https://tinyurl.com/2021-elixir-cwl

Introduction to the Common Workflow Language

Michael R. Crusoe @biocrusoe
CWL Project Leader #CommonWL https://orcid.org/0000-0002-2961-9670

https://tinyurl.com/2021-elixir-cwl

Background

- Computational workflows are routinely used for large scale analyses in many fields
- Replication, validation, and extension of scientific results are crucial for scientific progress
- Many workflows systems exist but few of the systems have
  - adoption, active user community, and sustained development support
  - the ability to painlessly port or extend their workflows to another system or platform
- Needed a **multi-lingual** workflow description **standard** between systems and for cross-vendor portability

https://tinyurl.com/2021-elixir-cwl

The CWL Project is a boutique SDO, part of Software Freedom Conservancy

The CWL project supports open consensus-based standards for command line data analysis workflows and tools. Specifically, the project supports the

- ***pre-standards process*** by providing a neutral place of convening to discuss, propose and test ideas about command-line tool based workflow standards and related topics.
- ***standardization process*** by stewarding the development and delivery of standards in accordance with the Open Stand principles.
- ***post-standards life cycle*** by (1) promoting the released standards, (2) developing and maintaining related training and tools, and by (3) tracking deficits and other post-standardization feedback.

https://tinyurl.com/2021-elixir-cwl

What is Common Workflow Language (CWL)?

- Open standard for describing analysis workflows and tools
  - Started as a grassroots effort by developers at BOSC codefest in 2014
  - Community based standards effort, not a specific software package
- Defined with a schema, specification and test suite
  - Reference implementation (cwltool) along with academic and commercial production implementations
- Portable and scalable across a variety of software and deployment environments
  - Supports the use of containers (e.g. Docker, Singularity)

- Designed to meet the needs of data-intensive science to improve the FAIRness of their workflows
  - CWL now used in Bioinformatics, Medical Imaging, Astronomy, High Energy Physics, Machine Learning, ... GeoSpatial?

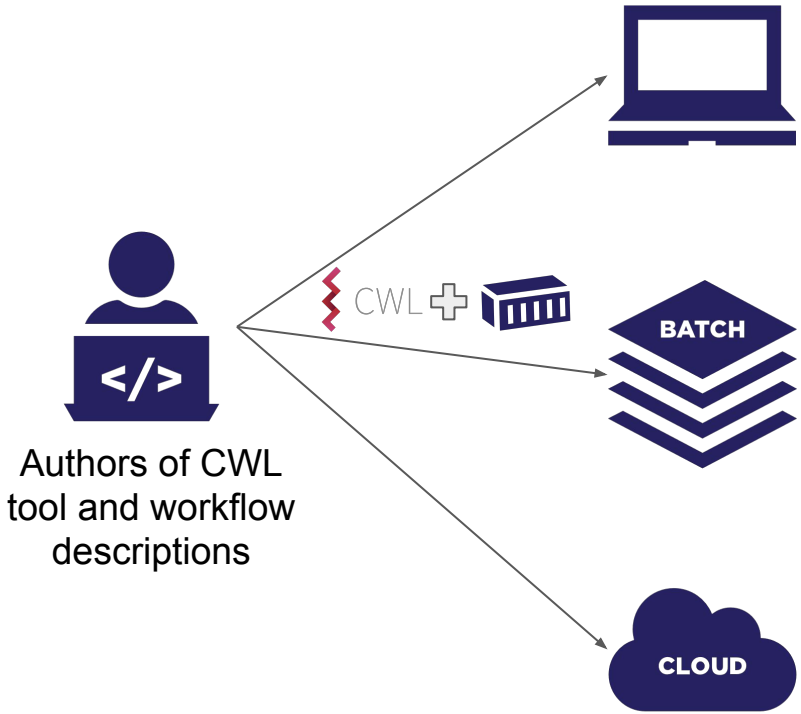
CWL: Two Standards in One

- CWL Command Line Tool Description standard: how to run a single tool; what inputs are required and allowed, what outputs are made and how to get them.
- CWL Workflow Description standard: connecting these CommandLineTools along with sub-Workflows into a workflow graph

Can use just the CommandLineTool CWL standard or the full combination of both.

