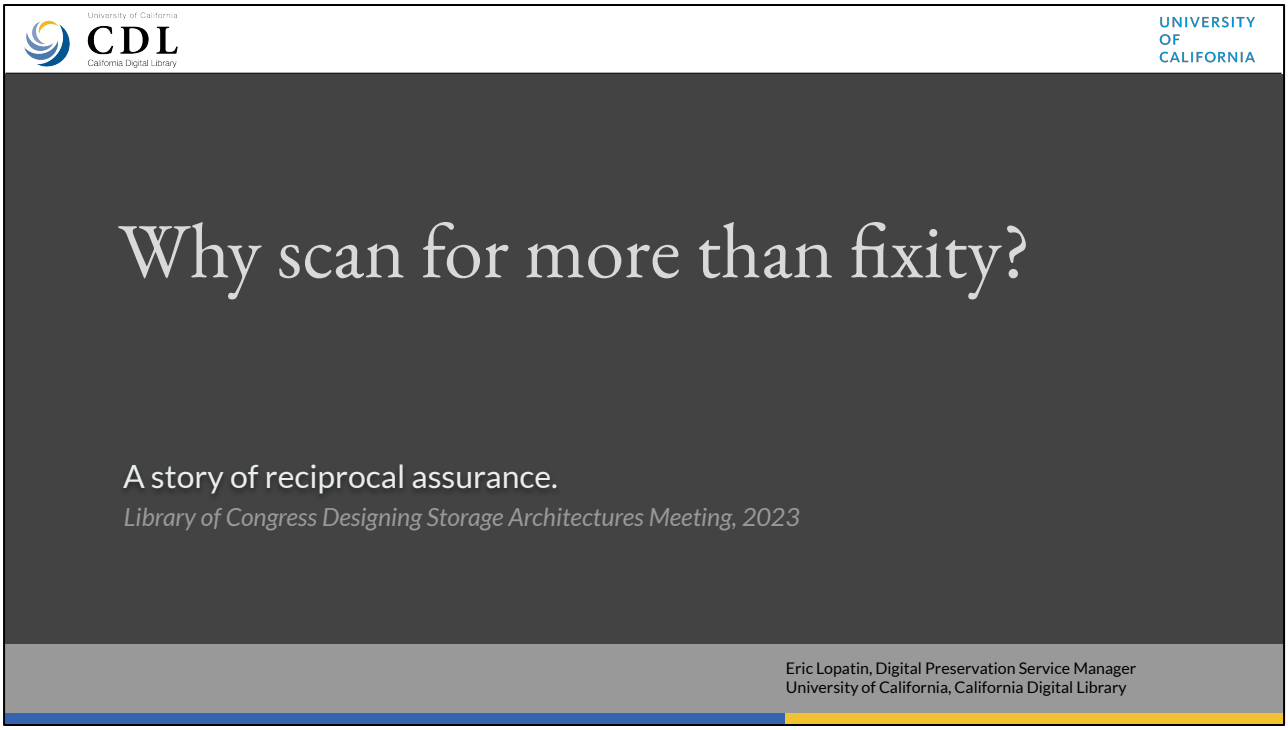
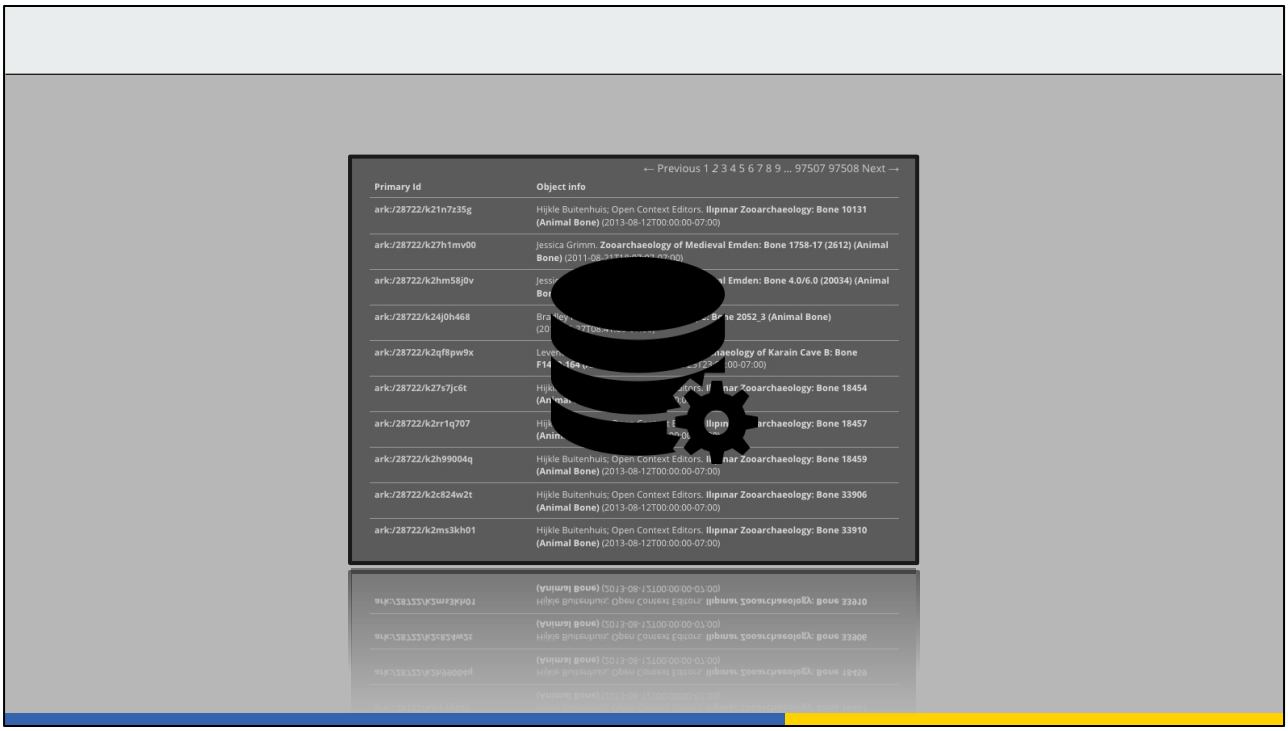
