The ORA4 DPS Hyrax/OCFL connector

Connecting the ORA4 Hyrax Review application to the OCFL-based ORA4 Digital Preservation Solution (ORA4 DPS)

About this document

- Version 0.1
- Author(s): Tom Wrobel <thomas.wrobel@bodleian.ox.ac.uk>
- Created: 23 April 2019
- Last modified: 25 April 2019

ORA4

The fourth generation of the Oxford Research Archive system will have three core application components (the core technology behind each is given in parentheses):

- ORA4 Public: A public search and discovery application (Blacklight)
- ORA4 Review: An ingest and mediated review application (Hyrax)
- ORA4 Digital Preservation Solution (ORA4 DPS): A digital preservation store to guarantee long-term service sustainability (OCFL)

Overview

The ORA4 Review system is responsible for maintaining objects in progress.

At certain gateways in the ORA4 workflow, a digital preservation copy of an object will be preserved in the ORA4 DPS.

The ORA4 DPS will constitute a set of ORA4 objects stored in an OCFL storage root: both binary files and serialised metadata.

The ORA4 DPS will also act as the basis for a disaster recovery system for ORA4 as a whole: i.e. ORA4 Review can be rebuilt from the ORA4 DPS, and ORA4 Public can be rebuilt from ORA4 Review

This presentation discusses the architecture that will enable ORA4 Review to import/export objects to and from the ORA4 DPS

Requirements

The ORA4 Digital Preservation Product Description requires that the ORA4 DPS *MUST*:

- Store ORA4 objects in multiple versions on the file system in a documented way that provides long-term data integrity and enables digital preservation best practices
- Round-trip an ORA4 object (files and metadata) between Hyrax and the digital storage solution and back again
- Provide ORA4 with the basis of a disaster recovery solution for both ORA4 Review and ORA4 Public
- Processes ORA4 objects sufficiently quickly to allow for the practical storage and recovery of the entire ORA4 Review system

The ORA4 DPS *SHOULD*:

- Interface with the Hyrax system via its Sword2 endpoint
- Use the Oxford Common Filesystem Layer (OCFL) standard for digital object storage

Project documentation

The requirements for this project have been turned into a a series of MoSCoW tagged user-story backed Github issues for the ORA4 Hyrax/DPS connector: https://github.com/tomwrobel/ora_ocfl/issues/

These issues also track whether or not the current command line clients can provide the required functionality.

The user stories themselves are listed at the end of this presentation.

Existing components (or in active development)

- ORA4 Hyrax application (Hyrax 2.4, built on Fedora 4.7)
- A Sword2 endpoint for the Hyrax application, capable of round-tripping ORA4 metadata and files (https://github.com/cottagelabs/willow_sword branch: feature/ora_customisations)
- A Hyrax worker to import ORA4 objects from JSON into ORA4 Review – this could be an alternative way to write content back to ORA4 if Sword2 proved too slow.
- A command line OCFL client. Several exist, but the most complete so far is Aaron Birkland's Go-based OCFL client: https://github.com/birkland/ocfl (in development)
- A prototype API for interacting with the OCFL client via REST: https://app.swaggerhub.com/apis/OCFL_client/ocfld/ (in development)
- A description of how ORA4 objects will be preserved in OCFL, including the XML format used for metadata serialisation: https://github.com/tomwrobel/ora_ocfl

Sword2 or not Sword2?

In what follows, the use of the Sword2 endpoint as a component is desirable but not essential. Both of the proposed architecture options come in two flavours (a+b), with and without Sword2 as a component. Points to consider

