Repair

Repairing is a process that allows for node-to-node resolution of corrupt files in Chronopolis. Rather than being an automated process, a node administrator must choose which files to repair and which node to repair from. This is to prevent unnecessary repairs (e.g. due to errors from a filesystem being offline) and also to allow for discussion and investigation about the collection prior to it being repaired.

Links

- Gitlab: Chron-Medic
- API Documentation

Installation

Download and install the rpm

Installation Notes

The rpm creates a Chronopolis user if it does not exist, and creates the following files/directories:

```
/etc/chronopolis
/etc/chronopolis/repair.yml
/etc/init.d/chron-repair
/usr/lib/chronopolis
/usr/lib/chronopolis/chron-repair.jar
/var/log/chronopolis
```

Configuration

The configuration for the repair service is done in the repair.yml under /etc/chronopolis

repair.yml

```
# Application Configuration for the Chronopolis Repair

# cron timers
## cron.repair: how often to check the Ingest Server repair endpoint
## cron.fulfilment: how often to check the Ingest Server fulfillments endpoint
cron:
  repair: 0 0/1 * * * *
  fulfillment: 0 0 * * * *

# general properties
## repair.stage: staging area to replicate files to before they are moved to preservation storage
## repair.preservation: preservation storage area
repair:
  stage: /export/repair/staging
  preservation: /preservation/bags

# Chronopolis Ingest API configuration
## ingest.endpoint: the url of the ingest server
## ingest.ucsername: the username to authenticate as
## ingest.password: the password to authenticate with
ingest:
  endpoint: http://localhost:8000
  username: node
  password: nodepass

# rsync configuration for fulfillments
## rsync.path: used if chrooting users rsyncing - the path under the chroot context
## rsync.stage: a staging area which fulfillments will be copied to
## rsync.server: the fqdn of the server nodes will replicate from
rsync:
  path: /export/repair/outgoing
  stage: /export/repair/outgoing
  server: loach.umiacs.umd.edu

# ACE AM configuration
## ace.am: the local ACE AM webapp
## ace.username: the username to authenticate as
## ace.password: the password to authenticate with
ace:
  am: http://localhost:8080/ace-am/
  username: admin
  password: admin

# spring properties
## spring.profiles.active: the profiles to use when running
##                         recommended: default, rsync
spring:
  profiles:
    active: default, rsync

# logging properties
## logging.file: the file to write logging statements to
## logging.level: the log level to filter on
logging.file: /var/log/chronopolis/repair.log
logging.level.org.chronopolis: INFO
```

Running

The Repair Service ships with a SysV style init script and has the basic start/stop/restart options. Customization of the script may be necessary if your java location needs to be specified.

> *service chron-repair start|stop|restart*

Workflow

Note that the workflow involves two nodes: one with **CORRUPT** data and one with **VALID data**

1. **CORRUPT** notices data in their Audit Manager (ACE-AM) is showing File Corrupt, indicating that checksums on disk have changed
    a. Discussion happens internally about who has this and can repair it
    b. SSH keys exchanged so that data transfer can occur for files which are to be repaired
2. **CORRUPT** logs on to the Ingest server and selects 'Request Repair' in order to create a 'Repair Request'
    a. Inputs ACE AM credentials to query for the corrupt collection
    b. Select the Collection
    c. Select the Files to repair and the Node where they will be Repaired
3. **VALID** logs onto the Ingest server and selects 'Fulfill Repair' in order to stage data for the repair
4. At this point, both **CORRUPT** and **VALID** nodes should start the Repair service
    a. The Repair service running at **VALID** will stage data and update the Repair
    b. The Repair service running at **CORRUPT** will
        i. Pull data from **VALID** into a staging area
        ii. Validate that the data transferred and matches the checksums in the ACE AM
        iii. Overwrite the corrupt files
        iv. Audit the files in the ACE AM
        v. Update the Repair with the result of the audit
5. Once complete, the Repair Service at each node can be stopped

Repair File Transfer Strategies

During design of the Repair service, it was noted that there are different ways of transferring content between Chronopolis Nodes:

- Direct transfer
    - through rsync
    - through ACE AM
- Indirect transfer through the Ingest Server

During development, support was added for each type of transfer, but only the direct rsync strategy was implemented. The direct ACE-AM transfer strategy requires additional development in the Audit Manager in order to support API Keys which can be used to access content. The indirect transfer through the Ingest Server was omitted as it was not deemed onerous for Chronopolis Nodes to exchange ssh keys.

Repair Types

Currently the Repair workflow handles repairing corrupt files, but does not cover other types of failure which can occur in the system. For example, in the past we have had issues with the Audit Manager (ACE-AM) having received invalid checksums from the underlying storage system, which then needed to be updated in order for an audit to pass successfully. We have also see ACE Token Stores be partially loaded which results in the need to re-upload the ACE Token Store so that we can ensure we are auditing against the ACE Tokens we created on ingestion of the collection.

Release Notes

Release 1.5.0

26 June, 2017

Initial release for the Chronopolis Medic (Repair) software to process Repair requests on the Chronopolis Ingest Server

- Repairs have namespaced areas when staging as to not interfere with other ongoing Repairs
- Staging files done via symbolic links (other staging options supported later)
- Rsync support for the main protocol for distributing files (other protocols supported later)
- Comparison with a nodes ACE-AM before files are moved into production storage
- Staging areas for both repairing and fulfilling nodes cleaned upon completion of a Repair