Installing, configuring, and using Clowder and its extractors

Set up Clowder instance locally

For developing and testing new extractors and previewers, it is the best for developers to set up the Clowder stack locally first. The core components of Clowder are packaged as Docker images and can be run on most of the computation resources via Docker the Virtual Environment. Follow the below steps to get started:

1. Install Docker: https://www.docker.com/get-started/
2. Install Git if you have not already: https://git-scm.com/
3. Use any command line tool to clone the clowder repository from Github by typing git clone https://github.com/clowder-framework/clowder.git. You can access this Github repository from this link: https://github.com/clowder-framework/clowder.
4. Navigate to Clowder's root directory in your command line cd clowder
5. Clowder can be configured through a docker-compose configuration file. Make sure you have the docker-compose.yml in the root of the Clowder directory. Start Clowder via your command line: docker-compose up -d. You can check all the running containers by typing docker ps . You should see running containers include clowder, mongodb, rabbitmq, and elasticsearch
6. Open your web browser to localhost:8000 and you can access the Clowder interface now. If you see Error 404, allow a minute for it to appear.
7. Create an account by running this in your terminal:

   ```
   docker run --rm -ti --network clowder_clowder -e  FIRST_NAME=Admin
   -e LAST_NAME=User -e EMAIL_ADDRESS=admin@example.com -e
   PASSWORD=catsarecute  -e ADMIN=true clowder/mongo-init
   ```

   Optionally, edit these properties to your liking:
   - FIRST_NAME
   - LAST_NAME
   - EMAIL_ADDRESS
   - PASSWORD
   - ADMIN (only set this if you want the user to have super admin rights, make sure at least one user has this).

8. To use Clowder, please follow the User Guide here: User Guide — Clowder 1.20.2 documentation
9. **For more information, please see the complete installation guide here: https://clowder-framework.readthedocs.io/en/latest/userguide/installing_clowder.html#clowder-developers-getting-started**

Enable extractors

One of the major features of Clowder is the ability to deploy custom extractors that perform various analytical tasks and extract information from the files and datasets uploaded to the system. Extractors can be triggered by different events such as

- File uploaded
- File added to dataset
- File remove from dataset
- Metadata added to file
- Metadata remove from file
- Metadata added to dataset
- Metadata removed from dataset
- File/Dataset manual submission to extractor
- etc

We have curated a list of extractors available on GitHub: https://github.com/clowder-framework/extractors-core. The default extractors offer simple quality of life improvements for image, video, pdf, and audio file previews while browsing Clowder.

To use default extractors, on command line start Clowder with below command:

```
docker-compose -f docker-compose.yml -f docker-compose.override.yml -f docker-compose.extractors.yml up -d
```

You can edit the extractor config file `docker-compose.extractors.yml` to include/exclude certain extractors.

Add new extractors

Write extractors

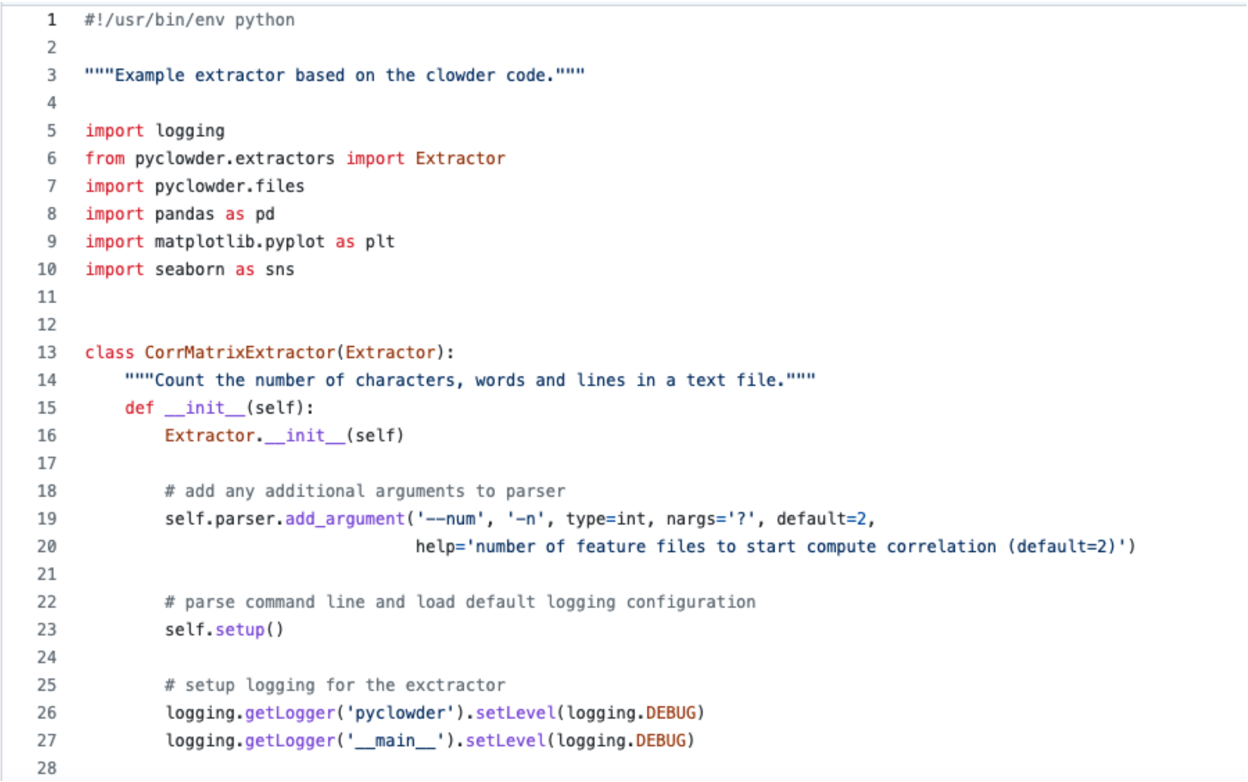
To write new extractors, pyClowder is a good starting point. It provides a simple Python library to write new extractors in Python. Please see the sample extractors directory for examples. That being said, extractors can be written in any language that supports HTTP, JSON and AMQP (ideally a RabbitMQ client library is available for it).

