https://tinyurl.com/NASEM-Workflows-CWL

COMMON WORKFLOW LANGUAGE

The power of Free & Open Standards, a community driven success story

Michael R. Crusoe: CWL Project Lead @biocrusoe

NASEM; Opportunities for Accelerating Scientific Discovery: Realizing the Potential of Advanced and Automated Workflows

2020-03-17

#CommonWL

What does "workflow" mean in CWL-land?

"Workflow" can mean many things!

CWL is the open standard for describing:

> batch style automated data analysis workflows, made from POSIX (Linux) command line tools

CWL is not for "physical world" or business process workflows; many other standards for that :-)

https://tinyurl.com/NASEM-Workflows-CWL

Common Workflow Language v1.0 & v1.1

- Common **declarative** format for tool & workflow execution
- Community based standards effort, not a specific software package; **Very extensible**
- Defined with a schema, specification, & test suite
- Designed for shared-nothing clusters, HPC clusters, cloud environments, and local execution
- Supports the use of containers engines (e.g. Docker, Singularity) and shared research computing clusters with locally installed software

Why have a standard?

- Standards create a surface for collaboration that promote innovation
- Research frequently dip in and out of different systems but interoperability is not a basic feature.
- Funders, journals, and other sources of incentives prefer standards over proprietary or single-source approaches

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 Q2: CWL v1.2 with workflow conditionals

CWL Design principles

- Low barrier to entry for implementers
- Support tooling such as generators, GUIs, converters
- Allow extensions, but must be well marked
- Be part of linked data ecosystem
- Be pragmatic

Well described tools and workflows → Save time, money

CWL tool descriptions can self describe the “shape” of the computation

- # of cores
- memory needs
- temporary and output storage estimations

This uses fixed values, or can be computed prior to scheduling based upon the input data & its metadata

http://www.commonwl.org/v1.0/CommandLineTool.html#Runtime_environment

Community Based Standards development

Different model than traditional nation-based or regulatory approach

We adopted the Open-Stand.org Modern Paradigm for Standards: Cooperation, Adherence to Principles (Due process, Broad consensus, Transparency, Balance, Openness), Collective Empowerment, (Free) Availability, Voluntary Adoption

Open Source Implementations

Full list at https://www.commonwl.org/#Implementations

Arvados from Curoverse / Veritas Genetics

CWLEXEC from IBM LSF

CWL-Airflow from BioWardrobe Team, CCHMC

Toil from UCSC & community contributors

REANA from CERN

Open Source Implementations

Full list at https://www.commonwl.org/#Implementations

Some are full platforms, others are just workflow executors.

Execution environments include:

- Local (Linux, OS X, Windows)
- HPC: Slurm, GridEngine, PBS, LSF, HTCondor, Apache Airflow
- Cloud: Amazon AWS, Google GCP, Mesos, OpenStack, MS Azure, Kubernetes

Editors, viewers, utilities, etc.

Rabix CWL GUI (“Composer”) also integrated into the Arvados Platform

Text editor support for Atom, Vim, emacs, Visual Studio, IntelliJ, and gedit courtesy community contributors

https://www.commonwl.org/#Editors_and_viewers

https://www.commonwl.org/#Converters_and_code_generators

EBI's metagenomics workflow scripts -> CWL

https://www.ebi.ac.uk/metagenomics/pipelines/3.0

9522 lines of Python, BASH, and Perl code (data analysis workflows logic mixed with operational details

