Metagenomic data analysis workflows in CWL from scratch to multi-environment production

Martin Beracochea (mbc@ebi.ac.uk)
Microbiome Informatics - EMBL-EBI
19th of January - 2021
Talk overview

- What is metagenomics and M Gnify?
- M Gnify evolution
- Common Workflow Language
- Deployment in the Cloud
Overview of approaches to microbiome research

- mRNA extraction
- DNA extraction
- PCR
- Metatranscriptome
- Metagenome
- Amplicon
- Sequencing
- Assembly, analysis and archiving of microbiome data
  - Metabarcoding/Amplicon
  - Metagenome
  - Metatranscriptome
  - Assembly

- High-performance computing and standardized pipelines
- Allows comparison of data
- Request analysis of public data
- Submit private (pre-publication) data for analysis
MGNify evolution

Pipeline version 4.1, released January 2018
MGnify evolution

Version 5.0, released December 2019

MGNify pipeline version 5

- Complex workflow with over 20 interconnected applications.
- Diverse runtimes and dependencies.
MGNify workflow evolution

- Version 4.1
  - Python codebase difficult to maintain (~9500 loc)
  - Tightly coupled to the EBI cluster

- Version 5.0
  - Significant increase in complexity
  - We wanted to improve
    - provenance
    - Interoperability
    - reproducibility of the pipeline
Common Workflow Language

• Common declarative format for tool and workflow execution.
• Standard developed by the community.
• Defined with a schema, specification and test suite.
• Supports the use of containers (e.g. Docker) and shared research computing clusters with locally installed software
Command Line Tool

Filetype and metadata

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

Executable

```bash
baseCommand: ["filter_contigs_len.py"]
```

Inputs

```cwl
inputs:
  fasta_file:
    type: File
    inputBinding:
      separate: true
      prefix: "-f"
  length:
    type: float
    inputBinding:
      prefix: "-l"
  outdir:
    type: Directory?
    inputBinding:
      separate: true
      prefix: "-o"
```

Outputs

```cwl
outputs:
  filtered_contigs_fasta:
    type: File
    outputBinding:
      glob: '*_filt*.fasta'
```
Workflow

cwlVersion: v1.0
class: Workflow

inputs:

input_fasta_file: # input assembly
  type: File

virsorger_virome:
  type: boolean
  default: false
  doc: Set this parameter if the input fasta is mostly viral. See: https://github.com/simroux/VirSorter/issues/50

outputs:

filtered_contigs:
  outputSource: length_filter/filtered_contigs_fasta
  type: File

virfinder_output:
  outputSource: virfinder/virfinder_output
  type: File

steps:

fasta_rename:
  label: Filter contigs
  run: ./Tools/FastaRename/fasta_rename.cwl
  in:
    input: input_fasta_file
  out:
    - renamed fasta
      name: renamed_fasta
    - home_map

length_filter:
  label: Filter contigs
  run: ./Tools/LengthFiltering/length_filtering.cwl
  in:
    fasta_file: fasta_rename/renamed_fasta
  out:
    - filtered_contigs_fasta
      name: filtered_contigs_fasta
      length:
        default: 1.0
Reutilization

Re-use of sub-workflows. We were able to run sub-workflows from MG-RAST
# CWL Implementations

