



SOFTWARE TOOL ARTICLE

REVISED Cluster Flow: A user-friendly bioinformatics workflow tool
[version 2; referees: 3 approved]

Philip Ewels ¹, Felix Krueger², Max Källér³, Simon Andrews²

¹Department of Biochemistry and Biophysics, Science for Life Laboratory, Stockholm University, Stockholm, Sweden

²Bioinformatics Group, The Babraham Institute, Cambridge, UK

³Science for Life Laboratory, School of Biotechnology, Division of Gene Technology, Royal Institute of Technology, Stockholm, Sweden

v2 First published: 06 Dec 2016, 5:2824 (doi: [10.12688/f1000research.10335.1](https://doi.org/10.12688/f1000research.10335.1))
Latest published: 02 May 2017, 5:2824 (doi: [10.12688/f1000research.10335.2](https://doi.org/10.12688/f1000research.10335.2))

Abstract

Pipeline tools are becoming increasingly important within the field of bioinformatics. Using a pipeline manager to manage and run workflows comprised of multiple tools reduces workload and makes analysis results more reproducible. Existing tools require significant work to install and get running, typically needing pipeline scripts to be written from scratch before running any analysis. We present Cluster Flow, a simple and flexible bioinformatics pipeline tool designed to be quick and easy to install. Cluster Flow comes with 40 modules for common NGS processing steps, ready to work out of the box. Pipelines are assembled using these modules with a simple syntax that can be easily modified as required. Core helper functions automate many common NGS procedures, making running pipelines simple. Cluster Flow is available with an GNU GPLv3 license on GitHub. Documentation, examples and an online demo are available at <http://clusterflow.io>.

Open Peer Review

Referee Status:

	Invited Referees		
	1	2	3
REVISED			
version 2	report		
published 02 May 2017			
version 1			
published 06 Dec 2016	report	report	report

- 1 **Alastair R. W. Kerr** , University of Edinburgh UK, **Shaun Webb**, University of Edinburgh UK
- 2 **Stephen Taylor**, University of Oxford UK, **Jelena Telenius**, University of Oxford UK
- 3 **David R. Powell**, Monash University Australia

Discuss this article

Comments (0)

Corresponding author: Max Källér (max.kaller@scilifelab.se)

How to cite this article: Ewels P, Krueger F, Källér M and Andrews S. **Cluster Flow: A user-friendly bioinformatics workflow tool [version 2; referees: 3 approved]** *F1000Research* 2017, 5:2824 (doi: [10.12688/f1000research.10335.2](https://doi.org/10.12688/f1000research.10335.2))

Copyright: © 2017 Ewels P *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution Licence](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the [Creative Commons Zero "No rights reserved" data waiver](#) (CC0 1.0 Public domain dedication).

Grant information: This work was supported by the Science for Life Laboratory and the National Genomics Infrastructure (NGI) as well as the Babraham Institute and the UK Biotechnology and Biological Sciences Research Council (BBSRC).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 06 Dec 2016, 5:2824 (doi: [10.12688/f1000research.10335.1](https://doi.org/10.12688/f1000research.10335.1))

REVISED Amendments from Version 1

In version 2 we added a few additions to the manuscript text in response to the referee comments. Primarily: description of 'param' pipeline modifiers; software version logging; and new cloud computing support.

See referee reports

Introduction

As the field of genomics matures, next-generation sequencing is becoming more and more affordable. Experiments are now frequently run with large numbers of samples with multiple conditions and replicates. The tools used for genomics analysis are increasingly standardised with common procedures for processing sequencing data. It can be inconvenient and error prone to run each step of a workflow or pipeline manually for multiple samples and projects. Workflow managers are able to abstract this process, running multiple bioinformatics tools across many samples in a convenient and reproducible manner.

Numerous workflow managers are available for next-generation sequencing (NGS) data, each varying in its approach and use. Many of the popular tools allow the user to create analysis pipelines using specialised domain specific languages (*Snakemake*¹, *NextFlow*², *Bpipe*³). Such tools allow users to rewrite existing shell scripts into pipelines and are principally targeted at experienced bioinformaticians with high throughput requirements. They can be used to create highly complex analysis pipelines that make use of concepts, such as divergent and convergent data flow, logic checkpoints and multi-step dependencies. Using such a free-form approach allows great flexibility in workflow design.

