APACHE SLING & FRIENDS TECH MEETUP
BERLIN, 25-27 SEPTEMBER 2017

Binary Data Management Features in Oak 1.8

Matt Ryan, Adobe / Conrad Wöltge, Netcentric

Who are we?

- Matt Ryan
  - Senior Computer Scientist, Adobe
  - Work on Apache Oak
  - Focus on data store

Who are we?

- Conrad Wöltge
  - CTO with Netcentric
  - 10 years with Sling
  - Profile with high performance, large scale environments

Disclaimer

- We are representing Oak today, not Adobe
  - Presentation content includes things the Oak team is exploring for a future version of Oak
  - Presentation content should not be interpreted as a promise or expectation of any future capability or feature of AEM

Agenda

- CompositeDataStore
- Other Binary Data Management Challenges in Apache Oak
  - Unmanaged Binaries
  - Direct Read-Only Access
- Q&A

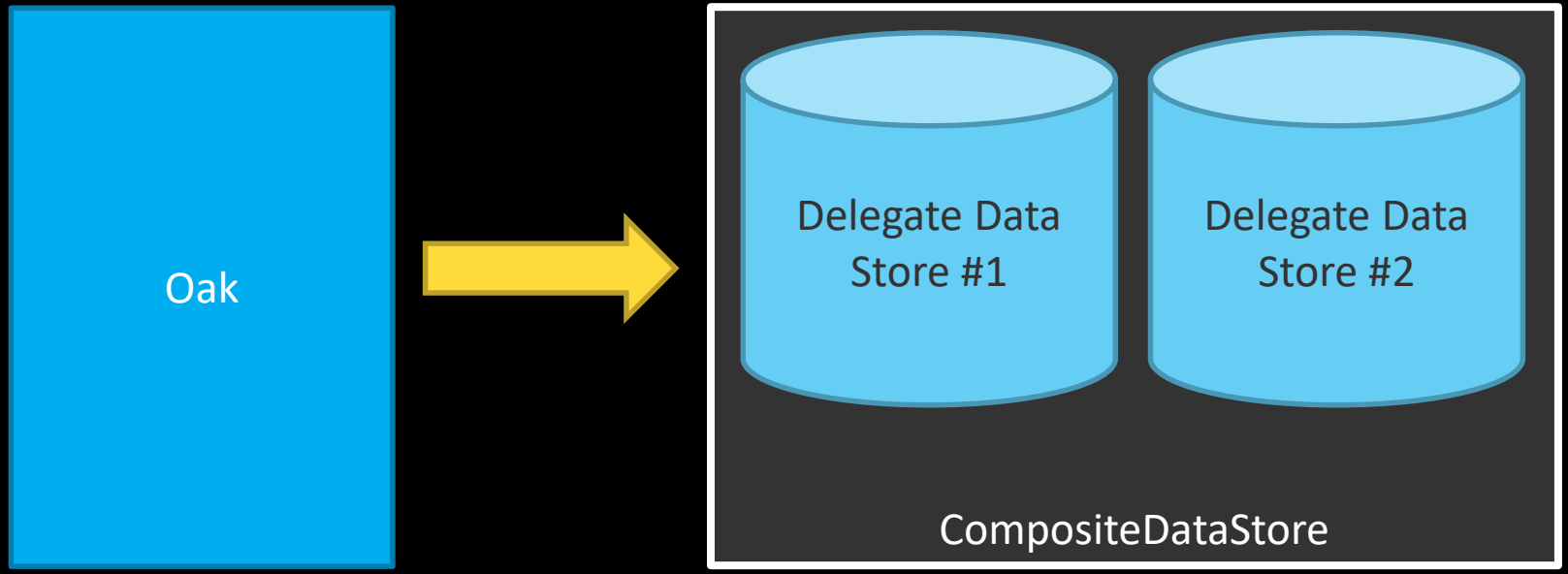
adaptTo()

Oak CompositeDataStore

Motivations for CompositeDataStore

- Have you ever wished you could:
  - Have read-only access to an Oak data store?
  - Automatically archive infrequently-used data?
  - Use criteria other than file size to determine where to store data?
  - Store different types of data in different data stores?

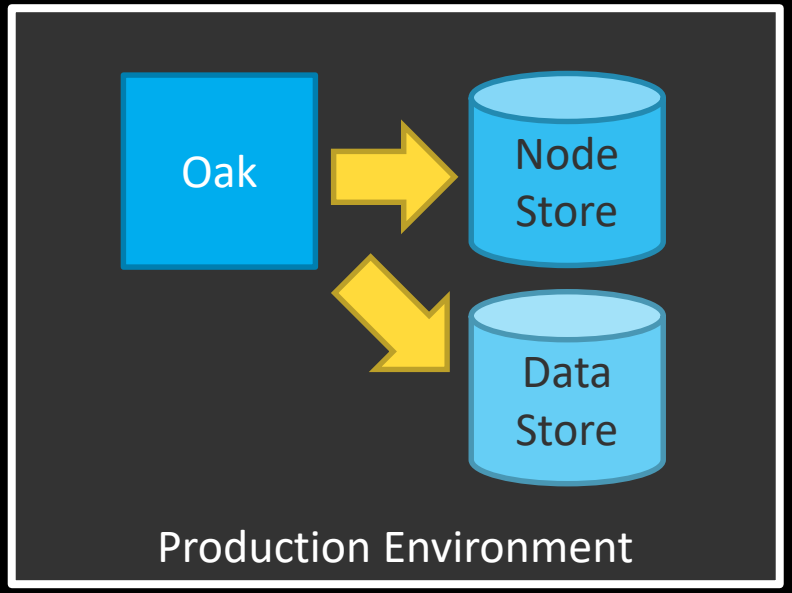
What is a “CompositeDataStore?”



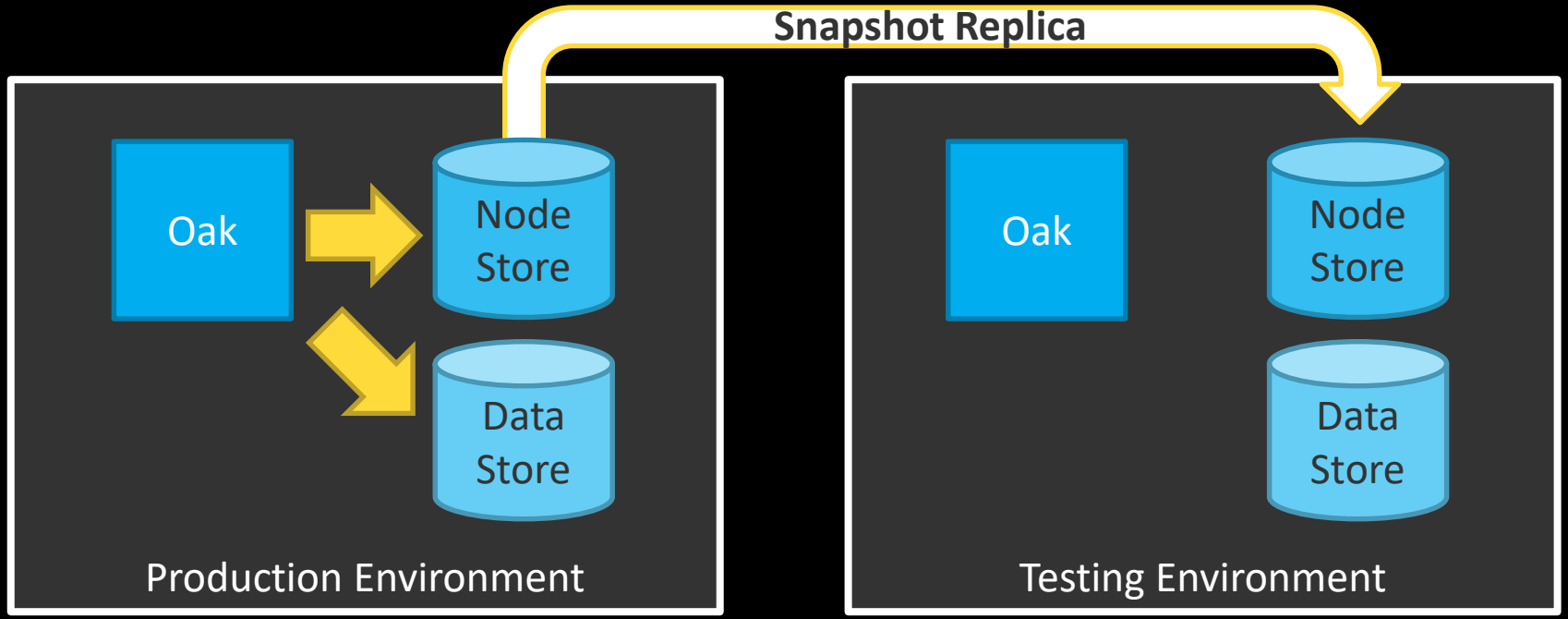
Use Case – Testing Environment

- User may wish to mimic production environment in a testing environment
  - Duplicating the production data store may be cost and time-prohibitive
  - What if Oak could access the production data store read-only?

