Extractor chaining pipelines

Extractors are useful for performing a specific task, such as calculating metadata to attach to a resource or generating additional output files. However, there is sometimes a need for complex processes where several extractors need to daisy-chan, particularly if an extractor needs multiple inputs that each need a separate individual extractor run on them first before running.

Some projects have started employing solutions for this situation, and this page is intended to discuss models that are currently in use to determine if use cases can be better supported.

TERRA-REF Pipeline

TERRA uses a unique pipeline of extractors for each of the ~10 main sensors on the sensing platform, and each pipeline includes 5-10 extractors that operate as discrete steps in the workflow.

Rulechecker

If all 10 extractors were all configured to listen on files being uploaded to datasets and use the check_message() step to determine whether to process or not, there would be tremendous overhead as each extractor would need to comb through 9 other unrelated sensors' data constantly looking for relevant data, as RabbitMQ would send the event notification to all of them each time. Instead, TERRA developed the "rulechecker" extractor (colloquially called the switchboard) to manage the data flow.

https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/extractors-rulechecker/browse

(needs some updates from the TERRA fork: https://opensource.ncsa.illinois.edu/bitbucket/users/mburnet2/repos/terraref-rulechecker/browse)

The basic setup:

- Individual extractors such as the bin2tif converter are NOT configured to listen for any events. They can only be triggered directly (e.g. something passes a message to their queue specifically by name)
- The rulechecker switchboard is configured to listen for "dataset.file.added" events. when an event lands, rulechecker evaluates many rules at once - these rules are akin to the check_message() functions of the downstream extractors.
    - e.g. "is this a stereoTop dataset with 2 BIN files? if yes, trigger bin2tif. if no, move on to next rule"
- In this way, uploading 10,000 files generates 10,000 messages for rulechecker, which uses it's switchboard set of rules to pass those messages along to the necessary downstream extractors. So instead of bin2tif, flir2tif, ply2las, etc. all getting 10,000 messages, they only get ~3,300 each. Rulechecker alone sees every message.

This is a way to manage the fact that, for dataset-level extractors, it is not possible to specify a MIME type or other parameter for triggering and instead dataset extractors must either evaluate every dataset message (which rulechecker does), or trigger via alternative means (the extractors rulechecker triggers).

Daisy-chaining directly

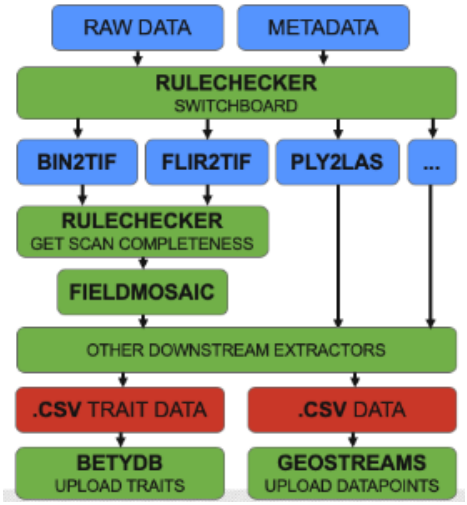
From there, each extractor can trigger the next extractor directly in the chain if possible, at the end of the process_message() function.

- bin2tif uses pyclowder's submit_extraction() function to pass it's geoTIFF outputs along to 3 additional extractors, for example.

This is the simplest option of extractor pipelines are simple direct 1-1 paths.

Collection-level extractors

For extractors such as the fieldmosaic stitcher that mosaics together 9000+ images from a single day, the rulechecker extractor is used once again. Each geoTIFF is passed back to rulechecker, which triggers a special rule to add that geoTIFF to a PSQL database maintaining a list of geoTIFFs for a specific day or scan. Once a threshold is met, all 9000 geoTIFFs are passed to the fieldmosaic extractor (each of the geoTIFFs from a different dataset) to stitch them all at once.



Rulechecker is useful when:

- a pipeline has many dataset extractors, but the pipeline has a lot of traffic which is not relevant to any individual dataset extractor. For TERRA, this makes sense because we have 10 different sensors and ~30 different products from those sensors. Without a switchboard filter, all the RabbitMQ queues for those extractors would be filled with irrelevant messages. Rulechecker reduces the traffic to those queues immensely.
- an extractor needs to trigger only when certain cross-dataset or cross-file conditions are met, e.g. "only when we have 100 CSVs from June 3 extracted" or "only when all 8000 datasets in this collection have Height metadata". Rulechecker has a PSQL database by default underlying it, and the rule_utils library included with rulechecker provides shortcut methods for reading and writing from the database as a means for tracking progress toward triggering a "big" extractor.

For in-depth example, see the terraref_switchboard() function in TERRA's rules.py, a file that is necessary for a rulechecker deployment defining which rules to execute on each dataset (https://opensource.ncsa.illinois.edu/bitbucket/users/mburnet2/repos/terraref-rulechecker/browse/rules.py).