Building a performant and accessible replacement for CONTENTdm using Valkyrie

Adam Wead • awead@psu.edu • amsterdamos • awead

Samvera Connect - October 11, 2018

PennS

Overview

1. Vital Statistics: Plan for replacing CONTENTdm with CHO
2. Using Valkyrie: Why we chose it and its impact so far
3. Building CHO: What we've done so far

Vital Statistics

Whats, whos, hows, and whys

CHO: Cultural Heritage Object (Repository)

Nathan Tallman, product owner

Adam Wead, technical lead

Carolyn Cole, developer

Michael Tribone, user-interface designer

+ Numerous stakeholders!

The Big Picture

- Export from CONTENTdm
- Remediate metadata in csv using OpenRefine
- Import collections into CHO via bags and csv
- Further metadata work, adding new content
- Export updated bags and metadata for preservation

PennS

Schedule

- Started October 25, 2017
- February 2018: 6 weeks of CHO + 4 weeks of Scholarsphere
- MVP[1] mini-releases after each 6-week cycle (three sprints)
- Currently working on MVP 4 of 7 (there will be more)
- complete MVP by fall 2019[2]
- first production release in 2020

1. MVP = minimum viable product
2. https://github.com/psu-libraries/cho/milestones

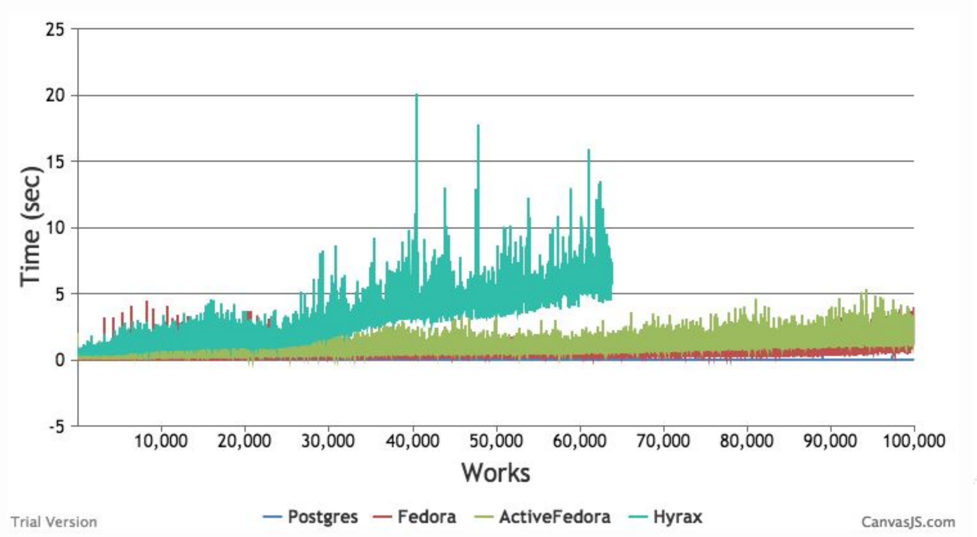
Using Valkyrie

The metrics told us we should.

Performance Limitations in Hyrax (2017)

- Penn State has collections with 385K+ items
- Hyrax 1.0 was unable to support this number
- **This includes the updated Solr configuration**[1]
- Using Postgres, Valkyrie proved to be more performant

