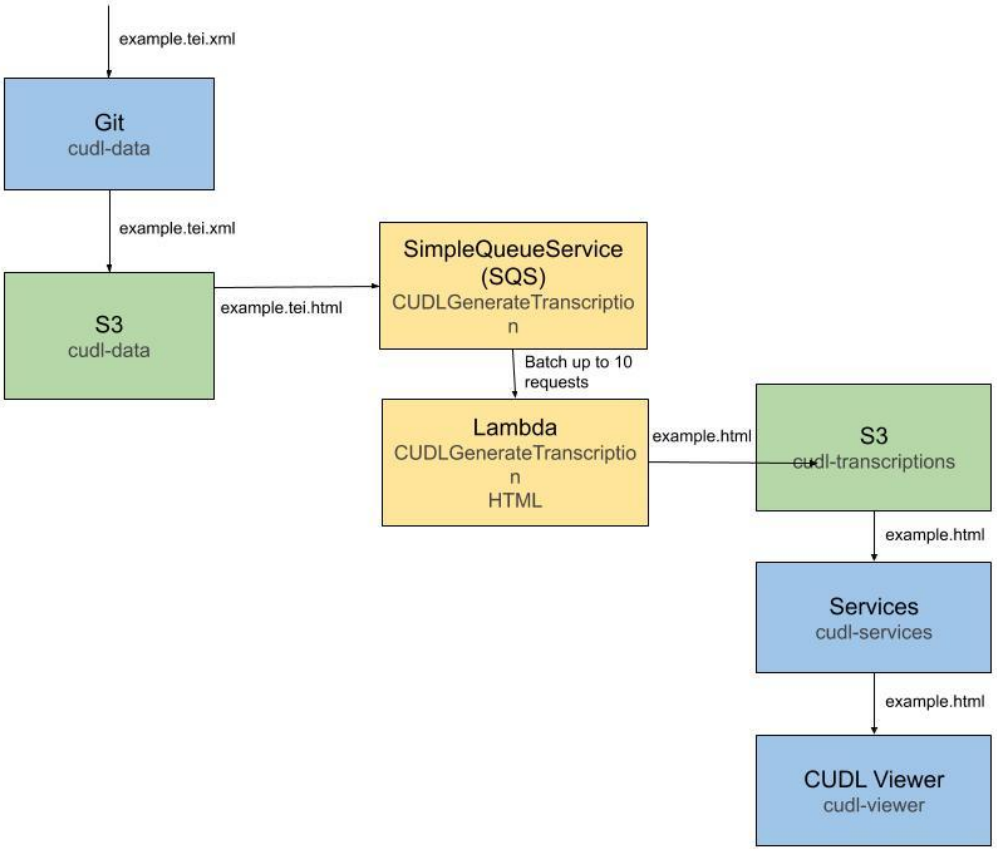
TEI to Transcription Processing



This process automatically generates transcriptions from TEI data committed to the cudl-data bitbucket git repository. These transcriptions should become available a few minutes later under the cudl-staging website.

What happens when a commit is made:

- A commit is made to cudl-data (master)

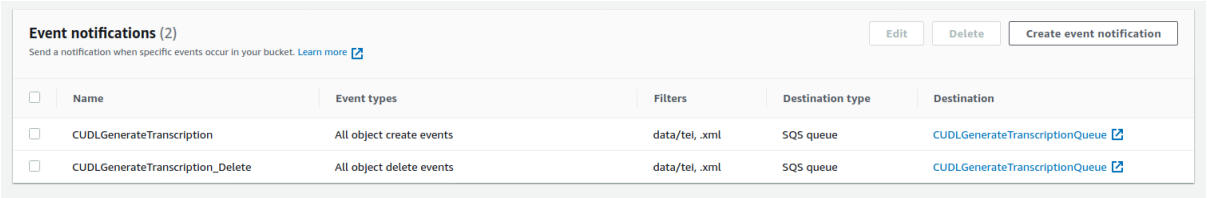
  https://bitbucket.org/CUDL/cudl-data/src/master/

- The bitbucket pipeline is triggered which copies data to the S3 bucket ‘cudl-data’:

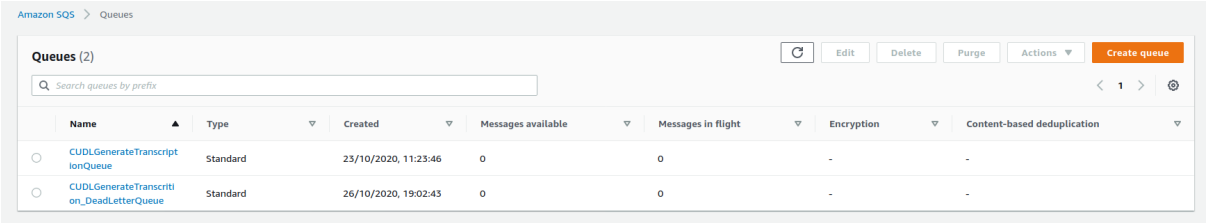
  https://s3.console.aws.amazon.com/s3/buckets/cudl-data?region=eu-west-1&tab=objects

- Any objects in this bucket that are changed trigger an event to be sent to the SQS (Simple Queue Service) Event Queue. You can see the trigger configuration under the s3 bucket properties as follows:

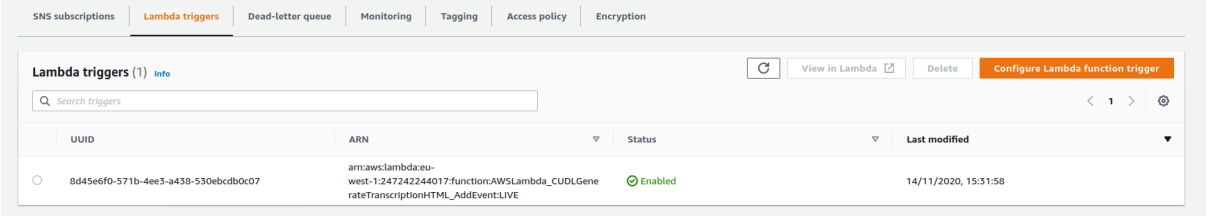
This shows that any create object event and any delete object event will be passed to the queue. There is one event for every object that is changed. So it a commit changes 100 files, there would be 100 events triggered.



- Under the queue interface you can see two queues are setup:



The queue these events are placed in is called ‘CUDLGenerateTranscriptionQueue’. This receives the events, and batches them up (if required) and then sends them to the Lambda function for processing. If there is an error processing the queue will try again, by sending the event back after a delay, a set number of times and then it adds the event to the dead letter queue. This is called ’CUDLGenerateTranscrition_DeadLetterQueue’. Where it is currently kept for a few days before being deleted.



You can see under the queue Lambda Triggers section the function it send the events to. This is called ‘**AWSLambda_CUDLGenerateTranscriptionHTML_AddEvent**’ and the part after the ’:’ is the **alias** for that function. We want the queue to only send events to the **LIVE** version of the function, as this is the tested version we know works as expected. See below for an explanation on how new versions of the lambda function are tested and released.

What does an event look like?

Events are JSON objects, an SQS event contains within it a number of SQSMessages (which is the batch functionality). There is a maximum of 10 messages. Each message corresponds to one event from S3. Within each message the body element contains the original JSON for the S3 event (Create or Delete event) from cudl-data.

