



# COMMON WORKFLOW LANGUAGE

**Opening up complex data analysis methods  
with the CWL standards**

Michael R. Crusoe      CWL Project Lead  
2017-10-23      [@biocrusoe](#) / [#CommonWL](#)  
OpenCon 2017 Lithuania

# MICHAEL R. CRUSOE, WHO IS THIS GUY?

<https://goo.gl/55pd52>



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 & received B.S. in Microbiology.

Co-author of an identity card **standard** for use by seafarers; accompanying ILO convention ratified by 30+ countries

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)

# "WORKFLOWS"

We use the word "workflows" as a shorthand for:

the collection of computer applications, scripts, and code used in **computational data analysis**

- how the applications are configured
- and how the data flows between them

(primarily in a research/scientific context)

# WHY USE A WORKFLOW MANAGEMENT SYSTEM?

Features **can** include:

**separation of concerns:** focus on the science being done first; then optimize execution later

**automatic job execution:** start a complicated analysis involving many pieces with a single command

**scaling** (across nodes, clusters, and possibly continents)

**automatically generated graphical user interfaces**  
(example: [Galaxy](#))

How was this file made? (**automatic provenance tracking**)

# 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)

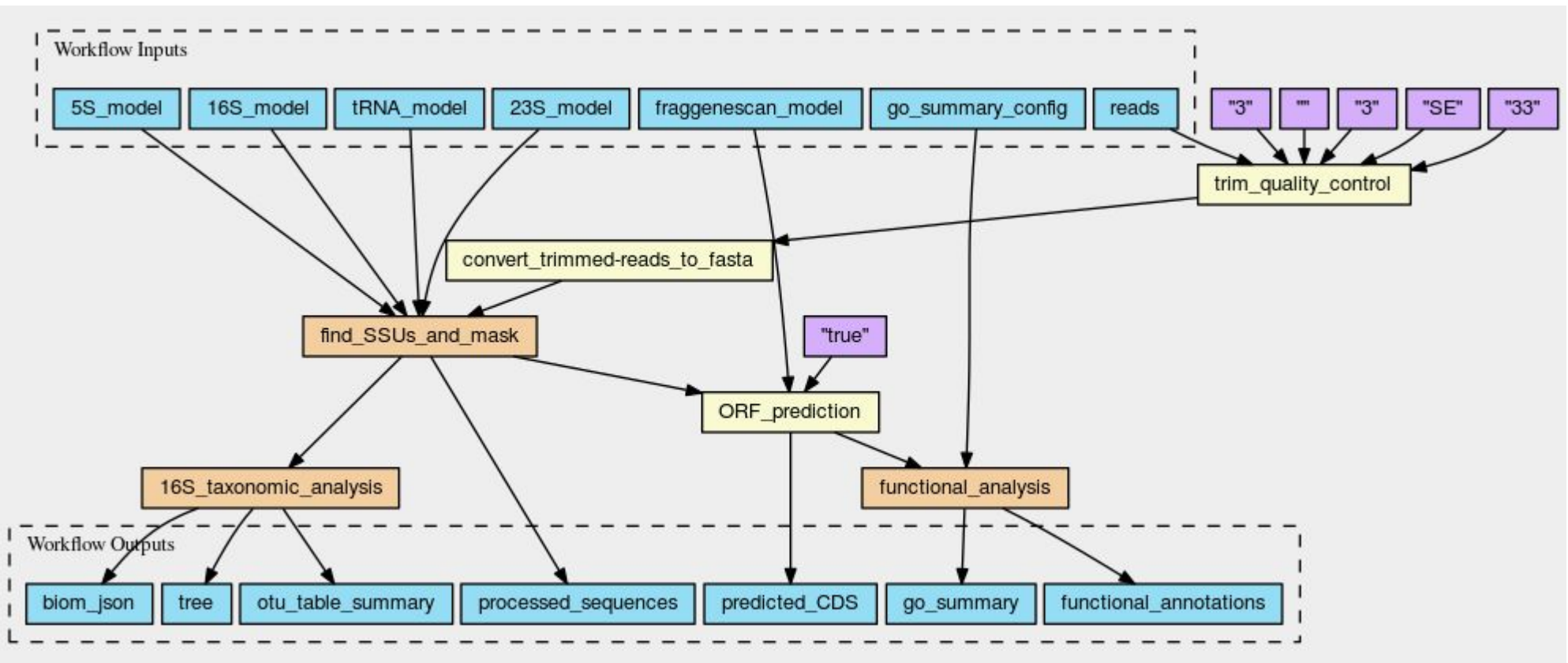
converted into

**2560** lines of CWL descriptions

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

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

# 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>

# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

1. Arvados <http://arvados.org>
2. Taverna <http://www.taverna.org.uk/>
3. Galaxy <http://galaxyproject.org/>
4. SHIWA <https://www.shiwa-workflow.eu/>
5. Oozie <https://oozie.apache.org/>
6. DNANexus [https://wiki.dnanexus.com/API-Specification-v1.0.0/IO-and-Run-Specifications#](https://wiki.dnanexus.com/API-Specification-v1.0.0/IO-and-Run-Specifications#https://wiki.dnanexus.com/API-Specification-v1.0.0/Workflows-and-Analyses#)  
<https://wiki.dnanexus.com/API-Specification-v1.0.0/Workflows-and-Analyses#>
7. BioDT <http://www.biodatomics.com/>
8. Agave <http://agaveapi.co/live-docs/>
9. DiscoveryEnvironment <http://www.iplantcollaborative.org/ci/discovery-environment>
10. Wings <http://www.wings-workflows.org/>
11. Knime <https://www.knime.org/>
