LINTOL README

Linux Set Up

This Challenge will require the use of python3, with pip3 package installer within a pipenv Virtual Environment

---

Installing python3

Open your Terminal
Check your current Python Version

$ python3 --version

If it does not recognise this command, then you are missing python3, in order to install it

$ sudo apt-get update
$ sudo apt-get install python3.7

Now repeat the python3 --version command and it should return python3.7

Installing pip3

Open your Terminal
Check your current Python Version

$ pip3 --version

If it does not recognise this command, then you are missing pip3, in order to install it

$ sudo apt-get update
$ sudo apt-get install python3-pip

Now repeat the pip3 --version command and it should return pip version and storage path

---

Installing pipenv

Open your Terminal
Check if you have pipenv installed

$ pipenv

If it does not recognise this command, then you are missing pipenv, in order to install it

$ pip3 install --user pipenv

Now repeat the pipenv command and it should return the command options

---

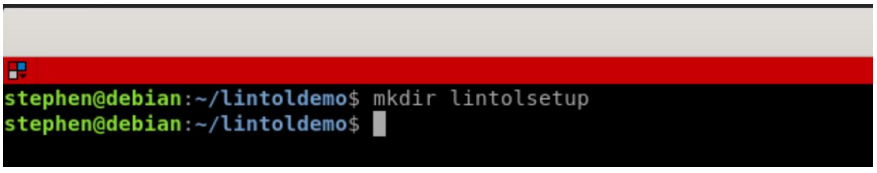
Challenge Set Up

Go to GitLab Repo and Fork Project to your own Workspace

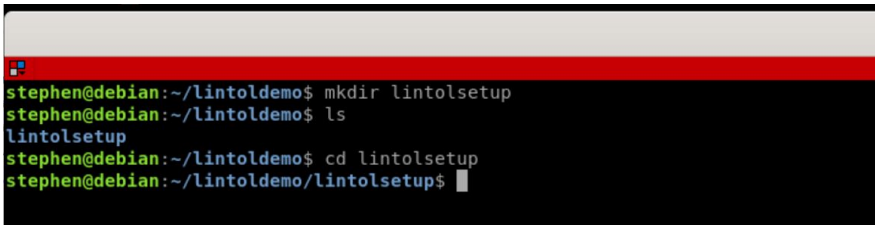
https://gitlab.com/lintol/coding-challenge-2021-exemplar-1/-/tree/master

Open Terminal

Create a new directory locally on your machine
$ mkdir "foldername"

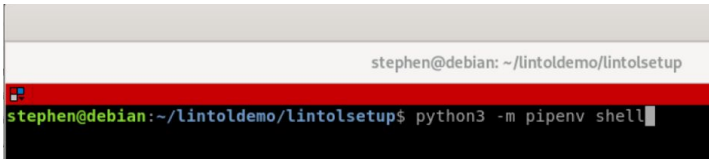


Clone the Forked Repo from Your Workspace into your new local directory
$ cd "foldername"
$ git clone "forked repo clone url" .

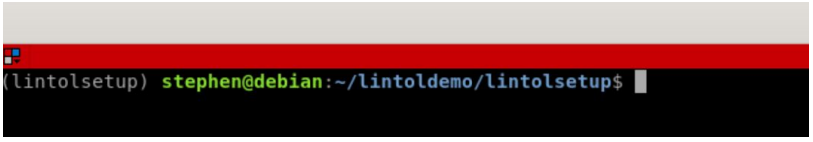


Your local directory should now contain the challenge files
$ ls

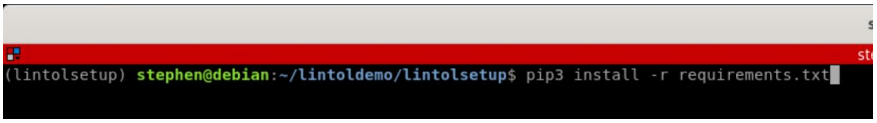
When you have the challenge files within your directory, it is time to install the dependencies, to do this we must start a new virtual environment
$ python3 -m pipenv shell