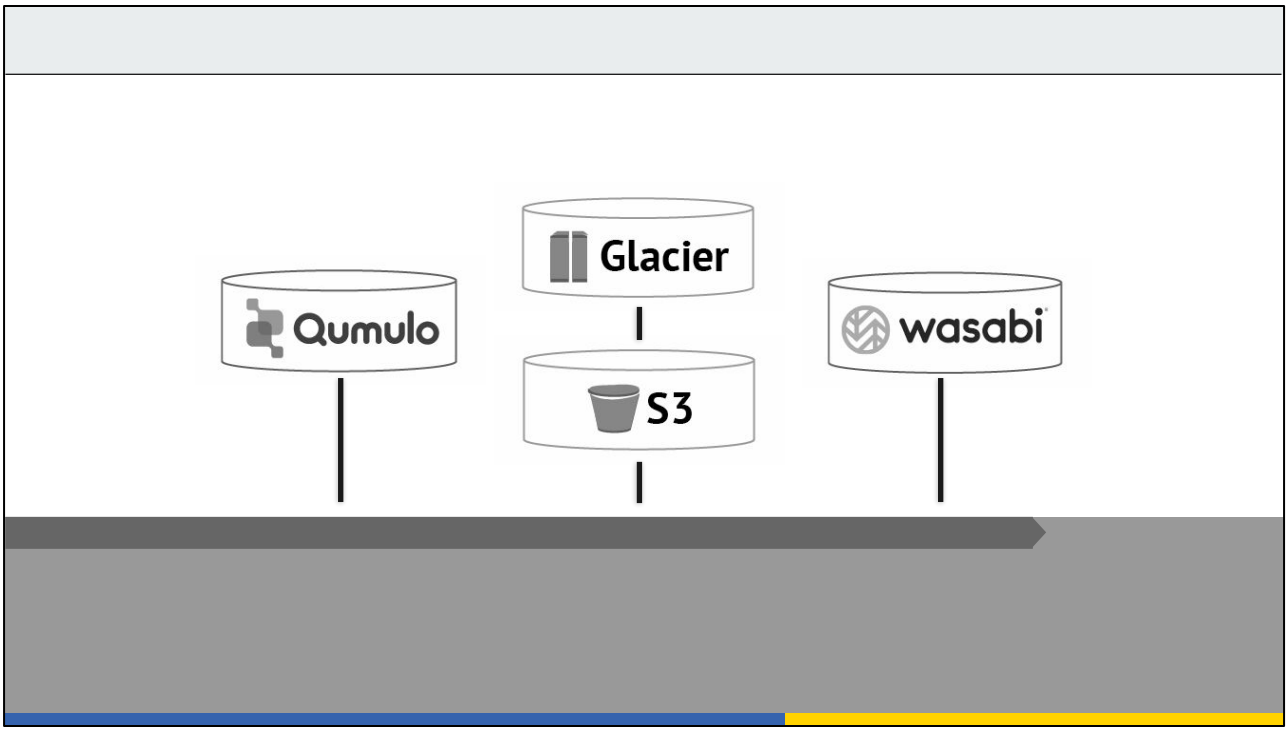


