WFPM: a novel WorkFlow Package Manager to enable collaborative workflow development via reusable packages

Junjun Zhang

NCI Containers and Workflows Interest Group (CWIG) Webinar Series
September 10, 2021

Agenda

- Challenges in workflow development and ICGC ARGO best practices
- How does WFPM help addressing the challenges?
- WFPM internals
- Demo
- Comparison with ‘similar’ initiatives
- Future development
- Q & A

Goals of ICGC ARGO

The International Cancer Genome Consortium Accelerating Research in Genomic Oncology (ICGC ARGO) aims to ***uniformly analyze*** specimens from ***100,000*** donors with high quality clinical data in order to address outstanding questions that are vital to the quest to defeat cancer.

- Over the next ten years ICGC ARGO aims to deliver **a million patient-years of precision oncology knowledge** to the world
- Coordinate the **integration of genomic and phenotypic data on 100,000 cancer patients** enrolled in clinical trials or from well annotated cohorts
- Use this detailed clinical and genomic data to **address key clinical and biological questions** of relevance to specific cancer types
- Make the **data available to the entire research community in a rapid and responsible way**, to accelerate research into the causes and control of cancer

https://www.icgc-argo.org/page/64/about-icgc-argo

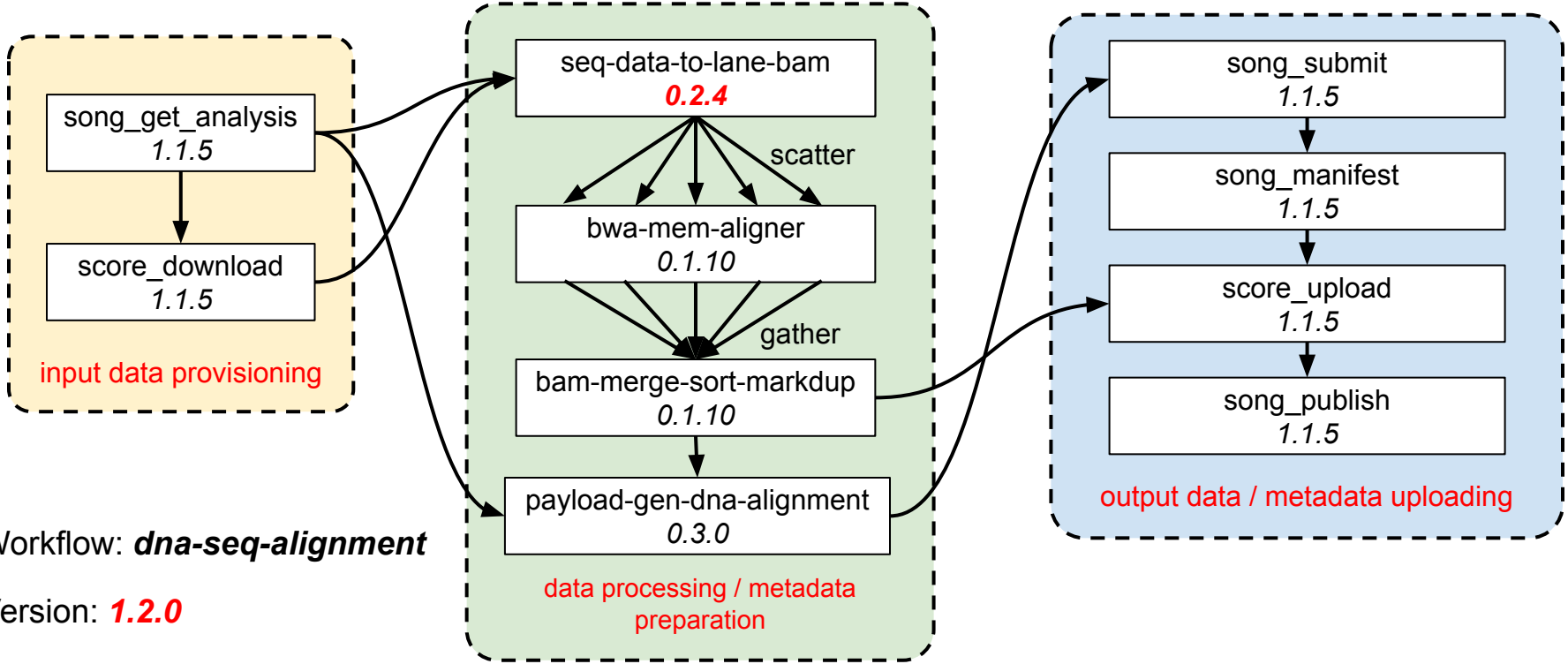
Challenges in workflow development

- Reproducibility
- Portability
- Code reuse and sharing
- Collaboration
- Testability

ICGC ARGO best practices - five guiding principles

- **Reproducible** - same input, same code, same result
  - containerize all software tools (including scripts, binary executables) and OS-dependent packages
  - tag every image build and associate it with workflow source code release
- **Portable** - run on different platforms, by different users
  - containerize all software tools
  - use cross-platform workflow languages and orchestration systems, eg, Nextflow, WDL or CWL etc
- **Composable** - enable collaborative development
  - break down big tasks into small tasks (**each carried out by a small software tool**)
  - **independently version and release every tool package and its associated container image**
  - a released tool is immutable and **can be imported into any workflow where it is needed**
  - a workflow can also be imported as a **sub-workflow** to build larger workflows
