



# COMMON WORKFLOW LANGUAGE

Michael R. Crusoe  [@biocrusoe@octodon.social](https://twitter.com/biocrusoe)  
CWL Project Leader

 <https://orcid.org/0000-0002-2961-9670>

All slide text is  
licensed [CC-BY-4.0](https://creativecommons.org/licenses/by/4.0/)



# Michael R. Crusoe

CWL co-founder, CWL Project Leader; Director of APIs and Standards, [Workflows Community Initiative \(WCI\)](#); ELIXIR Compute Platform member

Both the academic and industry sides of [my career](#) are very non-traditional!

Experience as a computer systems administrator, software engineer. B.S. in Microbiology from Arizona State University.

Finishing a part-time PhD program at VU Amsterdam (CompSci/Bioinformatics) on the theme of workflow systems.

Based in Berlin since 2019 and planning to stay here a long time!

# Why use a “workflow” approach?

Workflows are a type of **structured computing**.

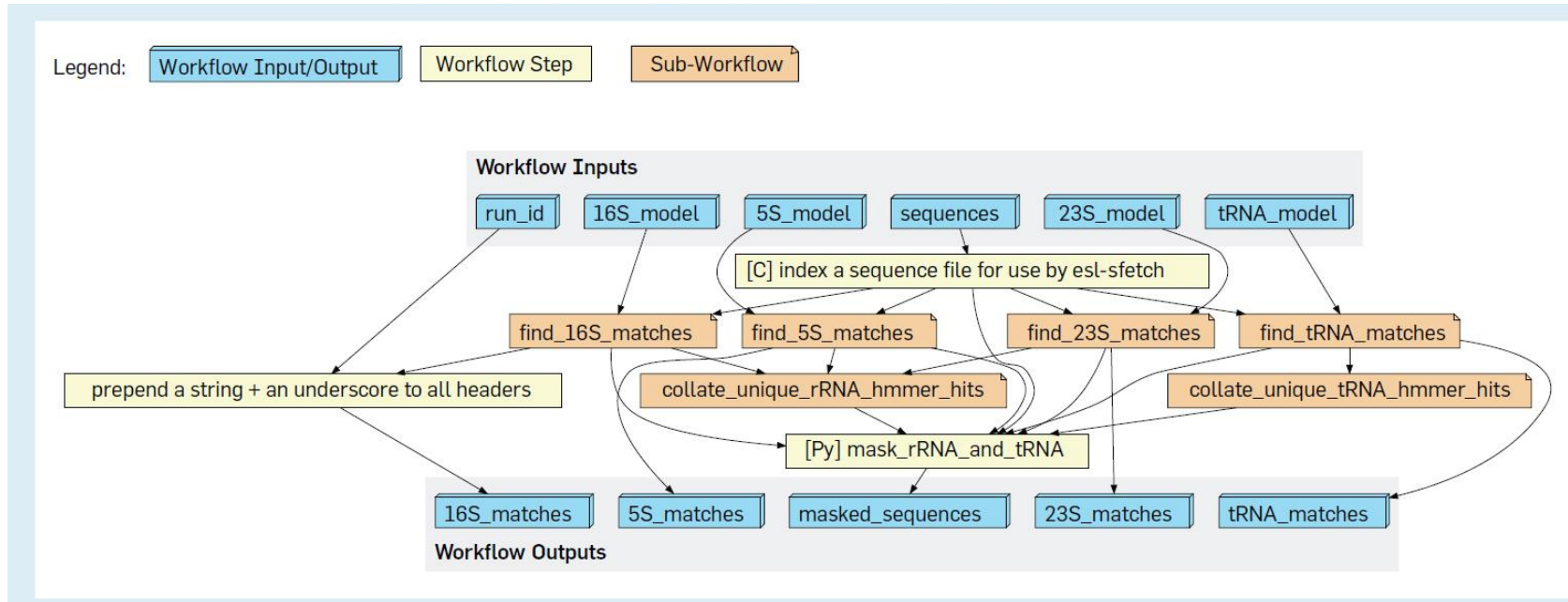
A good workflow language/system provides powerful abstractions that enable

- **Scaling** (analysis can run faster, or even with a huge amount of data)
- **Automation** (more breaks for the researcher == better mental health!)
- **Provenance** tracking (manage complexity: where did this file come from???)