CDL is a centralized resource for the ten University of California campuses. More specifically, our service is part of the UC Curation Center where we facilitate projects that focus on digital preservation, persistent identifiers, research data management and data management plans.

My presentation today is titled, “Why scan for more than fixity? A story of reciprocal assurance.”

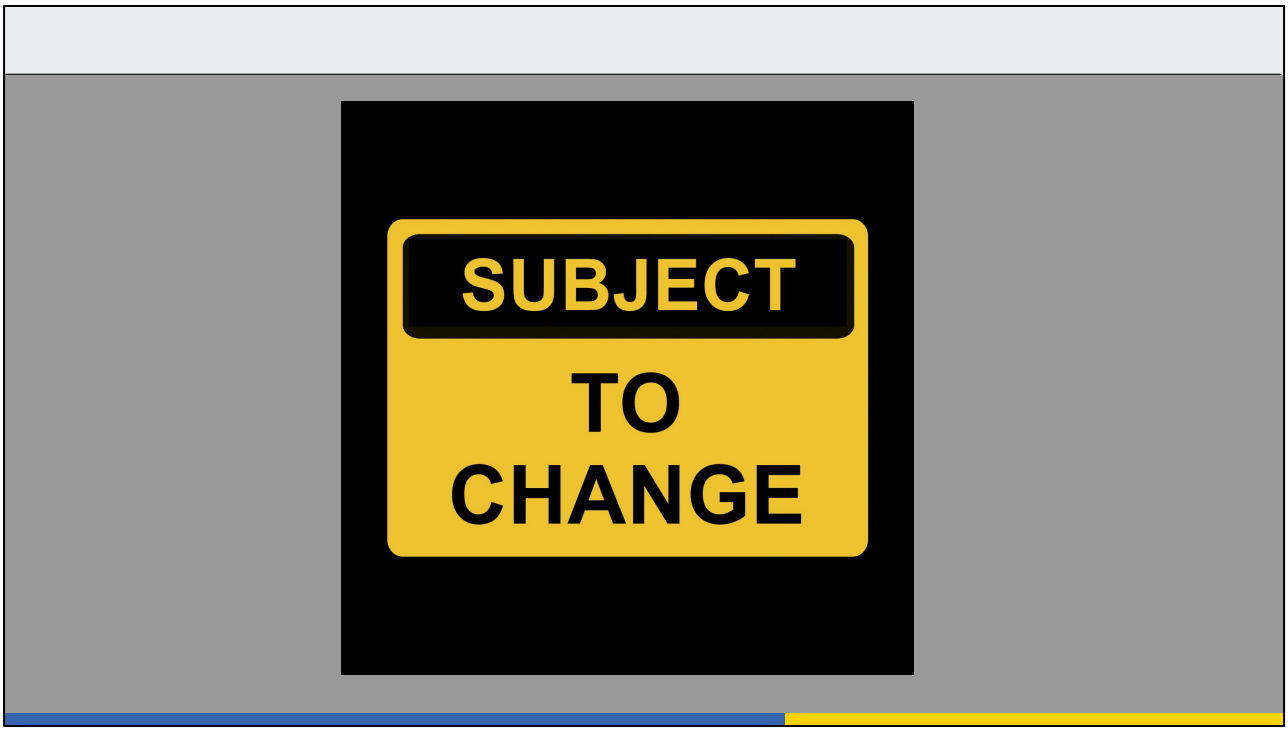


At a basic level, an integral part of any preservation system is its ability to track, via a core component or service, the existence of the containers and files that comprise the digital objects it is preserving. Our repository at the CDL, a.k.a. Merritt, uses this strategy, like so many others. But although the system records entries in its **core inventory database** as a final part of the ingest process, another, **less often considered** but **equally important characteristic of our preservation strategy is an ability to reconstitute a record in that database** based on extant cloud content; hence the reciprocal assurance.



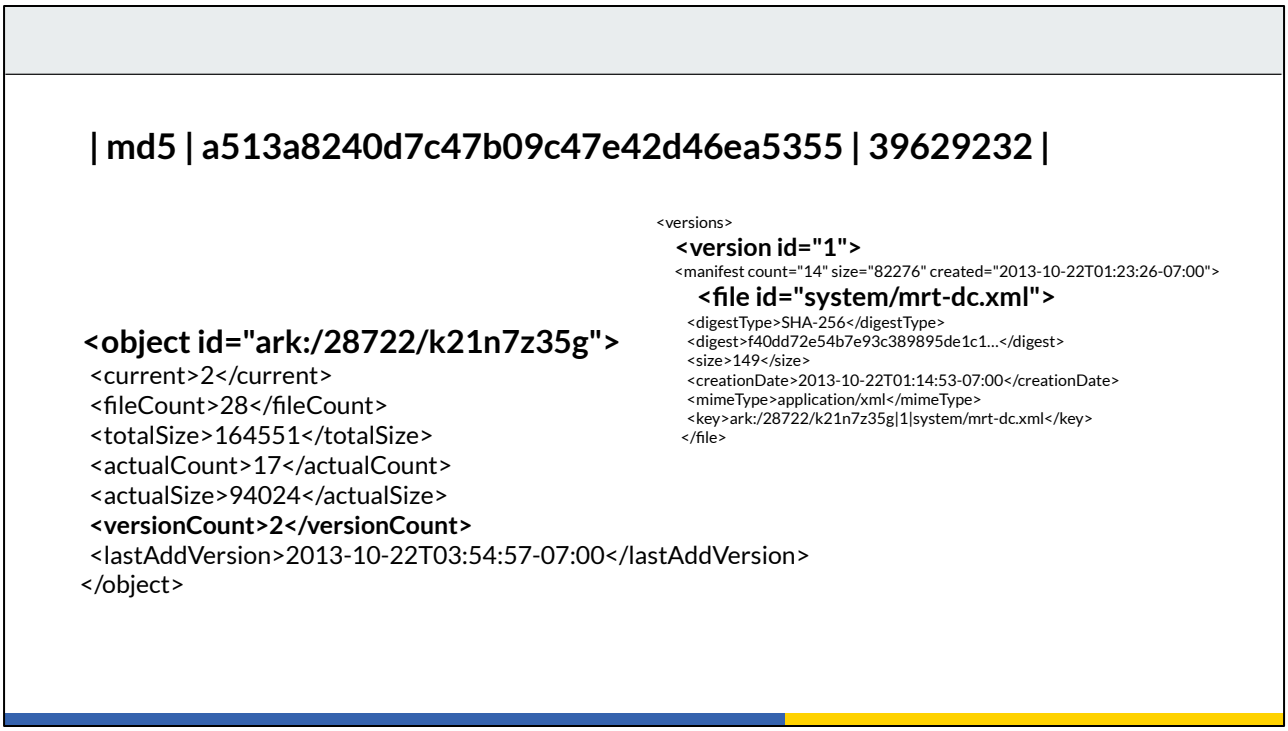
To this end, the team administering the system must be confident in its ability to correctly store and replicate objects across **multiple cloud storage nodes**. As streamlined as these processes are, nothing is perfect.