  - similar to tools, workflows are versioned, immutable once released
- **Findable** - easy to find by research community members
  - register components and workflows in public tool registries, such as Dockstore, BioContainers etc
  - release workflow source code via GitHub Releases
- **Testable** - deliver with high confidence
  - must have tests for every tool, component and workflow
  - configure and enable continuous integration testing as an integral part of the development process

Schematic diagram of the BWA-MEM alignment workflow



Workflow: ***dna-seq-alignment***

Version: ***1.2.0***

Software reuse

For decades, discussion of software reuse was far more common than actual software reuse. Today, the situation remains for bioinformatics workflow developers, they still write duplicated code, still hard to even reuse their own code.

Russ Cox, tech lead of the GO team. Our Software Dependency Problem (Jan 2019), https://research.swtch.com/deps

Build something amazing using reusable parts, collaboratively

Photo credit: Toronto Grand Prix Tourist Blog

What's missing? How can we reuse code in workflows?

- Recent advances in bioinformatics workflow development solutions have focused on addressing reproducibility, portability and transparency. It has been very successful. However support for code reuse is limited, particularly across different codebases.
- Major workflow languages have some level of support for importing code blocks
  - Nextflow (DSL2): ***include <file://>***
  - WDL: ***import <file:// or http://>***
  - CWL: ***run <file:// or http://>***
- Limitations of the current importing mechanism
  - importee can ***only*** be a single file
  - importee is ***location-dependent*** *(URL vs URI)*
  - ***no version control*** of the importees

So, what is WFPM?

- ***WFPM*** stands for *WorkFlow Package Manager*. The design was heavily inspired by ***npm***, the node.js package manager for ***JavaScript***. WFPM works very similar to ***npm*** as well.
- WFPM solution consists of two parts:
  - design of the structure for how to organize workflow code and relevant metadata to form a workflow package and the mechanism for how packages are versioned, released, and imported as dependencies.
  - command line interface (CLI) tool that provides dependency management and assistance to develop and manage workflow packages.