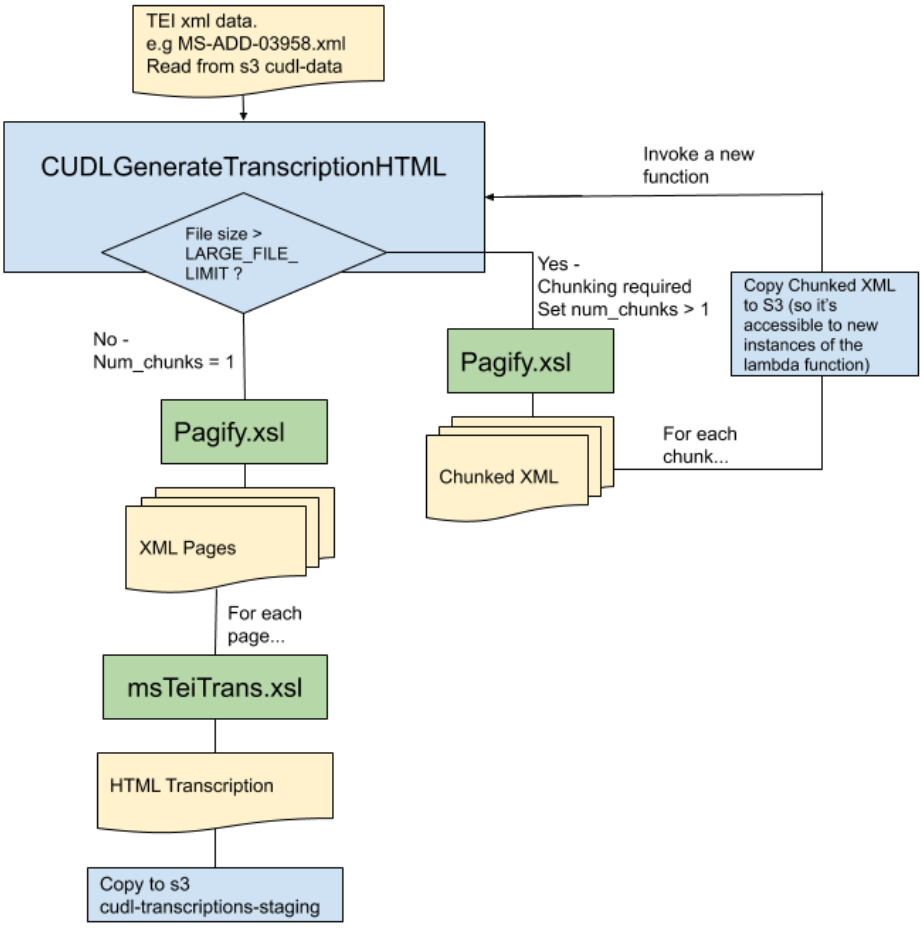
For example, an s3 event looks like:


```
{
  "Records":[
            {
            "eventVersion":"2.1",
            "eventSource":"aws:s3",
            "awsRegion":"eu-west-1",
            "eventTime":"2020-11-18T08:55:00.957Z",`
            "eventName":"ObjectCreated:Put",
            "userIdentity":{
                        "principalId":"AWS:AIDATTEGJH6Y2QMKG3XUR"
            },
            "requestParameters":{
                        "sourceIPAddress":"35.171.175.212"
            },
            "responseElements":{
            "x-amz-request-id":"045F39CA771367B6",

"x-amz-id-2":"fMuJx7C2pOGgKiuSAno/ivFq+RFDxsvlzaGSS0ZBl+8bW/zOnsZ65QvX+ELw4VT0K0jslXZ3ZT7CRMImju0A5f
zuapW6g9QP"
            },
            "s3":{
                        "s3SchemaVersion":"1.0",
                        "configurationId":"CUDLGenerateTranscription",
                        "bucket":{
                                    "name":"cudl-data",
                                    "ownerIdentity":{
                                                "principalId":"ATML35UCSMLB4"
                                    },
                                    "arn":"arn:aws:s3:::cudl-data"
                        },
                        "object":{
                                    "key":"data/tei/MS-ORCS-00004-00005/MS-ORCS-00004-00005.xml",
                                    "size":8981,
                                    "eTag":"796bec1837a6a53829a83198808aab21",
                                    "sequencer":"005FB4E16A31816465"
                        }
            }
    }
  ]
}
```


The information we are interested in is the type of event (create or delete) source s3 bucket and the object key.

What does the Lambda Function Do?



The code for the lambda function is at:

https://bitbucket.org/CUDL/transcription-lambda-transform/src/master/

For large files data is chunked because otherwise the limit of 15 mins runtime for a lambda function is reached. The pagify.xsl and msTeiTrans.xsl functions are used to convert the data into a set of html files.

Note: If the number of chunks produced by the chunking process is only 1 then the content is not pagified and cannot be chunked so it is processed as a single chunk (without spawning a separate process).

Bulk Processing

There are cases when all the data might need to be re-processed, for example changing the xslts, in which case we may want to do that locally and upload the contents to s3 directly.

There is an ant build file within the project that allows you to do this. See the README within the project for details.