It's unreasonable to expect the repository to have been a perfectly performing system over the course of the past 10 years, or to be one in the years to come. As we all know, change is the only constant and it’s a big reason behind why we're here.



There are so many variables in our systems that entail change, from API updates, to an ever increasing number of cloud-based service offerings, to changes in information security policies, compute and storage costs, development roadmaps, let alone alone the institutional policies and practices that our preservation efforts are rooted in.

Image credit: https://archive.org/details/SubjectToChangeLogo_201608

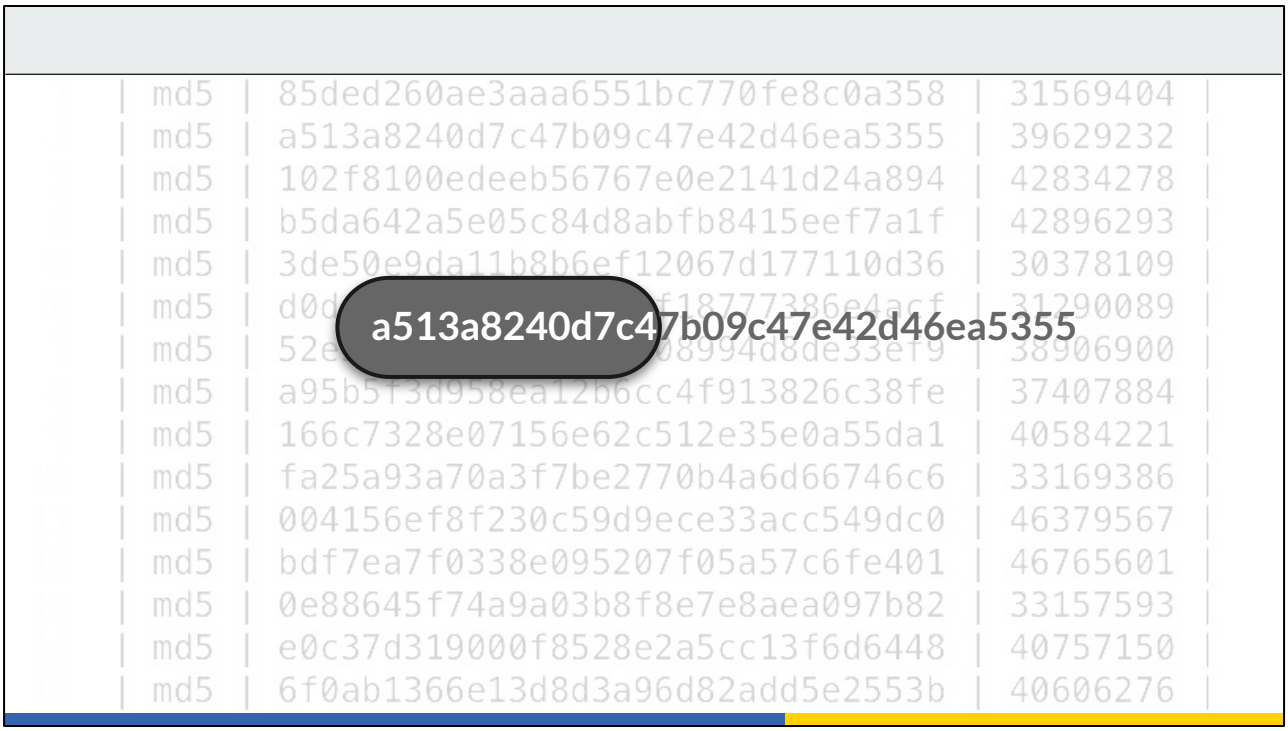


But let's focus on storage. In the face of change, we need a way to dependably scan for content on every type of cloud storage our system uses to ensure what we think is in the cloud, according to our database, is actually there, and vice versa.

