CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language

Michael Kotliar[1,*], Andrey V. Kartashov[1,*], and Artem Barski[1,2,#]

[1]Division of Allergy and Immunology, [2]Division of Human Genetics, Cincinnati Children's Hospital Medical Center and Department of Pediatrics, College of Medicine, University of Cincinnati, Cincinnati, OH

*Joint first author; #To whom correspondence should be addressed: Artem.Barski@cchmc.org.

Abstract

Massive growth in the amount of research data and computational analysis has led to increased utilization of pipeline managers in biomedical computational research. However, each of more than 100 such managers uses its own way to describe pipelines, leading to difficulty porting workflows to different environments and therefore poor reproducibility of computational studies. For this reason, the Common Workflow Language (CWL) was recently introduced as a specification for platform-independent workflow description, and work began to transition existing pipelines and workflow managers to CWL. Here, we present CWL-Airflow, an extension for the Apache Airflow pipeline manager supporting CWL. CWL-Airflow utilizes CWL v1.0 specification and can be used to run workflows on standalone MacOS/Linux servers, on clusters, or on variety cloud platforms. A sample CWL pipeline for processing of ChIP-Seq data is provided. CWL-Airflow is available under Apache license v.2 and can be downloaded from https://github.com/Barski-lab/cwl-airflow .

Supplementary information: Supplementary data are available online.

1 Introduction

Modern biomedical research has seen a remarkable increase in the production and computational analysis of large datasets, leading to an urgent need to share standardized analytical techniques. However, of the more than one hundred computational workflow systems used in biomedical research, most define their own specifications for computational pipelines (1), (https://github.com/pditommaso/awesome-pipeline). Furthermore, the evolving complexity of computational tools and pipelines makes it nearly impossible to reproduce computationally heavy studies or to repurpose published analytical workflows. Even when the tools are published, the lack of a precise description of the operating system environment and component software versions can lead to inaccurate reproduction of the analyses—or analyses failing altogether when executed in a different environment. In order to ameliorate this situation, a team of researchers and software

developers formed the Common Workflow Language (CWL) working group (http://www.commonwl.org/) with the intent of establishing a specification for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments. In order to achieve this, CWL specification provides a set of formalized rules that can be used to describe each command line tool and its parameters, and optionally a container (*e.g.*, a Docker image) with the tool already installed. CWL workflows are composed of one or more of such command line tools. Thus, CWL provides a description of the working environment and version of each tool, how the tools are "connected" together, and what parameters were used in the pipeline. Researchers using CWL are then able to deposit descriptions of their tools and workflows into a repository (*e.g.*, dockstore.org) upon publication, thus making their analyses reusable by others.

After version 1 of the CWL standard (2) was finalized in 2016, developers began adapting the existing pipeline managers to use CWL. For example, companies such as Seven Bridges Genomics and Curoverse are developing the commercial platforms Rabix (3) and Arvados (https://arvados.org) whereas academic developers (*e.g.*, Galaxy (4), Toil (5) and others) are adding CWL support to their pipeline managers.

Airflow (http://airflow.incubator.apache.org) is a lightweight workflow manager initially developed by AirBnB, which is currently an Apache Incubator project, and is available under a permissive Apache license. Airflow executes each workflow as a Directed Acyclic Graph (DAG) of tasks, in which tasks comprising the workflow are organized in a way that reflects their relationships and dependencies. DAG objects are initiated from Python scripts placed in a designated folder. Airflow has a modular architecture and can distribute tasks to an arbitrary number of workers, possibly across multiple servers, while adhering to the task sequence and dependencies specified in the DAG. Unlike many of the more complicated platforms, Airflow imposes little overhead, is easy to install, and can be used to run task-based workflows in various environments ranging from standalone desktops and servers to Amazon or Google cloud platforms. It also scales horizontally on clusters managed by Apache Mesos (6) and may be configured to send tasks to Celery (http://www.celeryproject.org) task queue. Here we present an extension of Airflow, allowing it to run CWL-based pipelines. This gives us a lightweight workflow management system with full support for CWL, the most perspective scientific workflow description language.