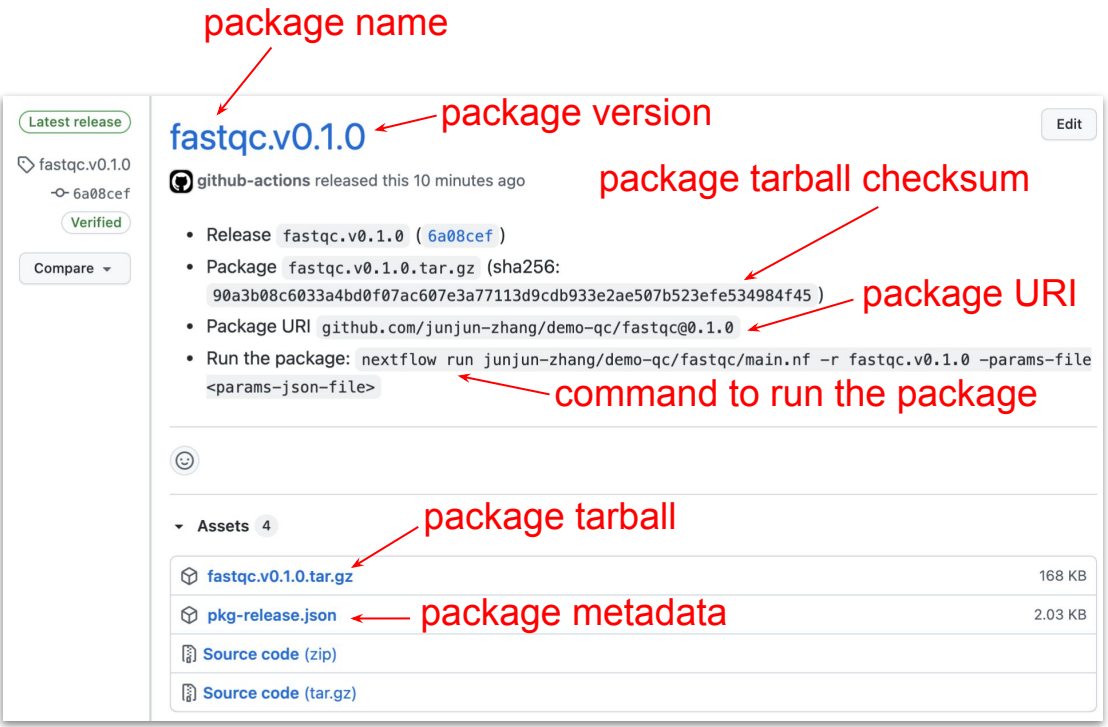
How does WFPM help addressing the challenges?

- ***WFPM*** provides a solution to enable composability as easy as ***plug & play***.
- Through a streamlined development process and automation at several key steps, ***WFPM CLI*** significantly lowers the barriers to develop standard reusable and shareable workflow packages, and ***ensures conformation to the established best practices***.
    - workflow code (currently ***Nextflow*** only) template generation
    - automated build, eg, create package tarball and metadata, create docker image and push to container registry
    - automated package testing
    - streamlined code review and releasing at GitHub
- To our knowledge, no similar solution exists for workflow development.

Anatomy of a WFPM project and its packages

```
my-wfpm-project                          # name of the WFPM project, also the repo name
├── .gitignore
├── .wfpm                                # WFPM project configuration file
├── .github
│   └── workflows
│       └── build-test-release.yml       # GitHub Actions code for automated CI/CD
├── LICENSE
├── LICENSE-short
├── README.md
├── demo-package                         # folder for the package named 'demo-package'
│   ├── pkg.json                         # pkg.json keeps basic package info, similar to package.json in npm
│   ├── main.nf                          # package entry script as specified in pkg.json
│   ├── nextflow.config                  # package nextflow default configuration
│   ├── Dockerfile                       # Dockerfile to build container image, with actual tool installed
│   ├── main.py                          # optional entry point, can be any scripting languages
│   ├── modules                          # a package may optionally have local modules
│   │   └── <local-module-1>
│   └── tests                            # folder for tests
│       ├── input                        # input data for tests
│       ├── expected                     # expected output for tests
│       ├── checker.nf                   # test launcher script
│       └── test-job-1.json              # test job 1, add more as needed
└── wfpr_modules                         # folder for installed dependent WFPM packages
    ├── <dependent-package-1@1.2.0>      # folder for installed package: dependent-package-1 ver: 1.2.0
    ├── <dependent-package-1@0.9.5>      # folder for installed package: dependent-package-1 ver: 0.9.5
    ├── <dependent-package-2@0.5.2>
    └── README.md
```