The prototype voice vital extractors are curated in this github repository: https://github.com/HarshitBagla/voice-vitals-extractor

There are three individual extractors located in separate folders:

- opensmile-feature-extractor utilizes the Open SMILE library https://audeering.github.io/opensmile/ to extract features of each audio clip. It is triggered by file uploads with file extensions of .wav; the extracted features are stored in tabular format as .csv file and uploaded by the extractor back to the original dataset.
- corr-matrix-extractor calculator the correlation matrix of the extracted feature within a given dataset. It needs a manual submit; and correlation calculation will only execute once a given number of audio clips exist in the dataset. This number can be set in the dockerfile --num 2
- spectrogram-vizualization-extractor extracts the spectrogram of an audio and attach the image as previews next to the audio clip

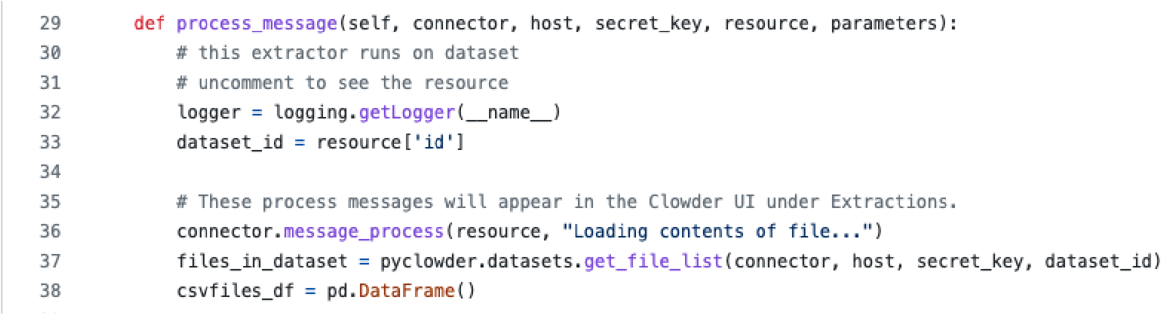
Core code of extractors explained:



```
#!/usr/bin/env python

"""Example extractor based on the clowder code."""

import logging
from pyclowder.extractors import Extractor
import pyclowder.files
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


class CorrMatrixExtractor(Extractor):
    """Count the number of characters, words and lines in a text file."""
    def __init__(self):
        Extractor.__init__(self)

        # add any additional arguments to parser
        self.parser.add_argument('--num', '-n', type=int, nargs='?', default=2,
                                 help='number of feature files to start compute correlation (default=2)')

        # parse command line and load default logging configuration
        self.setup()

        # setup logging for the exctractor
        logging.getLogger('pyclowder').setLevel(logging.DEBUG)
        logging.getLogger('__main__').setLevel(logging.DEBUG)

```

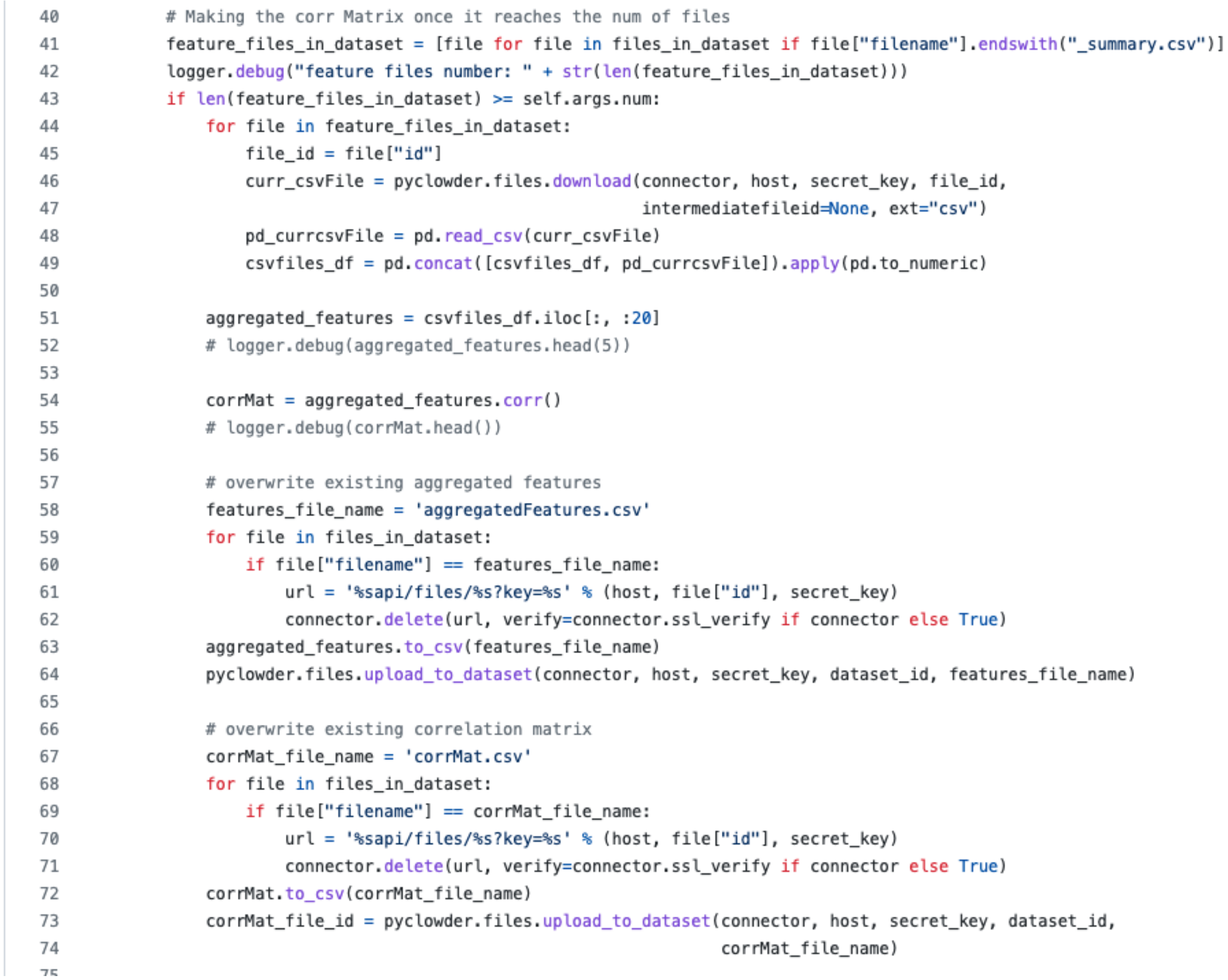
First, create a new class inheriting from the pyclowder extractors Extractor class. Make sure you include those boilerplate code for initiating extractor, adding arguments, setups, and logging.

