How to create a CWL Workflow for AWE

Quick start

Creating a CWL tools for pdftotext and wordcloud

Word Cloud install instructions can be found here: https://github.com/amueller/word_cloud

pdftotext

1. Setup execution environment
2. Create CWL tool

Create Docker container

1. open template.dockerfile and save as `<MY-NAME>.dockerfile`

```
# Build from base image
# FROM ubuntu:latest
FROM ubuntu:latest


# Update base images with latest patches
RUN apt-get update && apt-get upgrade -y


# Install dependencies
# RUN apt-get install -y \
#               ...
```

```
# Add additional commands
# RUN <COMMAND>

# Set entrypoint
# ENTRYPOINT [ "ls" ]
```

2. Add `poppler-utils` to the list of dependencies and save file

   `RUN apt-get install -y poppler-utils`

```
# Build from base image
# FROM ubuntu:latest
FROM ubuntu:latest


# Update base images with latest patches
RUN apt-get update && apt-get upgrade -y


# Install dependencies
RUN apt-get install -y poppler-utils
```

3. `docker build -t <name>:<tag> -f <MY-NAME>.dockerfile .`

   If you run into problems try the `--no-cache` option.

```
> docker build --no-cache  -t demo:2 -f Docker/Dockerfiles/demo2.dockerfile .
Sending build context to Docker daemon  221.2MB
Step 1/3 : FROM ubuntu:latest
...
```

4. `docker run --rm demo:2 pdftotext --help`

```
> docker run --rm demo:2 pdftotext --help
pdftotext version 0.41.0
Copyright 2005-2016 The Poppler Developers -
http://poppler.freedesktop.org
```

```
Copyright 1996-2011 Glyph & Cog, LLC
Usage: pdftotext [options] <PDF-file> [<text-file>]
  -f <int>              : first page to convert
  -l <int>              : last page to convert
  -r <fp>               : resolution, in DPI (default is 72)
  -x <int>              : x-coordinate of the crop area top left
corner
  -y <int>              : y-coordinate of the crop area top left
corner
  -W <int>              : width of crop area in pixels (default
is 0)
  -H <int>              : height of crop area in pixels (default
is 0)
  -layout               : maintain original physical layout
  -fixed <fp>           : assume fixed-pitch (or tabular) text
  -raw                  : keep strings in content stream order
  -htmlmeta             : generate a simple HTML file, including
the meta information
  -enc <string>         : output text encoding name
  -listenc              : list available encodings
  -eol <string>         : output end-of-line convention (unix,
dos, or mac)
  -nopgbrk              : don't insert page breaks between pages
  -bbox                 : output bounding box for each word and
page size to html.  Sets -htmlmeta
  -bbox-layout          : like -bbox but with extra layout
bounding box data.  Sets -htmlmeta
  -opw <string>         : owner password (for encrypted files)
  -upw <string>         : user password (for encrypted files)
  -q                    : don't print any messages or errors
  -v                    : print copyright and version info
  -h                    : print usage information
  -help                 : print usage information
  --help                : print usage information
  -?                    : print usage information
```

Create CWL tool for pdftotext

1. open template.tool.cwl and save as pdftotext.cwl

```
#!/usr/bin/env cwl-runner
cwlVersion: v1.0

# Type of definition
#   CommandLineTool , Workflow , ExpressionTool

class:  CommandLineTool

# optional label
# label: <LABEL FOR CWL TOOL>
# optional description/documentation
# doc: <DETAILED DESCRIPTION>

# optional hints for CWL execution
# hints:
# set execution environment for baseCommand
#  - class: DockerRequirement
#    dockerPull: <DOCKER IMAGE NAME>


# required, name of command line tool
baseCommand: <COMMAND>

# optional
arguments: <LIST OF CONSTANT OR DERIVED COMMAND LINE OPTIONS>

# required, input mapping
inputs: <LIST OF INPUT OPTIONS AND MAPPING TO COMMAND LINE>

# output mapping
outputs: <LIST OF NAMED OUTPUTS AND MAPPING TO COMMAND LINE
TOOL OUTPUT>
```

2. Set label to PDF-to-Text:

```
label: PDF-to-Text
```

3. Set name of executable:

```
baseCommand: pdftotext
```

4. We have created a Docker image for the command line tool:

```
hints:
# set execution environment for baseCommand
- class: DockerRequirement
# dockerPull: <NAME>:<TAG>
  dockerPull: demo:2
```