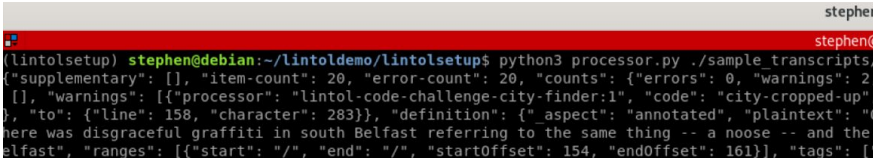
This will build the virtual environment and if successful update the terminal path (foldername) $



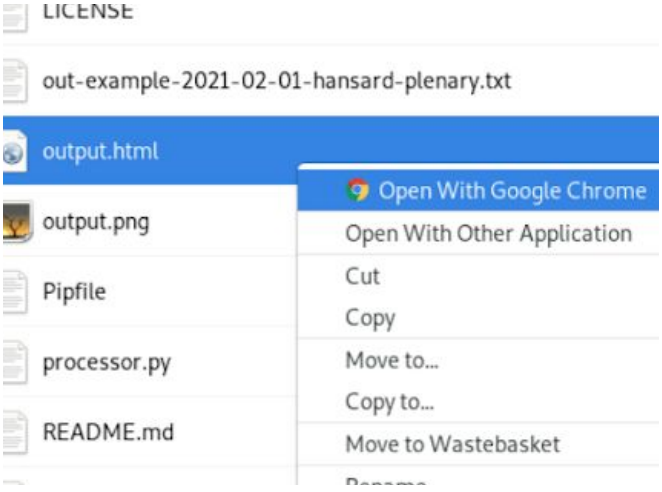
Once inside the virtual environment, you can now install the necessary dependencies (any errors rerun command)
(foldername) $ pip3 install -r requirements.txt

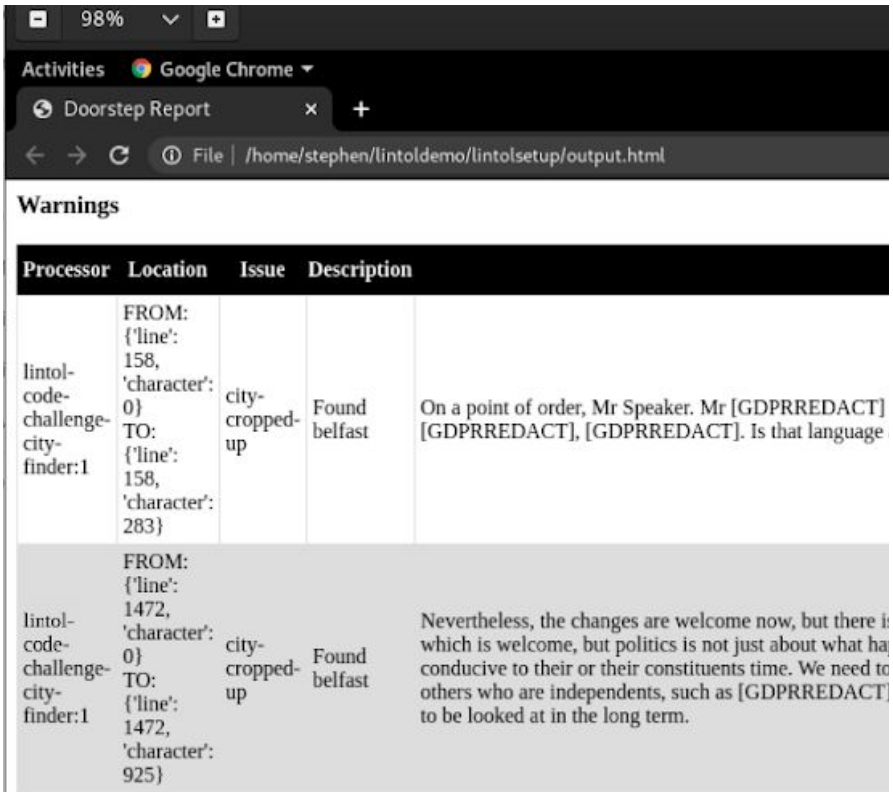


To test if you have set up the challenge enviroment successfully, you can now check by running the following command which will output the current demo processor results in the terminal
(foldername) $
python3 processor.py ./sample_transcripts/out-example-2021-02-01-hansard-plenary.txt