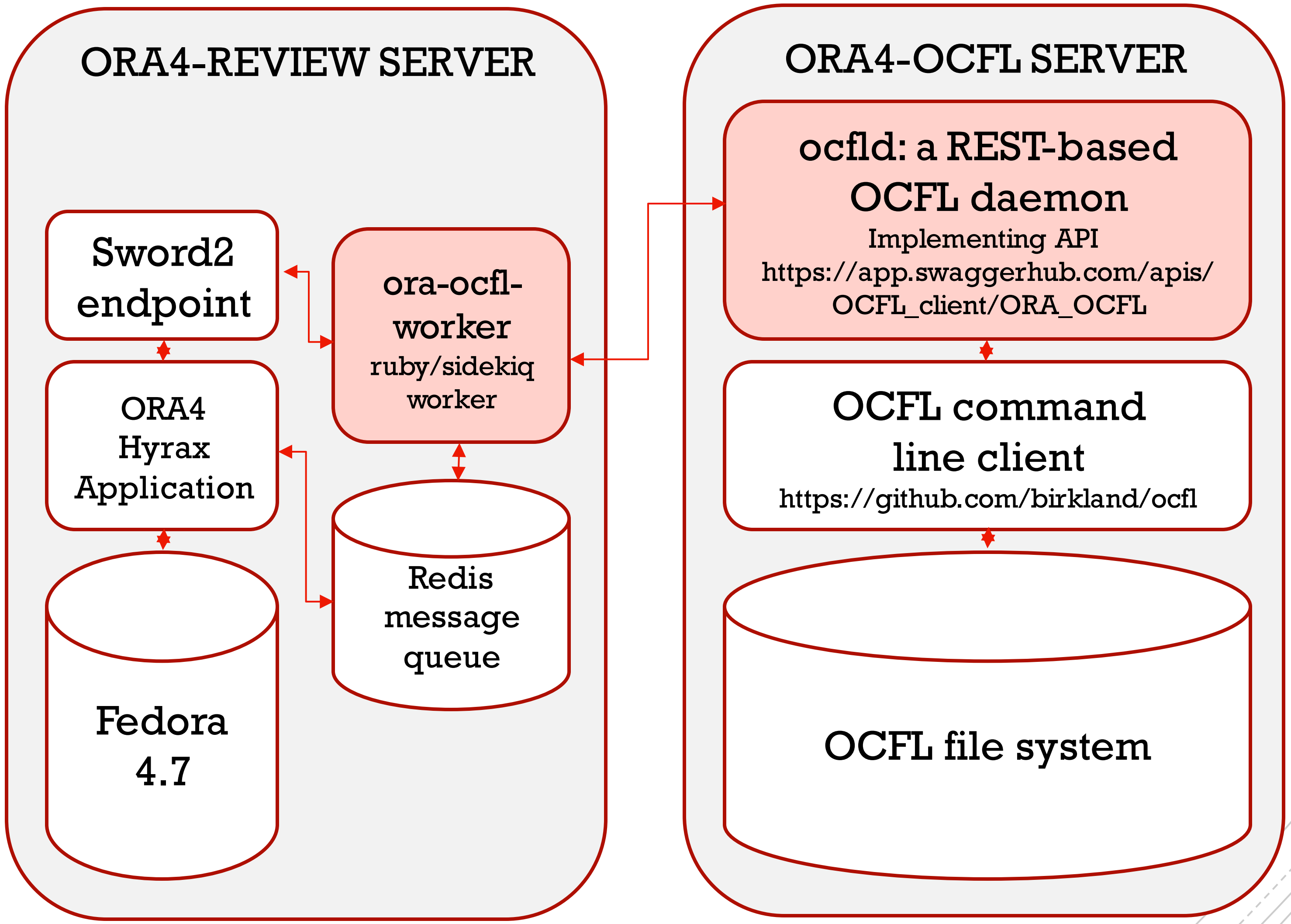
- Sword2 uses the same ingest/export mechanism as the Symplectic Elements Repotool 2 connector
- Sword2 is a proven method of moving ORA4 objects in and out of Hyrax
- Sword2 lowers overall system maintenance requirements, as we will be using this in any case for Symplectic Elements interaction
- The Sword2 endpoint might be too slow to effectively restore the entire ORA4 Review system. In order to provide an effective disaster recovery option, the system would have to be capable of processing 21 objects a minute (6 days for recovery at ORA3 size)
- A JSON-based import mechanism is being developed for ORA3-ORA4 migration (and speed-tuned for complete import of all ORA content) and this could be used instead.
- The code to represent ORA4 objects as JSON can only import JSON, not export it. We would need to write additional code to express ORA4 objects as JSON