Workflows complement **un-structured** approaches; they don't replace them.

# What type of workflows are we talking about?

Many tools, written in different programming languages



**Figure 1.** Excerpt from a large microbiome bioinformatics CWL workflow

# 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

# 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.

# CWL overview paper published in June 2022 CACM

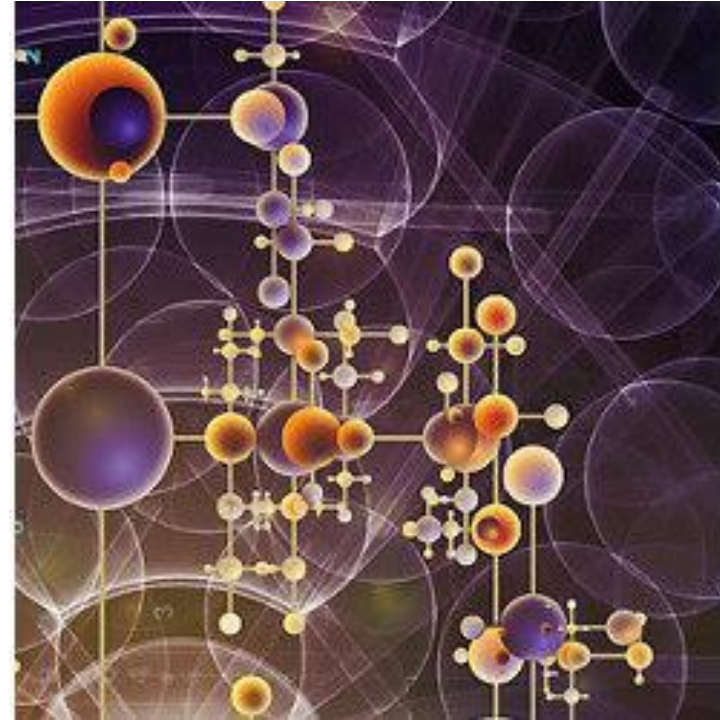
Michael R. Crusoe, Sanne Abeln, Alexandru Iosup, Peter Amstutz, John Chilton, Nebojša Tijanić, Hervé Ménager, Stian Soiland-Reyes, Bogdan Gavrilović, Carole Goble, and The CWL Community. 2022.

[Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language.](#)

Commun. ACM 65, 6 (June 2022), 54–63.

<https://doi.org/10.1145/3486897>

Many of the figures that follow are from this paper, which is licensed under [CC BY 4.0](#)





# 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.



# 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. (Last visit to CERN)

2019: CWL v1.1 released

2020: CWL v1.2 released with workflow conditionals

CWL included in the IEEE Std 2791™--2020 BioCompute Object standard

[Pegasus releases initial support for CWL](#)

2021: Support for CWL in Illumina Connected Analytics

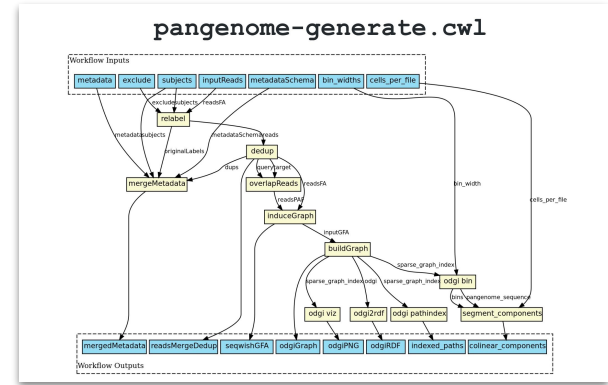
2022: CWL Article in CACM

2023: Amazon Omics adds CWL support

2024: CWL 10th anniversary!

