https://tinyurl.com/ADASS2019-CWL

COMMON
WORKFLOW
LANGUAGE

Saving time, money, and stress with standards-based workflows

Michael R. Crusoe: CWL Project Lead
#ADASS2019
2019-10-07

@biocrusoe
@commonwl
#CommonWL

http://bit.ly/rmghc18_CWL

#ADASS2019 Loves Pipelines

But...

Which pipeline framework to use?

Which workflow platform should you learn?

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/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

Permalink: https://s.apache.org/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.htmlhttp://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-projecthttps://wiki.openstack.org/wiki/Mistral#What_is_Mistral.3Fhttps://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/

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/cuneiformhttp://www.cuneiform-lang.org/
44. Anvaya http://www.ncbi.nlm.nih.gov/pubmed/22809419http://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.2004.02.002http://www.yawlfoundation.org/
56. Triquetrum https://projects.eclipse.org/projects/technology.triquetrumhttps://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/

Existing computational research workflow systems

62. NGLess: NGS with less work http://ngless.rtfd.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://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. (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+Enginehttps://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

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
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/ 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. COMBUSTI/O https://github.com/jarlebass/combustio http://hdl.handle.net/10037/9361
102. BioCloud https://github.com/ccwang002/biocloud-server-kaihttp://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/ http://edge.readthedocs.io/ https://lanl-bioinformatics.github.io/EDGE/

Existing computational research workflow systems

130. Digdag https://www.digdag.io/
131. Agua / Automated Genomics Utilities Agent http://aguadev.org
132. BioDepot Workflow Builder (BwB) https://github.com/BioDepot/BioDepot-workflow-builderhttps://doi.org/10.1101/099010
133. 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
134. Butler https://github.com/llevar/butler
135. adage / yadage https://github.com/diana-hep/adage https://github.com/diana-hep/yadage
136. HI-WAY: Execution of Scientific Workflows on Hadoop YARN https://github.com/marcbux/Hi-WAYhttps://openproceedings.org/2017/conf/edbt/paper-248.pdf
137. OpenMOLE https://github.com/openmole/openmole https://www.openmole.org/https://doi.org/10.3389/fninf.2017.00021
138. Biopet https://github.com/biopet/biopet
139. Nephele https://nephele.niaid.nih.gov/
140. TOPPAS http://doi.org/10.1021/pr300187f
141. SBpipe https://pdp10.github.io/sbpipe/ https://github.com/pdp10/sbpipehttps://doi.org/10.1186/s12918-017-0423-3
142. Dray http://dray.it/
143. GenomeVIP https://github.com/ding-lab/GenomeVIP https://doi.org/10.1101/gr.211656.116
144. GridSAM https://sourceforge.net/projects/gridsam/
145. Roddy https://github.com/eilslabs/Roddy
146. SciFlo (historical; doesn't seem to be maintained anymore) https://web.archive.org/web/20161118011409/https://sciflo.jpl.nasa.gov/SciFloWiki/FrontPage
147. GNU Guix Workflow Language https://git.roelj.com/guix/gwl.git#gnu-guix-workflow-language-extension https://github.com/UMCUGenetics/guix-workflows/blob/master/umcu/workflows/rnaseq.scm
148. Porcupine https://timvanmourik.github.io/Porcupine/

Existing computational research workflow systems

130. Ophidia http://ophidia.cmcc.it/
131. WebLicht https://weblicht.sfs.uni-tuebingen.de/
132. GATE Cloud https://cloud.gate.ac.uk/
133. SCIPION http://scipion.cnb.csic.es/m/home/ https://github.com/I2PC/scipion/wiki/Creating-a-Protocol
134. Ergatis http://ergatis.sourceforge.net/
135. TIGR "Workflow" https://sourceforge.net/projects/tigr-workflow/ http://tigr-workflow.sourceforge.net/
136. Archivematica https://wiki.archivematica.org/Main_Page (A preservation workflow system that implements the ISO-OAIS standard using gearman/MCP)
137. Martian http://martian-lang.org/about/
138. BioMAJ http://genouest.github.io/biomaj/
139. Conveyor http://conveyor.cebitec.uni-bielefeld.de (retired). https://academic.oup.com/bioinformatics/article/27/7/903/230562/Conveyor-a-workflow-engine-for-bioinformatic
140. Biopipe http://www.biopipe.org (appears to be defunct) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC403782/
141. Wildfire http://wildfire.bii.a-star.edu.sg/ https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-6-69
142. BioWBI http://bioinformatics.hsanmartino.it/bits_library/library/00079.pdf
143. BioWMS http://bioinformatics.hsanmartino.it/bits_library/library/00568.pdf
144. BioMoby http://biomoby.open-bio.org/ https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-523
145. SIBIOS http://ieeexplore.ieee.org/document/1309094/
146. NGSANE https://github.com/BauerLab/ngsane https://academic.oup.com/bioinformatics/article/30/10/1471/266879/NGSANE-a-lightweight-production-informatics
147. Pwrake https://github.com/misshie/Workflows https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3180464/
148. Nesoni https://github.com/Victorian-Bioinformatics-Consortium/nesoni
149. Skam http://skam.sourceforge.net/skam-intro.html
150. TREVA http://bioinformatics.petermac.org/treva/ http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0095217