Option 1: ora-ocfl-worker + ocfld

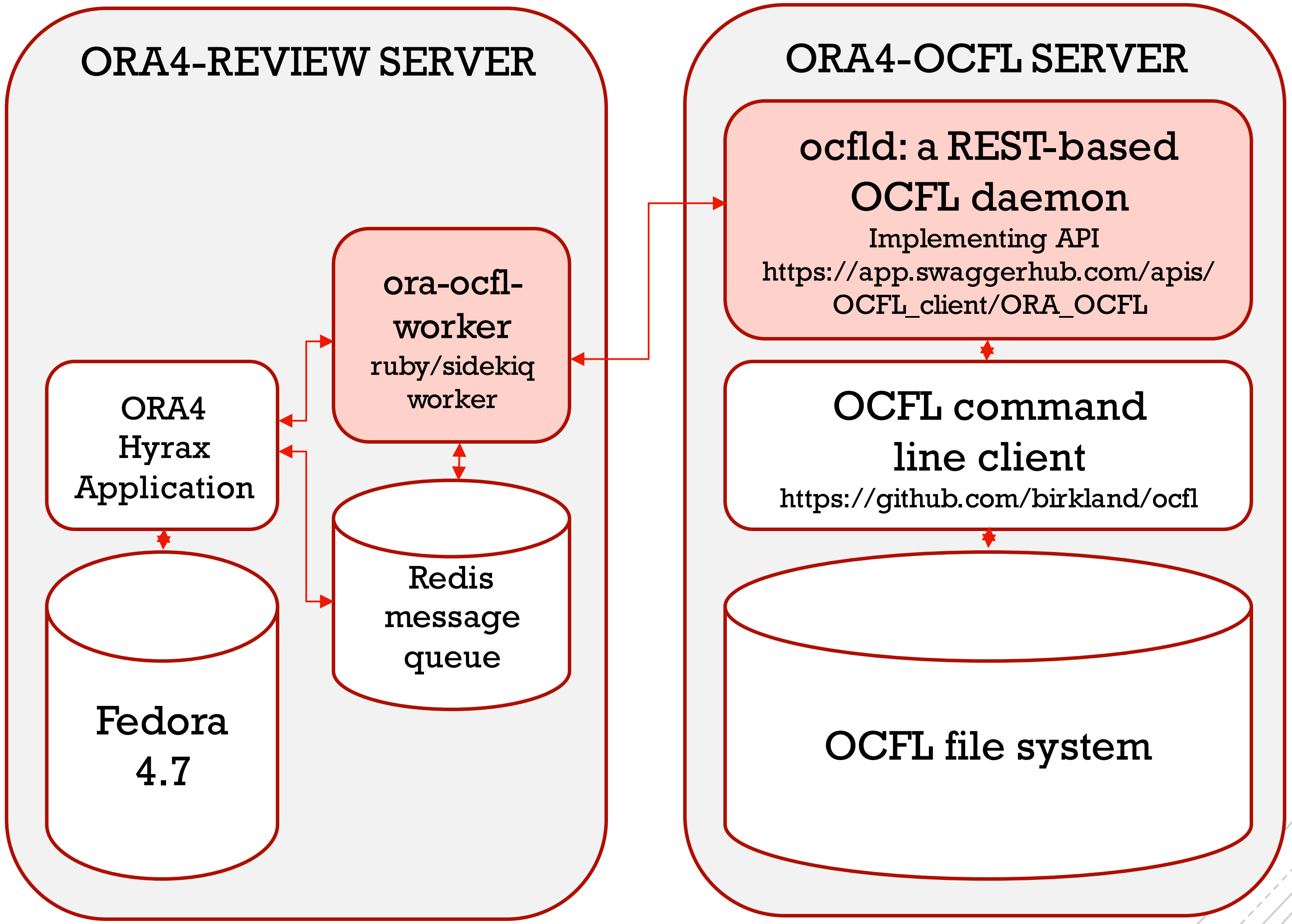
Option 1 envisages the following components:

- ora-ocfl-worker
  - Ruby Sidekiq worker
  - Tasks taken from a Redis queue
  - Reads ORA objects from Sword2 or Hyrax for ingest into ORA4 DPS via ocfld
  - Reads objects from ORA4 DPS via ocfld for ingest into Sword2 or Hyrax
- ocfld
  - REST-based client implementing an OCFL API
  - Wrapper code around command line OCFL clients

Architecture overview (option 1a)



Architecture overview (option 1b)



Benefits of Options 1a+b

- A REST endpoint allows the OCFL system to be on a completely separate server – this allows for considerable performance optimisation
- The REST endpoint would provide access control to the OCFL store, this is not provided by Option 2.
- The ora-ocfl-worker – the only ORA specific code – would be a relatively thin application.
- The OCFL REST endpoint should enable a relatively easy architectural swap with any Fedora6 OCFL based application.

Concerns with Options 1a+b

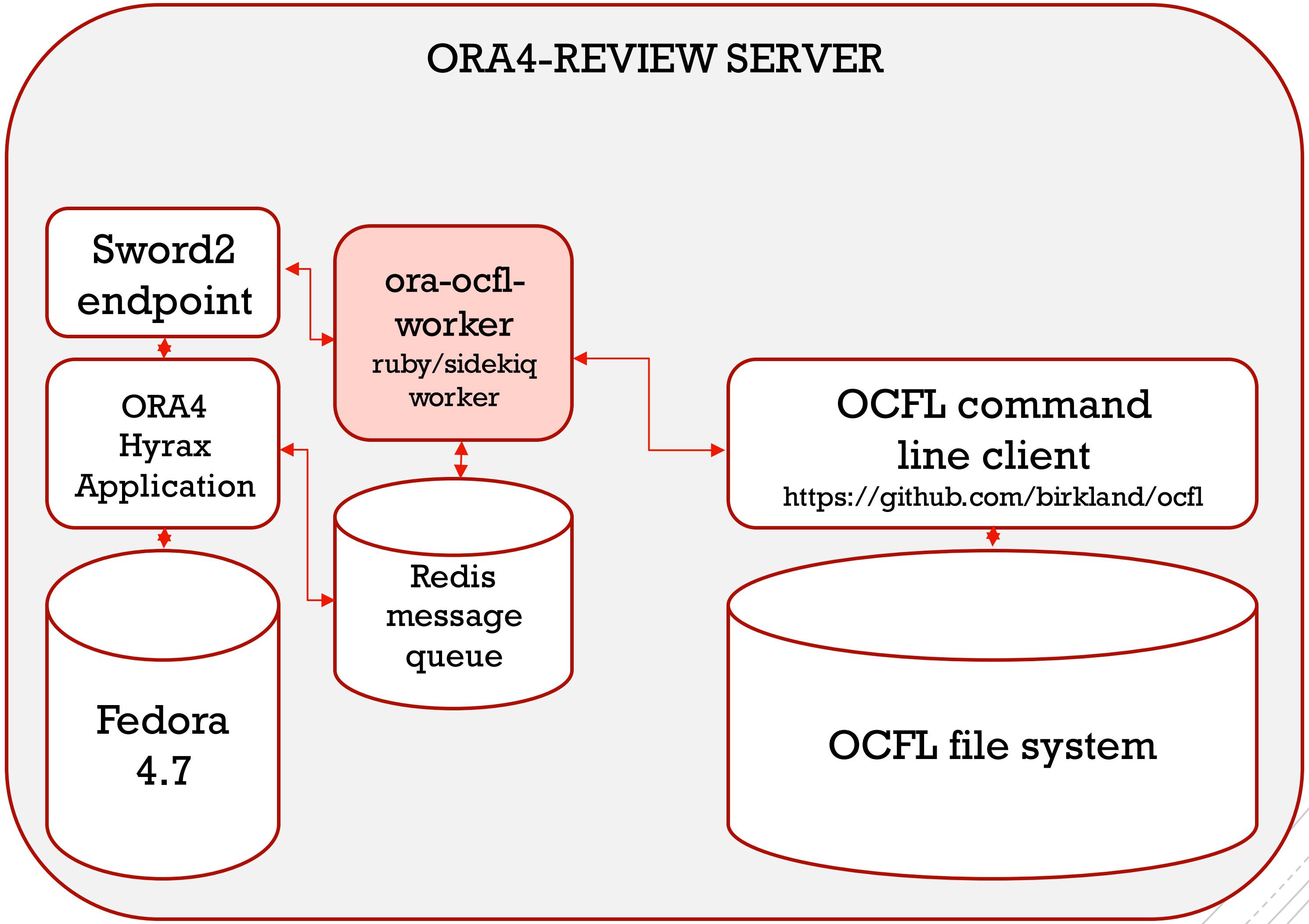
- The API is only in an early development stage
- Complexity: writing an OCFL daemon and a worker involves extra work
- Complexity: I know this is a repeat, but it's such a big issue that it's worth mentioning twice...

