https://goo.gl/0FsTRE

COMMON WORKFLOW LANGUAGE

The Common Workflow Language v1.0: ready for production, ready for the future

Michael R. Crusoe CWL Community Engineer
2016-10-25 @biocrusoe / #CommonWL
NETTAB 2016. Rome, Italy

https://goo.gl/0FsTRE

Michael R. Crusoe, who is this guy?

Phoenix, Arizona (Sonoran Desert), USA

Studied at Arizona State University: Computer Science; time in industry as a developer & system administrator (Google, others); returned to academia to study Microbiology.

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

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

https://goo.gl/0FsTRE

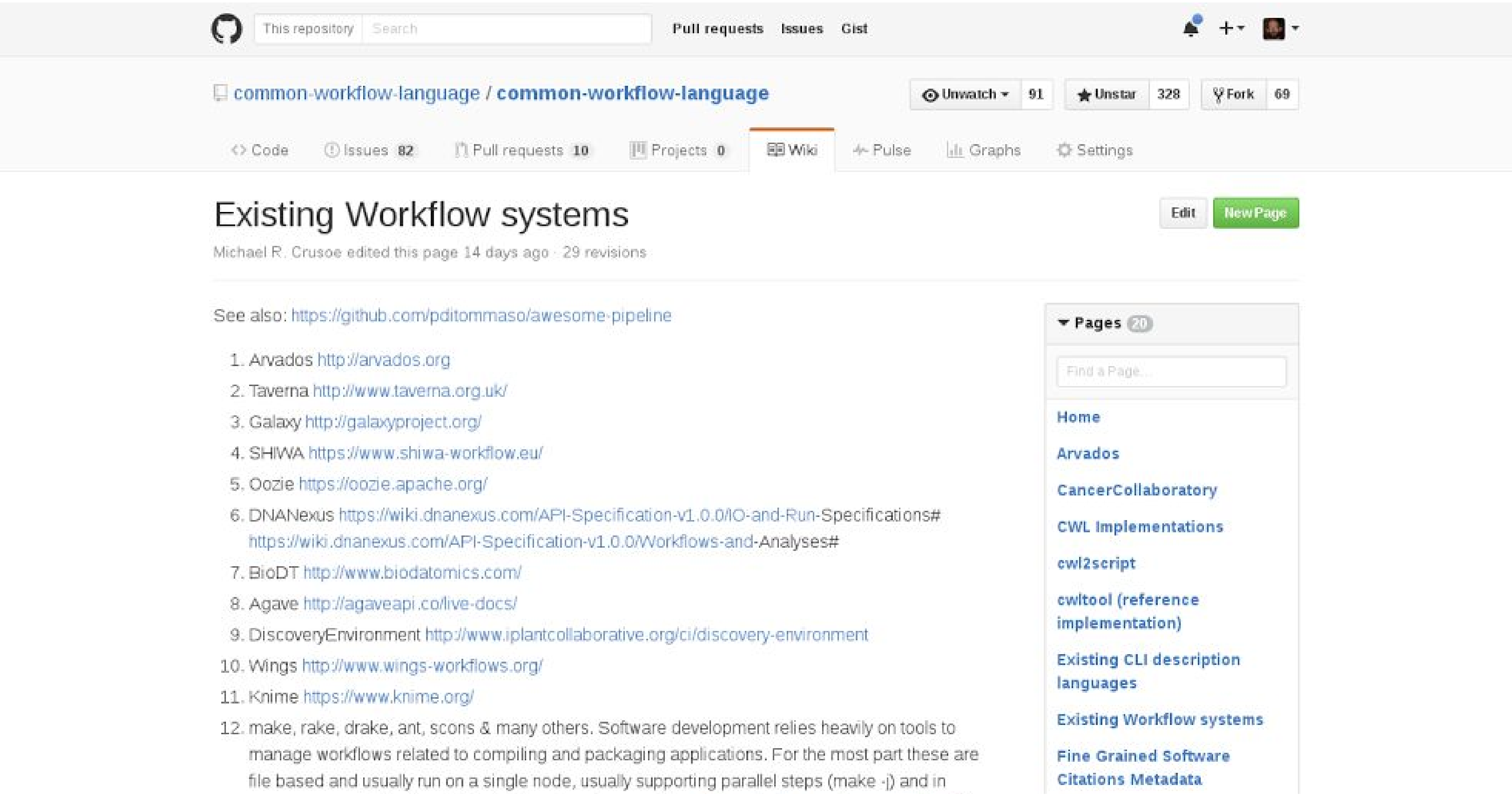
Why use a workflow management system?