2 Implementation

CWL-Airflow module extends Airflow's functionality with the ability to parse and execute workflows written with the current CWL specification (v1.0 (2)). The Apache Airflow code is extended with a Python package that defines four basic classes—*CWLStepOperator*, *JobDispatcher*, *JobCleanup*, and *CWLDAG*—as well as the *cwl_dag.py* script. The *cwl_dag.py* script is used by the Airflow scheduler to sequentially generate DAGs for newly added runs on the basis of input parameters and corresponding workflow descriptor files (Fig. 1).

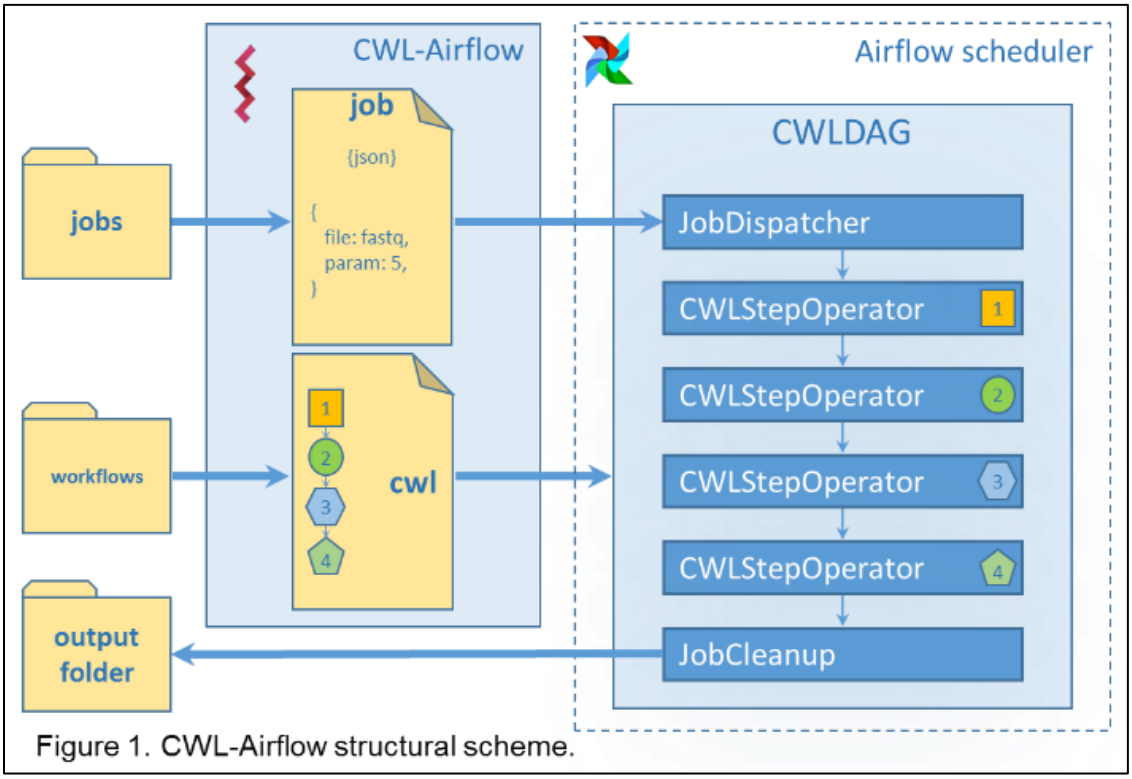


Figure 1. CWL-Airflow structural scheme.

To run a CWL workflow, a CWL descriptor file and an input parameters job file are placed in the *workflows* folder and in the *jobs*/*new* folder (Fig. S1), respectively. Paths to *workflows* and *jobs* folders are set in the airflow configuration. The *dags*_folder and *output*_folder point to the folder containing the *cwl_dag.py* Python script and the output folder for workflow results, respectively. The Airflow scheduler periodically runs *cwl_dag.py*, which searches for the job files in the *jobs/new* folder. If the resources are available, the new job file is moved to the *jobs*/*running* folder, and the corresponding workflow descriptor file is identified in the *workflows* folder by its matching filename. The CWLDAG is then created from the original CWL workflow descriptor as a collection of *CWLStepOperator* tasks, which are organized in a graph that reflects the workflow steps and their relationships and dependencies. Additionally, *JobDispatcher* and *JobCleanup* tasks

are added to the CWLDAG. *JobDisptacher* is used to serialize the input parameters job file and provide the CWLDAG with input data; *JobCleanup* returns the calculated results to the output folder. The generated CWLDAG is used by Airflow to run a workflow with a structure identical to the original CWL descriptor file.