Option 2: ora-ocfl-worker

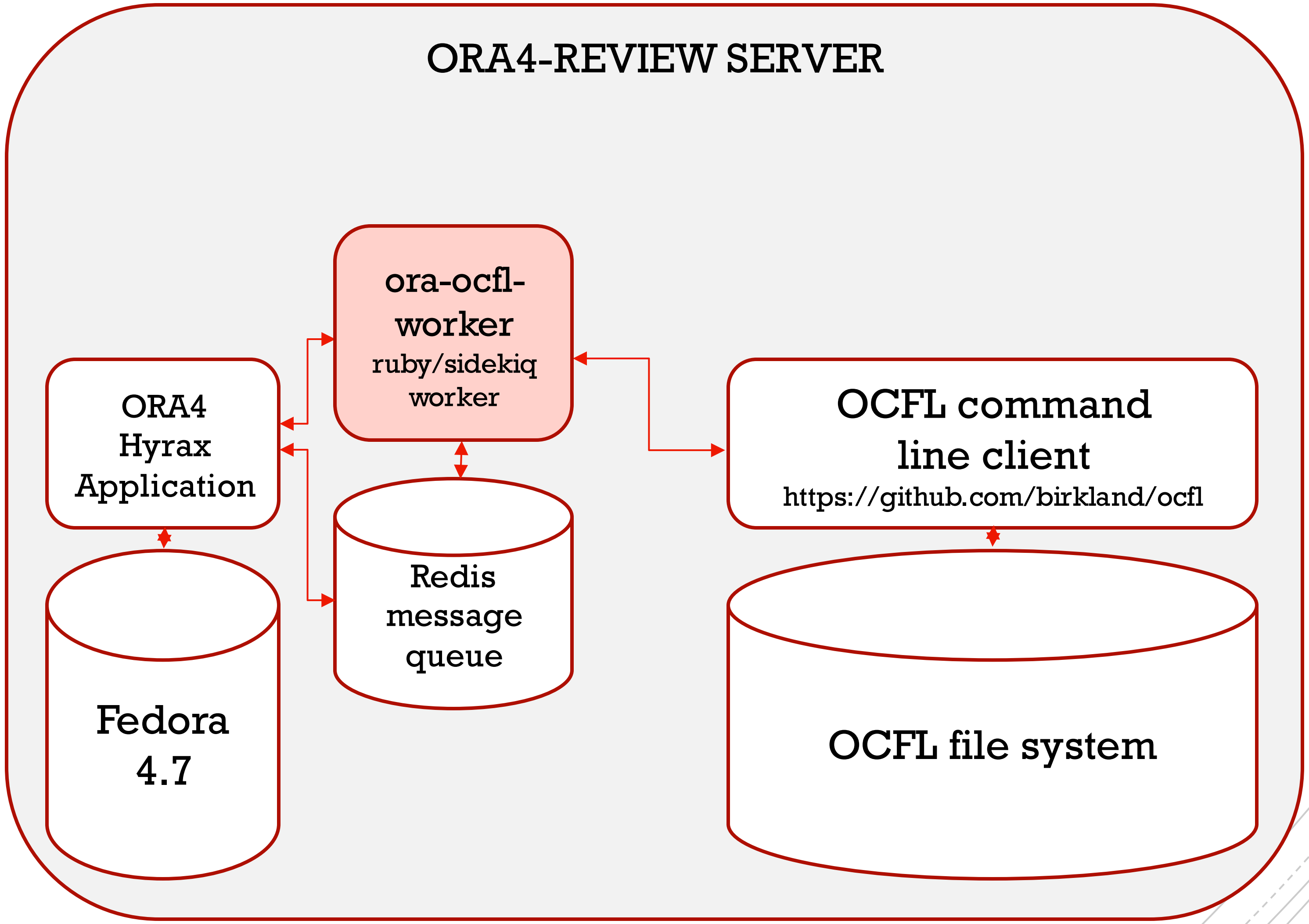
Option 2 envisages only one component:

- ora-ocfl-worker
  - Ruby Sidekiq worker
  - Tasks taken from a redis queue
  - Wrapper code around OCFL command line client
  - Reads ORA objects from Sword2 or Hyrax for ingest into ORA4 DPS via command line client
  - Reads objects from command line client for ingest into Sword2 or Hyrax

Architecture overview (option 2a)



Architecture overview (option 2b)



ORA4-REVIEW SERVER
ORA4 Hyrax Application
ora-ocfl-worker ruby/sidekiq worker
OCFL command line client https://github.com/birkland/ocfl
Fedora 4.7
Redis message queue
OCFL file system

Benefits of Options 2a+b

- Simpler than Option 1, requiring less code
- Only one ORA-developed component, rather than two
- Speed – the Go client is fast, so this method should be faster than adding an additional REST layer

Concerns with Options 2a+b

- No access/control authorisation layer between Review users and the ORA4 DPS
- Places high load on an already taxed server: ORA4 Review
- The ora-ocfl-worker would be a sizeable application, translating ORA4 concepts into OCFL concepts

Large file deposit and storage

- Currently, the Hyrax application has no mechanism for ingesting files larger than 2gb
- We will need a separate ingest mechanism for large files, and to store them outside of Fedora4
- If we have a separate ingest mechanism for large files, there's no reason we couldn't use it for ALL binary files
- The OCFL DPS could be the only location binary files are stored in the system
- Fedora could link to these files as external content, see https://wiki.duraspace.org/display/FEDORA5x/External+Content
- If we were to do this, then a REST API that provided an addressable URL for each binary files would be extremely useful (the command line clients cannot do this currently)

User stories

The following user stories are broken down by role.

- AS a repository manager
- AS an import/export process
- AS a digital preservation process
- AS a Sword2 OCFL connector

The numbers following the user stories refer to issue tickets at https://github.com/tomwrobel/ora_ocfl/issues/