If you would like to see an html output of the processor results, you can run the following command which will generate an output.html file in local directory with challenge files, which you can open in chrome
(foldername) $
ltldoorstep -o html --output-file output.html process ./sample_transcripts/out-example-2021-02-01-hansard-plenary.txt processor.py -e dask.threaded





Warnings

Processor	Location	Issue	Description	
lintol-code-challenge-city-finder:1	FROM: {'line': 158, 'character': 0} TO: {'line': 158, 'character': 283}	city-cropped-up	Found belfast	On a point of order, Mr Speaker. Mr [GDPRREDACT] [GDPRREDACT], [GDPRREDACT]. Is that language
lintol-code-challenge-city-finder:1	FROM: {'line': 1472, 'character': 0} TO: {'line': 1472, 'character': 925}	city-cropped-up	Found belfast	Nevertheless, the changes are welcome now, but there i which is welcome, but politics is not just about what ha conducive to their or their constituents time. We need to others who are independents, such as [GDPRREDACT] to be looked at in the long term.

Carrying out the Challenge

Open your favourite Text Editor and add your local directory with the challenge files to your workspace
The magic of the processor is held within the processor.py file which is where you are free to edit and update to implement your new creative ideas

For example, a City Finder function has already been created for you to show off some of the possibilities, however we could make a quick example such as a Month Finder?
This will follow a similar structure to the City Finder function, instead we will be searching through a Months Array, rather than Cities.

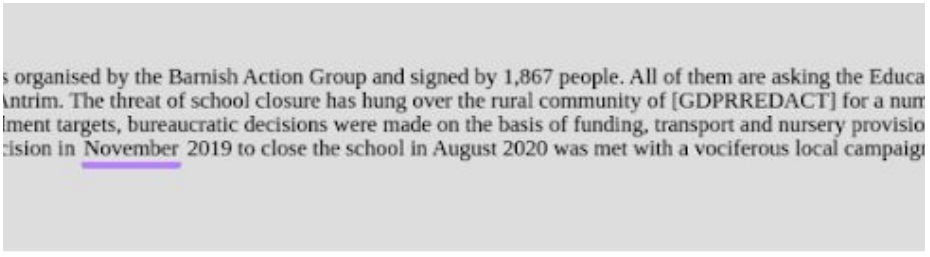
```
import logging
from dask.threaded import get

from ltldoorstep.processor import DoorstepProcessor
from ltldoorstep.aspect import AnnotatedTextAspect
from ltldoorstep.reports.report import combine_reports
from ltldoorstep.document_utils import load_text, split_into_pa

# We name some cities - we will do all our comparisons in lower
CITIES = ['armagh', 'belfast', 'derry', 'lisburn', 'newry', 'du

MONTHS = ['january', 'february', 'march', 'april', 'may', 'june

def city_finder(text, rprt):
    """
    Add report items to indicate where cities appear, and how o
    """

    # This doorstep utility splits a big text into paragraphs,
```

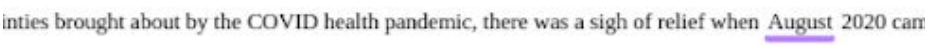
```
def month_finder(text, rprt):
    """
    Add report items to indicate where months appear, and how ofte
    """

    # This doorstep utility splits a big text into paragraphs, and
    # the punctuation. We do this splitting so that it is easier t
    # than scrolling through highlighted lines in one big document
    paragraphs = split_into_paragraphs(text)

    # This is our counter for months - we initialize every month's
    month_counts = {month: 0 for month in MONTHS}

    # Now we loop through the paragraphs - `enumerate` gives us a
    for para_n, (paragraph, line_number) in enumerate(paragraphs):

        # As mentioned above, we will search entirely in lowercase
        # a case insensitive search (others welcome too!)
        paragraph_lower = paragraph.lower()

        # We check this paragraph for month names...
        for month in MONTHS:
            # This gets a total count for the month
            month_occurrences = paragraph_lower.count(month)
```