# 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



# CWL Encourages Progressive Enhancement

```

cwlVersion: v1.0
class: CommandLineTool

inputs:
  readsFA: File

baseCommand: spoa

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

outputs:
  spoaGFA:
    type: stdout

```

Both describe the same tool.

The 2nd description is more helpful.

```

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

# 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](#)

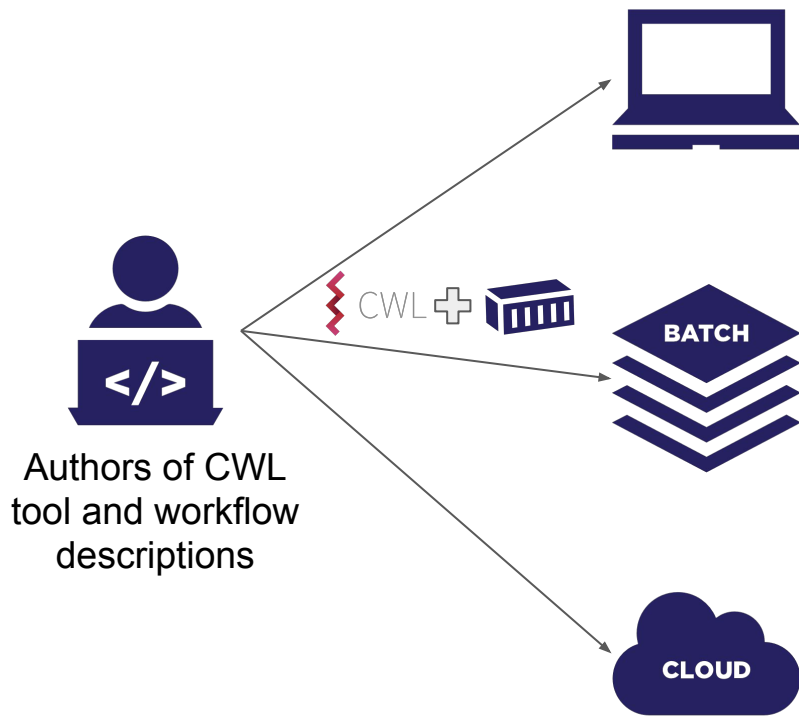
edam:[format\\_3016](#)

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

# CWL Enables Execution Portability



Local execution on Linux, macOS, and MS Windows via the CWL reference implementation (`cwltool`) and Docker/[uDocker](#)/Singularity/Podman/Apptainer...



Backends supported by various F/OSS CWL implementations

# CWL Enables Execution Portability

Containers are not required.

Some CWL runners map software package names to local  
conda/spack/environment module

See

<https://github.com/common-workflow-language/cwltool#leveraging-software-requirements-beta> (also supported by toil-cwl-runner)