We must scan not only for fixity purposes, **but** for elements such as **persistent identifier verification, version verification, object manifest integrity and system file integrity** among others. In other words, what is the expected state of any file in the system, and in the cloud?

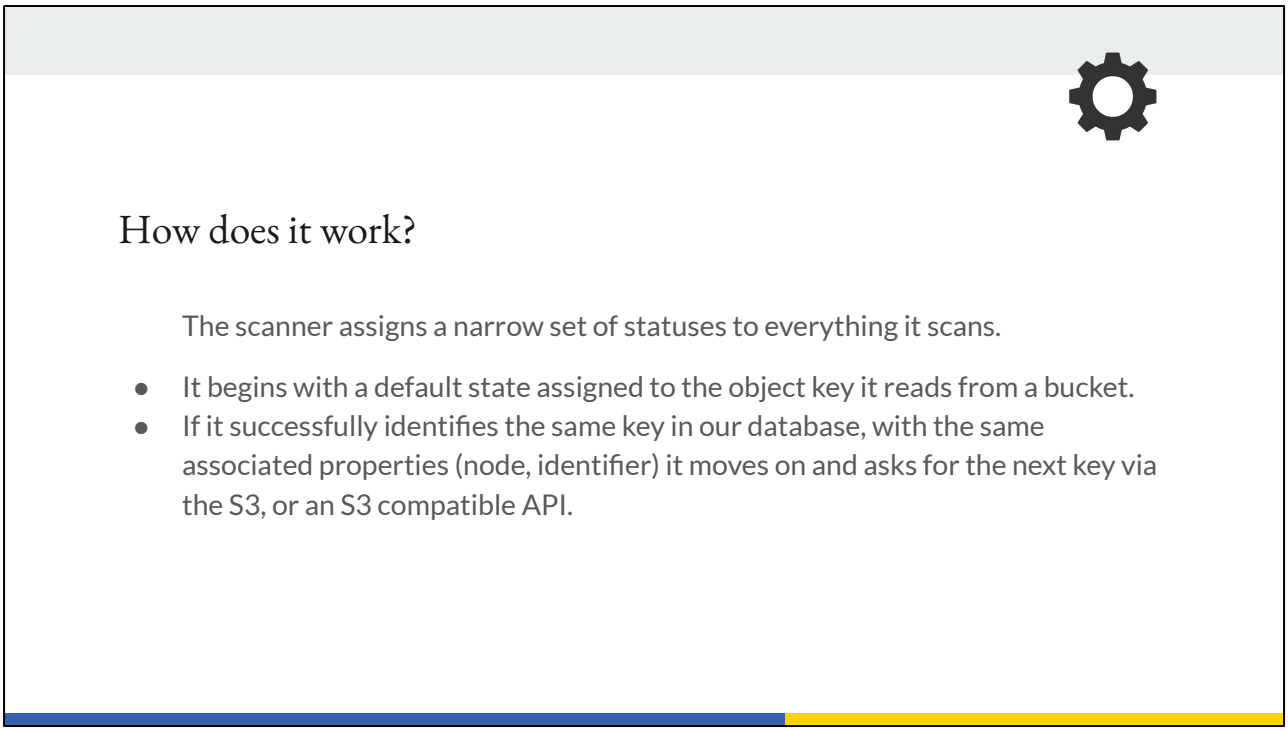
Only by knowing this can we be sure that, in the off chance our database undergoes **a catastrophic failure**, we'll be able to rebuild it and hold onto all the valuable information it houses. In a way, you could say we are interested in determining "collection health," but for much more than any one reason.

Image credit: Bear Photo Co., 1906, California State Library: https://calisphere.org/item/ark:/13030/hb7d5nb5vx/