Once the files are generated you can upload them directly to the cudl-transcriptions-staging bucket using the aws cli software.

See s3 sync command here:
https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-commands.html#using-s3-commands-managing-objects-sync

E.g. 'aws s3 sync <local path> <s3_path> --size-only
You can run with --dryrun to test what will be copied before running the command.

NOTE: Make sure the paths you are uploading to match your src file directory structure so you don't duplicate files.

Releases for the data:

Once the data is ready for release you can release it in the normal way, this will cause the data to be copied to the 'live' branch of the cudl-data git repository. Doing this will cause the bitbucket pipeline to run and sync data from cudl-transcriptions-staging to cudl-transcriptions-live.

This means no additional steps are needed to release the transcription data.

Releases for the code:

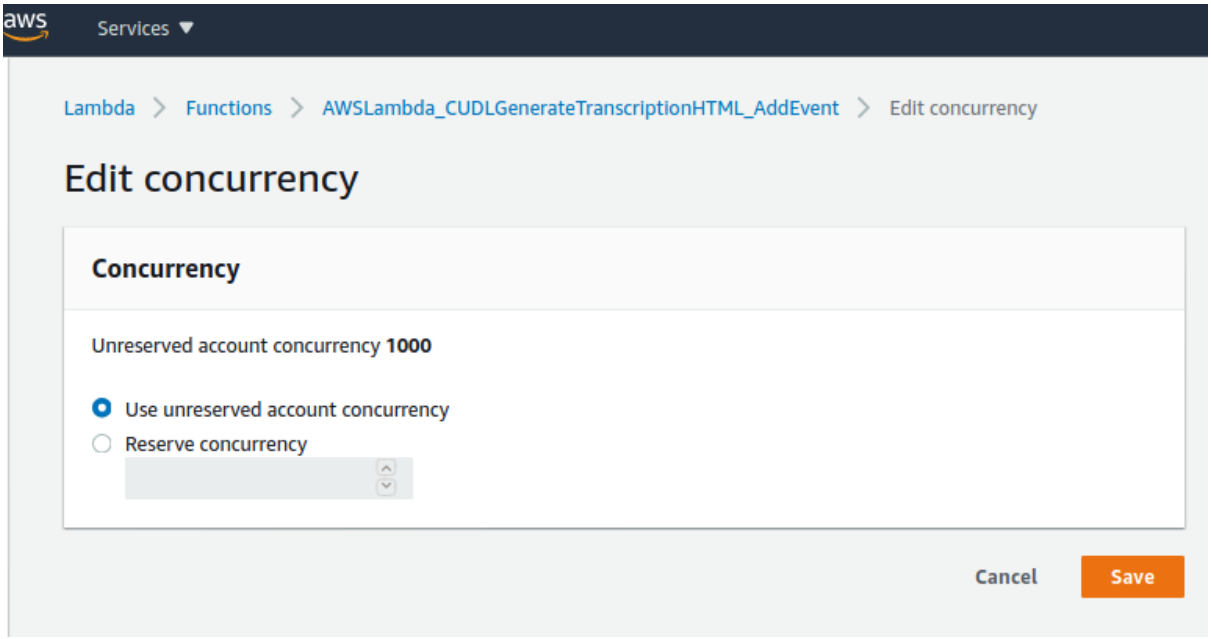
When the code is committed to the master branch of 'transcription-lambda-transform' it will cause a bitbucket pipeline to run. This does two things:

- It deploys the latest version of any web resources the transcriptions need to cudl-transcriptions-dev s3.
- It deploys the latest version of the lambda function code and tags a new version for it with the $LATEST alias. This sets the ENV variable to 'DEV' making the function use the cudl-transcriptions-dev bucket so it can be tested without impacting anything else.

  Once the code is tested it can be released to live by manually triggering the pipeline 'Publish to Live' from bitbucket which will tag the function with the LIVE alias and copy the web resources to the staging and live buckets.

This means that the function that's triggered with any data changes should be the LIVE version, but the DEV version is available for testing using data from cudl-transcriptions-dev. Once happy with the new version, run the bitbucket pipeline to make this version LIVE.

