# MIDI Analysis Framework

# Downloading and Building the Project

## Code Location

<https://github.com/CBIIT/MIDI>

## Requirements

Java

Maven 3.8.5

## Building the Project

mvn clean compile assembly:single

## Build Product

A single jar is created in the target directory. This jar includes all dependencies. This jar is later renamed to midi-analysis.jar

Directory of C:\MIDI2\target

12/28/2022 05:44 AM <DIR> .

12/28/2022 05:44 AM <DIR> ..

12/28/2022 05:44 AM <DIR> archive-tmp

12/28/2022 05:44 AM <DIR> classes

12/28/2022 05:44 AM <DIR> generated-sources

12/28/2022 05:44 AM <DIR> maven-status

12/28/2022 05:44 AM 7,323,796 midi-analysis-1.0-SNAPSHOT-jar-with-dependencies.jar

# Running the Script to Build CVS files of DICOM Headers

# Using Google Buckets and Zip Files

This is the preferred method for running the script to create the files. No further preparation is needed other than copying the jar file and mapping file to the machine that is going to run the script. This method is 10 times faster than mapping the buckets and no expansion of the zip files is necessary.

## Run the Script Named RunProcess

java -classpath "/home/gustafss/midi\_app/midi-analysis.jar" gov.nih.nci.midi.RunProcess tcia\_ctp tcia midi\_3\_ctp none /home/gustafss/midi\_app/ nih-nci-cbiit-midi-dev2 midi\_3\_ctp\_dataset.tgz midibigqueryupload

Parameters

1. The name of the run, this will be used to name the output csv
2. The component being tested {midi,tcia,ctp}
3. The bucket containing the zip file
4. The path to the crosswalk file, none means no crosswalk
5. The path to the output directory, this is a directory that is used to build the CSV file before it is transferred to the output bucket by the program
6. The project name, this is used to provide authorization for the program to access the buckets
7. The input zip file name
8. The output bucket

# Using Mapped Buckets and Expanded Zip Files

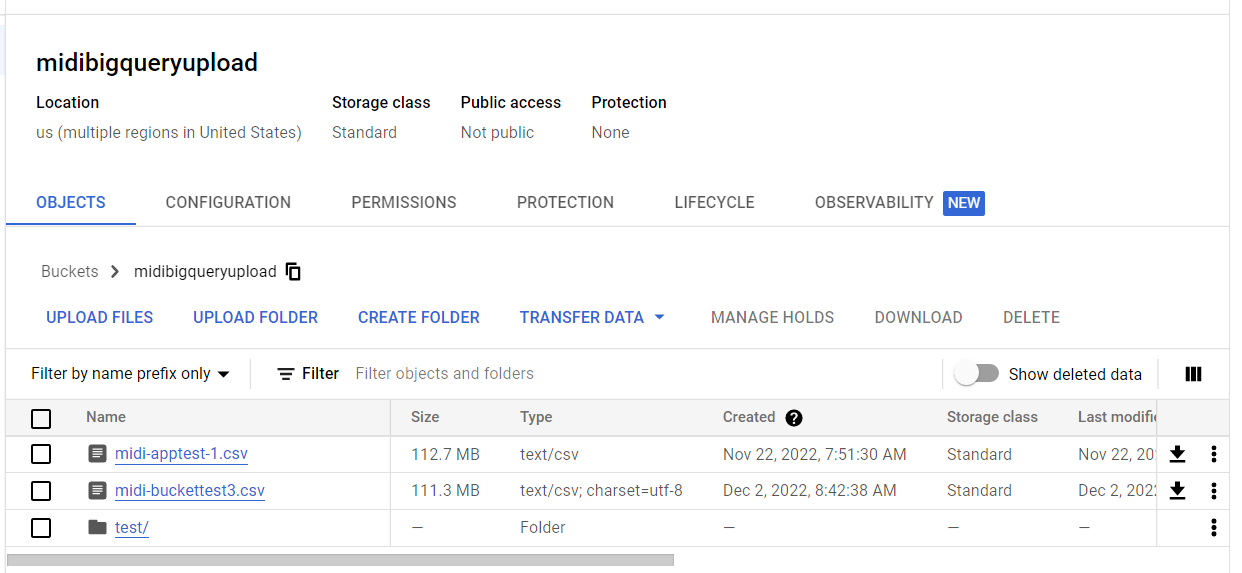
This is the legacy method for running the script to create the files. This method requires additional mapping of buckets to directories and will run much slower than the direct to bucket method