converted into

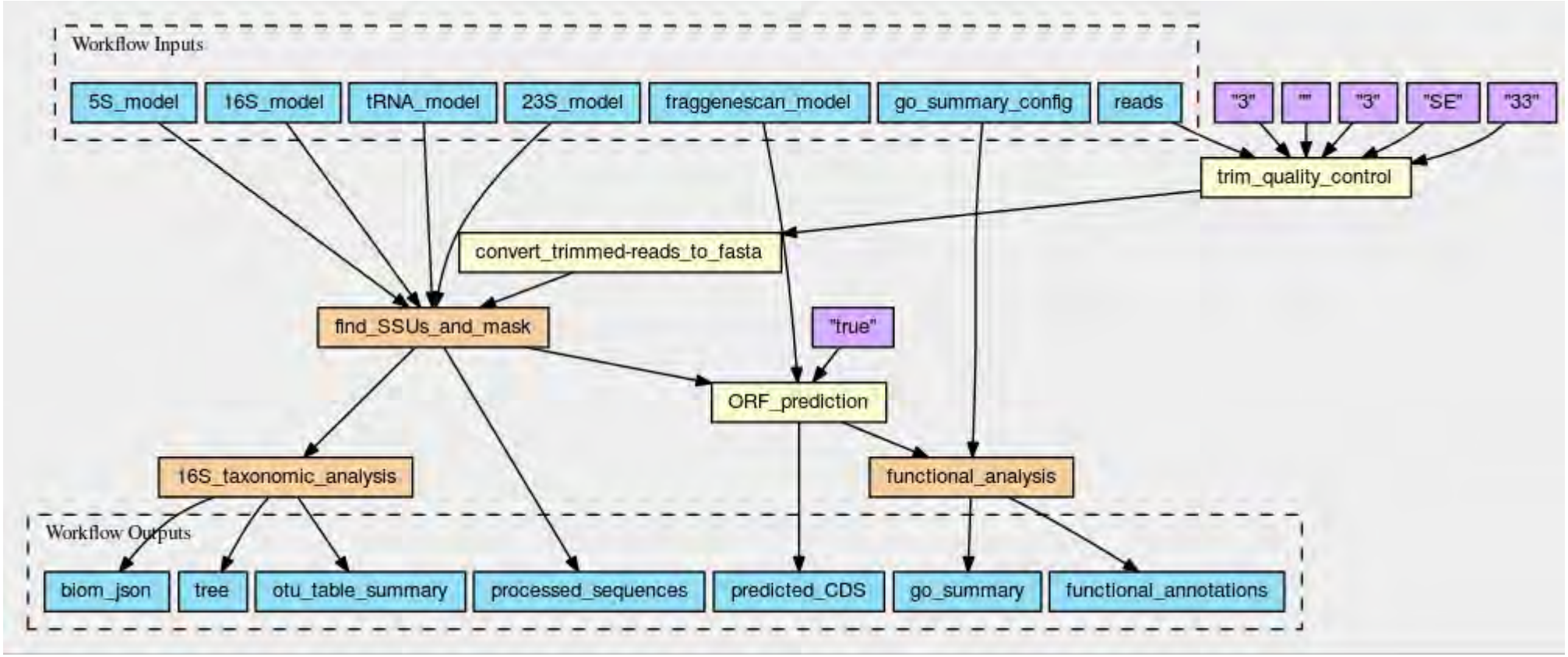
2560 lines of CWL descriptions

https://github.com/ProteinsWebTeam/ebi-metagenomics-cwl

(Lines of code counts via https://github.com/AlDanial/cloc#Stable)

https://tinyurl.com/NASEM-Workflows-CWL

EBI's metagenomics -> CWL project



Courtesy EMBL-EBI Metagenomics, visualization from

https://view.commonwl.org/workflows/github.com/ProteinsWebTeam/ebi-metagenomics-cwl/blob/master/workflows/rna-selector.cwl

https://tinyurl.com/NASEM-Workflows-CWL

From the Life Sciences...

https://tinyurl.com/NASEM-Workflows-CWL

...TO (ASTRO)PHYSICS AND BEYOND

netherlands eScience center

ASTRON

LOFAR

CWL Key Points

CWL, as a standard, allows us to move the interface between the researcher and the infrastructure to a much higher layer. **This frees the researcher to focus on their work and frees the e-infra providers to better optimize and balance their systems.**

This workflow standard already has a **growing ecosystem**: training materials (in three languages), visualizers, support for popular text editors and IDEs, standalone GUI, and more