3 Results

As an example, we used a workflow for basic analysis of ChIP-Seq data (Fig. S2). This workflow is a CWL version of a Python pipeline from BioWardrobe (7). It starts by using BowTie (8) to perform alignment to a reference genome, resulting in an unsorted SAM file. The SAM file is then sorted and indexed with samtools (9) to obtain a BAM file and a BAI index. Next MACS2 (10) is used to call peaks and to estimate fragment size. In the last few steps, the coverage by estimated fragments is calculated from the BAM file and is reported in bigWig format (Fig. S2). The pipeline also reports statistics, such as read quality, peak number and base frequency, and other troubleshooting information using tools such as fastx-toolkit (http://hannonlab.cshl.edu/fastx_toolkit/) and bamtools (https://github.com/pezmaster31/bamtools). The CWL workflow, together with tools and corresponding Docker images, may be found at https://github.com/Barski-lab/ga4gh_challenge/releases/tag/v0.02b.

4 Discussion

CWL-Airflow is one of the first pipeline managers supporting version 1.0 of the CWL standard and provides a robust and user-friendly interface for executing CWL pipelines. Unlike more complicated pipeline managers, the installation of Airflow and the CWL-Airflow extension can be performed with a single *pip install* command each. The comparison of CWL-Airflow and several other popular workflow managers is available in Table S1. Furthermore, as one of the most lightweight pipeline managers, Airflow contributes only a small amount of overhead to the overall execution of a computational pipeline. We believe, however, that this is a small price to pay for the ability to monitor and restart task execution afforded by Airflow and better reproducibility and portability of biomedical analyses as afforded by the use of CWL. In summary, CWL-Airflow will provide users with the ability to execute CWL workflows anywhere Airflow can run—from a laptop to cluster or cloud environment.

Acknowledgements: The authors thank all members of the CWL working group for their support and Shawna Hottinger for editorial assistance. The project was supported in part by Center for Clinical & Translational Research and Training (NIH CTSA grant UL1TR001425) and by NIH NIGMS New Innovator Award to AB (DP2GM119134).

Conflict of Interest: AVK and AB are co-founders of Datirium, LLC. Datirium, LLC provides bioinformatics software support services.

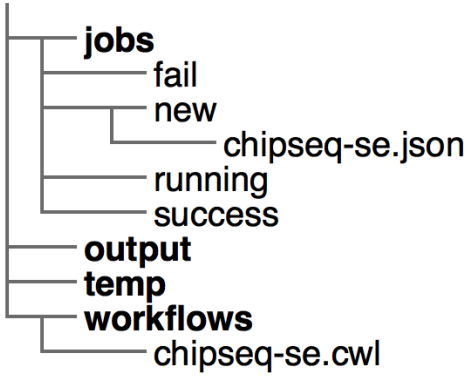
Author contributions statement: AVK and AB conceived the project, AVK and MK wrote the software, MK, AVK and AB wrote and reviewed the manuscript.