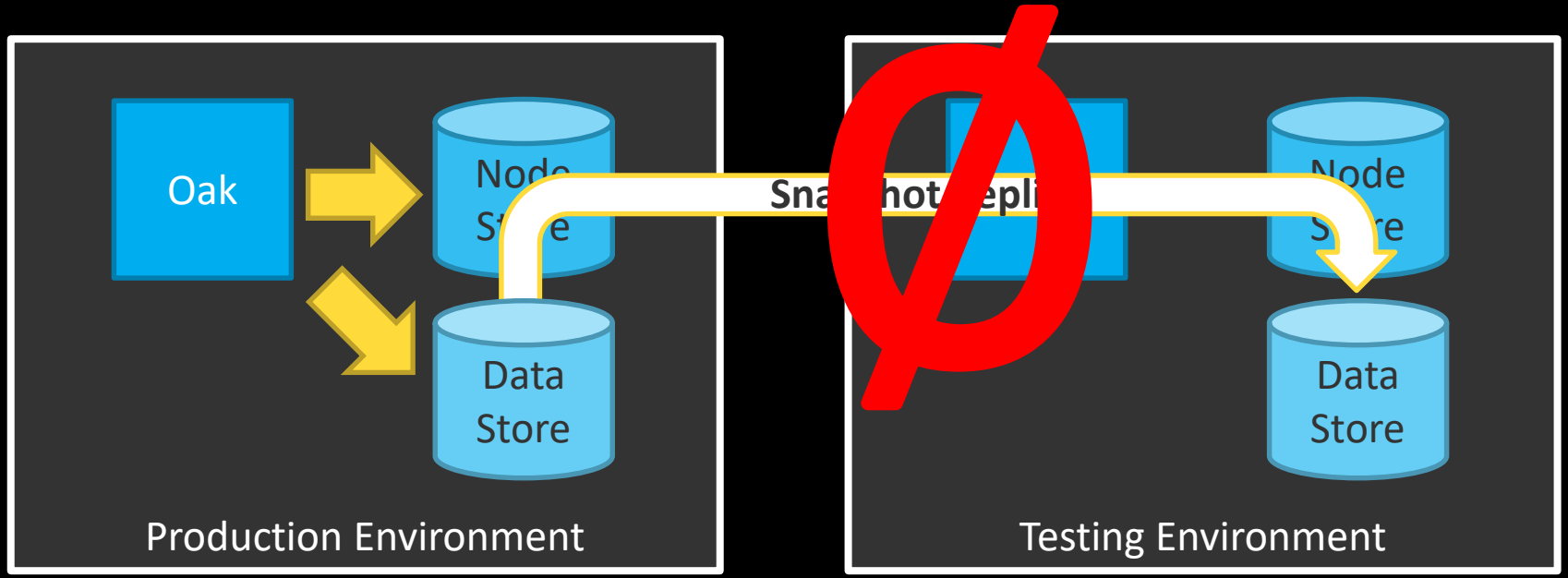
Use Case – Testing Environment



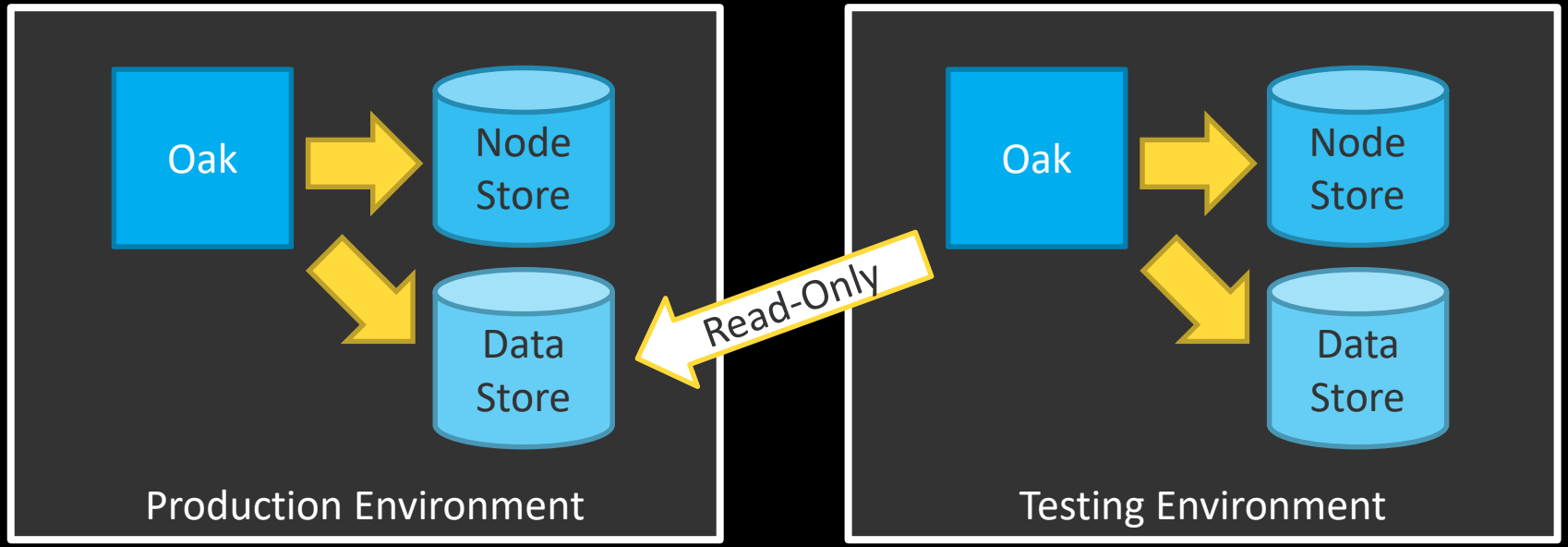
Use Case – Testing Environment



Use Case – Testing Environment



Use Case – Testing Environment

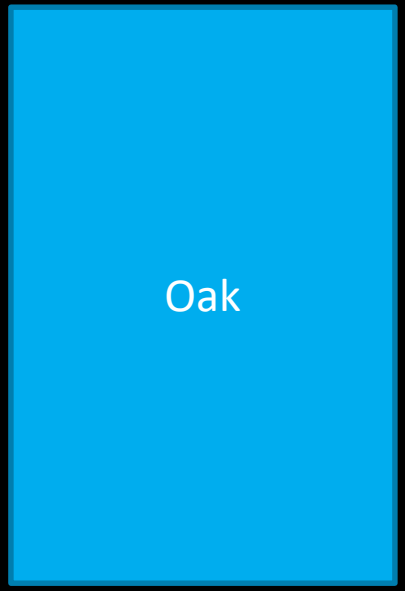


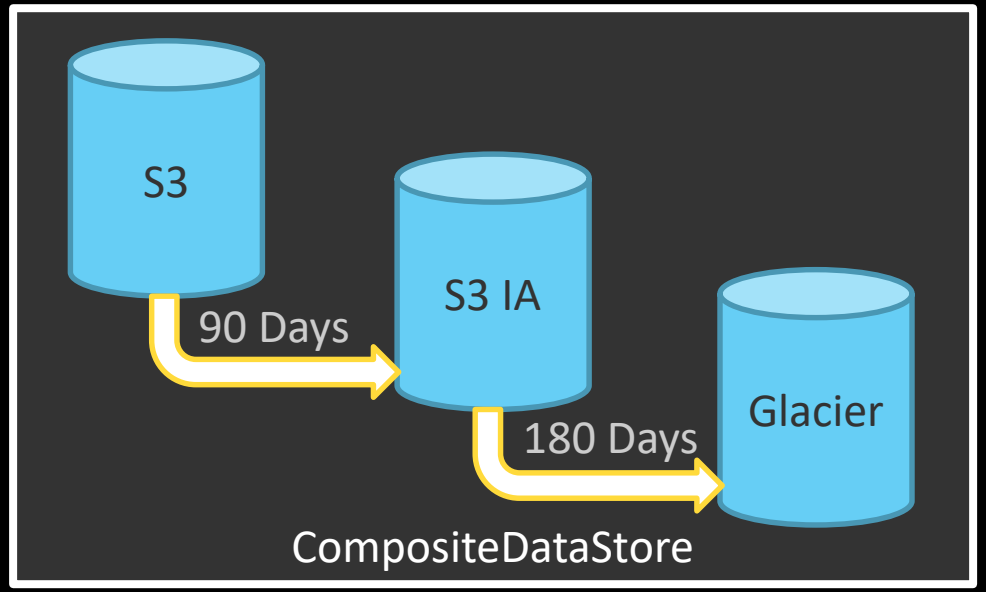
adaptTo()

Use Case – Automatic Archiving

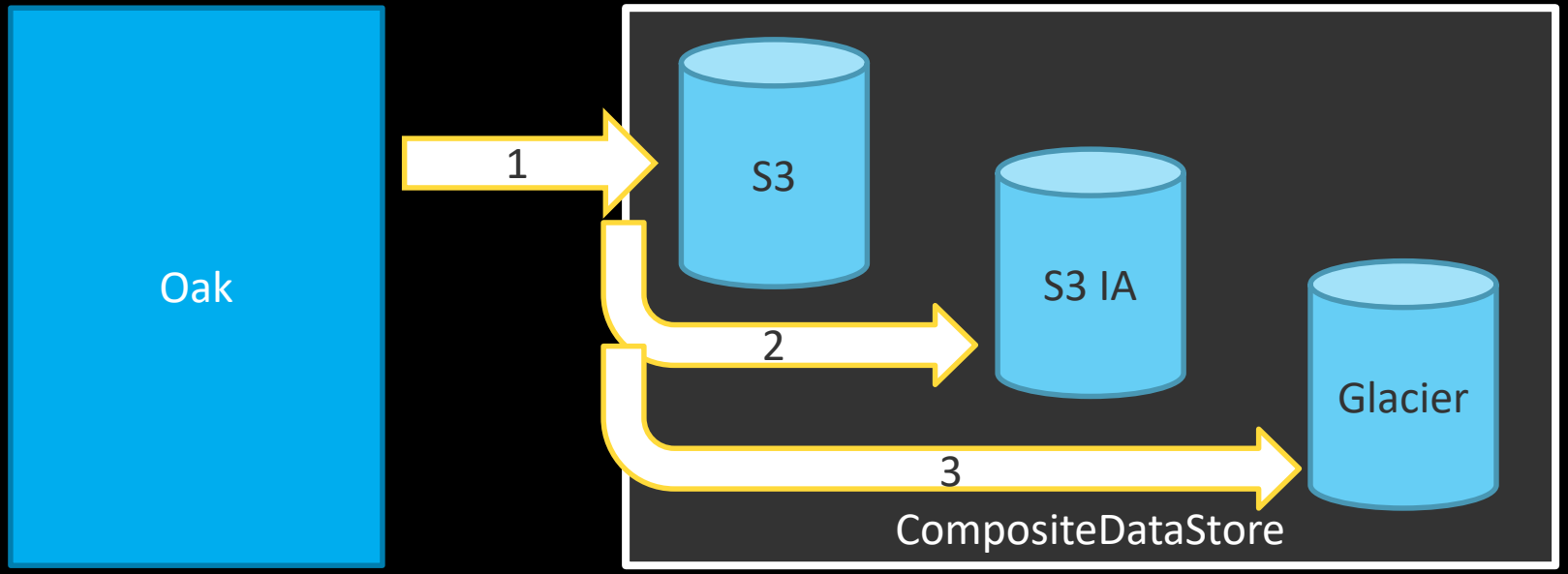
- Many cloud-storage systems offer different storage classes, or tiers
  - E.g. AWS S3, S3 IA, Glacier
- What if Oak could represent data in multiple storage classes in a single logical data store?

