Monday, January 28, 2019

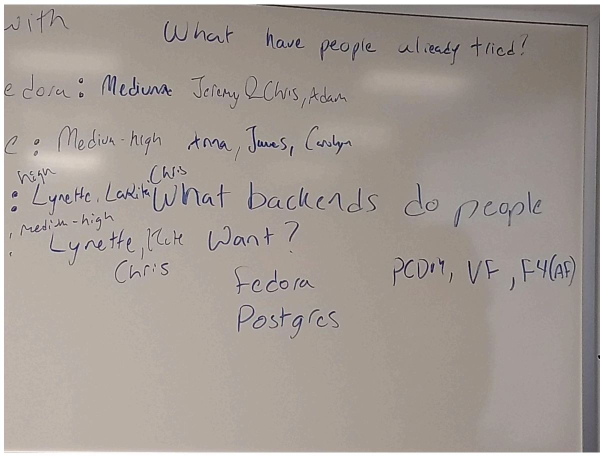
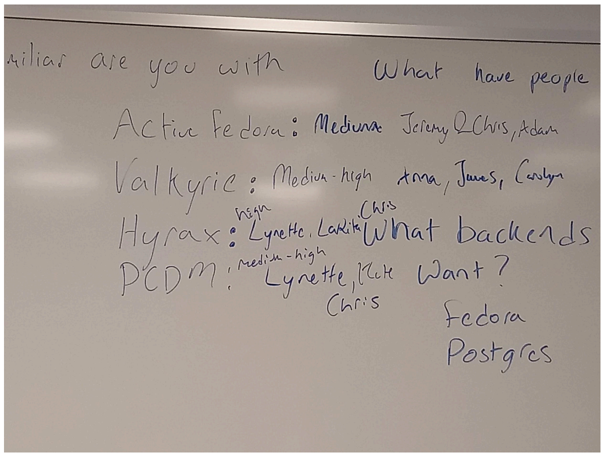
Attendees:

- Chris Colvard
- Josh Gum
- Michael Tribone
- Dan Coughlin
- Kate Lynch
- Carolynn Cole
- Jeremy Friesen
- LaRita Robinson
- Anna Headley
- James Griffin
- Lynette Rayle
- Tom Johnson (not attending first day due to travel issues)

Morning Plenary

Expertise in the Room:

- ActiveFedora - Medium -- Jeremy, Chris, Adam
- Valkyrie - Medium High -- Anna, James, Carolyn
- Hyrax - High -- Lynette, LaRita, Chris
- PCDM - Medium High -- Lynette, Kate, Chris

 with What have people already tried? edora: Medium Jeremy & Chris, Adam e: Medium-high Anna, James, Carolyn high: Lynette, LaRita Chris What backends do people Want? medium-high Lynette, Kate Chris Fedora Postgres PCDM, VF, F4(AF)	 miliar are you with What have people Active Fedora: Medium Jeremy & Chris, Adam Valkyrie: Medium-high Anna, James, Carolyn Hyrax: high Lynette, LaRita Chris What backends PCDM: medium-high Lynette, Kate Chris Want? Fedora Postgres

Cases Involving Hyrax/Valkyrie

- Penn State University
  - Very concerned regarding the future support and life of ScholarSphere
- Cornell
  - Has performance issues in terms of using ActiveFedora to support repository assets
- Princeton
  - Anna Headley of Princeton has worked with Hyrax primarily before she took her current position
  - Has a balance of experience in working with both Princeton's Valkyrie (only) repository and Hyrax at the Science History Institute
- Notre Dame
  - Still remains invested in Fedora 3 for their data persistence
  - Places an emphasis upon the need to ensure that Hyrax continue to be used moving forward
  - Less opinionated on what exactly needs to be in place for persistence
  - Fedora 4 does not have a preservation option, but this is not currently available
  - Notes that making it feasible to use Fedora has its appeal….given that many are going to need to support investing in it from an administrative standpoint
- U. Penn
  - Too much data for Fedora 4 to remain viable, performance concerns
  - Seeks to use Valkyrie backend in production
- IU
  - Seeking to continue to invest in moving ahead with the Avalon Project
  - IU will continue to use Fedora backend; backend choice for Avalon implementers
- OSU
  - Actively developing new hydra heads, need to ensure that Valkyrie can assist with this for scalable storage

Familiarity with Persistence Layers and Backends

- ActiveFedora (Medium): Jeremy, Chris, Adam
- Valkyrie (Medium - High): Anna, James, Carolyn
- Hyrax (High): Lynette, LaRita, Chris
- PCDM (Medium - High): Lynette, Kate, Chris

What was decided at Samvera Connect?