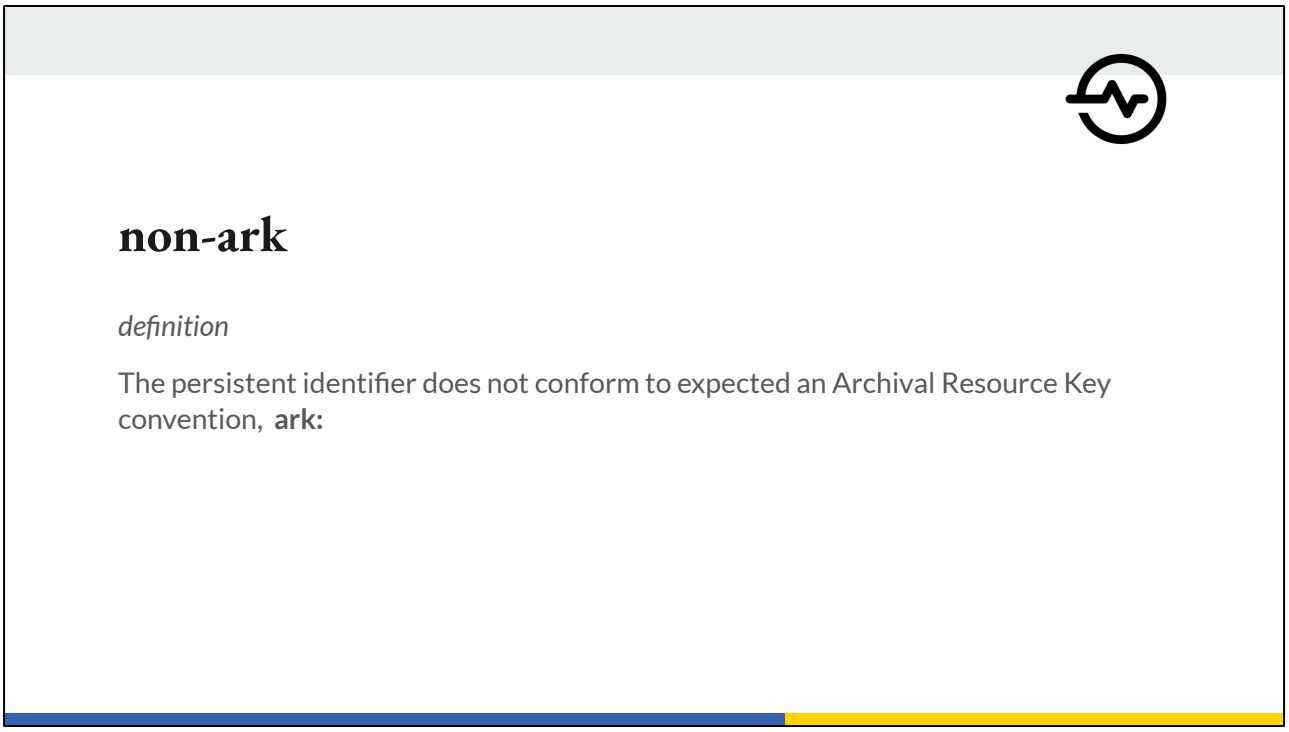


It's for these reasons in particular, for the health of the content, its stewarding system and as a form of curation that our team implemented **a cloud scan process**. And for better or worse, we discovered a significant number of inconsistencies that had built up over the years. Which is also why we would highly recommend this practice to our colleagues in the community.

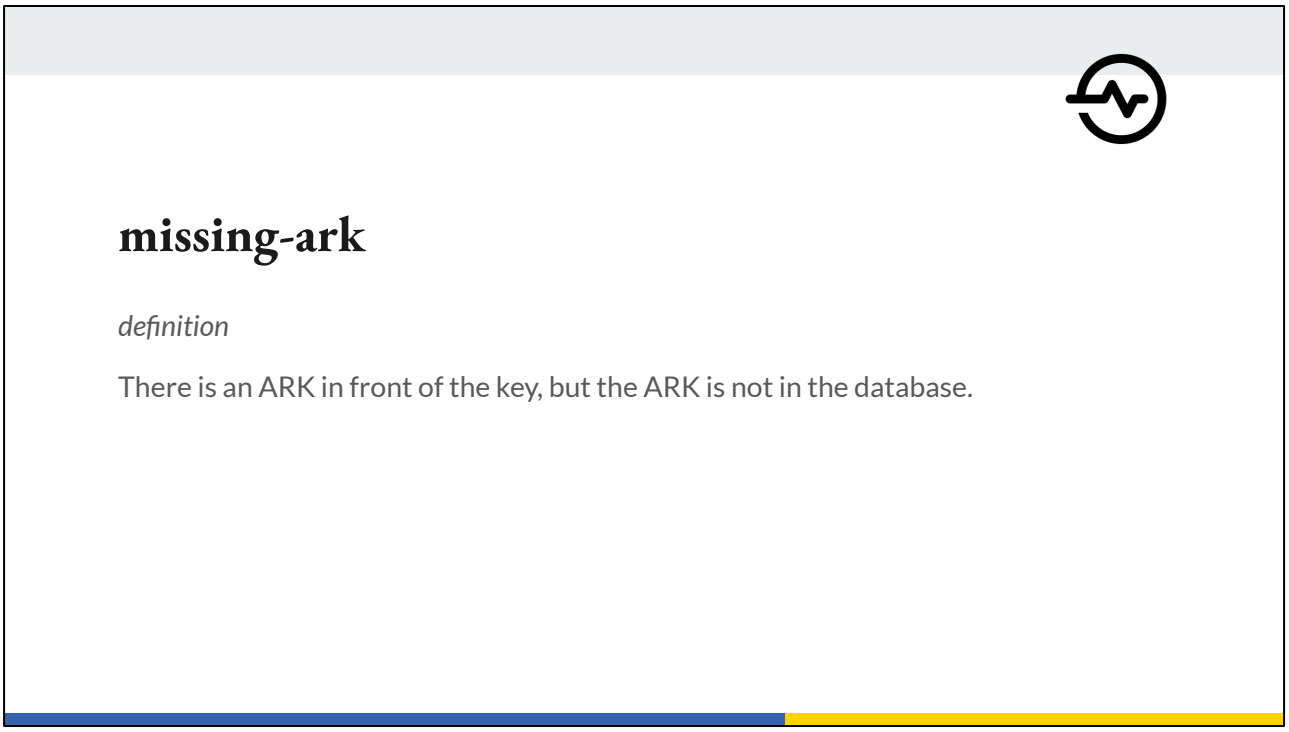
What were some of those inconsistencies? Let’s review how the scanner works and I’ll mention them as we go.