self.parser.add_argument() is used to pass in custom arguments when starting this extractor container. For example, we include "num" parameter here; pass the "num" value in Dockerfile:
CMD python3 CorrMatrixExtractor.py --heartbeat 40 --num 2



```
    def process_message(self, connector, host, secret_key, resource, parameters):
        # this extractor runs on dataset
        # uncomment to see the resource
        logger = logging.getLogger(__name__)
        dataset_id = resource['id']

        # These process messages will appear in the Clowder UI under Extractions.
        connector.message_process(resource, "Loading contents of file...")
        files_in_dataset = pyclowder.datasets.get_file_list(connector, host, secret_key, dataset_id)
        csvfiles_df = pd.DataFrame()
```

process_message() method will have access to the triggering clowder event and information through the parameters passed in, such as connector, secret_key, resource, etc. Pyclowder provides easy-to-use wrapper functions to work with those parameters for various tasks. For example, you can get a list of files within a given dataset by pyclowder.datasets.get_file_list().

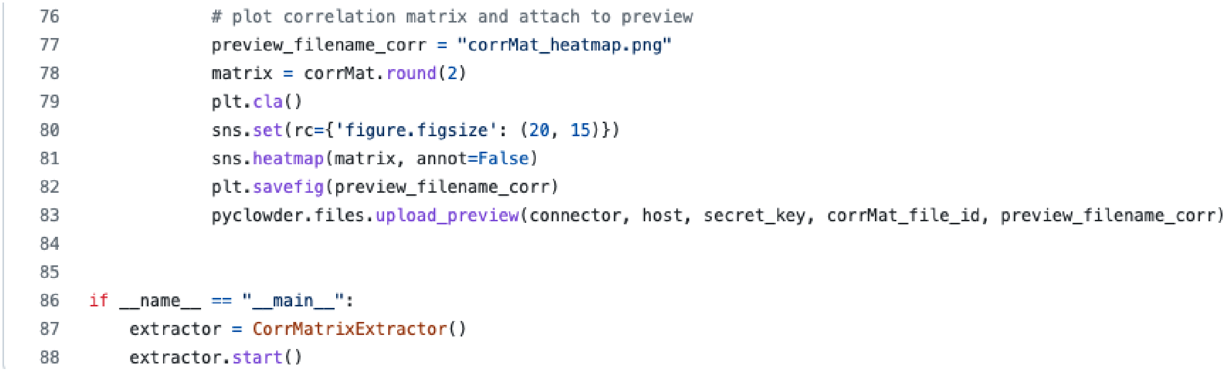


```
        # Making the corr Matrix once it reaches the num of files
        feature_files_in_dataset = [file for file in files_in_dataset if file["filename"].endswith("_summary.csv")]
        logger.debug("feature files number: " + str(len(feature_files_in_dataset)))
        if len(feature_files_in_dataset) >= self.args.num:
            for file in feature_files_in_dataset:
                file_id = file["id"]
                curr_csvFile = pyclowder.files.download(connector, host, secret_key, file_id,
                                                        intermediatefileid=None, ext="csv")
                pd_currcsvFile = pd.read_csv(curr_csvFile)
                csvfiles_df = pd.concat([csvfiles_df, pd_currcsvFile]).apply(pd.to_numeric)

            aggregated_features = csvfiles_df.iloc[:, :20]
            # logger.debug(aggregated_features.head(5))

            corrMat = aggregated_features.corr()
            # logger.debug(corrMat.head())

            # overwrite existing aggregated features
            features_file_name = 'aggregatedFeatures.csv'
            for file in files_in_dataset:
                if file["filename"] == features_file_name:
                    url = '%sapi/files/%s?key=%s' % (host, file["id"], secret_key)
                    connector.delete(url, verify=connector.ssl_verify if connector else True)
            aggregated_features.to_csv(features_file_name)
            pyclowder.files.upload_to_dataset(connector, host, secret_key, dataset_id, features_file_name)

            # overwrite existing correlation matrix
            corrMat_file_name = 'corrMat.csv'
            for file in files_in_dataset:
                if file["filename"] == corrMat_file_name:
                    url = '%sapi/files/%s?key=%s' % (host, file["id"], secret_key)
                    connector.delete(url, verify=connector.ssl_verify if connector else True)
            corrMat.to_csv(corrMat_file_name)
            corrMat_file_id = pyclowder.files.upload_to_dataset(connector, host, secret_key, dataset_id,
                                                                corrMat_file_name)
```

For the correlation matrix calculation, we first get a list of CSV files following the filename pattern _summary.csv within a given dataset. Once the file number surpasses the given number (for example, if more than 2 files exist in the dataset), we start to aggregate the features from those files and calculate the correlation.

