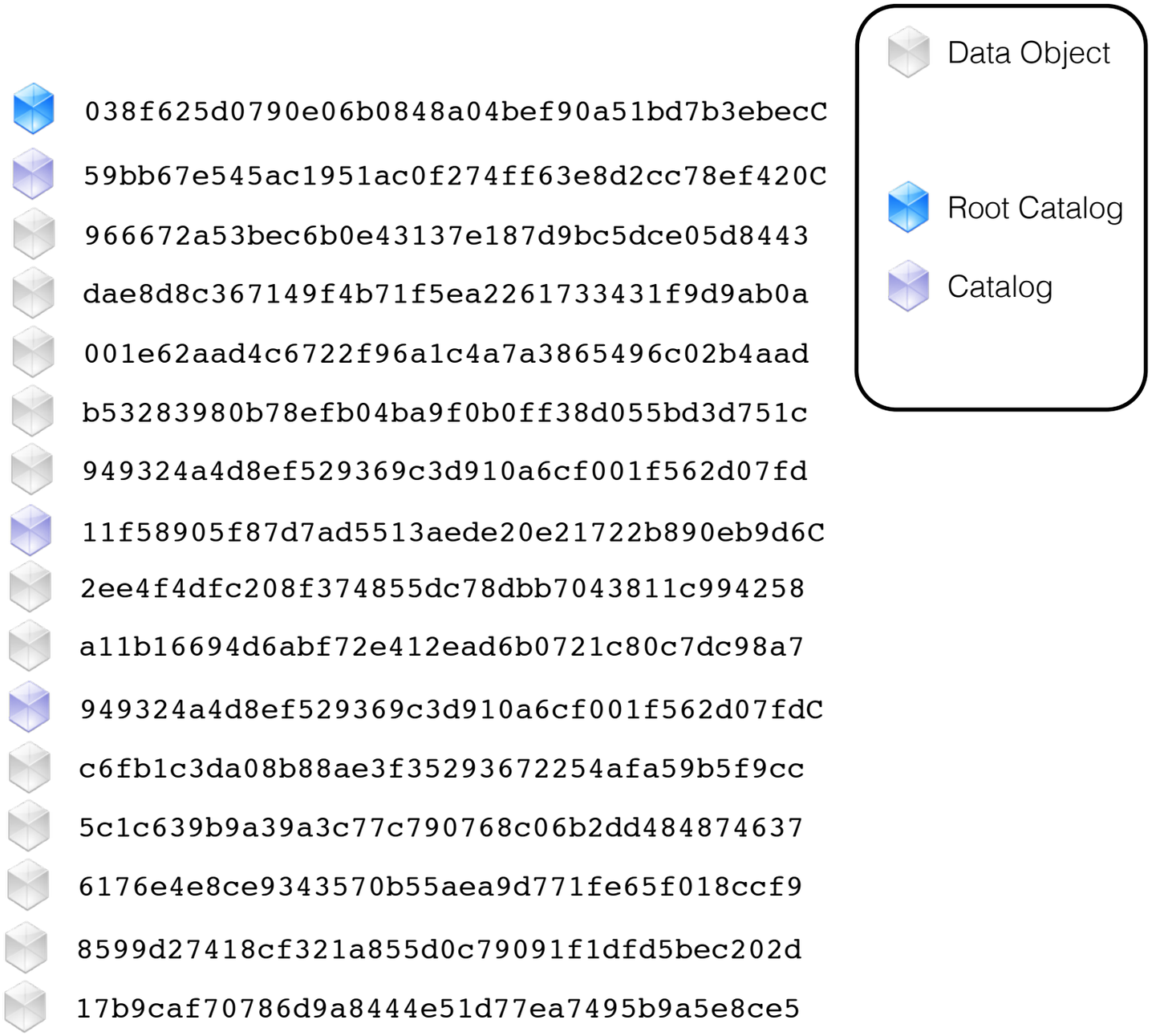


From POSIX to Blob-Objects

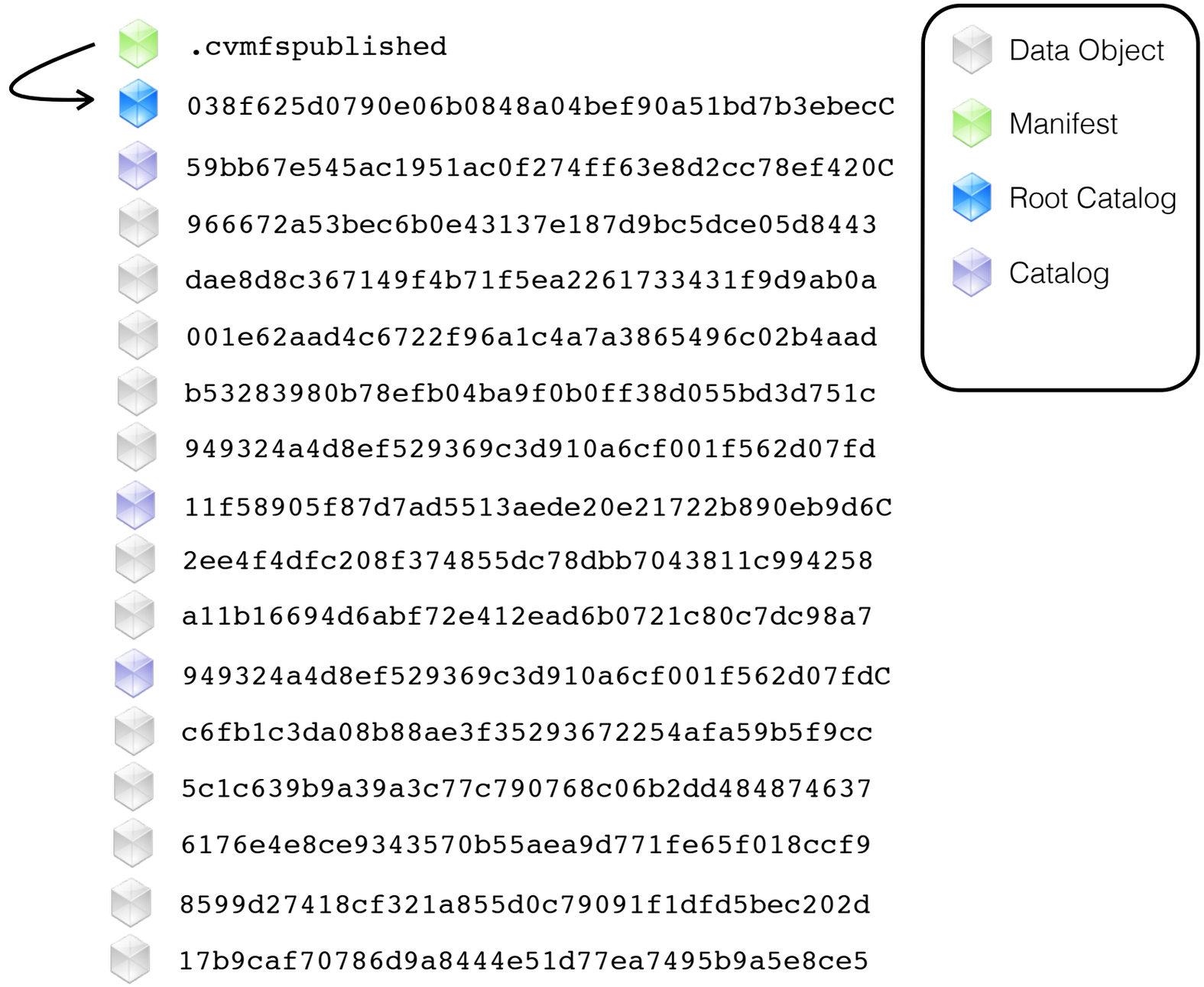


From POSIX to Blob-Objects

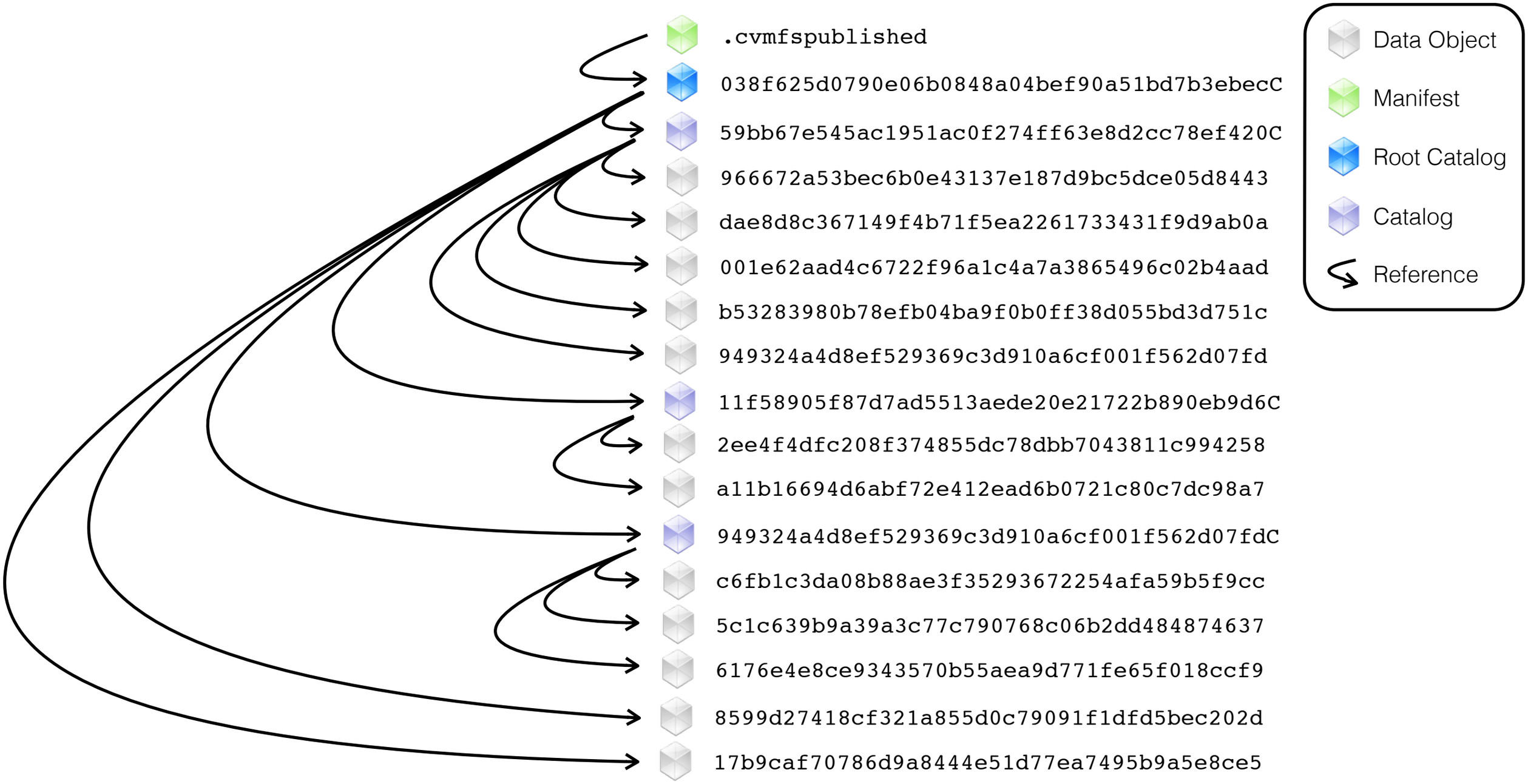
From POSIX to Blob-Objects



From POSIX to Blob-Objects

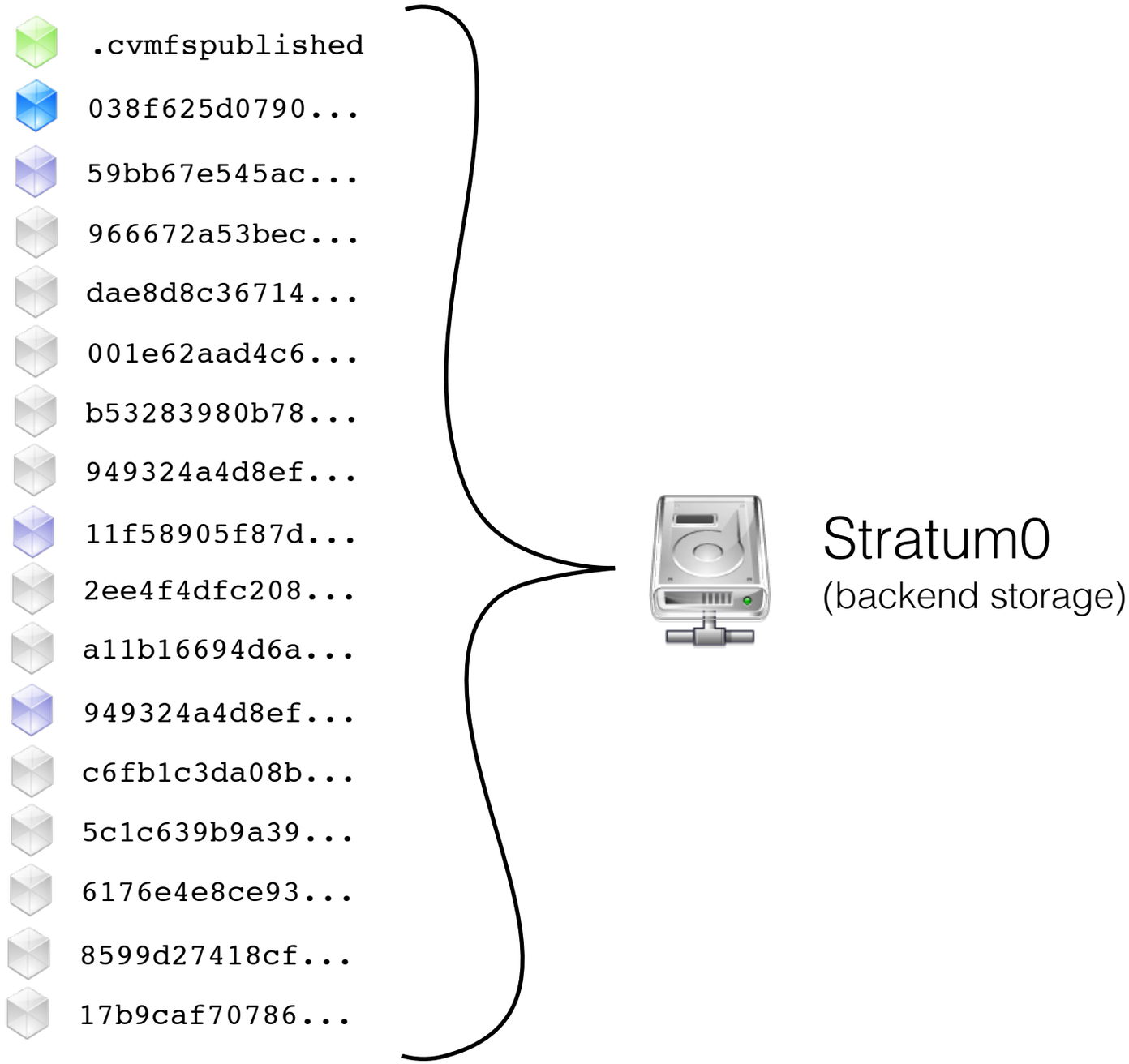


From POSIX to Blob-Objects



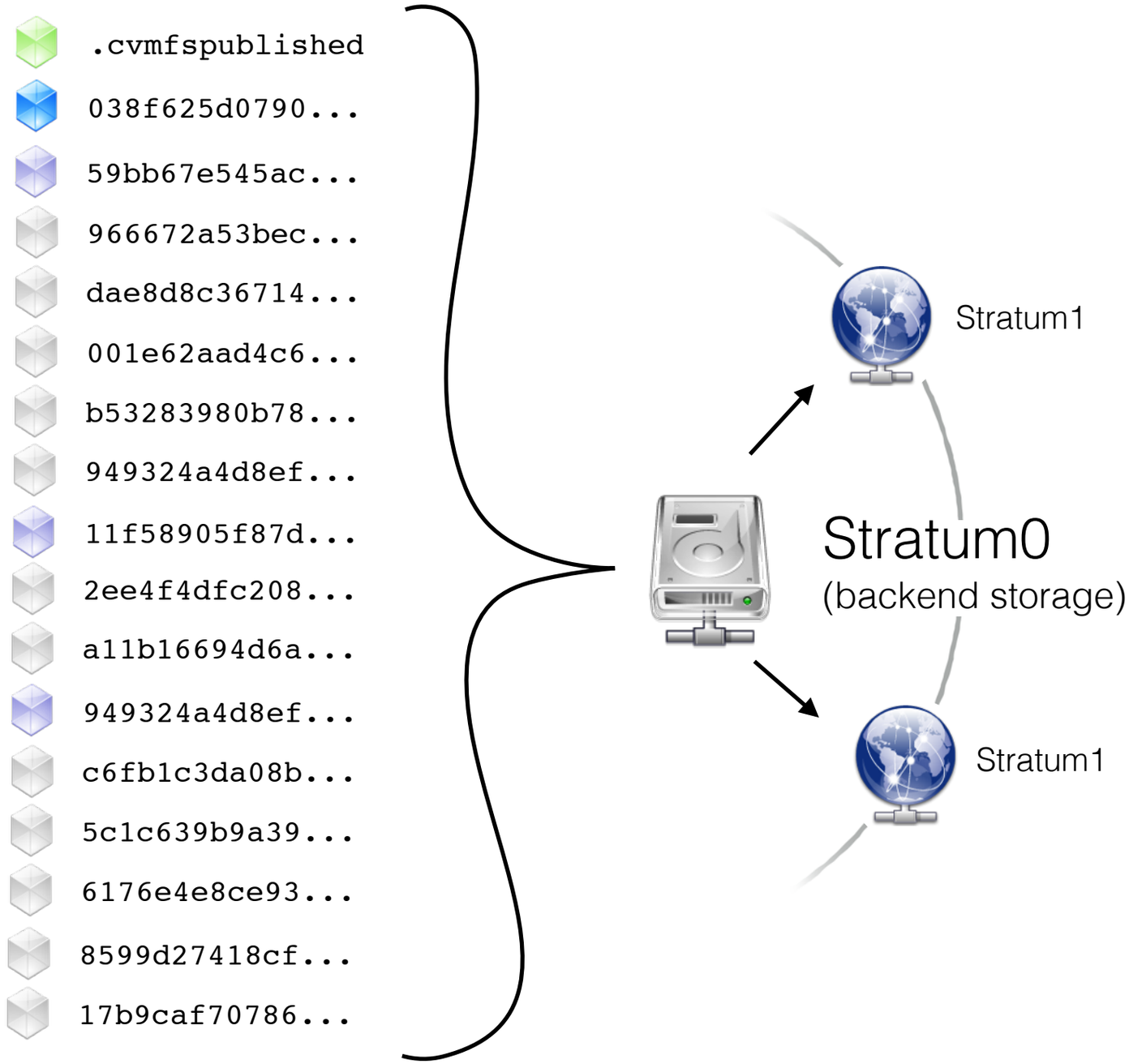
From POSIX to Blob-Objects

- **Merkle tree**
  - only .cvmfspublished needs to be signed
- **Content Addressable Storage**
  - File de-duplication
  - Trival file integrity checks
- **Flat Namespace**
  - Perfect for HTTP caching
  - Minimal storage API requirements (PUT, GET, [DELETE])



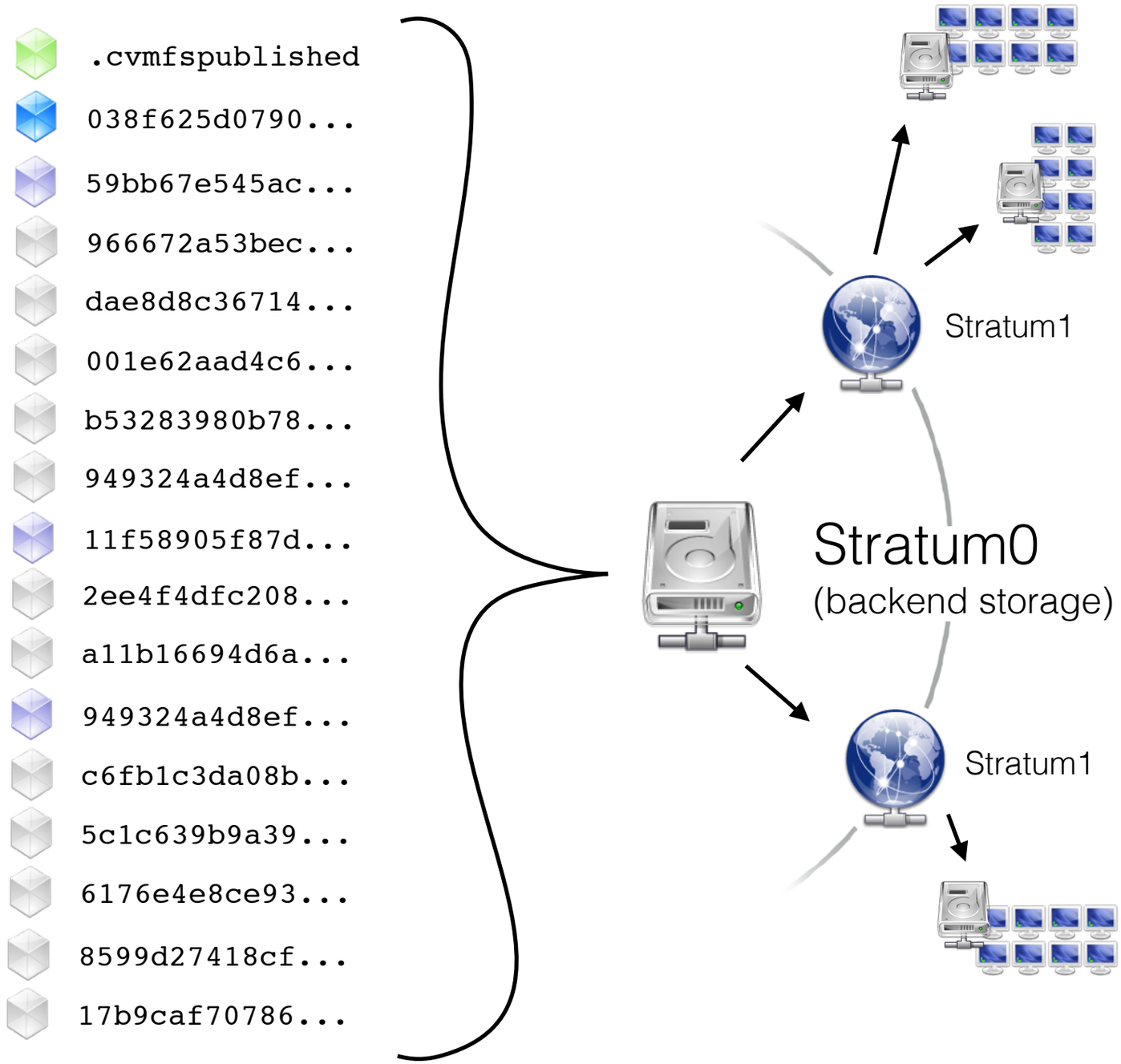
From POSIX to Blob-Objects

- **Merkle tree**
  - only .cvmfspublished needs to be signed
- **Content Addressable Storage**
  - File de-duplication
  - Trival file integrity checks
- **Flat Namespace**
  - Perfect for HTTP caching
  - Minimal storage API requirements (PUT, GET, [DELETE])



From POSIX to Blob-Objects

- **Merkle tree**
  - only .cvmfspublished needs to be signed
- **Content Addressable Storage**
  - File de-duplication
  - Trival file integrity checks
- **Flat Namespace**
  - Perfect for HTTP caching
  - Minimal storage API requirements (PUT, GET, [DELETE])



Repository Storage

- **RESTful backend storage** - PUT, GET, (DELETE)
  - POSIX compliant file systems
  - Key-Value stores (using S3 API - Seppo Heikkila)

Stratum0
(backend storage)