5 References

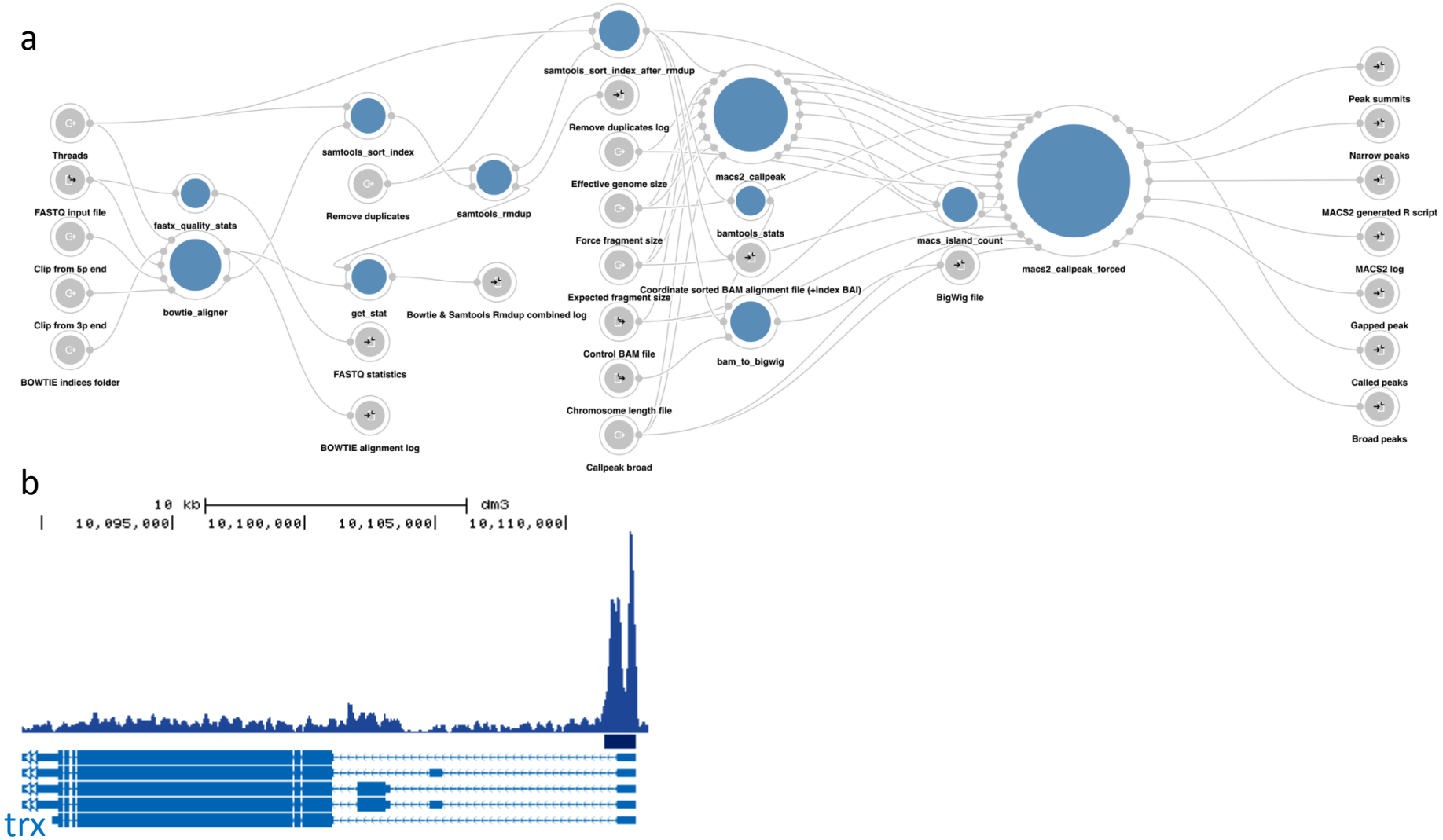
1. Leipzig,J. (2017) A review of bioinformatic pipeline frameworks. *Brief. Bioinform.*, **18**, 530–536.
2. Amstutz,P., Crusoe,M.R., Tijanic,N., Chapman,B., Chilton,J., Heuer,M., Kartashov,A., Leehr,D., Menager,H., Nedeljkovich,M., *et al.* (2016) Common Workflow Language, v1.0. 10.6084/M9.FIGSHARE.3115156.V2.
3. Kaushik,G., Ivkovic,S., Simonovic,J., Tijanic,N., Davis-Dusenbery,B. and Kural,D. (2016) RABIX: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions. *Pac. Symp. Biocomput.*, **22**, 154–165.
4. Giardine,B., Riemer,C., Hardison,R.C., Burhans,R., Elnitski,L., Shah,P., Zhang,Y., Blankenberg,D., Albert,I., Taylor,J., *et al.* (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Res.*, **15**, 1451–5.
5. Vivian,J., Rao,A., Nothaft,F.A., Ketchum,C., Armstrong,J., Novak,A., Pfeil,J., Narkizian,J., Deran,A.D., Musselman-Brown,A., *et al.* (2016) Rapid and efficient analysis of 20,000 RNA-seq samples with Toil. *bioRxiv*, **2**, 62497.
6. Hindman,B., Konwinski,A., Zaharia,M., Ghodsi,A., Joseph,A.D., Katz,R., Shenker,S. and Stoica,I. (2011) Mesos: A platform for fine-grained resource sharing in the data center. *Proc. 8th USENIX Conf. Networked Syst. Des. Implement.*
7. Kartashov,A. V. and Barski,A. (2015) BioWardrobe: an integrated platform for analysis of

epigenomics and transcriptomics data. *Genome Biol.*, **16**, 158.