12. make, rake, drake, ant, scons & many others. Software development relies heavily on tools to manage workflows related to compiling and packaging applications. For the most part these are file based and usually run on a single node, usually supporting parallel steps (make -j) and in some cases able to dispatch build steps to other machines  
(<https://code.google.com/p/distcc/>) (<https://github.com/Factual/drake>)
13. Snakemake <https://bitbucket.org/snakemake/snakemake>
14. BPIPE <http://bpipe.org>
15. Ruffus <https://code.google.com/p/ruffus/>
16. NextFlow <http://nextflow.io>
17. Luigi <http://github.com/spotify/luigi>

<https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems>

# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

18. SciLuigi. Helper library built on top of Luigi to ease development of Scientific workflows in Luigi: <http://github.com/samuell/sciluigi>
19. GATK Queue <https://www.broadinstitute.org/gatk/guide/topic?name=queue>
20. Yabi <https://ccg.murdoch.edu.au/yabi>
21. seqware Workflows are written in Java and executed using the Oozie Workflow Engine on Hadoop or SGE clusters. Uses Zip64 files to group the workflow definition file, workflow itself, sample settings, and data dependencies in a single file that can be exchanged between SeqWare users or archived. <https://seqware.github.io/> <https://seqware.github.io/docs/6-pipeline/>
22. Ketrew <https://github.com/hammerlab/ketrew>
23. Pegasus <http://pegasus.isi.edu/>
24. Airflow <https://github.com/airbnb/airflow>
25. Cosmos  
<https://cosmos.hms.harvard.edu/documentation/index.html><http://bioinformatics.oxfordjournals.org/content/early/2014/07/24/bioinformatics.btu385.full> [paper] Cosmos2: <https://github.com/LPM-HMS/COSMOS2> <http://cosmos.hms.harvard.edu/COSMOS2/>
26. Pinball <https://github.com/pinterest/pinball>
27. bcbio <https://bcbio-nextgen.readthedocs.org/en/latest/>
28. Chronos <https://github.com/mesos/chronos>
29. Azkaban <https://azkaban.github.io/>
30. Apache NiFi <https://nifi.apache.org/docs/nifi-docs/html/overview.html>
31. flowr (R-based) <http://docs.flowr.space/> <https://github.com/sahilseth/flowr>
32. Mistral  
<https://github.com/arteria-project>[https://wiki.openstack.org/wiki/Mistral#What\\_is\\_Mistral.3F](https://wiki.openstack.org/wiki/Mistral#What_is_Mistral.3F)<https://wiki.openstack.org/wiki/Mistral/DSLv2>
33. nipype <http://nipype.org/nipype/>
34. End of Day <https://github.com/joestubbs/endofday>
35. BioDSL <https://github.com/maasha/BioDSL>
36. BigDataScript <http://pcingola.github.io/BigDataScript/>
37. Omics Pipe: uses Ruffus <http://sulab.scripps.edu/omicspipe/>
38. Ensembl Hive <https://github.com/Ensembl/ensembl-hive>



# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

39. QuickNGS <http://bifacility.uni-koeln.de/quickngs/web>
40. GenePattern <http://www.broadinstitute.org/cancer/software/genepattern/>
41. Chipster <http://chipster.csc.fi/>
42. The Genome Modeling System <https://github.com/genome/gms>
43. Cuneiform, A Functional Workflow Language <https://github.com/joergen7/cuneiform><http://www.cuneiform-lang.org/>
44. Anvaya  
<http://www.ncbi.nlm.nih.gov/pubmed/22809419>[http://webapp.cabgrid.res.in/biocomp/Anvaya/ANVAYA\\_Main.html#HOWTO\\_INSTALL\\_ANVAYA](http://webapp.cabgrid.res.in/biocomp/Anvaya/ANVAYA_Main.html#HOWTO_INSTALL_ANVAYA)
45. Makeflow <http://ccl.cse.nd.edu/software/makeflow/>
46. Airavata <http://airavata.apache.org/>
47. Pyflow <https://github.com/Illumina/pyflow>
48. Cluster Flow <http://clusterflow.io>
49. Unipro UGENE <http://ugene.net/> <https://dx.doi.org/10.7717/peerj.644>
50. CloudSlang <http://www.cloudslang.io/>
51. Stacks <http://catchenlab.life.illinois.edu/stacks/>
52. Leaf <http://www.francesconapolitano.it/leaf/index.html>
53. omictools <http://omictools.com/>
54. Job Description Language. The Job Description Language, JDL, is a high-level, user-oriented language based on Condor classified advertisements for describing jobs and aggregates of jobs such as Direct Acyclic Graphs and Collections.  
<https://edms.cern.ch/ui/file/590869/1/WMS-JDL.pdf>
55. YAWL yet another workflow language <http://dx.doi.org/10.1016/j.is.2004.02.002><http://www.yawlfoundation.org/>
56. Triquetrum <https://projects.eclipse.org/projects/technology.triquetrum><https://github.com/eclipse/triquetrum/>
57. Kronos <https://github.com/jtaghiyar/kronos>
58. qsubsec <http://doi.org/10.1093/bioinformatics/btv698> <https://github.com/alastair-droop/qsubsec>
59. YesWorkflow <http://yesworkflow.org>
60. GWF - Grid WorkFlow <https://github.com/mailund/gwf> <http://mailund.github.io/gwf/>
61. Fireworks. <https://pythonhosted.org/FireWorks/>

# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

- 62. NGLess: NGS with less work <http://ngless.rtf.d.io>
- 63. pypipegraph <https://github.com/TyberiusPrime/pypipegraph>
- 64. Cromwell <https://github.com/broadinstitute/cromwell>
- 65. Dagobah - Simple DAG-based job scheduler in Python. <https://github.com/thieman/dagobah>
- 66. sushi <https://github.com/uzh/sushi>
- 67. Clinical Trial Processor - A program for processing clinical trials data. [http://mirwiki.rsna.org/index.php?title=MIRC\\_CTP](http://mirwiki.rsna.org/index.php?title=MIRC_CTP)
- 68. Noodles <http://nlesc.github.io/noodles/>
- 69. Swift <http://swift-lang.org/main/>
- 70. Consonance (runs SeqWare & CWL) <https://github.com/Consonance/consonance/wiki>
- 71. Dog <https://github.com/dogtools/dog>
- 72. Produce <https://github.com/texttheater/produce>
- 73. LONI Pipeline <http://pipeline.loni.usc.edu/>
- 74. Cpipe <https://github.com/MelbourneGenomics/cpipe>
- 75. AWE <https://github.com/MG-RAST/AWE>
- 76. (Py)COMPSs <https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar/>
- 77. KLIKO <https://github.com/gijzelaerr/kliko>
- 78. Script of Scripts <https://github.com/BoPeng/SOS> <http://vatlab.github.io/SOS/>
- 79. XNAT Pipeline Engine  
<https://wiki.xnat.org/display/XNAT/Pipeline+Engine><https://wiki.xnat.org/display/XNAT/XNAT+Pipeline+Development+Schema>
- 80. Metapipeline <https://github.com/TorkamaniLab/metapipeline>
- 81. OCCAM (Open Curation for Computer Architecture Modeling) <https://occam.cs.pitt.edu/>
- 82. Copernicus <http://www.copernicus-computing.org>
- 83. iRODS Rule Language <https://github.com/samuell/irods-cheatsheets/blob/master/irods-rule-lang-full-guide.md>
- 84. VisTrails <https://www.vistrails.org>
- 85. Bionode Watermill <https://github.com/bionode/bionode-watermill>

# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

86. BIOVIA Pipeline Pilot Overview <http://accelrys.com/products/collaborative-science/biovia-pipeline-pilot/>
87. Dagman A meta-scheduler for HTCondor <https://research.cs.wisc.edu/htcondor/dagman/dagman.html>
88. UNICORE [https://www.unicore.eu/docstore/workflow-7.6.0/workflow-manual.html#wf\\_dialect](https://www.unicore.eu/docstore/workflow-7.6.0/workflow-manual.html#wf_dialect)
89. Toil (A scalable, efficient, cross-platform and easy-to-use workflow engine in pure Python) <https://github.com/BD2KGenomics/toil>
90. Cylc <https://cylc.github.io/cylc/>
91. Autodesk Cloud Compute Canon <https://github.com/Autodesk/cloud-compute-cannon>
92. Civet <https://github.com/TheJacksonLaboratory/civet>
93. Cumulus <https://github.com/Kitware/cumulus>
94. High-performance integrated virtual environment (HIVE) <https://hive.biochemistry.gwu.edu>
95. Cloudgene <http://cloudgene.uibk.ac.at/cloudgene-yaml>
96. FASTR [https://bitbucket.org/bigr\\_erasmusmc/fastr/](https://bitbucket.org/bigr_erasmusmc/fastr/) <http://fastr.readthedocs.io/en/stable/>
97. BioMake <https://github.com/evoldoers/biomake> <http://dx.doi.org/10.1101/093245>
98. remake <https://github.com/richfitz/remake>
99. SciFloware <http://www-sop.inria.fr/members/Didier.Parigot/pmwiki/Scifloware/>
100. OpenAlea <http://openalea.gforge.inria.fr/dokuwiki/doku.php>  
<https://hal.archives-ouvertes.fr/hal-01166298/file/openalea-PradalCohen-Boulakia.pdf>
101. COMBUSTIO <https://github.com/jarlebass/combustio> <http://hdl.handle.net/10037/9361>
102. BioCloud <https://github.com/ccwang002/biocloud-server-kai> <http://doi.org/10.6342/NTU201601295>
103. Triana <http://www.trianacode.org/>
104. Kepler <https://kepler-project.org/>
105. Anduril <http://anduril.org/site/>
106. dgsh <http://www.dmst.aueb.gr/dds/sw/dgsh/>
107. EDGE bioinformatics: Empowering the Development of Genomics Expertise [https://bioedge.lanl.gov/edge\\_ui/](https://bioedge.lanl.gov/edge_ui/) <http://edge.readthedocs.io/>  
<https://lanl-bioinformatics.github.io/EDGE/>
108. Pachyderm <http://pachyderm.io/http://pachyderm.readthedocs.io/en/stable/advanced/advanced.html>

# EXISTING COMPUTATIONAL RESEARCH WORKFLOW SYSTEMS

- 109. Digdag <https://www.digdag.io/>
- 110. Agua / Automated Genomics Utilities Agent <http://aguadev.org>
- 111. BioDepot Workflow Builder (BwB) <https://github.com/BioDepot/BioDepot-workflow-builder><https://doi.org/10.1101/099010>
- 112. IMP: a pipeline for reproducible reference-independent integrated metagenomic and metatranscriptomic analyses  
<http://r3lab.uni.lu/web/imp/> <https://doi.org/10.1186/s13059-016-1116-8>
- 113. Butler <https://github.com/llevar/butler>
- 114. adage / yadage <https://github.com/diana-hep/adage> <https://github.com/diana-hep/yadage>
- 115. HI-WAY: Execution of Scientific Workflows on Hadoop YARN  
<https://github.com/marcbux/Hi-WAY><https://openproceedings.org/2017/conf/edbt/paper-248.pdf>
- 116. OpenMOLE <https://github.com/openmole/openmole> <https://www.openmole.org/><https://doi.org/10.3389/fninf.2017.00021>
- 117. Biopet <https://github.com/biopet/biopet>
- 118. Nephele <https://nephele.niaid.nih.gov/>
- 119. TOPPAS <http://doi.org/10.1021/pr300187f>
- 120. SBpipe <https://pdp10.github.io/sbpipe/> <https://github.com/pdp10/sbpipe><https://doi.org/10.1186/s12918-017-0423-3>
- 121. Dray <http://dray.it/>
- 122. GenomeVIP <https://github.com/ding-lab/GenomeVIP> <https://doi.org/10.1101/gr.211656.116>
- 123. GridSAM <https://sourceforge.net/projects/gridsam/>
- 124. Roddy <https://github.com/eilslabs/Roddy>
- 125. SciFlo (historical; doesn't seem to be maintained anymore)  
<https://web.archive.org/web/20161118011409/https://sciflo.jpl.nasa.gov/SciFloWiki/FrontPage>

<https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems>

# 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

# COMMON WORKFLOW LANGUAGE V1.0

- 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, academic clusters, cloud environments, and local execution
- Supports the use of containers (e.g. Docker) and shared research computing clusters with locally installed software

# FROM THE LIFE SCIENCES...

**CUROVERSE™**

**Galaxy**  
PROJECT

SevenBridges  
genomics



Institut Pasteur





# ELIXIR: European infrastructure for biological information

Data infrastructure for Europe's life-science research:



*Data*



*Interoperability*



*Tools*



*Compute*



*Training*



*Marine metagenomics*



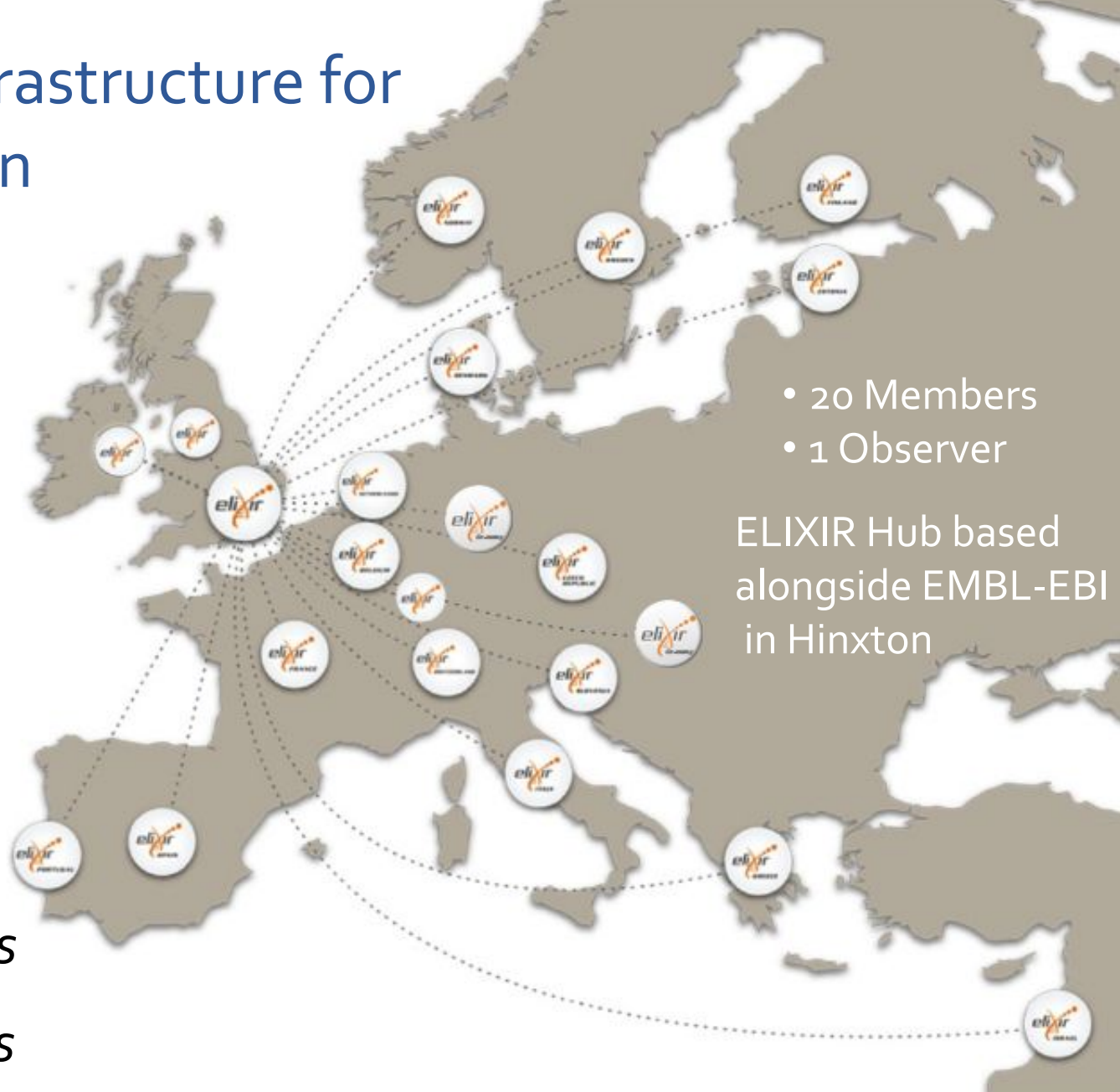
*Crop and forest plants*



*Human data*



*Rare diseases*



- 20 Members
- 1 Observer

ELIXIR Hub based alongside EMBL-EBI in Hinxton



[www.elixir-europe.org](http://www.elixir-europe.org)



[@ELIXIREurope](https://twitter.com/ELIXIREurope)