Whilst powerful, this flexibility comes at the price of complexity. Setting up new analysis pipelines with these tools can be a huge task that deters many users. Many NGS genomics applications don't require such advanced features and can instead be run using a simple, mostly linear, file based system. Cluster Flow aims to fill this niche: numerous modules for common NGS bioinformatics tools come packaged with the tool ([Supplementary Table 1](#)), along with ready to run pipelines for standard data types. By using a deliberately restricted data flow pattern, Cluster Flow is able to use a simple pipeline syntax. What it lacks in flexibility it makes up for with ease of use; sensible defaults and numerous helper functions make it simple to get up and running.

Cluster Flow is well suited to those running analysis for low to medium numbers of samples. It provides an easy setup procedure with working pipelines for common data types out of the box, and is great for those who are new to bioinformatics.

Methods

Implementation

Cluster Flow is written in Perl and requires little in the way of installation. Files should be downloaded from the web and added to the user's bash `PATH`. Command line wizards then help the user to

create a configuration file. Cluster Flow requires pipeline software to be installed on the system and directly callable or available as environment modules, which can be loaded automatically as part of the packaged pipelines.

Operation

Cluster Flow requires a working Perl installation with a few minimal package dependencies, plus a standard bash environment. It has been primarily designed for use within Linux environments. Cluster Flow is compatible with clusters using Sun GRIDEngine, SLURM and LSF job submission software. It can also be run in 'local' mode, instead submitting background jobs using bash.

Pipelines are launched using the `cf` Perl script, with input files and other relevant metadata provided as command line options. This script calculates the required jobs and launches jobs accordingly.

Modules and pipelines

Cluster Flow uses modules for each task within a pipeline. A module is a standalone program that uses a simple API to request resources when Cluster Flow launches. The module then acts as a wrapper for a bioinformatics tool, constructing and executing a suitable command according to the input data and other specified parameters. The online Cluster Flow documentation contains extensive documentation about how to write new modules, making it possible for users to create new modules for missing tools.

Where appropriate, modules can accept `param` modifiers on the command line or in pipeline scripts that change the way that a module runs. For example, custom trimming options can be supplied to the *Trim Galore!* module to change its behaviour. The parameters accepted by each module are described in the Cluster Flow documentation.

Modules are strung together into pipelines with a very simple pipeline configuration script ([Supplementary Figure 1](#)). Module names are prefixed with a hash symbol (`#`), and tab spacing indicates whether modules can be run in parallel or in series. Parameters recognised by modules can be added after the module name or specified on the command line to customise behaviour.

Genomes

Cluster Flow comes with integrated reference genome management. At its core, this is based on a configuration file listing paths to references with an ID and their type. An interactive command line wizard helps with building this file, able to automatically search for common reference types. Once configured, the genome ID can be specified when running Cluster Flow, making multiple reference types available for that assembly. This makes pipelines simple and intuitive to launch ([Figure 1A](#)).

Pipeline Tracking

Unlike most other pipeline tools, Cluster Flow does not use a running process to monitor pipeline execution. Instead, it uses a file-based approach, appending the outputs of each step to '.run' files. When running in a cluster environment, cluster jobs are queued using the native dependency management. Cluster Flow can also be run locally, using a bash script in a background job to run modules

A Launch Pipeline

```
$ cf --genome GRCh37 fastq_bismark *.fastq.gz
```

B Check Status

```
$ cf --qstat
=====
Cluster Flow Pipeline: fastq_bismark
Submitted:             14 seconds ago
Working Directory:    /home/clusterflow/demo/demofiles
Cluster Flow ID:      fastq_bismark_1432818534
=====

- bismark_align [8 cores]
  - bismark_deduplicate
  - bismark_methXtract
  - bismark_report

- trim_galore [3 cores]
  - bismark_align
  - bismark_deduplicate
  - bismark_methXtract
  - bismark_report

- trim_galore [3 cores]
```