ELIXIR (EUROPEAN) APPROACH TO SCIENCE STANDARDS

Standards are marketplace, pick and choose the ones that make sense

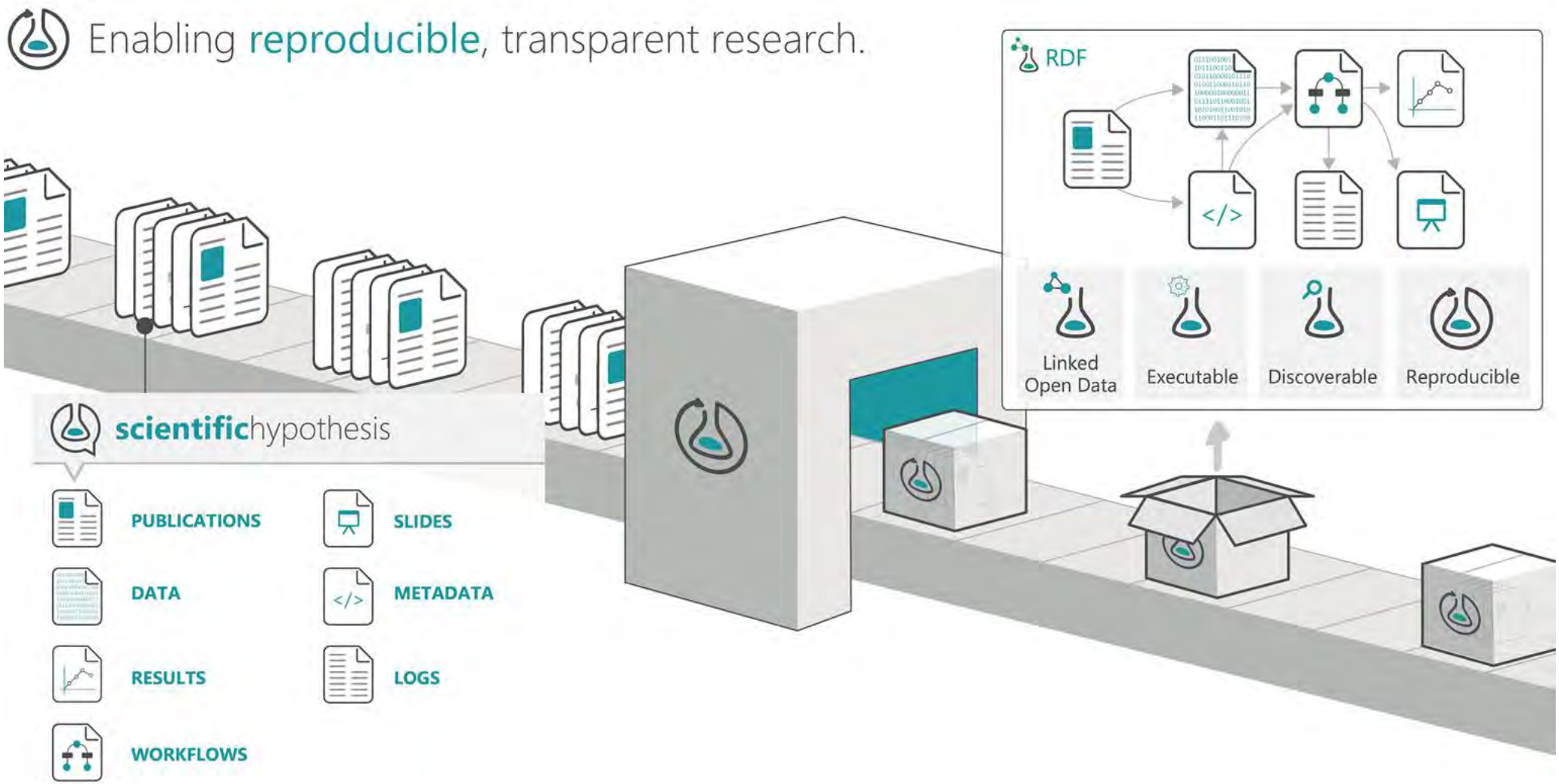
Specific (sub-)disciplines of science/research should lead the governance of standards that are specific to their area

Common needs and cross-discipline concerns should share standards and technology where possible

Thus we layer standards from the most generic to the most specific into a complete product.