Existing computational research workflow systems

151. EGene https://www.semanticscholar.org/paper/EGene-a-configurable-pipeline-generation-system-fo-Durham-Kashiwabara/4c0656195b5efcdd3aa7bdcb55fc95a957c150aa https://academic.oup.com/bioinformatics/article/30/18/2659/2475637/EuGene-PP-a-next-generation-automated-annotation
152. WEP https://bioinformatics.cineca.it/wep/ https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-S7-S11
153. Microbase http://www.microbasecloud.com/
154. e-Science Central http://www.esciencecentral.co.uk/ https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3538293/
155. Cyrille2 https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-96
156. PaPy https://code.google.com/archive/p/papy/ https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3051902/
157. JobCenter https://github.com/yeastrc/jobcenter https://scfbm.biomedcentral.com/articles/10.1186/1751-0473-7-8
158. CoreFlow https://www.ncbi.nlm.nih.gov/pubmed/24503186
159. dynamic-pipeline https://code.google.com/archive/p/dynamic-pipeline/
160. XiP http://xip.hgc.jp/wiki/en/Main_Page https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3530915/
161. Eoulsan http://www.outils.genomique.biologie.ens.fr/eoulsan/ https://www.ncbi.nlm.nih.gov/pubmed/22492314
162. CloudDOE http://clouddoe.iis.sinica.edu.tw/
163. BioPig https://github.com/JGI-Bioinformatics/biopig https://www.ncbi.nlm.nih.gov/pubmed/24021384
164. SeqPig https://github.com/HadoopGenomics/SeqPig https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3866557/
165. zymake http://www-personal.umich.edu/~ebreck/code/zymake/
166. JMS https://github.com/RUBi-ZA/JMS http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0134273
167. CLC Genomics Workbench https://www.qiagenbioinformatics.com/products/clc-genomics-workbench/
168. NG6 http://ng6.toulouse.inra.fr/ https://www.ncbi.nlm.nih.gov/pubmed/22958229
169. VIBE http://www.incogen.com/vibe/
170. WDL (Workflow Description Language) https://github.com/broadinstitute/wdl

Existing computational research workflow systems

171. SciFlow https://github.com/kaizhang/SciFlow (not to be confused with SciFloware and SciFlo).
172. Bioshake https://github.com/PapenfussLab/bioshake
173. SciPipe http://scipipe.org
174. Kapacitor / TICKscripts https://docs.influxdata.com/kapacitor/v1.3//tick/
175. AiiDA: Automated Interactive Infrastructure and Database for Computational Science http://www.aiida.net/
176. Reflow: a language and runtime for distributed, integrated data processing in the cloud https://github.com/grailbio/reflow
177. Resolwe: an open source dataflow package for Django framework https://github.com/genialis/resolwe
178. Yahoo! Pipes (historical) https://en.wikipedia.org/wiki/Yahoo!_Pipes
179. Walrus https://github.com/fjukstad/walrus
180. Apache Beam https://beam.apache.org/
181. CLOSHA https://closha.kobic.re.kr/
https://www.bioexpress.re.kr/go_tutorialhttp://docplayer.net/19700397-Closha-manual-ver1-1-kobic-korean-bioinformation-center-kogun82-kribb-re-kr-2016-05-08-bioinformatics-workflow-management-system-in-bio-express.html
182. WopMars https://github.com/aitgon/wopmars http://wopmars.readthedocs.io/
183. flowing-clj https://github.com/stain/flowing-clj
184. Plumbing and Graph https://github.com/plumatic/plumbing
185. LabView http://www.ni.com/en-us/shop/labview.html
186. MyOpenLab http://myopenlab.org/
187. Max/MSP https://cycling74.com/products/max/
188. NoFlo https://noflojs.org/
189. Flowstone http://www.dsprobotics.com/flowstone.html
190. HyperLoom https://code.it4i.cz/ADAS/loom https://code.it4i.cz/ADAS/loom
191. Dask http://dask.pydata.org/en/latest/ https://github.com/dask/dask