- The need for a survey to understand community needs and desires.
- The need for a group to meet at a dev congress to make initial decisions on the path forward in a way that meets the community desires.

Conceptualizing the Work

Migration	Code	Data	Documentation
Fully Backwards Compatible	JG18 (ActiveFedora delegation-based approach to Valkyrie support)	**PS19 (This Sprint)**	
…	CAC19	CAC19	
Not Compatible/No Migration Strategy	JC/CJC17 (Exploratory attempt at Hyrax/Valkyrie integration)	JG18 JC/CJC17? JF19	JC/CJC17?

- This will assist in assessing the amount of effort involved in each actionable task
- Example: Valkyrie Resource models vs. ActiveFedora Models
- We may not be well-resourced enough to address documentation thoroughly
    - All agree to this
- In terms of a full migration package, we do still need to address this

What has been tried?

Valkyrie sprint of Fall 2017 (JC17)

Tinkering with AF::Base as a Valkyrie::Resource, goal to make the test suite green. Changed lots of code, no consideration for migration. Writing data to the memory adapter. Was happening concurrent with Collections Extension.

- Ended with single digit number of tests not working, but no idea if it was functional as an app.
- UI may not have been fired up and tested
- Exploratory effort, what was a good first pass
- There are interesting and valuable insights into the branch
- Collection extension sprints (with lots of code-change) ran concurrent w/ this sprint. Lots of changes

One Week Experiment prior to Samvera Connect 2018 (JG18)

Override ActiveFedora#save delegate to Valkyrie. Was able to get data persisted. However, tackling the ActiveFedora API was going to be large

One Week Experiment from Penn State (CAC19)

Is there a way to piece in Valkyrie while still having the code work with minimal layers. Wrapped Valkyrie Resource around ActiveFedora; Could read/write from ActiveFedora or read/write Valkyrie::Resource. Created a Valkyrie persister that just uses ActiveFedora. Creating for each individual model was "easy" but the relationship linking between two models was more challenging. There is a "do not merge" PR - https://github.com/psu-stewardship/scholarsphere/pull/1515/files

- All of the tests passed, and Scholarsphere still worked
- Needed to change the Rails oft-default mysql database to PostgresQL
- People will eventually have to write custom mapping, and we would need to document that
- The PCDM mapping is something that may be a challenge, but that challenge is not exposed to those that adopt Hyrax (but instead those that maintain Hyrax)
- For vanilla Hyrax we may be able to write most of the code

Active Fedora Life Cycle

Eventually, ActiveFedora needs to be replaced. At some point we will have a long-term support contract. Referencing Branch by Abstraction approach.

Some want to migrate away from ActiveFedora. Others either do not, or do not want to right away. Possible paths forward for ActiveFedora...

High desire Not a high priority	● Keep data structure in Fedora ● Wrap Valkyrie with AF API (limit code migration proposed by JF) ● Keep API of AF but improve performance with Valkyrie ● Wrap AF with Valkyrie (similar to CAM19, limits data migration) ● ... ● Keep code in ActiveFedora

- Is there a way to flip the dependency of this?
- ActiveFedora provides methods for model decoration...but can we gut it entirely and ensure that just calls Valkyrie ChangeSetPersister methods?
  - Otherwise, keeping ActiveFedora is a rough commitment
  - ActiveFedora just wraps Valkyrie
- Which is more important, the data migration or the code migration?

- If community adopters had to choose one, which is the higher priority?
- Carolyn's approach can be implemented in a reasonable time frame
  - Other approaches...can these be addressed as quickly?
- One of the challenges is...what does a proper Fedora object look like across Fedora 4 and Samvera releases?
  - It hasn't been all that consistent
- Hyrax WG should look to build upon the work which is undertaken this week
- Conclusion: ActiveFedora (the code base) will continue to live in Hyrax (for the next immediate releases) as Valkyrie is injected
  - This will ensure backwards-compatibility with the data
- Note that there are also some queries which need to be supported within Valkyrie
  - Collections management features will require these of Valkyrie
  - Also, adopters may well be using ActiveFedora directly for queries in custom implementations
  - We need to ensure that these do not break
- As an assumption here is that we favor data over code
  - We believe we can be more flexible on ActiveFedora code-base overrides as we believe there is less interactions

Assumptions

- People will eventually have to write some custom mapping code for their specific models/relationships.
- Active Fedora (the code base) will live past this sprint. This supports full backward compatibility of data
- Favor backward compatibility of data over code