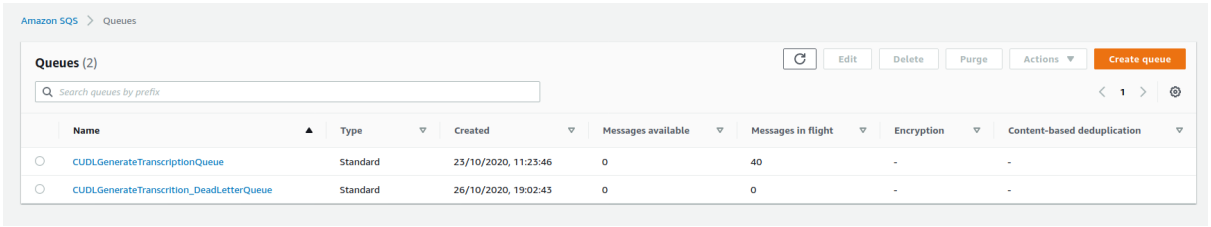
Concurrency
By default lambda has a limit for 1000 concurrent invocations (our account limit). There may be circumstances (when testing etc) where you want to limit this which can be done on the lambda configuration by assigning reserved concurrency for the function as below:



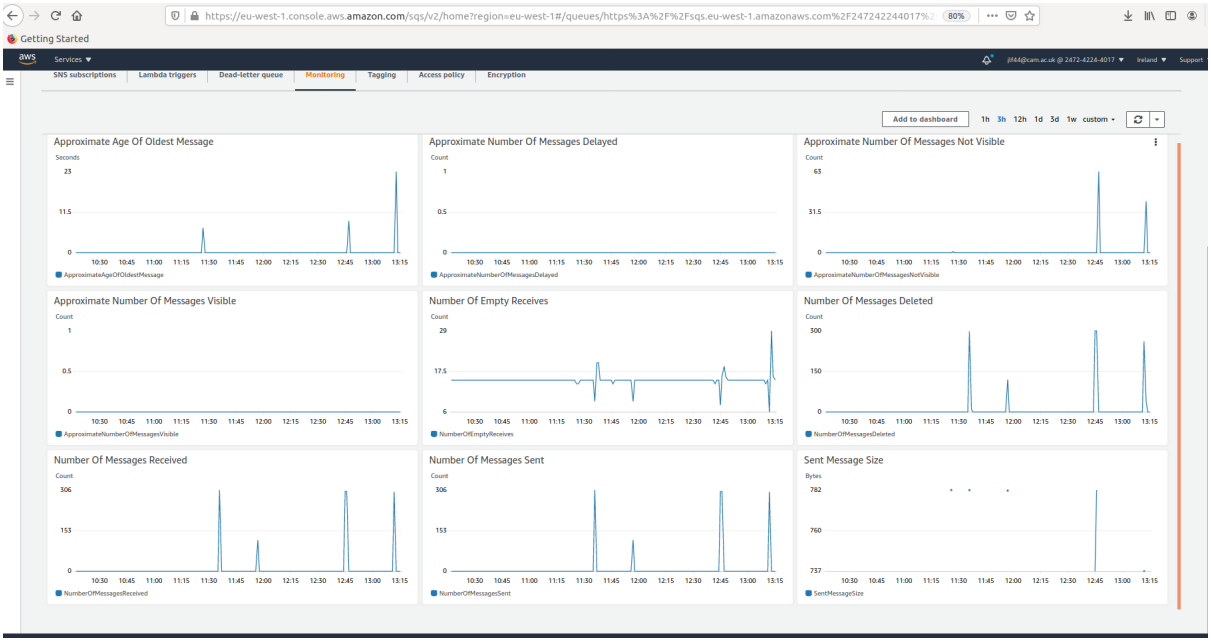
When testing I have used a reserved concurrency of 10.