# 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

# 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)

# What's new in the CWL standards since 2017?

Major features: [workflow level conditionals](#), abstract [operations](#), absolute paths for container inputs

[ToolTimeLimit](#) hint to assist “wall time” based schedulers.

[WorkReuse](#) hint that certain steps should not be cached

[InplaceUpdateRequirement](#) hint that destructive data edits are allowed (requested by radio astronomers)

181 new conformance tests!

Forward compatibility guaranteed by the via the [cwl-upgrader](#) script or the reference CWL runner

# Michael's questions for workflow platforms/users

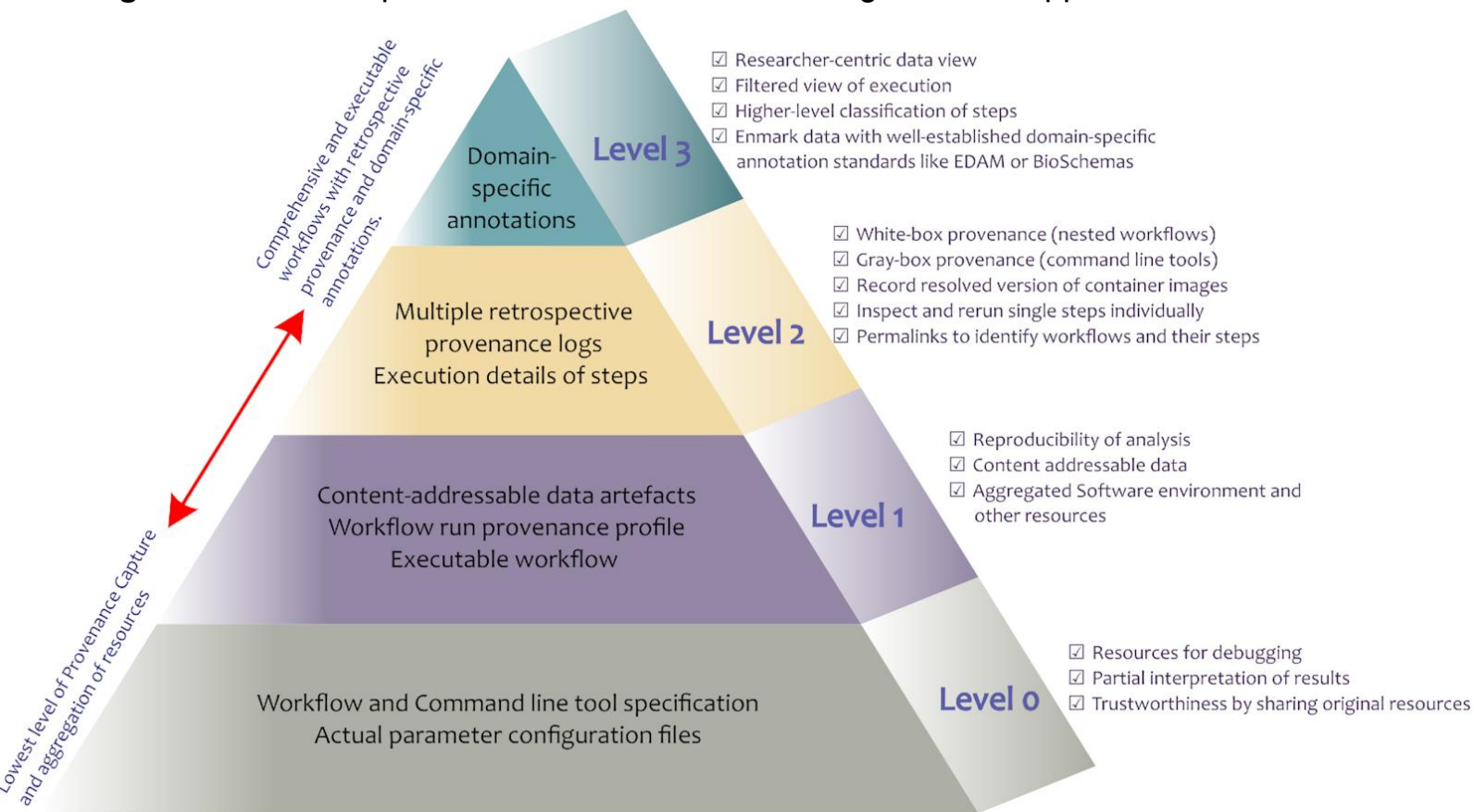
Where did your data come from?

How do you track that? (identifiers, metadata, etc..)

Researchers move between systems while doing their analysis, this is normal.

How do you connect the stories of what happened in these systems?  
(provenance)

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

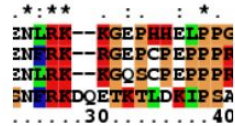