Success

- Data can stay the same
- Code in Hyrax is backwards compatible (as much as possible)
- There is a documented pattern for upgrading generated hyrax code or other document patterns such as metadata overrides
- Regular commits to master that maximize backwards compatibility
- Create a solid design plan for how to move forward with implementation

Sticky Parts

- Relationships
- ActiveFedora calls (quantity in Hyrax)
- Queries thru ActiveFedora
- File Versioning
- ActiveFedora support of Fedora and Solr

- Decision about Valkyrie ChangeSets and Forms interactions
- Backwards compatibility of data and code will be hard
- Automated tests will remain slow until we can use in-memory adapter

Afternoon Breakouts

Explore Code Migration

Summary:

4 options are listed below. We decided to do an experiment with approach #3 based on CAC's work last week. We split into two groups. One focused on making similar changes to models in Hyrax. The other group explored moving some of the changes down into ActiveFedora.

4 Options:
#0 Basic Branch by Abstraction

- See white board image for #0 and #1 below.
- AF wraps Valkyrie, possibly an AF branch/gem
    - Pros
        - Almost no code changes
        - AF continues to work as is (allows for deprecation)
    - Cons
        - Two gems to maintain

#1 AF wraps Valkyrie (JG18, JF19)

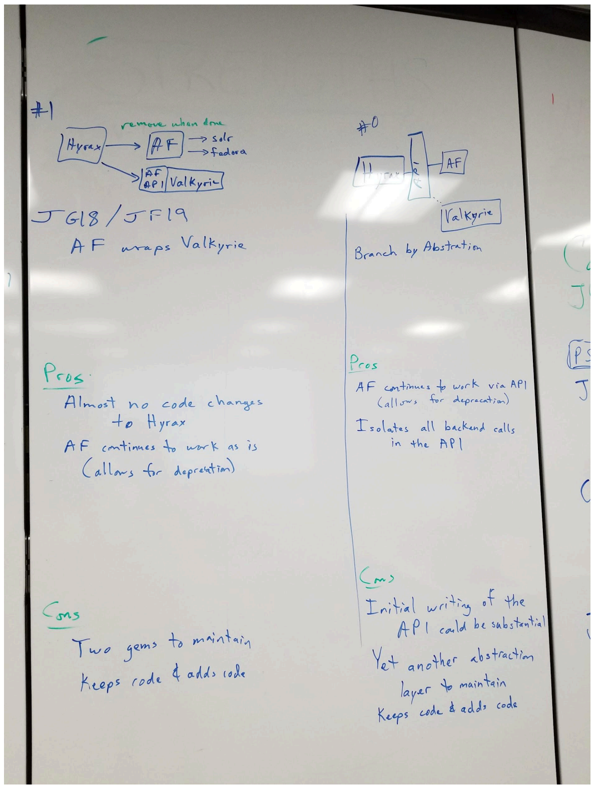
- See white board image for #0 and #1 below.
- Branch by Abstraction : API layer in front of AF that eventually supports Valkyrie
    - Pros
        - AF continues to work via API (allows for deprecation)
    - Cons
        - Initial writing of the API could be substantial
        - Yet another abstraction layer to maintain
        - Keeps and adds more code

#2 Change Everything - Non-incremental (JC17 - valkyrie branch in Hyrax)

- See white board image for #2 and #3 below.
- Change everything; Valkyrie replaces AF, non-incremental
    - Pros
        - 80%+ done, could be used to inform another effort on top of master

- Cons
    - Breaks everything initially, big code migration
    - Lots of code changes
    - Significant rebase / rework to catch up to master

#3 Valkyrie wraps AF (CAC19)
- See white board image for #2 and #3 below.
- Wrapped AF Model, transitional eventually replaced with Valkyrie Adapter
    - Pros
        - Pure implementation of Valkyrie (datamapper)
        - Path to incremental commits
        - AF continues to work as is (allows for deprecation)
        - Moves Hyrax to to use DRY which could simplify code
        - Removes code
    - Cons
        - Integrating datamapper pattern to Hyrax, replaces direct calls to AF with direct calls to Valkyrie
        - Valkyrie to AF process not pure Valkyrie adapter



