Archiving Files

This documentation may be out of date. For the most recent documentation for the archival extractors, see the READMEs in GitHub: https://github.com/clowder-framework/extractors-archival

The archival process in Clowder is an optional addon to the already optional RabbitMQ extractor framework.

The definition of "archiving" is left to the implementation of each archival extractor, and "unarchiving" is the exact inverse of that process.

Two archival extractor implementations exist that currently depend on which ByteStorageDriver you are using:

- **ncsa.archival.disk:** Moves the file from one specially-designated folder on disk to another **(requires write access to Clowder's data directory)**
- **ncsa.archival.s3:** Changes the Storage Class of an object stored in S3 **(requires write access to Clowder's bucket in S3)**

The two options cannot currently be mixed, meaning that if Clowder uses DiskByteStorage then you must use the Disk archiver.

If neither of these two extractors fit your use case, pyclowder can be used to quickly create a new archival extractor that fits your needs.

Process Overview

When a file is first uploaded, it is placed into a temp folder and created in the DB with the state CREATED.

At this point, users can start associating metadata with the new file, even though the actual file bytes are not yet available through Clowder's API.

Clowder then begins transferring the file bytes to the configured ByteStorage driver.

Once the bytes are completely uploaded into Clowder and done transferring to the data store, the file is marked as PROCESSED.

At this point users can access the file bytes via the Clowder API and UI, and are able download the file as normal.

If the admin has configured the archival feature (see below), then the user is also offered a button to Archive the file.

On Archive

If a user chooses to Archive the file, then it is sent to the configured archival extractor with a parameter of ***operation=archive***.

The extractor performs whatever operation it deems as "archiving" - for example, copying to a network file system.

The page will refresh automatically after a few seconds - for most cases, this hard-coded delay is enough to pick up the new File status of ARCHIVED on refresh.

Finally the file is marked as ARCHIVED, and (if configured) the user is given the option to Unarchive the file.

If the user attempts to download an ARCHIVED file, then they should be presented with a prompt to notify the admin for a request to unarchive.

On Unarchive

If a user chooses to Unarchive a file, then it is sent to the configured archival extractor with a parameter of ***operation=unarchive***.

The extractor performs the inverse of whatever operation that it previously defined as "archiving", bringing the file bytes back to where Clowder can access them for download.

NOTE: The page will refresh automatically after a few seconds - for most cases, this hard-coded delay is enough to pick up the new File status of PROCESSED on refresh.

Finally the file is marked as PROCESSED, and the user should be once again given the option to Archive the file and requests to download the file bytes should succeed.

Configuration Options / Defaults for Clowder

To use the archival feature, the RabbitMQ plugin must be enabled and properly configured.

With the RabbitMQ plugin enabled, the following defaults are configured in application.conf, but can be overridden by using a custom.conf file:

Configuration Path	Default	Description
`archiveEnabled`	`false`	If true, Clowder should perform a lookup once per day to see if any files uploaded the past hour are candidates for archival.
`archiveExtractorId`	`"ncsa.archival.disk"`	The id of the Extractor to use for archival • Use ncsa.archival.`disk` for `DiskByteStorageDriver` • Use ncsa.archival.`s3` for `S3ByteStorageDriver`
`archiveAllowUnarchive`	`false`	If true, the UI should offer a way to Unarchive a file that is ARCHIVED

Automatic File Archival

If configured (see below), Clowder can automatically archive files of sufficient size after a predetermined period of inactivity.

By default, this behavior is disabled.

The default values after enabling the feature will cause files that are over 1MB and have not been downloaded in that last 90 days to be automatically archived.

Both the file size and the inactivity period can be configured according to your preferences.

Configuration Path	Default	Description
`archiveAutoInterval`	`0`	If == 0, disable automatic archiving. If > 0, check every `interval` seconds for candidates for automatic archival.
`archiveAutoDelay`	`120`	Number of seconds to wait before starting the first iteration of the automatic archival loop.
`archiveAutoAfterInactiveCount`	`90`	Number of units a file can go un-downloaded before it is considered "inactive".
`archiveAutoAfterInactiveUnits`	`days`	The units for the inactivity timeout above (e.g. "90 days" old)
`archiveAutoAboveMinimumStorageSize`	`1000000`	The minimum number of bytes for a file to be considered as a candidate for automatic archival.

ncsa.archival.disk

This image has been pre-built as `clowder/extractors-archival-disk` .