- **Pluggable storage connector implementation**
  - Facilitate implementation of native Key-Value store APIs
  - Like: Basho Riak, Ceph, Amazon Dynamo, …

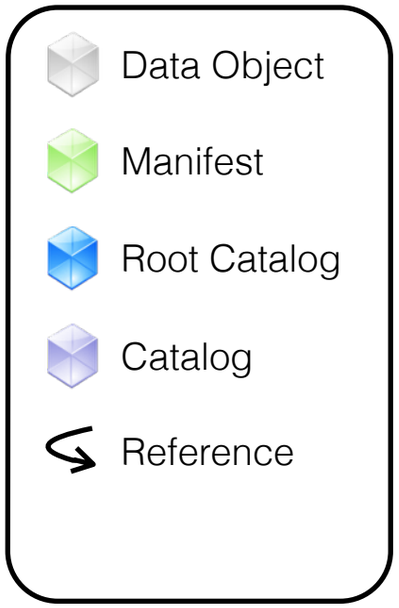
File System History

Named Repository Snapshots

File System History

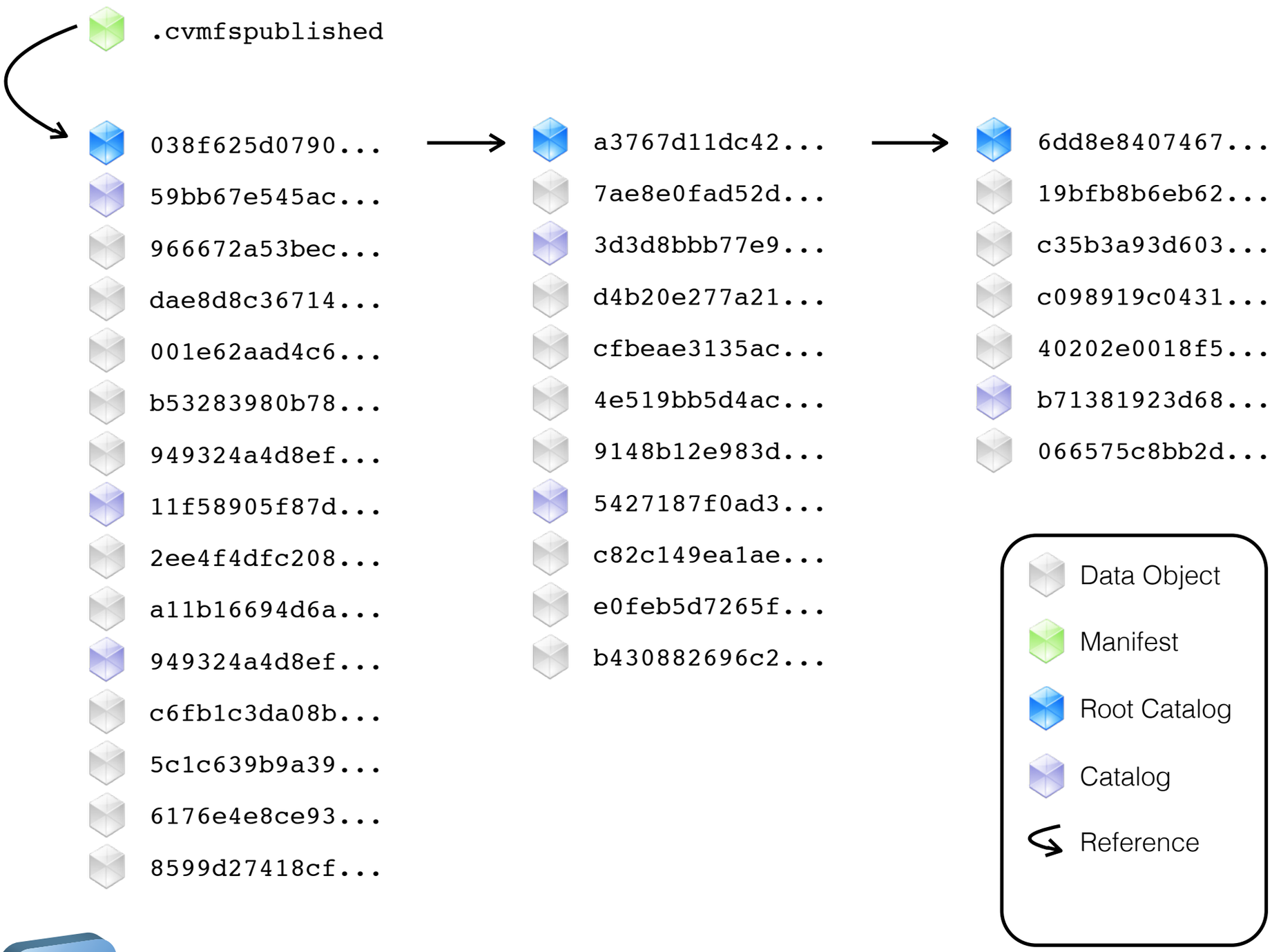
- **Client:** Mount historic revisions of a repository
  - Use legacy software in its contemporary environment
  - Long term data preservation
- **Stratum0:** Rollback to previous revisions
  - Undo - Overwrite broken revisions with previous version