Lambda: Logs

You can see the requests coming in on the SQS panel at:

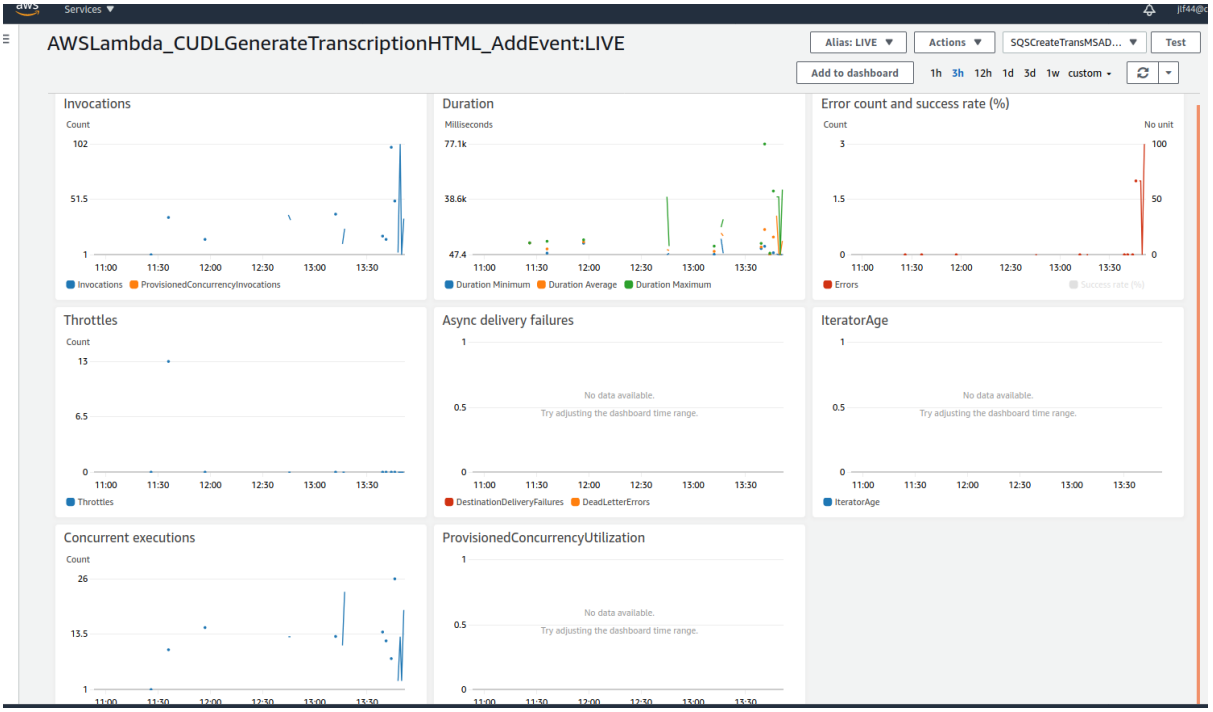


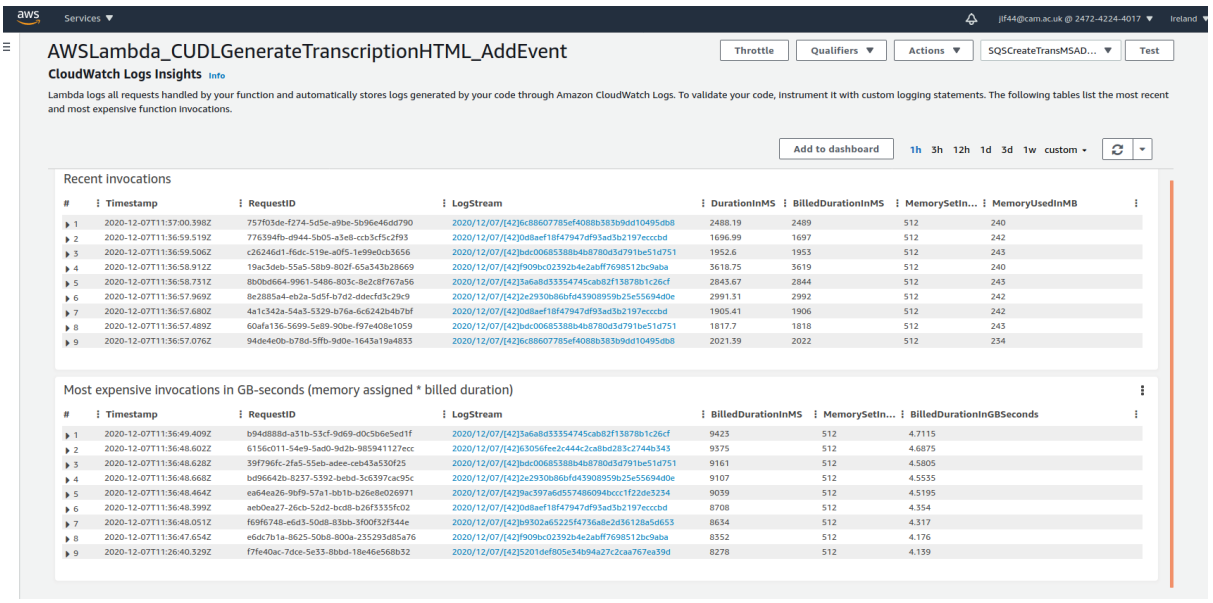
Graph view:



You can see graphs for the recent invocations of the lambda function at the monitoring tab, e.g:

https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions/AWSLambda_CUDLGenerateTranscriptionHTML_AddEvent?tab=monitoring



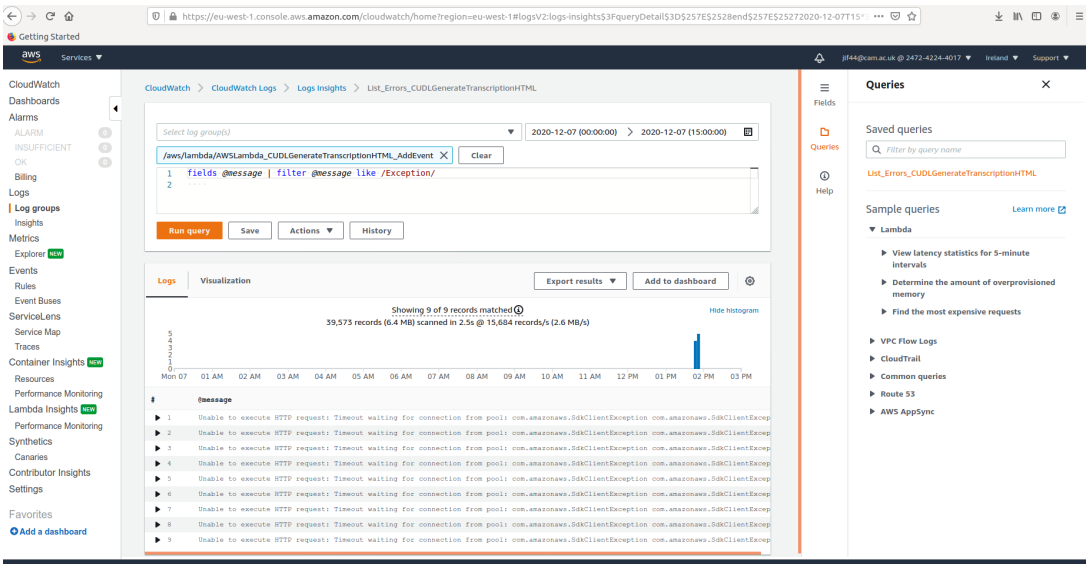


You can find logs at cloudwatch under:

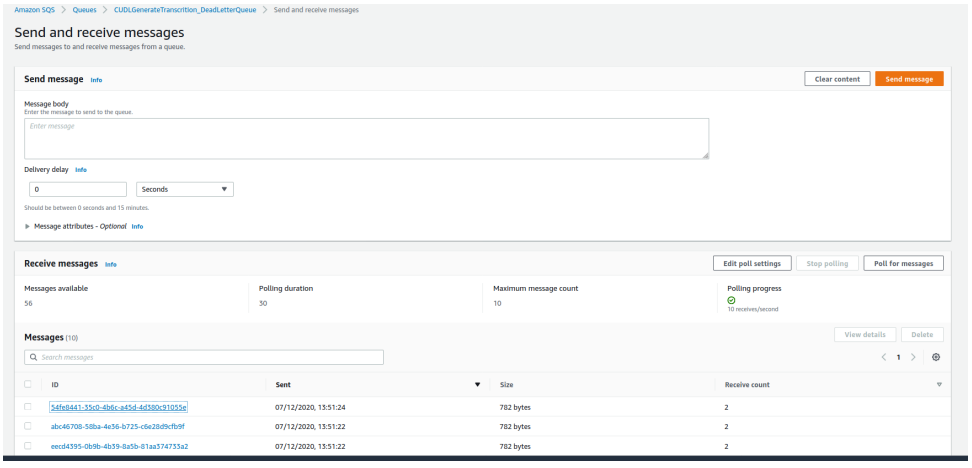
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#logsV2:log-groups/log-group/$252Faws$252Flambda$252FAWSLambda_CUDLGenerateTranscriptionHTML_AddEvent

