Published on Mar 07, 2023 by Arcadia Science **DOI**: 10.57844/arcadia-cc5j-a519

Creating reproducible workflows for complex computational pipelines

A workflow orchestration framework can streamline repeatable tasks and make workflows broadly usable. From several options, we chose Nextflow due to the ease of deploying across platforms, vibrant nf-core community, and ability to manage and monitor workflows with Nextflow Tower.

Contributors (A-Z)

Feridun Mert Celebi, Seemay Chou, Jase Gehring, Megan L. Hochstrasser, Elizabeth A. McDaniel, Austin H. Patton, Taylor Reiter, Dennis A. Sun

Version 3 · Mar 31, 2025

Purpose

One of our goals at Arcadia Science is to create maximally useful computational tools, including our workflows. To increase the usability of our science, we want to openly share as many of these workflows as possible, while respecting the licenses of the underlying software and our translational goals. This is challenging in the life sciences. Computational pipelines have many individual steps, deal with varying sizes and formats of files, and use tools from diverse language ecosystems (e.g. Python, bash, R). This creates complications in maintaining software dependencies and compute environments, making workflows hard to use, reproduce, and iterate on.

Workflow frameworks have emerged as solutions to these challenges **[1][2][3][4]**. Each of these systems comes with benefits and drawbacks. In this pub, we share our decision-making process for selecting Nextflow as our workflow framework for standardized computational tasks. We have explored many tools, including but not limited to Snakemake **[4]**, LatchBio, Nextflow **[2]**, and Prefect. All of these tools present different trade-offs.

In summary, we sought a general solution within our budgetary constraints that minimized frustrations while aligning with our scientific goals and open-science principles. We knew our selection wouldn't necessarily meet each requirement perfectly, but we kept our mission of maximal utility at the center of our thought process.

- This pub is part of the **project**, "Useful computing at Arcadia." Visit the project narrative for more background and context.
- A simplified template for creating Nextflow workflows is available on GitHub. Use the template or see how we put it together.

Key considerations

We considered three key selection criteria. Our ideal tool for creating open-source pipelines would be:

Easily shareable with others

We want our pipelines to be accessible to others outside of Arcadia without substantial infrastructure or configuration on their end. The ideal system would be deployable on any computational infrastructure (e.g. local laptop, HPC, cloud providers, etc.). We did not want to require community users to "buy in" to a platform that they are not already using to access and use our workflows.

Reasonably cost- and time-effective

Our software team is currently small with a particular focus on building useful, focused products. We have limited bandwidth to tackle DevOps challenges, so we wanted a simple solution that:

1. Allowed us to run our pipelines easily on Amazon Web Services (AWS), our default data storage and compute tool.
2. Had a straightforward integration with AWS Batch, which would allow us to run pipelines in parallel (fast) on AWS Spot instances (cheap).

Flexible for Arcadians to learn and use

We aim to keep computation in the hands of scientists, each an expert in their respective domains. Forcing them to learn a new language would create a huge barrier to entry. Not only do bioinformatics tools span multiple languages in general, but our scientists have a diverse background of languages they are familiar with. Overall our scientists are familiar with bash, R, and Python — languages that are popular for computing in the life sciences. Our workflow solution therefore had to be able to execute code in these languages.

Which workflow systems did we consider?

We considered a range of solutions, but mostly centered our conversations around Snakemake, Nextflow, LatchBio, and Prefect. The former two are well-known workflow solutions in the life sciences used in both academic and industrial settings, whereas LatchBio is a new bioinformatics platform company providing compute solutions for scientists. The software/data engineering space also developed many tools to tackle the challenge of orchestrating the deployment of high-throughput computational pipelines. This is why our comparison included Prefect, an open-source workflow management tool commonly used for ETL (extract, transform, load) jobs. We compared these solutions, keeping in mind our key considerations:

	Nextflow	Snakemake	LatchBio	Prefect
Cost	✓ Free, paid license for Tower*	✓ Free	😑 Free for academics, pay per usage for industry	✓ Free, paid license for Prefect Cloud
Open source	✓	✓	✓	✓
Easy-to-learn language that life scientists are familiar with	✗ Steep learning curve	😑 Make-like DSL	✓ Python decorators	✓ Python decorators
Multi-language support	✓	✓	😑 Yes, but using Python subprocess calls	😑 Yes, but using Python subprocess calls
Easy resource management	✓	✓	✓	✗
Integrates with AWS Batch	✓ Tower* makes this more streamlined	😑 Support with AWS Genomics CLI, but has limitations	✗ Must use their platform	😑 Yes, but challenging
Easily deployable across diverse platforms	✓	😑 Sort of	✗ Must use their platform	😑 Sort of
Actively used by the life sciences community	✓	✓	😑 Not yet/growing space	✗

*Seqera Labs offers Nextflow Tower (also referred to as Tower), a monitoring and management system for launching Nextflow workflows on a variety of platforms.