(Optional) Building the Image

To build the Disk archival extractor's Docker image, execute the following commands:

```
git clone https://opensource.ncsa.illinois.edu/bitbucket/scm/cats/extractors-archival-disk.git
cd extractors-archival-disk/
docker build -t clowder/extractors-archival-disk .
```

Configuration Options

The following configuration options must match your configuration of the DiskByteStorageDriver:

Environment Variable	Command-Line Flag	Default Value	Description
`ARCHIVE_SOURCE_DIRECTORY`	`--archive-source`	`$HOME/clowder/data/uploads/`	The current directory where Clowder stores it's uploaded files

ARCHIVE_TARGET_DIRECTORY	--archive-target	$HOME/clowder/data/archive/	The target directory where the archival extractor should store the files that it archives. Note that this path can be on a network or other persistent storage.

Example Configuration: Archive to another folder

In Clowder, configure the following:

```
# storage driver
service.byteStorage=services.filesystem.DiskByteStorageService

# disk storage path
#clowder.diskStorage.path="/Users/lambert8/clowder/data"    # MacOSX
clowder.diskStorage.path="/home/clowder/clowder/data"      # Linux

# disk archival settings
archiveEnabled=true
archiveExtractorId="ncsa.archival.disk"
archiveAllowUnarchive=true

archiveAutoInterval=86400
archiveAutoDelay=300
archiveAutoAfterInactiveCount=90
archiveAutoAfterInactiveUnits=days
archiveAutoAboveMinimumStorageSize=1000000
```

To run the Disk archival extractor with this configuration:

```
docker run --net=host -itd -e MOUNTED_PATHS='{ "/Users/lambert8/clowder/data":"/home/clowder/clowder/data" }' -
v /Users/lambert8/clowder/data/:/home/clowder/clowder/data -e ARCHIVE_SOURCE_DIRECTORY="/home/clowder/clowder
/data/uploads/" -e ARCHIVE_TARGET_DIRECTORY="/home/clowder/clowder/data/archive/" clowder/extractors-archival-
disk
```

NOTE 1: `MOUNTED_PATHS` configuration is currently required without modifications to the Python code, since we require direct write access to the data directory. This prevents us from needing to download the file to archive or unarchive it.

NOTE 2: on MacOSX, you may need to run the extractor with the `--net=host` option to connect to RabbitMQ.

ncsa.archival.s3

This image has been pre-built as `clowder/extractors-archival-s3` .

(Optional) Building the Image

To build the S3 archival extractor's Docker image, execute the following commands:

```
git clone https://opensource.ncsa.illinois.edu/bitbucket/scm/cats/extractors-archival-s3.git
cd extractors-archival-s3/
docker build -t clowder/extractors-archival-s3 .
```

Configuration Options

The following configuration options must match your configuration of the S3ByteStorageDriver:

Environment Variable	Command-Line Flag	Default Value	Description
AWS_S3_SERVICE_ENDPOINT	--service-endpoint <value>	https://s3.amazonaws.com	Which AWS Service Endpoint to use to connect to S3. Note that this may depend on the region used, but can also be used to point at a running MinIO instance.
AWS_ACCESS_KEY	--access-key <value>	""	The AccessKey that should be used to authorize with AWS or MinIO
AWS_SECRET_KEY	--secret-key <value>	""	The SecretKey that should be used to authorize with AWS or MinIO
AWS_BUCKET_NAME	--bucket-name <value>	clowder-archive	The name of the bucket where the files are stored in Clowder.
AWS_REGION	--region <value>	us-east-1	**AWS only**: the region where the S3 bucket exists

AWS_ARCHIVED_STORAGE_CLASS	--archived-storage-class <value>	INTELLIGENT_TIERING	The S3 StorageClass to set for objects that are ARCHIVED.
AWS_UNARCHIVED_STORAGE_CLASS	--unarchived-storage-class <value>	STANDARD	The S3 StorageClass to set for objects that are not archived (aka PROCESSED).