# RO-Crate

RO-Crates are a great way to empower researchers to not lose details as they move between different scientific computing environments.

For workflow execution, there is the [WorkflowRun RO-Crate profile](#); this will replace CWLProv soon.

# CWL Example Uses



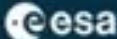
See more at <https://www.commonwl.org/gallery/>







# Interesting recent adopters of CWL

- Open Geospatial Consortium is soon to formalize adoption of their “OGC API - Processes for processing geospatial information on the web” standard that include CWL as their workflow language of choice.
- The EBRAINS (European distributed digital infrastructure for brain and brain-inspired research) [have adopted CWL](#)



# European Space Agency's Datalabs Pipeline Service

THE EUROPEAN SPACE AGENCY 

ESA Datalabs (0.0.0/BETA)      

Workspace Catalogue **JWST.pipeline.cwl**

Graph Code Test Details Push Pipeline - 0.0.1 [latest] (changed)

Workspace

My Pipelines in Workspace

System Pipelines in Workspace Pull

▼ jwst

▶ dependencies

**JWST.pipeline.cwl**

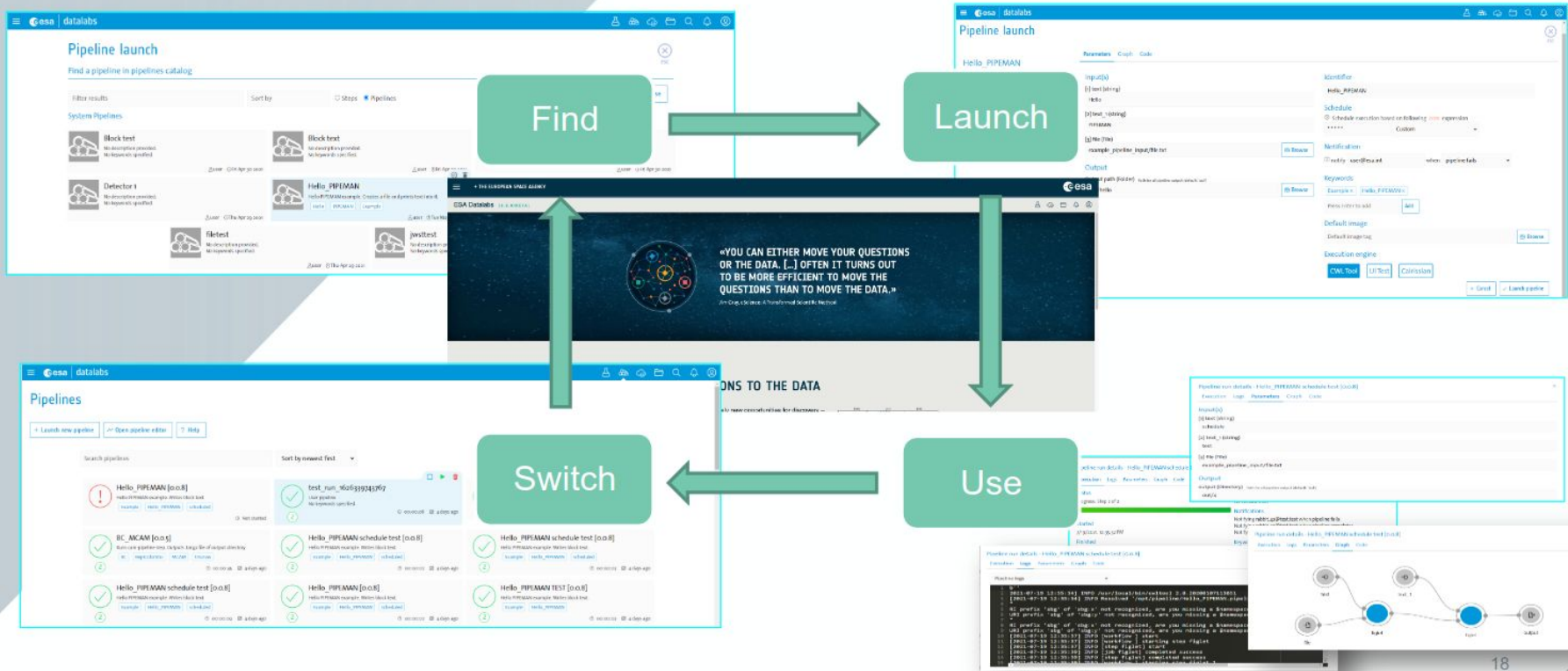
manifest.json