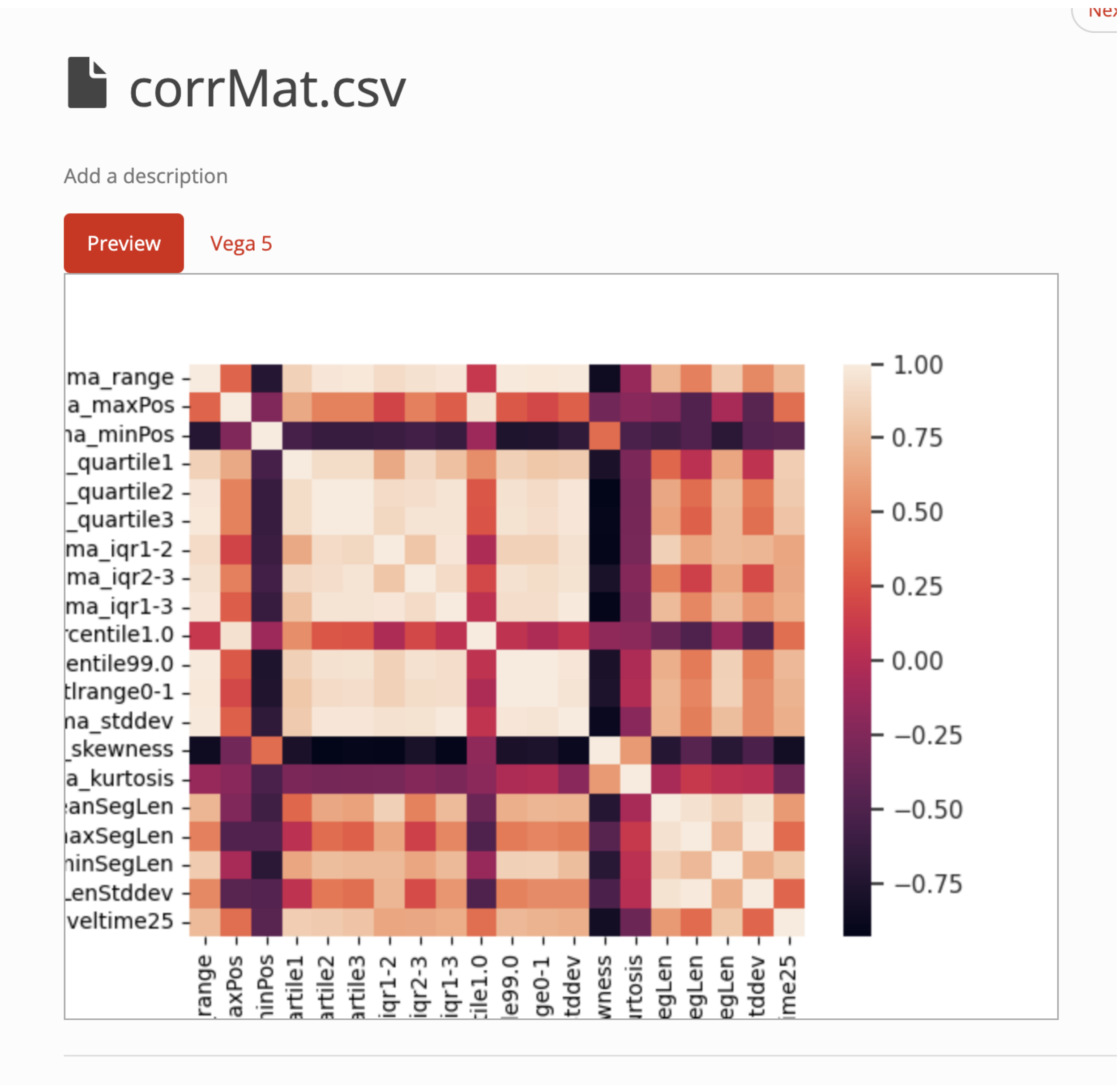
The aggregated features will be overwritten and stored back to the original dataset. The correlation matrix will be stored as CSV file and overwritten/uploaded back to the original dataset.

Note that we use pyclowder.files.pload_to_dataset() to upload; We use more fundamental methods on connector for deleting files by directly requesting the endpoints: connector.delete(url, verify=connector.ssl_verify if connector else True).



```python
            # plot correlation matrix and attach to preview
            preview_filename_corr = "corrMat_heatmap.png"
            matrix = corrMat.round(2)
            plt.cla()
            sns.set(rc={'figure.figsize': (20, 15)})
            sns.heatmap(matrix, annot=False)
            plt.savefig(preview_filename_corr)
            pyclowder.files.upload_preview(connector, host, secret_key, corrMat_file_id, preview_filename_corr)


if __name__ == "__main__":
    extractor = CorrMatrixExtractor()
    extractor.start()
```

We can attach certain output of extractors as a **visualization** to a file or a dataset. The previewer of clowder will pick it up and render it on the UI. To do that, we use the method pyclowder.files.upload_preview().



corrMat.csv

Add a description

Note: how the extractors will be triggered is decided in extractor_info.json

a. triggered by audio file uploads

```
"process": {
      "file": [
          "audio/*"
      ]
  },
```

b. for manual triggering:

```
"process": {
        "dataset": [
                "file.manual"
         ]
 }
```