Use Case – Automatic Archiving





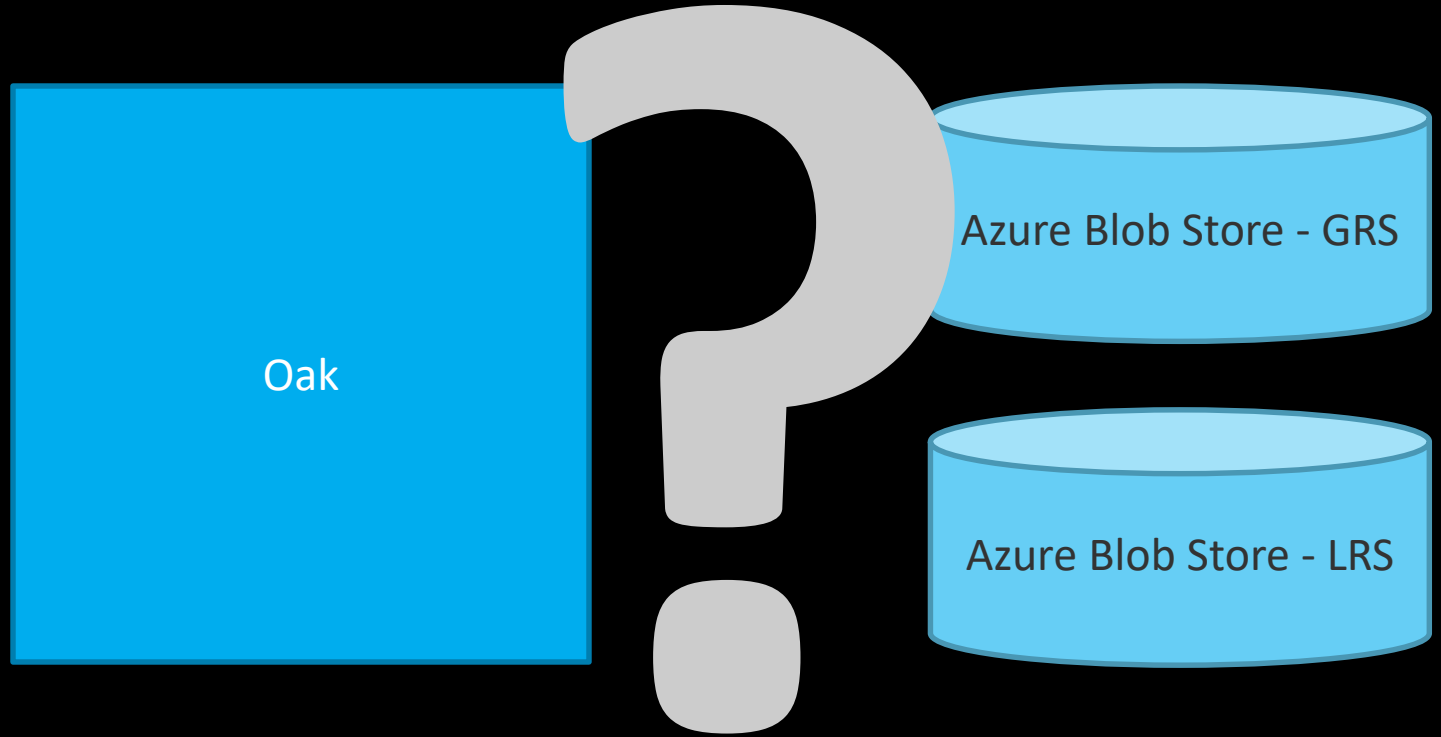
Use Case – Automatic Archiving



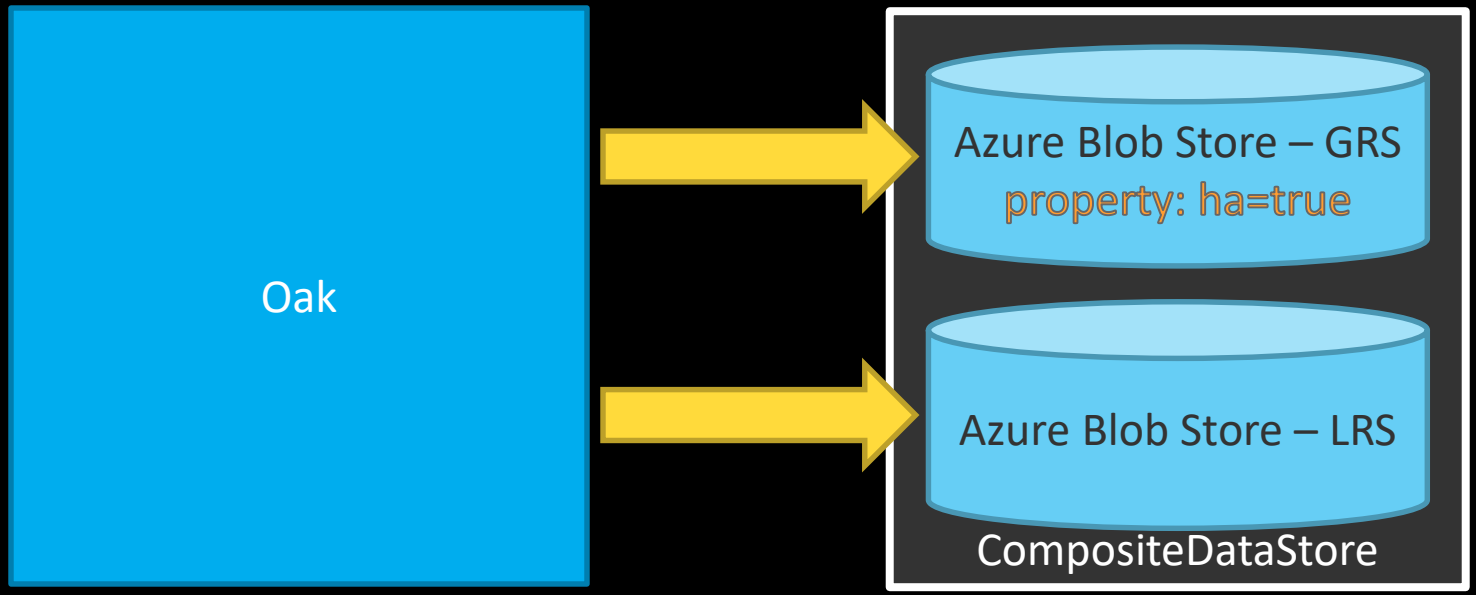
Use Case – Designate Highly-Available Data

- Some types of storage may offer more highly-available storage, at a higher expense
  - Putting the entire repo in HA storage costs more
  - Putting the entire repo in standard storage means none of it is as highly available
- What if Oak could select where to store data based on some criteria tied to the data?

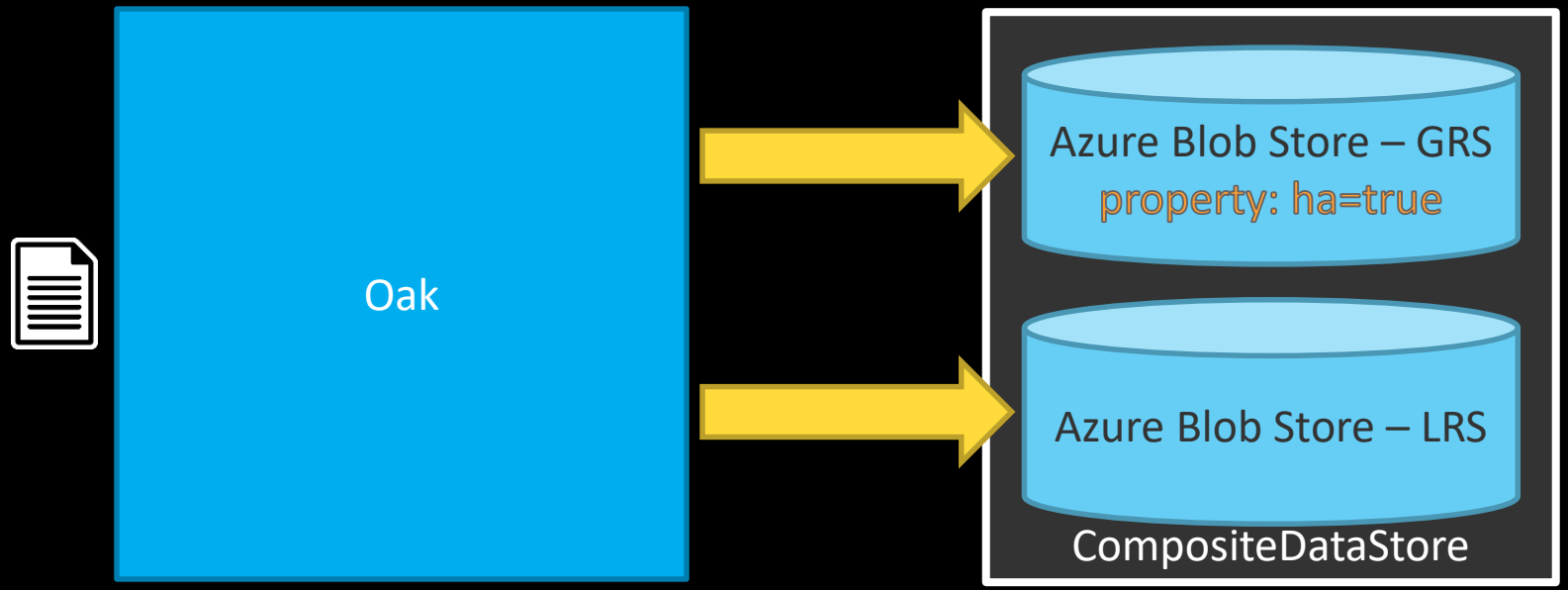
Use Case – Designate Highly-Available Data