```
graph LR; A[Pipeline input FITS file] --> D1((Detector1)); B[Keyword in input file] --> D1; C[Detector1 CRDS override list] --> D1; D[CRDS cache overrides] --> D1; D1 --> E[Detector1 intermediate output]; E --> I2((Image2)); F[CRDS cache] --> I2; G[CRDS Context] --> I2; H[Image 2 CRDS cache override list] --> I2; I[Keyword in output file] --> I2; I2 --> J[FITS file output]
```

Validation Execution logs

[https://doi.org/10.1007/978-981-97-0041-7\\_1](https://doi.org/10.1007/978-981-97-0041-7_1)

# Pipelines Catalogue & Utilisation Lifecycle



## European Space Agency's Datalabs Pipeline Service

Validated for the James Webb Space Telescope data reduction pipeline, for example:

[https://s2e2.cosmos.esa.int/www/esadatalabs\\_iapplications/Validation\\_case\\_JWST.html](https://s2e2.cosmos.esa.int/www/esadatalabs_iapplications/Validation_case_JWST.html)

More information at [https://doi.org/10.1007/978-981-97-0041-7\\_1](https://doi.org/10.1007/978-981-97-0041-7_1)

# CWL 2024 Conference & 10th Anniversary Celebration

In-person talks and break-out sessions in Amsterdam, May 15 & 16th; hackday on May 17th

<https://www.commonwl.org/conferences/2024/>

Still room to squeeze in a couple more talks, please ask me!

# 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!

# Participating in the CWL Community

- <https://www.commonwl.org/>
- Getting Started
  - Novice tutorial (START HERE): <https://carpentries-incubator.github.io/cwl-novice-tutorial/>
  - User guide: [https://www.commonwl.org/user\\_guide/](https://www.commonwl.org/user_guide/)
- Support, Community and Contributing
  - Forum: <https://cwl.discourse.group/>
  - Chat: <https://matrix.to/#/#cwl:matrix.org>
  - GitHub: <https://github.com/common-workflow-language/>
- [Weekly video chat](#)

# 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

Thank you!

Questions?

<https://www.commonwl.org>



Backup slides..

# 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

# US NIH Funded Analysis platforms using CWL

- [NCI Cancer Genomics Cloud](#)
- [NHLBI BioData Catalyst - Seven Bridges](#)
- [NCI Genomic Data Commons](#)
- [Gabriella Miller Kids First Data Resource Center](#) (cancer and structural birth defects) ([link to their CWL workflows](#))
- [CAVATICA](#) (pediatric disease)

# MGnify: expert curated web service backed by CWL

Free web service, curated by the European Bioinformatics Institute

Analysis and archiving of raw microbiome data to help determine the taxonomic diversity and functional & metabolic potential of environmental samples.

<https://www.ebi.ac.uk/metagenomics> <https://doi.org/10.1093/nar/gkac1080>

Users need no workflow experience; the CWL source is available for study and re-use: <https://github.com/EBI-Metagenomics>

The logo for MGnify features the word "MGnify" in a bold, black, sans-serif font. The letter "G" is stylized with a black microscope icon integrated into its upper right curve. The "i" is lowercase and has a dot, while the "y" is lowercase and has a tail that extends downwards.

**MGnify**

# CWL Workflows published by the US NIH NCBI

## Prokaryotic Genome Annotation Pipeline (PGAP)

Previous versions (2001-2017) written using NCBI specific workflow systems  
In ~2018 PGAP was rewritten to use the CWL standards.

All (200 000) prokaryote genomes in [RefSeq](#) are annotated via this workflow; and the workflow is actively maintained and improved by NCBI staff.

<https://github.com/ncbi/pgap>

<https://doi.org/10.1093/nar/gkaa1105>



# CWL Workflows published by the US NIH NCBI

## Read assembly and Annotation Pipeline Tool (RAPT)

“a one-step application for the genome assembly and gene annotation of archaeal and bacterial isolates”

Written and maintained by the staff of the US National Institutes of Health, National Center for Biotechnology Information.

<https://github.com/ncbi/rapt>