File System History

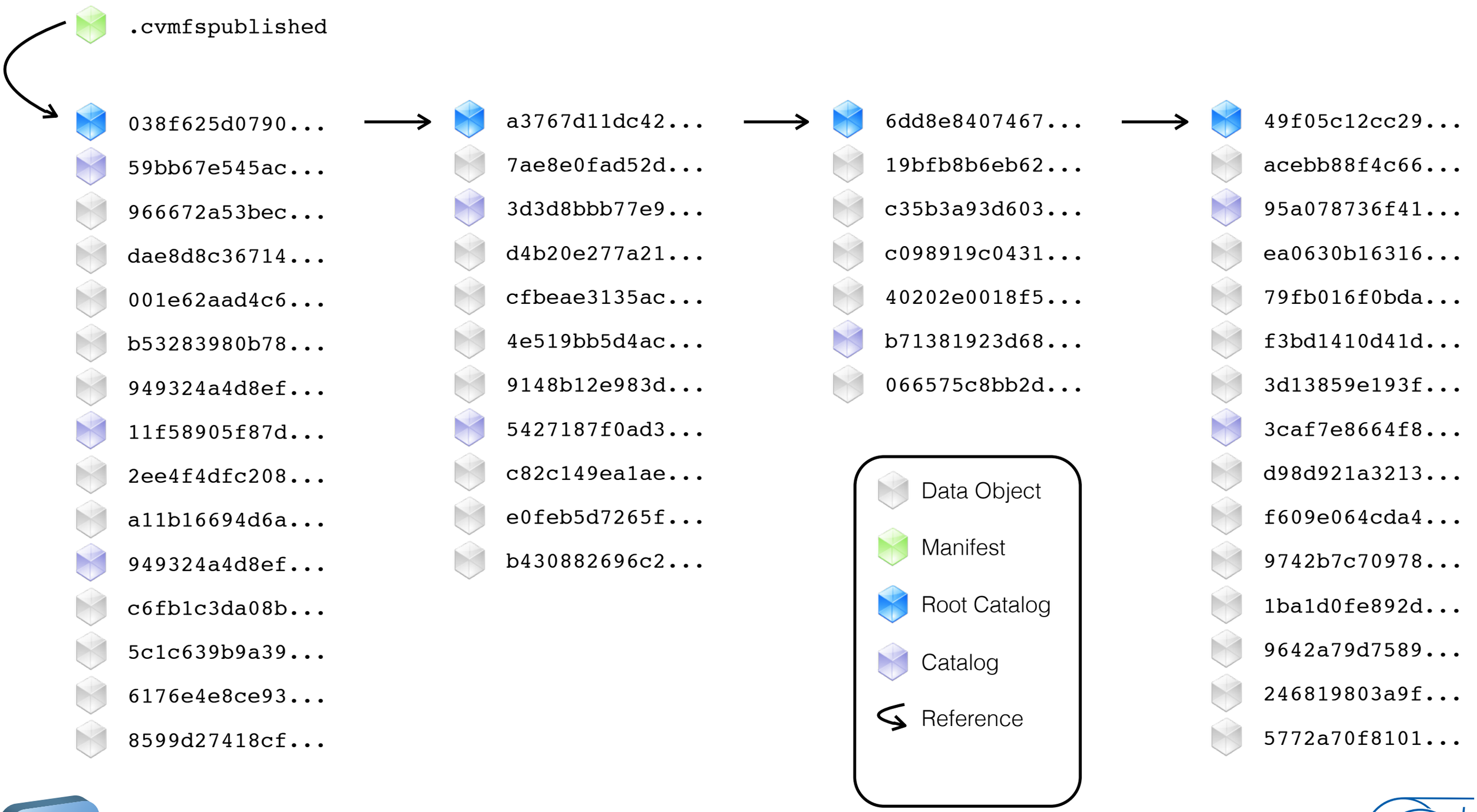


File System History

File System History

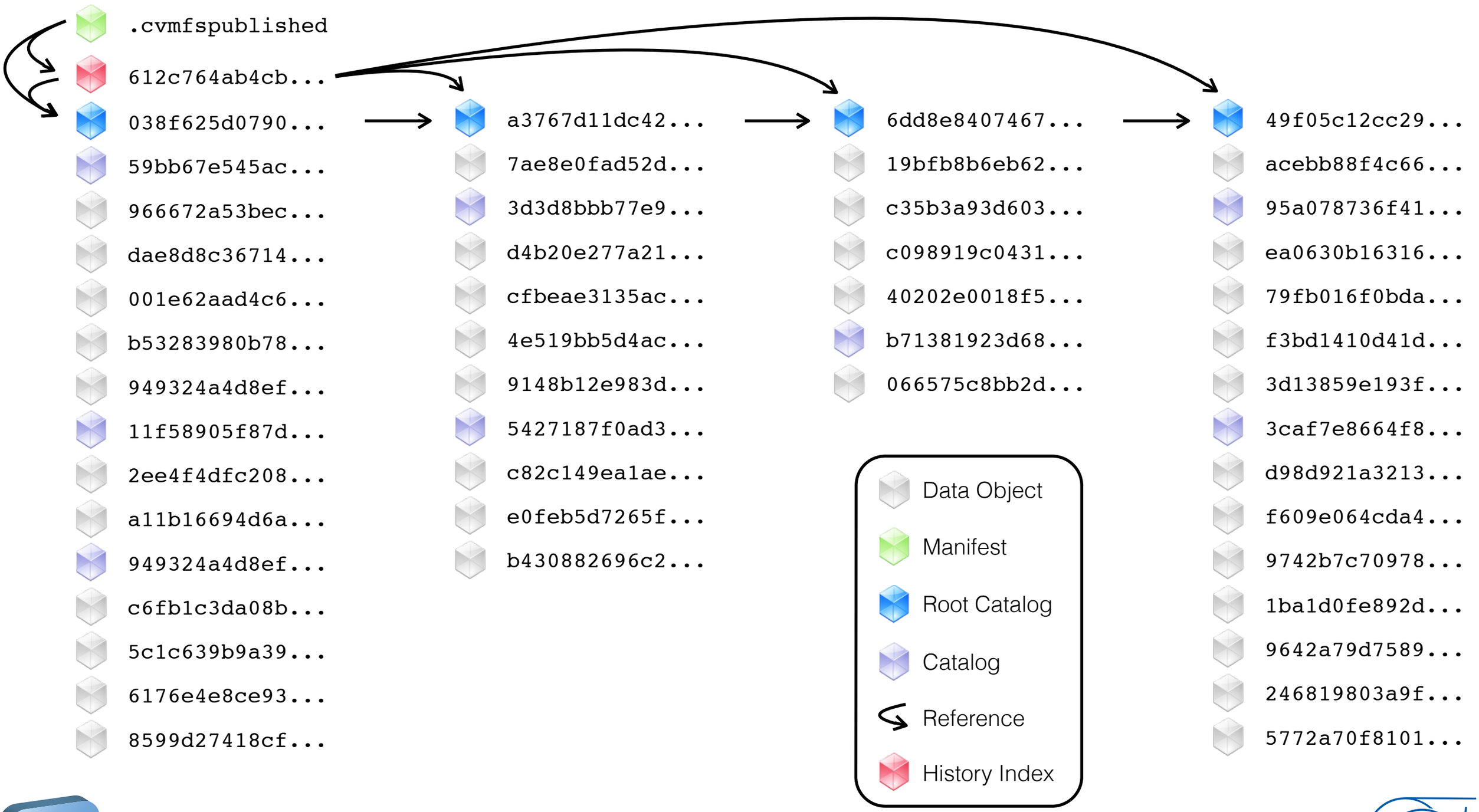


File System History

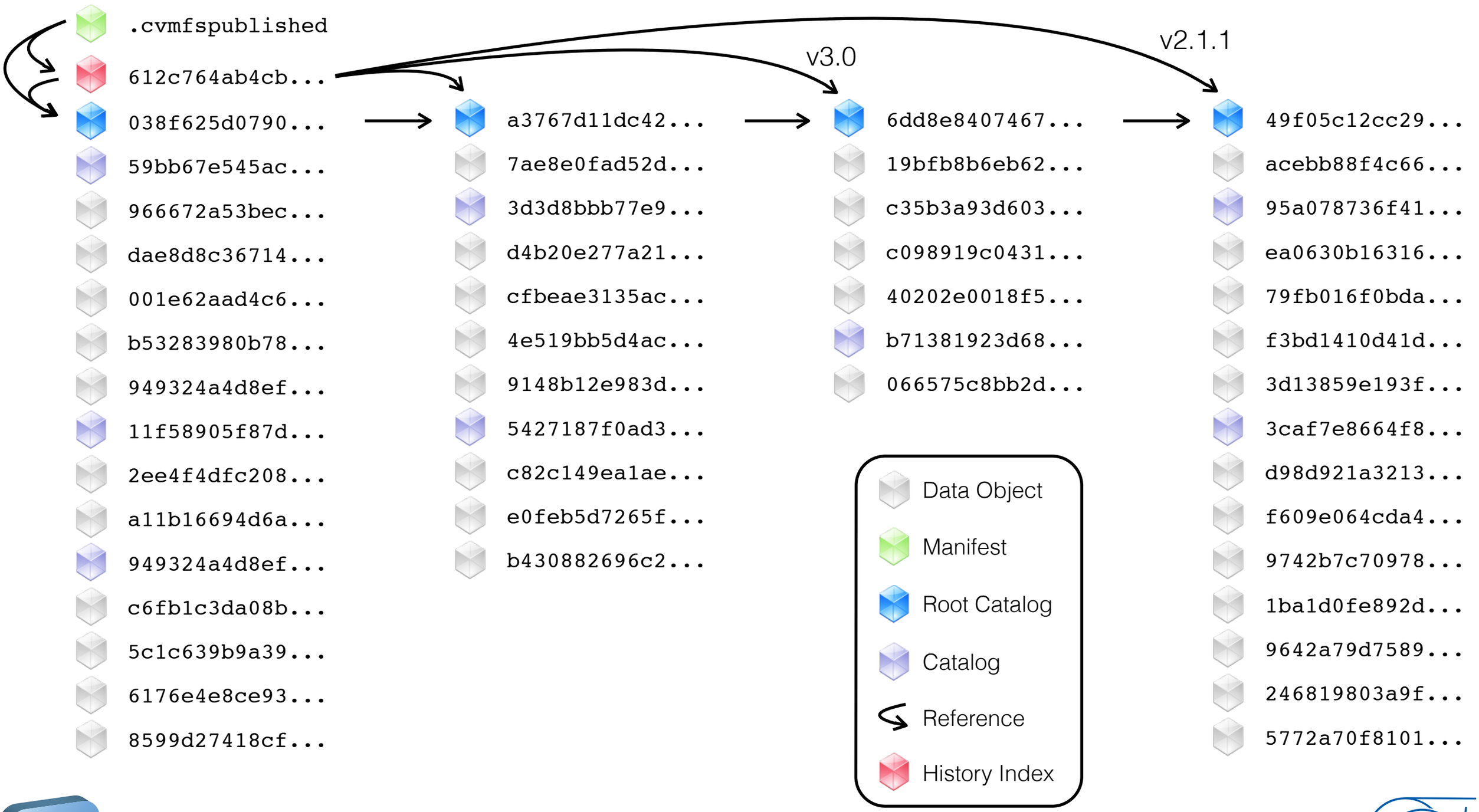


File System History

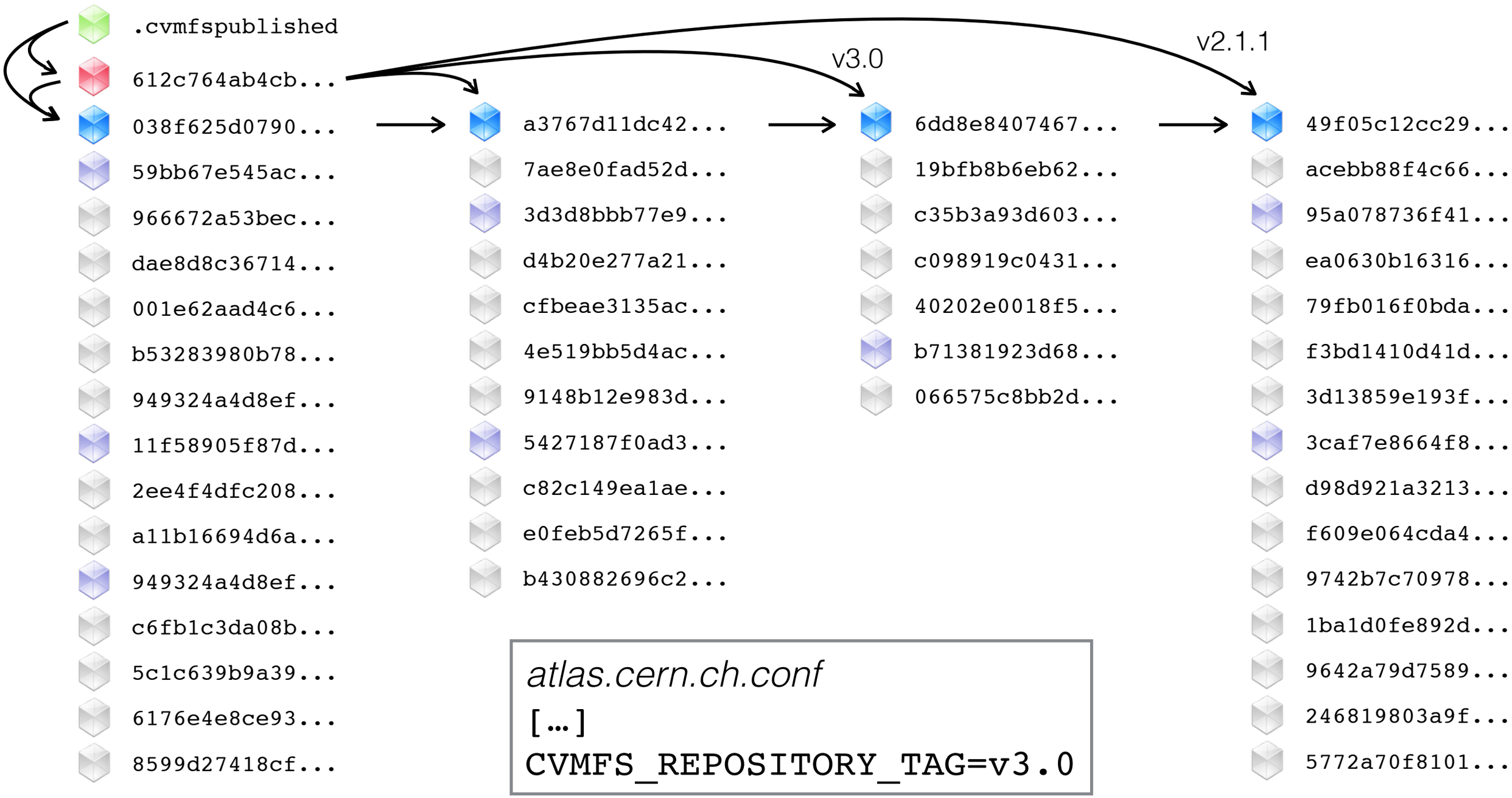
File System History



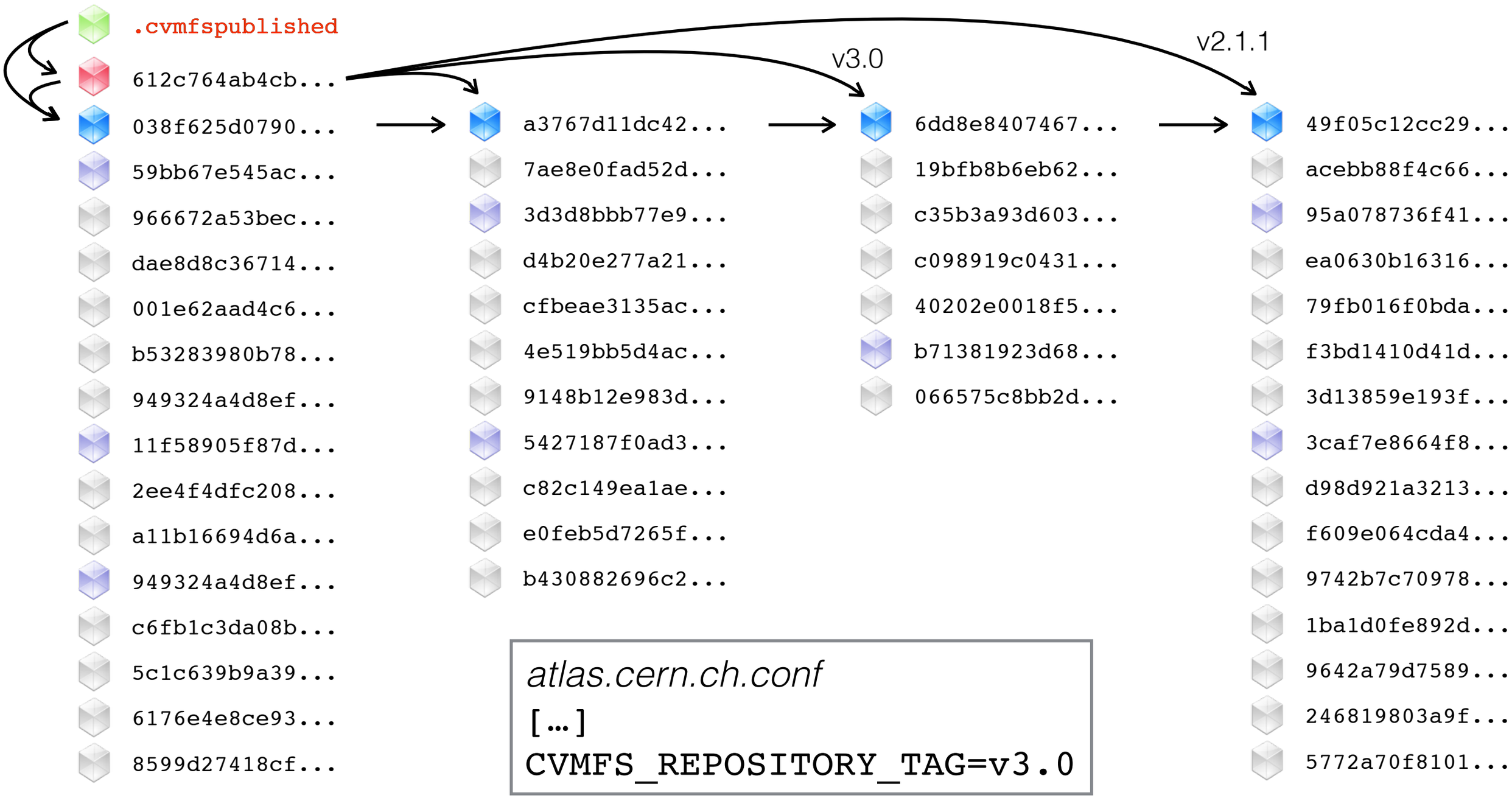
File System History



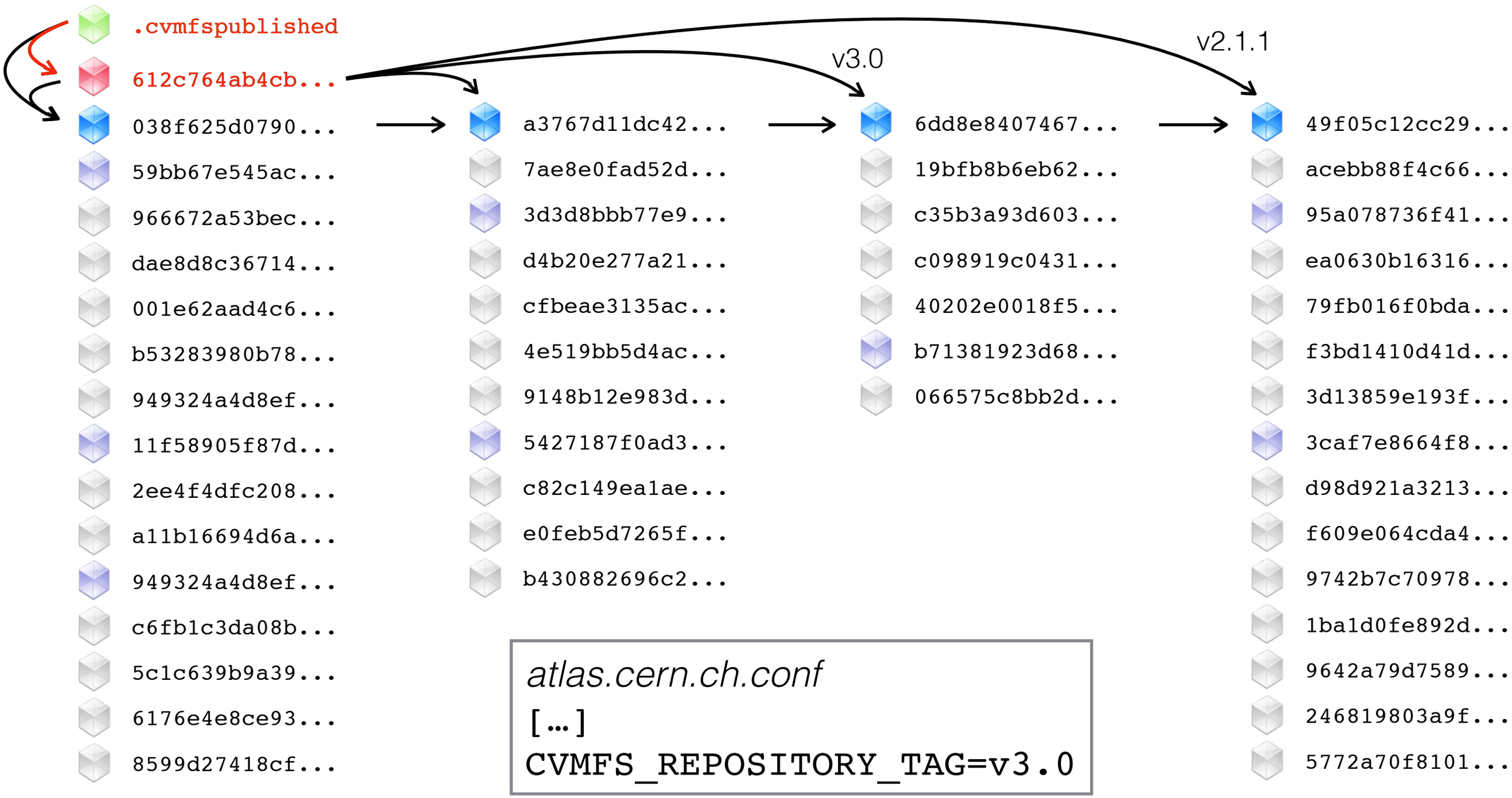
File System History



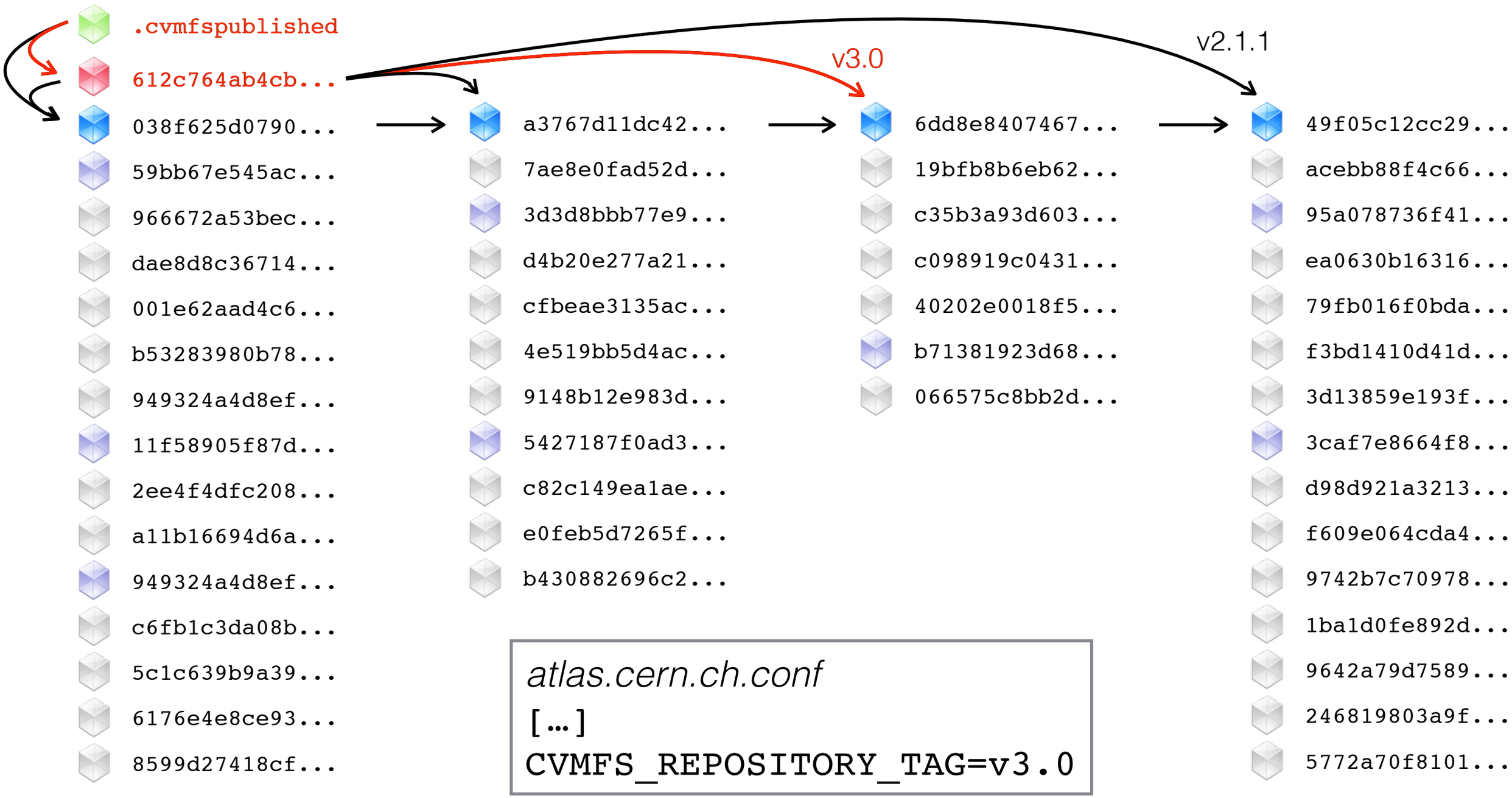
File System History



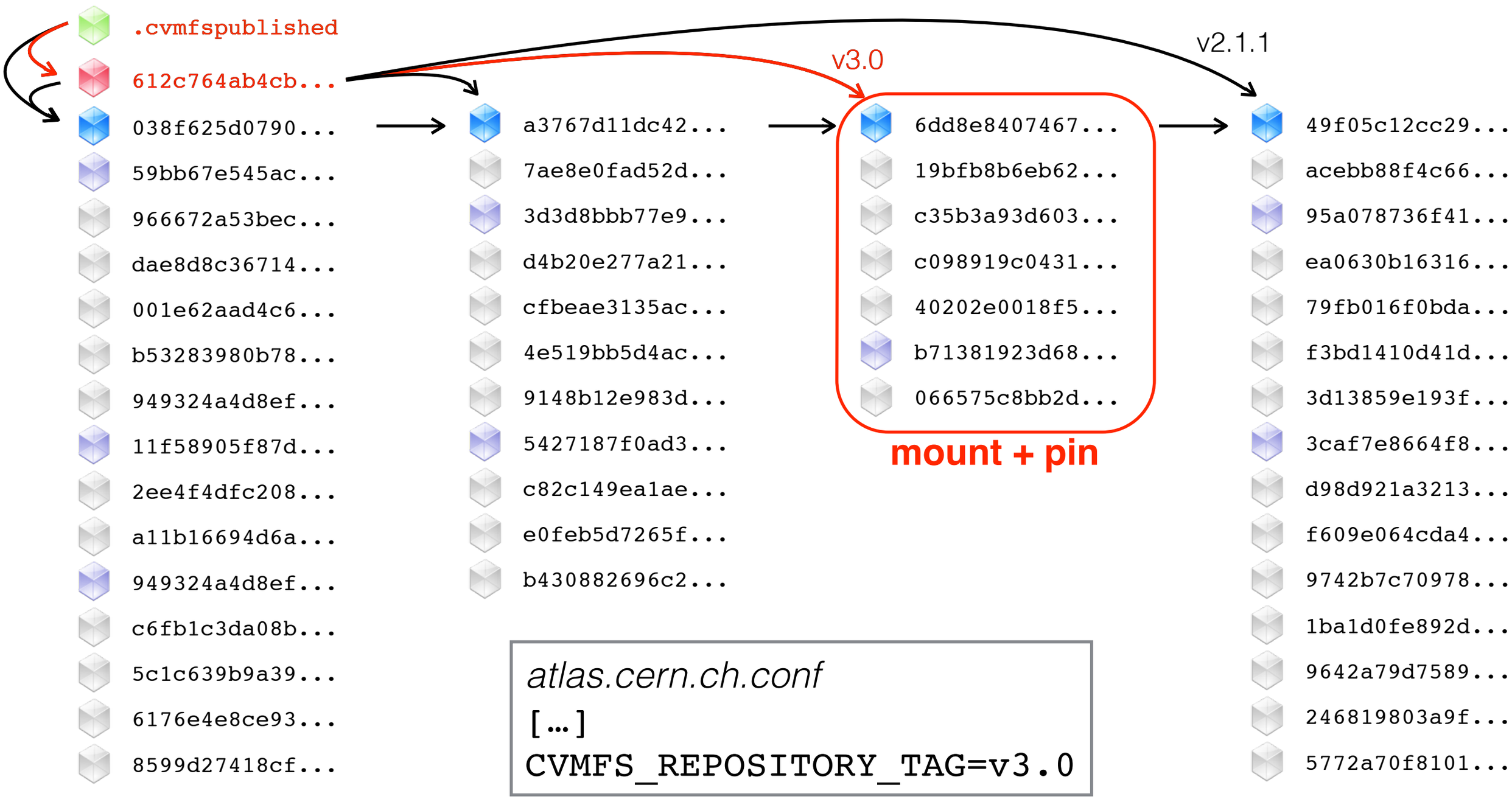
File System History



File System History



File System History



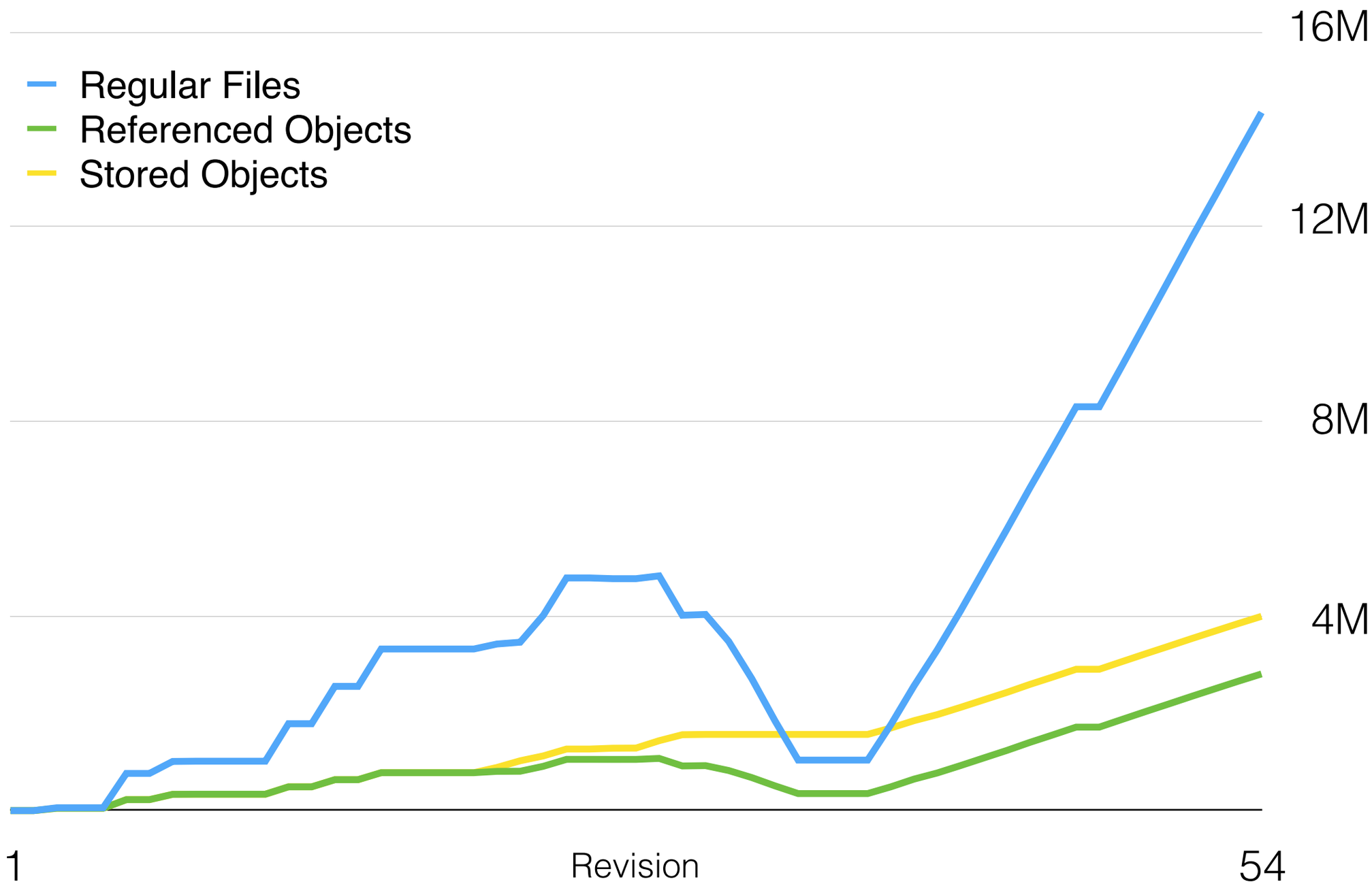
Garbage Collection

Permanently Removing
Overwritten or Deleted Files
in Volatile Repositories

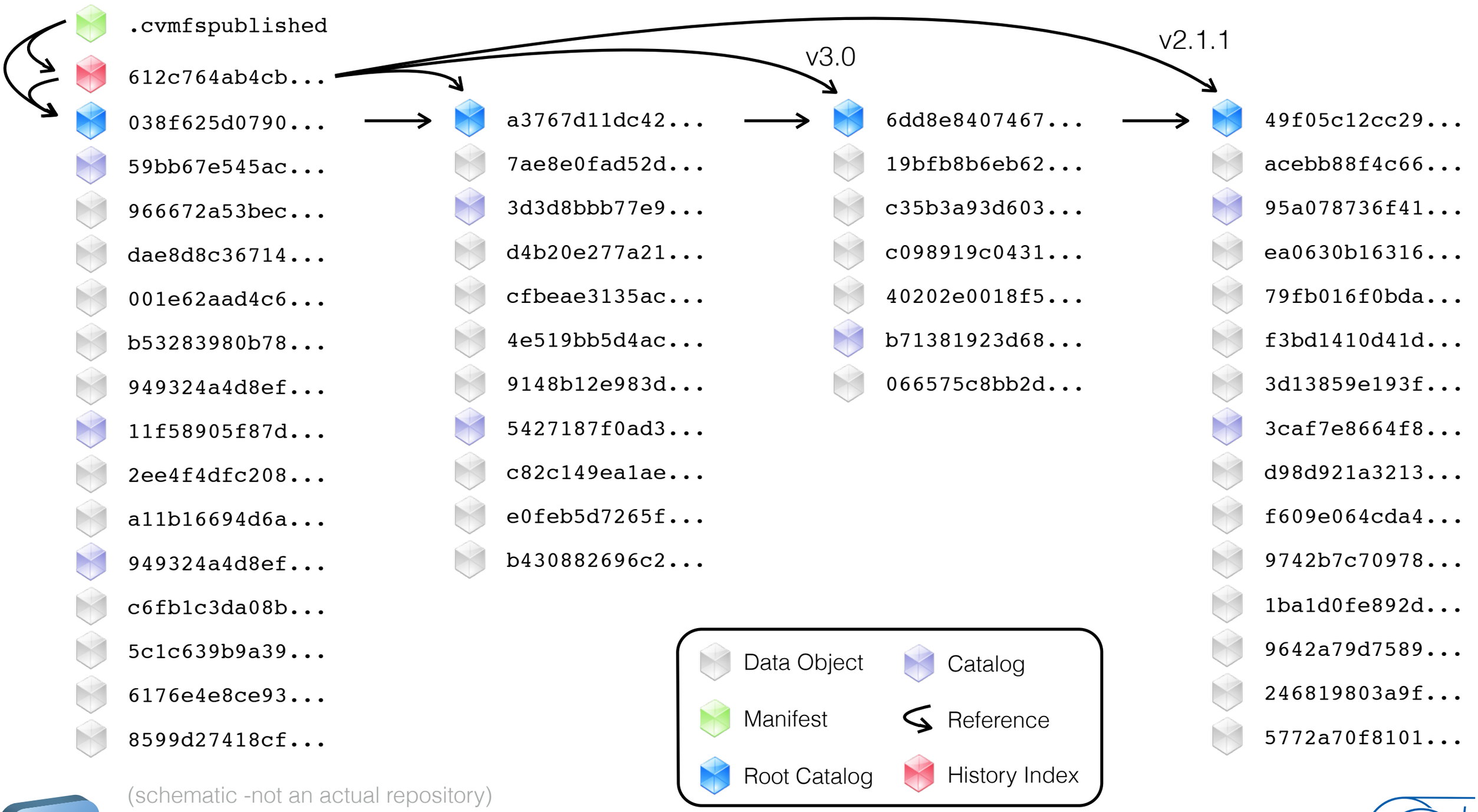
Garbage Collection

- CernVM-FS backend initially designed as **insert-only**
- **New use-case:** LHC experiment's nightly build repositories
  - high update rate (up to twice a day)
  - large volume of newly staged files (10-100GiB)
  - short lived revisions (stay online max. two weeks)
- Insert-only quickly fills up backend storage!

LHCb Nightly Builds

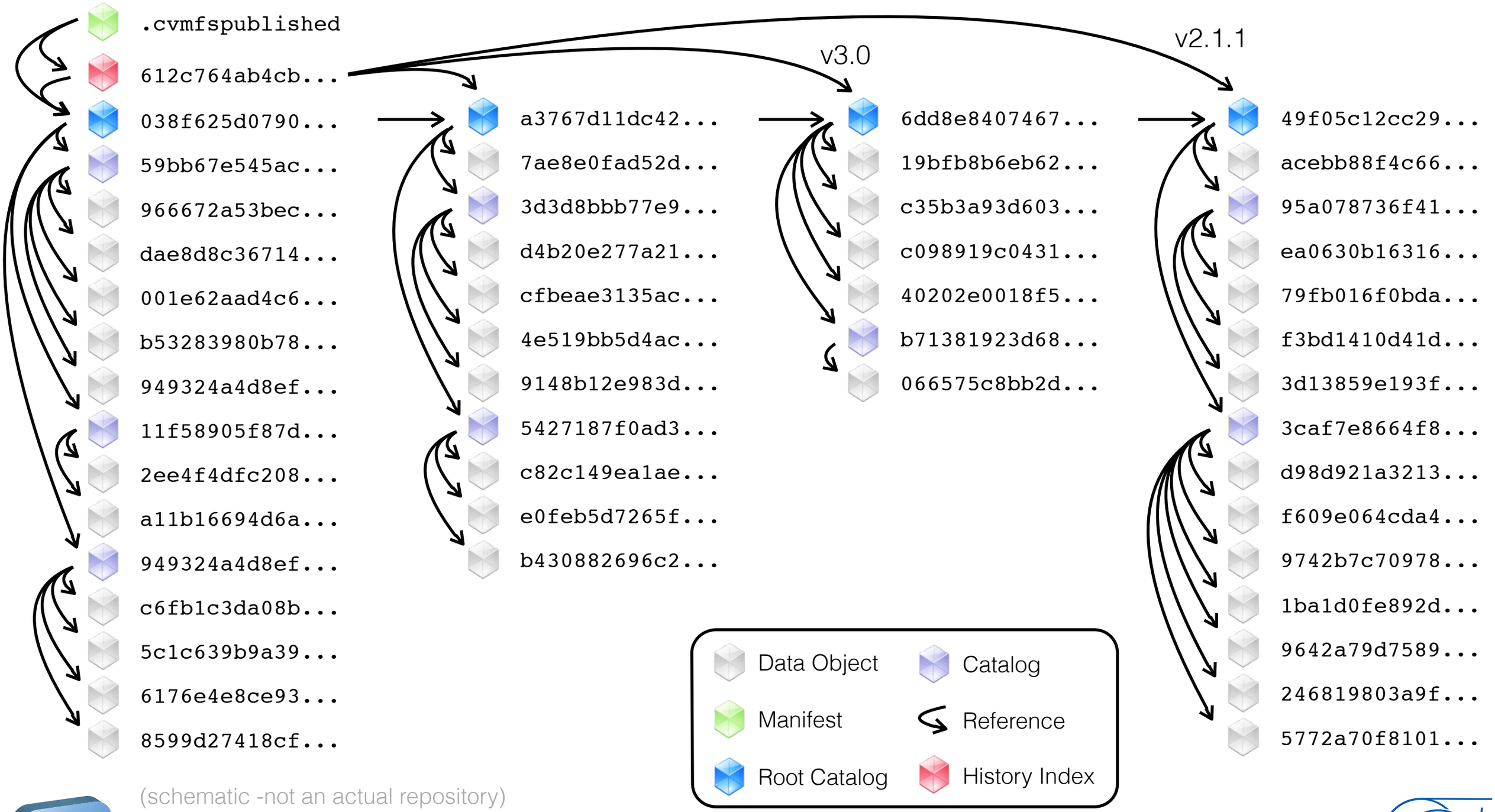


Garbage Collection



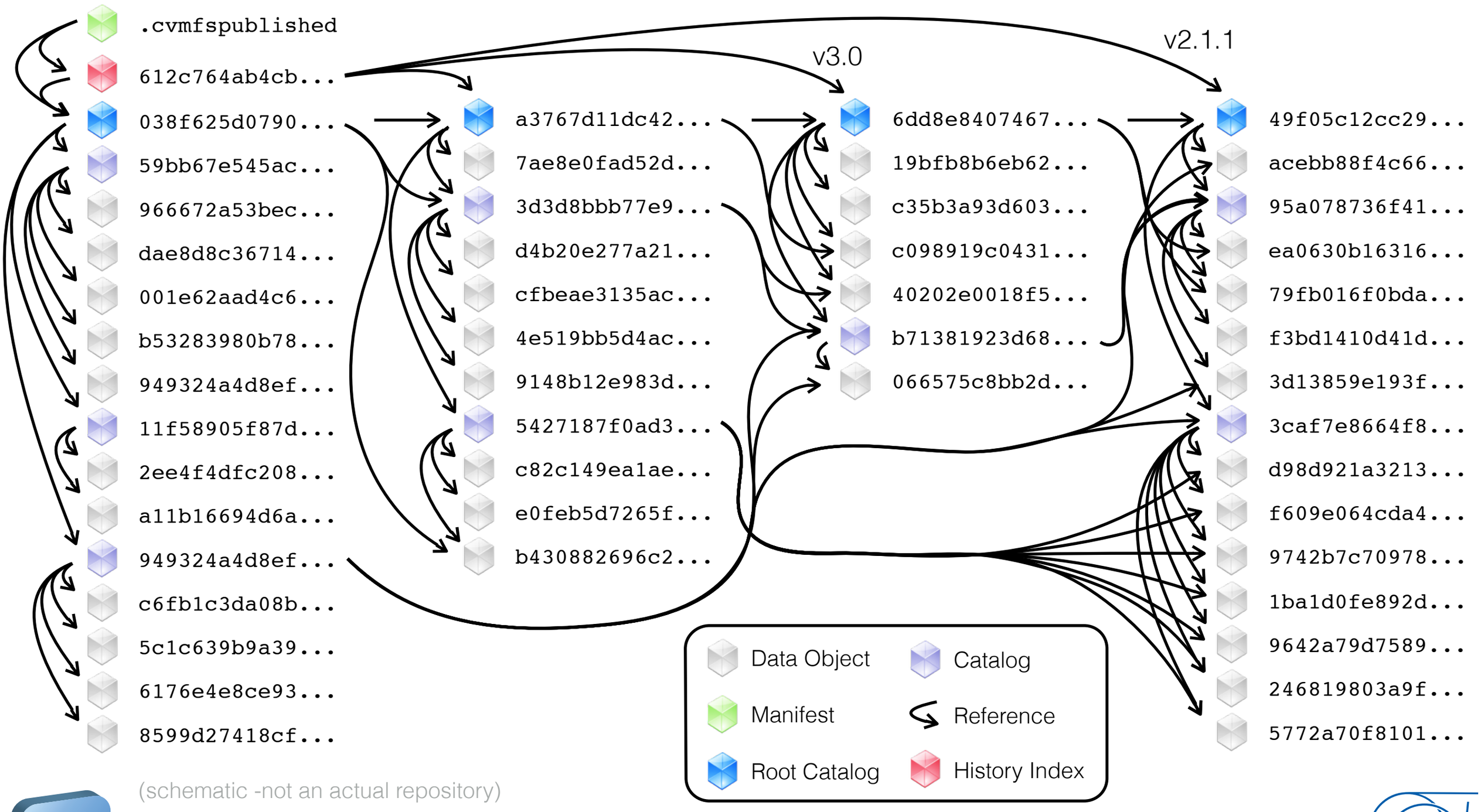
(schematic -not an actual repository)