Comparing Valkyrie and Hyrax

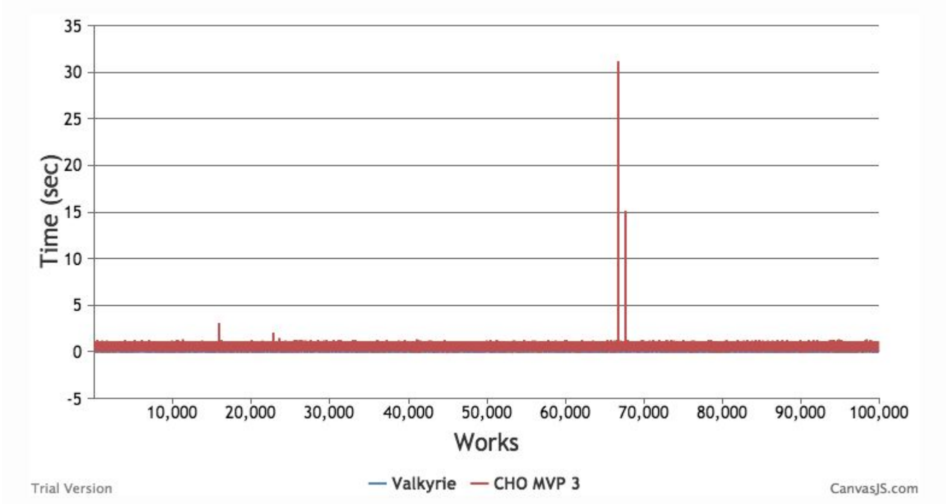


1. http://awead.github.io/fedora-tests

Comparing CHO: Then vs. Now

- CHO lags behind our original data sample
- Time per work is flat for both
- Original Valkyrie sample took 7 minutes
- CHO took 43 minutes
- Network latency could be an issue: local Solr vs. VM

Valkyrie versus CHO MVP 3[1]



1. http://awead.github.io/cho-benchmarks

Opportunities

- New stuff: functional Ruby, dry-ruby, transactions
- Re-envisioning code and practices
- Collaborative sprints with Princeton
    - Single-valued attributes
    - Optimistic locking
- **LOOKING FOR MORE ADOPTERS!!!!**

Challenges

- No Hyrax "freebies"
  - UI
  - PCDM Modeling
  - Derivative generation
  - Characterization
- Dynamic property definitions
- IIIF integration and Universal Viewer

Choosing to confront problems we know we can solve versus problems we do not, or cannot solve.

Building CHO

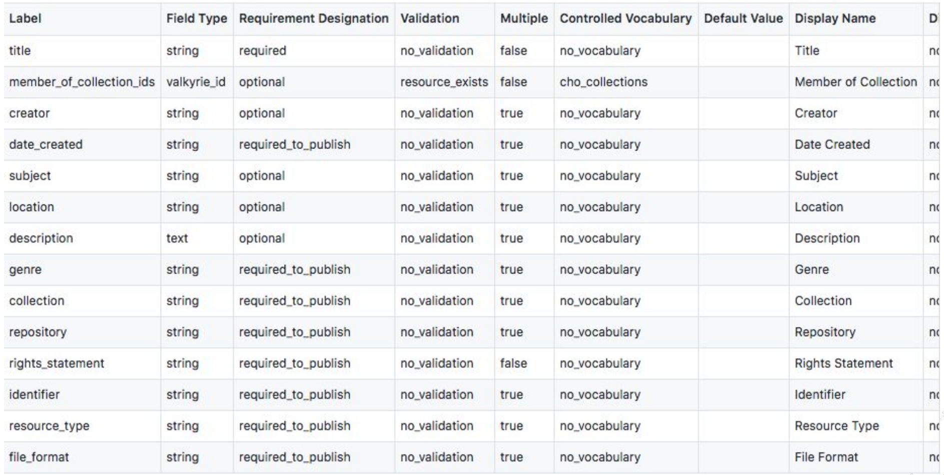
The CSV "API"

- Every component has a csv interface
- Updating and creating collections and works via csv import and export
- Creating works with files from bags
- Defining the properties on resources
- Defining a property's behaviors
    - Controlled vocabulary
    - Validation
    - Transformation
    - Default values

The Data Dictionary

- Metadata specialists define fields characteristics
- Seeded into CHO as Valkyrie resources
- Dynamically assigned to application resources (works, collections, etc.)
- Selectively applied to all resource, or work subtypes