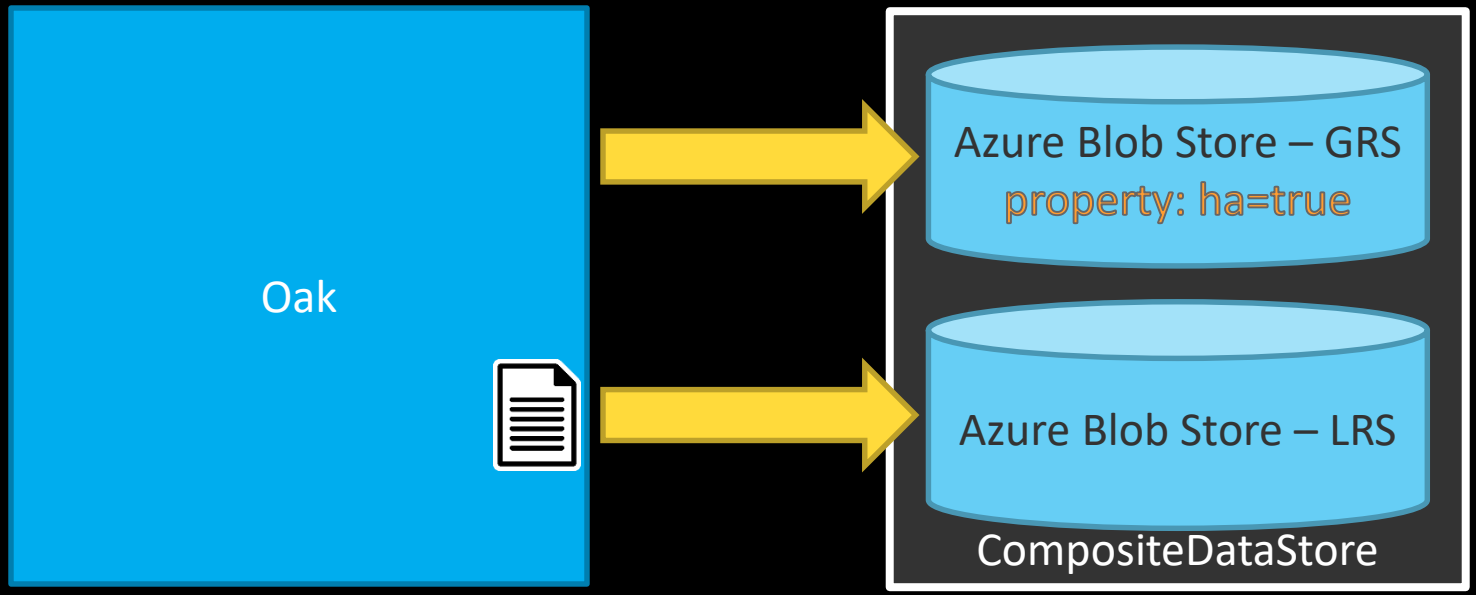
Use Case – Designate Highly-Available Data



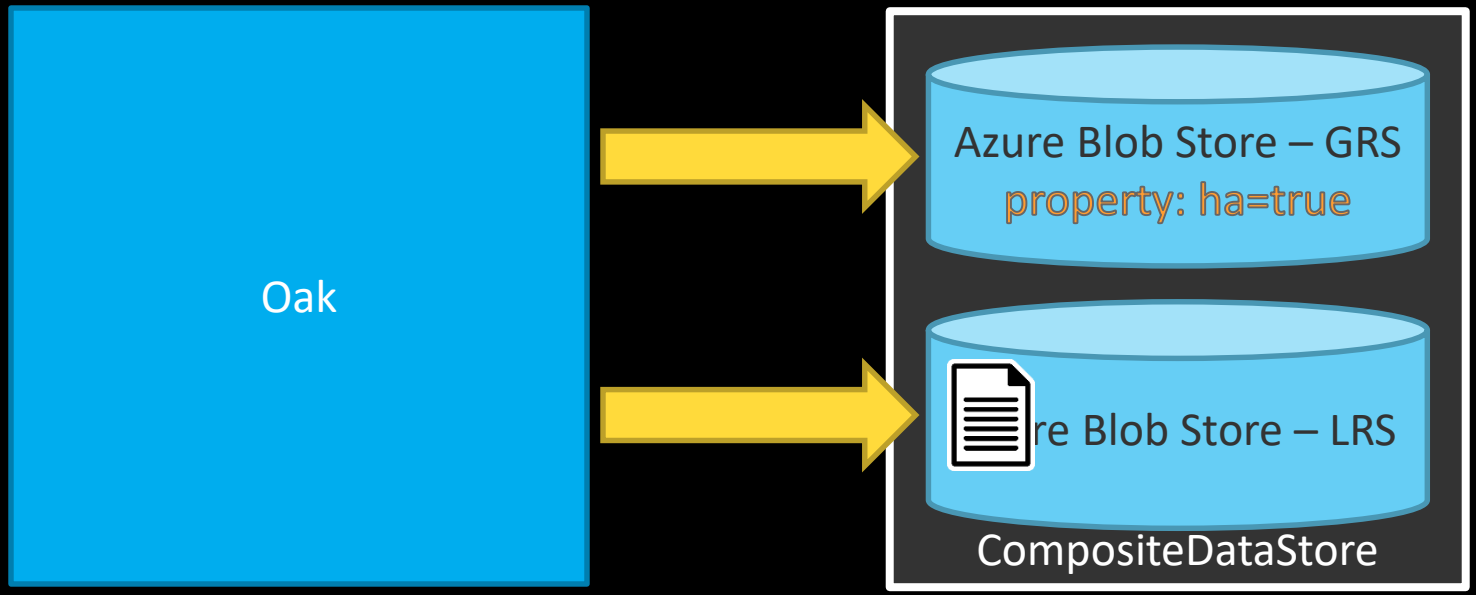
Use Case – Designate Highly-Available Data



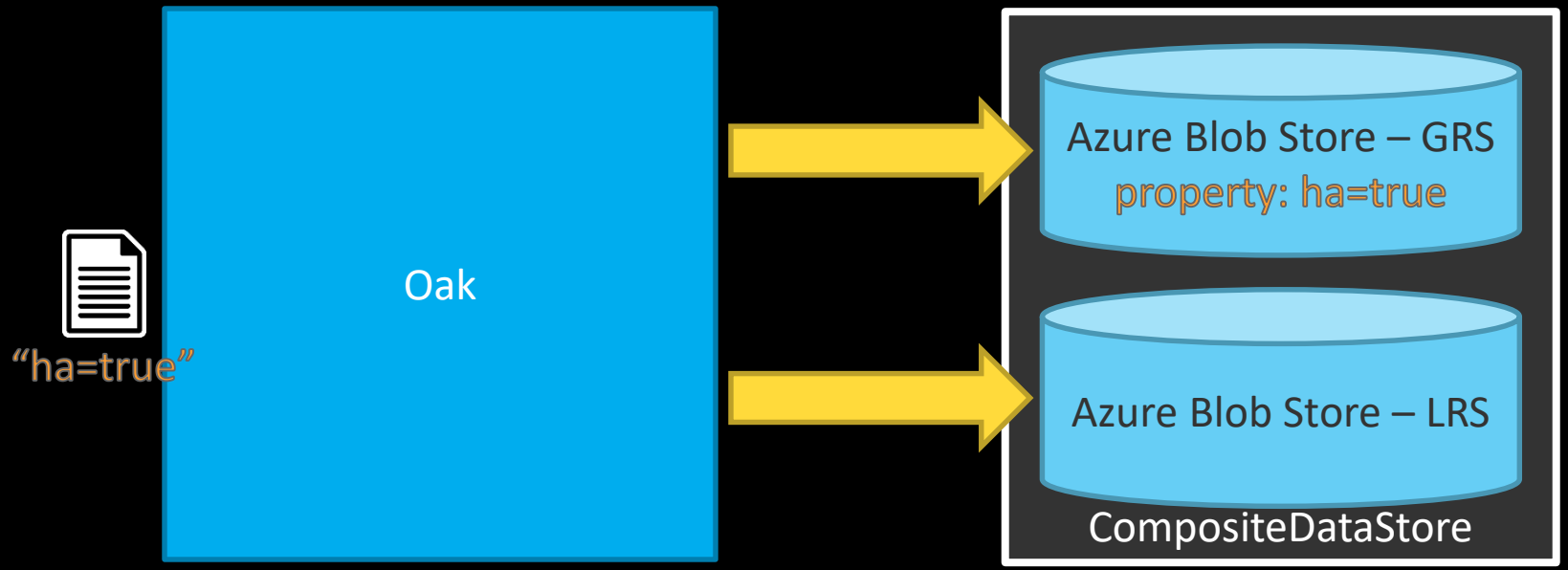
Use Case – Designate Highly-Available Data



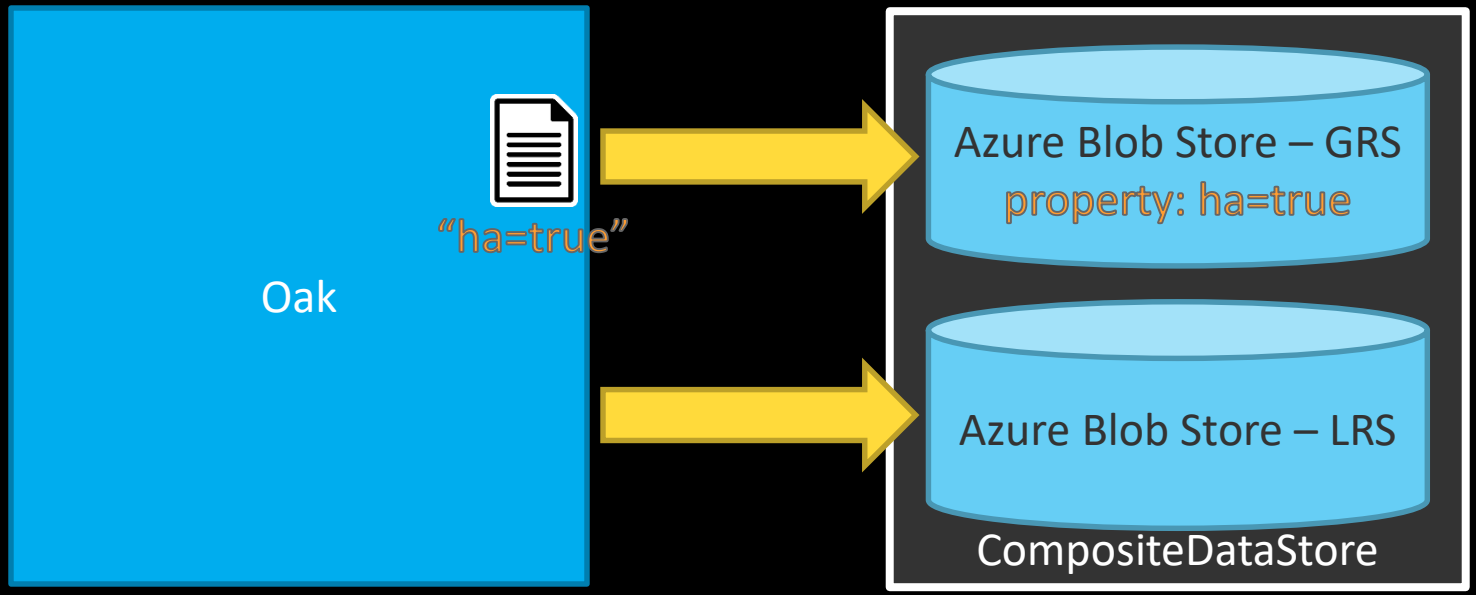
Use Case – Designate Highly-Available Data



