Avalon technical deep-dive

Date
01 Oct 2015

Attendees
- Michael B. Klein - Northwestern
- Chris Colvard - Indiana University

Goals
- Provide some in-depth technical coverage into the underpinnings of the architecture and systems involved in the current version of Avalon.
- Also cover at least briefly along the way, where Avalon is headed.

Summary
Avalon is currently comprised of:
- Ruby 2.1
- Rails 4.0
- Hydra 8.0
- Solr (no version provided)
- A database
- Delayed_Job (ruby gem)

Avalon can use external groups and leverages Omniauth (among others) to tie into (get class lists from) services such as LDAP or CAS.

Avalon provides interoperability with LMS / CMS systems (Sakai, Blackboard) through the use of its LTI Tool.

Avalon supports permalinks.

Avalon uses z39.50 SRU to support bibliographic records.

Avalon uses MediaElement.js for interaction with streaming media controls.

In the new *.ova Avalon build released for this connect, "hydra2015" is the password for everything. Also, "archivist1@example.com" is the default user baked into the application.

Question: Why "delayed_job" over "redis"? Answer: Avalon will probably be moving to Redis / Resque in the future.

Avalon uses OpenCast Matterhorn as it's transcoding engine. However, Avalon does not work with the out-of-the-box version of Matterhorn.

The Avalon technical team is building up ActiveEncode (github, confluence) as an abstraction layer to contain the necessary changes to Matterhorn as well as code related to transcoding in general.

Streaming is done through either RTMP or HLS (originally an Apple standard). Streaming engine options within Avalon are as follows:
- Red 5. RTMP only. Can serve pre-rendered HLS.
- Adobe Media Server. Can use RTMP. Can serve HLS through Apache.
- Wowza. Capable of both RTMP or HLS.

Avalon authorization code only works with Adobe Media Server. *(needs further explanation).

Nginx and Lighttpd are both not yet supported with Avalon. Supporting either platform would require the ability to exclusively use the http protocol for streaming.

Copy content first, then manifests. * - was related to batch importing, needs further context / detail.

Avalon has an "about" page which details information concerning the current operation of the application.
- http://[avalon]/about
- http://[avalon]/about/health.html or http://[avalon]/about/health.json
- http://[avalon]/about/fedora.html or http://[avalon]/about/fedora.json
- http://[avalon]/about/delayed_job.html or http://[avalon]/about/delayed_job.json
- Pages for other parts of the application are also available.

It will be important to protect the http://[avalon]/about/* pages. Best practice would be to use apache to protect those pages.

avalon.yml
- dropbox:
  - upload_uri: cifs://media.example.edu/avalon_dropbox
Avalon has undergone a few iterations of philosophical changes regarding how to handle configuration.

- Not all configurations use a consistent style across the application
- There are some possible leftover artifacts from earlier iterations of configuration philosophy.

Avalon::GROUP_LDAP

- gist.github.com/mbklein [didn't catch the url, but there is a gist out there on this topic]

Masterfile Management - it is possible to set a location for the masterfile to be relocated to once transcoding has been completed.

Matterhorn tends to get gummed up with stuck jobs. There are scripts included to clean up Matterhorn jobs.

Avalon's Layout & Content views are not very separate. There has been thought about abstracting the two out, but so far not enough interest / priority.

[Probably in part due to the topic of the line above] adding, removing, or customizing metadata fields is really tricky. One "change" to metadata would require representation in multiple different places as well as changing behavior in multiple different places.

[What is a jumpfile? What does it have to do with SHA?]

What is "Zencoder"? Chris Colvard did a proof of concept. Could replace Opencast Matterhorn. Amazon Elastic Encode was also considered.

Avalon technical team does not recommend Red5 for production. Doubles sizes on masterfiles. Seeking behavior sometimes is wonky.