Once we have completed the Month Finder function, we will have to call the updated function in the Class and define it in the output section

```
class CityFinderProcessor(DoorstepProcessor):
    """
    This class wraps some of the Lintol magic under the hood, that lets us plug
    our city finder into the online version, and create reports mixing and matching
    from various processors.
    """

    # This is the type of report we create - it could be tabular (e.g. CSV), geospatial
    # (e.g. GeoJSON) or document, as in this case.
    preset = 'document'

    # This is a unique code and version to identity the processor. The code should be
    # hyphenated, lowercase, and start with lintol-code-challenge
    code = 'lintol-code-challenge-city-finder:1'

    # This is a short phrase or description explaining the processor.
    description = "City Finder for Lintol Coding Challenge"

    # Some of our processors get very complex, so this lets us build up execution graphs
    # However, for the coding challenge, you probably only want one or more steps.
    # To add two more, create functions like city_finder called town_finder and country_finder,
    # then uncomment the code in this function (and remove the extra parenthesis in the 'output' line)
    def get_workflow(self, filename, metadata={}):
        workflow = {
            'load-text': (load_text, filename),
            'get-report': (self.make_report,),
            #'step-A': (city_finder, 'load-text', 'get-report'),
            'step-B': (month_finder, 'load-text', 'get-report'),
            # 'step-C': (country_finder, 'load-text', 'get-report'),
            'output': (workflow_condense, 'step-B') #, 'step-B', 'step-C')
        }
```

Then when we have updated the code, to use the processor in our own creative way, we can view the output by running the html command as before and refreshing the google chrome page
(foldername) $ ltldoorstep -o html --output-file output.html process sample_transcripts/out-example-2021-02-01-hansard-plenary.txt processor.py -e dask.threaded





The challenge is to see how creative you can get with the processor.py file, the html output is simply meant to be a way to view the results and the presentation of this will not be counted.

Repeat these processes in order to keep checking how your updates are coming along and until you are happy with what you have. When you have added in your updates, viewed the output in the html file and are happy to submit, you can then push to git lab

---

Submitting the Challenge

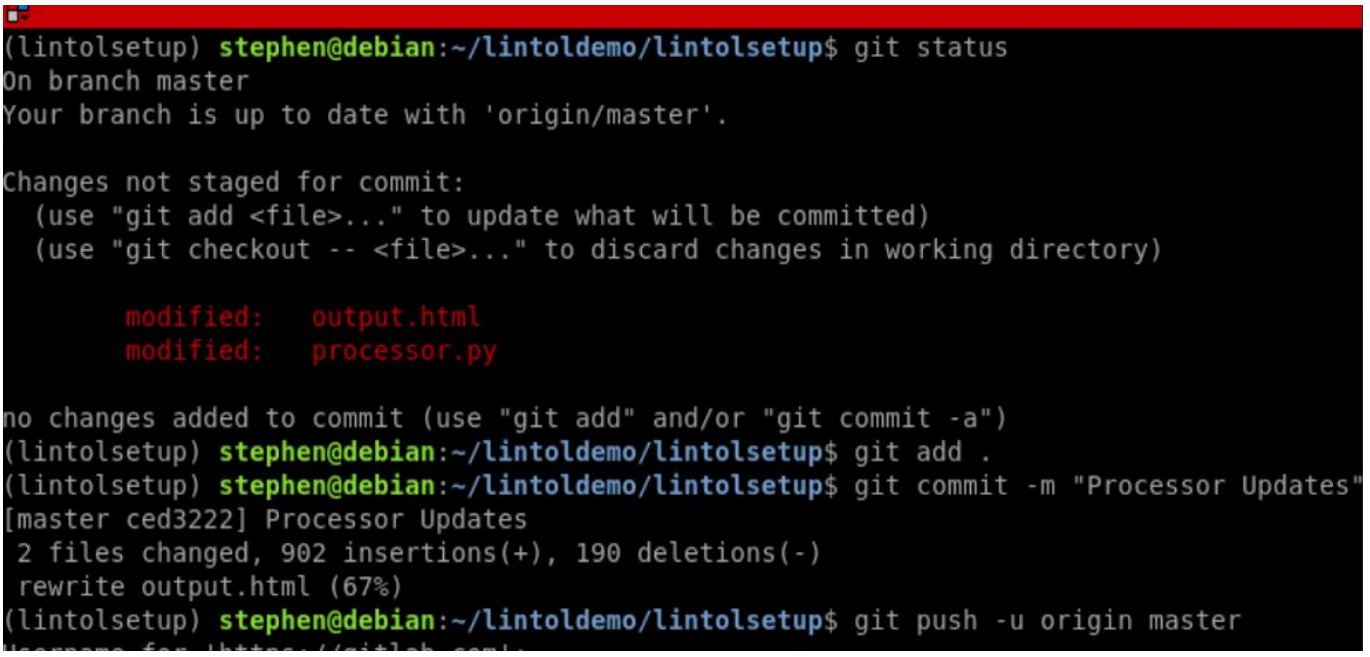
When you are happy with your testing locally and want to submit some code to see if it runs as you expect on gitlab pages simply git push to your new forked repo