Garbage Collection



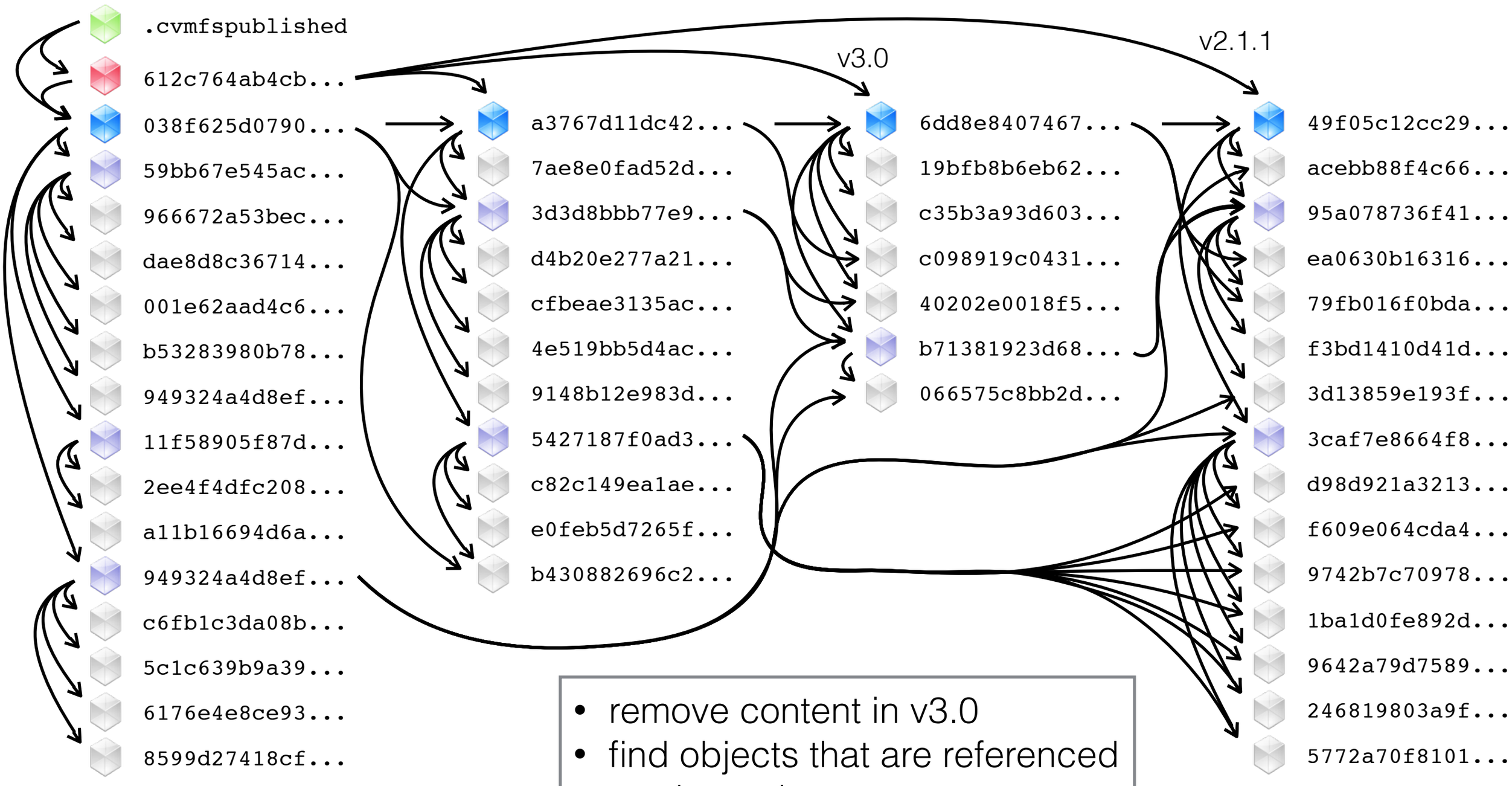
(schematic -not an actual repository)

Garbage Collection



(schematic -not an actual repository)

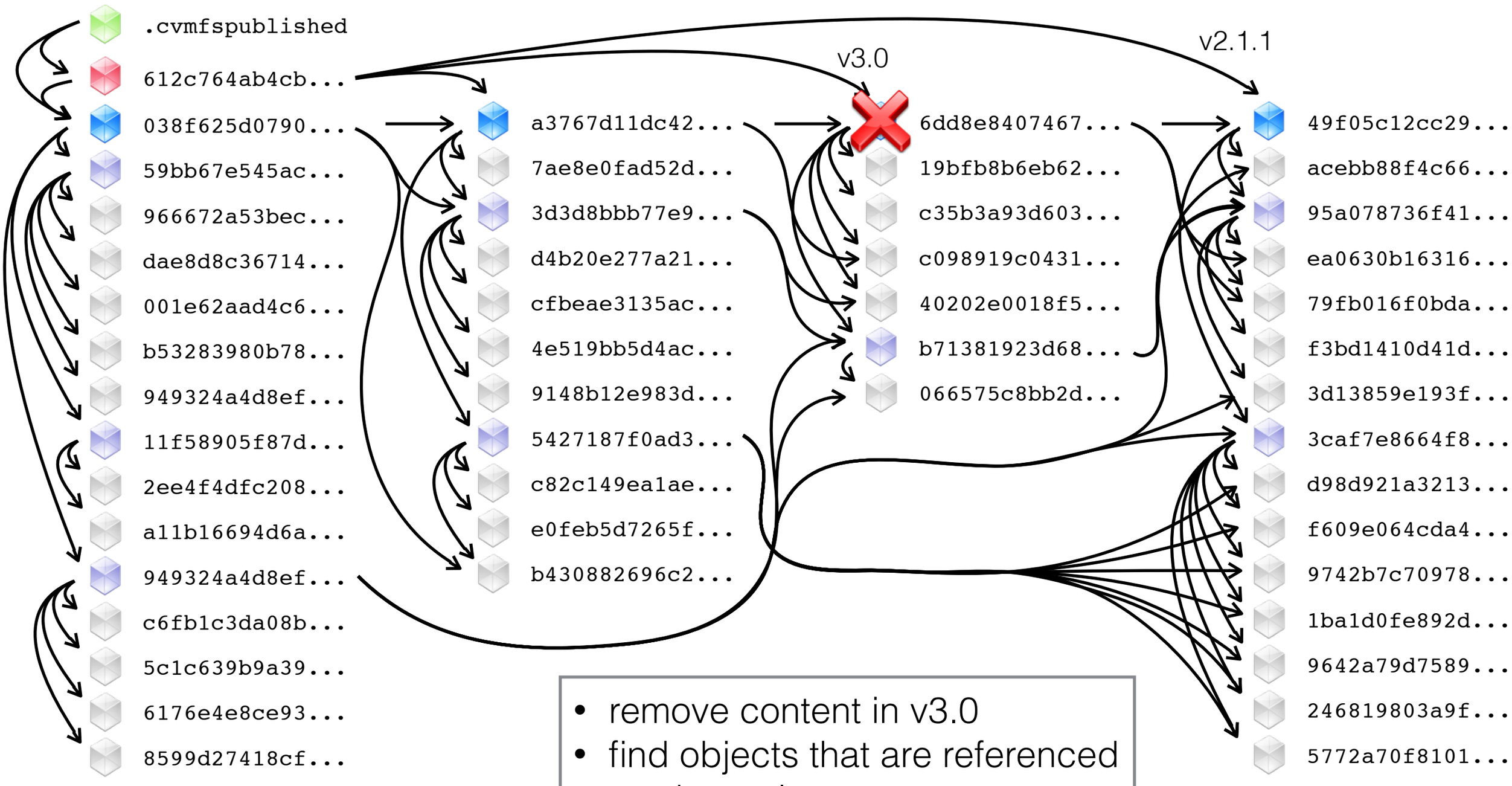
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

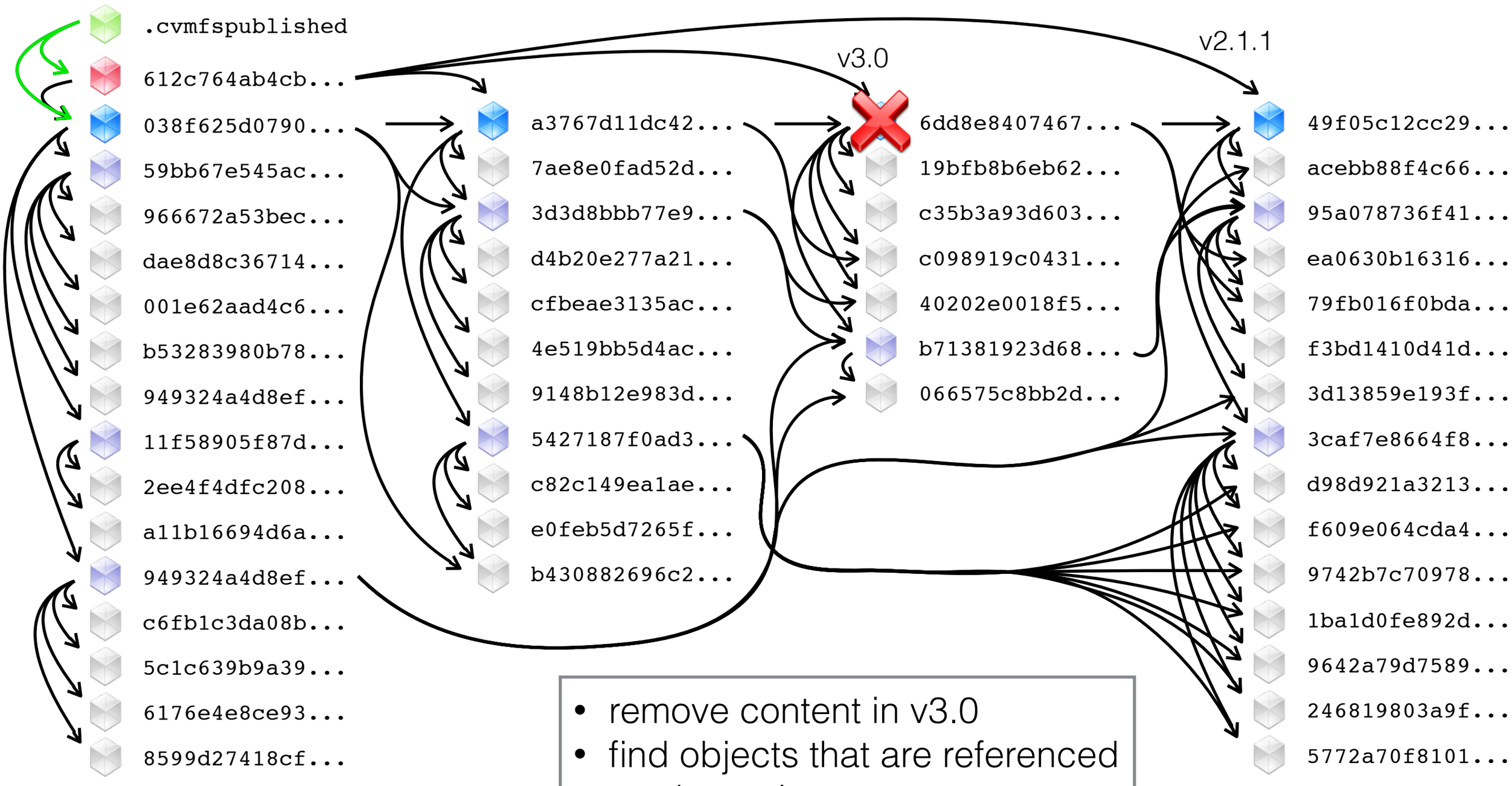
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

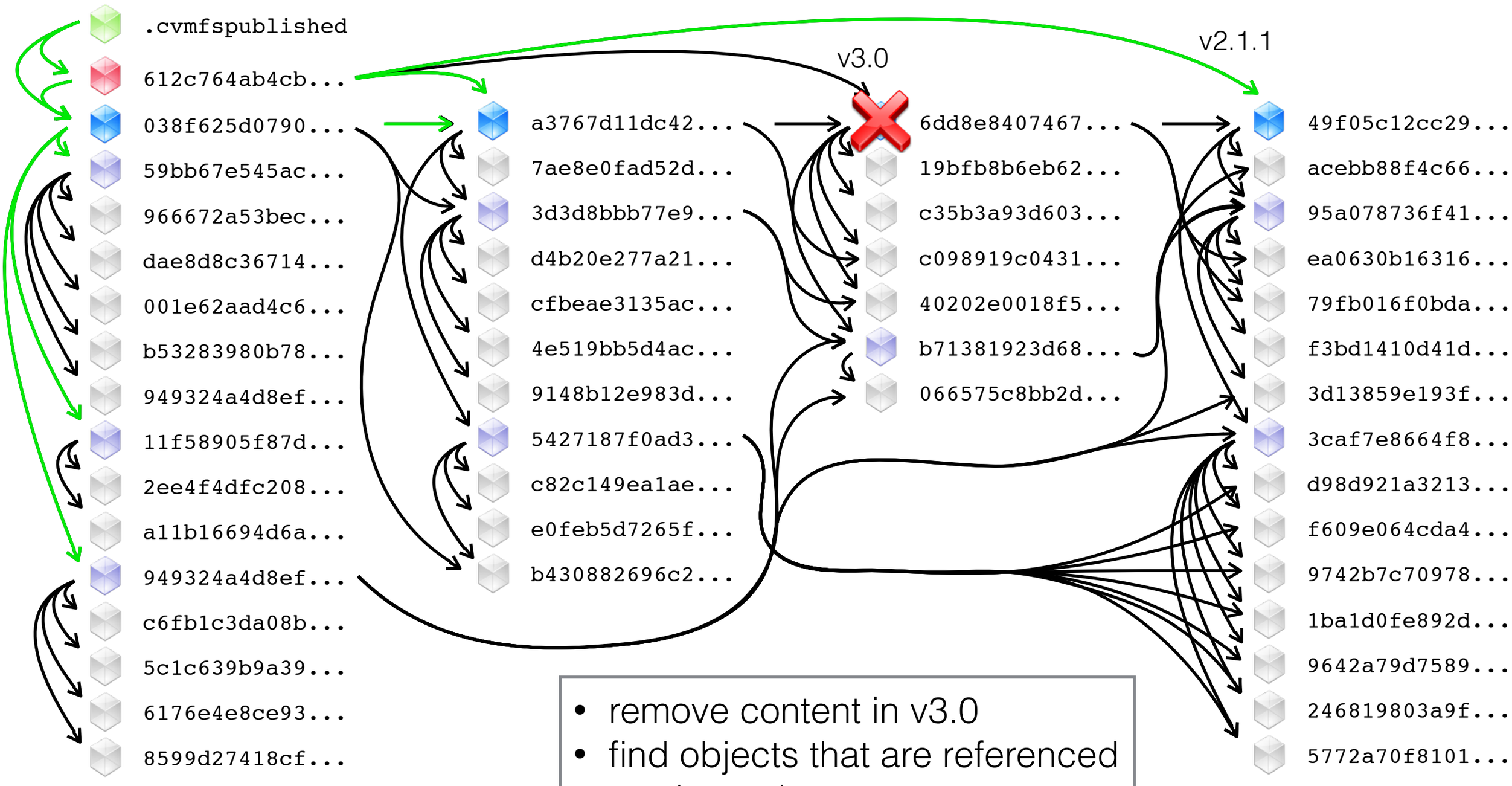
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

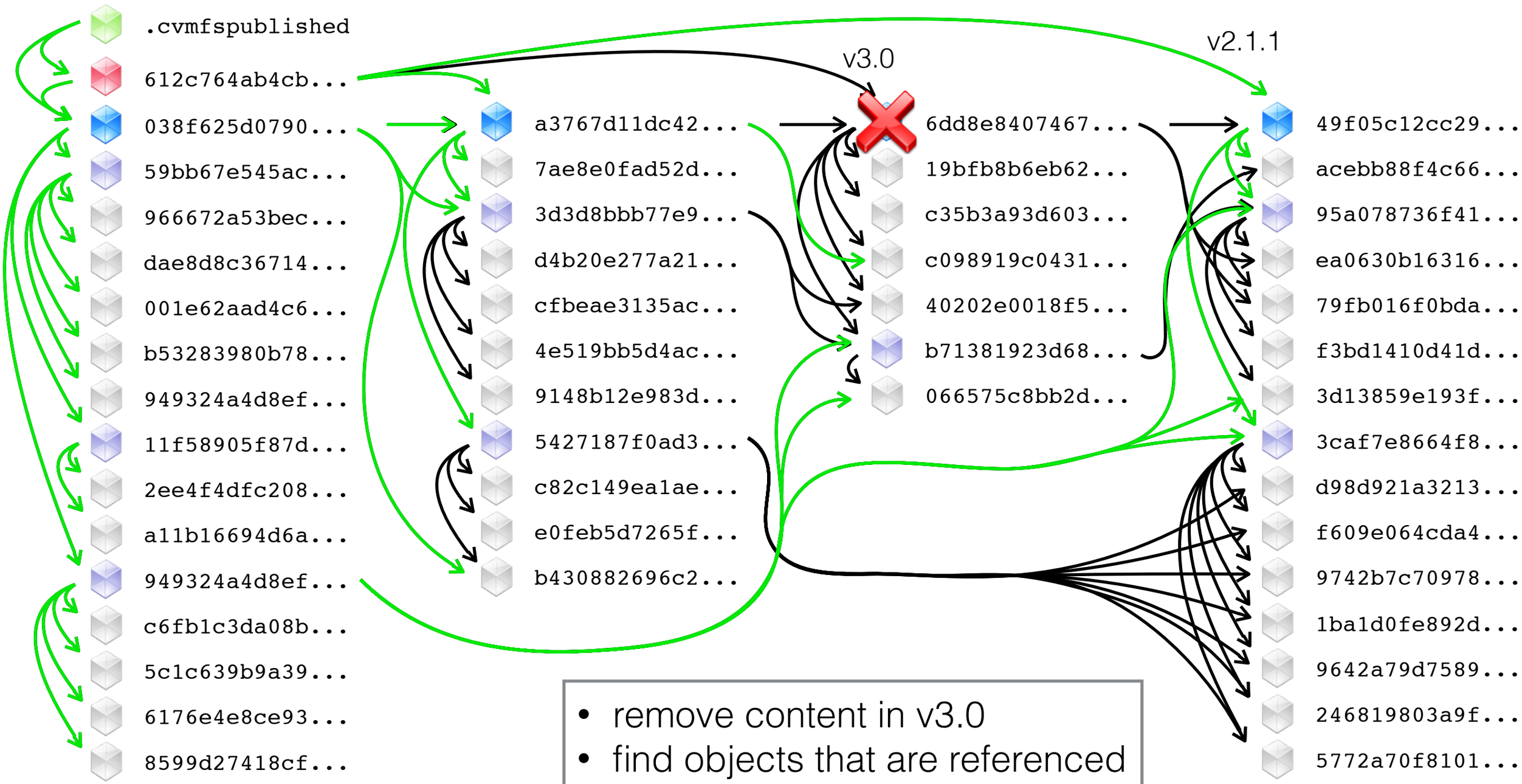
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

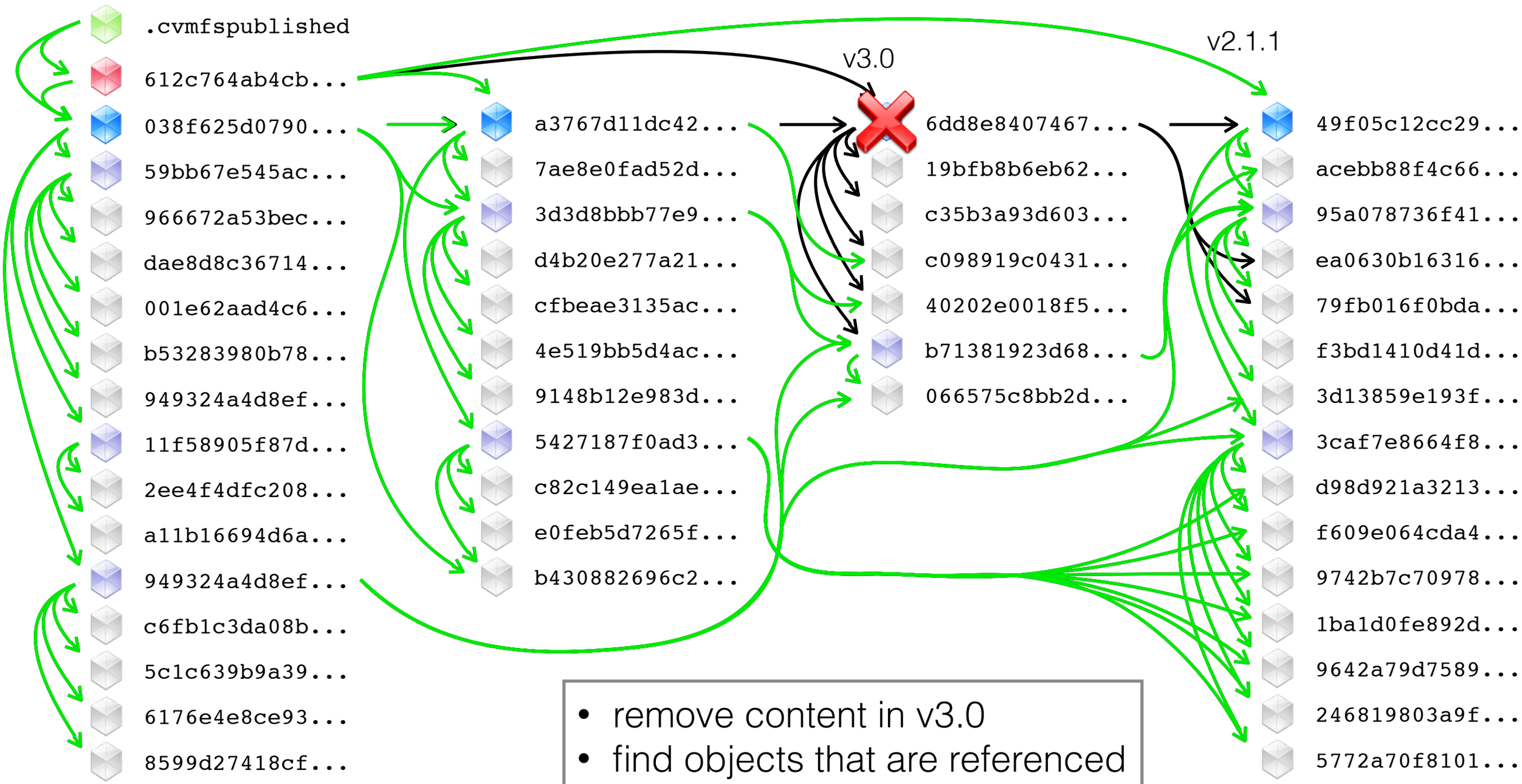
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

