National Cancer Institute

U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES

National Institutes of Health

Building FAIR Bioinformatics Workflows: A Case Study Using De Novo Mutations Variant Calling in Snakemake

Wei Zhu

11/13/2024

Supporting the Division of Cancer Epidemiology and Genetics

Cancer Genomics Research

Outline

- The FAIR data principles
- 10 quick tips for building FAIR workflows
  - Most of bioinformaticians are both workflow users and developers.
- Example: simple bash script VS Snakemake workflow
- Workflow management systems facilitate the development of FAIR workflows
  - Overview of bioinformatics analysis workflow
  - Common workflow management system for bioinformatics
- Walk through each of the 10 quick tips
  - Case study: TriosCompass
- Summary
- References

The FAIR data principles

https://www.biomeris.it/en/introduction-fair-principles/

Building FAIR workflows

PLOS COMPUTATIONAL BIOLOGY

OPEN ACCESS

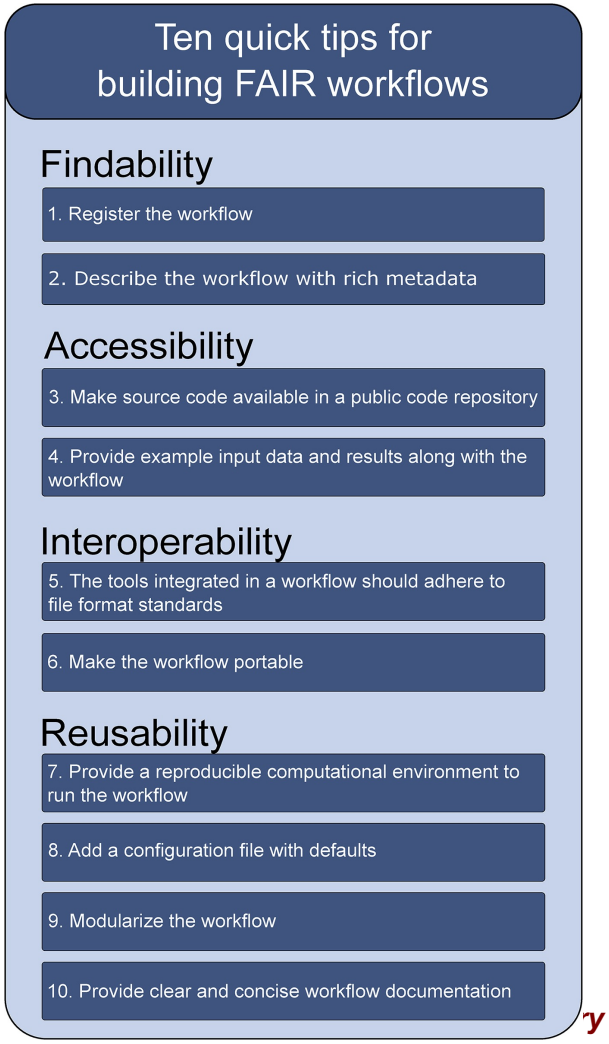
EDUCATION

Ten quick tips for building FAIR workflows

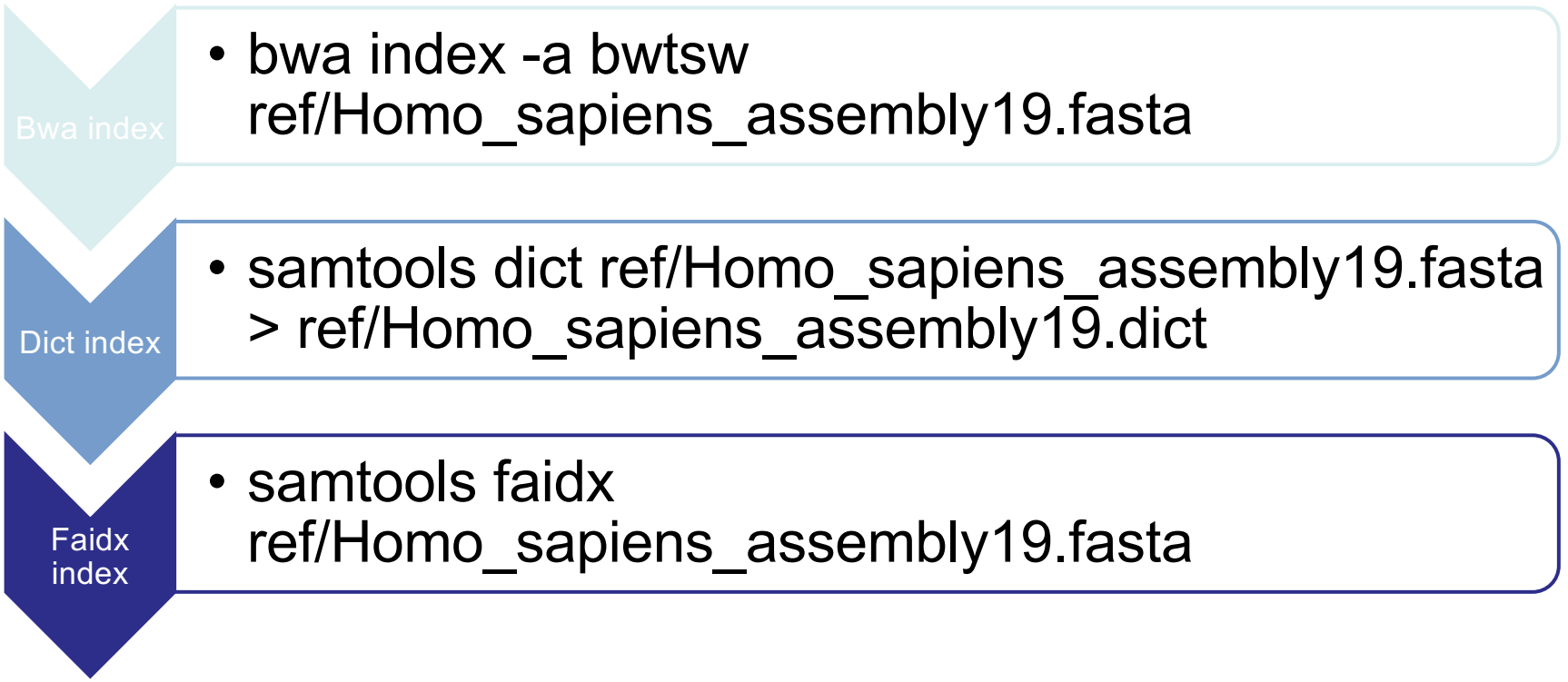
Casper de Visser, Lennart F. Johansson, Purva Kulkarni, Hailiang Mei, Pieter Neerincx, K. Joeri van der Velde, Péter Horvatovich, Alain J. van Gool, Morris A. Swertz, Peter A. C. 't Hoen, Anna Niehues