https://tinyurl.com/2021-elixir-cwl

CWL Enables Execution Portability



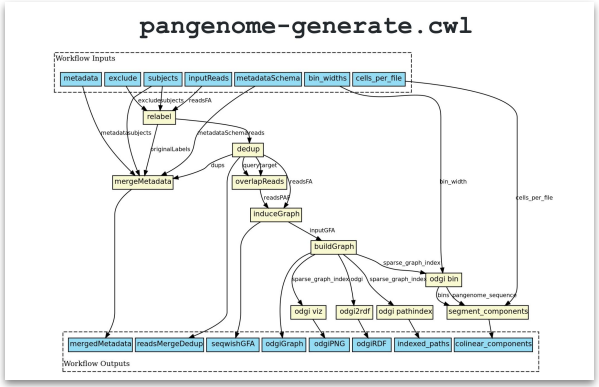
Authors of CWL tool and workflow descriptions

Local execution on Linux, macOS, and MS Windows via the CWL reference implementation (`cwltool`) and Docker/uDocker/Singularity/podman/...

Backends supported by various F/OSS CWL implementations

https://tinyurl.com/2021-elixir-cwl

CWL Technical Details



- CWL file contains a tool or workflow description
- Human readable
  - Written in YAML or JSON
  - Many optional fields to increase readability and reusability (i.e. “doc”, “label”, “SoftwarePackage”, “format”)
- Input/outputs are explicitly stated
- Designed to be modular and easy to reuse components
  - CWL Workflows are graphs made up of CWL tool descriptions
- Designed for high-throughput (grid and cloud) computing
  - Distribute steps over many compute nodes
  - Data movement handled by the CWL-aware workflow engine
- Encourages well-marked vendor/user extensions
  - Supporting progress without hurting portability

https://tinyurl.com/2021-elixir-cwl

CWL Encourages Progressive Enhancement

```yaml
cwlVersion: v1.0
class: CommandLineTool

inputs:
  readsFA: File

baseCommand: spoa

arguments: [ $(inputs.readsFA), -G, -g, '-6' ]

outputs:
  spoaGFA:
    type: stdout
```

```yaml
cwlVersion: v1.0
class: CommandLineTool

doc: |
  Spoa (SIMD POA) is a c++ implementation of the partial order alignment (POA) algorithm
  (as described in 10.1093/bioinformatics/18.3.452) which is used to generate consensus
  sequences (as described in 10.1093/bioinformatics/btg109). It supports three alignment
  modes: local (Smith-Waterman), global (Needleman-Wunsch) and semi-global alignment
  (overlap), and three gap modes: linear, affine and convex (piecewise affine). It also
  supports Intel SSE4.1+ and AVX2 vectorization (marginally faster due to high latency
  shifts), SIMDe and dispatching.

inputs:
  readsFA:
    format: edam:format_1929
    type: File
  doc: |
    Input FASTA file containing a set of sequences to be aligned in order to generate
    a genome graph.  For best results, the sequences should be sorted by length (longest
    to shortest) and quality (best to worst).

hints:
  DockerRequirement:
    dockerPull: "quay.io/biocontainers/spoa:3.4.0--hc9558a2_0"
  ResourceRequirement:
    ramMin: $(15 * 1024)
    outdirMin: $(Math.ceil(inputs.readsFA.size/(1024*1024*1024) + 20))

requirements:
  InlineJavascriptRequirement: {}

baseCommand: spoa

arguments: [ $(inputs.readsFA), -G, -g, '-6' ]

stdout: $(inputs.readsFA.nameroot).g6.gfa

outputs:
  spoaGFA:
    type: stdout
    format: edam:format_3976  # GFA
    doc: Output in Graphical Fragment Assembly (GFA) format.

$namespaces:
  edam: http://edamontology.org/
```

Community Maintained File Format Identifier

Dynamic Resource Requirements

Both describe the same tool.

The 2nd description is more helpful.

https://tinyurl.com/2021-elixir-cwl

CWL Data Model

The basic unit is a command line tool. 

CWL Types: strings, numbers, file/directories, or records that combine these; or arrays of any of these types. Union and optional types too.

Files can have a further specialization via the “format” field: a URI that identifies the file type

iana:application/geo+json
edam:format_3016

CWL does not dictate the source of these format identifiers, each community of users should define their own.

CWL Technical Details cont.