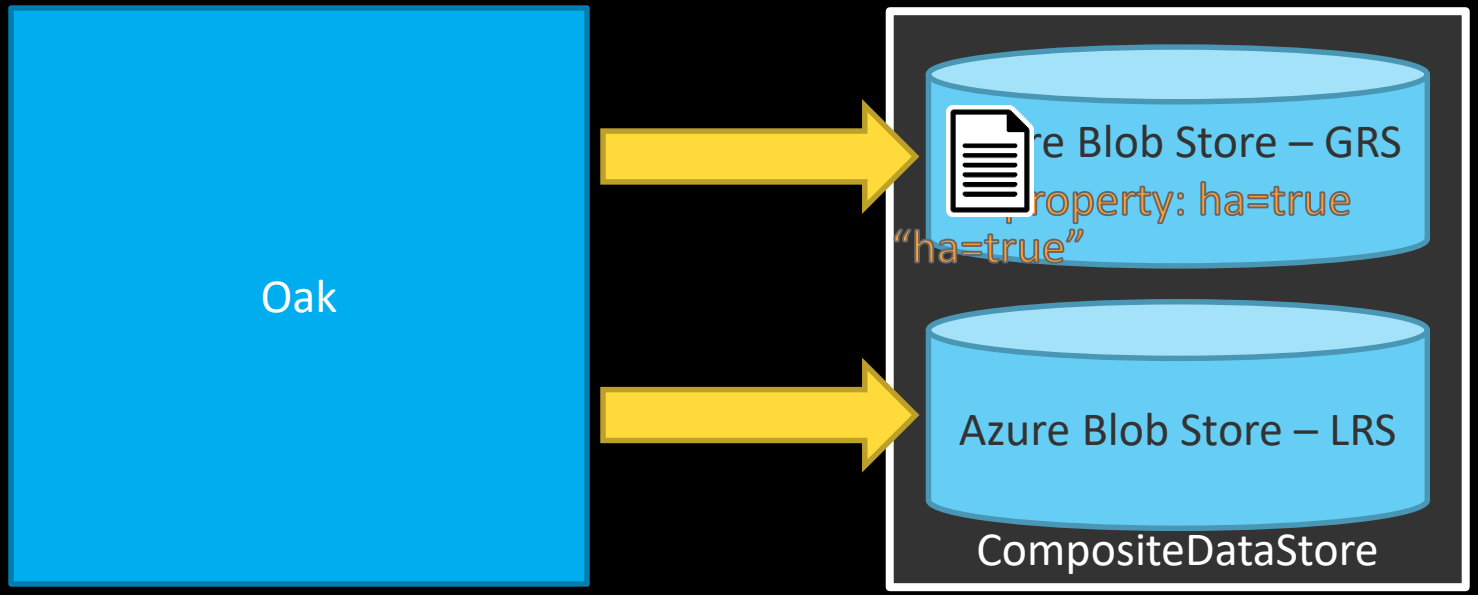
Use Case – Designate Highly-Available Data



Use Case – Designate Highly-Available Data



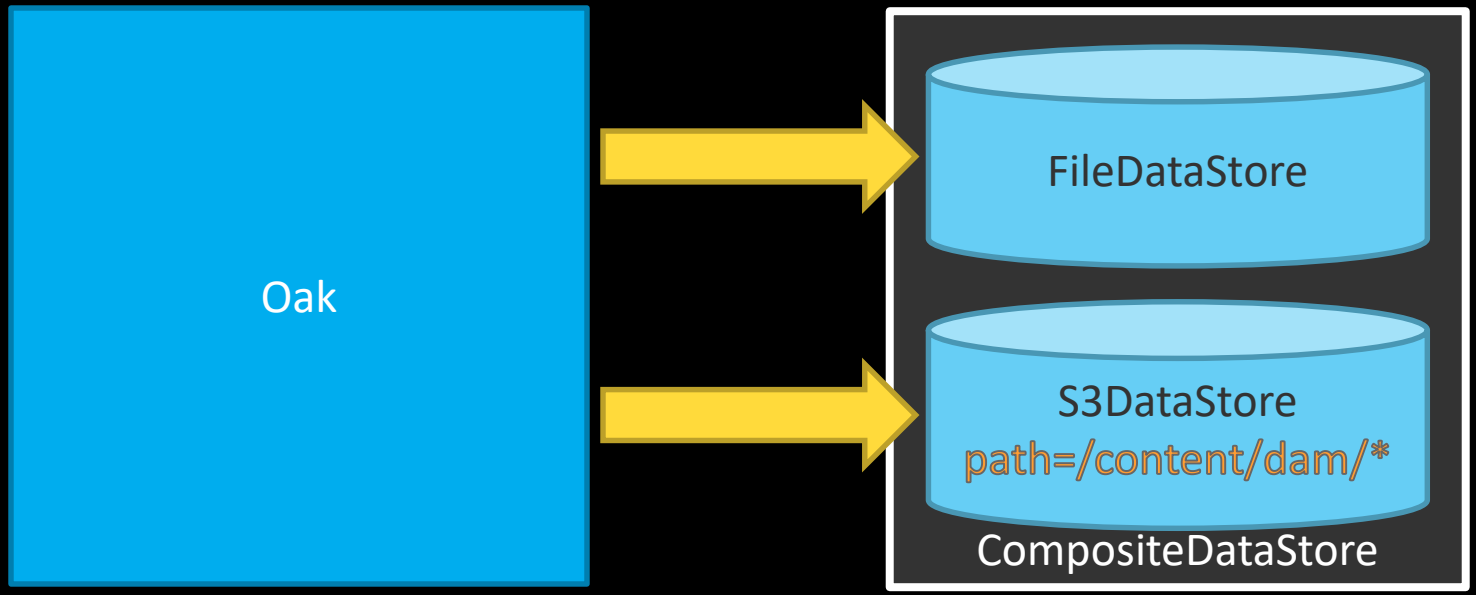
Use Case – Designate Highly-Available Data



Use Case – Store Binaries by Type

- An Oak user may wish to make use of cloud storage certain types of files, but keep other files local
- What if Oak could choose where to put a binary based on the node type or path?

Use Case – Store Binaries by Type



Status

- Demoable POC –Testing Environment
  - Code is available at: https://github.com/mattvryan/jackrabbit-oak/tree/composite-data-store

Demo

Additional Binary Data Management Challenges in Apache Oak

adaptTo()

Challenges

- How do we quickly add a large content repository to Oak?
- How do we responsibly allow direct access to a binary?

Challenges

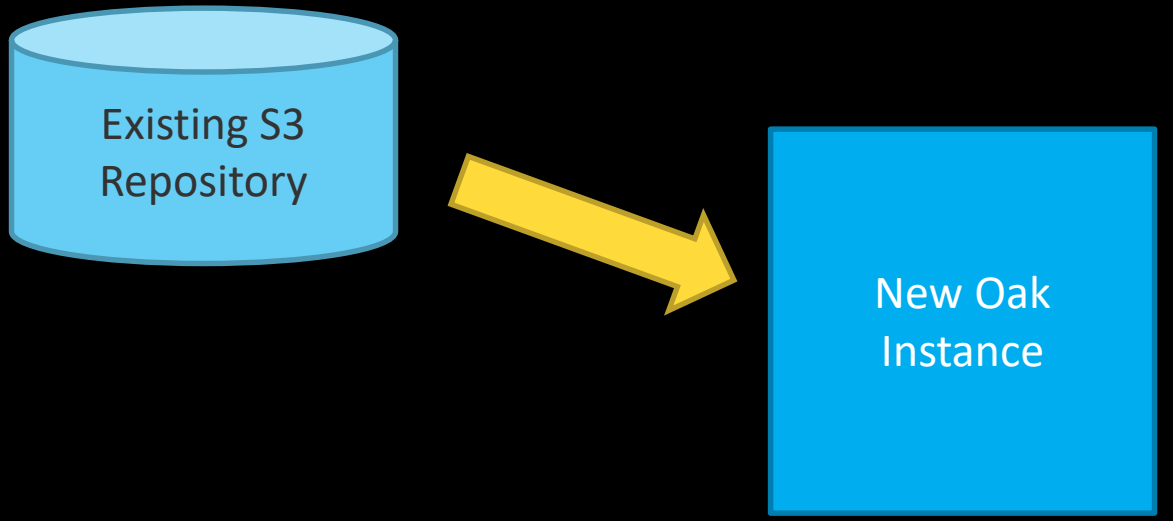
- How do we quickly add a large content repository to Oak?
- How do we responsibly allow direct access to a binary?