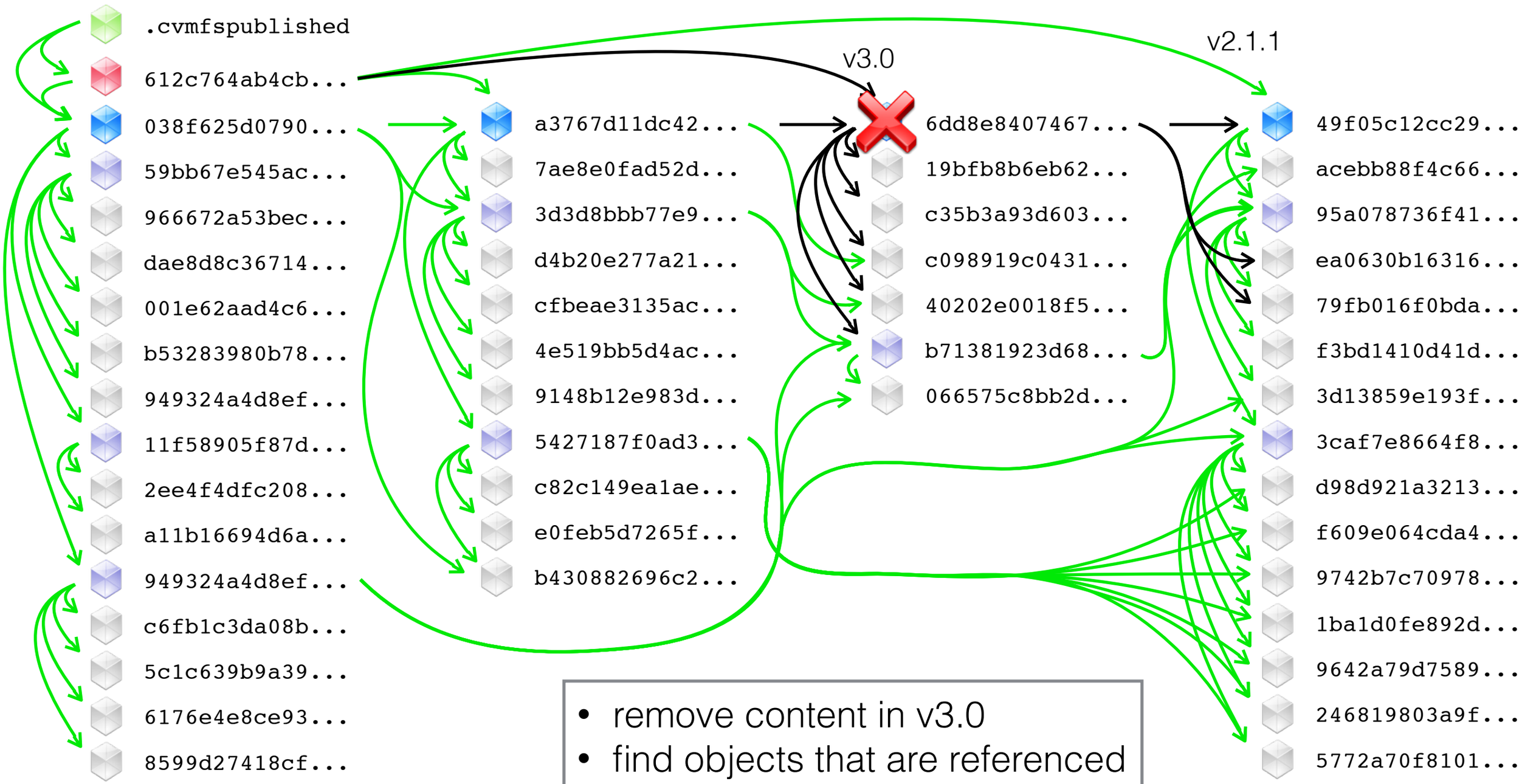
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

Garbage Collection

(schematic -not an actual repository)

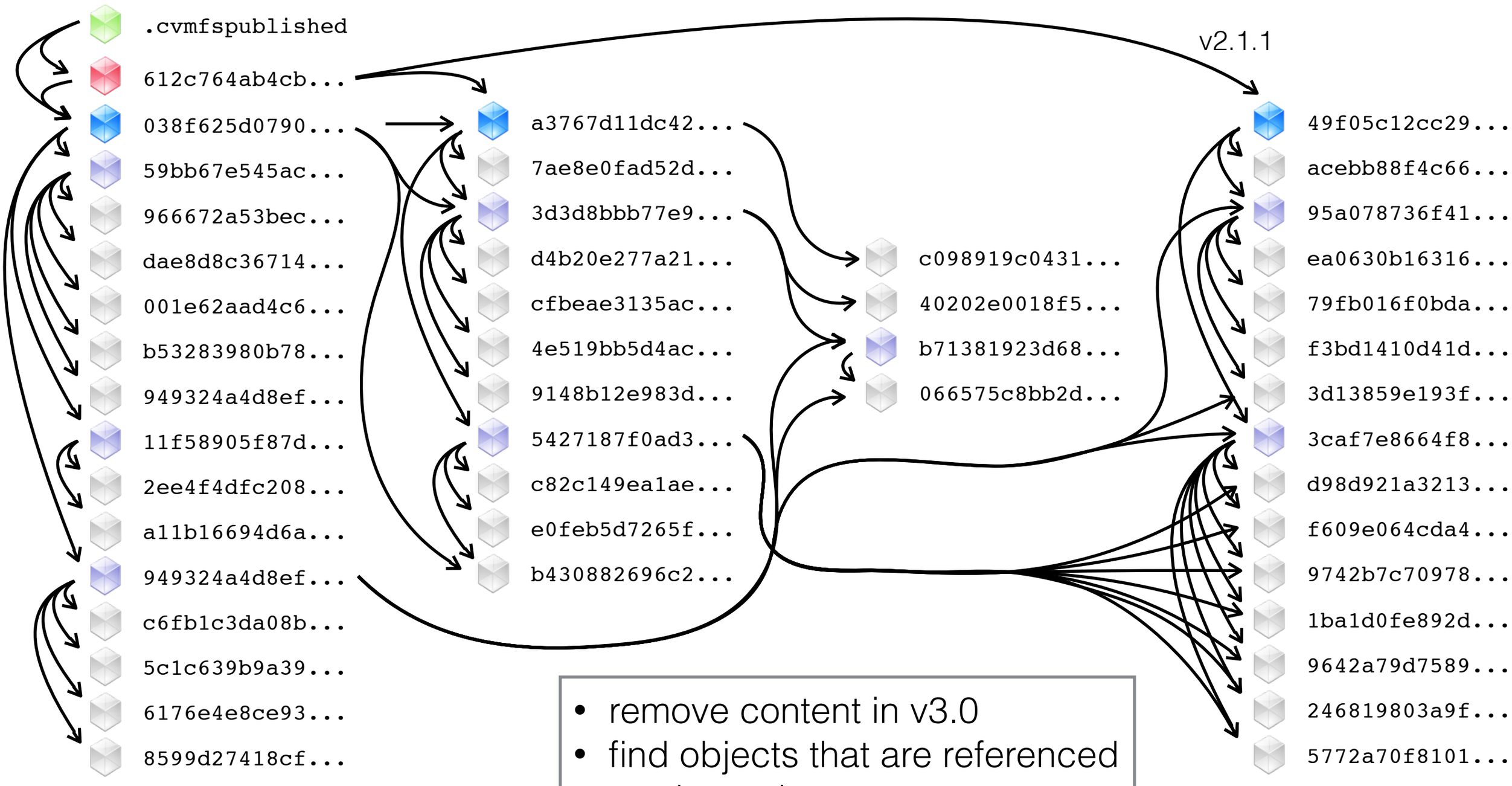
- remove content in v3.0
- find objects that are referenced nowhere else

Garbage Collection

(schematic -not an actual repository)

- remove content in v3.0
- find objects that are referenced nowhere else

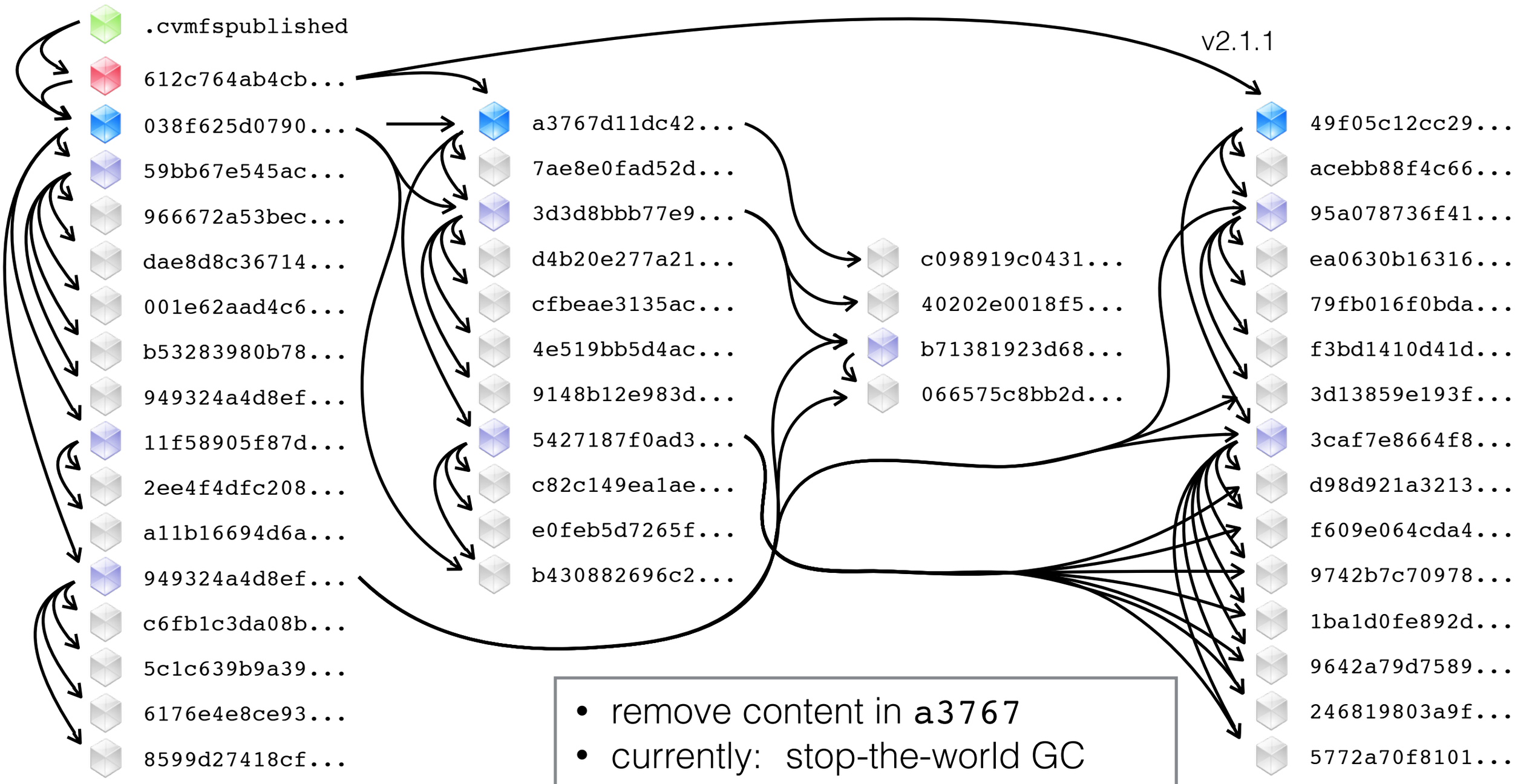
Garbage Collection



- remove content in v3.0
- find objects that are referenced nowhere else

(schematic -not an actual repository)

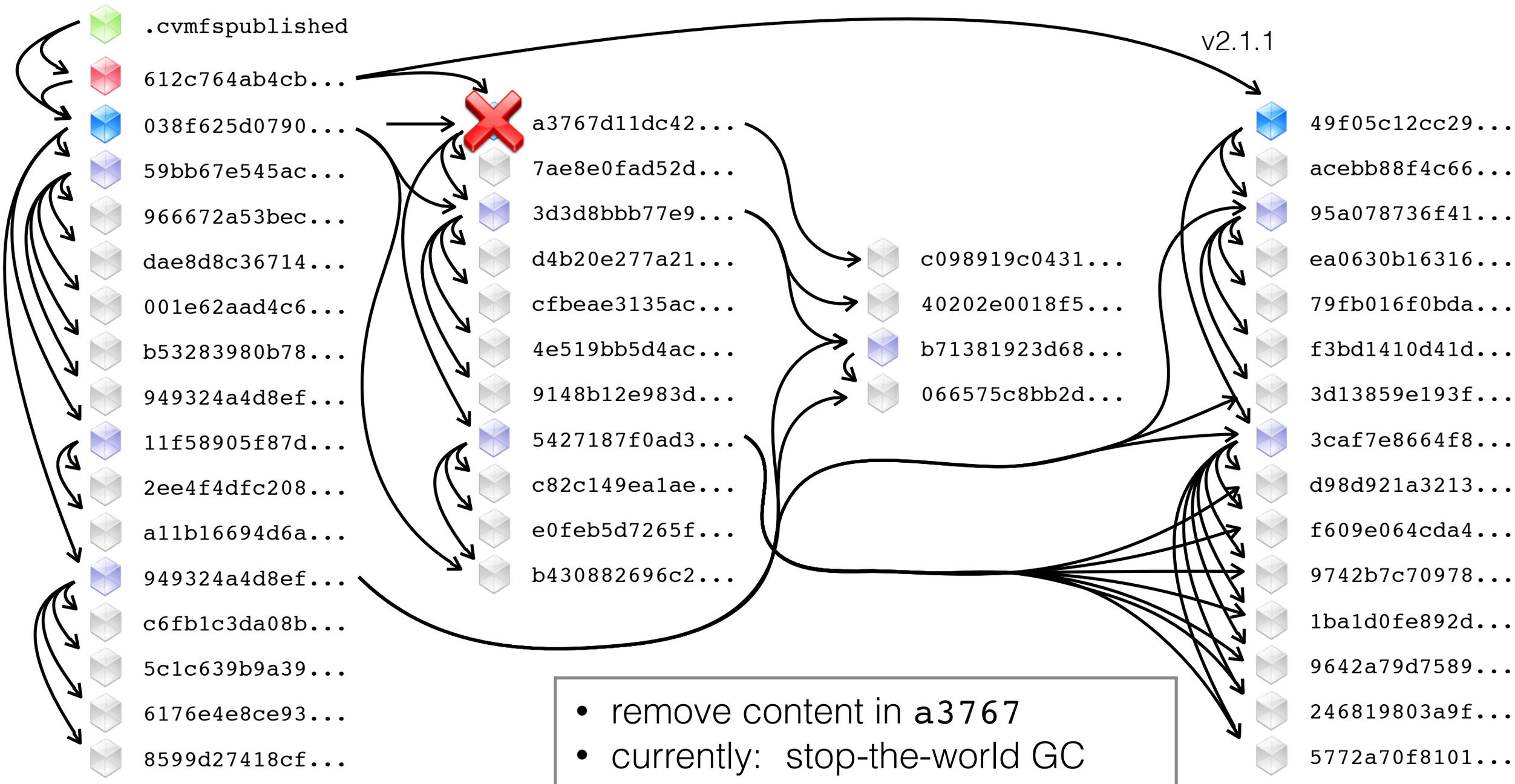
Garbage Collection



- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

(schematic -not an actual repository)

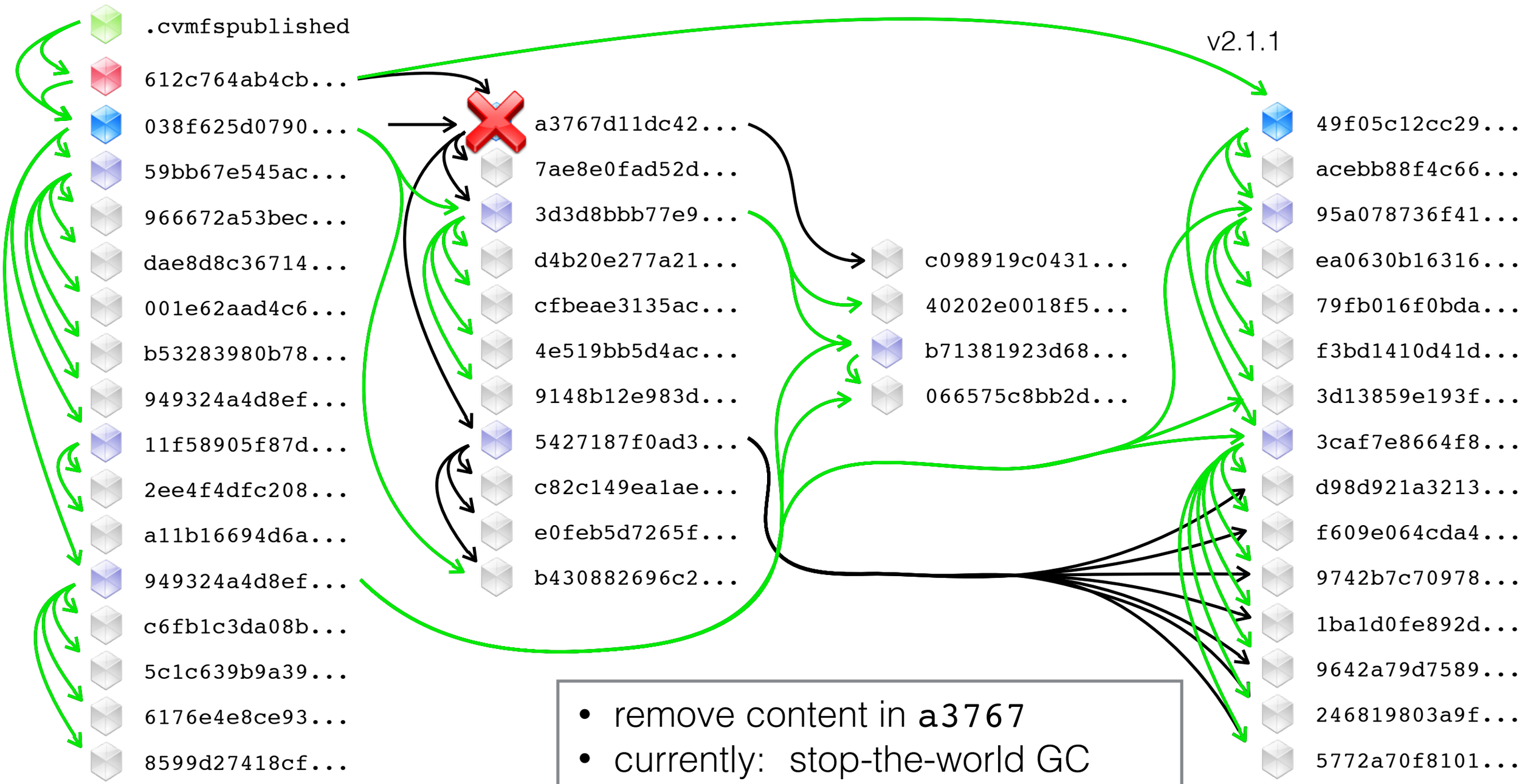
Garbage Collection



- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

(schematic -not an actual repository)

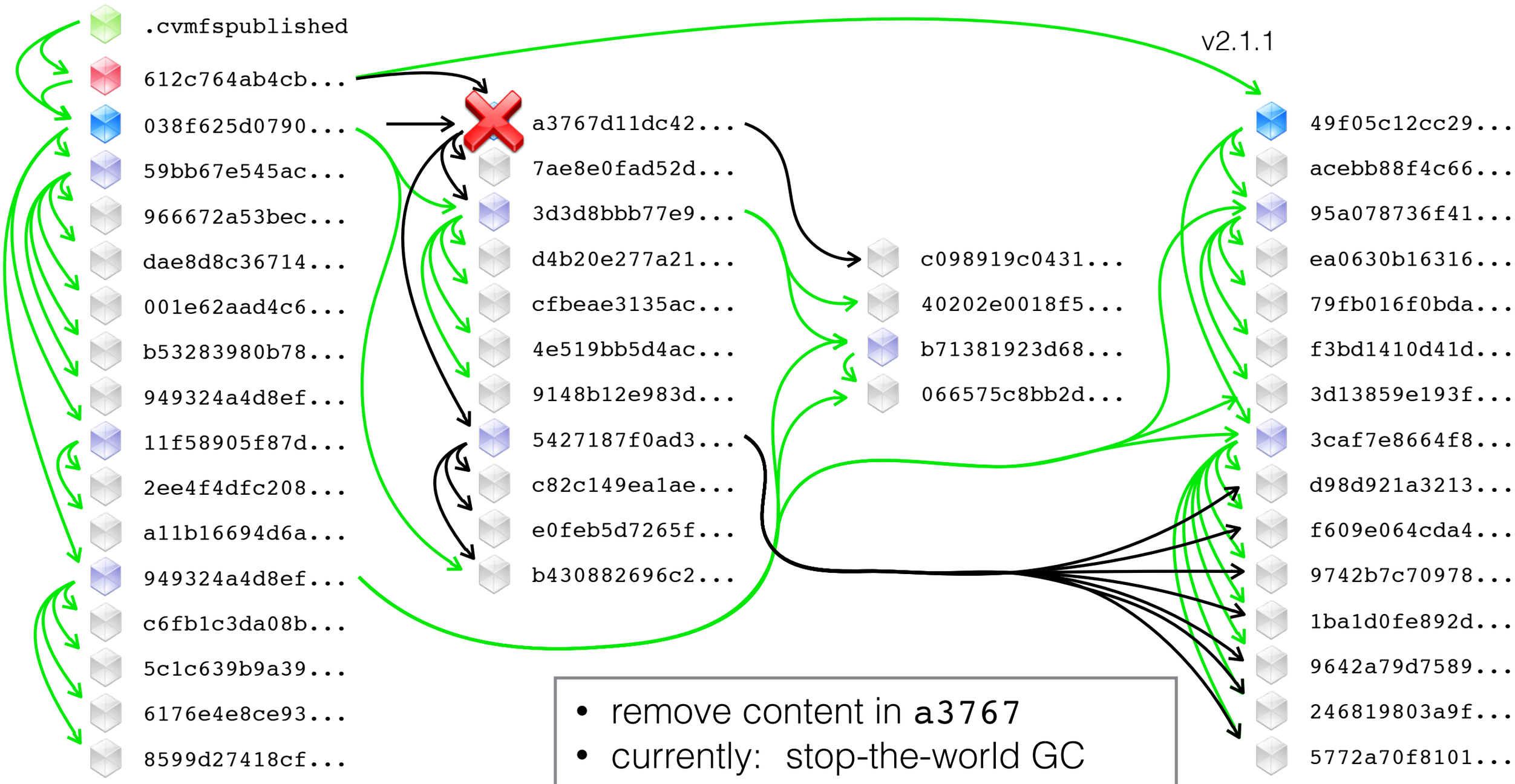
Garbage Collection



- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

(schematic -not an actual repository)