Existing computational research workflow systems

193. Stimela https://github.com/SpheMakh/Stimela https://github.com/SpheMakh/Stimela/wiki https://www.acru.ukzn.ac.za/~cosmosafari2017/wp-content/uploads/2017/02/makhathini.pdf
194. JTracker http://www.jthub.co/ https://github.com/icgc-dcc/jtracker
195. PipelineDog http://pipeline.dog/ https://github.com/zhouanbo/pipelinedog https://doi.org/10.1093/bioinformatics/btx759
196. DALiuGE https://arxiv.org/abs/1702.07617 https://github.com/ICRAR/daliuge https://daliuge.readthedocs.io/
197. Overseer https://github.com/framed-data/overseer
198. Squonk https://squonk.it/
199. GC3Pie https://github.com/uzh/gc3pie
200. Fractalide https://github.com/fractalide/fractalide
201. TOGGLe https://doi.org/10.1101/245480 http://toggle.southgreen.fr/
202. Askalon http://www.askalon.org
203. Eclipse ICE (The Integrated Computational Environment) https://www.eclipse.org/ice
204. Sandia Analysis Workbench (SAW) http://www.sandia.gov/saw/
205. dispel4py https://github.com/dispel4py/dispel4py
206. Jobber https://pypi.python.org/pypi/Jobber/0.1.4
207. NeatSeq-Flow http://neatseq-flow.readthedocs.io/
208. S4M https://bitbucket.org/uqokorn/s4m_base/wiki/Home
209. Loom http://med.stanford.edu/gbsc/loom.html https://github.com/StanfordBioinformatics/loom http://loom.readthedocs.io/en/latest/templates.html
210. Watchdog https://doi.org/10.1186/s12859-018-2107-4 https://github.com/klugem/watchdog
211. phpflo https://github.com/phpflo/phpflo
212. BASTet: Berkeley Analysis and Storage Toolkit https://openmsi.nersc.gov/openmsi/client/bastet.html https://biorack.github.io/BASTet/ https://doi.org/10.1109/TVCG.2017.2744479
213. Tavaxy: Pattern based workflow system for the bioinformatics domain http://www.tavaxy.org/

Existing computational research workflow systems

214. Ginflow: Decentralised adaptive workflow engine https://ginflow.inria.fr/
215. SciApps: A cloud-based platform for reproducible bioinformatics workflows https://doi.org/10.1093/bioinformatics/bty439 https://www.sciapps.org/
216. Stoa: Script Tracking for Observational Astronomy https://github.com/petehague/Stoa
217. Collective Knowledge (CK) framework http://cknowledge.org/
218. QosCosGrid (QCG) http://www.qoscosgrid.org/ http://www.qoscosgrid.org/trac/qcg-broker/wiki/qcg-advanced-client%20
219. High-Throughput Binding Affinity Calculator (HTBAC) https://htbac.readthedocs.io/en/latest/ https://github.com/radical-cybertools/htbac https://arxiv.org/abs/1801.01174
220. BioWorkbench (Swift-based) https://arxiv.org/abs/1801.03915 https://github.com/mmondelli/bioworkbench
221. ENVI Task Engine https://gbdxdocs.digitalglobe.com/docs/envi-task-engine https://www.harrisgeospatial.com/Learn/Whitepapers/TabId/2359/ArtMID/10212/ArticleID/17299/Workflow-Tools-in-ENVI.aspx https://envi-py-engine.readthedocs.io/en/latest/index.html
222. Pypeline https://github.com/cgarciae/pypeln
223. mpipe http://vmlaker.github.io/mpipe/
224. idseq-dag https://github.com/chanzuckerberg/idseq-dag
225. Piper (based upon GATK Queue) https://github.com/NationalGenomicsInfrastructure/piper
226. Apache Object Oriented Data Technology (OODT) http://oodt.apache.org/
227. JX Workflow (DSL for Makeflow) https://ccl.cse.nd.edu/software/manuals/jx-quick.html http://ccl.cse.nd.edu/research/papers/jx-escience-2018.pdf
228. The Adaptable IO System (ADIOS), ADIOS using applications can be the orchestrated into a workflow http://csmd.ornl.gov/adios
229. GenPipes https://doi.org/10.1101/459552 https://bitbucket.org/mugqic/genpipes/src/master/
230. Argo https://argoproj.github.io/ https://github.com/argoproj/argo https://github.com/argoproj/argo/blob/master/examples/README.md
231. Reana https://reana.readthedocs.io/en/latest/ https://github.com/reanahub/reana
232. Cuisine Framework https://www.astron.nl/~renting/cuisine.html