These show the output of the latest function runs, including intermediate files generated etc. If there are any errors you can debug the causes here. Remember that chunked data will cause separate invocations so will have separate logs.

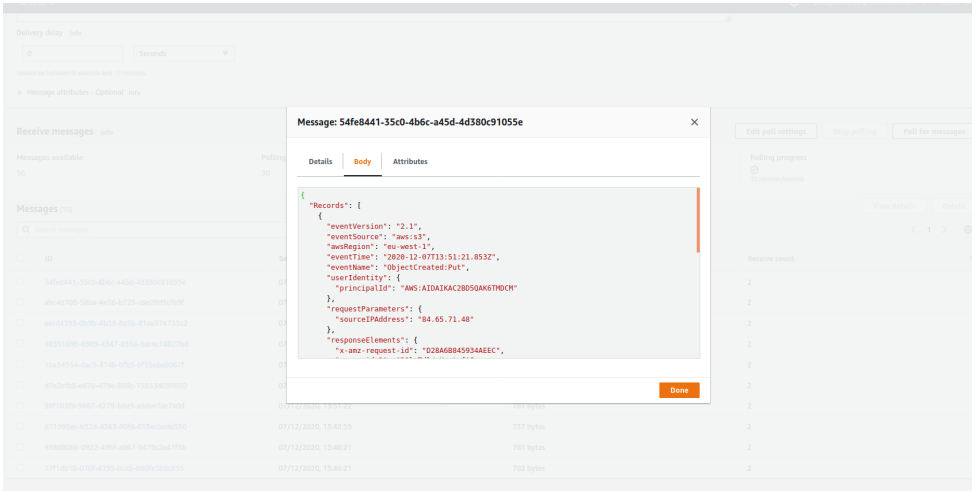
You can specifically look in the Log Insights section and use the saved query to find the log files for any errors:



You can view individual messages in a queue by going to the SQS Queue and selecting “Send and Receive Messages”, then “Poll for messages”.



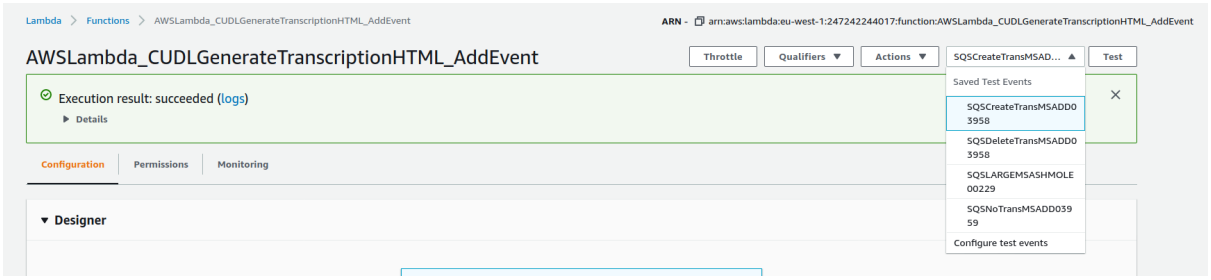
And you can view the body of that message:



Lambda: Tests

The function itself has tests that are configured under the configurations tab:

https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions/AWSLambda_CUDLGenerateTranscriptionHTML_AddEvent?tab=configuration



This is setup with some sample events for data with and without transcriptions and for data that needs to be chunked or not. You can see the result in the cloudwatch logs and processed files will be under the “html/test” directory on cudl-transcriptions-dev. Source data is taken from the s3 data directory under the “test” folder.

This means tests can use a fixed set of data that’s separate to the live data set.