Features **can** include:

- 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)

https://goo.gl/0FsTRE

Existing computational research workflow systems



Existing Workflow systems

Michael R. Crusoe edited this page 14 days ago · 29 revisions

See also: https://github.com/pditommaso/awesome-pipeline

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/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

Pages 20

Find a Page...

- Home
- Arvados
- CancerCollaboratory
- CWL Implementations
- cwl2script
- cwltool (reference implementation)
- Existing CLI description languages
- Existing Workflow systems
- Fine Grained Software Citations Metadata

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

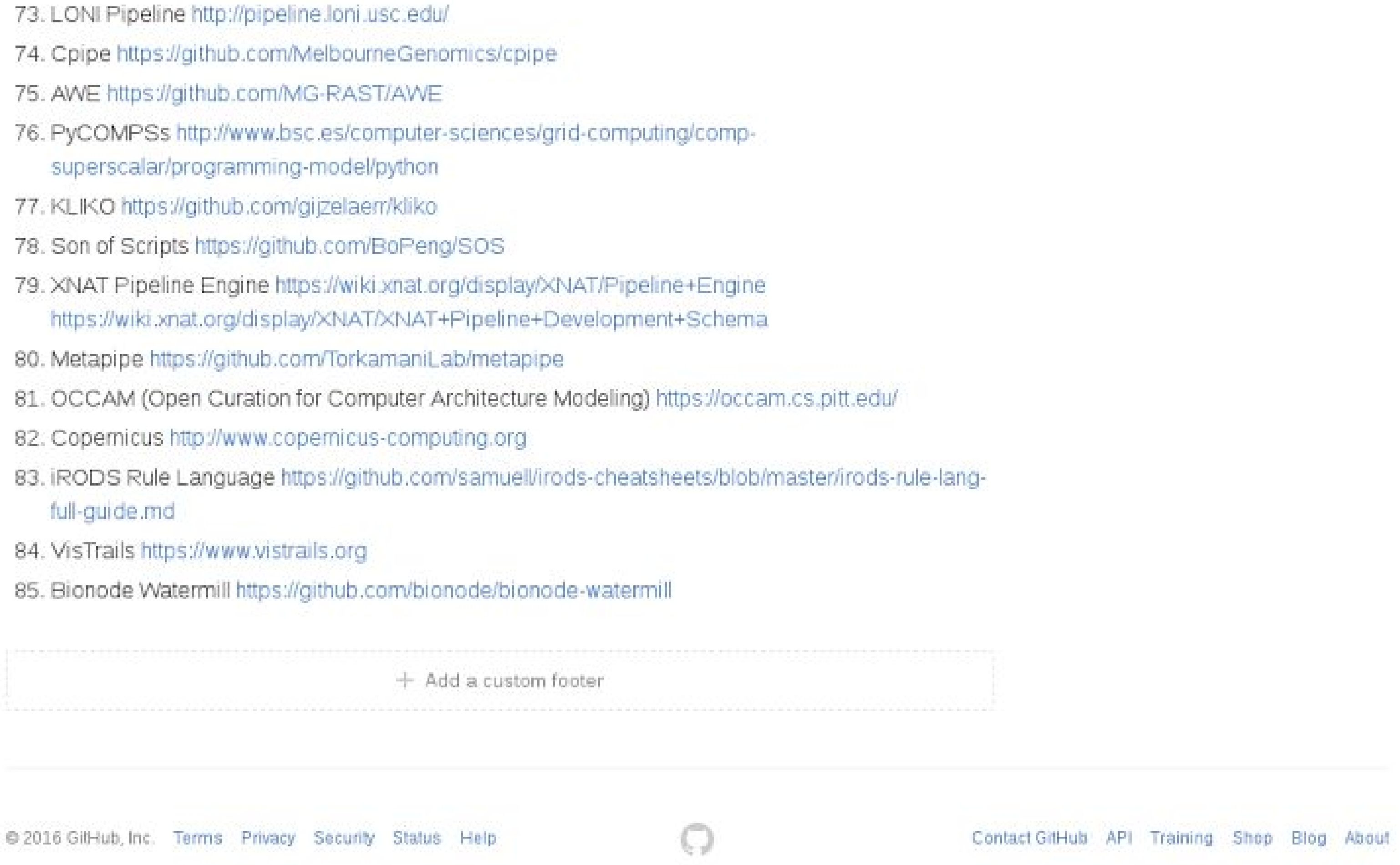
13. Snakemake https://bitbucket.org/johanneskoester/snakemake
14. BPipe http://bpipe.org
15. Ruffus https://code.google.com/p/ruffus/
16. NextFlow http://nextflow.io
17. Luigi http://github.com/spotify/luigi
18. SciLuigi. Helper library built on top of Luigi to ease development of Scientific workflows 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/DSLv2
33. nipype http://nipy.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
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