Garbage Collection



- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

(schematic -not an actual repository)

Garbage Collection

(schematic -not an actual repository)

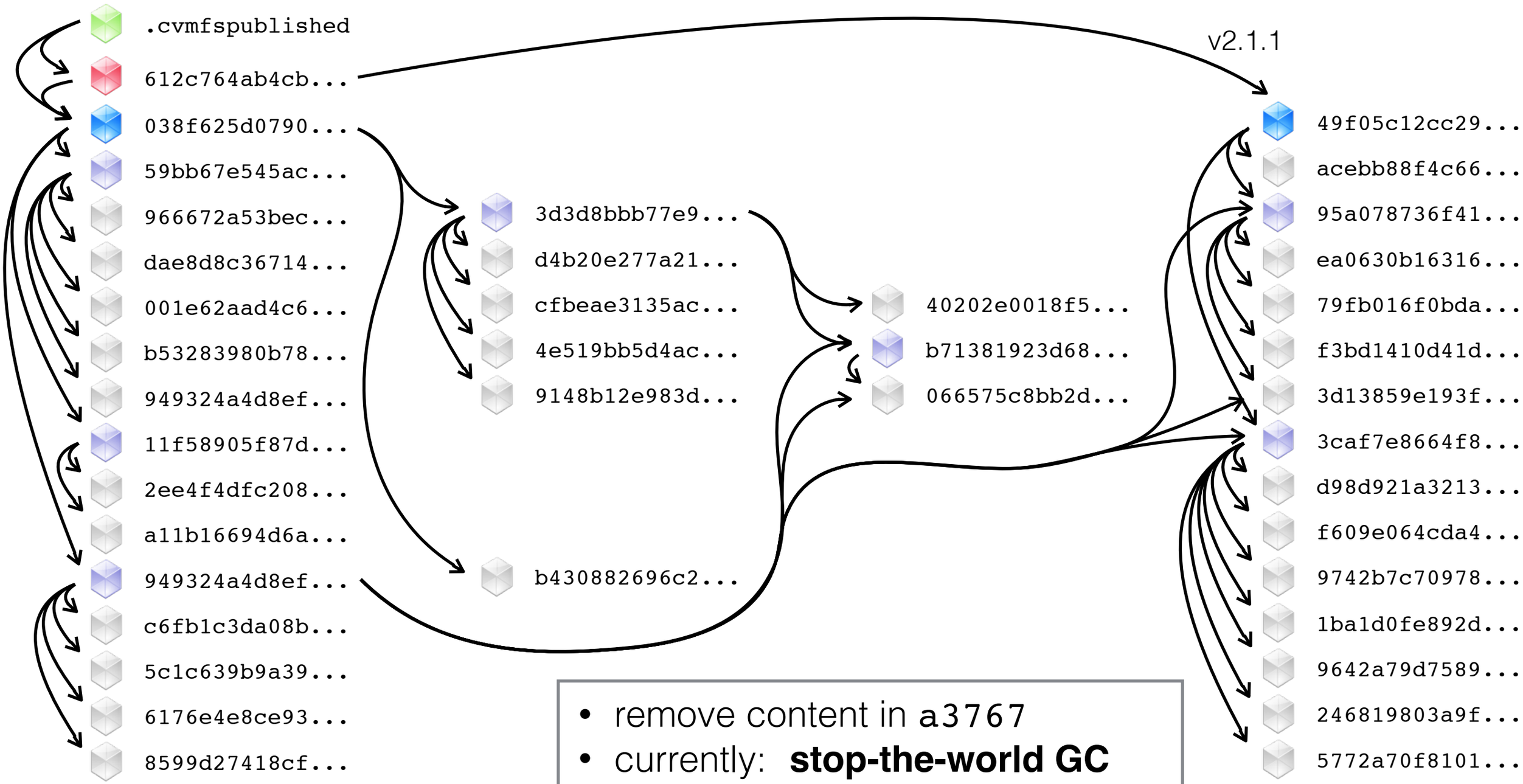
- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

Garbage Collection

- remove content in `a3767`
- currently: stop-the-world GC
  no concurrent publish

(schematic -not an actual repository)

Garbage Collection



(schematic -not an actual repository)

- remove content in `a3767`
- currently: **stop-the-world GC**
  no concurrent publish

Garbage Collection

- Mark-and-Sweep implementation
- Two-stage approach:
  - Traverse preserved catalogs and log referenced objects
  - Traverse condemned catalogs and match against log
- Full walk of the repository's catalog graph required
- Time consuming task

Smart Stratum1 Servers

Automatic Stratum1 Ordering,
Push Replication to Stratum1 Servers

Smart Stratum1 Servers

- Equip Stratum1 servers with RESTful API
- Automatic Stratum1 ordering on the client side
  - Based on GeoIP database to determine closest replicas
  - (to come in CernVM-FS 2.1.20 - Dave Dykstra - Fermilab/OSG)
- Triggered (instant) replication of new revisions
  - Repository's private key for authentication

Wrap up

Main New Features

- **Alternative Storage Backends** based on key-value stores through the S3 API (Seppo Heikkila)
- **Named Snapshots and History** for long term software accessibility and error recovery
- **Garbage Collection** for rapidly changing repositories (expected in CernVM-FS 2.1.20)
- **Transactional Repository Updates** for better publishing performance and a robust backend

Other New Features

- **Automatic Ordering of Stratum1 Mirrors** based on geo-IP location (Dave Dykstra - OSG)
- **Configuration Bootstrap Repositories** to facilitate public key and configuration distribution (not yet released)
- **Fully Parallel File Processing** to speed up snapshot publishing
- **Chunking of Large Files** for better cache exploitation and traffic efficiency

CERN

Backup Slides

What is CernVM-FS?

What is CernVM-FS?

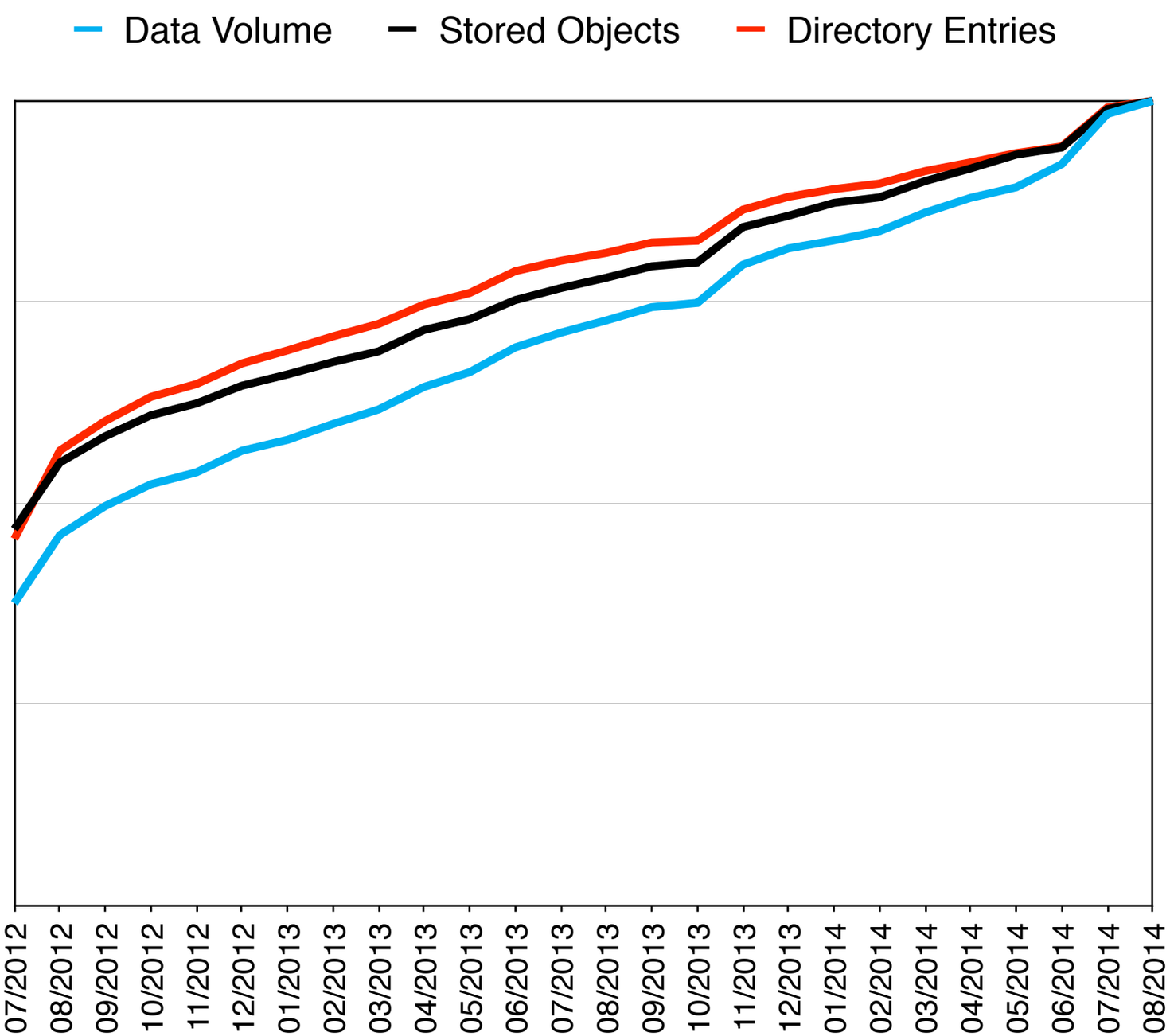
Problem:
Scalable, Fast and Reliable Global Software Distribution

What is CernVM-FS?

Software

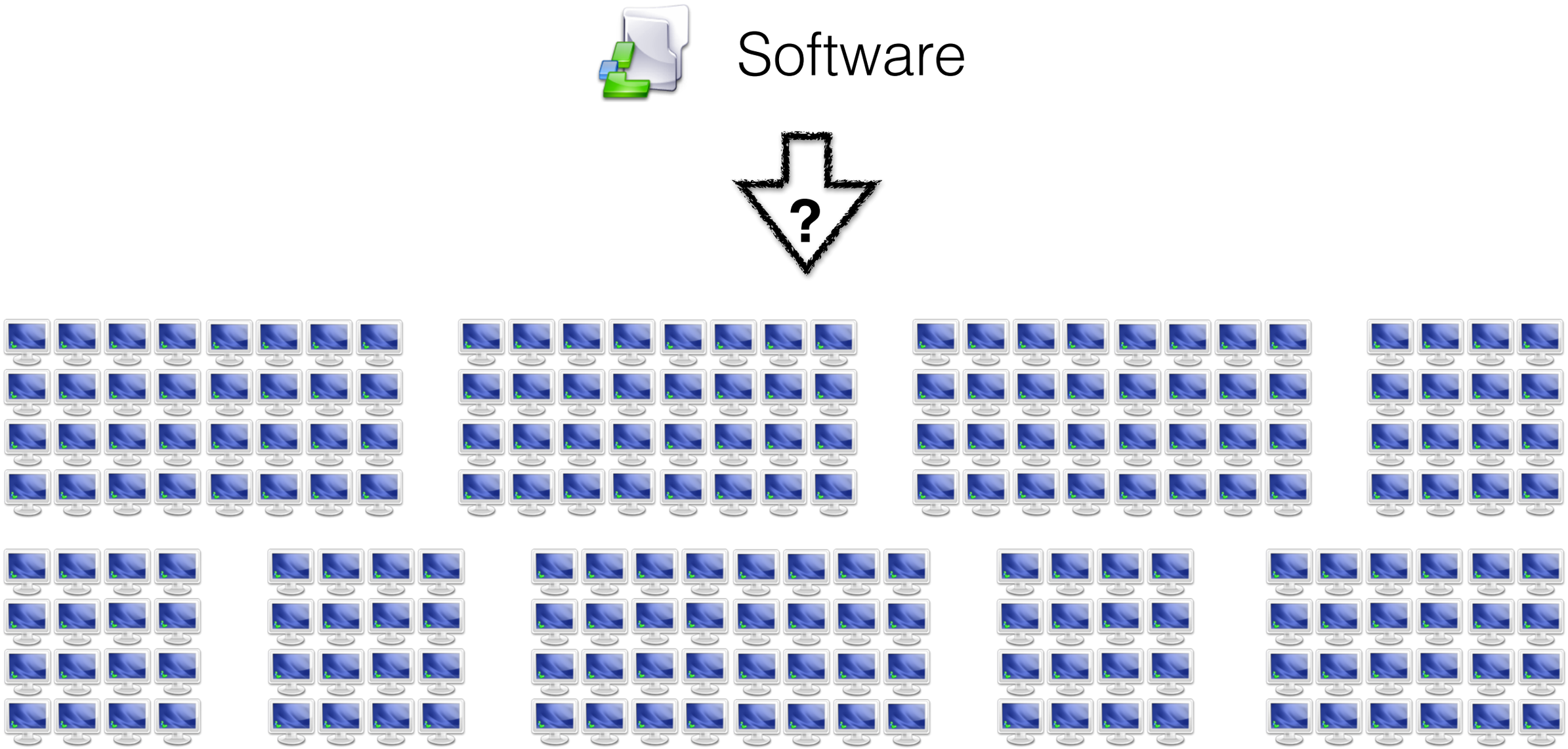
Problem:
Scalable, Fast and Reliable Global Software Distribution

Repository Growth



- Example Repository: **atlas.cern.ch**
- Size approximately doubled in two years
- Maximal values:
  - Data: 2.1 TiB
    Entries: 48.0 M
    Objects: ~3.8 M

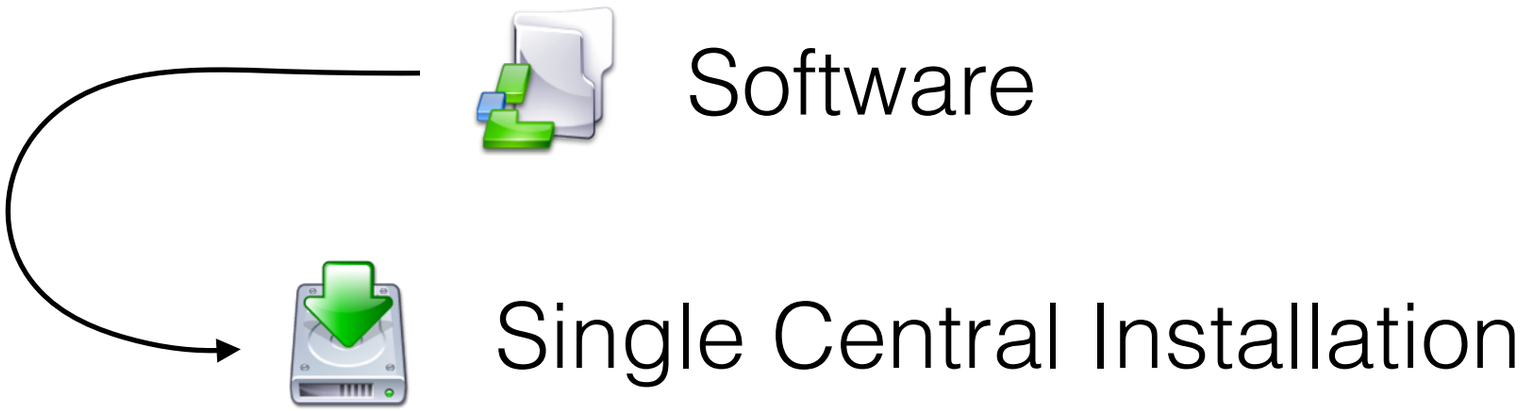
What is CernVM-FS?



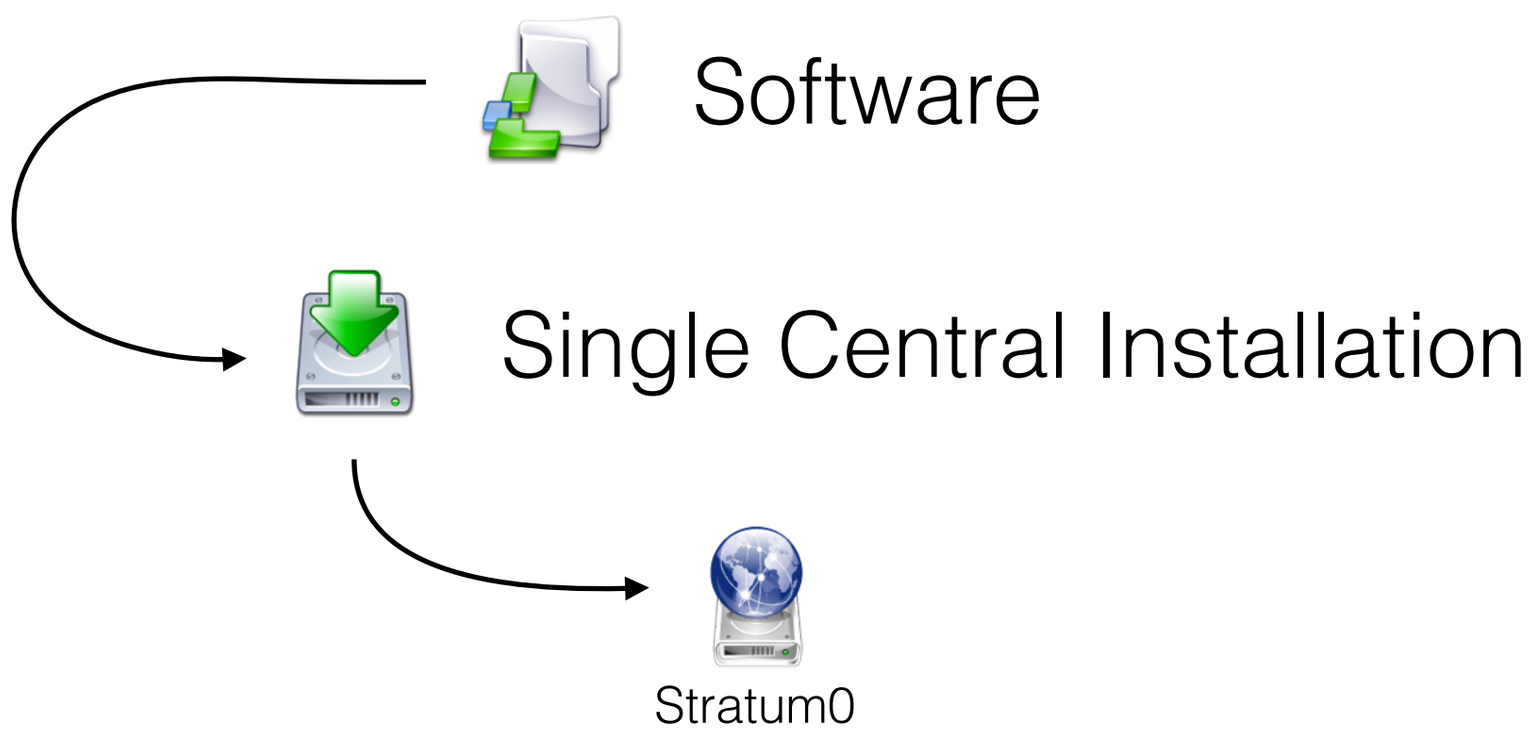
What is CernVM-FS?

Software

What is CernVM-FS?



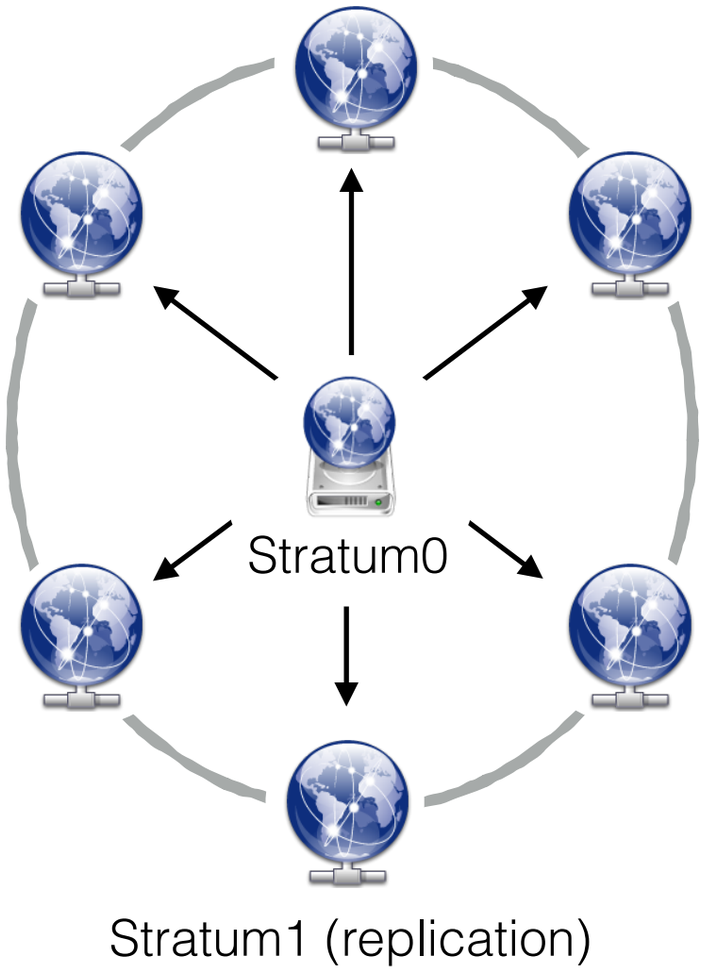
What is CernVM-FS?



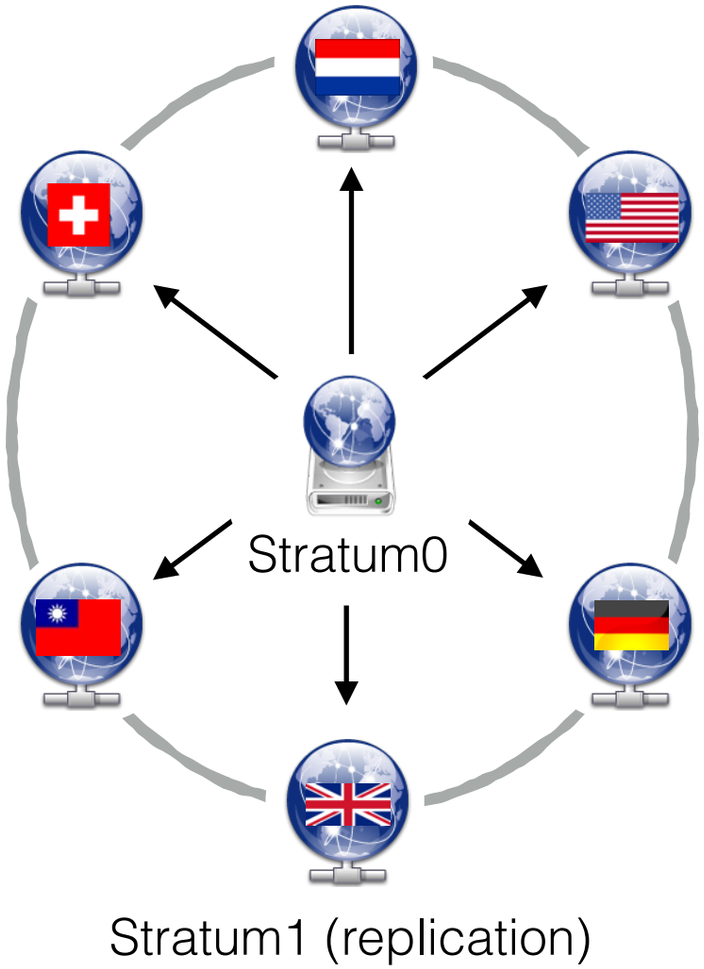
What is CernVM-FS?