What made Nextflow + Tower the right solution for us

Taking our criteria into consideration, we chose to move forward with writing workflows in Nextflow and deploying through AWS Batch via a paid license for Nextflow Tower.

This solution ultimately won out because:

- Nextflow integrates nicely with containerization technologies like Docker and Singularity to encourage easy deployability across platforms
- Vibrant open-source community where individuals test, develop, and share building blocks for workflows
- Straightforward to run on AWS Batch with or without Nextflow Tower
- Works across languages and varying compute needs, although this comes at the cost of a steep learning curve of a new Domain-Specific Language (DSL) and programming paradigm

We did not pursue the other options listed above for various reasons. Although Snakemake is widely used in the life sciences community, there were several logistical challenges to integrating Snakemake workflows with AWS Batch to take advantage of cost savings with parallelizing spot EC2 instances. LatchBio had a very streamlined development and user experience. However, using LatchBio would require us to ask downstream users to “buy in” to their platform to be able to test and use our pipelines. This was a dealbreaker for us. Finally, similar to LatchBio, Prefect’s development experience was great. But the lack of adoption in the bioinformatics space, coupled with the inability to specify compute resource requirements at the task level made Prefect a nonviable option for us. As these tools and constraints evolve, we may reconsider our choice.

Easily deployable workflows on any platform

The Nextflow workflow system is open source and has an active community maintaining it and providing support. By design, Nextflow encourages containerization leveraging Docker or Singularity images. This facilitates deployment on a variety of

platforms (e.g. in the cloud, locally, or on an institution's HPC), making our workflows usable by others with various setups. For users who cannot use Docker or Singularity, workflows can also specify their environments with conda.

Seamless monitoring and management of AWS Batch jobs with Tower

The main draw of Nextflow was the ability to use Nextflow Tower, an intuitive GUI that provides seamless integration with AWS Batch and built-in monitoring and management of AWS Spot instances. Note that Tower is not required to use any workflows — it is just a complementary layer that facilitates the execution and distribution of workflow processes across a variety of systems, not just AWS Batch. By using Tower, we can more easily integrate with AWS Batch and tap into cost savings with spot-instances without a lot of overhead, all while enabling non-expert users to interactively launch pipelines through the Tower GUI.

We purchased a license for Nextflow Tower that enables us to have three power users and three launchpad users, where power users can both configure compute environments and launch workflows, and launchpad users can only launch workflows. Originally, our idea was to enable all scientists at Arcadia to be able to launch their own workflows using the Tower GUI. However, this was extremely cost-prohibitive. Instead, we now use the Tower API and Tower launch hooks to automate the launch of our workflows while maintaining Tower for management and monitoring purposes. You can find an example implementation of this on Seqera's blog or our seqqc pipeline **[5]**.

Vibrant nf-core community and resources

Nextflow has a vibrant community and resources like trainings, hackathons, and a Slack space for peer learning, all available through nf-core **[6]**. Most importantly, nf-core provides helper tools for setting up new workflows with a template and integrating community-sourced modules to streamline the development process. Nextflow modules are typically individual steps used in a larger workflow, so having a community-maintained repository of repeatedly used modules to integrate into different workflows is a nice feature. Similarly, community-developed and maintained

workflows (such as nf-core/MAG **[7]** and nf-core/viralrecon **[8]** make it easy to apply quick, existing solutions to data.

This wealth of resources signaled active use of Nextflow by the broader community, suggesting that our workflows could be useful to others outside of our organization.

Challenges and lessons learned

Overall, we grappled with choosing a workflow system for quite a while before eventually deciding to dive into using Nextflow and Tower. However, we leaned on our operating principles, including “When in doubt, just try the experiment” and “No decision is truly irreversible,” and moved forward since this solution met most of our requirements. We have already started to write and launch Nextflow workflows with Tower using AWS cloud computing for our standardized work.

As mentioned above, no tool is perfect and its usage is context-dependent. Here, we’ll describe some challenges that we have run into building Nextflow pipelines and how we’ve solved them.

Navigating learning curves

One of the greatest challenges to institutionalizing the use of Nextflow at Arcadia has been its relatively steep learning curve. We know this because we ran a small experiment with one of our target scientists to assess exactly how steep that curve could be, time-wise. In short, this took a lot of time for several reasons.

First, stitching together distinct modules requires the developer to know both the workflow logic as well as all relevant coding languages. Most of our computational scientists are familiar with bash, R, and Python programming languages for data exploration and formal analysis, whereas Nextflow is written in Groovy, a superset of Java. Although you don’t have to know all of the ins and outs of Groovy to write Nextflow workflows, it can be extremely challenging to learn a new programming paradigm (flow-based programming via channels) and a DSL.

