Pipeline Tools Documentation

Release 0.59.1.dev111+gd9e22df

Mint Team

Oct 21, 2021

Contents

Note: This tool is still under active development, so there could be significant changes to its API.

This package provides utilities for retrieving files from the data storage service for the Human Cell Atlas, and for submitting an analysis bundle to the Human Cell Atlas Data Coordination Platform.

It also contains functions for interacting with Google Cloud Storage.

The steps in the submission process are as follows:

- Create analysis.json
- Create submission envelope and upload metadata
- Get URN needed to stage files
- Stage files by using the hca-cli
- Confirm submission

CHAPTER 1

Installing

Install it like this:

```
pip install git+git://github.com/HumanCellAtlas/pipeline-tools.git
```

You can use the cloud storage functions like this:

```
import pipeline_tools
from pipeline_tools import gcs_utils
bucket, object = gcs_utils.parse_bucket_blob_from_gs_link('gs://my-bucket/path/to/
↪object')
```

The rest of the package consists of scripts that are meant to be invoked from the command line as described below.

CHAPTER 2

Usage

2.1 get_analysis_workflow_metadata.py

Utility function fetches required information for creating submission to Ingest service, such as the Cromwell workflow metadata, the UUID of the analysis workflow, and the version of the given pipeline.

Invoke it like this:

```
get-analysis-workflow-metadata \
  --analysis_output_path ${analysis_output_path} \
  --cromwell_url ${cromwell_url} \
  --use_caas ${use_caas} \
  --caas_key_file ${caas_key_file}
```

All arguments are required, except the *–caas_key_file*, which is set to *None* by default.

2.2 create_analysis_metadata.py

Creates both analysis_protocol.json and analysis_process.json files, which are following different versions of the HCA metadata schema. For a full list of the HCA metadata schemas, check here.

Invoke it like this:

```
create-analysis-metadata \
  --analysis_id ${workflow_id} \
  --metadata_json ${metadata_json} \
  --input_bundles ${input_bundle_uuid} \
  --reference_bundle ${reference_bundle} \
  --run_type ${run_type} \
  --method ${method} \
  --schema_url ${schema_url} \
  --analysis_process_schema_version ${analysis_process_schema_version} \
```

(continues on next page)

(continued from previous page)

```
  --analysis_protocol_schema_version ${analysis_protocol_schema_version} \
  --pipeline_version ${pipeline_version} \
  --analysis_file_version ${analysis_file_version} \
  --inputs_file ${write_objects(inputs)} \
  --outputs_file ${write_lines(outputs)} \
  --format_map ${format_map}
```

All arguments are required.

2.3 create_envelope.py

Creates submission envelope and uploads metadata files.

Invoke it like this:

```
create-envelope \
  --submit_url ${submit_url} \
  --analysis_process_path analysis_process.json \
  --analysis_protocol_path analysis_protocol.json \
  --schema_url ${schema_url} \
  --analysis_file_version ${analysis_file_version}
```

All arguments are required.

2.4 get_staging_urn.py

Obtains URN needed for staging files. Queries ingest API until URN is available. The URN (Uniform Resource Name) is a long string that looks like this: hca:sta:aws:staging:{short hash}:{long hash}

It gets decoded by the hca-cli to extract the staging location and credentials needed to stage files.

Invoke it like this:

```
get-staging-urn \
  --envelope_url ${submission_url} \
  --retry_seconds ${retry_seconds} \
  --timeout_seconds ${timeout_seconds} > submission_urn.txt
```

envelope_url is required

2.5 get_files_to_upload.py

Gets a list of files to be uploaded(staged) by the HCA-CLI, writes the list to disk.

Invoke it like this:

```
get-files-to-upload \
    --files ${sep=' ' files} \
    --uploaded_files $uploaded_files
```

Both arguments are required.

2.6 confirm_submission.py

Confirms submission. This causes the ingest service to finalize the submission and create a bundle in the storage service.

Waits until submission status is "Valid", since submission cannot be confirmed until then.

Invoke it like this:

```
confirm-submission \
  --envelope_url ${submission_url} \
  --retry_seconds ${retry_seconds} \
  --timeout_seconds ${timeout_seconds}
```

envelope_url is required

CHAPTER 3

Testing

3.1 Running unit tests

To run unit tests, first create a virtual environment with the requirements:

```
virtualenv test-env
source test-env/bin/activate
pip install -r requirements.txt -r test-requirements.txt
```

Then, run unit tests from the root of the pipeline-tools repo like this:

```
bash test.sh
```

To run schema integration tests, do:

```
export TEST_SUITE="latest_schema"
bash test.sh
```

CHAPTER 4

API documentation

CHAPTER 5

Table of Contents