+ Add a custom sidebar

Clone this wiki locally

`https://github.com/common`

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
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/
62. NGLess https://github.com/luispedro/ngless
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://mircwiki.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. PyCOMPSs http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/programming-model/python
77. KLIKO https://github.com/gijzelaerr/kliko
78. Son of Scripts https://github.com/BoPeng/SOS
79. XNAT Pipeline Engine https://wiki.xnat.org/display/XNAT/Pipeline+Engine https://wiki.xnat.org/display/XNAT/XNAT+Pipeline+Development+Schema
80. Metapipe https://github.com/TorkamaniLab/metapipe
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

+ Add a custom footer
© 2016 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About

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 format for bioinformatics (and more!) 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

https://goo.gl/0FsTRE

Participating Organizations & Projects

CUROVERSE

Galaxy

SevenBridges genomics

Workflow 4Ever

Taverna

Institut Pasteur

Your logo here?

HARVARD T.H. CHAN SCHOOL OF PUBLIC HEALTH

Cincinnati Children's

BROA INSTIT

OHSU

UCSC

The Agave Platform

CYVERSE

TACC

netherlands eScience center

wellcome trust sanger institute

https://goo.gl/0FsTRE

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.

https://goo.gl/0FsTRE

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

https://goo.gl/0FsTRE

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

Early Adopters

(US) **National Cancer Institute Cloud Pilots** (Seven Bridges Genomics, Institute for Systems Biology)