#1
Hyrax
AF
solr
fedora
remove when done
AF API
Valkyrie
#0
Hyrax
AF
Valkyrie

JG18 / JF19
AF wraps Valkyrie

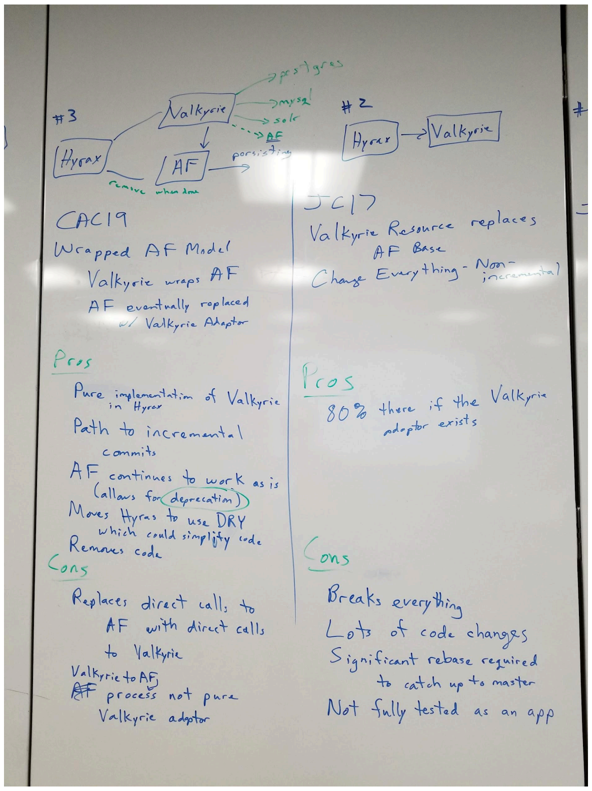
Pros
Almost no code changes to Hyrax
AF continues to work as is (allows for deprecation)

Cons
Two gems to maintain
Keeps code & adds code

Branch by Abstration

Pros
AF continues to work via API (allows for deprecation)
Isolates all backend calls in the API

Cons
Initial writing of the API could be substantial
Yet another abstraction layer to maintain
Keeps code & adds code



#3
Hyrax
Valkyrie
postgres
mysql
solr
AF
AF
persisting
remove when done
#2
Hyrax
Valkyrie

CAC19
Wrapped AF Model
Valkyrie wraps AF
AF eventually replaced w/ Valkyrie Adaptor

Pros
Pure implementation of Valkyrie in Hyrax
Path to incremental commits
AF continues to work as is (allows for deprecation)
Moves Hyras to use DRY which could simplify code
Removes code

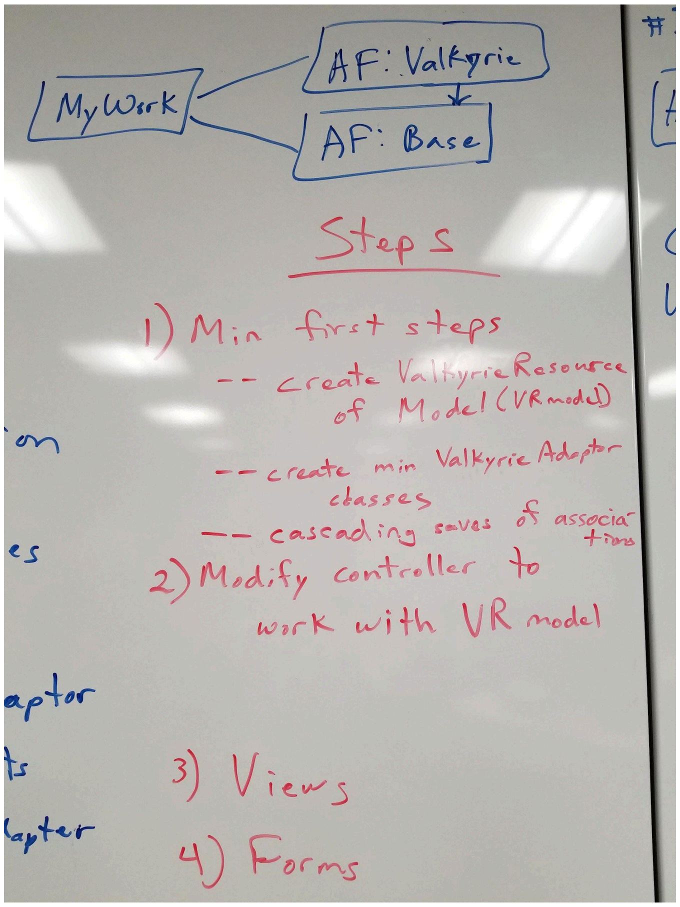
Cons
Replaces direct calls to AF with direct calls to Valkyrie
Valkyrie to AF, ~~AF~~ process not pure Valkyrie adaptor

JC17
Valkyrie Resource replaces AF Base
Change Everything - Non-incremental

Pros
80% there if the Valkyrie adaptor exists

Cons
Breaks everything
Lots of code changes
Significant rebase required to catch up to master
Not fully tested as an app

Incremental steps for implementing option #3.