WFPM package is released using GitHub release assets



- No need for a centralized package registry (unlike ***npm***)
- Release process is fully automated as part of Continuous Integration / Continuous Delivery
- One GitHub repository is able to host source code of multiple workflow packages
  - a feature designed in WFPM from the beginning
  - avoid maintaining too many repos
  - every package is still completely independent from each other, with its own release cycle
- Package related container image is built by GitHub Actions and registered under GitHub Container Repository (***ghcr.io***)
- Released WFPM package can imported by anyone into their own workflow codebase

Dependency installation and import

- One of the package manager's main tasks is to resolve, download and install dependencies

```
my-wfpm-project
├── demo-dna-seq-processing-wf
│   ├── pkg.json                                    # pkg.json declares dependencies
│   ├── main.nf                                     # workflow entry point, imports dependencies
│   └── wfpr_modules -> ../wfpr_modules
└── wfpr_modules/github.com/icgc-argo/demo-wfpkgs   # wfpr_modules to keep installed dependencies
    ├── demo-bwa-mem-aligner@1.22.0
    ├── demo-bam-merge-sort-markdup@1.12.1
    ├── demo-utils@1.1.0
    └── demo-utils@1.2.0
```

- Import dependencies (*pseudo* code in Nextflow script ***main.nf***)

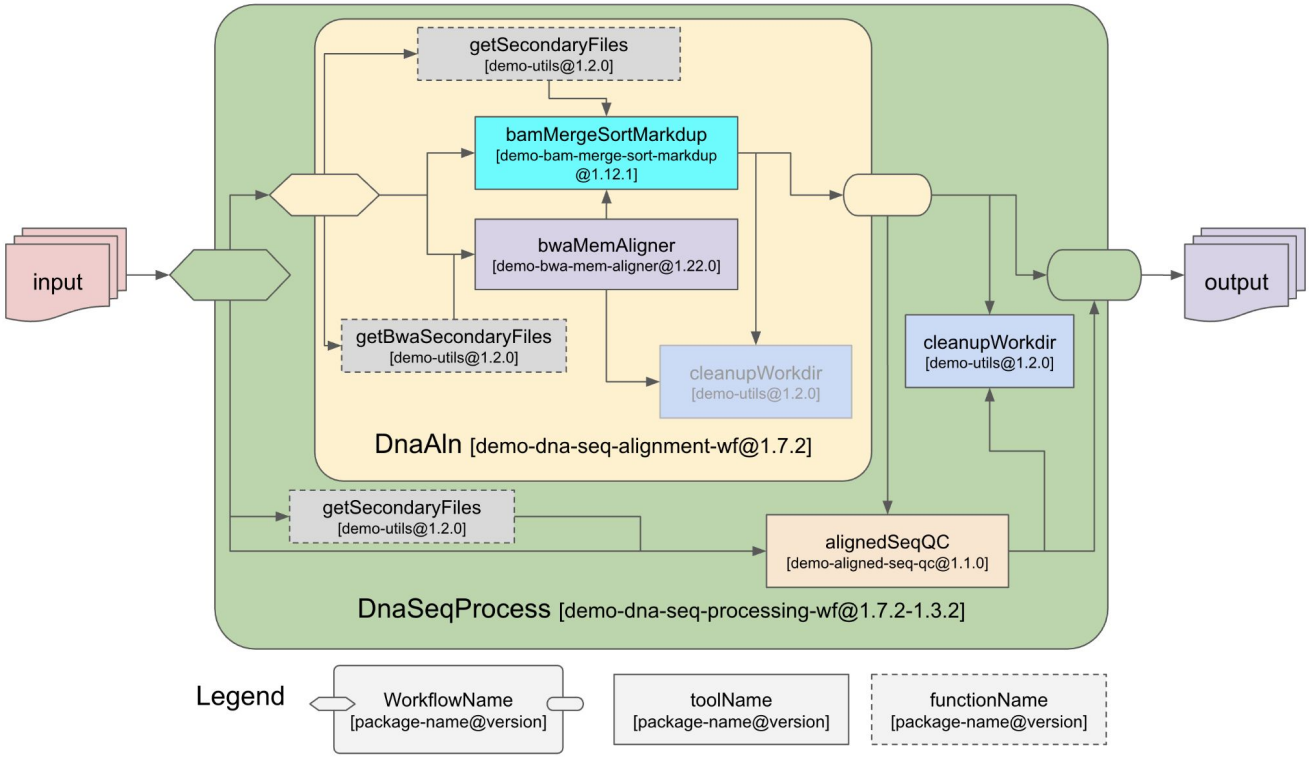
```
include { getSecondaryFiles }   from "demo-utils@1.1.0/main.nf"
include { cleanupWorkdir }      from "demo-utils@1.2.0/main.nf"
include { bwaMemAligner }       from "demo-bwa-mem-aligner@1.22.0/bwa-mem-aligner.nf"
include { bamMergeSortMarkdup } from "demo-bam-merge-sort-markdup@1.12.1/bam-merge-sort-markdup.nf"
```