Make sure you are on the master branch
$ git status

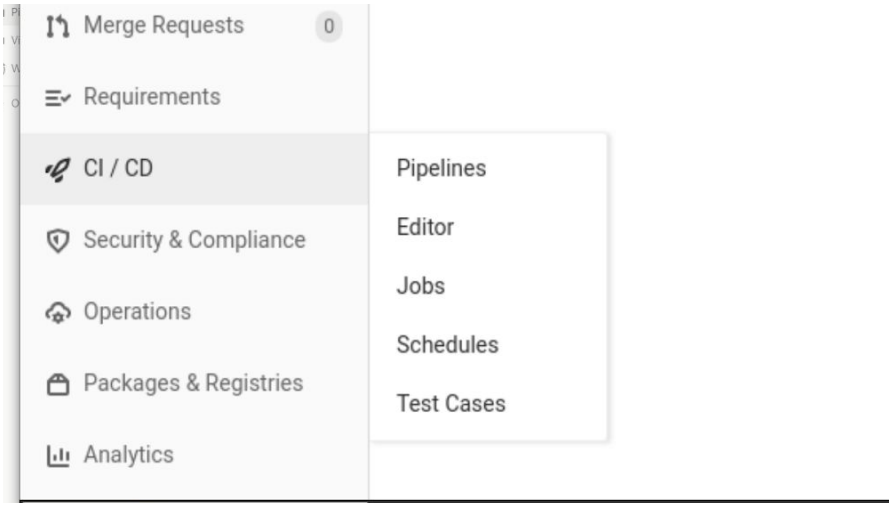
Then Add all of your changes
$ git add .

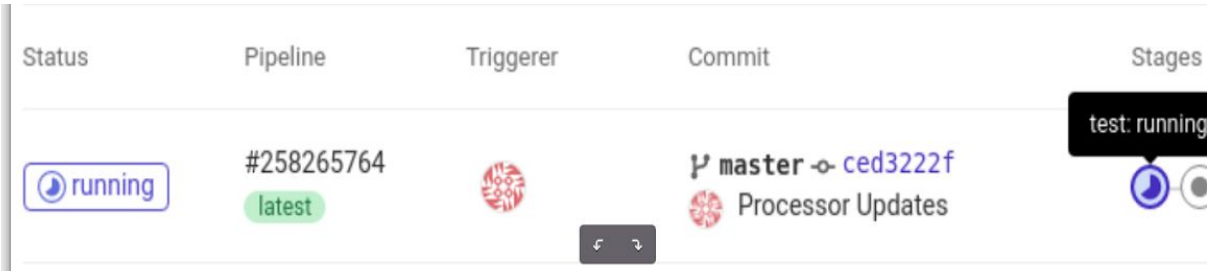
Write a commit message to help understand what is different about this commit
$ git commit -m "message"