- Workflow graph can be exported as linked-data (RDF/JSON-LD)
- Supports provenance exporting using existing standards and ontologies: W3C Prov, IETF BagIt, wfdesc, wfprov
- CWL's object model enables a variety of infrastructure-specific optimizations
  - **Cost and/or data-location aware scheduling**
  - (User overridable) caching of results
  - Streaming in-/out- of object stores; or between steps
- Hundreds of conformance tests are used to ensure portability independent of vendor
- Workflow validation catches many sneaky syntax errors before runtime

Data locality with CWL

Input and output files are modeled in CWL as rich object with identifier (URI/IRI) and other metadata.

Platforms that understand CWL can use these identifiers to **send compute to near the location of data**.

In combination with the resource matchmaking this can conversely result in data being sent to specialized compute resources as configured by the operator (or machine learning)

Proposed enhancement to CWL data model

input value restrictions / validations · Issue #764

Refinements to the existing CWL types have been proposed, but need implementation before they can be voted on.

```
string: Regular expressions, string sets
int/long: Integer intervals, integer sequences, integer sets
float/double: (Real) intervals, integer intervals, real sets
```

Goal is to catch validation errors sooner, produce more helpful (G)UIs, and prevent execution of workflows/tools that doomed to fail

How to extend CWL for your own needs?

CWL 💓 community/vendor extensions!

1. Do let the CWL community know how your needs aren't being met.
2. Experiment with alternative syntax via additional Requirements. Fork the CWL reference runner (cwltool) or another CWL implementation to implement your ideas.
3. Make sure that your extensions are namespaced, so that other systems can still read your CWL documents.
4. Let the CWL community know about your progress as you go.
5. If it makes sense, make a formal proposal for possible inclusion in a future version of the CWL standards!

CWL v1.2 released 2020-08-10!

3 new features: **workflow level conditionals**, abstract operations, absolute paths for container inputs

20 cleanups and clarifications of corner cases in the specifications

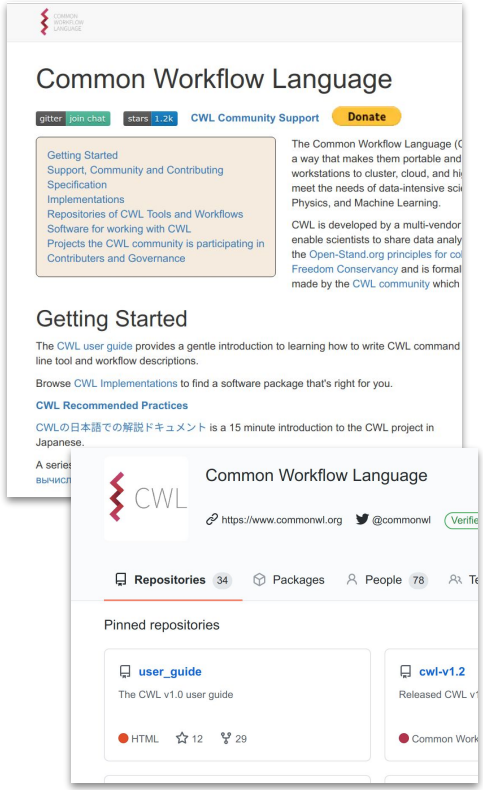
Forward compatibility via the `cwl-upgrader` script or the reference CWL runner

Available today in the CWL reference runner (cwltool), Arvados, and toil-cwl-runner. Support in additional commercial providers is forthcoming.

https://tinyurl.com/2021-elixir-cwl

Participating in the CWL Community

- https://www.commonwl.org/
- Getting Started
  - User guide: https://www.commonwl.org/user_guide/
- Support, Community and Contributing
  - Forum: https://cwl.discourse.group/
  - Chat: https://gitter.im/common-workflow-language/home
  - GitHub: https://github.com/common-workflow-language/
  - Social Media: @commonwl & #CommonWL
- Weekly video chat



https://tinyurl.com/2021-elixir-cwl

Common Workflow Language

Is a vendor neutral open standard

Is a community-first project

Designed with an open and transparent governance

Improves interoperability and portability

Increases reusability and reproducibility

Enables parallelization and scale

Is supported by an ecosystem of tools, libraries, and editor plugins

https://tinyurl.com/2021-elixir-cwl

Thank you!

Questions?

https://www.commonwl.org

https://tinyurl.com/2021-elixir-cwl

Backup slides..