Second, implementing this solution with our AWS setup requires workflows to be containerized via Docker or Singularity, which requires our scientists to adopt a

different mental model for putting together workflows than they might previously be used to. Granted, most cloud-based solutions require containerization, not just Nextflow. We have plans to address some of these topics so any scientist can eventually feel comfortable with these tasks through our internal Arcadia Users' Group (AUG) for peer-led teaching and learning of computational skills.

While there was a time cost for navigating the above problems, more quantitatively defining that cost did provide a helpful constraint for thinking about when this effort might be worthwhile.

Highly flexible, once configured

One of the biggest reasons we chose Nextflow was the numerous resources available through the nf-core community. We mostly find the nf-core helper tools useful for our work, but there are issues we've had to grapple with. We were initially thrilled with how seemingly fast one could get up and going with a new Nextflow workflow by using the nf-core tools to create a template workflow and add premade modules. However, we realized that there are significantly more options that the nf-core helper tools give a user to make a Nextflow workflow than what the average user of Nextflow wants to do. Although this complexity may be helpful for developers familiar with Nextflow, it can also be intimidating to new users.

This led us to build a pruned, simplified template to reduce the cognitive load on our scientists in developing their workflows. This approach allows us to deploy smaller additions in shorter time periods.

> Try the template for yourself [DOI: 10.5281/zenodo.7717173], or check out how we built it [DOI: 10.5281/zenodo.7690298].

Making this decision for yourself

There are so many workflow systems to choose from that it can feel impossible to narrow it down to one solution. Even though Arcadia has unique open-science constraints, our decision ultimately wasn't influenced by this alone. Being able to

reproduce computational work internally is an important part of performing rigorous scientific research.

If you are weighing options, consider some lessons we have learned that you could take into consideration for your team or organization:

1. **What are your available computing resources and constraints?**
   These can include how much physical space you have, cost restrictions, the size of your team, time available to spend on DevOps, etc.
2. **What types of pipelines do you plan to implement?**
   Will these pipelines use a single language? Will they need varying amounts of compute power? How often will you want to run these pipelines?
3. **How broadly reusable do you want your pipelines to be?**
   For example, Nextflow is highly parameterized and modular so that little pieces can be reused across different workflows, which differs from other solutions, such as Snakemake.
4. **Who is the intended audience for your workflows?**
   Different solutions make sense depending on the usage needs of your resulting pipelines.
5. **What are the people around you at your organization already familiar with or comfortable picking up?**
   Successfully implementing a new workflow system in your organization is partly dependent on being able to find support from others around you when you run into issues.

What's next?

Overall, it is an exciting time for bioinformatics tooling with multiple options in this space for streamlining standardized workflows and many up-and-coming solutions. Watch this space as we release workflows to meet the needs of our current science. We designed our first workflow to perform quality checks and analyze sequencing data to ensure basic standards are met before making our data publicly available and continuing with downstream analyses **[5]**.

Acknowledgements

We thank Kelsey Florek at the Wisconsin State Lab of Hygiene for early discussions about Nextflow and Nextflow Tower that helped kickstart our work!

We thank Jaclyn Taroni and Josh Shapiro at the Childhood Cancer Data Lab for early discussions about their experience with Nextflow!

References

1. Reiter T, Brooks† PT, Irber† L, Joslin† SEK, Reid† CM, Scott† C, Brown CT, Pierce-Ward NT. (2021). Streamlining data-intensive biology with workflow systems. https://doi.org/10.1093/gigascience/giaa140

2. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. (2017). Nextflow enables reproducible computational workflows. https://doi.org/10.1038/nbt.3820

3. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanić N, Ménager H, Soiland-Reyes S, Gavrilovic B, Goble C. (2021). Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language. https://doi.org/10.48550/ARXIV.2105.07028

4. Köster J, Rahmann S. (2012). Snakemake—a scalable bioinformatics workflow engine. https://doi.org/10.1093/bioinformatics/bts480

5. Chou S, Reiter T. (2024). Speeding up the quality control of raw sequencing data using seqqc, a Nextflow-based solution. https://doi.org/10.57844/ARCADIA-CXN6-CH62

6. Ewels PA, Peltzer A, Fillinger S, Patel H, Alneberg J, Wilm A, Garcia MU, Di Tommaso P, Nahnsen S. (2020). The nf-core framework for community-curated bioinformatics pipelines. https://doi.org/10.1038/s41587-020-0439-x

7 Krakau S, Straub D, Gourlé H, Gabernet G, Nahnsen S. (2022). nf-core/mag: a best-practice pipeline for metagenome hybrid assembly and binning. https://doi.org/10.1093/nargab/lqac007

8 Harshil Patel, Varona S, Monzón S, Espinosa-Carrasco J, Heuer ML, Nf-Core Bot, Underwood A, Gabernet G, Ewels P, MiguelJulia, Kelly S, Stevin Wilson, , Erika, Sameith K, Garcia MU, Jcurado, Menden K. (2022). nf-core/viralrecon: nf-core/viralrecon v2.5 - Manganese Monkey. https://doi.org/10.5281/ZENODO.6827984