# Preparing the run environment

Use instance midi-linux-vm-analysis -1

Mapping to buckets

Output Bucket – There is a single output bucket that will hold the output csv files



Create the directory you want to use for output and map it

cd midi\_app/

mkdir output

gcsfuse midibigqueryupload /home/gustafss/midi\_app/output

Create the directory you want to use for input and map it to the bucket where the DICOM files are using the switch --implicit-dirs to discover existing directories

mkdir input

cd input

mkdir 21\_patient

gcsfuse --implicit-dirs 21patients\_10-19 /home/gustafss/midi\_app/input/21\_patient

## Run the Script Named RunProcess

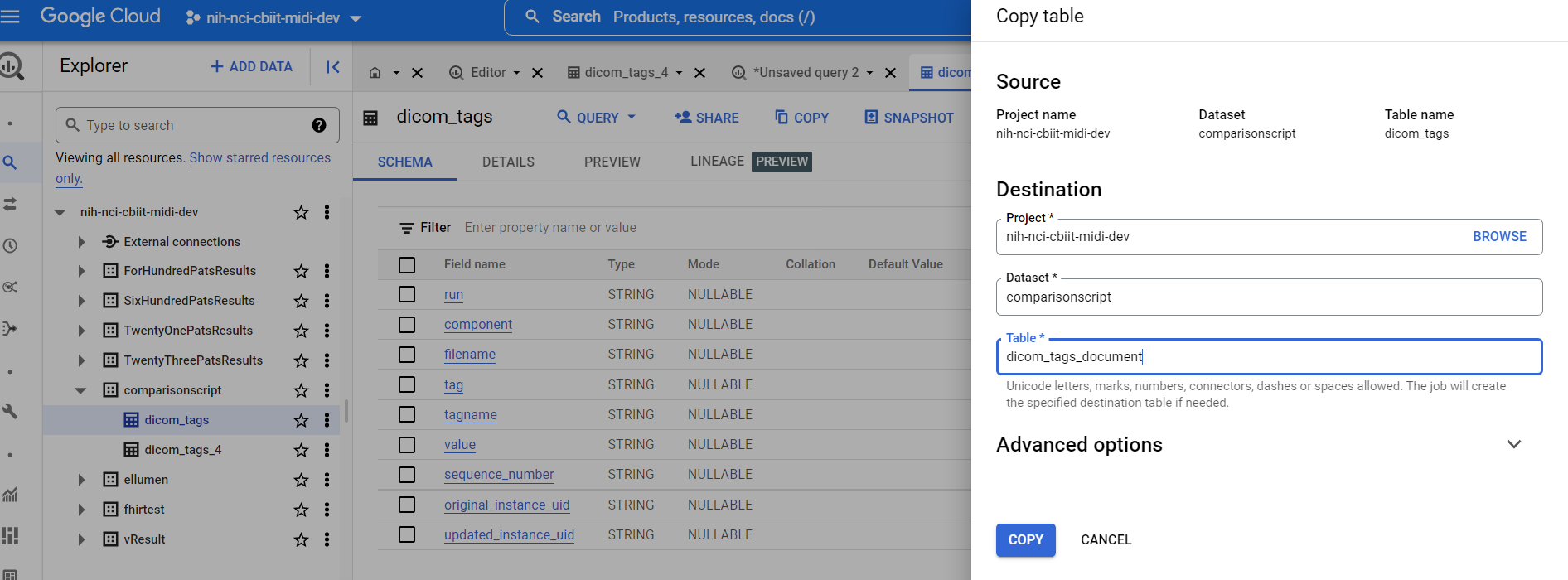
java -classpath "midi-analysis.jar" gov.nih.nci.midi.RunProcess buckettest3 midi /home/gustafss/input/21\_patient none /home/gustafss/midi\_app/output