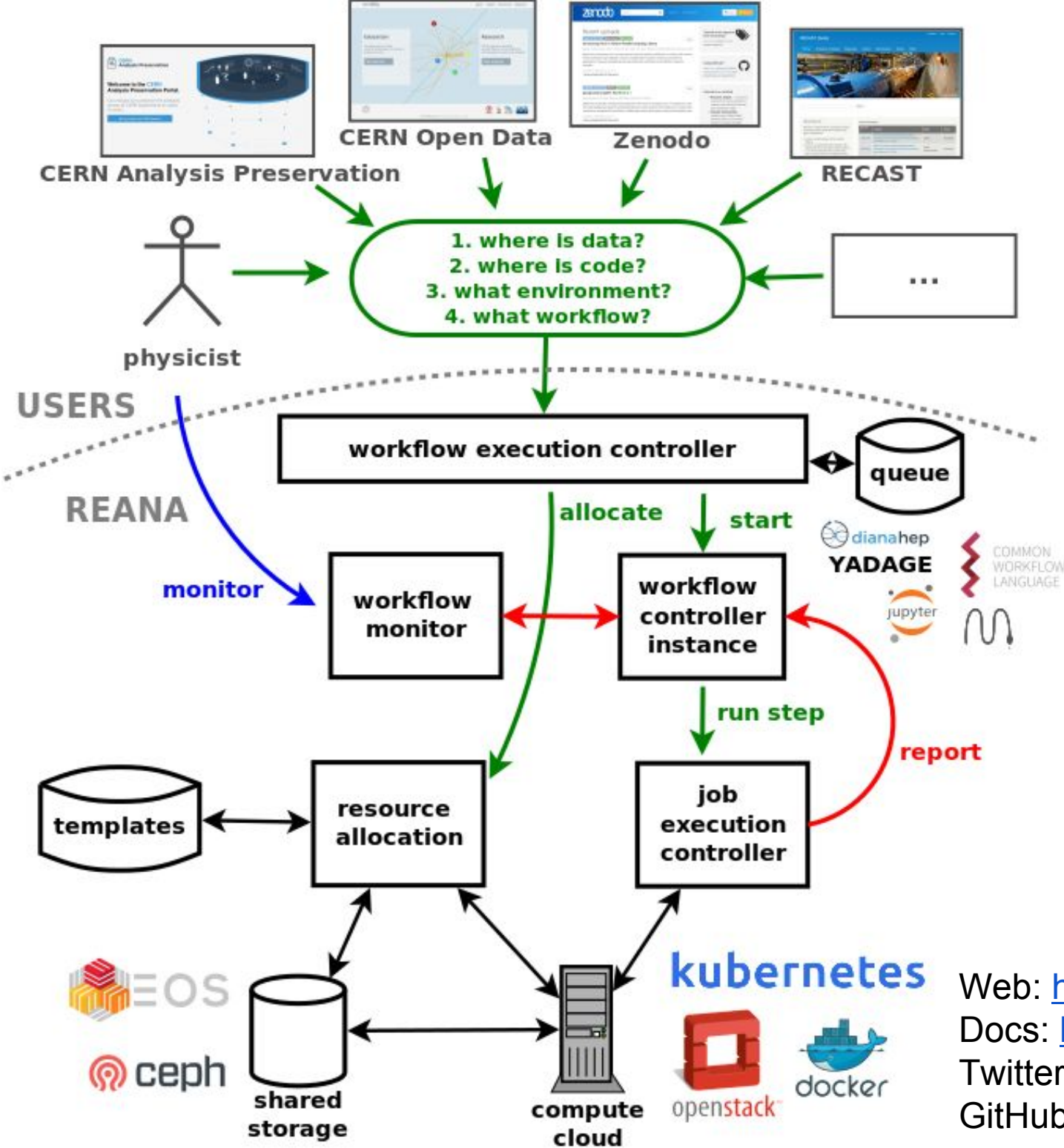
...TO (ASTRO)PHYSICS AND BEYOND

netherlands

eScience center

ASTRON  LOFAR





Web: <http://reana.io>

Docs: <http://reana.readthedocs.io>

Twitter: <https://twitter.com/reanahub>

GitHub: <https://github.com/reanahub>

# THE CWL MODEL

CWL tool descriptions turn POSIX<sup>†</sup> 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

<sup>†</sup>The reference CWL runner runs on Microsoft Windows using Docker software containers

# WELL DESCRIBED 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](http://www.commonwl.org/v1.0/CommandLineTool.html#Runtime_environment)

# 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)

# CWL FEATURES THAT SUPPORT FAIR & EOSC PRINCIPLES

- Identifiers for data, software, and authors
- “plug in” support for community specific ontologies and formalized vocabularies
- Vendor & platform independent; grid, cluster, and/or cloud

# 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

# WHY USE THE COMMON WORKFLOW LANGUAGE?

Develop your pipeline on your local computer  
(optionally with containers)

Execute on your research cluster or in the cloud

Deliver to users via workbenches like Arvados, Rabix,  
Toil. Galaxy, Apache Taverna, AWE, Funnel (GCP)  
support is in alpha stage.



# 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

# 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

# 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](#)

[illegible]

# SOFTWARE CONTAINERS & CWL

CWL v1.0.x has built in (optional) support for Docker software containers

CWL descriptions can also contain more generic software requirements; can be used to make applications available using Docker, Singularity, conda, Debian, or any other packaging system (like **CVMFS**).