https://tinyurl.com/NASEM-Workflows-CWL

ResearchObject.org standard overview



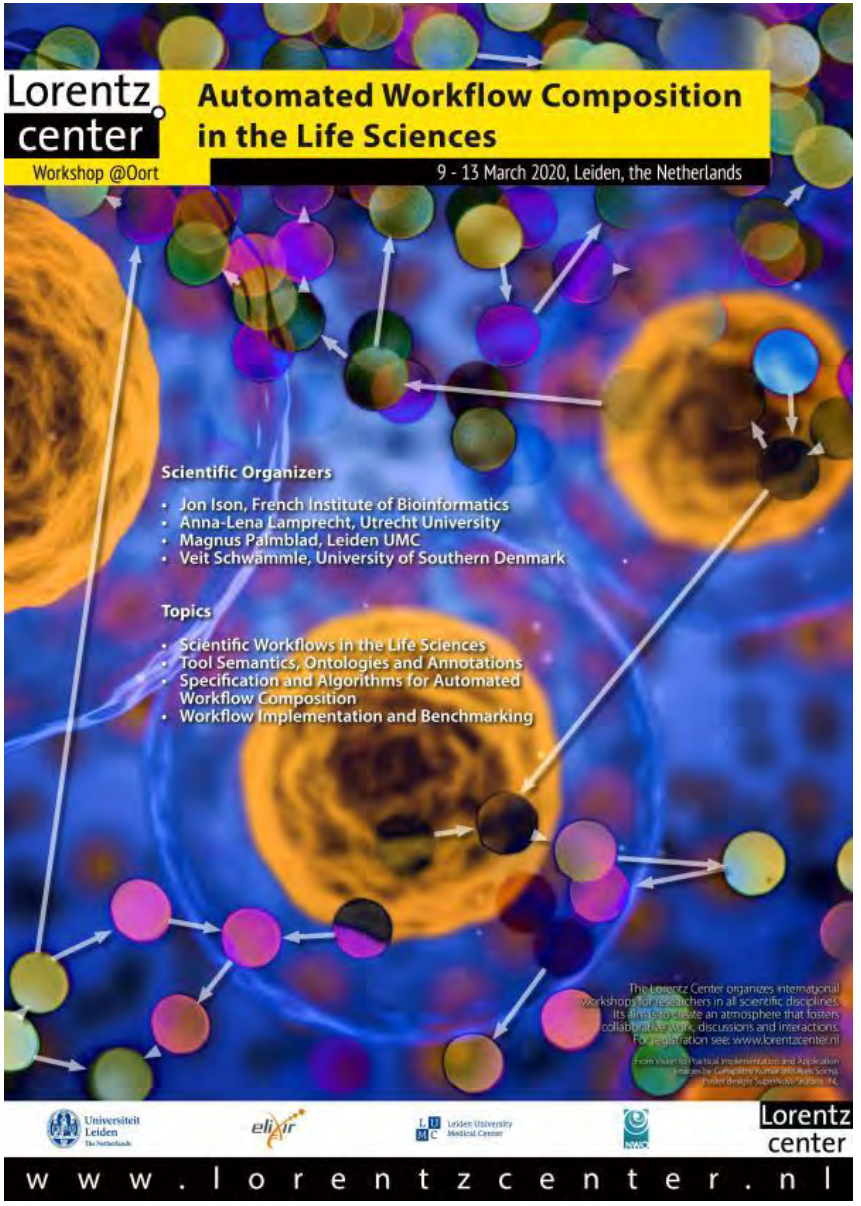
https://tinyurl.com/NASEM-Workflows-CWL

A RECENT AUTOMATED WORKFLOW WORKSHOP

Consensus:

Need to implement the **17 FAIR principles** widely to enable the automated / assisted composition of workflows.

Can't effectively do ML/AI without well labeled data and tools.



https://tinyurl.com/NASEM-Workflows-CWL

Thanks!

https://www.commonwl.org

Backup slides!

https://tinyurl.com/NASEM-Workflows-CWL

Michael R. Crusoe

From Phoenix, Arizona (Sonoran Desert), USA

Studied at Arizona State: Comp. Sci.; time in industry as a developer & system administrator (Google, others); returned to ASU for B.S. in Microbiology.

Introduced to bioinformatics via Anolis (lizard) genome assembly and analysis (Kenro Kusumi, Arizona State)

Returned to software engineering as a Research Software Engineer for k-h-mer project (C. Titus Brown, Michigan State, then U. of California, Davis)

Now based out of Berlin, Germany

Assisting EOSC Pilot, ELIXIR, ASTERICS

Common Workflow Language v1.1 standards

version 1.1 released on 2019-06-08

- “secondaryFiles” can now be explicitly marked as required or not.
- Parameter names on tools declared directly in a workflow no longer conflict with the enclosing workflow
- Addition of “stdin” type shortcut for CommandInputParameter
- Added “ToolTimeLimit” feature, allows setting an upper limit on the execution time of a CommandLineTool.

CWL v1.1 New Features, continued

- Added “WorkReuse” feature, allowing to enable or disable the reuse behavior for a particular tool or step for implementations that support reusing output from past work.
- Added “NetworkAccess” feature, allowing to indicate whether a process requires outgoing network access.
- The “position” field of the CommandLineBinding can now be calculated from a CWL Expression.
- The exit code of a CommandLineTool invocation is now available to expressions in outputEval as “$(runtime.exitCode)”

2 small changes from CWL v1.0.x

- The CWL runtime no longer loads listings of "Directory" objects by default.
  If you need the contents of the directory to manipulate in an expression, read about the "loadListing" field.
- CWL command line tools running inside containers now must have **network access disabled by default.** This was done to improve security and reproducibility. Use the new process requirement NetworkAccess to explicitly enable

CWL v1.1 Details

https://www.commonwl.org/v1.1/CommandLineTool.html#Changelog

https://www.commonwl.org/v1.1/Workflow.html#Changelog

Status of CWL v1.1 availability

CWL reference runner fully supports CWL v1.1 since version 1.0.**20190607**183319

Next release of Arvados will support v1.1

Toil v3.21.0a (unreleased) passes 226 of 253 of the CWL v1.1 conformance tests.

Support for CWL v1.1 in Seven Bridges is being developed by them

Galaxy-CWL project is targeting CWL v1.1 (no release timeline yet)

CWL v1.1 released 2019-06-07!

8 new features: `stdin` shortcut, secondary files can now be optional, `NetworkAccess`, `WorkReuse`, `LoadListingRequirement`, and `ToolTimeLimit` execution hints, `position` can be specified from a CWL Expression, addition of `runtime.exitcode`

50 cleanups and clarifications of corner cases in the specification

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

Support for CWL v1.1 in Arvados, toil-cwl-runner, and commercial providers is forthcoming

Funding

Currently, only one FTE! (M. Crusoe). Lots of in-kind donations from participant projects & vendors.

NGO/charity in the USA is legal home of the project (Software Freedom Conservancy, a 501(c)(3))

M. Crusoe recently formed a public enterprise in Lithuania (VšĮ "Darbo eigos") to assist with coordinating & funding CWL work in Europe.

CWL is a standards community & pan-discipline; most traditional funding sources don’t know what to do with us.