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

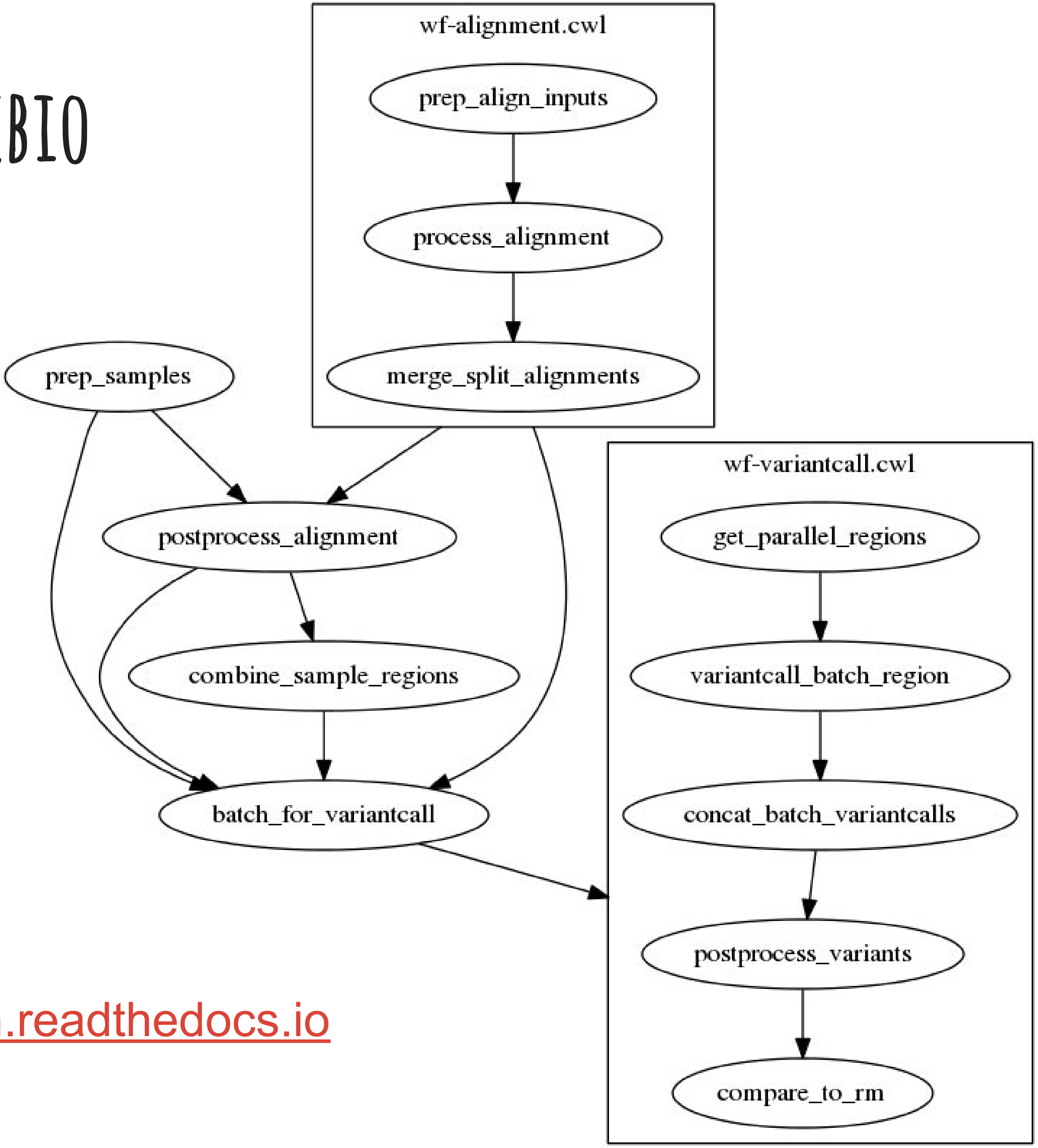
bcbio: Validated, scalable, community developed variant calling, RNA-seq and small RNA analysis (docs, BOSC 2016 talk: video, slides) (Brad Chapman et al.)

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

Early Adopter: bcbio

runs locally, on Microsoft Azure (via Curoverse), or on any other CWL compatible platform



https://bcbio-nextgen.readthedocs.io

https://goo.gl/0FsTRE

Announcing: v1.0!

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

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

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)
- 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.

https://goo.gl/0FsTRE

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

```
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

Adapted from Peter Amstutz’s presentation, licensed CC-BY-SA

https://goo.gl/0FsTRE

Challenges

Giving a standard to a community that is “free as in puppies”: How does the community participate? How will maintenance be funded?

CWL isn’t the only effort that has these needs; can we join with related efforts?

https://goo.gl/0FsTRE

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 computationaly derived research results were made

researchobject.org

COMMON
WORKFLOW
LANGUAGE

What's next for the Common Workflow Language?

Public charity to own the standard

Tooling improvements

Release of {argparse,click}2cwl by GSoC mentee Anton Khodak (based on code from the Galaxy Project & Eric Rasche)

More mature implementations (Galaxy, Taverna, …?)

Integration with researchobject.org standards for attribution, provenance, and metadata guidance.

CWL Even in London Nov 1-4

See https://goo.gl/aVmpCF

Very open to new participants!

Thanks!

http://commonwl.org