C E-mail Notification

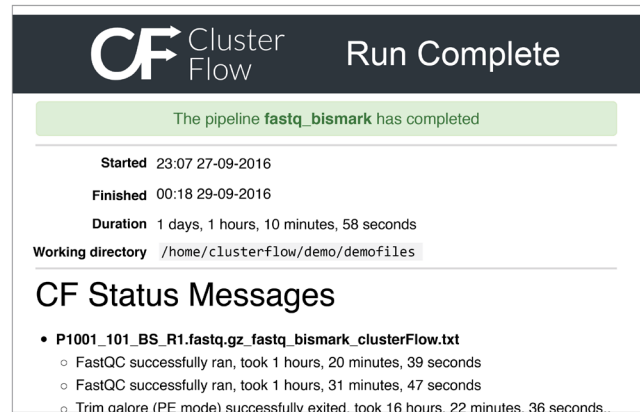


Figure 1. Process for (A) Launching an analysis pipeline, (B) checking its status on the command line and (C) a typical notification e-mail.

in series. The current status can be queried using a subcommand, which prints the queued and running steps for each pipeline along with information such as total pipeline duration and the working directory (Figure 1B).

Notifications and logging

When pipelines finish, Cluster Flow automatically parses the run log files and builds text and HTML summary reports describing the run. These include key status messages and list all commands executed. Any errors are clearly highlighted both within the text and at the top of the report. This report is then e-mailed to the user for immediate notification about pipeline completion, clearly showing whether the run was successful or not (Figure 1C).

Cluster Flow modules collect the software version of the tools used when they run. These are standardised, saved to the log files and included in the summary e-mail upon pipeline completion. Cluster Flow logs are recognised by the reporting tool *MultiQC*⁴ allowing reporting of software versions and pipeline details in MultiQC reports alongside output from the pipeline tools. System information (PATH, user, loaded environment modules, sysinfo) is also logged to the submission log when a pipeline is started.

Helper functions

Much of the Cluster Flow functionality is geared towards the end-user, making it easy to launch analyses. It recognises paired-end and single-end input files automatically, grouping accordingly and triggering paired-end specific commands where appropriate.

Regular expressions can be saved in the configuration that will automatically merge multiplexed samples before analysis and FastQ files are queried for encoding type before running. If URLs are supplied instead of input files, Cluster Flow will download and run these, enabling public datasets to be obtained and analysed in a single command. Cluster Flow is also compatible with *SRA-explorer* (<https://ewels.github.io/sra-explorer/>), which fetches download links for entire SRA projects. Such features can save a lot of time for the user and prevent accidental mistakes when running analyses.

Use cases

Cluster Flow is designed for use with next-generation sequencing data. Most pipelines take raw sequencing data as input, either in FastQ or SRA format. Outputs vary according to the analysis chosen and can range from aligned reads (eg. BAM files) to quality control outputs to processed data (eg. normalised transcript counts). Tool wrappers are written to be as modular as possible, allowing custom data flows to be created.

The core Cluster Flow program is usually installed centrally on a cluster. This installation can have a central configuration file with common settings and shared reference genome paths. Users can load this through the environment module system and create a personal configuration file using the Cluster Flow command line setup wizard. This saves user-specific details, such as e-mail address and cluster project ID. In this way, users of a shared cluster can be up and running with Cluster Flow in a matter of minutes.

Cloud-computing is becoming an increasingly practical solution to the requirements of high-throughput bioinformatics analyses. Unfortunately, the world of cloud solutions can be confusing to newcomers. We are working with the team behind Alces Flight (<http://alces-flight.com>) to provide a quick route to using the Amazon AWS cloud. Alces Flight provides a simple web-based tool for creating elastic compute clusters which come installed with the popular Open Grid Scheduler (SGE). Numerous bioinformatics tools are available as environment modules, compatible with Cluster Flow. We hope that Cluster Flow will soon be available and preconfigured as an such an app, allowing a powerful and simple route to running analyses in the cloud in just a few minutes with only a handful of commands.

Finally, Cluster Flow can also easily be used on single node clusters in *local* mode, as a quick route to running common pipelines. This is ideal for testing, though as there is no proper resource management it is not recommended for use with large analyses.