- In WFPM, import must always specify full version of the dependent package
  - Extra safeguard for reproducibility
  - The well-known *diamond dependency problem** is nonexistent, different versions of the same package can coexist without causing any trouble. Similar approach is taken by the GO language** as part of major version strategies

* *https://www.well-typed.com/blog/2008/04/the-dreaded-diamond-dependency-problem*

** *https://blog.golang.org/v2-go-modules#TOC_3.*

Composing large workflows from reusable packages



Demo workflow source code repo: *https://github.com/icgc-argo/demo-wfpkgs*

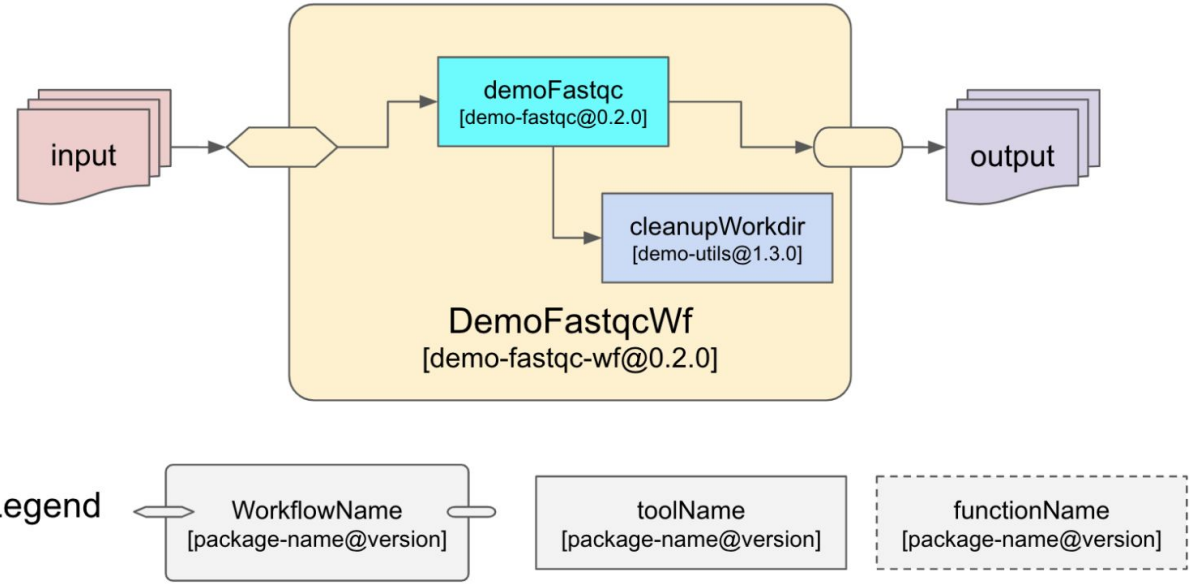
WFPM CLI tool usage summary

```
wfpm --help
Usage: wfpm [OPTIONS] COMMAND [ARGS]...

Options:
  -d, --debug / --no-debug  Show debug information in STDERR.
  -v, --version             Show wfpm version.
  --help                    Show this message and exit.

Commands:
  init       Start a workflow package project with necessary scaffolds.
  install    Install dependencies for the package currently being worked on.
  list       List local and installed dependent packages.
  new        Start a new package with necessary scaffolds.
  nextver    Start a new version of a released or in development package.
  outdated   List outdated dependent packages.
  test       Run tests.
  uninstall  Uninstall packages.
  workon     Start work on a package, display packages released or in dev.
```