Dictionary CSV File



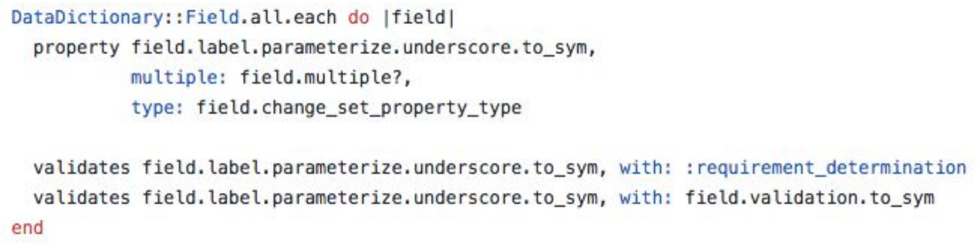
Label	Field Type	Requirement Designation	Validation	Multiple	Controlled Vocabulary	Default Value	Display Name	D
title	string	required	no_validation	false	no_vocabulary		Title	n
member_of_collection_ids	valkyrie_id	optional	resource_exists	false	cho_collections		Member of Collection	n
creator	string	optional	no_validation	true	no_vocabulary		Creator	n
date_created	string	required_to_publish	no_validation	true	no_vocabulary		Date Created	n
subject	string	optional	no_validation	true	no_vocabulary		Subject	n
location	string	optional	no_validation	true	no_vocabulary		Location	n
description	text	optional	no_validation	true	no_vocabulary		Description	n
genre	string	required_to_publish	no_validation	true	no_vocabulary		Genre	n
collection	string	required_to_publish	no_validation	true	no_vocabulary		Collection	n
repository	string	required_to_publish	no_validation	true	no_vocabulary		Repository	n
rights_statement	string	required_to_publish	no_validation	false	no_vocabulary		Rights Statement	n
identifier	string	required_to_publish	no_validation	true	no_vocabulary		Identifier	n
resource_type	string	required_to_publish	no_validation	true	no_vocabulary		Resource Type	n
file_format	string	required_to_publish	no_validation	true	no_vocabulary		File Format	n

https://github.com/psu-libraries/cho/blob/master/config/data_dictionary/data_dictionary_production.csv

Dynamic Field Definitions

- Everything is defined on resources, change sets, and SolrDocument
- Schemas "filter" fields for editing/display
- Loaded at runtime
- Eventually will be changeable in a live application
- Still WIP!

Loading fields on a change set:



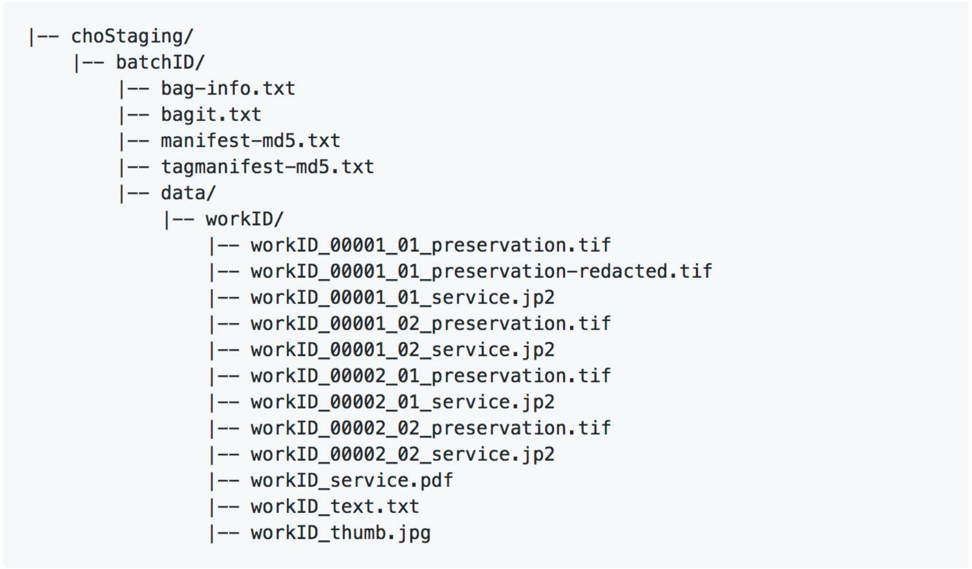
```
DataDictionary::Field.all.each do |field|
  property field.label.parameterize.underscore.to_sym,
           multiple: field.multiple?,
           type: field.change_set_property_type

  validates field.label.parameterize.underscore.to_sym, with: :requirement_determination
  validates field.label.parameterize.underscore.to_sym, with: field.validation.to_sym
end
```

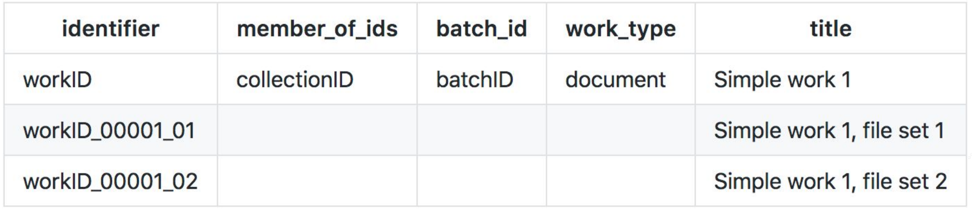
https://github.com/psu-libraries/cho/blob/master/app/cho/data_dictionary/fields_for_change_set.rb