5. The script has to main positional arguments, PDF input file and text output file:

```
inputs:
  pdf:
    type: File
    doc: PDF input file to extract text from
    inputBinding:
      position: 1
  text:
    type: string
    doc: Name for text output file
    inputBinding:
      position: 2
```

6. Collect the output:

```
# output mapping
outputs:
  extractedText:
    type: File
    outputBinding:
      glob: $(inputs.text)
```

7. Test CWL tool, if you have cwl-runner in your path:

```
cwl-runner pdftotext.cwl --help
```

otherwise:

```
docker run -v `pwd`/pdftotext.cwl:/pdftotext.cwl
mgrast/awe-submitter:develop cwl-runner /pdftotext.cwl --help
```

```
/usr/bin/cwl-runner 1.0.20180116213856
Resolved '/pdftotext.cwl' to 'file:///pdftotext.cwl'
usage: /pdftotext.cwl [-h] --pdf PDF --text TEXT [job_order]

positional arguments:
  job_order    Job input json file

optional arguments:
  -h, --help   show this help message and exit
  --pdf PDF    PDF input file to extract text from
  --text TEXT  Name for text output file
```

8. Create a submission directory and and add a pdf file into it, e.g.:

```
mkdir -p submission
cp <my-document>.pdf submission/demo.pdf
```

9. Create CWL job file for pdftotext tool and save it as job.yaml in the parent directory of the submission directory/ The tool has two input parameter:
    a. `pdf` -> points to a file
    b. `text` -> name for output file

```
pdf:
  class: File
  path: submission/demo.pdf
text: demo.txt
```

10. Submit workflow and job:

```
docker run -ti --network skyport2_default --rm \
```

```
-v `pwd`/pdftotext.cwl:/pdftotext.cwl \
-v `pwd`/demo2.job.yaml:/job.yaml \
-v `pwd`/submission:/submission \
mgrast/awe-submitter:develop \
/go/bin/awe-submitter   \
--pack   \
--shockurl=${SHOCK_SERVER}  \
 --serverurl=${AWE_SERVER}  \
/pdftotext.cwl \
/job.yaml
```


```
{
    "extractedText": {
        "class": "File",
        "location":
"http://shock:7445/node/905863b6-bdde-40de-bd98-e7ff44903d51?download"
,
        "path":
"/Users/Andi/Development/MG-RAST-Repo/Skyport2/live-data/awe-worker/wo
rk/e7/8d/63/e78d63df-768c-4f3e-8f76-0c87221d00fe__entrypoint_wrapper_s
tep_0/tmp/nWI0JO/demo.txt",
        "basename": "demo.txt",
        "dirname": "",
        "nameroot": "",
        "nameext": "",
        "checksum": "sha1$b6272ceb6da71f9a1ed61069dfde46856851fb70",
        "size": 40649,
        "secondaryFiles": null,
        "format": "",
        "contents": ""
    }
}
```


11. To retrieve the workflow output open the Shock Browser and download the files.

    ~~To retrieve the output file copy the url from “location” and download the file with curl:~~

    ```
    curl \
    "http://localhost:8001/shock/api/node/905863b6-bdde-40de-bd98-e7ff44903d
    51?download" \
    -o extracted_text.txt
    ```

12.

13.

WordCloud

Extend existing Dockerfile for wordcloud

1. open previous Dockerfile `<MY-NAME>.dockerfile`

```
# Build from base image
FROM ubuntu:latest

# Update base images with latest patches
RUN apt-get update && apt-get upgrade -y

# Install dependencies
RUN apt-get install -y poppler-utils
```

2. Add dependencies, to install `wordcloud` you need `pip`. Add the ubuntu package `python-pip`.

```
# Build from base image
FROM ubuntu:latest

# Update base images with latest patches
RUN apt-get update && apt-get upgrade -y

# Install dependencies
RUN apt-get install -y \
  poppler-utils \
  python-pip
```

3. Install `wordcloud`

```
RUN pip install wordcloud
```

4. Save file and rebuild container

```
docker build -t <name>:<tag> -f <MY-NAME>.dockerfile .
```

Create CWL wordcloud tool description

2. Create CWL tool description
    a. changed into CWL/Tool directory
    b. copy `template.tool.cwl` to **<MY_TOOL>**`.tool.cwl`
    c. open **<MY_TOOL>**`.tool.cwl` in text editor
3. Add tool to workflow

docker build -t mgrast/wordcloud:demo -f Docker/Dockerfiles/wordCloud.dockerfile .

```
# Build from base image
# FROM ubuntu:late
```