Parameters

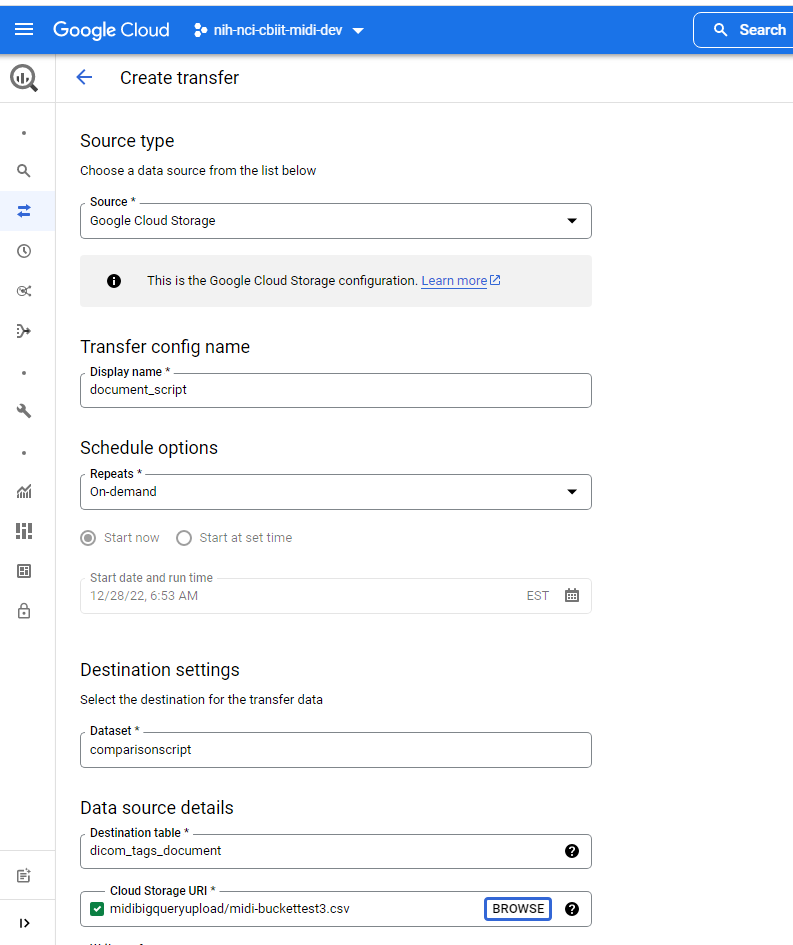
1. The name of the run, this will be used to name the output csv
2. The component being tested {midi,tcia,ctp}
3. The path to the input directory
4. The path to the crosswalk file, none means no crosswalk
5. The path to the output directory

# Load to BigQuery

Create a new table for the load based on dicom\_tags



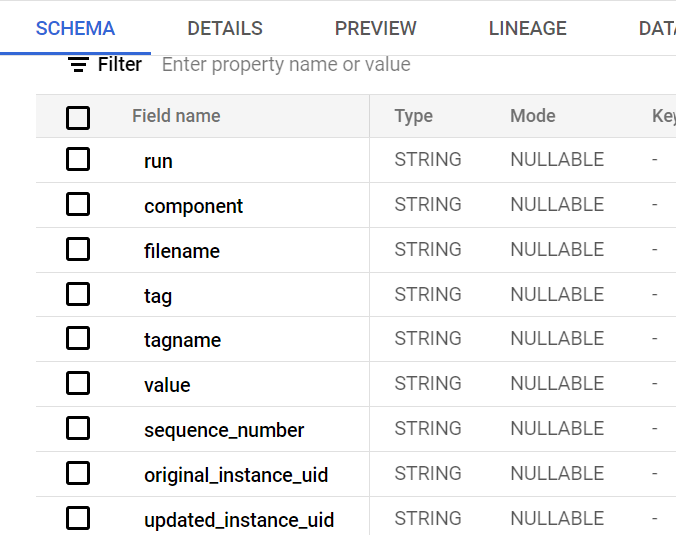
Load the data from the output file in the bucket by creating and running a data transfer



You can now use SQL to compare runs by joining the tables on the tags. Once you complete analysis you simply delete the table

# Analyzing Data in Big Query

Once the data is in Big Query the analysis can begin. The current basic design of the tables is shown below.



## Adding document support columns

In order to track missing documents on various runs a column denoting if the document is in the set that is being compared to. As an example here is the CTP output being compare to TCIA curated and later source images being compared to TCIA.