Linked Data & CWL

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

Example: can use the EDAM ontology (ELIXIR-DK) to specify file formats and reason about them:

“FASTQ Sanger” encoding is a type of FASTQ file

Example: samtools-sort.cwl

File type & metadata

```
class: CommandLineTool
cwlVersion: v1.0
doc: Sort by chromosomal coordinates
```

Runtime environment

```
hints:
  DockerRequirement:
    dockerPull: quay.io/cancercollaboratory/dockstore-tool-samtools-sort
```

Input parameters

```
inputs:
  aligned_sequences:
    type: File
    format: edam:format_2572  # BAM binary alignment format
    inputBinding:
      position: 1
```

Executable

```
baseCommand: [samtools, sort]
```

Output parameters

```
outputs:
  sorted_aligned_sequences:
    type: stdout
    format: edam:format_2572
```

Linked data support

```
$namespaces: { edam: "http://edamontology.org/" }
$schemas: [ "http://edamontology.org/EDAM_1.15.owl" ]
```

File type & metadata

```
class: CommandLineTool
cwlVersion: v1.0
doc: Sort by chromosomal coordinates
```

- Identify as a CommandLineTool object
- Core spec includes simple comments
- Metadata about tool extensible to arbitrary RDF vocabularies, e.g.
  - Biotools & EDAM
  - Dublin Core Terms (DCT)
  - Description of a Project (DOAP)

Runtime Environment

```
hints:
  DockerRequirement:
    dockerPull: quay.io/[…]samtools-sort
```

- Define the execution environment of the tool
- “requirements” must be fulfilled or an error
- “hints” are soft requirements (express preference but not an error if not satisfied)
- Also used to enable optional CWL features
  - Mechanism for defining extensions

Input parameters

```
inputs:
  aligned_sequences:
    type: File
    format: edam:format_2572  # BAM binary format
    inputBinding:
      position: 1
```

- Specify name & type of input parameters
  - Based on the Apache Avro type system
  - null, boolean, int, string, float, array, record
  - File formats can be IANA Media/MIME types, or from domain specific ontologies, like EDAM for bioinformatics
- “inputBinding”: describes how to turn parameter value into actual command line argument

Example: samtools-sort.cwl

File type & metadata

```
class: CommandLineTool
cwlVersion: v1.0
doc: Sort by chromosomal coordinates
```

Runtime environment

```
hints:
  DockerRequirement:
    dockerPull: quay.io/cancercollaboratory/dockstore-tool-samtools-sort
```

Input parameters

```
inputs:
  aligned_sequences:
    type: File
    format: edam:format_2572  # BAM binary alignment format
    inputBinding:
      position: 1
```

Executable

```
baseCommand: [samtools, sort]
```

Output parameters

```
outputs:
  sorted_aligned_sequences:
    type: stdout
    format: edam:format_2572
```

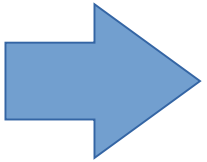
Linked data support

```
$namespaces: { edam: "http://edamontology.org/" }
$schemas: [ "http://edamontology.org/EDAM_1.15.owl" ]
```

Command Line Building

Input object

```
aligned_sequences:
  class: File
  location: example.bam
  format: http://edamontology.org/format_2572
```



```
inputs:
  aligned_sequences:
    type: File
    format: edam:format_2572
    inputBinding:
      position: 1
```

```
baseCommand: [samtools, sort]
```

- Associate input values with parameters
- Apply input bindings to generate strings
- Sort by “position”
- Prefix “base command”

```
[“samtools”, “sort”, “example.bam”]
```

Output parameters

```
outputs:
  sorted_aligned_sequences:
    type: stdout
    format: edam:format_2572
```

- Specify name & type of output parameters
- In this example, capture the STDOUT stream from “samtools sort” and tag it as being BAM formatted.

https://tinyurl.com/NASEM-Workflows-CWL

Workflows

- Specify data dependencies between steps
- Scatter/gather on steps
- Can nest workflows in steps
- Still working on:
- Conditionals & looping

Example: grep & count

```
class: Workflow
cwlVersion: v1.0
```

```
requirements:
  - class: ScatterFeatureRequirement
```

```
inputs:
  pattern: string
  infiles: File[]
```

```
outputs:
  outfile:
    type: File
    outputSource: wc/outfile
```

```
steps:
  grep:
    run: grep.cwl
    in:
      pattern: pattern
      infile: infiles
    scatter: infile
    out: [outfile]

  wc:
    run: wc.cwl
    in:
      infiles: grep/outfile
    out: [outfile]
```

Source file: https://github.com/common-workflow-language/workflows/blob/2855f2c3ea875128ff62101295897d8d11d99b94/workflows/presentation-demo/grep-and-count.cwl

https://tinyurl.com/NASEM-Workflows-CWL

Example: grep & count

```
class: Workflow
cwlVersion: v1.0
```

```
requirements:
  - class: ScatterFeatureRequirement
```

```
inputs:
  pattern: string
  infiles: File[]
```

```
outputs:
  outfile:
    type: File
    outputSource: wc/outfile
```

```
steps:
  grep:
    run: grep.cwl
    in:
      pattern: pattern
      infile: infiles
    scatter: infile
    out: [outfile]

  wc:
    run: wc.cwl
    in:
      infiles: grep/outfile
    out: [outfile]
```