MyWork
AF: Valkyrie
AF: Base

Steps

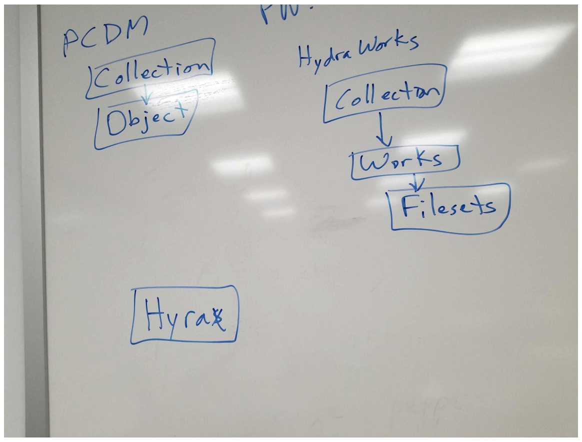
1) Min first steps
-- create ValkyrieResource of Model (VR model)
-- create min Valkyrie Adaptor classes
-- cascading saves of associations

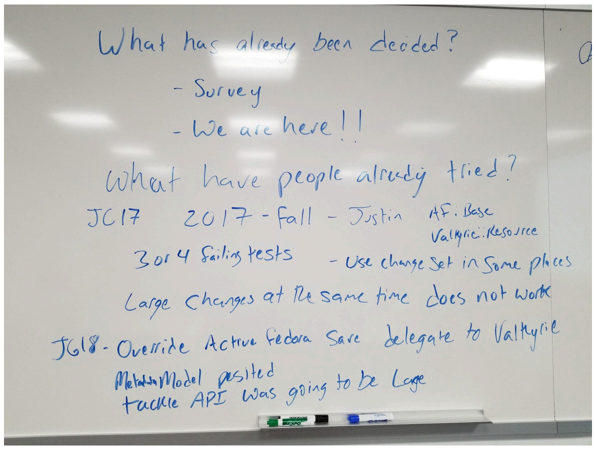
2) Modify controller to work with VR model

3) Views

4) Forms

Whiteboards at the end of the day




What has already been decided?

- Survey
- We are here!!

What have people already tried?

JC17 2017 - Fall - Justin AF::Base Valkyrie::Resource

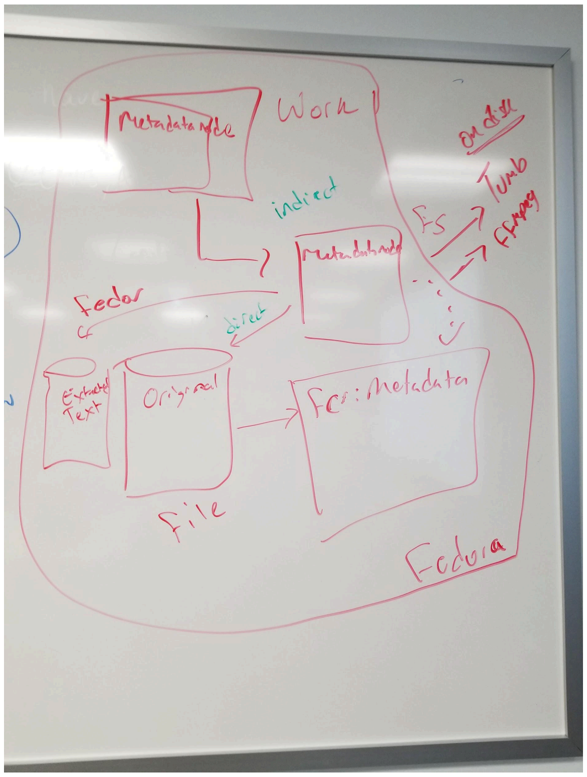
3 or 4 failing tests - Use change set in some places