Linked Data & CWL

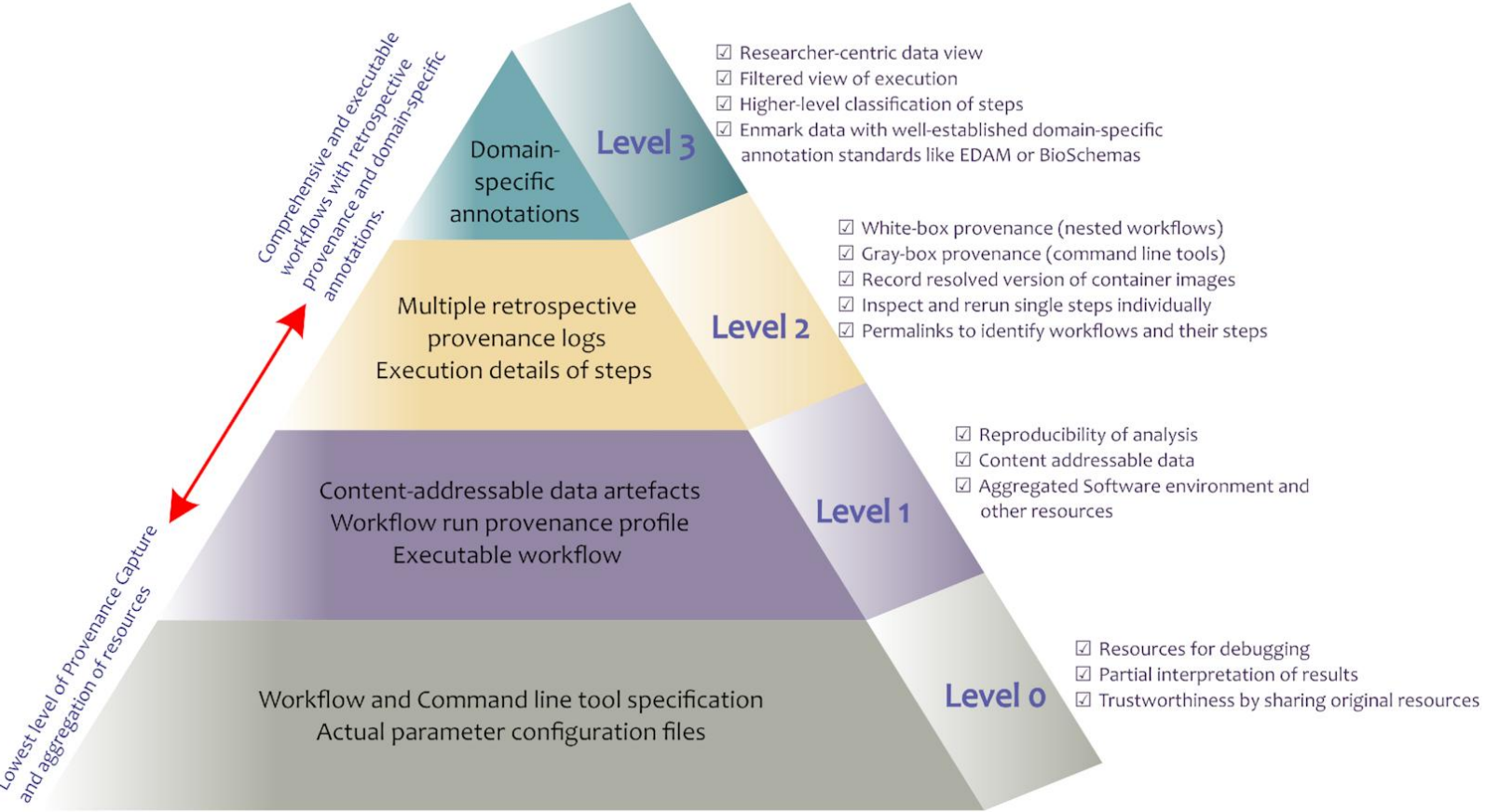
- Hyperlinks are common currency
- Bring your own RDF ontologies for metadata
- Supports SPARQL to query

Example: can use the EDAM ontology to specify file formats and reason about them:

"FASTQ Sanger" encoding is a type of FASTQ file

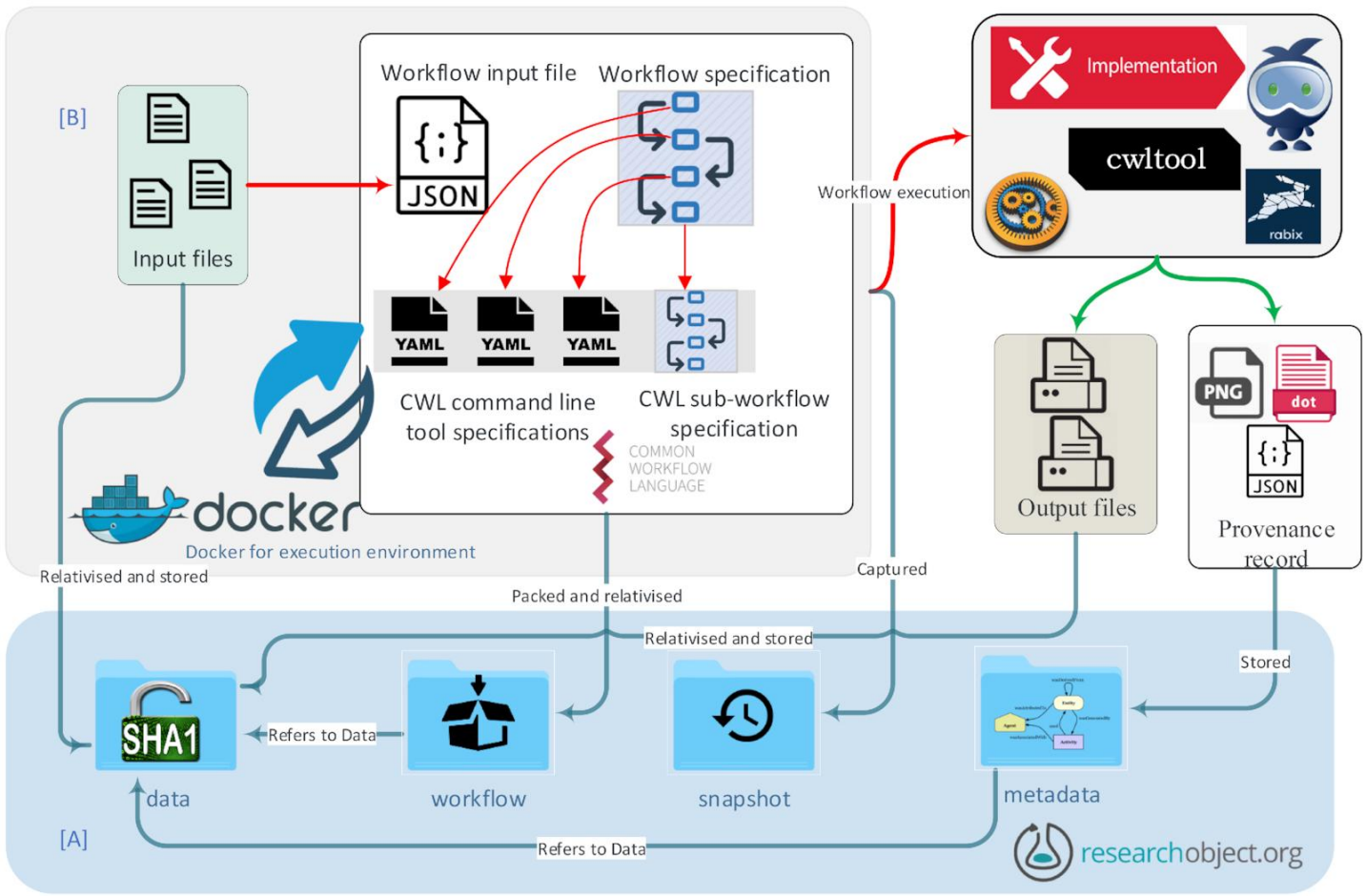
https://tinyurl.com/2021-elixir-cwl

Figure 2: Levels of provenance and resource sharing and their applications.



Gigascience, Volume 8, Issue 11, November 2019, giz095, https://doi.org/10.1093/gigascience/giz095

s://tinyurl.com/2021-elixir-cwl



Gigascience, Volume 8, Issue 11, November 2019, giz095, https://doi.org/10.1093/gigascience/giz095

Timeline

2014 Bioinformatics Open Source Conference CodeFest:
4 software engineers & a whiteboard

2015: CWL “draft-2” version, commercial vendor (SBG) releases product in December.

2016: CWL v1.0 released

2017: CWL v1.0.1 and v1.0.2 released.
Now 4 public implementations

2018: **IBM** released their CWL implementation for LSF.

2019: CWL v1.1 released

2020: CWL v1.2 released with workflow conditionals, work on CWL v1.2.1 and beyond commences