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Platform support</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwltool</td>
<td>Reference implementation of CWL</td>
<td>Linux, OS X, Windows, local execution only</td>
</tr>
<tr>
<td>Arvados</td>
<td>Distributed computing platform for data analysis on massive data sets. Using CWL on Arvados</td>
<td>AWS, GCP, Azure, Slurm</td>
</tr>
<tr>
<td>Toil</td>
<td>Toil is a workflow engine entirely written in Python.</td>
<td>AWS, Azure, GCP, Grid Engine, HTCondor, LSF, Mesos, OpenStack, Slurm, PBS/Torque</td>
</tr>
<tr>
<td>CWL-Airflow</td>
<td>Package to run CWL workflows in Apache-Airflow (supported by BioWardrobe Team, CCHMC)</td>
<td>Linux, OS X</td>
</tr>
<tr>
<td>REANA</td>
<td>RE usable ANALyses</td>
<td>Kubernetes, CERN OpenStack (OpenStack Magnum)</td>
</tr>
<tr>
<td>CWLEXEC</td>
<td>Apache 2.0 licensed CWL executor for IBM Spectrum LSF, supported by IBM for customers with valid contracts.</td>
<td>IBM Spectrum LSF 10.1.0.3+</td>
</tr>
</tbody>
</table>

Source: [https://www.commonwl.org/#Implementations](https://www.commonwl.org/#Implementations)
# CWL Implementations

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
<th>Platform support</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwltool</td>
<td>Reference implementation of CWL.</td>
<td>Linux, OS X, Windows, local execution only</td>
</tr>
<tr>
<td>Arvados</td>
<td>Distributed computing platform for data analysis on massive data sets. Using CWL on Arvados</td>
<td>AWS, GCP, Azure, Slurm</td>
</tr>
<tr>
<td>Toil</td>
<td>Toil is a workflow engine entirely written in Python.</td>
<td>AWS, Azure, GCP, Grid Engine, HTCondor, LSF, Mesos, OpenStack, Slurm, PBS/Torque</td>
</tr>
<tr>
<td>CWL-Airflow</td>
<td>Package to run CWL workflows in Apache-Airflow (supported by BioWardrobe Team, CCHMC)</td>
<td>Linux, OS X</td>
</tr>
<tr>
<td>REANA</td>
<td>RE usable ANALyses</td>
<td>Kubernetes, CERN OpenStack (OpenStack Magnum)</td>
</tr>
<tr>
<td>CWLEXEC</td>
<td>Apache 2.0 licensed CWL executor for IBM Spectrum LSF, supported by IBM for customers with valid contracts.</td>
<td>IBM Spectrum LSF 10.1.0.3+</td>
</tr>
</tbody>
</table>

Source: [https://www.commonwl.org/#Implementations](https://www.commonwl.org/#Implementations)
CWL Implementations

Toil

A scalable, efficient, cross-platform pipeline management system written entirely in Python and designed around the principles of functional programming.

http://toil.ucsc-cgl.org/
CWL + Toil our journey

We have faced numerous issues with Toil and our CWL implementation:

- Slow development process
- Toil logs are extremely verbose and that complicates debugging
- The intricacies of communication between Toil and LSF
- Overall complexity of our pipeline in terms of numbers of tools and size of metagenomics datasets
- CWL features not available in Toil, mostly solved now
Tool performance tuning

Scaling up some tools, for example HMMER or antiSMASH
Cloud infrastructure / Workflow deployment
We carried out 2 projects with the OCI team.

First project:
- Running memory intensive step of our pipeline in OCI infrastructure.
- Bumped into an obscure kernel issue in Oracle Linux.

Second project:
- Run the whole pipeline using bare-metal instances in OCI.
- Increased throughput compared EBI cluster.
Summary

- CWL-based metagenomics pipeline
- Cloud infrastructure using Terraform
- Successfully deployed on OCI and Embassy (GCP recently)

Future Work

- Orchestration between environments
- Dataflow
- Optimize job-packing
Acknowledgements

EMBL  Rob Finn

**MGNify Service Team**  **Finn Research Team**

Germana Baldi  Alex Almeida
Juan Caballero  Ales Escobar
Tanya Gurbich  Danilo Horta
Varsha Kale    Sara Kashaf
Satwant Kaur   Felix Langer
Lorna Richardson  Paul Saary
Kate Sakharova
Nicolo Lazzaro

**Cloud Consultants**  **CWL**

Tony Wildish  Michael Crusoe

**Demonstrators**

Alexander Sczyrba