Demo: create two WFPM packages



Package URIs:

- **cleanupWorkdir**: github.com/icgc-argo/demo-wfpkgs/demo-utils@1.3.0 (already available)
- **demoFastqc**: github.com/icgc-tcga-pancancer/awesome-wfpkgs1/demo-fastqc@0.2.0 (to be built in case 1)
- **DemoFastqcWf**: github.com/icgc-tcga-pancancer/awesome-wfpkgs2/demo-fastqc-wf@0.2.0 (to be built in case 2)

ICGC ARGO production workflow development

- ARGO workflow source code: *https://github.com/icgc-argo-workflows*
- **Five core workflows** developed by the ARGO DCC Bioinformatics Team:
  - DNA-Seq BWA-MEM alignment
  - Sanger WGS variant calling
  - Sanger WXS variant calling
  - GATK Mutect2 variant calling
  - and Open Access Filtering for somatic variant calls
- Over **50** reusable packages, **nine** of them are used in **all** workflows without any code duplication
- Globally distributed ARGO working group members have begun to collaborate to develop WFPM packages for more analytic workflows

Comparison with other workflow code sharing initiatives

	wfpm	*nf-core*	*cwldep*	*bio-cwl-tools*	*snakemake-wrappers*
single step tool modules	Y	Y	Y	Y	Y
subworkflows as shareable modules	Y	N	?	N	N
module development full lifecycle support	Y	Y	N	N	N
advanced dependency management	Y	N	N	N	N
individual module versioning	Y	N	N	N	N
individual module release artifact	Y	N	N	N	N
workflow language agnostic	Y	N	N	N	N
active development	Y	Y	N	Y	Y

***wfpm**: https://wfpm.readthedocs.io; **nf-core**: https://nf-co.re; **cwldep**: https://github.com/common-workflow-language/cwldep;*
***bio-cwl-tools**: https://github.com/common-workflow-library/bio-cwl-tools; **snakemake-wrappers**: https://snakemake-wrappers.readthedocs.io*

WFPM future development

- Support for other container technologies, eg, *Singularity, Podman* etc.
- Support for other container registries, eg, *Quay, DockerHub* etc.
- Support for other source code control system, eg, *GitLab, Bitbucket* etc.
- Integration with Dockstore (*https://dockstore.org*) to register WFPM packages in a GA4GH-compliant manner
- Input parameter schema support and auto-generated package usage documentation
- Although currently only *Nextflow* is supported, WFPM's designed to be workflow language agnostic, plan to add support for *WDL* and *CWL* etc.
- Building the community within ICGC ARGO and beyond, around both WFPM development and WFPM-based workflow package development

Acknowledgement

Support and funding

Funding provided by the Government of Ontario.

ARGO workflow code contributors (so far)

- Dušan Andrić
- Andrej Benjak
- Kate Eason
- Hillary Elrick
- Alvaro Feriz
- Andre Kahles
- Alex Lepsa
- Alvin Ng
- Desiree Schnidrig
- Morgan Taschuk
- Jose Espejo Valle-Inclan
- Yuk Kei Wan
- Linda Xiang
- Junjun Zhang

Thank you!

WFPM and ARGO workflow resources

- WFPM is licensed under *Apache* 2.0
- WFPM source code: *https://github.com/icgc-argo/wfpm*
- WFPM documentation: *https://wfpm.readthedocs.io*
- ARGO workflows: *https://github.com/icgc-argo-workflows*