8. Langmead,B., Trapnell,C., Pop,M. and Salzberg,S.L. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.
9. Li,H., Handsaker,B., Wysoker,A., Fennell,T., Ruan,J., Homer,N., Marth,G., Abecasis,G. and Durbin,R. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
10. Zhang,Y., Liu,T., Meyer,C.A., Eeckhoute,J., Johnson,D.S., Bernstein,B.E., Nusbaum,C., Myers,R.M., Brown,M., Li,W., *et al.* (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol.*, **9**, R137.



Supplementary Figure 1. CWL-Airflow working directory structure.



Supplementary Figure 2. Using CWL-Airflow for analysis of ChIP-Seq data. (a) ChIP-Seq data analysis pipeline visualized by Rabix Composer. (b) Drosophila embryo H3K4me3 ChIP-Seq data (SRR1198790) were processed by our pipeline and CWL-Airflow. UCSC genome browser view of tag density and peaks at trx gene is shown.

Table S1. Features of CWL-Airflow and other popular workflow managers.

Feature	CWL-Airflow	Rabix Bunny	Galaxy	Toil	Arvados	Cromwell
System type	Workflow management system	Workflow runner	Workflow management system	Workflow management system	Workflow management system	Workflow management system
Programming language	Python	Java	Python	Python	Ruby	Java
License type	Apache License 2.0	Apache License 2.0	Academic Free License 3.0	Apache License 2.0	Apache License 2.0	BSD 3-Clause
Supported workflow description specifications	CWL v1.0, python based code	CWL v1.0	own XML-based language, CWL is planned	CWL v1.0, python based code	CWL v1.0, own JSON-based language	WDL, CWL is planned
DB backend	in-memory DB, MySQL, PostgreSQL, Microsoft SQL, any supported by SQLAlchemy	in-memory DB	in-memory DB, MySQL, PostgreSQL	through the job store (local directory or AWS, Azure, GCP location)	PostgreSQL	in-memory DB, MySQL
Installation	pip install	jar file	installation script, multiple steps	pip install	multiple steps, 5 nodes required	jar file
Mesos	Yes	No	Yes	Yes	No	Yes
GCP	Yes	No	Yes	Yes	Yes	Yes
AWS	Yes	No	Yes	Yes	Yes	Yes
Azure	Yes	No	No	Yes	Yes	No
OpenStack	Yes	No	No	Yes	Yes	No
HPC/HTC	Yes	No	Yes	Yes	Yes	Yes
Web interface for task monitoring and management	Yes	No	Yes	No	Yes	No
DAG tree/graph view	Yes	No	Yes	No	Yes	No
Gantt chart for task duration and overlap	Yes	No	No	No	No	No
GUI control of DAG/task	Yes	No	Yes	No	Yes	No

execution flow						
Rest API	Yes	No	Yes	No	Yes	Yes
Workflow execution calendar scheduling	Yes	No	No	No	No	No
Workflow recurring execution	Yes	No	No	No	No	No
Workflow execution timeout	Yes	No	No	Yes	No	No
Automatic task rerun if failed	Yes	No	No	Yes	Yes (in case of a temporary failure)	No
Cross DAGs task dependencies	Yes	No	No	No	No	No
Task parallelization within DAG	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic DAG creation	Yes	No	No	Yes	No	No
Backfilling DAGs	Yes	No	No	No	No	No
Tasks prioritizations	Yes	No	No	No	No	No
Workflow portability due to standardized and commonly used workflow description specification	Yes	Yes	No	Yes	Yes	No
Simultaneous running multiple workflows	Yes	No	Yes	Yes	Yes	Yes
Implements workflow queue	Yes	No	Yes	Yes	Yes	Yes