Tool to run

Scatter over input array

Connect output of “grep” to input of “wc”

Connect output of “wc” to workflow output

Use Cases for the CWL standards

Publication reproducibility, reusability

Workflow creation & improvement across institutions and continents

Contests & challenges

Analysis on non-public data sets, possibly using GA4GH job & workflow submission API

How to search for a tool, or for a workflow

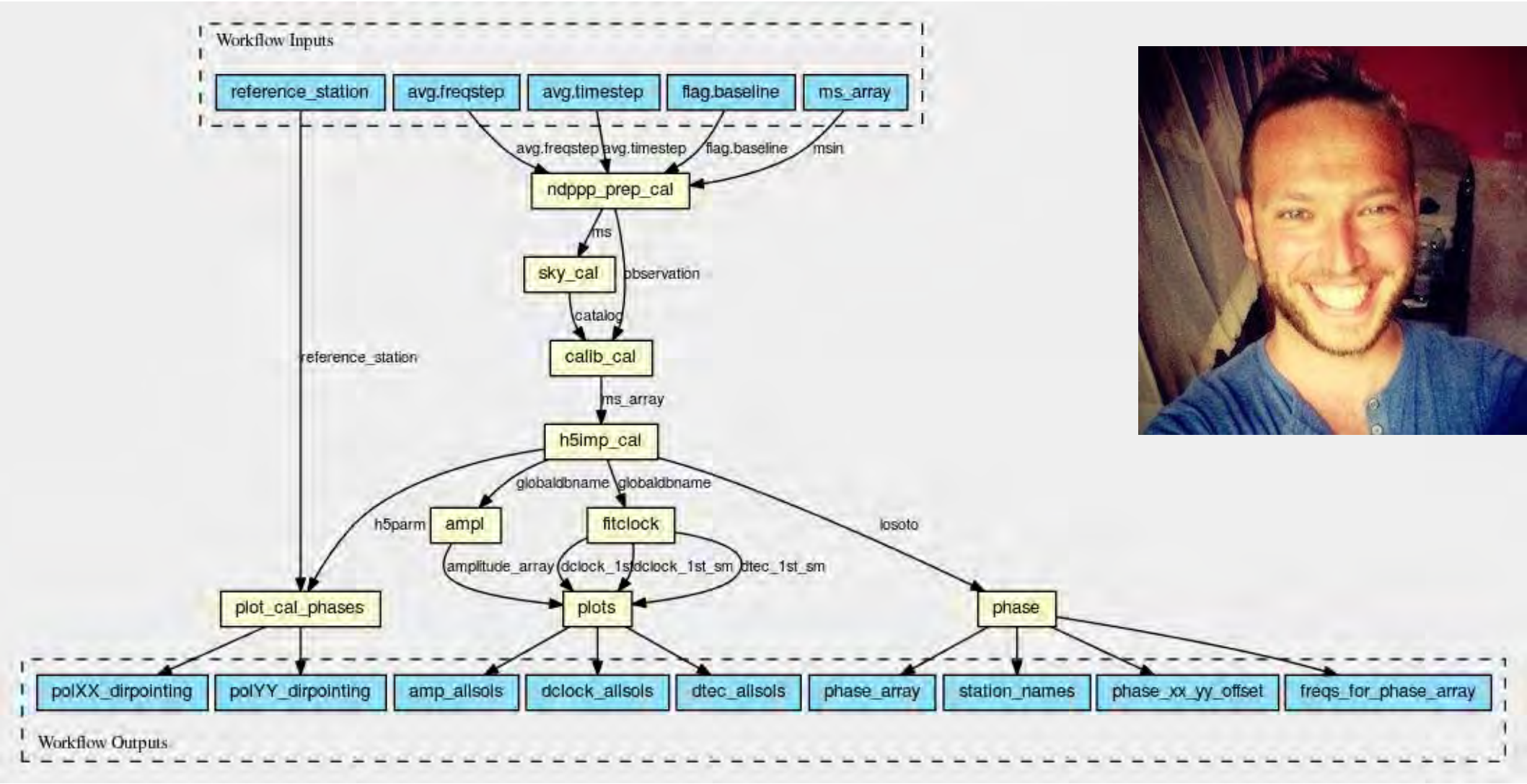
GitHub
Search for CWL documents using **extension:cwl cwlVersion + <your search terms>**, for example **extension:cwl cwlVersion picard**.