Example Configuration: S3 on AWS in us-east-2 Region

In Clowder, configure the following:

```
# storage driver
service.byteStorage=services.s3.S3ByteStorageService

# AWS S3
clowder.s3.serviceEndpoint="https://s3-us-east-2.amazonaws.com"
clowder.s3.accessKey="AWSACCESSKEYKASOKD"
clowder.s3.secretKey="aWSseCretKey+asAfasf90asdASDADAOaisdoas"
clowder.s3.bucketName="bucket-on-aws"
clowder.s3.region="us-east-2"

# S3 archival settings
archiveEnabled=true
archiveExtractorId="ncsa.archival.s3"
archiveAllowUnarchive=true

archiveAutoInterval=86400
archiveAutoDelay=300
archiveAutoAfterInactiveCount=90
archiveAutoAfterInactiveUnits=days
archiveAutoAboveMinimumStorageSize=1000000
```

NOTE: Changing the Region typically requires changing the S3 Service Endpoint.

To run the S3 archival extractor with this configuration:

```
docker run --net=host -itd -e AWS_S3_SERVICE_ENDPOINT='https://s3-us-east-2.amazonaws.com' -e
AWS_ACCESS_KEY='AWSACCESSKEYKASOKD' -e AWS_SECRET_KEY='aWSseCretKey+asAfasf90asdASDADAOaisdoas' -e
AWS_BUCKET_NAME='bucket-on-aws' -e AWS_REGION='us-east-2' clowder/extractors-archival-s3
```

NOTE: on MacOSX, you may need to run the extractor with the `--net=host` option to connect to RabbitMQ.

Example Configuration: MinIO

In Clowder, configure the following to point the S3ByteStorageDriver and the archival extractor at your running MinIO instance:

```
# storage driver
service.byteStorage=services.s3.S3ByteStorageService

# Minio S3
clowder.s3.serviceEndpoint="http://localhost:8000"
clowder.s3.accessKey="AMINIOACCESSKEYKASOKD"
clowder.s3.secretKey="aMinIOseCretKey+asAfasf90asdASDADAOaisdoas"
clowder.s3.bucketName="bucket-on-minio"

# S3 archival settings
archiveEnabled=true
archiveExtractorId="ncsa.archival.s3"
archiveAllowUnarchive=true

archiveAutoInterval=86400
archiveAutoDelay=300
archiveAutoAfterInactiveCount=90
archiveAutoAfterInactiveUnits=days
archiveAutoAboveMinimumStorageSize=1000000
```

NOTE: MinIO ignores the value for "Region", if one is specified.

To run the S3 archival extractor with this configuration:

```
docker run --net=host -itd -e AWS_S3_SERVICE_ENDPOINT='http://localhost:8000' -e
AWS_ACCESS_KEY='AMINIOACCESSKEYKASOKD' -e AWS_SECRET_KEY='aMinIOseCretKey+asAfasf90asdASDADAOaisdoas' -e
AWS_BUCKET_NAME='bucket-on-minio' -e AWS_ARCHIVED_STORAGE_CLASS='REDUCED_REDUNDANCY' clowder/extractors-
archival-s3
```

NOTE: on MacOSX, you may need to run the extractor with the `--net=host` option to connect to RabbitMQ.

Common Pitfalls

Gotcha: attempting to download an ARCHIVED File via the Clowder API's `/api/files/:id` endpoint will result in a 404

This is important for debugging, as it was a little confusing until I realized what was going on.

Perhaps 404 is the wrong error code here, or maybe it just needs a better error message for this edge case.

There may be an implication that there is no model found with that ID, when really the problem is that it's internal state simply prevents it from being downloaded.

My vote is for 418 I'm a teapot, but something like **409 Conflict** or **410 Gone** would likely get the point across.

412 Precondition Failed and **417 Expectation Failed** also look promising, but I'm not sure if these have other underlying implications for the browser.

Gotcha: communication issues between containers

NOTE: This issue can be tricky to workaround, as it is typically very environment-specific or setup-specific.

For example, this can happen if you're running RabbitMQ and the Extractor in separate containers, where the RabbitMQ container was created with `--net=host` and your Extractor was not.

As another example, this can happen on MacOSX if Clowder is running in IntelliJ on the host, with RabbitMQ and the extractor running in Docker containers (without `--net=host`).

Since one container is on the host network and the other is on the Docker bridge network, the two containers cannot communicate with each other.

The following configuration snippet can be added to Clowder's `custom.conf` to override the hostnames where both expect to find each other:

```
clowder.rabbitmq.uri="amqp://guest:guest@<PRIVATE IP>:5672/%2F"
clowder.rabbitmq.exchange="clowder"
clowder.rabbitmq.clowderurl="http://<PRIVATE IP>:9000"
```