Conclusions

We describe Cluster Flow, a simple and lightweight workflow manager that is quick and easy to get to grips with. It is designed to be as simple as possible to use - as such, it lacks some features of other tools such as the ability to resume partially completed pipelines and the generation of directed acyclic graphs. However, this simplicity allows for easy installation and usage. Packaged modules and pipelines for common bioinformatics tools mean that users don't have to start from scratch and can get their first analysis launched within minutes. It is best suited for small to medium sized research groups who need a quick and easily customisable way to run common analysis workflows, with intuitive features that help bioinformaticians to launch analyses with minimal configuration.

Supplementary material

Typical pipeline script and a list of modules with tool description and URL. The script shows the analysis pipeline for reduced representation bisulfite sequencing (RRBS) data, from FastQ files to methylation calls with a project summary report. Pipeline steps will run in parallel for each read group for steps prefixed with a hash symbol (#). All input files will be channelled into the final process, prefixed with a greater-than symbol (>). List of modules excludes Core Cluster Flow modules. List valid at time of writing for Cluster Flow v0.4.

[Click here to access the data.](#)

Software availability

Cluster Flow available from: <http://clusterflow.io>

Source code available from: <https://github.com/ewels/clusterflow>

Archived source code as at time of publication: <https://doi.org/10.5281/zenodo.57900>

License: GNU GPLv3

Author contributions

PE wrote the tool and manuscript. FK provided coding help and advice. MK supported further development and contributed to the manuscript. SA conceived the initial concept, helped with code and provided manuscript feedback. All authors were involved in the revision of the draft manuscript and have agreed to the final content.

Competing interests

No competing interests were disclosed.

Grant information

This work was supported by the Science for Life Laboratory and the National Genomics Infrastructure (NGI) as well as the Babraham Institute and the UK Biotechnology and Biological Sciences Research Council (BBSRC).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

The authors would like to thank S. Archer, J. Orzechowski Westholm, C. Wang and R. Hamilton for contributed code and discussion.

References

- Köster J, Rahmann S: **Snakemake—a scalable bioinformatics workflow engine.** *Bioinformatics.* 2012; **28**(19): 2520–2522. [PubMed Abstract](#) | [Publisher Full Text](#)
- Di Tommaso P, Chatzou M, Floden EW, *et al.*: **Nextflow enables reproducible computational workflows.** *Nat Biotechnol.* 2017; **35**(4): 316–319. [PubMed Abstract](#) | [Publisher Full Text](#)
- Sadeghi SP, Pope B, Oshlack A: **Bpipe: a tool for running and managing bioinformatics pipelines.** *Bioinformatics.* 2012; **28**(11): 1525–1526. [PubMed Abstract](#) | [Publisher Full Text](#)
- Ewels P, Magnusson M, Lundin S, *et al.*: **MultiQC: Summarize analysis results for multiple tools and samples in a single report.** *Bioinformatics.* 2016. **32**(19): 3047–3048. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)

Open Peer Review

Current Referee Status:   

Version 2

Referee Report 03 May 2017

doi:[10.5256/f1000research.12430.r22445](https://doi.org/10.5256/f1000research.12430.r22445)



Alastair R. W. Kerr 

Wellcome Trust Centre for Cell Biology, University of Edinburgh, Edinburgh, UK

I thank the authors for thoroughly addressing the points raised. I have no further suggestions for the manuscript.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Referee Report 16 February 2017

doi:[10.5256/f1000research.11134.r18293](https://doi.org/10.5256/f1000research.11134.r18293)



David R. Powell

Monash Bioinformatics Platform, Monash University, Clayton, VIC, Australia

This paper describes a pipeline tool, Cluster Flow, (<http://clusterflow.io/>) specifically for bioinformatics processing. Cluster Flow is well documented, and comes with many pipelines, and modules. Pipelines are built by combining modules. Modules define how to run specific tools, including the CPU and RAM requirements. The tool works by specifying a pipeline to run, which then creates a shell script that either submits jobs to a cluster or runs locally depending on configuration.