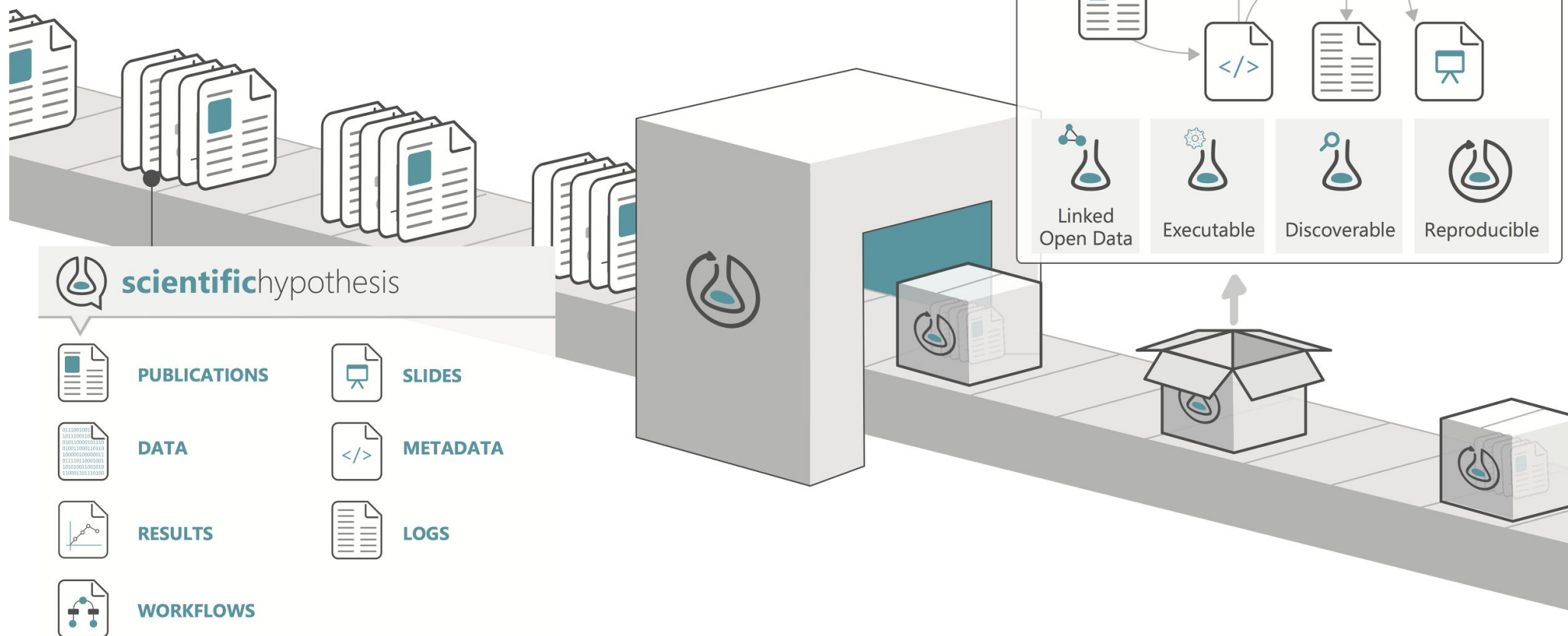
<http://www.commonwl.org/v1.0/CommandLineTool.html#SoftwareRequirement>

Example with reference CWL runner:

<https://github.com/common-workflow-language/cwltool#leveraging-softwarerequirements-beta>

# RESEARCHOBJECT.ORG STANDARD OVERVIEW

 Enabling **reproducible**, transparent research.



# 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.
  - Bi.tools & EDAM
  - Dublin Core Terms (DCT)
  - Description of a Project (DOAP)
- GA4GH Tool Registry project will develop best practices for metadata & attribution



# 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
```

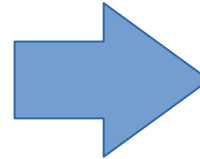
## Linked data support

```
$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.