For the table created from CTP we add in\_curated\_doc it is set to Y when the tag is in a document that is in the curated data set. For the table created from source data we add in\_curated\_doc as well as in\_ctp\_docs which is set to Y when the tag is in a document that is in the ctp data set. Once the fields are set we can compare sets of documents where we know the document is in both sets by adding the condition as with in\_curated\_doc=’Y’ to compare CTP to TCIA curated. Below is an example of populating this column:

update `nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_ctp` run1

 set run1.in\_curated\_doc = 'Y'

where run1.original\_instance\_uid in

(select distinct run2.original\_instance\_uid

        from `nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_data` run2)

## Queries to compare runs

Below is a query to compare CTP to TCIA curated, note by using a full join we can see when tags are not in a dataset by looking for nulls

SELECT run1.original\_instance\_uid,

run1.filename,

run1.tag,

run1.value,

run2.original\_instance\_uid,

run2.filename,

run2.tag,

run2.value

FROM `nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_ctp` run1

full join `nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_data` run2

on run1.original\_instance\_uid=run2.original\_instance\_uid

and run1.tag=run2.tag

and run1.sequence\_number=run2.sequence\_number

where (run1.tag is null

or run2.tag is null

or run2.value is distinct from run1.value)

and run1.in\_curated\_doc='Y'

## Tables for MIDI3

A number of tables were created to support MIDI3, all have been truncated to protect PHI. The table structures are captured in the GitHUb repository in the sql directory in a filed name MIDI Tables.

* nih-nci-cbiit-midi-dev2.comparisonscript.midi-tcia\_ctp – The MIDI Pipeline output for the TCIA CTP file set.
* nih-nci-cbiit-midi-dev2.comparisonscript.midi-tcia\_source – The MIDI Pipeline output for the TCIA Source file set.
* nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_source – The TCIA Source file set.
* nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_ctp – The TCIA Source file set.
* nih-nci-cbiit-midi-dev2.comparisonscript.tcia-tcia\_data -– The TCIA Curated file set.

## Queries for MIDI3

The queries used to analyze the MIDI data tables. Note the Filenames queries need to be output to BigQuery tables as they are to large to export. The source sql of these queries are in the GitHub repository under sql/Queries

* Count Images in SOURCE not in MIDI.sql
* Filenames MIDI Source No UIDS Tags not null in MIDI dataset not in TCIA with Source Values.sql
* Filenames MIDI Source No UIDs Updated tags in MIDI not updated in TCIA.sql
* Filenames MIDI Source No UIDs Updated tags in MIDI vs Update tags in TCIA.sql
* Filenames MIDI Source No UIDs Updated tags in TCIA not updated in MIDI.sql
* MIDI CTP Images in Source not in MIDI.sql
* MIDI CTP Images in Source not in TCIA.sql
* MIDI CTP Tags not null in MIDI dataset not in TCIA with Source value.sql
* MIDI CTP Updated tags in MIDI from source not updated in TCIA including values.sql
* MIDI CTP Updated tags in MIDI from source vs Update tags in TCIA from source including values.sql
* MIDI CTP Updated tags in TCIA from source not updated in MIDI including values.sql
* MIDI Not Updated UIDS.sql
* MIDI Source No UIDs Tags not null in MIDI dataset not in TCIA with Source value.sql
* MIDI Source Duplicates.sql
* MIDI Source Images in Source not in CTP.sql
* MIDI Source Images in Source not in MIDI.sql
* MIDI Source Images in Source not in TCIA curated data.sql
* MIDI Source Images in Source not in TCIA.sql
* MIDI Source No UIDS Tags not null in MIDI dataset not in TCIA with Source Values (1).sql
* MIDI Source No UIDS Tags not null in MIDI dataset not in TCIA with Source Values.sql
* MIDI Source No UIDs Updated tags in MIDI not updated in TCIA.sql
* MIDI Source No UIDs Updated tags in MIDI vs Update tags in TCIA from source including values.sql
* MIDI Source No UIDs Updated tags in TCIA not updated in MIDI including values.sql
* MIDI Source Update In Curated for MIDI.sql
* MIDI Source Updated tags in MIDI from source not updated in TCIA including values.sql
* MIDI Source Updated tags in MIDI from source vs Update tags in TCIA from source including values (1).sql
* MIDI Source Updated tags in MIDI from source vs Update tags in TCIA from source including values.sql
* MIDI Source Updated tags in TCIA from source not updated in MIDI including values.sql
* MIDI Source Updated tags not null in MIDI from source not updated in TCIA including values.sql
* TCIA Curated not in Source.sql
* TCIA Duplicates.sql
* TCIA MIDI Duplicates.sql
* TCIA Source Duplicates.sql