Existing computational research workflow systems

233. Niassa https://github.com/oicr-gsi/niassa https://oicr-gsi.github.io/niassa-docs/
234. pypeFLOW https://github.com/PacificBiosciences/pypeFLOW
235. Tiny Cloud Engine http://ka.cb.k.u-tokyo.ac.jp/tce/
236. Xbowflow https://github.com/ChrisSuess/Project-Xbow/tree/master/xbowflow
237. AdaptiveMd https://github.com/markovmodel/adaptivemd
238. Meshroom https://github.com/alicevision/meshroom
239. LSST Data Management https://github.com/lsst/pipe_base
240. CGAT-core https://github.com/cgat-developers/cgat-core
241. Prefect https://docs.prefect.io/
242. Apache SCXML engine https://commons.apache.org/proper/commons-scxml/guide/core-engine.html https://commons.apache.org/proper/commons-scxml/guide/scxml-documents.html
243. IceProd https://github.com/WIPACrepo/iceprod
244. AnADAMA2 http://huttenhower.sph.harvard.edu/anadama2
245. Luna https://luna-lang.org/
246. Passerelle https://code.google.com/archive/a/eclipselabs.org/p/passerelle
247. Kurator-Akka https://github.com/kurator-org/kurator-akka
248. Jug https://doi.org/10.5334/jors.161
249. Node-RED https://nodered.org/
250. Databolt Flow https://github.com/d6t/d6tflow
251. Frictionless Data Package Pipelines https://github.com/frictionlessdata/datapackage-pipelines
252. DataFlows https://github.com/datahq/dataflows
253. Volcano https://github.com/volcano-sh/volcano

Permalink: https://s.apache.org/existing-workflow-systems

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.

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

What are the technical requirements of CWL?

Tools in CWL workflows can be written in any computer language: Python, C++, Fortran, Java, …

Tool authors do not need to use a particular library or API, just need a command line interface which they probably already have.

Docker format software containers are recommended, but not required. Workflow runners can use any Docker compatible container engine: Singularity, udocker, rkt, …

https://tinyurl.com/ADASS2019-CWL

From the Life Sciences...