Cluster Flow is designed to be simple to use, but it does lack basic pipeline features such as being able to automatically re-run stages of a pipeline.

It is not clear whether parameters can be changed when running a pipeline. For example, selecting different adaptors for trimming, or different mapping thresholds for a short read aligner. While Cluster Flow is designed to be simple, it seems such a feature would be commonly needed.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 25 Apr 2017

Philip Ewels, Stockholm University, Sweden

We thank the reviewer for this review. Pipeline parameters can indeed be changed (for example, adapters and trimming lengths) using the --params command line option. New documentation about this has been written and mention of it added to the manuscript (section: *Modules and pipelines*).

Competing Interests: No competing interests were disclosed.

Referee Report 13 February 2017

doi:10.5256/f1000research.11134.r18292



Stephen Taylor¹, Jelena Telenius²

¹ Computational Biology Research Group (CBRG), Weatherall Institute of Molecular Medicine (WIMM), John Radcliffe Hospital, University of Oxford, Oxford, UK

² Weatherall Institute of Molecular Medicine (WIMM), University of Oxford, John Radcliffe Hospital, Headington, Oxford, UK

The authors present a useful automation pipeline for institutes, where significant amounts of similar analyses are run on daily basis. It's nice that potential users can get an idea of software by using the interactive web based terminal session (on <http://clusterflow.io/>). We recommend the software should be published but we have the following comments/questions.

1) We installed the software fairly easily to run in 'local mode', although we couldn't get it to run using Sun Grid Engine. It would be useful to put more documentation and/or examples here.

2) How easy it is to add non Perl code to the software?

It appears Perl is the main language to configure the pipelines but we were wondering about other languages. Are there standard procedures or templates for including R scripts and passing parameters to them, for example?

3) Can one fine-tune the pipeline while running it?

In contrast to adding changes to the pipelines, or adding new tools to the pipelines (which is more a system admin / senior bioinformatician task), one often needs to make frequent calls about "which parameters suit the analysis of this sequencing library the best" e.g. Thresholds for peak calling in ChIP-Seq. Can such thresholds be easily applied running the pipeline on the fly?

4) Which kind of visualisation/report generating software do the authors recommend?

As the pipeline produces a folder full of output results, it makes sense to have software to inspect these results. Which kind of software do you recommend to be used to this kind of task? Is there a concept of building reports? For example, is it recommended to use Labrador with CF (<https://github.com/ewels/labrador/>)?

5) How do the authors envisage managing multiple versions of very similar pipelines across different users and use cases without things becoming confusing and to encourage reuse of pipelines, rather than just creating new instances?

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 25 Apr 2017

Philip Ewels, Stockholm University, Sweden

Many thanks for your time in reading the Cluster Flow manuscript and your helpful comments. We have revised the manuscript to address these points and are grateful for the help in improving the quality of the paper. Responses to specific comments are described below:

1) We installed the software fairly easily to run in 'local mode', although we couldn't get it to run using Sun Grid Engine. It would be useful to put more documentation and/or examples here.

We have extended the installation documentation available on the website. A walkthrough screencast tutorial is available on the Cluster Flow homepage (and YouTube). More concrete examples are difficult due to the varying setups of different clusters, though we continue to provide support via GitHub issues and e-mail.

It appears Perl is the main language to configure the pipelines but we were wondering about other languages. Are there standard procedures or templates for including R scripts and passing parameters to them, for example?

Any language can be used for software modules, however Perl is recommended because of the available Cluster Flow functions which greatly simplify the interaction with the core program. There have been example modules bundled with Cluster Flow for Python and R, but they are difficult to maintain with the changes to the core Cluster Flow code and not advertised as a result. From experience, we find that it is usually easier to write a simple perl module which in turn executes downstream custom scripts.

In contrast to adding changes to the pipelines, or adding new tools to the pipelines (which is more a system admin / senior bioinformatician task), one often needs to make frequent calls about "which parameters suit the analysis of this sequencing library the best" e.g. Thresholds for peak calling in ChIP-Seq. Can such thresholds be easily applied running the pipeline on the fly?