Stratum0

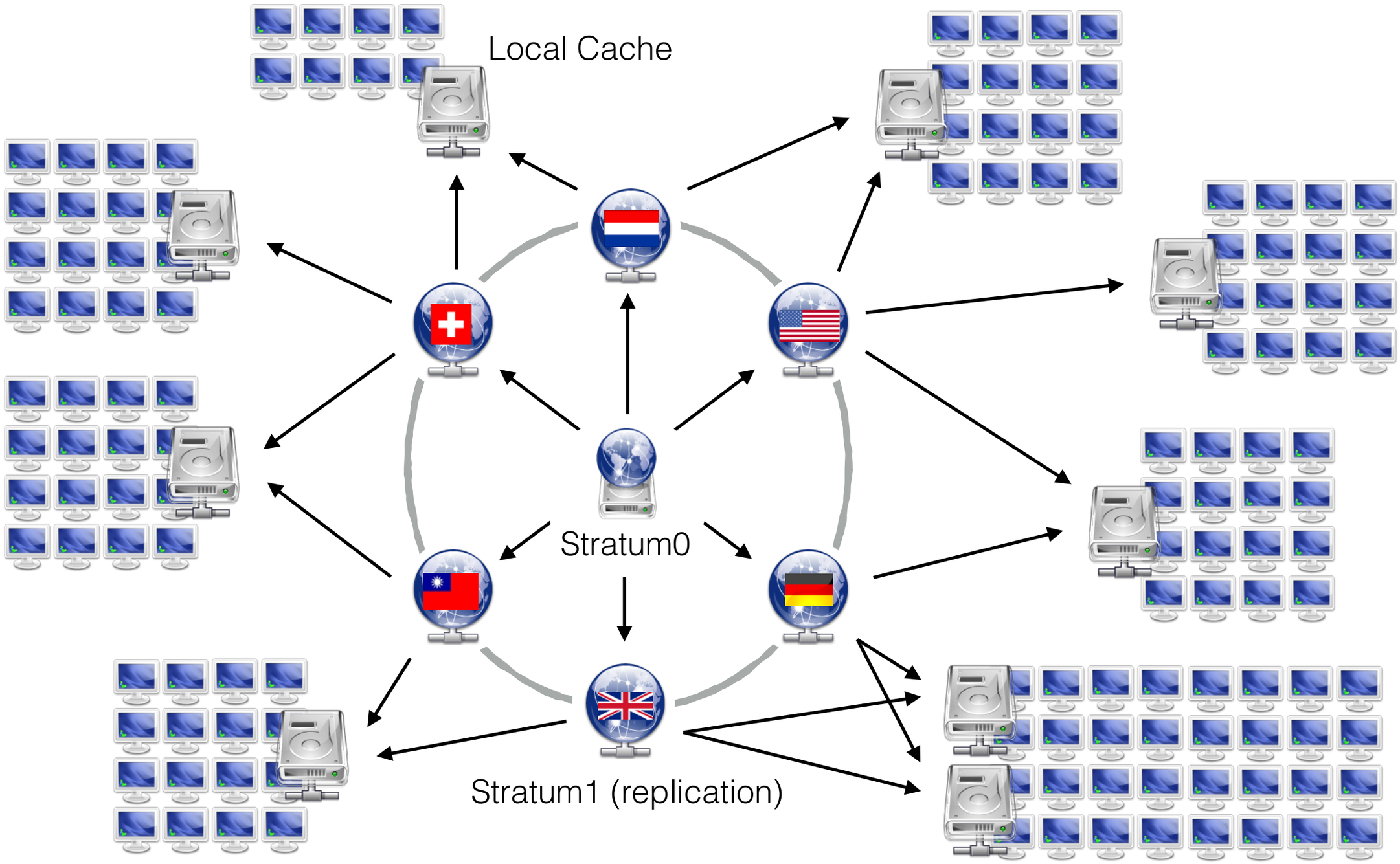
What is CernVM-FS?



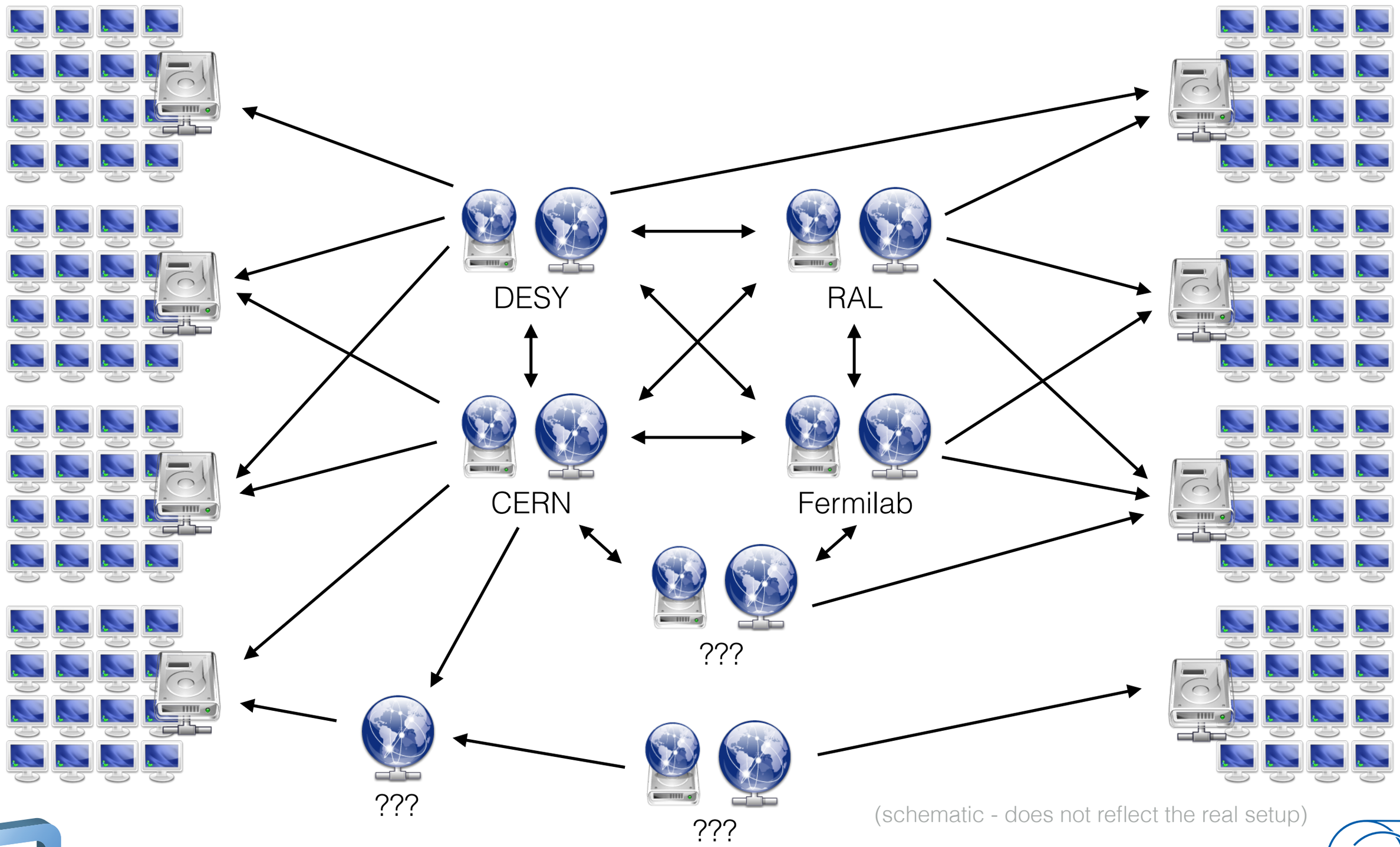
What is CernVM-FS?



What is CernVM-FS?



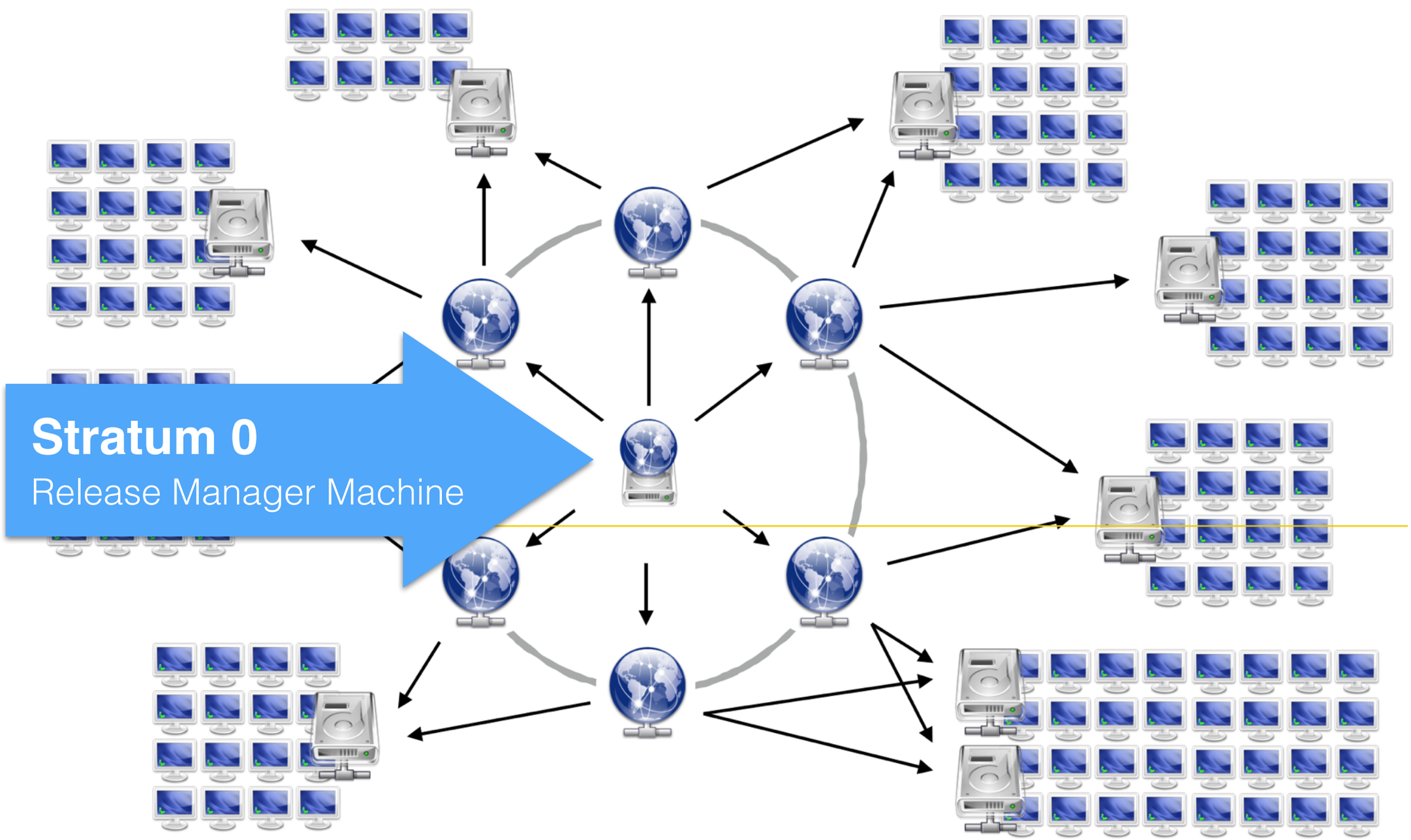
What is CernVM-FS?



(schematic - does not reflect the real setup)

CernVM-FS 2.1 Server

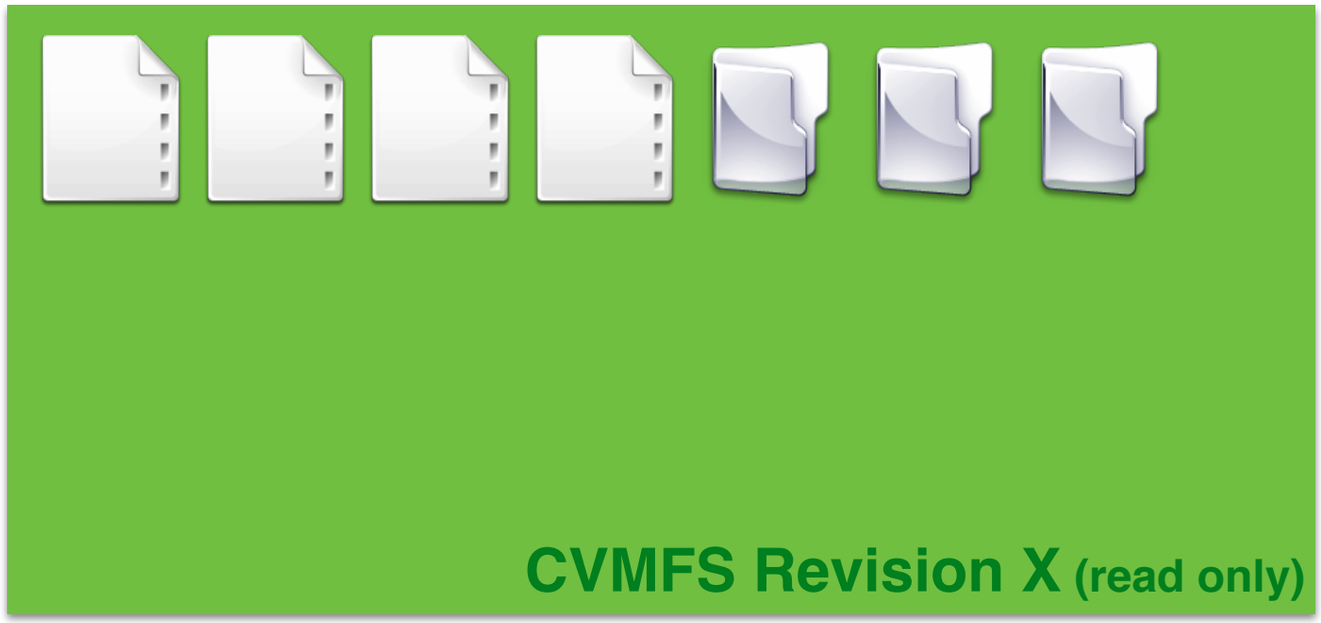
Transactional Repository Updates using a Union File System


Stratum 0
Release Manager Machine

Updating a Repository

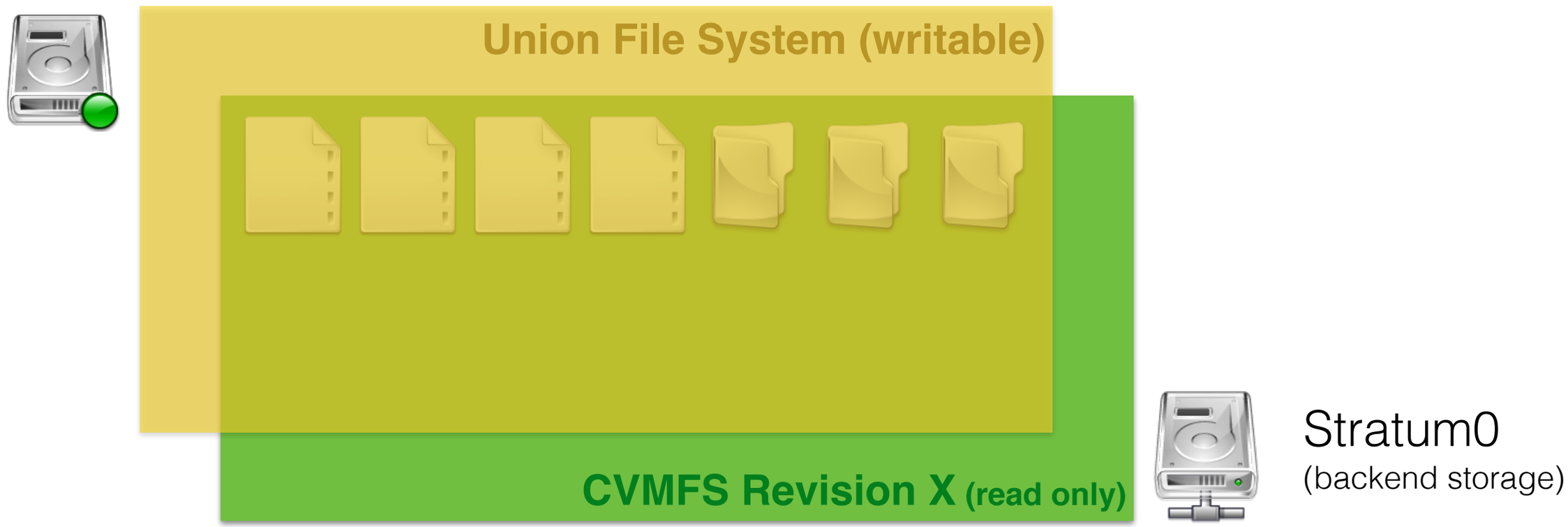
Stratum0
(backend storage)

Updating a Repository

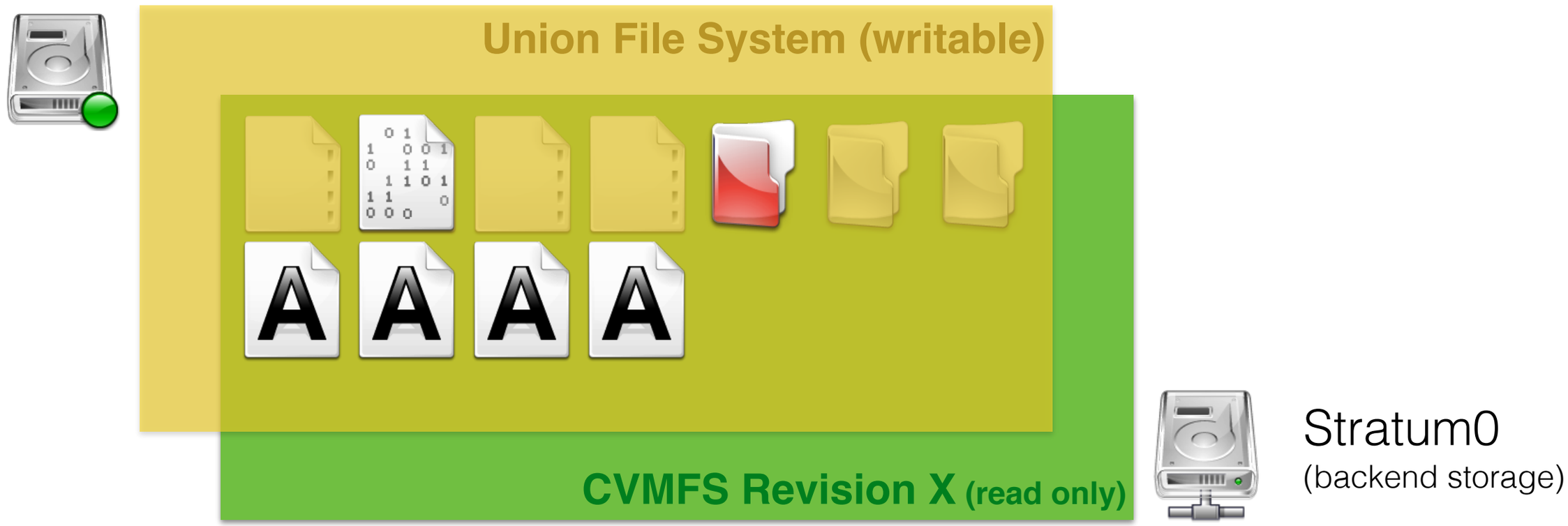


Stratum0
(backend storage)