CUROVERSE

Galaxy PROJECT

SevenBridges genomics

Workflow 4Ever

Taverna

Institut Pasteur

elixir

HARVARD T.H. CHAN SCHOOL OF PUBLIC HEALTH

Cincinnati Children's

BROA INSTIT

UCSC

The Agave Platform

CYVERSE

TACC

OHSU

wellcome trust sanger institute

bioexcel

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

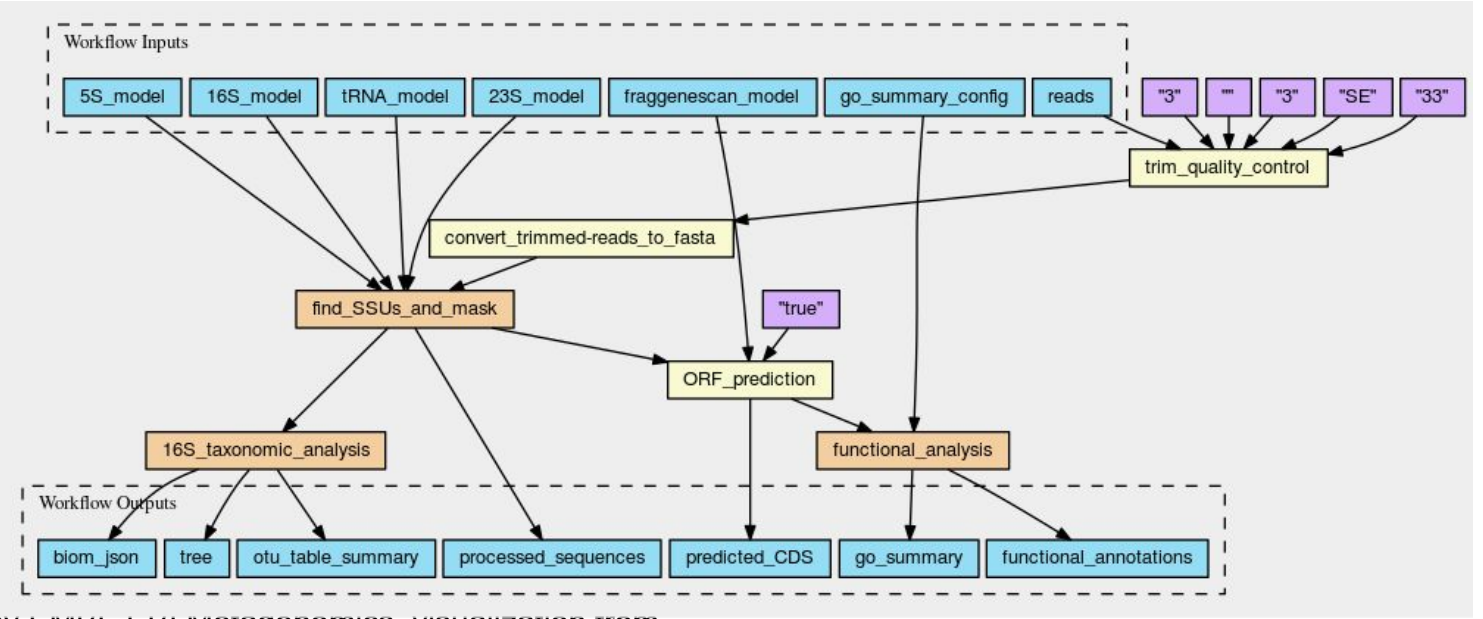
2560 lines of CWL descriptions

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

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

https://tinyurl.com/ADASS2019-CWL

EBI's metagenomics Pipeline as CWL



Courtesy EMBL-EBI Metagenomics, visualization from
https://view.commonwl.org/workflows/github.com/ProteinsWebTeam/ebi-metagenomics-cwl/blob/master/workflows/rna-selector.cwl
More about this story: https://doi.org/10.1038/d41586-019-02619-z

https://tinyurl.com/ADASS2019-CWL

...to (astro)physics and beyond

netherlands eScience center

ASTRON

LOFAR

dianahep

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

How CWL is used

Some write CWL manually

Some write/edit CWL using a graphical interface

Automated conversion from non-standardized workflows.

Programmatic CWL generation from other languages or libraries.

Some use CWL to document and make portable an existing workflow even if their current workflow engine/platform does not support CWL.

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

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

CWL enables well described tools and workflows

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

Data locality with CWL

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

Platforms that understand CWL can use these identifiers to send compute to 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)

https://goo.gl/WDcAx7

CWL enables Money & Time savings

https://renci.org/technical-reports/tr-19-01/

"[with CWL we are] able to scale the workflows across geo-distributed regions across clouds and place the jobs close to the input datasets for improved performance in network I/O and reduced cost for egress network traffic."

https://renci.org/technical-reports/tr-19-02/

"running the CWL workflows [using] "cost-aware" [scheduling] saves up to $2,092 (92.2%) in total for 30 workflow runs"

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**, **(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