Whilst parameters and thresholds cannot be altered once a pipeline is running, it is possible to tweak such settings when launching an analysis. This is done by using the --params command line flag (can also be specified within pipeline files). We were somewhat shocked to realise that there was no documentation of this feature anywhere and have added a new section to the Cluster Flow documentation. This describes all available --params for every module. We have added mention of this to the manuscript (section: *Modules and pipelines*).

As the pipeline produces a folder full of output results, it makes sense to have software to inspect these results. Which kind of software do you recommend to be used to this kind of task? Is there a concept of building reports? For example, is it recommended to use Labrador with CF (<https://github.com/ewels/labrador>)?

Labrador is able to view some results from Cluster Flow pipelines (such as the .html completion report). However, the authors have written another tool called MultiQC which is able to summarise all results from a pipeline into a single html file [1]. See <http://multiqc.info> for more information.

5) How do the authors envisage managing multiple versions of very similar pipelines across different users and use cases without things becoming confusing and to encourage reuse of pipelines, rather than just creating new instances?

This is of some concern to the authors, and we encourage users to submit new modules and pipelines back to the main Cluster Flow repository so that they are available to everyone. However, we do not want to sacrifice flexibility, and so aim for maximum traceability by saving pipeline and module information for every run. A central repository for modules and pipelines has also been discussed (see comments to review 1 above) but is not yet being actively worked on.

[1] MultiQC: Summarize analysis results for multiple tools and samples in a single report. Philip Ewels, Måns Magnusson, Sverker Lundin and Max Källér. *Bioinformatics* (2016) doi: [10.1093/bioinformatics/btw354](https://doi.org/10.1093/bioinformatics/btw354)

Competing Interests: No competing interests were disclosed.

Referee Report 19 December 2016

doi:[10.5256/f1000research.11134.r18295](https://doi.org/10.5256/f1000research.11134.r18295)



Alastair R. W. Kerr , **Shaun Webb**

Wellcome Trust Centre for Cell Biology, University of Edinburgh, Edinburgh, UK

As the software is available and in use in multiple institutions (and thus tried and tested), I have no problems with accepting the manuscript. I feel that the manuscript and/or the linked documentation would benefit from some changes noted below.

Install

Having a copy of the install instruction in the downloaded tarball would be useful.

The “cf” executable uses the FindBin Perl module to establish the location of the script and hence the relative path to the CF Perl modules. Therefore the install must add the clusterflow directory to the PATH and would not function if the “cf” executable was symlinked to a directory on the PATH. This should be made clear in the install instructions although this is alluded to in the manuscript.

Adding genomes

The program can add genomes from installed locations in the filesystem. A helper script to autoinstall

from Ensembl/UCSC public sites would be a benefit. Moreover it is unclear if missing index files for mapping programs are generated automatically and permanently stored when running pipelines. This would be useful and easy to implement.

Metadata

I am glad to see the workflow captures metadata such as software versions and this should be highlighted in the manuscript. A reporting tool to extract this information, perhaps in a tabular format, from the log files would be useful.

Reproducibility

Output from the pipelines are depended on the software versions on the PATH. This is not ideal and an easy way to configure software versions would be useful to allow reproducible pipelines. I assume that “modules” are what the maintainers imagine most people would use? Docker would have been a nice solution.

Adding programs

There is information in the on-line documentation to add new programs to clusterflow by writing wrappers. This functionality should be noted in the manuscript.

Upgrades

It is unclear how clusterflow can be upgraded (I assume that new tarball needs to be downloaded) and whether there are repositories for new pipelines or tools. For example it would be useful for a community facility for depositing new tools and pipelines.

Language

Is providing compatibility with the common workflow language [CWL]¹ a possibility or a likelihood?

Resources

I would like more detail on the following:

How exactly is runs/threads/memory managed on a single node cluster? How happens if multiple users each run cf? Are instances aware of each other? Do the scripts check how many jobs are running or how much free memory is available?

References

1. Amstutz P, Tijanić N, Chapman B, Chilton J, Heuer M, Kartashov A, Lee H, Ménager H, Nedeljkovich M, Scales M, Soiland-Reyes S, Stojanovic L: Common Workflow Language, v1.0. *Figshare*. 2016.