Google
Search for CWL documents using **filetype:cwl cwlVersion + <your search terms>**, for example **filetype:cwl cwlVersion picard**

Can also browse https://view.commonwl.org/workflows

https://tinyurl.com/NASEM-Workflows-CWL

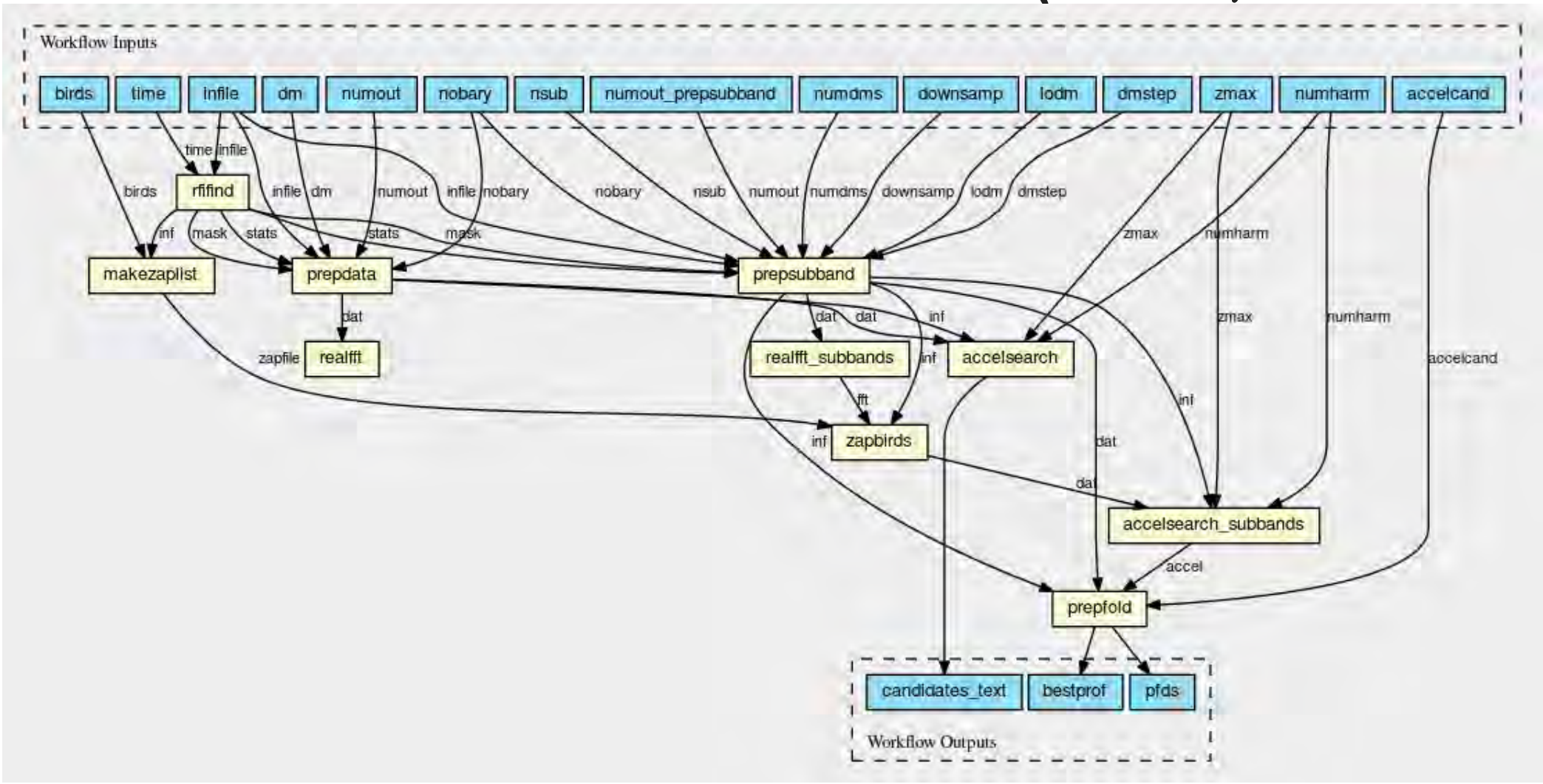
The LOFAR pre-facet calibration pipeline



https://view.commonwl.org/workflows/github.com/EOSC-LOFAR/prefactor-cwl/blob/63fcfba8e17a18343eede9cc3bf1c81a10dc2dcc/prefactor.cwl

https://tinyurl.com/NASEM-Workflows-CWL

Searching for Pulsars with PRESTO (& CWL)



https://view.commonwl.org/workflows/github.com/EOSC-LOFAR/prefactor-cwl/blob/63fcfba8e17a18343eede9cc3bf1c81a10dc2dcc/prefactor.cwl (in progress)

The LOFAR pre-facet calibration pipeline

LOFAR pipelines currently written in ‘parsets’ language **unique to that team**

Gijs Molenaar packaged the software in the KERN suite (3rd party software packages for Ubuntu Linux LTS) and used those packages to create Docker/Singularity containers

Gijs (with some assistance from me) then converted the “parset” based pipeline to a Common Workflow Language version

The CWL model for tools

CWL tool descriptions turn POSIX† command-line data analysis tools into functions

- well defined and named inputs & outputs
- typed

These inputs and outputs are connected into “data flow” style workflows

†The reference CWL runner runs on Microsoft Windows using Docker software containers

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 where or near the location of data.