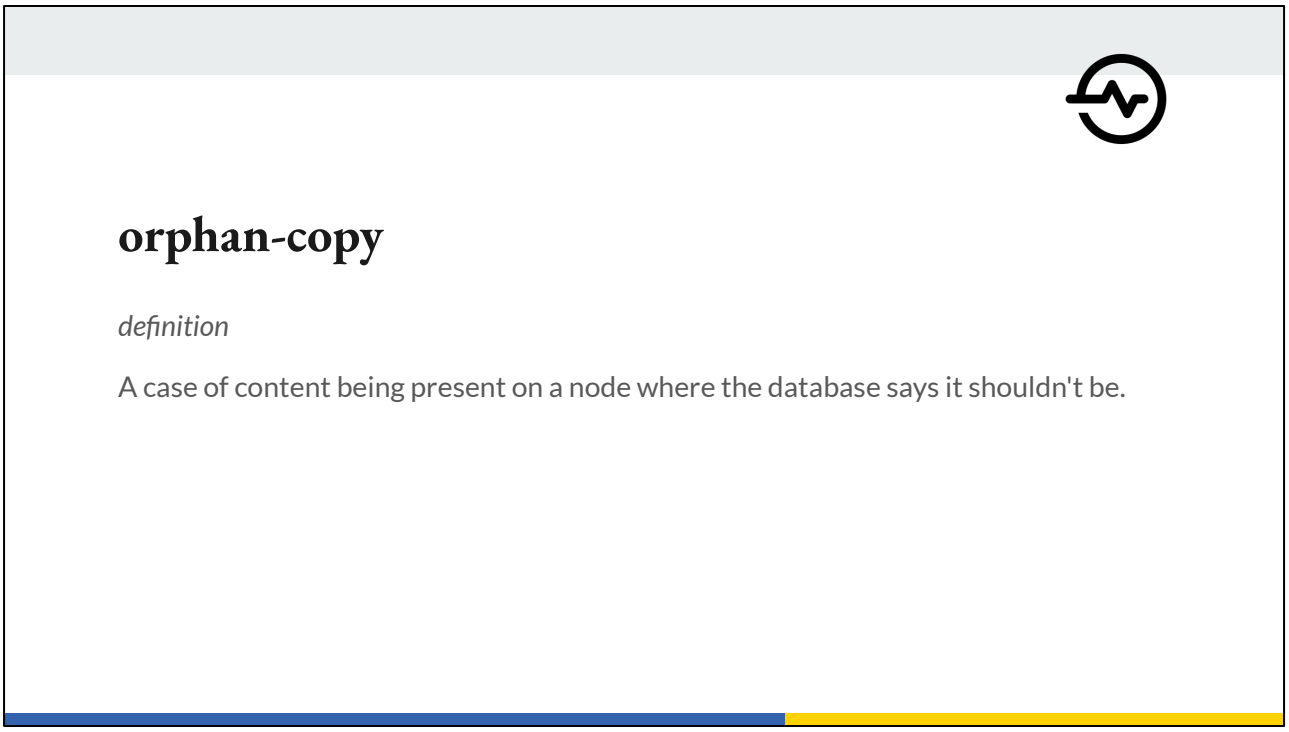
Here we experienced a limitation of the S3-compatible API layer that sits atop Qumulo storage: requesting the next key took hours in a bucket with millions of files. Not so with S3 and Wasabi, which responded almost immediately. For Qumulo, we must instead request a bucket inventory file to iterate over from the data center.