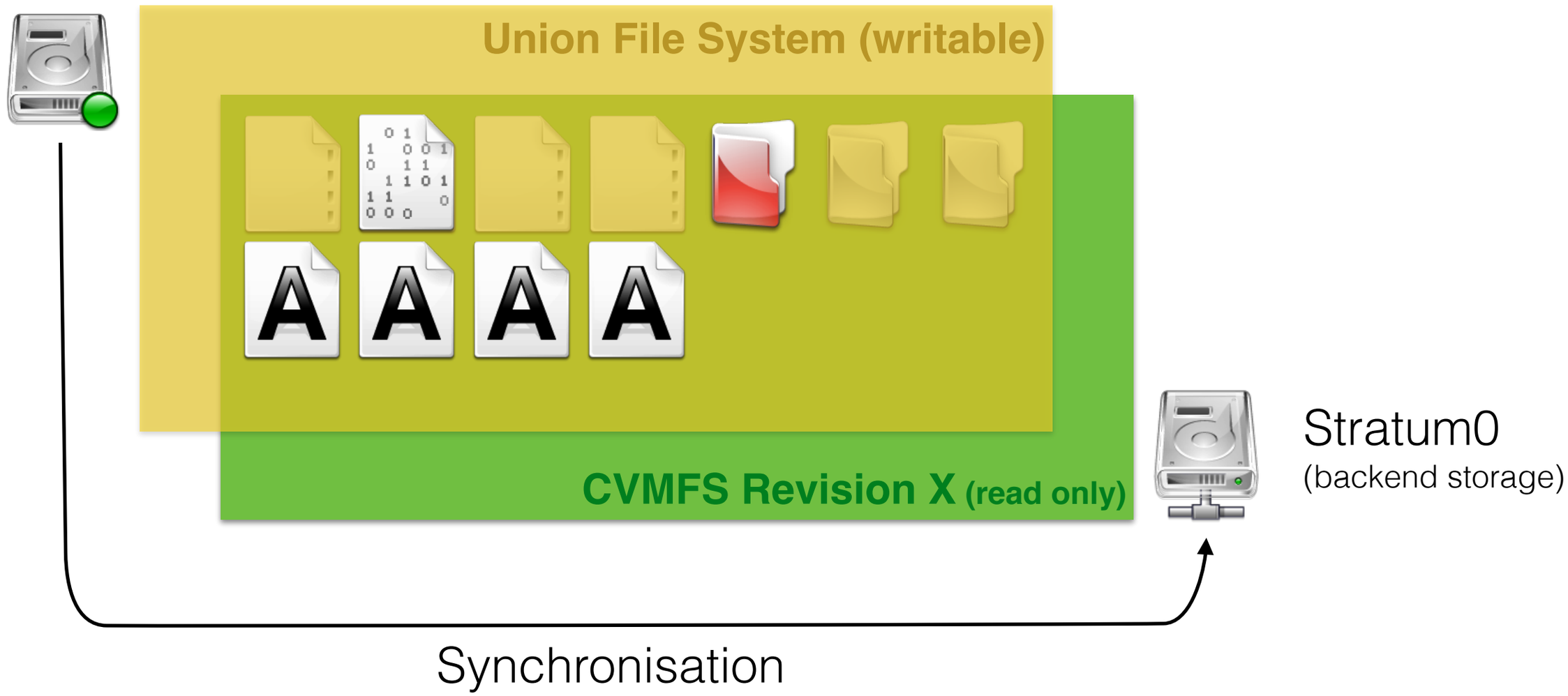
Updating a Repository



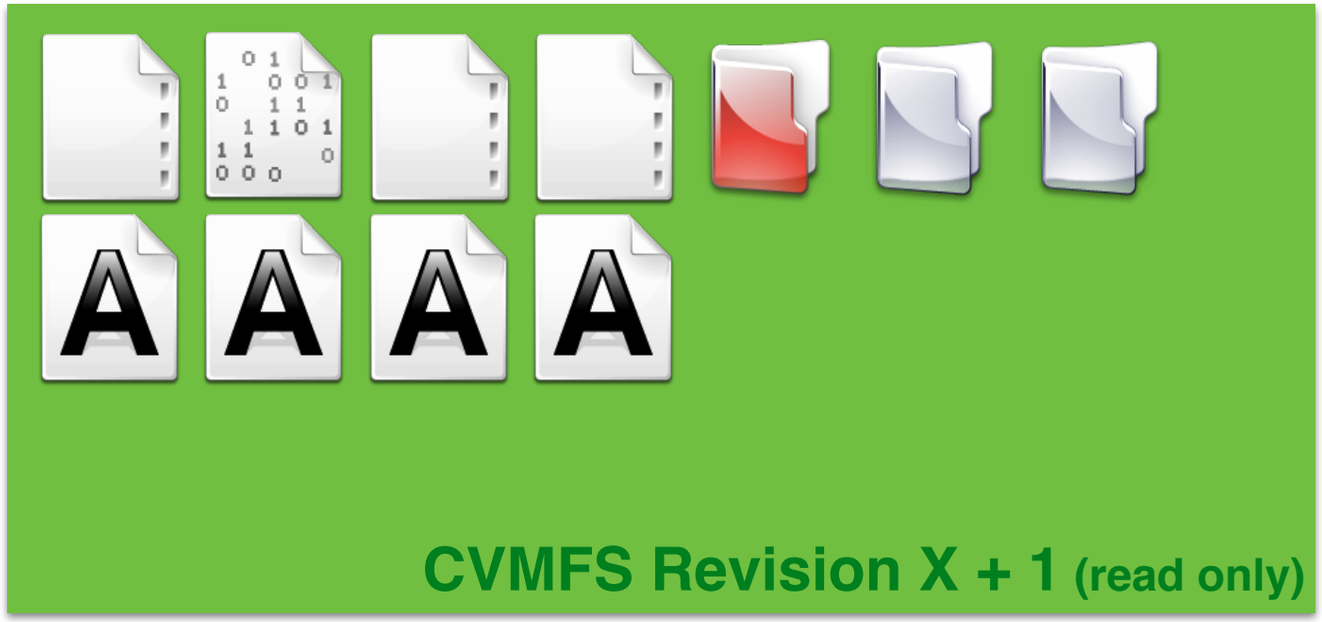
Updating a Repository



Updating a Repository

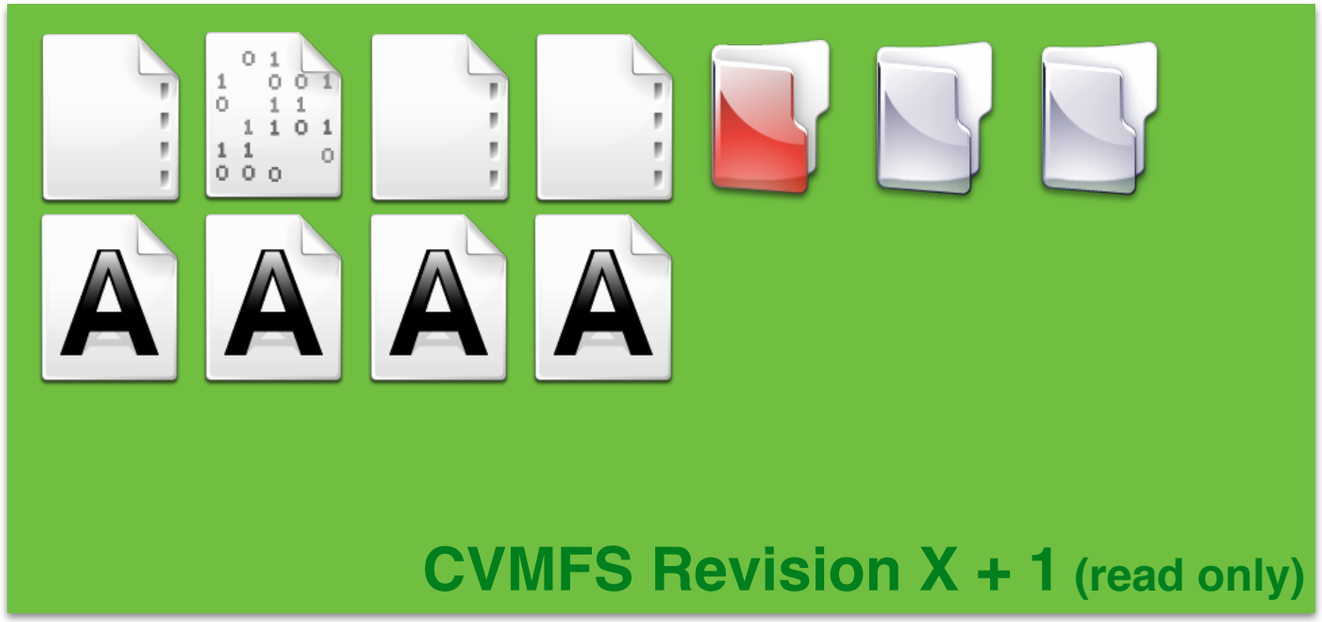


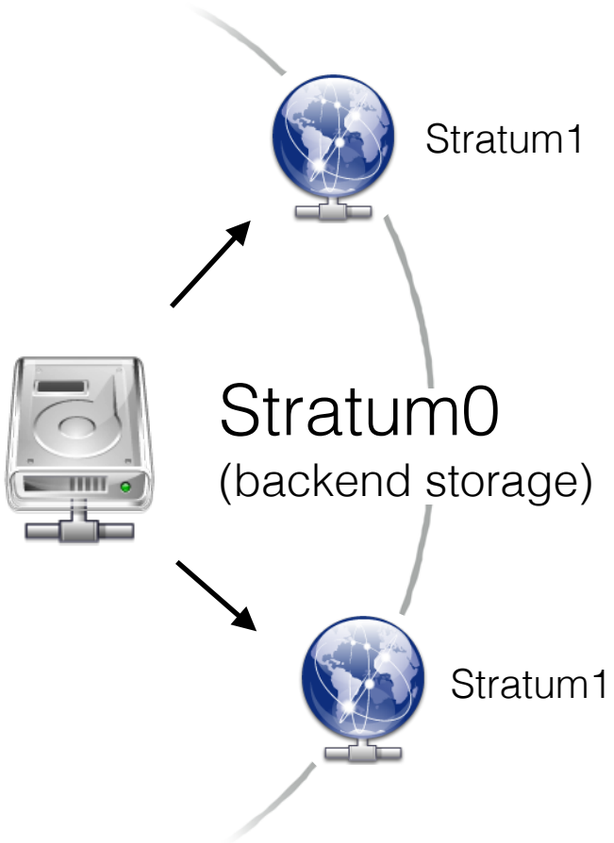
Updating a Repository



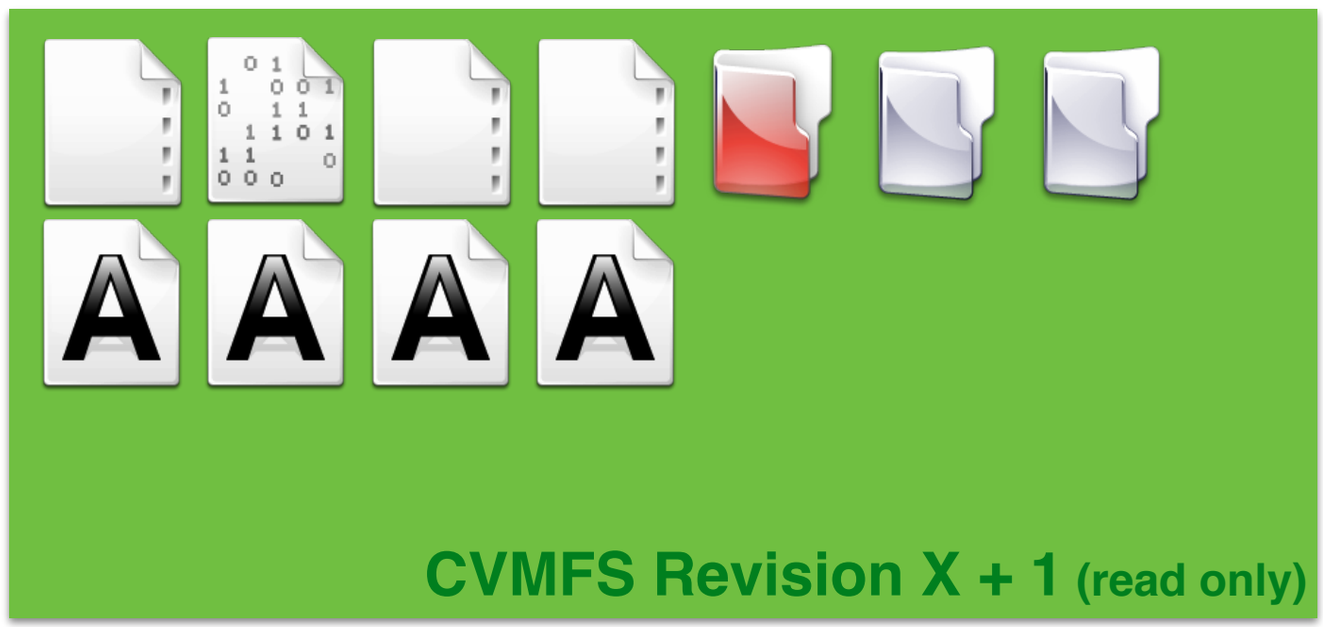
Stratum0
(backend storage)

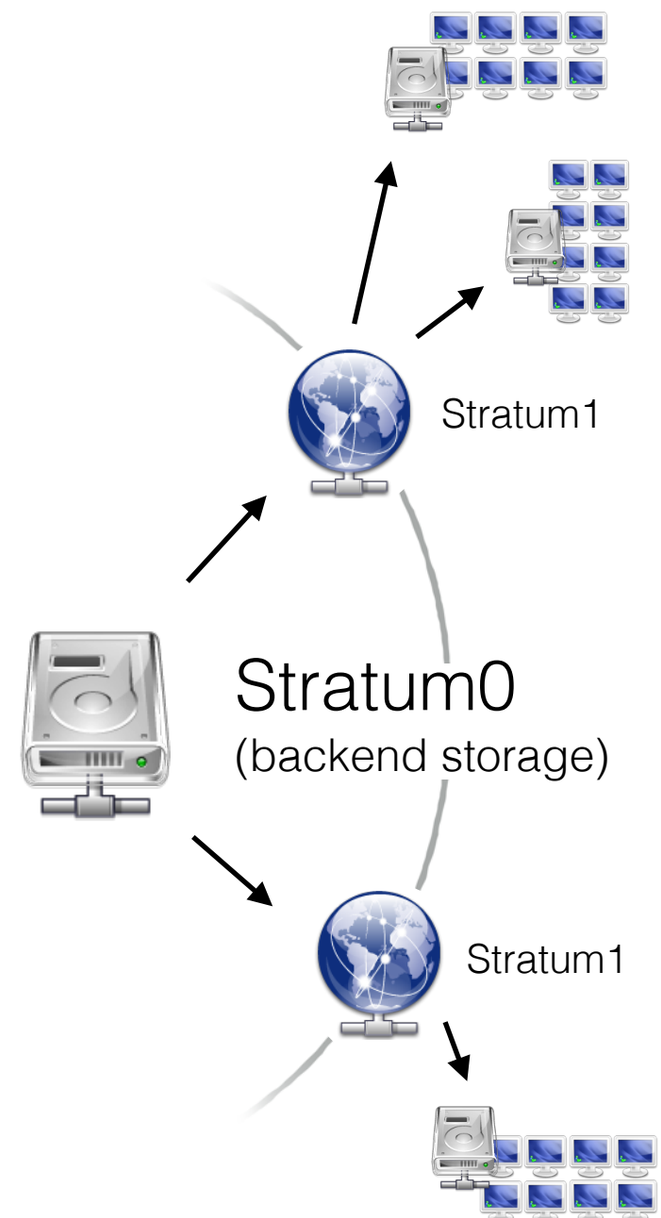
Updating a Repository



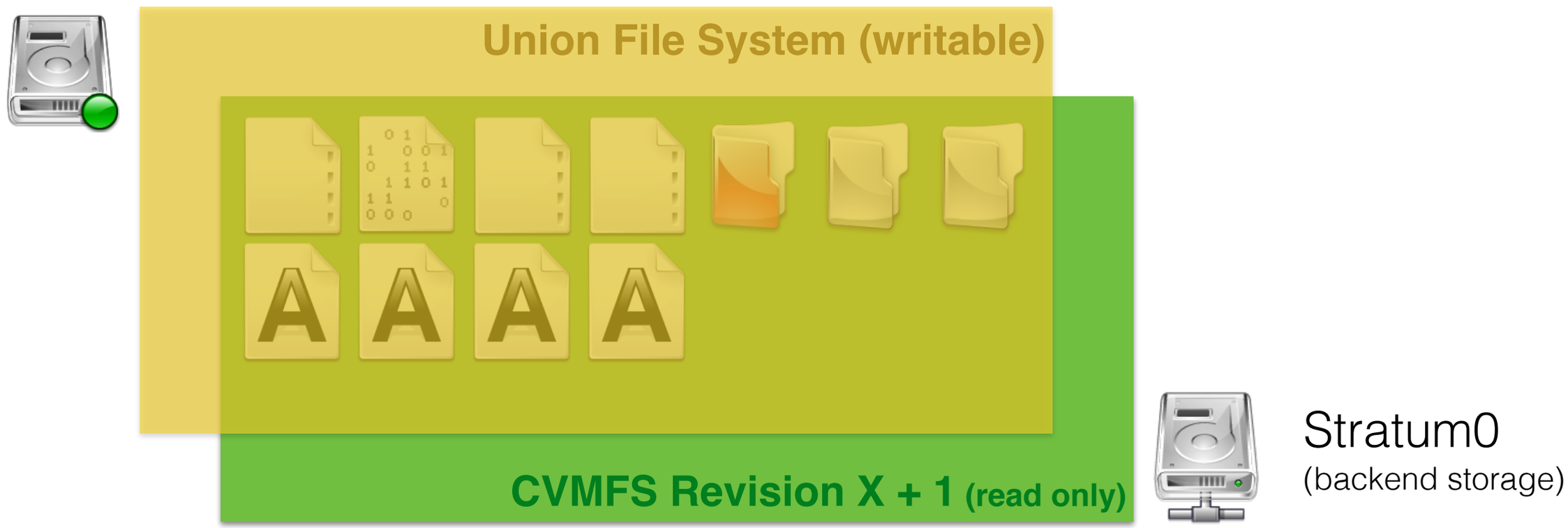


Updating a Repository

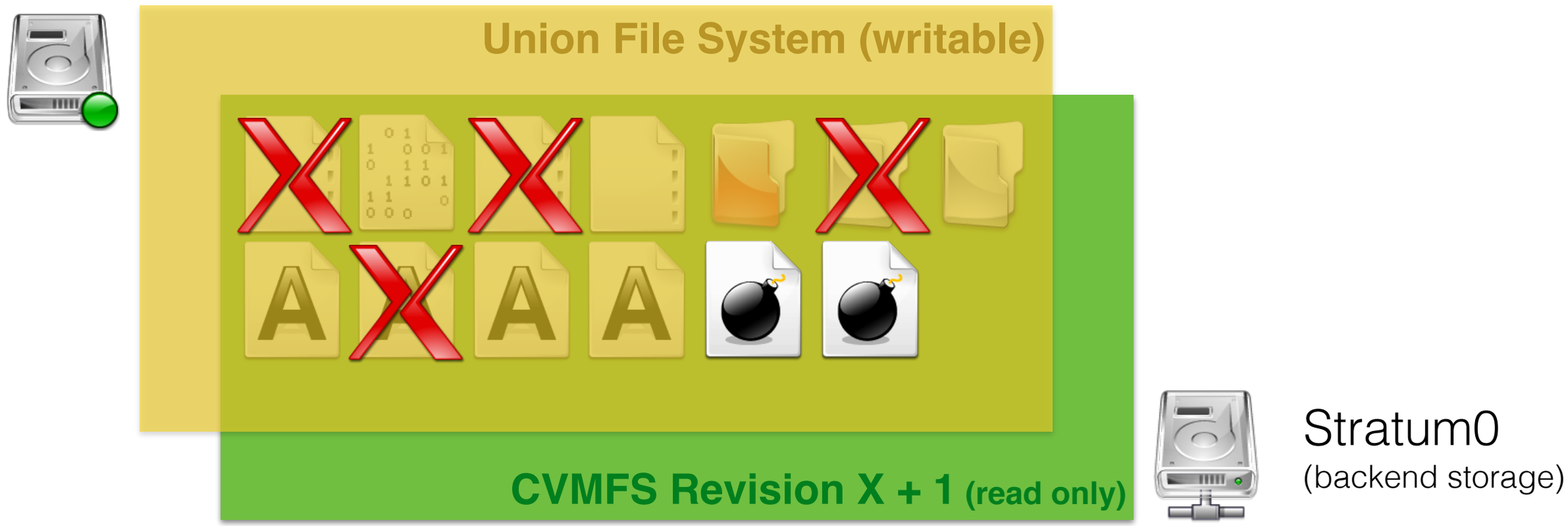




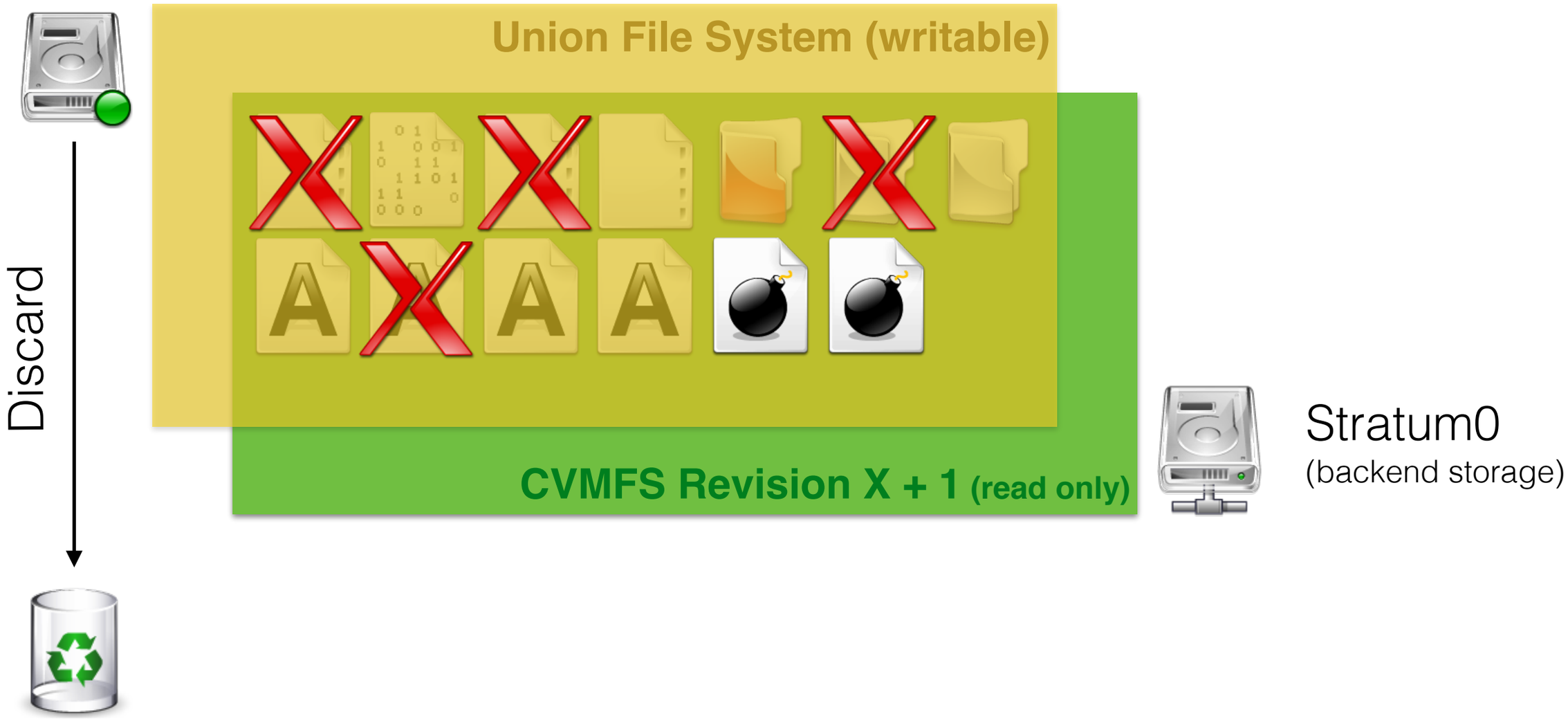
Updating a Repository



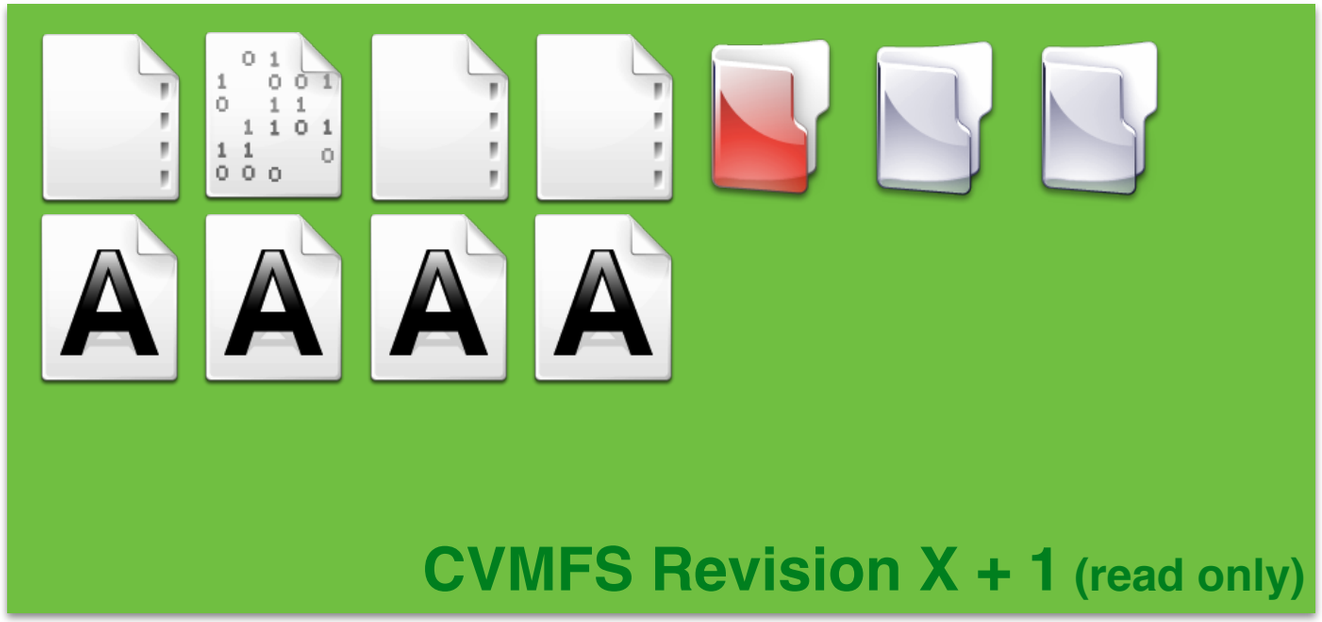
Updating a Repository



Updating a Repository



Updating a Repository



Stratum0
(backend storage)

What is CernVM-FS?

Abstract

The CernVM-File System (CVMFS) is a snapshotting read-only file system designed to deliver software to grid worker nodes over HTTP in a fast, scalable and reliable way. In recent years it became the de-facto standard method to distribute HEP experiment software in the WLCG and starts to be adopted by other grid computing communities outside HEP.

This paper focusses on the recent developments of the CVMFS Server, the central publishing point of new file system snapshots. Using a union file system, the CVMFS Server allows for direct manipulation of a (normally read-only) CVMFS volume with copy-on-write semantics. Eventually the collected changeset is transformed into a new CVMFS snapshot, constituating a transactional feedback loop.

The generated repository data is pushed into a content addressable storage requiring only a RESTful interface and gets distributed through a hierarchy of caches to individual grid worker nodes. Besides practically all POSIX-compliant file systems, lately CVMFS allows to use highly-scalable key-value storage systems through the Amazon S3 API.

Additonally we describe recent features, such as file chunking, repository garbage collection, fast replication and file system history that enable CVMFS for a wider range of use cases.