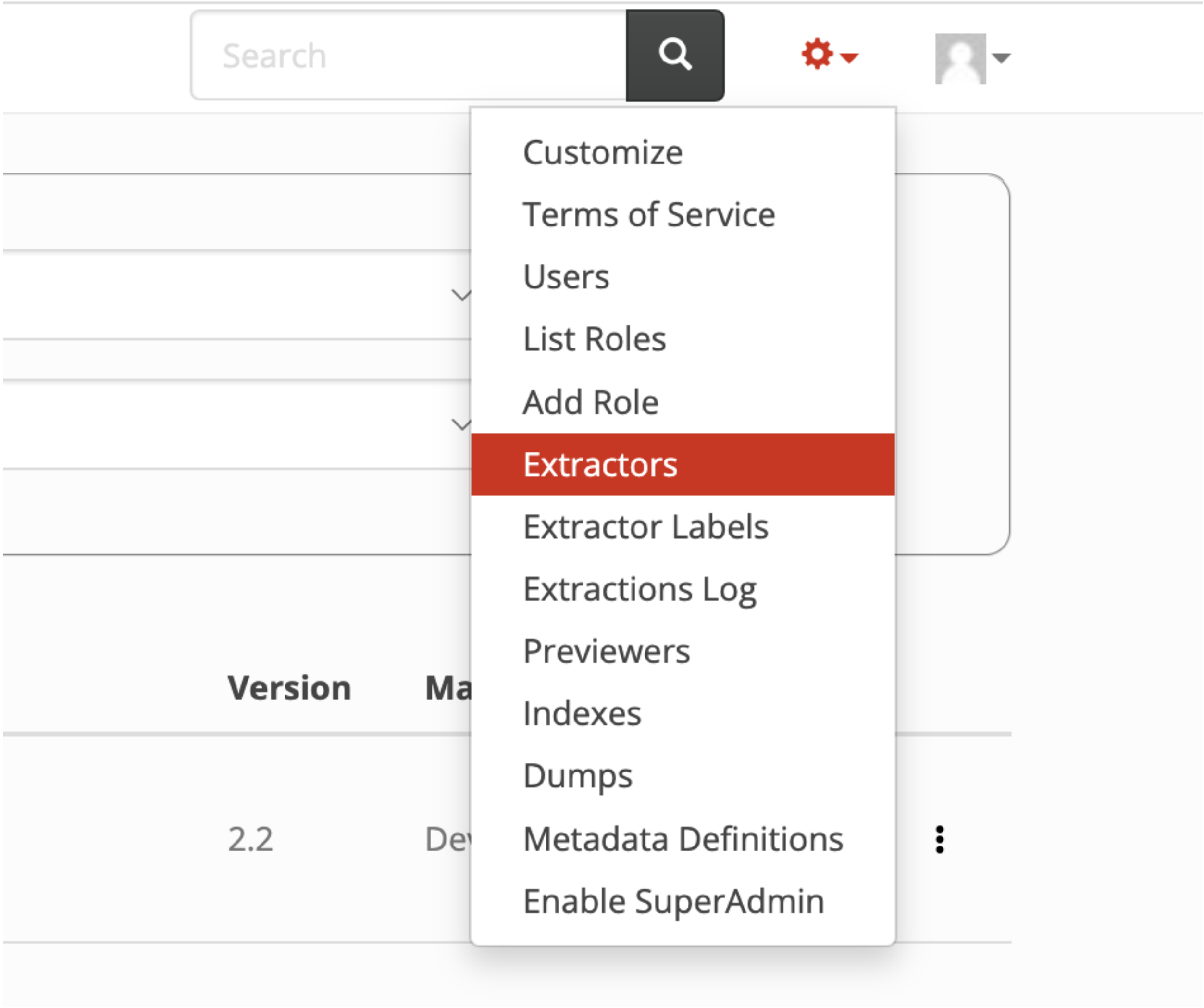
Deploy new extractors

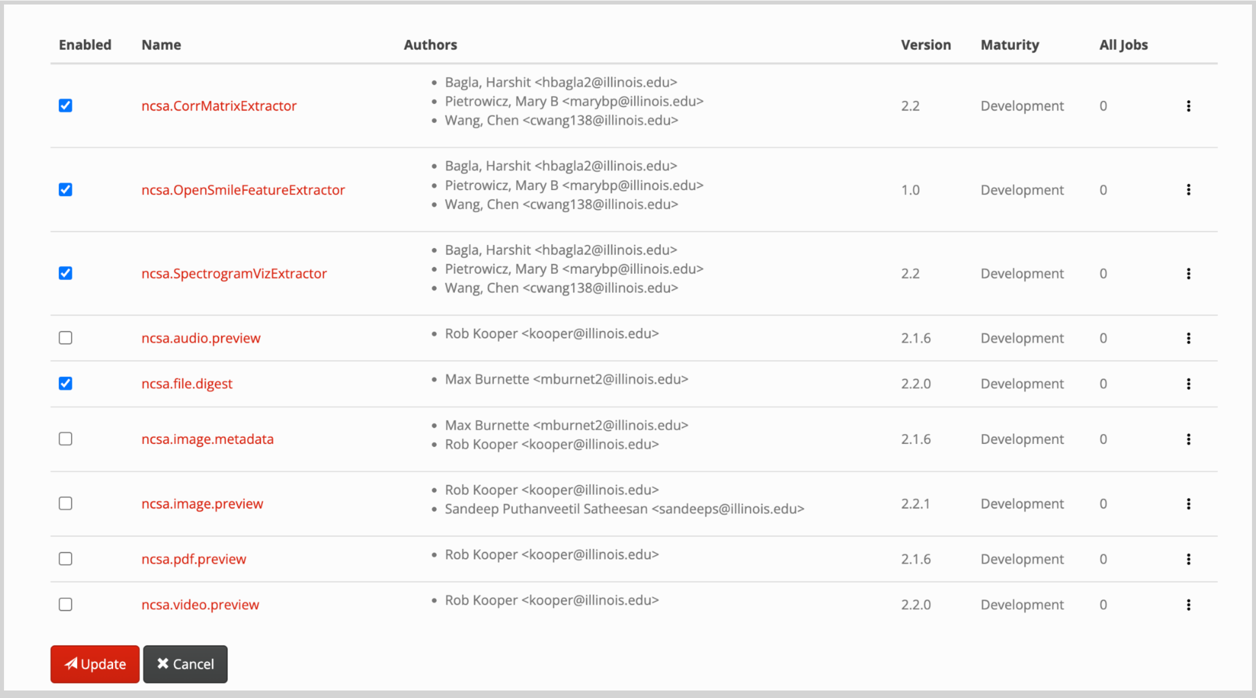
To deploy these extractors, please follow the below steps:

2. Individually navigate to each of the folders and build the docker image for each of the extractors. Make sure Dockerfile exists at the root level of each folder; if needed, you can modify the configuration of each image by editing Dockerfile. Type docker build . -t <image_name>:<image_version>. For example:
   ```
   docker build . -t opensmile-feature-extractor:2.2
   docker build . -t corr-matrix-extractor:latest
   docker build . -t spectrogram-visualization-extractor:1.0
   ```
3. For best practice, keep the **same** name and the version of the image in the extractor_info.json file inside each of the extractors folders you are located in, under the fields “name” and “version”.
4. Check if each image (extractor) is successfully built by docker images.
5. Edit docker-compose.extractors.yml file inside the clowder repository to include new extractors:
   ```
   <name_of_the_extractor>
           image  <image_name>:<version>
           restart: unless-stopped
           networks:
           - clowder
           depends_on:
           - rabbitmq
           - clowder
           environment:
           -
   RABBITMQ_URI=${RABBITMQ_URI:-amqp://guest:guest@rabbitmq/%2F}
   ```

Remember to follow the indentation.

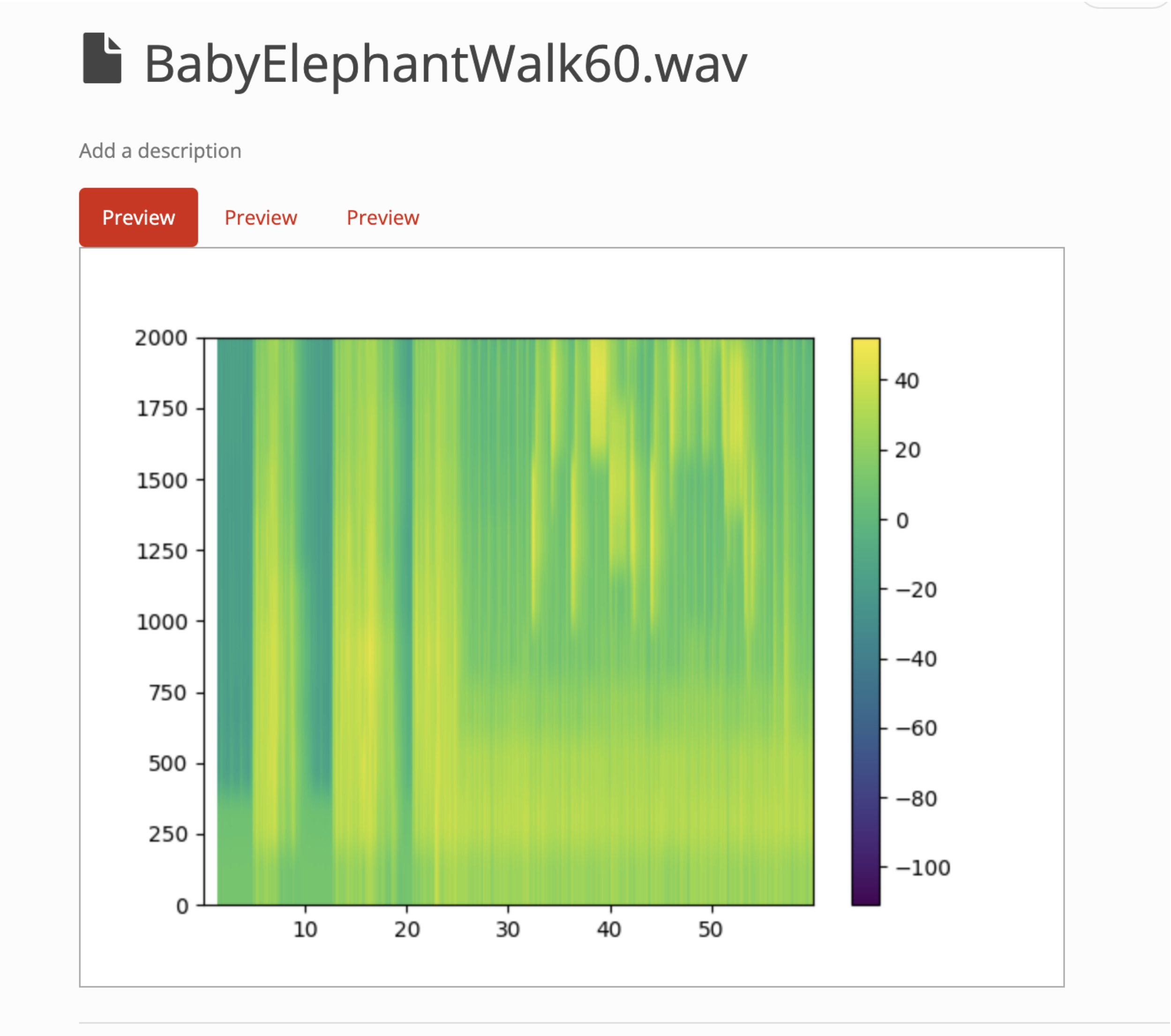
6. Run clowder `docker-compose -f docker-compose.yml -f docker-compose.override.yml -f docker-compose.extractors.yml up -d`
7. Enable newly added extractors:
   a. Login to clowder. Make sure you enable superAdmin privilege
   b. On the topbar locate “Extractors”
   c. Enable the newly added extractors and click “Update”