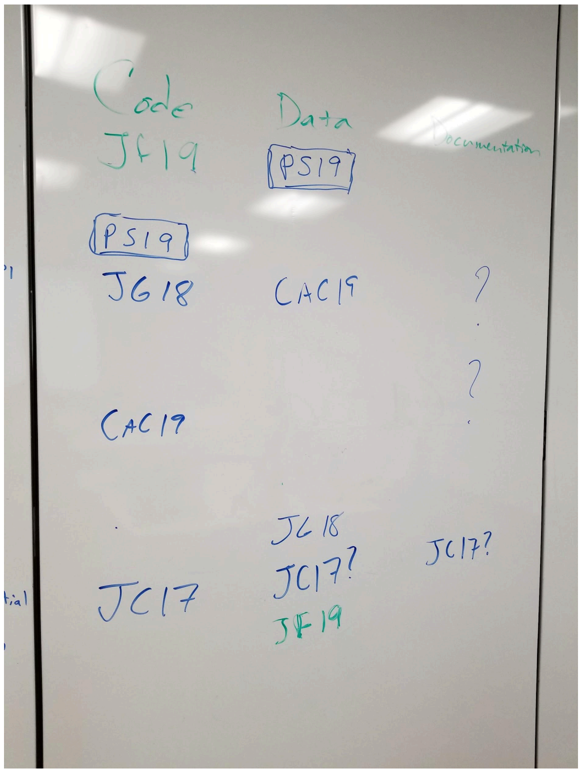
Large changes at the same time does not work

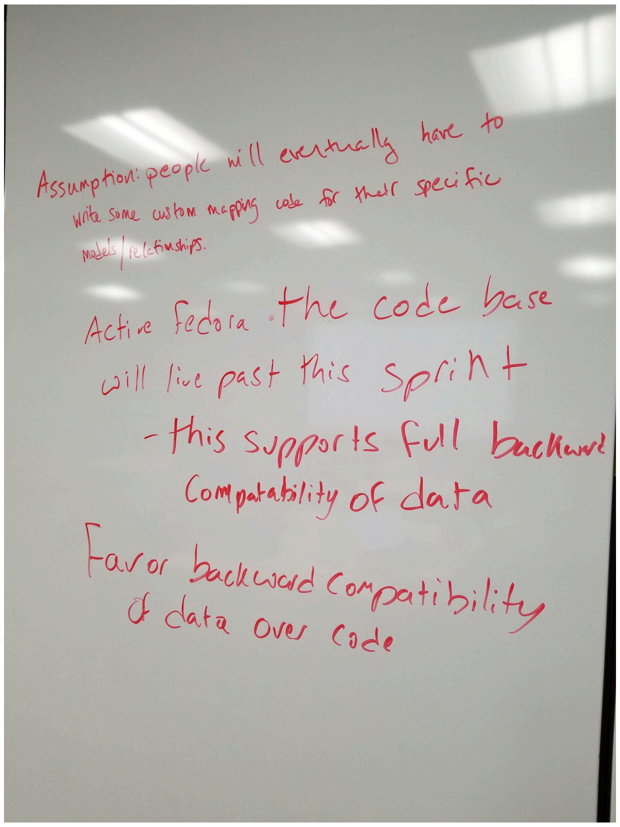
JG18 - Override Active Fedora Save delegate to Valkyrie

MetadataModel posited

tackle API was going to be Large



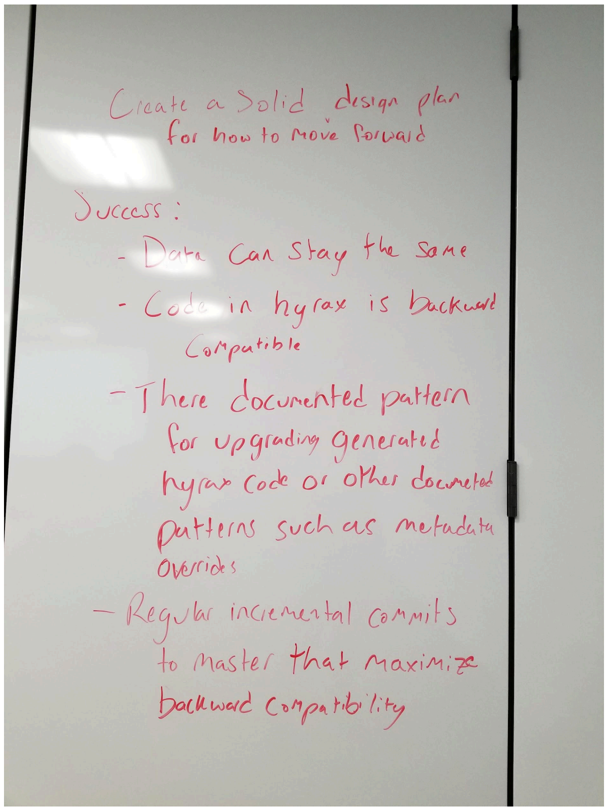

Code
Data
Documentation
JF19
PS19
PS19
JG18
CAC19
?
?
CAC19
JG18
JC17?
JC17?
JC17
JF19



Assumption: people will eventually have to write some custom mapping code for their specific models/relationships.