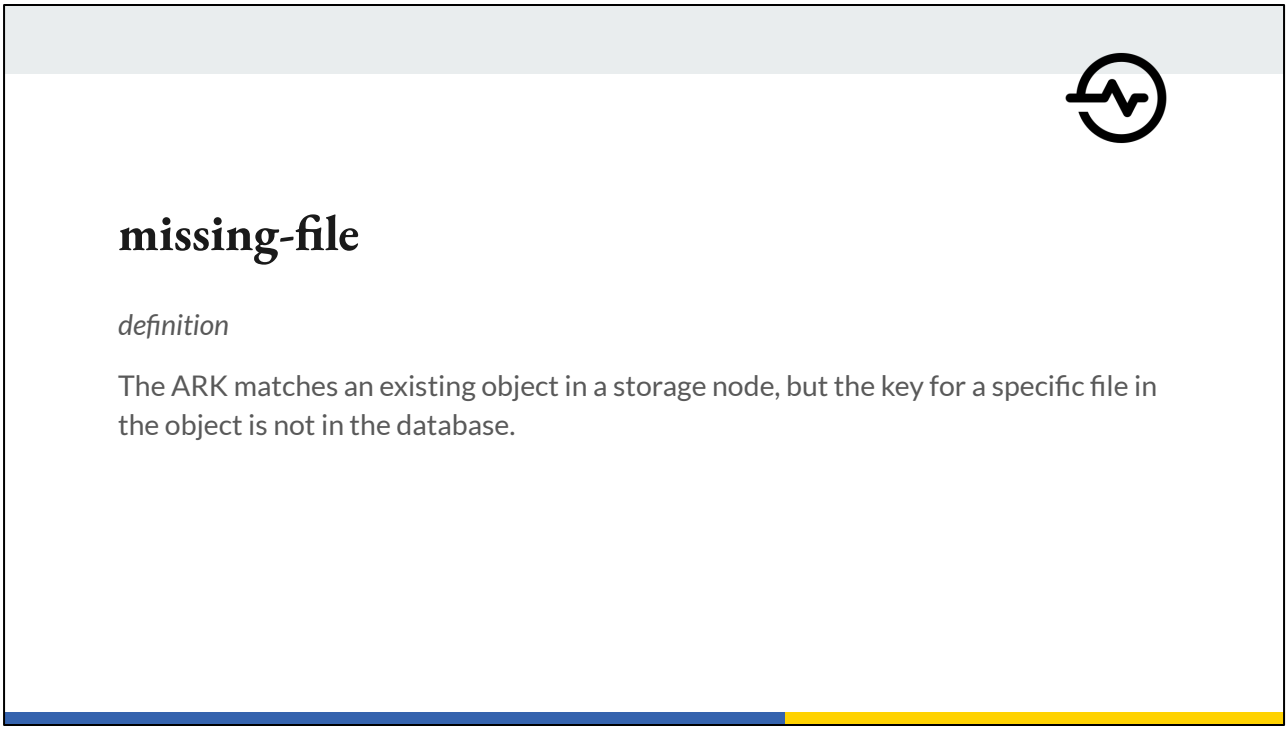
- **non-ark**: The S3 key does not begin with “ark:”
  - The persistent identifier for the key does not conform to an expected Archival Resource Key (ARK) convention (ark:)
    - Can occur when when experimenting with a new S3 command from the command line.
    - Very few of these in production, but many in stage and development environments.



- **missing-ark:** There is an ark in front of the key, but the ark is not in the database.
  - A storage node health check (run every 15 minutes) was leaving files behind with keys that did not conform to our identifier expectations.
    e.g. ark:/99999/test.manifest



- **orphan-copy:** A case of content being present on a node where the database says it shouldn't be.
  - Generally files or objects left over from a migration from one storage node to another, or from an incomplete object deletion.



An object in the cloud contains files from an additional version*, but entries for these files are not present in the database. In other words, a part of the ingest process stalled: the system wrote content to the cloud, but associated entries were not recorded in the database (half-completed object version). This can be caused by our own ingest process via the system's UI, or by submissions from automated, upstream services that may have been unsuccessful and occurred with ineffective error reporting. These were the most difficult to analyze, resulting in a need to contact depositors and other systems administrators.

* "Version clobber": HA ingest: Two storage hosts would pick the same item off the queue to work with, meaning a version of the object would be created twice.

Keys deleted over time

	Jan. 2022	May 2022	Oct. 2022	Jan. 2023
AWS S3 Primary 25,831,597	33,177	23	47	33
Qumulo Primary 54,408,010	2,711	1,347	23	10
Wasabi Secondary 52,495,892	28,888	0	6	0
AWS Glacier Secondary 26,664,351	85,058	0	231	0

Here you can see the number of overall keys to scan as of last week, along with counts deleted after the initial and subsequent scans. With added safeguards and fixes now in place, our most recent scan resulted in only a few items to review.

In conclusion, the very activities that are common to so many digital preservation repositories – migration to new storage services, upstream submissions, object replication and object versioning – are possible triggers for the issues that were uncovered. In this sense, we feel it's important to not only have robust processes that execute these activities, but to have an equally robust method for monitoring their results in the cloud.

Our scan is run every four months, at which point the team gathers and reviews flagged items.

For us, not only has it been a means to improve the overall health of

content in the repository, but it’s proven to be a repeatable mechanism that helps better our work as content stewards, and one that we feel will continue to do so regardless of how we decide to innovate in the future.

Questions? Comments?

Eric Lopatin
eric dot lopatin at ucop dot edu
github.com/elopatin-uc3
Digital Preservation Services Manager

University of California Curation Center (UC3)
California Digital Library