Clowder Data Archiving Support

Driving Scenario I: TERRA-REF project uses file path uploading to create file entries in Clowder that point to mounted file paths (i.e. data bytes are not stored in MongoDB). TERRA's 1PB allocation on storage condo is filling up, necessitating some files (likely starting with raw data from 2016+) be moved into tape storage or offsite backup. This will be done manually and recovering files from the archive will require manual action.

Driving Scenario II: Industry partnership project would like to move files that have not been downloaded after X days automatically from S3 storage to Glacier. However, they would also like a button to automatically schedule the file to be restored from Glacier back to S3.

In both scenarios, we want to retain entries in Clowder for data that we archive for referencing and metadata purposes.

Completed work

https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/clowder/pull-requests/1364/overview

This pull request adds the following:

- support for ARCHIVED status (to be displayed in right column of a File page)
- /files/:id/archive endpoint that can be POSTed to to assign that status
- if user attempts to download archived file, a new window will open with the /email form, with subject and body pre-populated to indicate the user wants to retrieve the file from the archive. This email would go to server admins.
- Add a "This email was sent on behalf of..." footer to emails if the mimicuser config = false, so in cases where we don't want to spoof an email address (e.g. industry partner) there is still enough to get back in touch with the correct user.

Proposed future design

The pull request doesn't yet address the desire to support automatically archiving/unarchiving files on a user request. We have discussed one possible architecture that would leverage extractors to perform these two tasks.

- Implement Archiver and Unarchiver extractors per use case
  - Basic (TERRA-REF)
    - Archiver basically does nothing, or maybe emails site admins
  - AWS (Industry Partner)
    - Archiver will contain credentials necessary to move a file from S3 to Glacier
    - Unarchiver will move a file from Glacier to S3
- Add a "life limit" policy that can be set at the User/Space/Instance level (in order of preference) which, if above 0, will trigger Archiver automatically if the file has not been downloaded in that many days
- There would also be the ability for any script/process/other extractor to trigger the Archiver as desired, so if there is an extractor that is the final step in a workflow, it could notify Clowder to Archive the working files necessary for the workflow while preserving the final outputs.

Low-level Implementation Ideas

- When the module is disabled, we could offer an e-mail based strategy to process archive requests (as in "Completed Work" above)
- To enable the module, configure one or more backup extractors (possibly in an array in the configuration?)
- When the module is enabled, unarchived files offer an "Archive" option in the UI - when pressed, config is checked for which extractor should be used for archival and request is sent
  - If useful, this could potentially offer multiple archive targets if more than one is configured - e.g. S3, NFS, Glacier, etc

05/29 discussion notes

- We have an Archive/Unarchive button alongside Download, and Download is hidden if the file is Status: Archived
  - Sends an 'archive' or 'unarchive' parameter to extractor so 1 extractor can handle both modes
- Add a new Archive permission that we can configure like the others - gives us nice control over who can archive things, where
- extractor_info has a category that allows the UI to filter which are shown
  - Could potentially implement support for the Process block in extractor info as well - trigger on S3/Mongo/Disk storage, certain MIME types, etc.
- Global and per-space lifetime setting (30 = archive if file is not downloaded for 30 days)
  - https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/clowder/browse/app/services/mongodb/MongoDBSpaceService.scala#449 this might have been the start of implementation, but this function is not used anywhere

Open Questions

- Is this new optional functionality a Play! Framework "module"? Perhaps this is something entirely different?
- Should we support multiple archive options simultaneously (similar to the proposed Multiple Storage Backend feature)?
- Could we easily provide a common pattern for extractor developers to use as a base for such archive/unarchive extractors?
  - Would such a pattern require two different extractors? Could we easily parameterize the `process_message` call?

- Should the backup extractors also be listed in the Manual Submission view with the other extractors?
    - This could confuse end users, but as long as extractor is **SAFE** this should be ok - this may take careful design