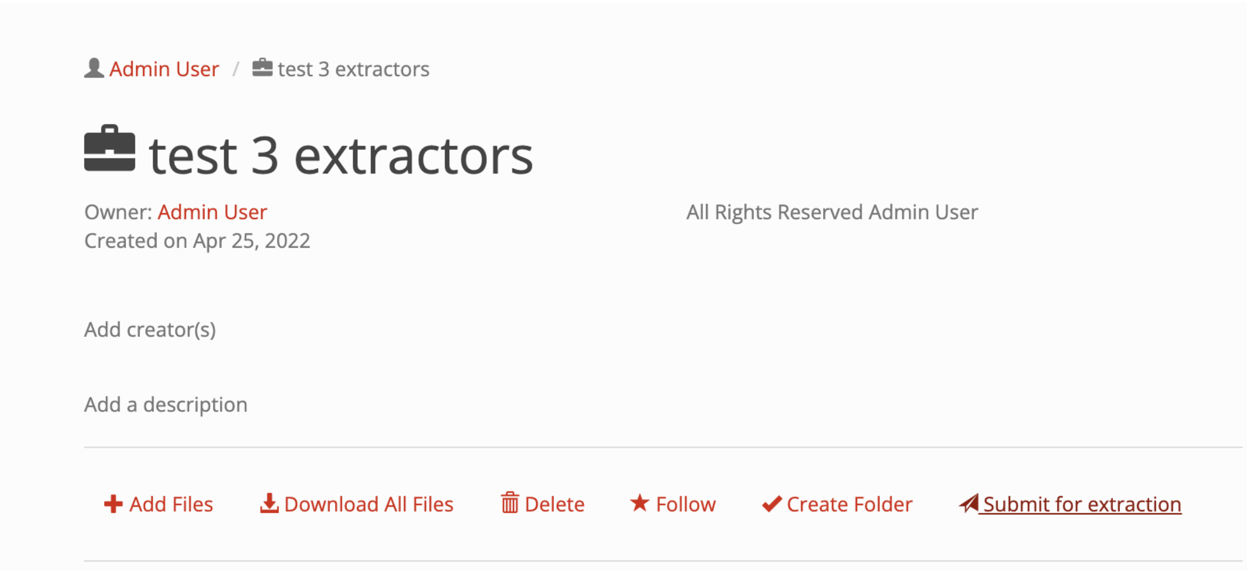


Enabled	Name	Authors	Version	Maturity	All Jobs	
☑	ncsa.CorrMatrixExtractor	• Bagla, Harshit <hbagla2@illinois.edu> • Pietrowicz, Mary B <marybp@illinois.edu> • Wang, Chen <cwang138@illinois.edu>	2.2	Development	0	⋮
☑	ncsa.OpenSmileFeatureExtractor	• Bagla, Harshit <hbagla2@illinois.edu> • Pietrowicz, Mary B <marybp@illinois.edu> • Wang, Chen <cwang138@illinois.edu>	1.0	Development	0	⋮
☑	ncsa.SpectrogramVizExtractor	• Bagla, Harshit <hbagla2@illinois.edu> • Pietrowicz, Mary B <marybp@illinois.edu> • Wang, Chen <cwang138@illinois.edu>	2.2	Development	0	⋮
□	ncsa.audio.preview	• Rob Kooper <kooper@illinois.edu>	2.1.6	Development	0	⋮
☑	ncsa.file.digest	• Max Burnette <mburnet2@illinois.edu>	2.2.0	Development	0	⋮
□	ncsa.image.metadata	• Max Burnette <mburnet2@illinois.edu> • Rob Kooper <kooper@illinois.edu>	2.1.6	Development	0	⋮
□	ncsa.image.preview	• Rob Kooper <kooper@illinois.edu> • Sandeep Puthanveetil Satheesan <sandeeps@illinois.edu>	2.2.1	Development	0	⋮
□	ncsa.pdf.preview	• Rob Kooper <kooper@illinois.edu>	2.1.6	Development	0	⋮
□	ncsa.video.preview	• Rob Kooper <kooper@illinois.edu>	2.2.0	Development	0	⋮

8. To test, upload a new .wave file. You should see previews



9. Besides automatic event (such as upload/download/delete/etc) triggering, we can also manually submit a file or a dataset to certain extractors. For example, we want to manually submit a dataset to correlation matrix extraction. First navigate to that dataset; under description click “submit for extraction”; then you should be able to click the “submit” button for the extraction you’d like to perform



Admin User / test 3 extractors

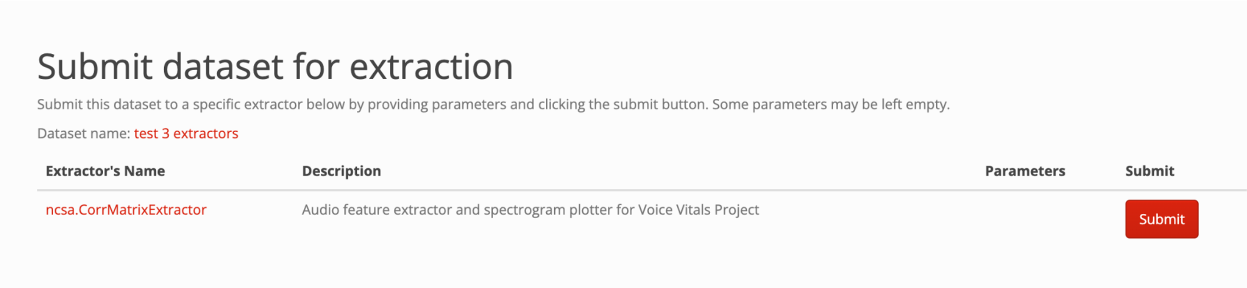
test 3 extractors

Owner: Admin User
Created on Apr 25, 2022

All Rights Reserved Admin User

Add creator(s)

