LOFAR busy week

Matthijs van der Wild

A crash course on the Common Workflow Language

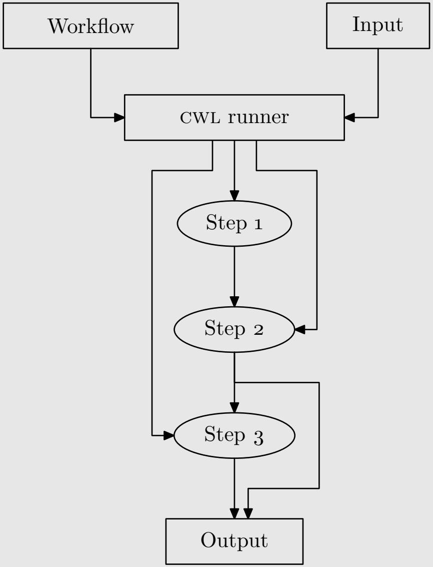
What is cwl?

- A specification for workflows

  CWL is a tool to connect command line tools through YAML files, and can itself be run entirely on the command line.

- Concatenate cli tools

  CWL handles intermediate I/O and temporary directories automatically.



What does a cwl file look like?

At its most basic: like a YAML or JSON file.

Consider the following file `input.yaml`:

```
old_directory:
  class: Directory
  path: "$HOME/old"
new_dir:
  class: string
  path: "new"
```

These values can be used by the file `copy_dir.cwl` on the right:

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: [cp, -r]
inputs:
  old_directory:
    type: Directory
    inputBinding:
      position: 1
  new_directory_name:
    type: string
    default: "new_dir"
    inputBinding:
      position: 2
outputs:
  new_directory:
    type: Directory
    outputBinding:
      glob: $(inputs.new_dir)
```

How to run cwl?

CWL is a specification. There are various implementations:

- `cwltool`
- `toil`
- ...

These should be available on COSMA and the Herts cluster.

For local installations:

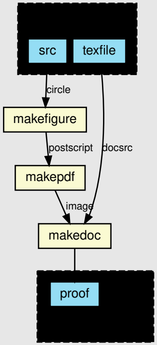
```
$ python -m venv venv
$ source venv/bin/activate
$ (venv) pip install -U pip setuptools wheel
$ (venv) pip install cwltool
```

Example use:

```
cwltool pipeline.cwl inputs.yaml
```

From steps to workflows

Commandline tools don't make pipelines: Workflows make pipelines!



```
cwlVersion: v1.2
class: Workflow

inputs:
  src:
    type: File
  texfile:
    type: File

outputs:
  proof:
    type: File
    outputSource: makedoc/document
```

```
steps:
  makefigure:
    run: draw.cwl
    in:
      circle: src
    out: [drawing]
  makepdf:
    run: convert.cwl
    in:
      postscript: makefigure/drawing
    out: [newpdf]
  makedoc:
    run: typeset.cwl
    in:
      docsrc: texfile
      image: makepdf/newpdf
    out: [document]
```

https://github.com/lonbar/workflow

Containers!

- Have CWL pull in a container.

- Run CWL with an external container.

Option 1:

```
hints:
  DockerRequirement:
    dockerPull: astronrd:linc
```

```
cwltool --singularity \
  workflow.cwl inputs.yaml
```

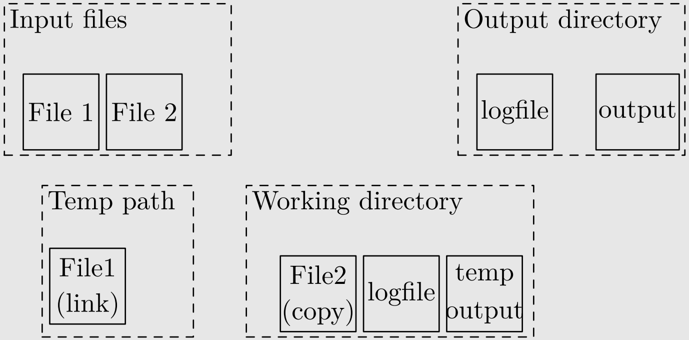
Option 2:

```
singularity exec --bind $IMAGEDIR,$WORKFLOWDIR,$OUTPUTDIR \
  cwltool --no-container workflow.cwl inputs.yaml
```

`cwltool` has lots of options:
relevant ones are listed at **https://linc.readthedocs.io/en/latest/**

Staging input files

```
requirements:
  InitialWorkDirRequirement:
    listing:
      - $(inputs.src)
```



Other relevant requirements

- `ResourceRequirement`

  Allocates node resources at runtime.

- `SubworkflowFeatureRequirement`

  Allows for workflows to be used as workflow steps.

- `InlineJavascriptRequirement`

  Allows use of simple `javascript` expressions in CWL files.

- `ScatterFeatureRequirement`

  Allows for iteration over multiple input files

These and more are documented at the CWL standard:

https://www.commonwl.org/v1.2/

Logging

CWL captures `stdout` and `stderr`.

```
stdout: output.log
stderr: output_error.log

outputs:
  logfiles:
    type: Directory[]
    outputBinding:
      glob: output*.log
```

Detailed input

Multiple or optional imputs:

```
inputs:
 msin:
  type: Directory[]
  inputBinding:
    position: 1
    prefix: msin=
 msout:
  type: string?
  default: $(inputs.msin.nameroot)
  inputBinding:
    position: 2
    prefix: msout=
```

The corresponding input file could look like:

```
msin:
  - class: Directory
    path: "/Data/Observation/MS1.ms"
  - class: Directory
    path: "/Data/observation/MS2.ms"
```

The name and location of input can be accessed by variables such as

`basename`, `nameroot`, `nameext`, `location`.

Further resources

- The CWL standard: **https://www.commonwl.org/v1.2/**
- The CWL user guide: **https://www.commonwl.org/user_guide/**
- The LINC documentation: **https://linc.readthedocs.io/en/latest/**
- These slides and example scripts: **https://github.com/lonbar/busyweek**
- Alexander Drabent and me :)