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 25 Apr 2017

Philip Ewels, Stockholm University, Sweden

Many thanks for your time in reading the Cluster Flow manuscript and your helpful comments. We have revised the manuscript to address these points and are grateful for the help in improving the quality of the paper. Responses to specific comments are described below:

Having a copy of the install instruction in the downloaded tarball would be useful.

All of the Cluster Flow documentation is included with the downloaded tarball as markdown files within the 'docs' directory. However, we agree that this could be more visible. We have rewritten the main README.md file (also shown on the GitHub front page) to include brief installation instructions with a link to the longer documentation.

The "cf" executable uses the FindBin Perl module to establish the location of the script and hence the relative path to the CF Perl modules. Therefore the install must add the clusterflow directory to the PATH and would not function if the "cf" executable was symlinked to a directory on the PATH. This should be made clear in the install instructions although this is alluded to in the manuscript.

Thank you for alerting us to this issue. We have updated Cluster Flow to use \$RealBin instead of \$Bin, which makes it work with symlinks.

The program can add genomes from installed locations in the filesystem. A helper script to autoinstall from Ensembl/UCSC public sites would be a benefit. Moreover it is unclear if missing index files for mapping programs are generated automatically and permanently stored when running pipelines. This would be useful and easy to implement.

We agree that such a helper script would be useful and will look into writing this for a future release. At the time of writing, missing index files are not automatically generated. We recommend using [illumina iGenomes](#) where appropriate. We have written a [helper tool](#) to add centralised reference genomes for users running Cluster Flow on the Swedish UPPMAX clusters.

I am glad to see the workflow captures metadata such as software versions and this should be highlighted in the manuscript. A reporting tool to extract this information, perhaps in a tabular format, from the log files would be useful.

Mention of this has been added to the manuscript (section: *Notifications and logging*). In the new Cluster Flow release (v0.5), modules have been updated to extract and standardise the version numbers. These are now included in summary e-mails and parsed by a new Cluster Flow module written for MultiQC. MultiQC produces also machine-readable versions of this data (tsv, csv, json or yaml).

Output from the pipelines are depended on the software versions on the PATH. This is not ideal and an easy way to configure software versions would be useful to allow reproducible pipelines. I assume that "modules" are what the maintainers imagine most people would use? Docker would have been a nice solution.

As the reviewer suggests, Cluster Flow was primarily designed for use with environment modules as a method to standardise tools used and support for that is built in. We have added to the Cluster Flow submission log file, which now contains information about all loaded environment modules, all

directories currently on the PATH, the current user and information about the compute environment. Users are able to run Cluster Flow inside a docker container if desired, using local mode.

There is information in the on-line documentation to add new programs to clusterflow by writing wrappers. This functionality should be noted in the manuscript.

This is now described within the manuscript (section: *Modules and pipelines*).

It is unclear how clusterflow can be upgraded (I assume that new tarballs need to be downloaded) and whether there are repositories for new pipelines or tools. For example it would be useful for a community facility for depositing new tools and pipelines.

This is correct, Cluster Flow is updated by replacing the program files with a new tarball download. A community repository for Cluster Flow modules and pipelines has previously been discussed, and we hope to be able to work on this project in the future.

Is providing compatibility with the common workflow language [CWL] a possibility or a likelihood?

The authors have looked into such compatibility in the past, including discussing the point with Michael Crusoe when he visited our institute to present CWL. Unfortunately, due to differing assumptions and architectures within the two systems it seems unlikely that this will be pursued.

How exactly is runs/threads/memory managed on a single node cluster? How happens if multiple users each run cf? Are instances aware of each other? Do the scripts check how many jobs are running or how much free memory is available?

Running Cluster Flow in local mode on a single node cluster is fairly simplistic. There is no resource management and instances are not aware of each other. It was primarily written for easy testing and low-throughput runs where this will not be a problem. If running a lot of jobs on a single server we recommend installing a job management system such as SLURM. We have added a sentence describing this to the manuscript (section: *Use Cases*).

Competing Interests: No competing interests were disclosed.