Then push to your Repo
$ git push -u origin master

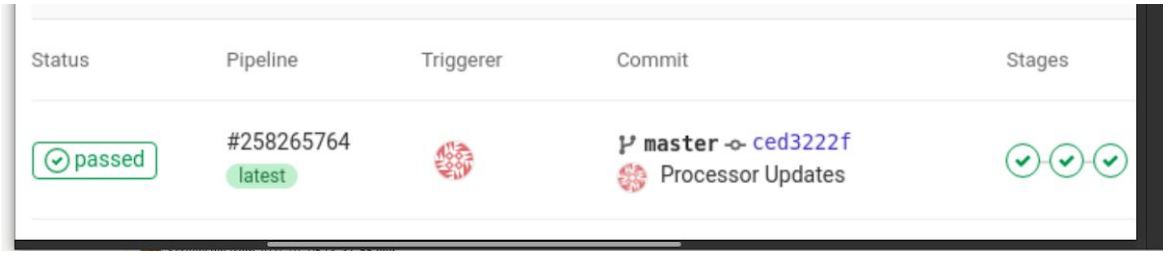


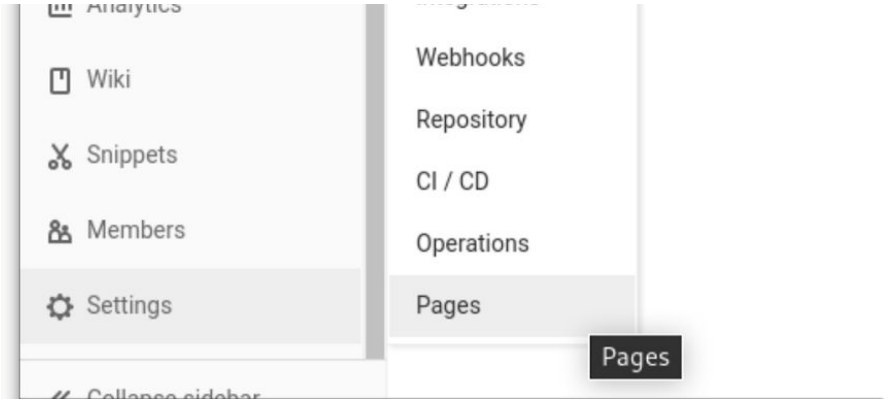
Navigate to your GitLab repo where you pushed the changes to and go to the CI/CD Tab in the Menu and the Pipelines tab in the submenu - here you can see if your push has been processed successfully by looking through each job stage