Adding Existing Repositories to Oak

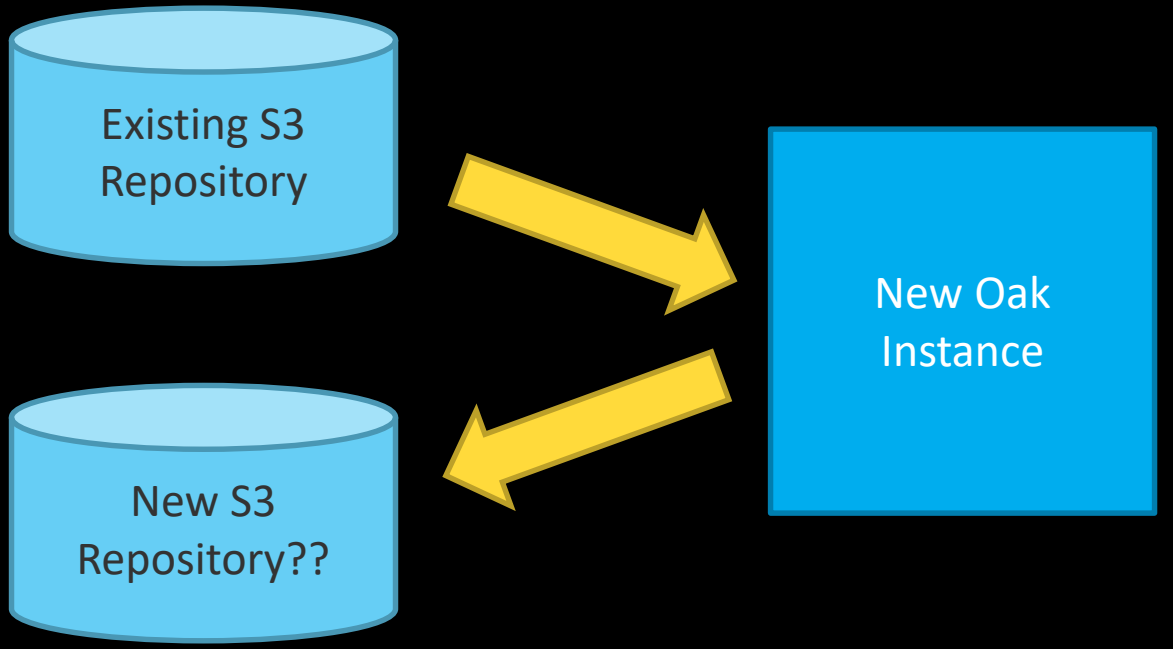
User has existing 1TB content repository

Adding Existing Repositories to Oak

User wants to use Oak to manage the content repository…



Adding Existing Repositories to Oak



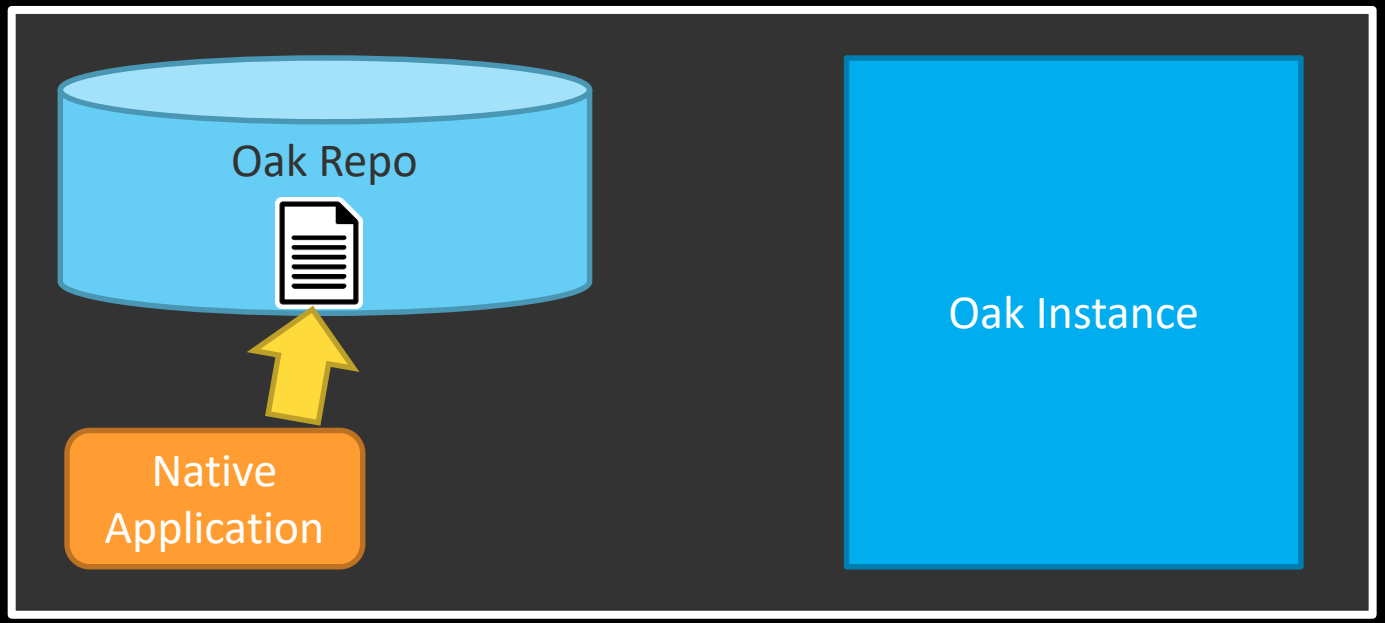
User must import entire existing repository into a new repository

Adding Existing Repositories = Expensive

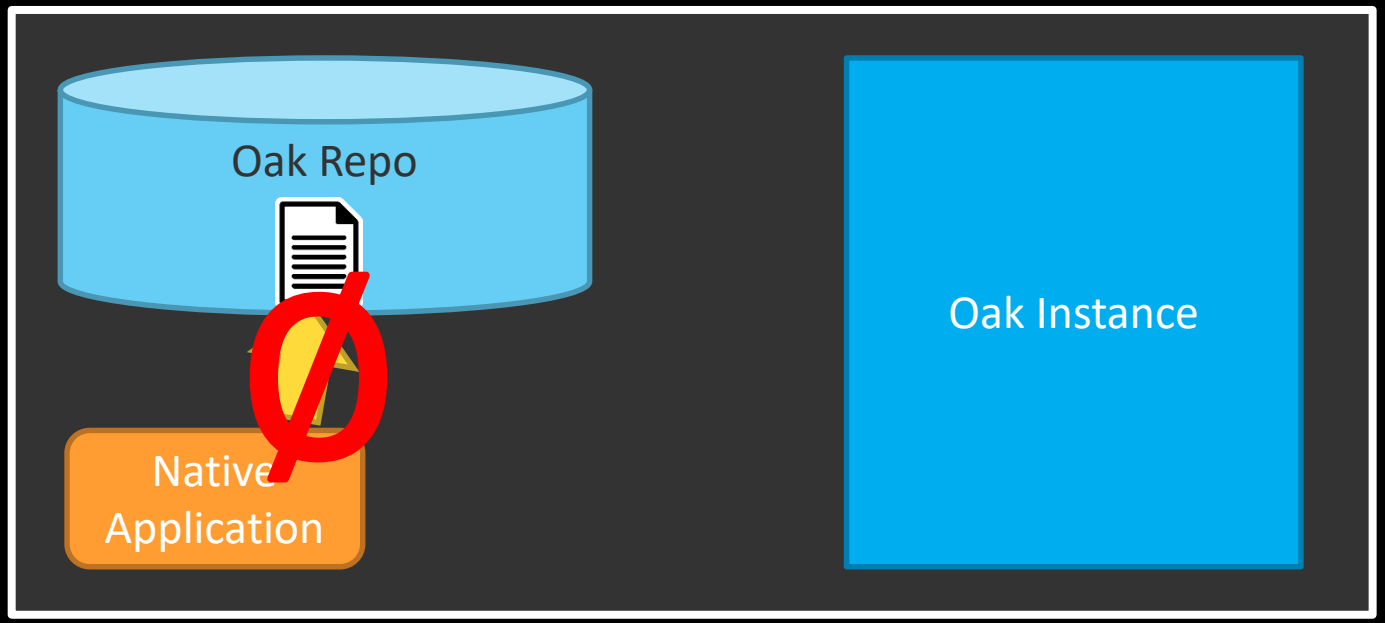
Challenges

- How do we quickly add a large content repository to Oak?
- How do we responsibly allow direct access to a binary?