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

Software Containers & CWL

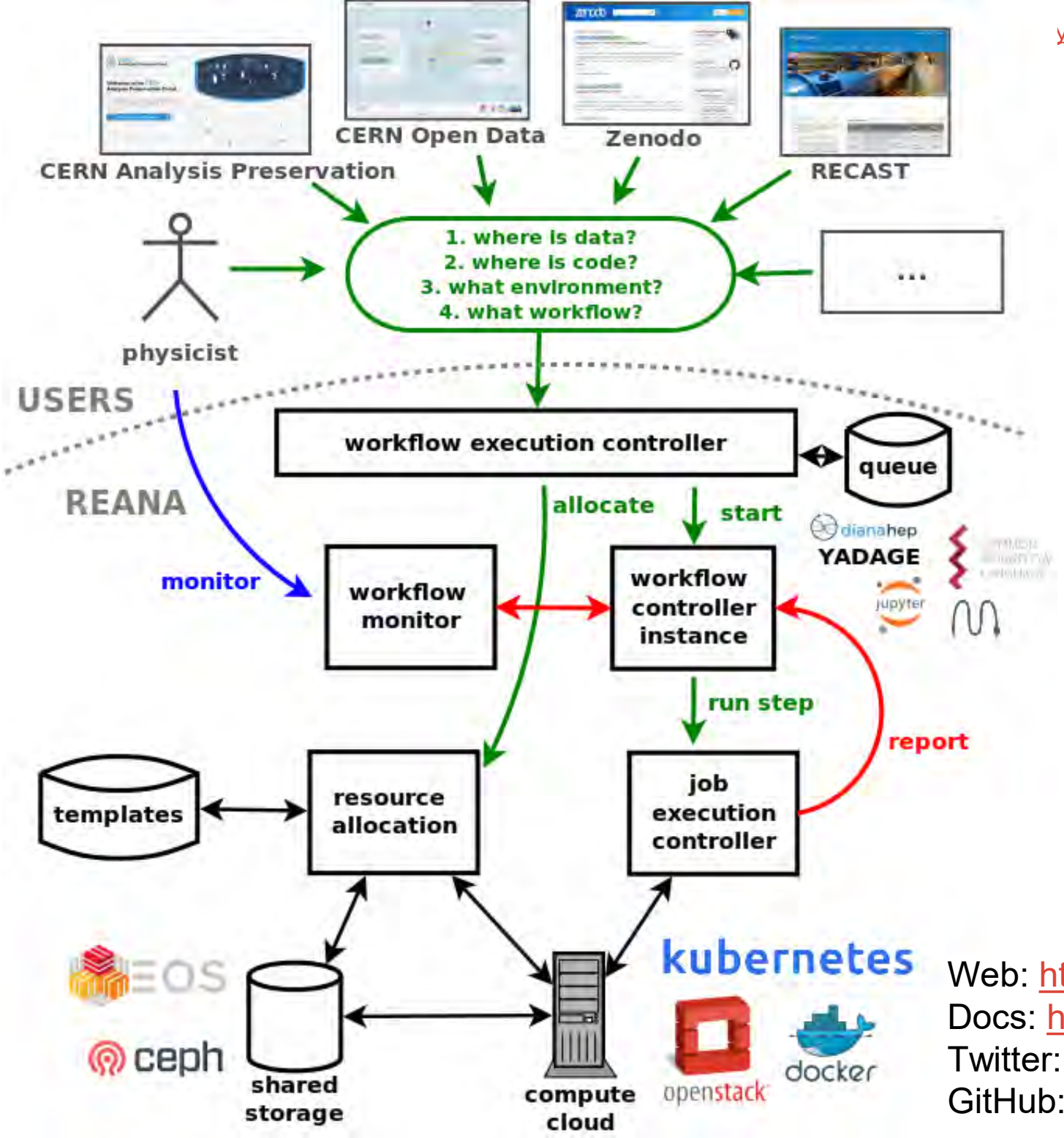
CWL v1.0.x has built in (optional) support for Docker software containers. The CWL reference runner has support for running Docker containers using the Docker, Singularity, **uDocker**, or dx-docker runtimes.

CWL descriptions can also contain more generic software requirements; can be used to make applications available using Docker, **Singularity**, **(bio)conda**, Debian, etc.

http://www.commonwl.org/v1.0/CommandLineTool.html#SoftwareRequirement

Example with reference CWL runner:
https://github.com/common-workflow-language/cwltool#leveraging-softwarerequirements-beta

yurl.com/NASEM-Workflows-CWL



Web: http://reana.io
Docs: http://reana.readthedocs.io
Twitter: https://twitter.com/reanahub
GitHub: https://github.com/reanahub

Extensibility a core feature

Vendors are encouraged to develop new features as well marked extensions.

(Inspired by modern web standards development practices)

These extensions are then candidates for inclusion as official extensions, or perhaps required elements of a future version of the standard.

Example
arv:ReuseRequirement is part of CWL v1.1 as WorkReuse.

CWL + Provenance = CWLProv

https://doi.org/10.1093/gigascience/giz095

https://slides.com/soilandreyes/2019-06-10-ro-or2019#/