Open Source Implementations

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

Arvados from Curoverse / Veritas Genetics

CWLEXEC from IBM LSF

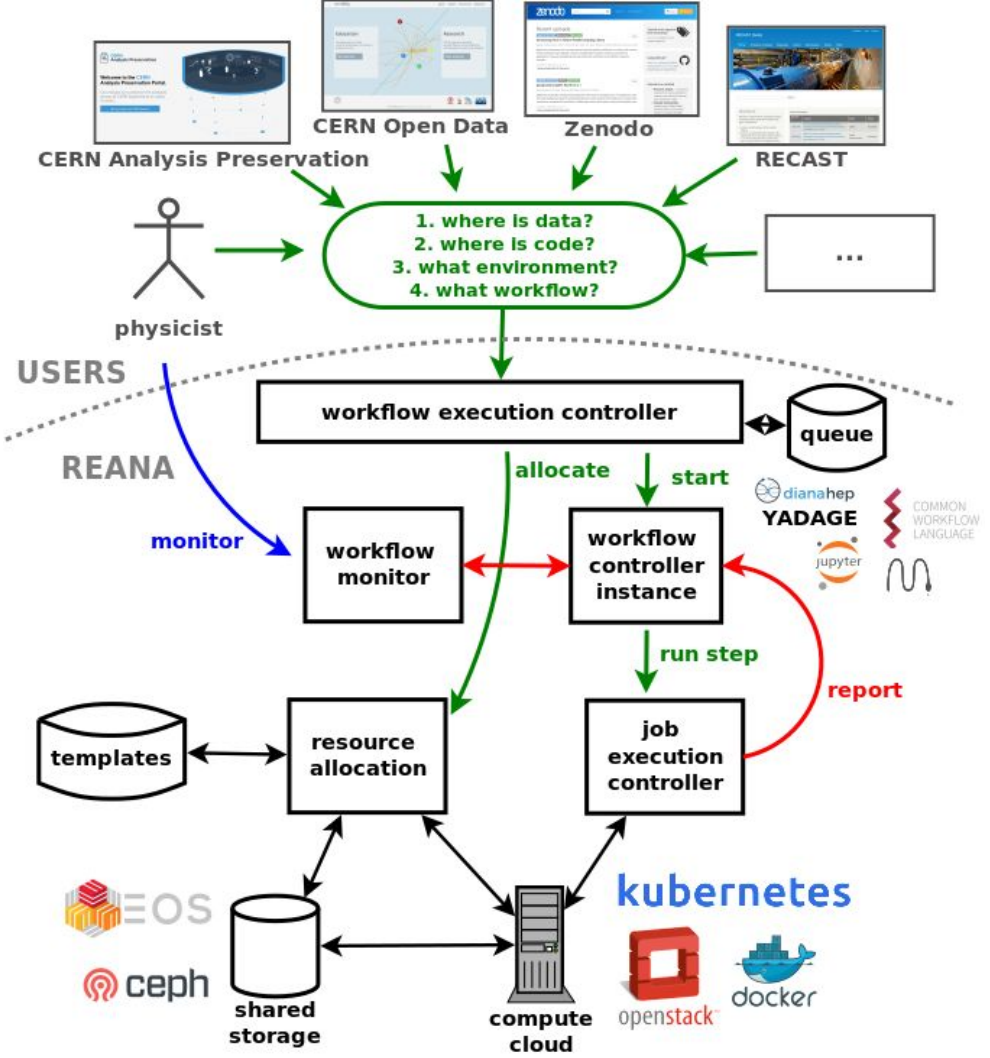
CWL-Airflow from BioWardrobe Team, CCHMC

Toil from UCSC & community contributors

Rabix Bunny from Seven Bridges

REANA from CERN

https://tinyurl.com/ADASS2019-CWL



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

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

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.

What about Astronomy?

EOSCPilot LOFAR Science Demonstrator: Showed that CWL was expressive enough for an astronomy pipeline. Led to an extension to CWL v1.0 “InplaceUpdateRequirement” that was incorporated in CWL v1.1.

ASTERICS OBELICS CWL Project: successfully demonstrated CWL running on production data and hardware