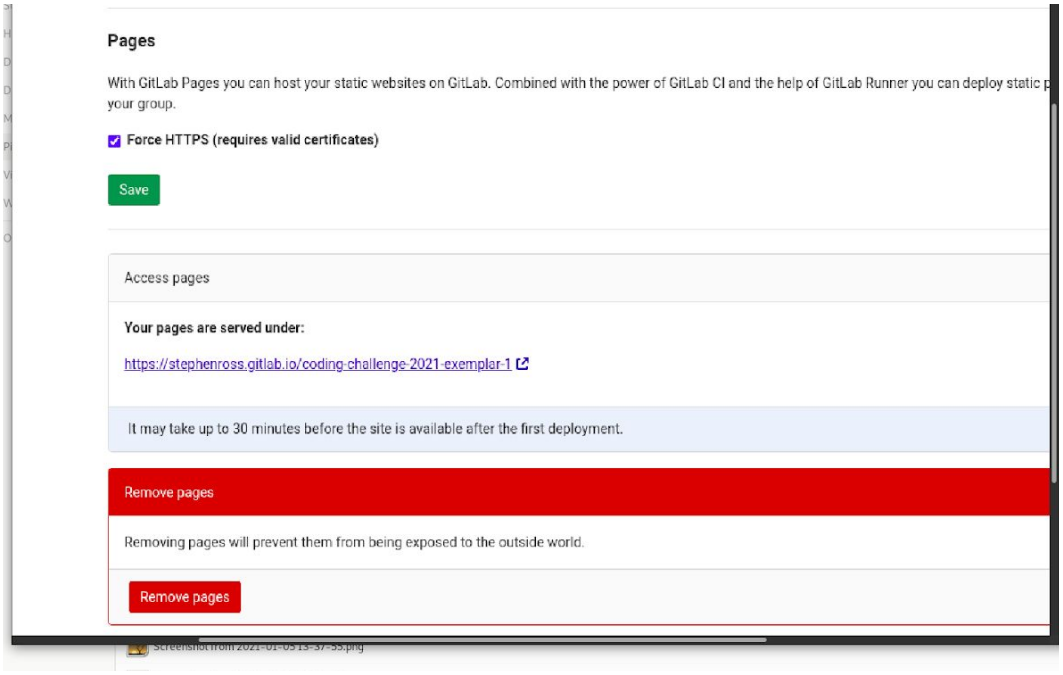


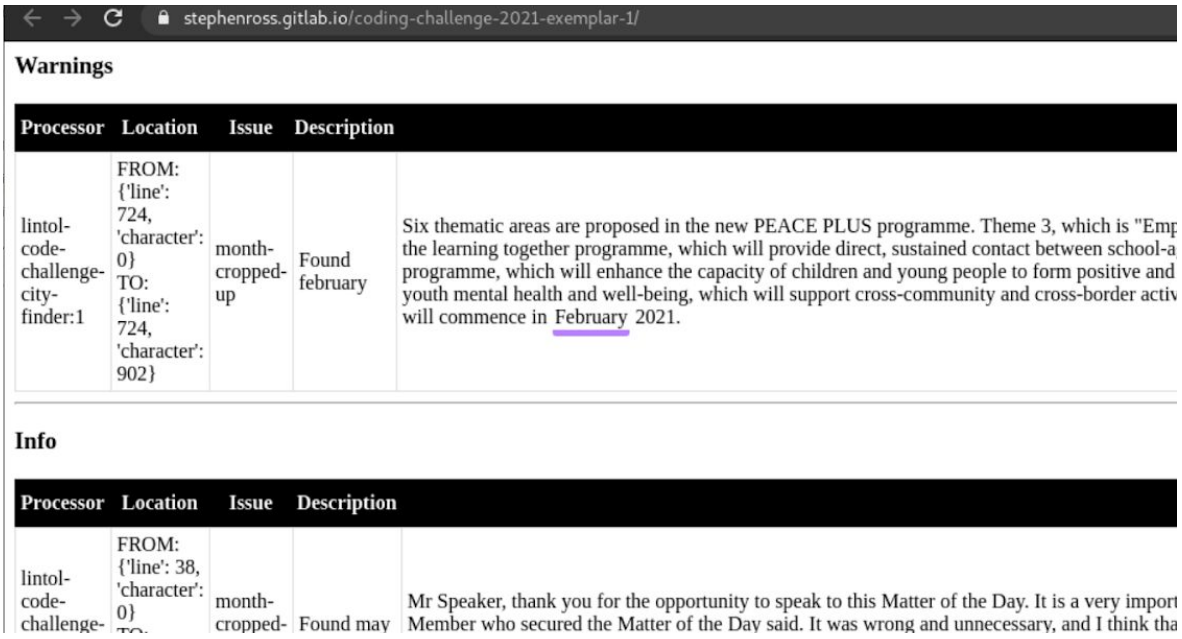
If all has ran successsfully and all stages show green ticks, it has been sucessuflly processed and should be on your Gitlab pages





To view your Gitlab Pages, go to the Settings Tab in the Menu and the Pages tab in the submenu and there should be an Access Pages link to a server where you can view the output.html of your process





Repeat these processes in order to keep checking how your updates are coming along and until you are happy with what you have