# 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
```

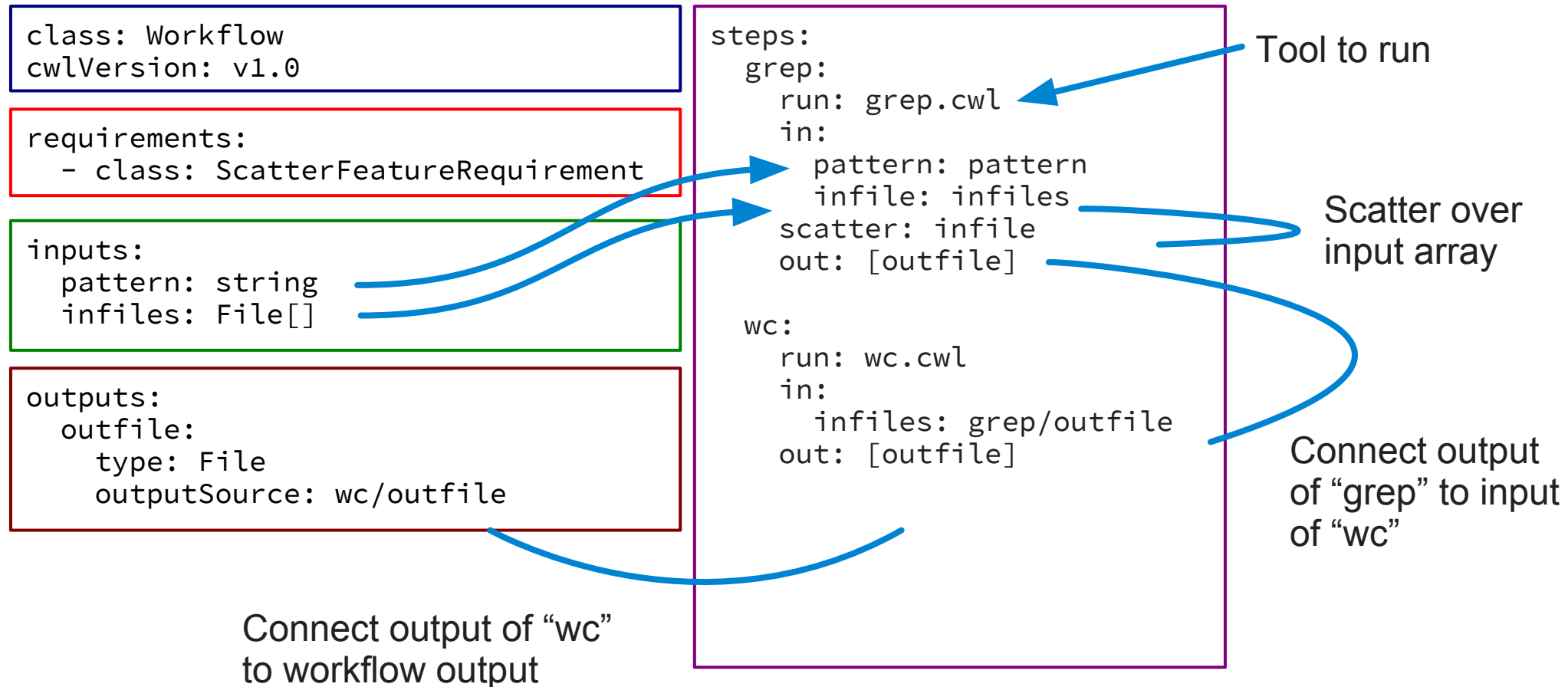
```
steps:
  grep:
    run: grep.cwl
    in:
      pattern: pattern
      infile: infiles
    scatter: infile
    out: [outfile]

  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>

# Example: grep & count





# Thanks!

<http://www.commonwl.org>

BACKUP SLIDES!

# OTHER EARLY ADOPTERS INCLUDE

(US) [National Cancer Institute Cloud Pilots](#) (Seven Bridges Genomics, Institute for Systems Biology)

**Cincinnati Children's Hospital Medical Research Center**  
(Andrey Kartashov & Artem Barski)

**bcbio**: Validated, scalable, community developed variant calling, RNA-seq and small RNA analysis ([docs](#), BOSC 2016 talk: [video](#), [slides](#)) (Brad Chapman et al.)

Duke University, Center for Genomic and Computational Biology: **GENOMICS OF GENE REGULATION** project (BOSC 2016 talk: [video](#), [slides](#), [poster](#)) (Dan Leehr et al.)

NCI **DREAM SMC-RNA Challenge** (Kyle Ellrott et al.)

[Presentation](#) ; [GA4GH Workflow Execution DREAM](#)

# 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.

# CWL BACKENDS AS OF 2017/06/21

	Airflow	Arvados	Toil	Rabix Executor	Ref. Impl.
				local & TES only	cwltool -- local only
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
			<input checked="" type="checkbox"/>		
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
			<input checked="" type="checkbox"/>		
	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		
			<input checked="" type="checkbox"/>		
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Capabilities are self reported and CWL support can vary depending on configuration.

# CWL V1.0 RELEASED JUNE 2016

<http://www.commonwl.org/v1.0/>

## Authors:

Peter Amstutz, Arvados Project, Curoverse  
Michael R. Crusoe, Common Workflow Language project  
Nebojša Tijanić, Seven Bridges Genomics

## Contributors:

Brad Chapman, Harvard Chan School of Public Health  
John Chilton, Galaxy Project, Pennsylvania State University  
Michael Heuer, UC Berkeley AMPLab  
Andrey Kartashov, Cincinnati Children's Hospital  
Dan Leehr, Duke University  
Hervé Ménager, Institut Pasteur  
Maya Nedeljkovich, Seven Bridges Genomics  
Matt Scales, Institute of Cancer Research, London  
Stian Soiland-Reyes, University of Manchester  
Luka Stojanovic, Seven Bridges Genomics

# HOW DID WE DO IT?

Initial group started at [BOSC Codefest 2014](#)

Moved to open mailing list and extended onto GitHub & then Gitter chat

Frequent (twice a month or more) video chats to work through design issues with summaries emailed

Some participants doing CWL community work during their day jobs, some on “nights & weekends”.

In October 2015 Seven Bridges sponsored one of the co-founders (M. Crusoe) to work full time on the project

# A GRAND OPPORTUNITY

if:

properly funded and embraced by the wider community

then:

the researchobject.org standards + CWL could fulfill  
the huge need for an executable and complete  
description of how computationally derived research  
results were made