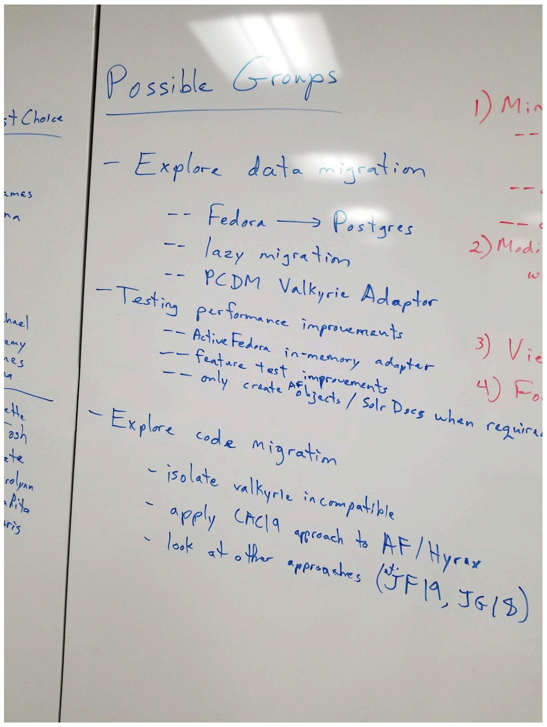
Active Fedora the code base will live past this sprint
- this supports full backward compatability of data

Favor backward compatibility of data over code



Create a Solid design plan for how to move forward

Success:
- Data can stay the same
- Code in hyrax is backward compatible
- There documented pattern for upgrading generated hyrax code or other documented patterns such as metadata overrides
- Regular incremental commits to master that maximize backward compatibility



st Choice

ames
na

hael
emy
nes
a

ette
osh
ete
rolynn
Rita
ris

Possible Groups

- Explore data migration
  -- Fedora ⟶ Postgres
  -- lazy migration
  -- PCDM Valkyrie Adaptor
- Testing performance improvements
  -- Active Fedora in-memory adapter
  -- feature test improvements
  -- only create AF objects / Solr Docs when require
- Explore code migration
  - isolate valkyrie incompatible
  - apply CAC19 approach to AF/Hyrax
  - look at other approaches (ex. JF19, JG18)

1) Mi
--
--
--
2) Modi
w
3) Vie
4) Fo

Tuesday, January 29, 2019

Links to work underway:

Active Fedora

- adding methods for valkyrie
- https://github.com/samvera-labs/valkyrie_active_fedora

Hyrax

- building a resource factory
- https://github.com/samvera/hyrax/tree/wings
- Discovered and reported issue that may exist in Valkyrie implementation (https://github.com/samvera-labs/valkyrie/issues/647)

Vanilla Hyrax on Valkyrie

- getting a working example app
- https://github.com/samvera-labs/hyrax-on-wings

Hydra Head

- valkyrizing embargo class
- https://github.com/samvera/hydra-head/tree/valkyrie

Hyrax spec optimization

- https://github.com/samvera/hyrax/pull/3483
- https://github.com/samvera/hyrax/pull/3487
- https://github.com/samvera/hyrax/pull/3488
- (WIP): https://github.com/samvera/hyrax/compare/master...jrgriffiniii:feature-tests?expand=1

Other work:

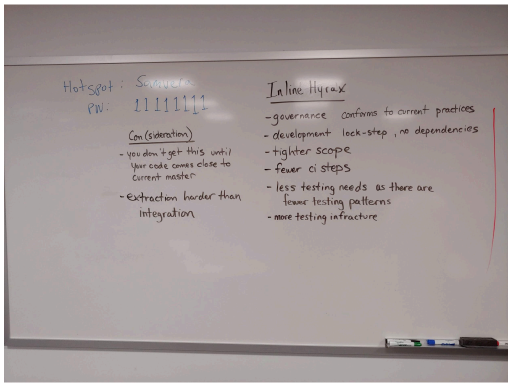
- A document explaining the particular / prohibitive challenges of developing a pcdm/fedora/hyrax Valkyrie Adapter:

https://docs.google.com/document/d/1qdZ1kcCRiNDOdHB5dJypUlDXDr5q81w7YObFpWrH9F4/edit#
- fighting with dry gem releases

Wednesday, January 30, 2019

Discussion Concerning Hyrax / Valkyrie integration…

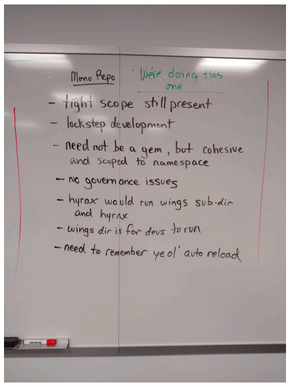
To gemify, inline, or mono-repo?