Direct Binary Access



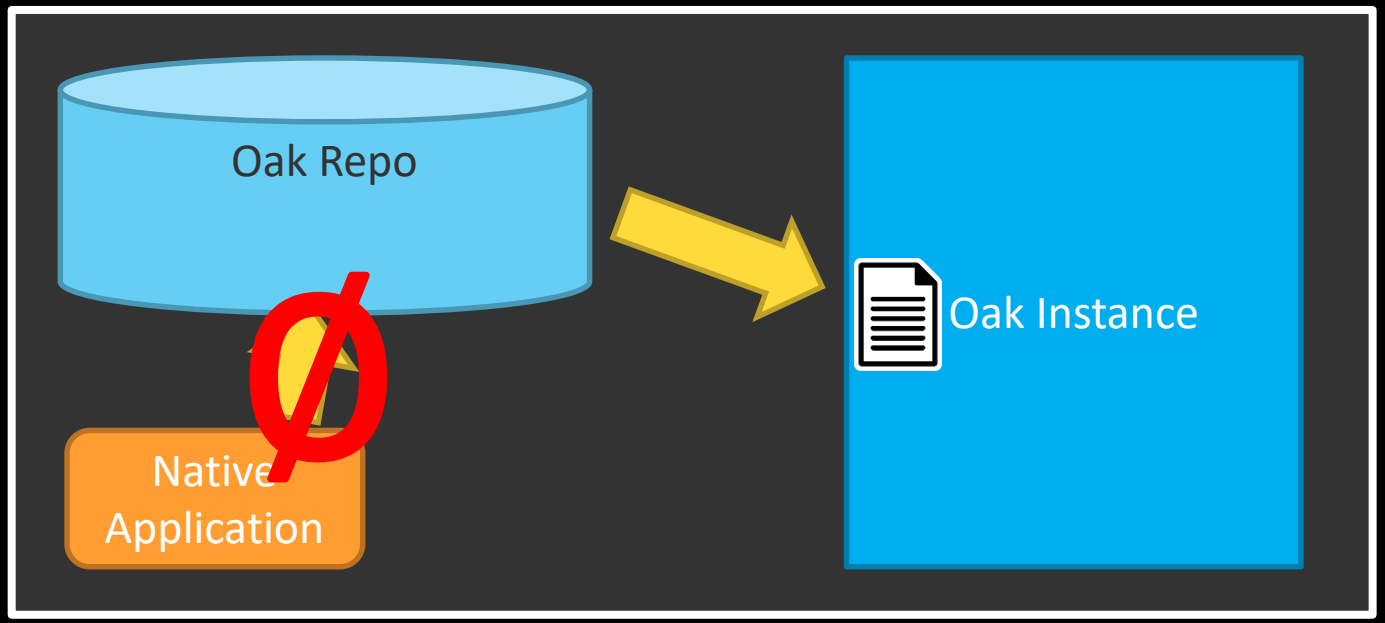
Direct Binary Access



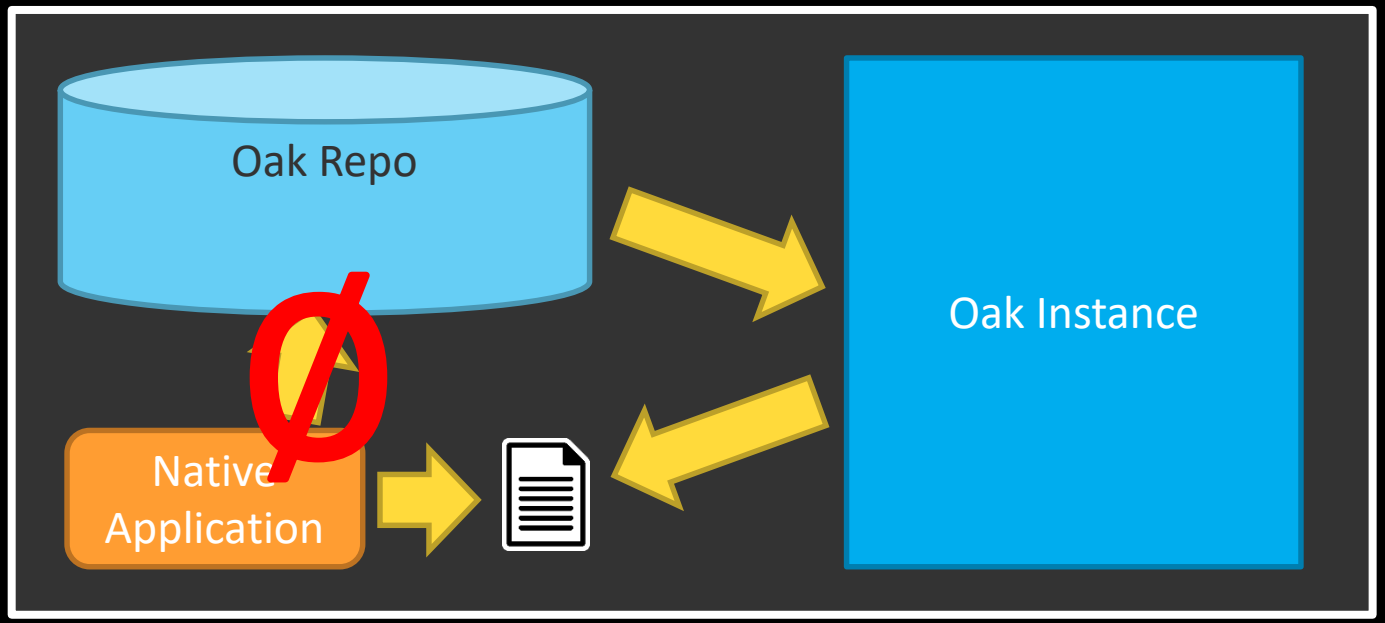
Challenges with Direct Access

- Oak maps nodes to blobs
- Oak controls access rights
- Oak can’t support data that is accessed directly

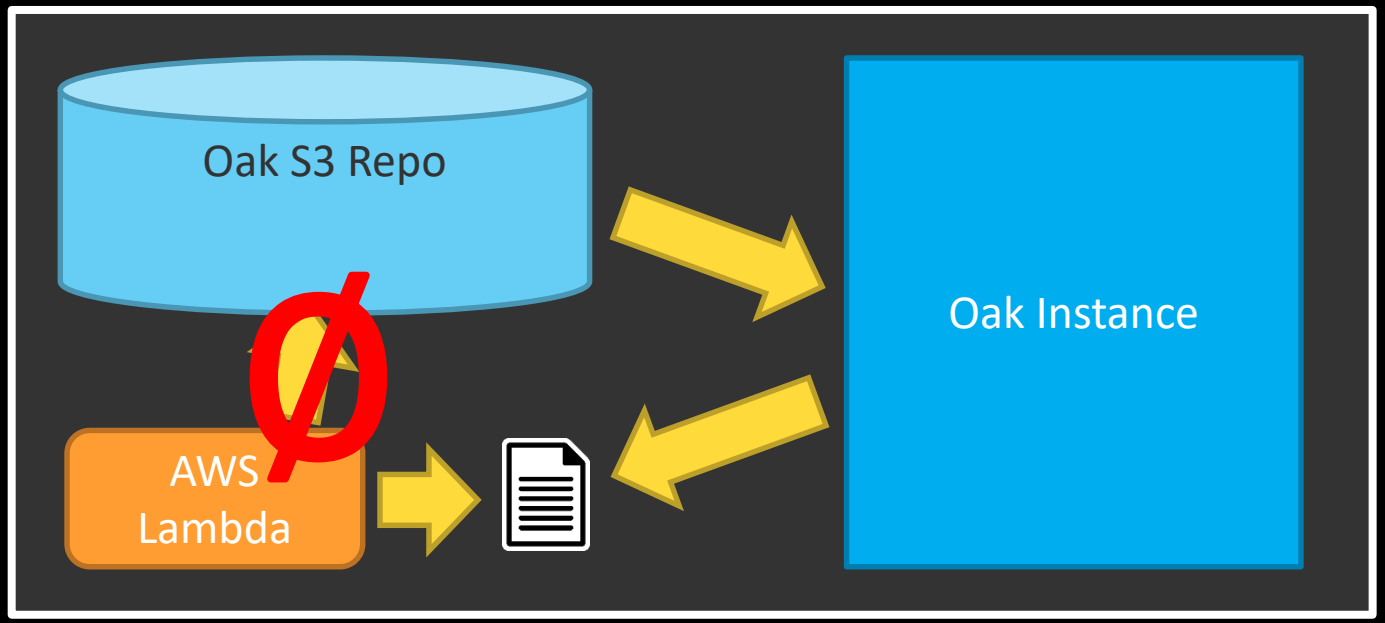
Direct Binary Access



Direct Binary Access



Direct Binary Access



Processing Large Binary Files = Expensive

€

Unmanaged Binary Support

adaptTo()

Unmanaged Binary

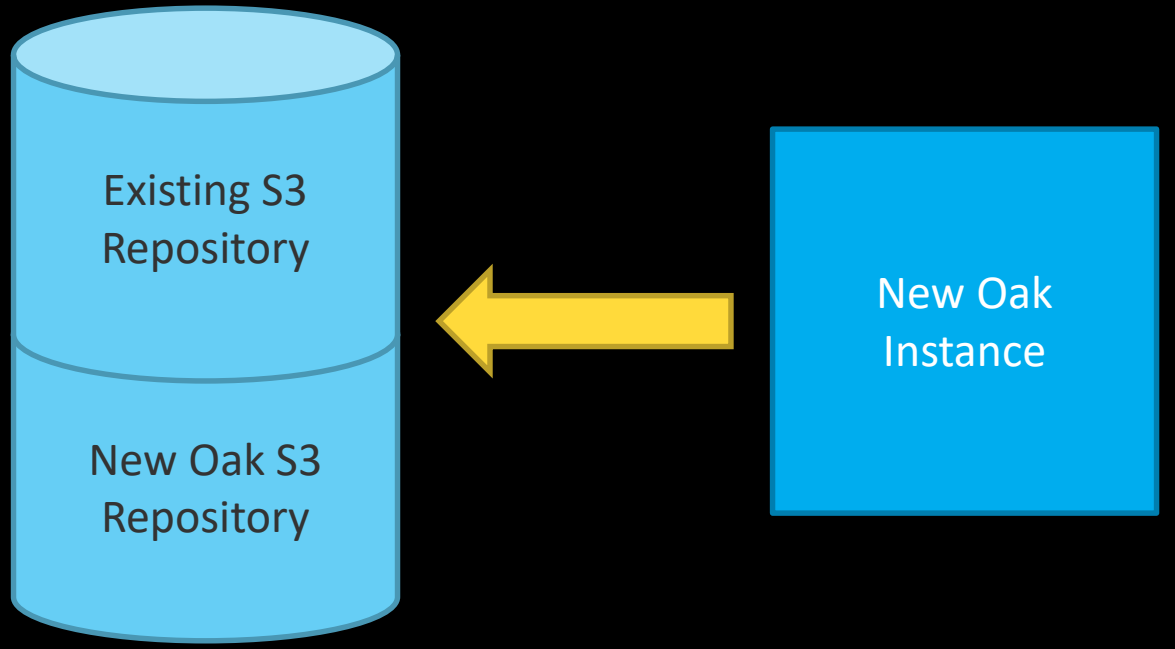
- Oak does not “manage” the binary
  - No deduplication
  - No garbage collection
  - No indexing
  - No guarantees on access permission, availability or existence