User stories: as a repository manager

- As a respository manager I want to see the history of an object, including who was responsible for changes and when those changes occurred so that I can ensure that repository policies are being followed (#4)
- As a repository manager I want to be able to delete a file completely from an object so that I can comply with external requirements for data security and privacy (#5)
- As a repository manager I want my code to interact with an OCFL instance on a separate server so that I can move OCFL-based content between servers (#9, related to #20)
- As a repository manager I want to get a list of an object's contents for a given version so that I can provide a verifiable record of the object's state at that point in time (#10)
- As a respository manager I want to be able to get a list of files that have changed since a given version so I can measure the rate of change in the repository (#11)
- As a repository manager I want to be able to read and write 100 objects with 4mb of total size a minute to provide my system with an effective disaster recovery solution (#12)
- As a repository manager I want to be able to read and write 100 objects with 4mb of total size a minute so that I can re-export records when I update the data model (#12)
- As a repository manager I want my OCFL objects to be protected against unauthorised access and modification (#17)
- As a repository manager, I would like the ORA OCFL client to interact with Hyrax via the Sword2 interface created for Symplectic Elements Repotool 2 so that I can re-use existing tested code for crosswalking metadata and ingesting objects (#18)
- As a repository manager, I want to release any code I create as open source so that I can participate in the OCFL and Hyrax/Samvera communities (#19)
- As a repository manager I would like to interact with the OCFL layer via REST so that the OCFL system can be interacted with over a network to promote community acceptance and for better fit with the rest of the ORA4 stack (#20, related to #9)
- As a repository manager I want the system to record any digital preservation actions taken by Hyrax (#21, related to #7 and #16)
- As a repository manager I want my import export process to read a job from a queue so that I can import/export asynchronously (#23)
- AS a repository manager I want to be able to ensure that an object that should have been exported or imported has been fully exported or imported so I can verify that my system is working as expected (#25)

User stories: as an import/export process

- As an import/export process I want to be able to remove a file from an object so that the object can have the correct files, and only the correct files, in its current version (#1)
- As an import/export process I want to be able to access all of a file's properties knowing only its object ID and filename so I can interact with it without knowing where it is stored on disk in advance (#2)
- As an import export process I want to get a list of all files in a given version of an object and their checksums so that I can compare the stored object with another version of the object in a different system (#3)
- As an import/export process I want to get a checksum of all files associated with a version of an object so that I can see if I need to add a specific file to the OCFL object (#3)
- As an import/export process I want to add files to a previously saved object so that I can update it (#13)
- As an import/export process I want to be able to create objects in the OCFL system and read them back, so that I can store and retrieve objects (#14)
- As an import export process I want to generate a new version of an object as a BagIt bag for import into Hyrax (#15)
- As an import/export process I want to record the digital preservation actions performed on object/file ingest so that I can provide this information to downstream digital preservation agents (#16, related to #21)
- As an import/export process I want to be able to migrate objects between Hyrax and OCFL by using the Sword2 interface so that I can use existing code wherever possible (#18)
- As an import/export process I want to be able to detect and recover from failures in object processing so that I can ensure that objects are correctly represented in both Hyrax and OCFL (#27)

User stories: as a digital preservation process

- As a digital preservation process I want to be able to access all of a file's properties knowing only its object ID and filename so I can interact with it without knowing where it is stored on disk in advance (#2)
- As a digital preservation process I want to get a checksum of all files associated with a version of an object so that I can see if any records have changed (#3)
- As a digital preservation process I want to be able to validate the OCFL version of a given object by ID so that I know that the OCFL store is valid (#6)
- As a digital preservation process I want to be able to record a PREMIS action on an object so that significant preservation metadata can be logged with the object (#7, related to #21)
- As a digital preservation process I want to be able to read the PREMIS history of an object so I can generate a history of the object's digital preservation (#8)

User stories: as a Sword2 OCFL connector

- AS a Sword2 OCFL connector I want to read an object (metadata and binary files) from the OCFL layer so I can restore it into Hyrax (#18)
- AS a Sword2 OCFL connector I want to add a binary file and its metadata to an object in order to back it up to the OCFL layer (#18)
- AS a Sword2 OCFL connector I want to save an entire object in order to back it up to the OCFL layer (#18)
- AS a Sword2 OCFL connector I want to save a new version of an object in order to back it up to the OCFL layer (#18)
- AS a Sword2 OCFL connector I only want to update binary files if they need updating (in either Hyrax or Sword2) (#24)
- As a Sword2 OCFL connector I want to ensure that objects are protected against concurrent writes so that I can ensure that objects remain valid and in an expected state (#26)
- As a Sword2/OCFL connector I want to be able to detect system and object transfer failures and report them correctly so that problems are identified and reported quickly (#27)