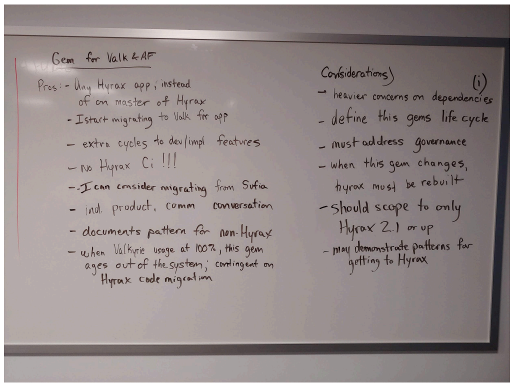
Hotspot : Samvera
PW: 11111111

Con (sideration)
- you don't get this until your code comes close to current master
- extraction harder than integration

Inline Hyrax
- governance conforms to current practices
- development lock-step, no dependencies
- tighter scope
- fewer ci steps
- less testing needs as there are fewer testing patterns
- more testing infracture



Mono Repo
'We're doing this one
- tight scope still present
- lockstep development
- need not be a gem, but cohesive and scoped to namespace
- no governance issues
- hyrax would run wings sub-dir and hyrax
- wings dir is for devs to run
- need to remember ye ol' auto reload



Gem for Valk & AF

Pros: - Any Hyrax app, instead of on master of Hyrax
- I start migrating to Valk for app
- extra cycles to dev/impl features
- No Hyrax Ci !!!
- - I can consider migrating from Sufia
- ind. product, comm conversation
- documents pattern for non-Hyrax
- when Valkyrie usage at 100%, this gem ages out of the system; contingent on Hyrax code migration

Considerations)
(i)
- heavier concerns on dependencies
- define this gems life cycle
- must address governance
- when this gem changes, hyrax must be rebuilt
- Should scope to only Hyrax 2.1 or up
- may demonstrate patterns for getting to Hyrax

Decision

We will create a namespace in Hyrax which will include all of the Hyrax modifications to inline Valkyrie usage.

We have Wings!

1st PR merged: https://github.com/samvera/hyrax/pull/3486

Thursday, January 31, 2019

Hyrax / Valkyrie integration…

Plus/Delta

Jeremy F: With apologies as whiteboard note taker for not originally capturing Haki Sack as a "+" and failing to credit Carolyn for her proof of concept (I had written PSU).

- + Haki Sack

- + Space for niche work
- + For fast lib/wings.rb spec feedback
- - No Steve on site
- + Having Trey available via Slack
- - S…L…O…W……T…E…S…exit 1
- + Push to get code in, but not w/o conversations
- + Varied experiences with high skill-level
- + Documentation for upcoming contributors
- + Show proof of concept in app
- + Dockerizing Hyrax on Wings app
- - Particular problem space was crowded, needed to find adjacent work (e.g. Valkyrie tickets, speeding up tests)
- - Dry-RB won't go 1.0.0 already
- - We went down separate approaches a little too long early in the week
- - Cold location
- + Having survey to review and build success criteria
- + People checking in on others
- - Tech wifi antics
- + Taking time to plan out how to do this against master
- + Working on CircleCI (having time to focus on this w/ "pressure" for fast moving tests)
- + To Massive and Plentiful whiteboards
- + Community acting on an identified existential crisis
- + Lead time for sprint week (opportunity to review plans)
- + On Site helps quick flow of info
- + Taking time to review what was done, individual skills, options w/ pro/con comparison
- + People did exploratory work before arriving
- + Breaking to talk through decisions/questions that emerged
- + Having a lunch bringer
- + Morning, Lunch, and Closing check-ins each day
- + Driveable for many
- + Collegiality, pairing and Switching
- + For having a full week
- + To gain exposure to Valkyrie and the approach
- - Travel coordination woes
- + For heavy documentation on 1st day
- + Small changes
- + Community ground work
- + For Carolyn Cole's proof of concept, this gave a concrete approach