Unmanaged Binary

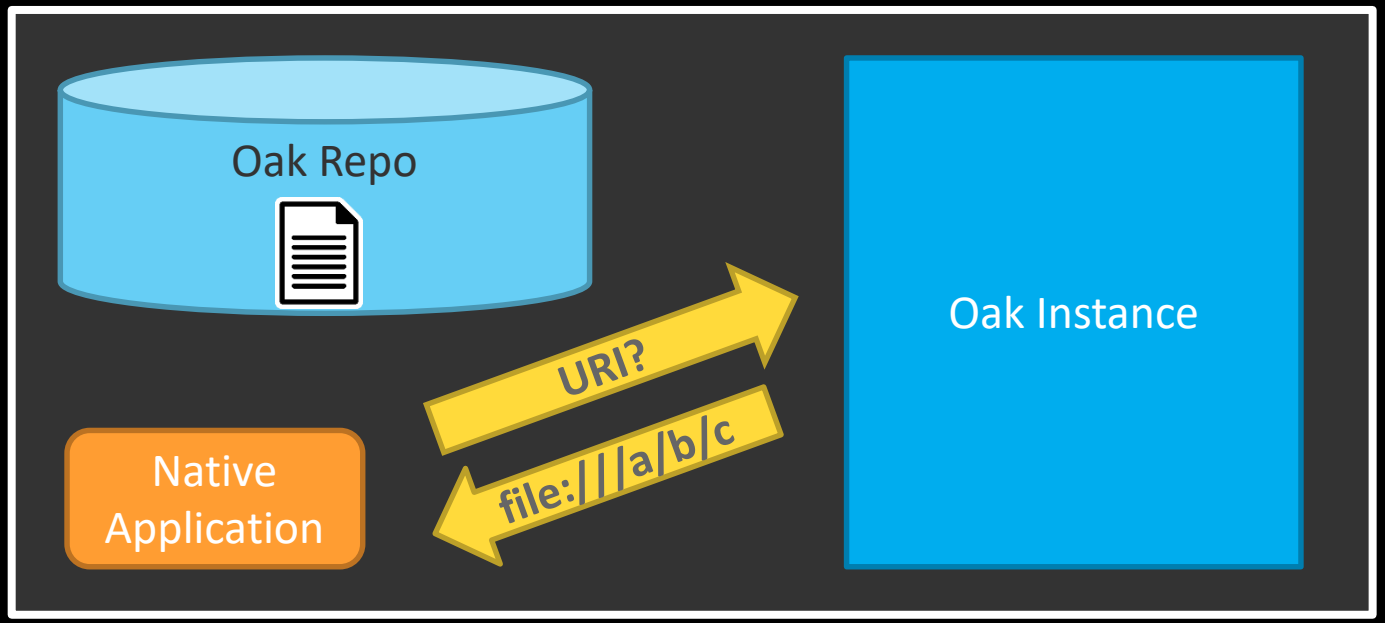
- Oak knows about the binary
  - Oak stores a reference to the binary
  - Oak can fulfill requests for the binary
  - Oak can return a URI for direct access to the binary

Adding Existing Repositories to Oak

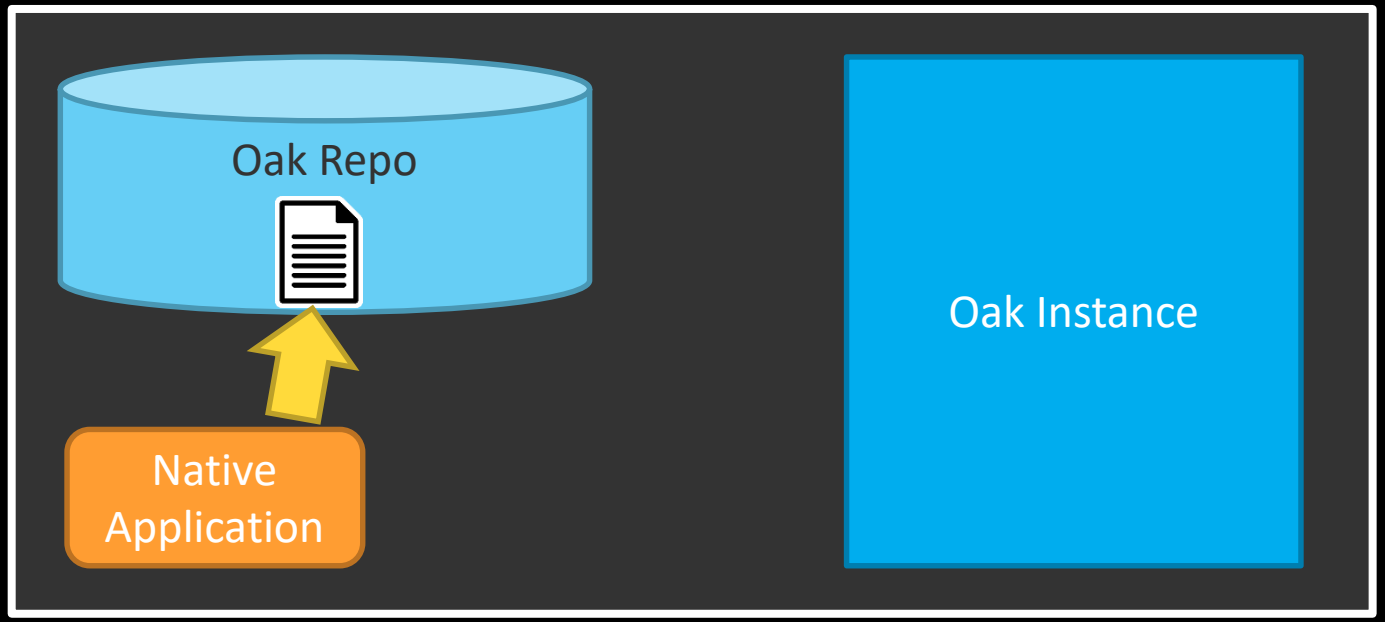
Oak can reference existing (unmanaged) repo immediately – new (managed) binaries in same bucket



Direct Binary Access



Direct Binary Access



Possible Usage (not final)

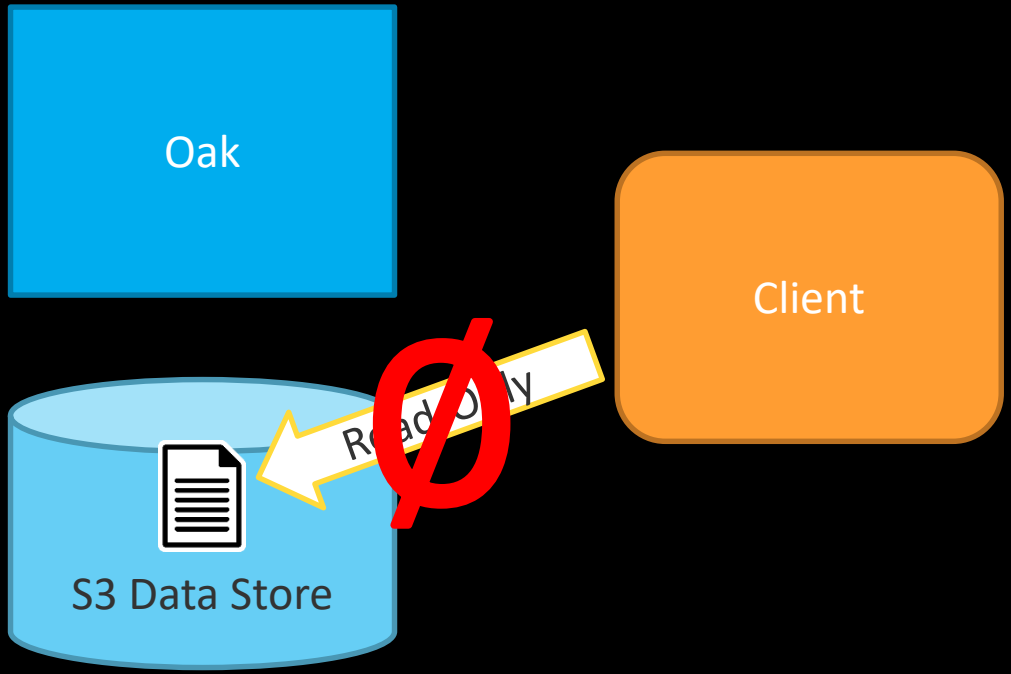
```
Resource resource =
  resourceResolver.getResource(
    “/path/to/unmanaged_binary.jpg”
  );
URI unmanagedBinaryUri =
  resource.adaptTo(URI.class);
```

Status

- We are hoping to start working on this soon
  - Does anyone want to help? Let's talk

Direct Read-Only Access via Signed S3 URI

What About Read-Only Access?



Read-Only Access Disallowed Because:

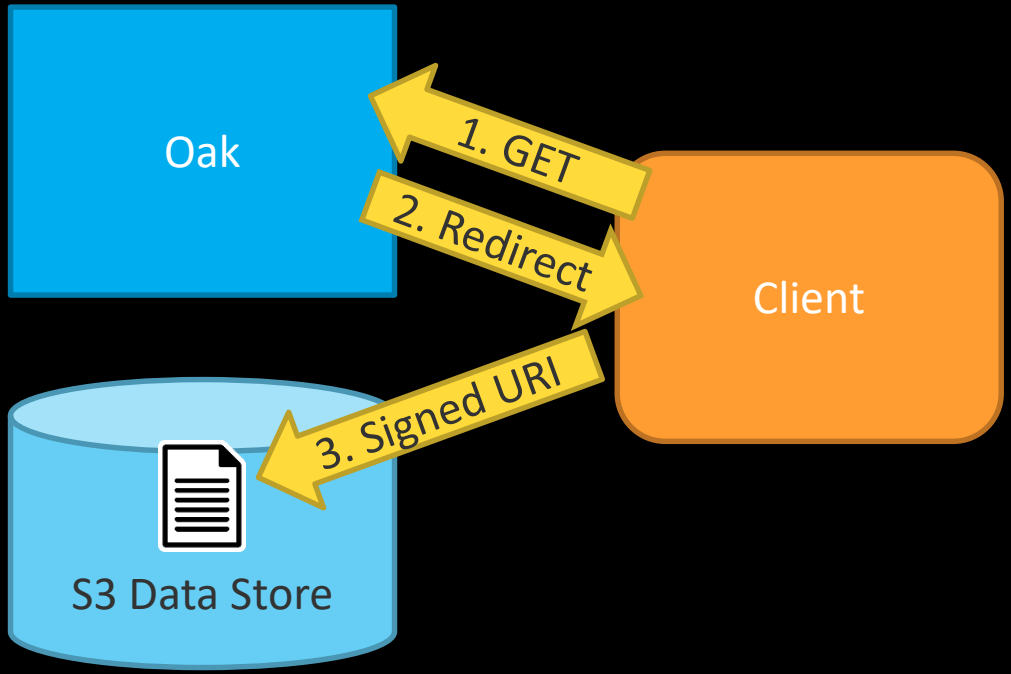
- Only Oak knows the real binary name
- Only Oak knows whether the client has rights to access the binary

adaptTo()

Oak Signed URI

- Data store can deliver a signed URI on request if:
  - Backend supports it (e.g. S3, Azure)
  - User has rights to the object
- Short TTL
- Issued as redirect from Sling

Oak Read-Only Access



1. Client requests binary from Oak
2. Oak creates signed URI and returns as redirect
3. Client requests binary directly from S3

adaptTo()

Q&A

Contact Info

Matt:

- @mattvryan
- linkedin.com/in/mvryan/

Conrad:

- @mahatmac
- linkedin.com/in/conradwoeltge/


adaptTo()