Bags

- Uploaded as a zip
- Validate structure and integrity
- Create multiple works with one or more file sets
- Generate derivatives as needed
- Determines file set use
- CSV supplies the metadata



```
|-- choStaging/
    |-- batchID/
        |-- bag-info.txt
        |-- bagit.txt
        |-- manifest-md5.txt
        |-- tagmanifest-md5.txt
        |-- data/
            |-- workID/
                |-- workID_00001_01_preservation.tif
                |-- workID_00001_01_preservation-redacted.tif
                |-- workID_00001_01_service.jp2
                |-- workID_00001_02_preservation.tif
                |-- workID_00001_02_service.jp2
                |-- workID_00002_01_preservation.tif
                |-- workID_00002_01_service.jp2
                |-- workID_00002_02_preservation.tif
                |-- workID_00002_02_service.jp2
                |-- workID_service.pdf
                |-- workID_text.txt
                |-- workID_thumb.jpg
```



identifier	member_of_ids	batch_id	work_type	title
workID	collectionID	batchID	document	Simple work 1
workID_00001_01				Simple work 1, file set 1
workID_00001_02				Simple work 1, file set 2

https://github.com/psu-libraries/cho/wiki/File-Specifications#simple-work-folder-of-manuscript-materials-mvp

Benchmarks

- Benchmarking is built into the codebase
- Simulate collections of any size
- Randomized metadata using the Faker gem
- Random binary files to simulate storage
- Run as rake tasks
- Reports individual creation times for use in charts and graphs
- Total time used to gauge performance impact

Benefits of Benchmarking

- Measure performance results after each MVP mini-release
- Identify performance impacts early
- Avoid bad architecture or code decisions
- Creates a complete feedback loop from coding to release
- Identify devops needs early
- Ex: IVP6 firewall issue was impacting performance in MVP 2

Accessibility First

- Penn State Policy UL-AD15[1] requires WCAG 2.0[2] AA compliance
- Approved in in 2005
- WCAG 2.1[3] published in June 2018
- No Javascript (for now)
- Maximize client-side HTML5, ex. datalist elements for selects
- Ensure we are meeting standards with each release
- Manual tests involving
  - WAVE accessibility toolkit
  - Keyboard navigation
  - Screen-reader integration with JAWS and MacOS Voiceover

1. https://libraries.psu.edu/policies/ul-ad15
2. Web Content Accessibility Guidelines 2.0 https://www.w3.org/TR/WCAG20/
3. Web Content Accessibility Guidelines 2.1 https://www.w3.org/TR/WCAG21/

PennS

No Active Record

- Valkyrie resources are used throughout
- AR used only for gem dependencies such as Devise
- Consistency: all resources have the same interface
- Why support multiple database abstractions in the same application?

Walking the Path...

- When deciding on change in code, dependency, technique, or practice, take each decision to its complete conclusion
- Sometimes things start to look worse before they can look better
- Not making a choice is itself a choice
- Example: Webpacker in Rails
    - Tested React, Angular, Elm, Vue
    - Ultimately decided "none"
- Require accessible interfaces using standard HTML5
- Leverage Javascript via progressive enhancement

Questions?

Thank You!

Thanks to the Samvera Connect 2018 committee, the University of Utah, and everyone else who made the Connect 2018 conference possible.

Special thanks to Penn State University Libraries and my team at DSRD.

Shout outs to the Princeton dev team!

PennS

Notes and Links

Valkyrie: https://github.com/samvera-labs/valkyrie

CHO: https://github.com/psu-libraries/cho

Samvera Connect 2017 Talk: http://awead.github.io/presentations/fedora-tests

Valkyrie Performance Testing: http://awead.github.io/fedora-tests

CHO Performance Testing: http://awead.github.io/cho-benchmarks

Dry Ruby: https://dry-rb.org/

Penn State Policy UL-AD15 on Web Accessibility: https://libraries.psu.edu/policies/ul-ad15

PennS