Add a description



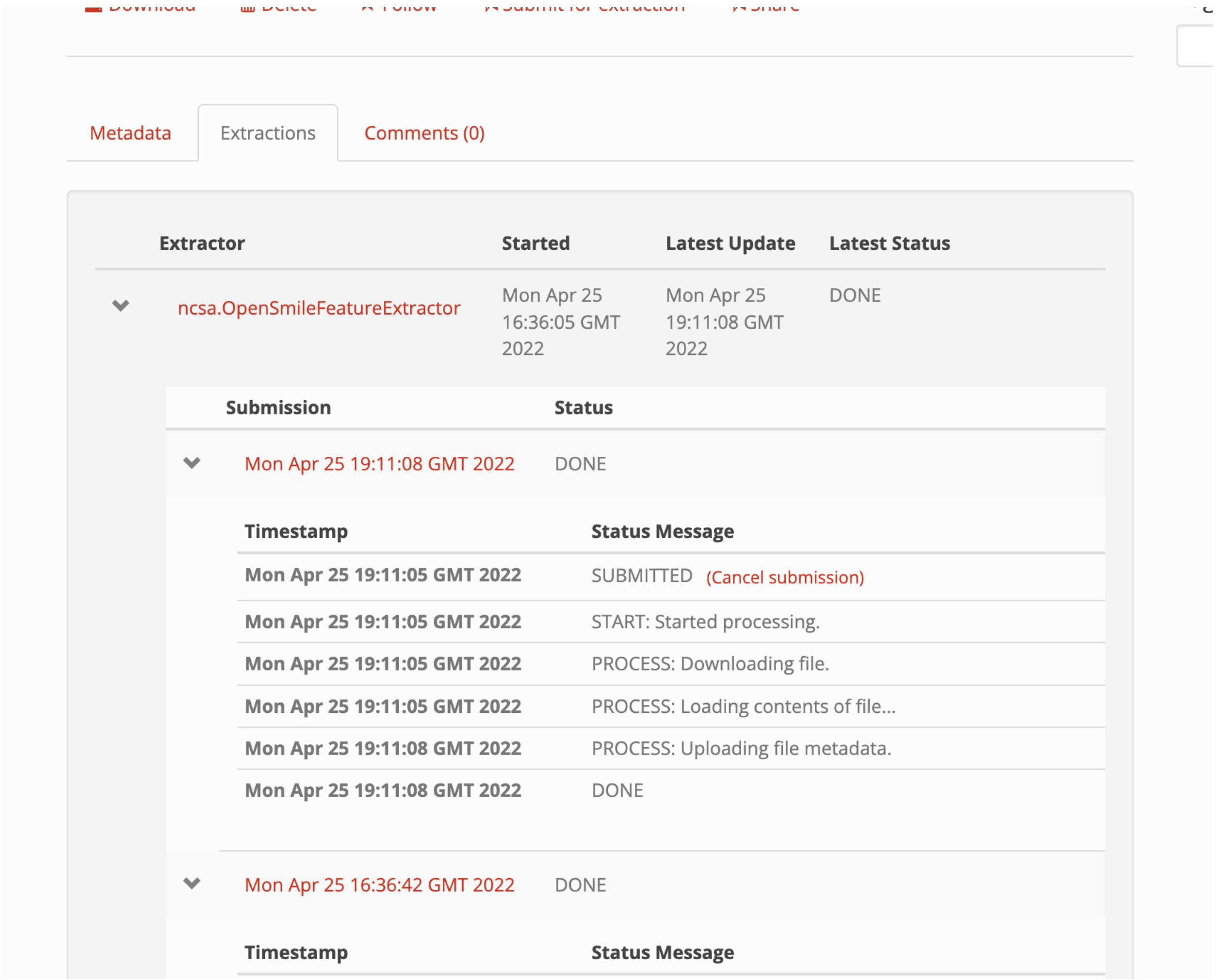
Submit dataset for extraction

Submit this dataset to a specific extractor below by providing parameters and clicking the submit button. Some parameters may be left empty.

Dataset name: test 3 extractors

Extractor's Name	Description	Parameters	Submit
ncsa.CorrMatrixExtractor	Audio feature extractor and spectrogram plotter for Voice Vitals Project		Submit

10. To check the status of extraction, you first navigate to the specific file/dataset you would like to check. Under the tab “Extractions” it will list all the past extraction activities for each of the extractors.



Extractor	Started	Latest Update	Latest Status
ncsa.OpenSmileFeatureExtractor	Mon Apr 25 16:36:05 GMT 2022	Mon Apr 25 19:11:08 GMT 2022	DONE

Submission	Status
Mon Apr 25 19:11:08 GMT 2022	DONE

Timestamp	Status Message
Mon Apr 25 19:11:05 GMT 2022	SUBMITTED (Cancel submission)
Mon Apr 25 19:11:05 GMT 2022	START: Started processing.
Mon Apr 25 19:11:05 GMT 2022	PROCESS: Downloading file.
Mon Apr 25 19:11:05 GMT 2022	PROCESS: Loading contents of file...
Mon Apr 25 19:11:08 GMT 2022	PROCESS: Uploading file metadata.
Mon Apr 25 19:11:08 GMT 2022	DONE

Submission	Status
Mon Apr 25 16:36:42 GMT 2022	DONE

Timestamp	Status Message

Additional notes:

1. If you experience any issue with file uploads and see the below error message in the console:
   [ERROR] - application - Could not create folder on disk /home/clowder/data/uploads/xx/xx/xx
   [ERROR] - application - Could not save bytes, deleting file xxx
   you can try this command:
   `docker-compose exec -u 0 clowder chmod 777 /home/clowder/data`
2. If running the extractor results in a "Failed to establish a new connection: [Errno 111] Connection refused", this is a Docker networking issue. Containers must be able to talk to each other (Clowder talking to RabbitMQ).
   To resolve, open `clowder/conf/application.conf` search for and set the RabbitMQ message queue URL:
   clowder.rabbitmq.clowderurl="http://host.docker.internal:9000"