https://tinyurl.com/ADASS2019-CWL

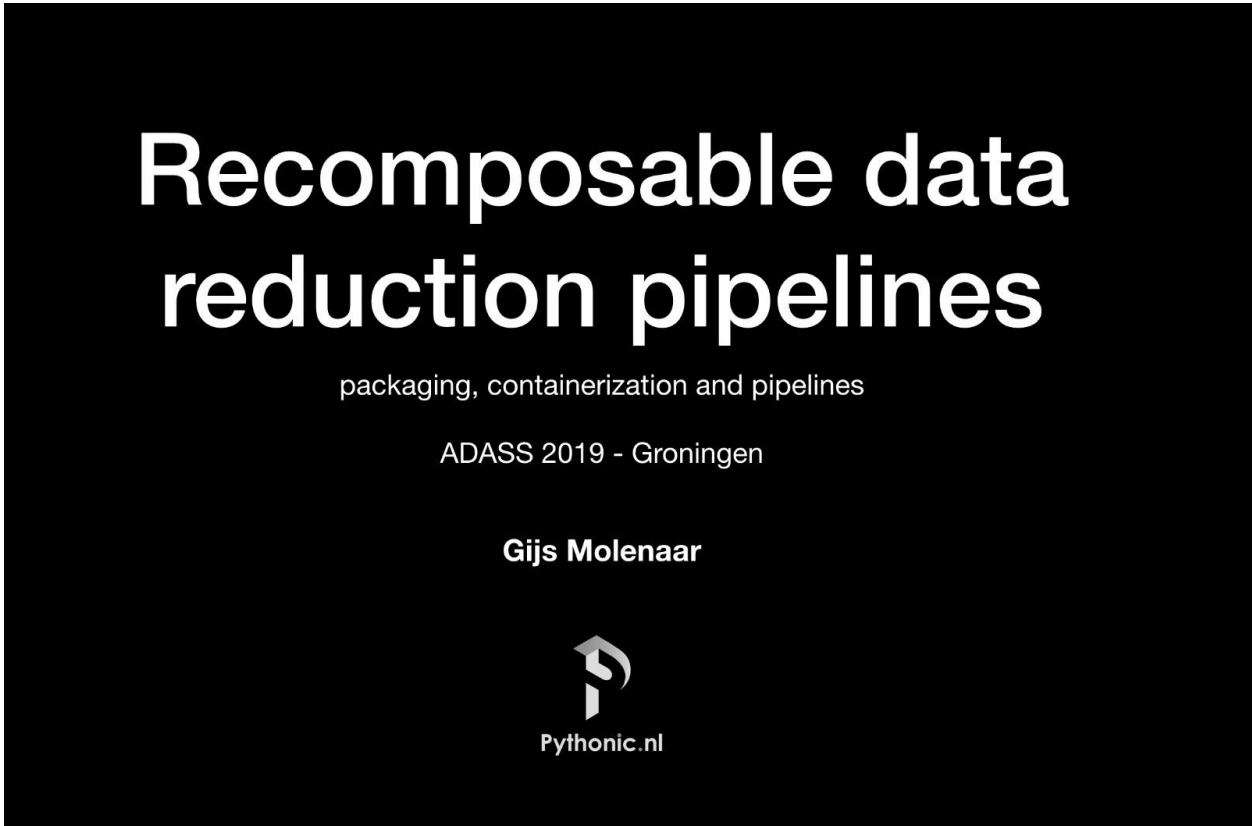
What about Astronomy?

Look for

Gijs Molenaar's

presentation on

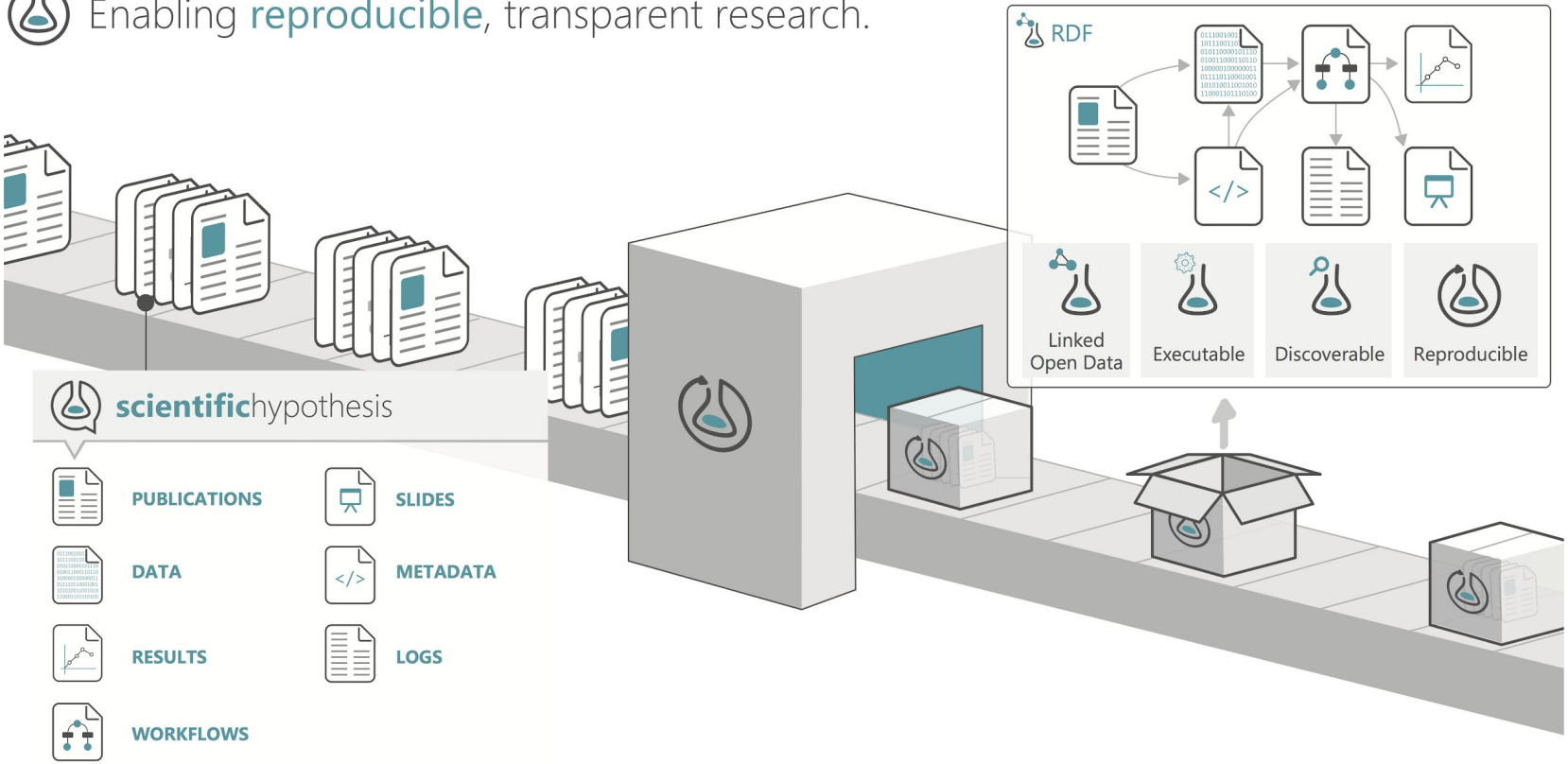
Wednesday, 16:45



https://tinyurl.com/ADASS2019-CWL

ResearchObject.org standard overview

Enabling **reproducible**, transparent research.



CWL + Provenance = CWLProv

Demonstrates the reuse of existing provenance standards combined with a ResearchObject to package the result of a workflow along with the complete story of how it came to be.

Accepted for GigaScience; preprint at https://doi.org/10.5281/zenodo.3336124

See also:

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

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

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

https://tinyurl.com/ADASS2019-CWL

Thanks!

COMMON
WORKFLOW
LANGUAGE

https://www.commonwl.org

Backup slides!

https://tinyurl.com/ADASS2019-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

Assisted EOSC Pilot & ASTERICS; Part time with ELIXIR(-NL); 100% CWL

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

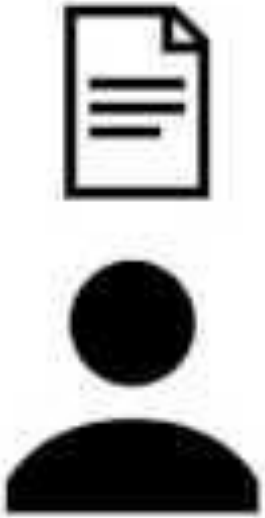
2018: **IBM** released their CWL implementation for LSF.

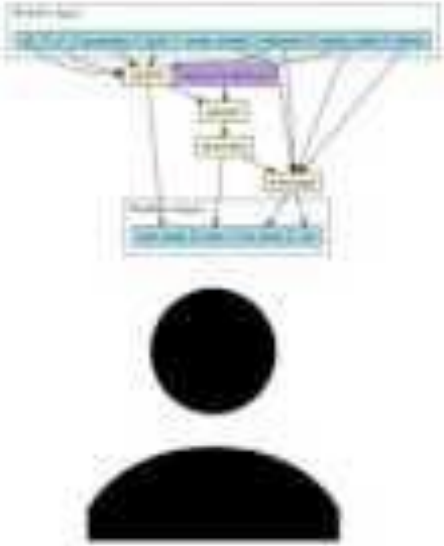
2019: CWL v1.1 released

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





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
```

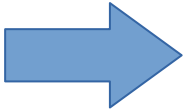
Linked data support

```
$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.

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

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

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

NOTE: The cwl-upgrader works around the above changes; v1.0 to v1.1 upgrades are completely automated and pain free.

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 v1.2 ???

Agreed to release CWL v1.2 in July 2020, or once workflow conditionals are finished

CWL mini-conference 2019-10-17 & 18th

https://github.com/common-workflow-language/common-workflow-language/issues/868

If you are interested in attending but lack the financial resources or face other barriers to your participation then please get in touch directly: mrc@commonwl.org