Published: September 28, 2023 • https://doi.org/10.1371/journal.pcbi.1011369

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1011369



A simple bash workflow to index reference genome



Run a simple bash workflow in parallel

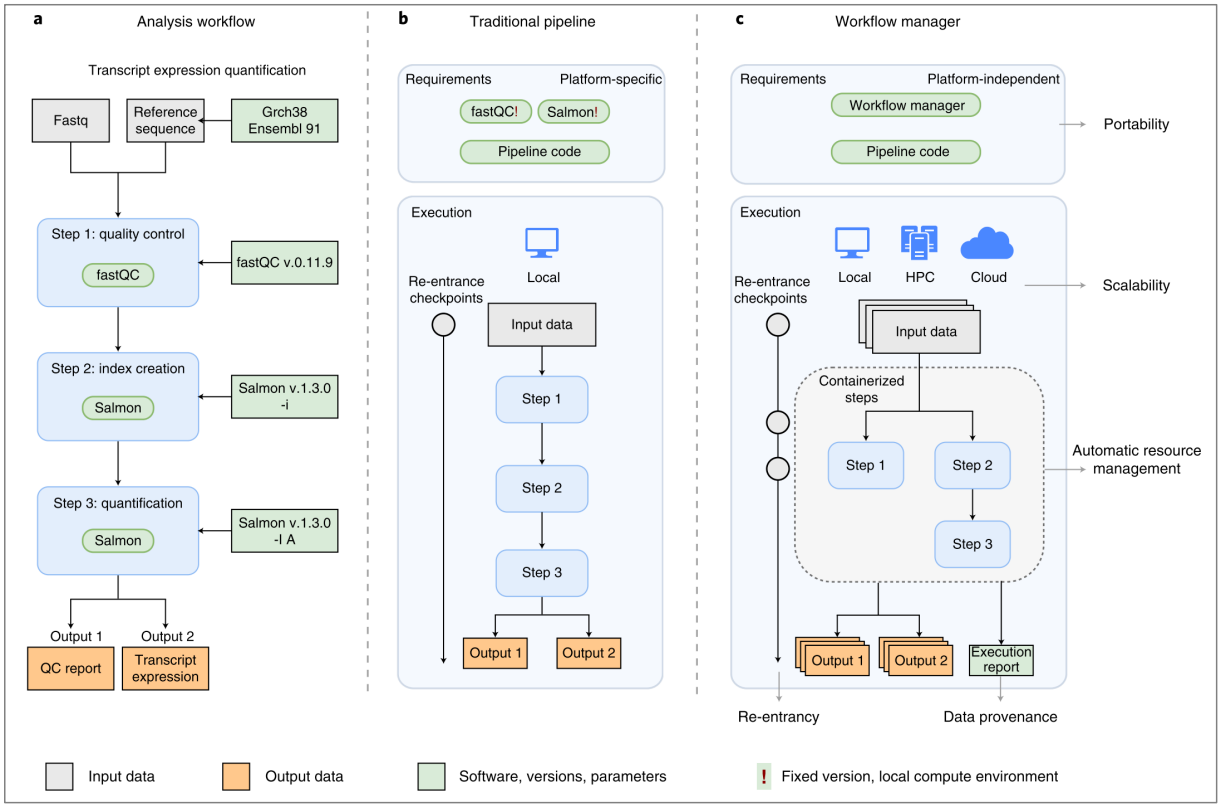
```
ls ref/*.fasta | parallel --dry-run 'bwa index -a bwtsw {} ; samtools
dict {} > {.}.dict ; samtools faidx {}'  > ref_index.cmd
```

```
swarm --partition norm -m bwa,samtools -J ref_index --verbose 3  --
time 10:00:00 -t 2 -g 40 -f ref_index.cmd
```

```
swarm --partition norm -m bwa/0.7.17,samtools/1.21 -J ref_index --
verbose 3  --time 10:00:00 -t 2 -g 40 -f ref_index.cmd
```

- Simple and easy to develop.
- Limitations
  - Not portable: dependencies on bwa and samtools
  - Do not know the expected output.
  - No prevention for pipeline breakage.
    - Add touch flag for completeness may improve the bash workflow.

Overview of bioinformatics analysis workflow



- Benefits from using Workflow management systems compared to the traditional pipeline
  - Portability
  - Scalability
  - Automatic resource management
  - Data provenance

Wratten et al (2021); Nature Methods

Workflow managers for bioinformatics

Table 1 | Overview of workflow managers for bioinformatics (top, editable version; bottom, image version)

Tool	Class	Ease of use[a]	Expressiveness[b]	Portability[c]	Scalability[d]	Learning resources[e]	Pipeline initiatives[f]
Galaxy	Graphical	●●●	●○○	●●●	●●●	●●●	●●○
KNIME	Graphical	●●●	●○○	○○○	●●◐	●●●	●●○
Nextflow	DSL	●●○	●●●	●●●	●●●	●●●	●●●
Snakemake	DSL	●●○	●●●	●●◐	●●●	●●○	●●●
GenPipes	DSL	●●○	●●●	●●○	●●○	●●○	●●○
bPipe	DSL	●●○	●●●	●●○	●●◐	●●○	●○○
Pachyderm	DSL	●●○	●●●	●○○	●●○	●●●	○○○
SciPipe	Library	●●○	●●●	○○○	○○○	●●○	○○○
Luigi	Library	●●○	●●●	●○○	●●◐	●●○	○○○
Cromwell + WDL	Execution + workflow specification	●○○	●●○	●●●	●●◐	●●○	●●○
cwltool + CWL	Execution + workflow specification	●○○	●●○	●●◐	○○○	●●●	●●○
Toil + CWL/ WDL/Python	Execution + workflow specification	●○○	●●●	●◐○	●●●	●●○	●●○

Wratten et al (2021); Nature Methods

Snakemake workflow to index reference genome

https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/rules/ref.smk

```
rule bwa_index:
    input:
        genome,
    output:
        idx=multiext(genome, ".amb", ".ann", ".bwt", ".pac", ".sa")
    threads:
        config["threads"]["bwa_index"]
    wrapper:
        "v2.3.2/bio/bwa/index"

rule genome_dict:
    input:
        genome,
    output:
        genome_dict,
    # conda:
    #     "../envs/samtools.yaml"
    singularity: "docker://euformatics/samtools:1.19.2"
    shell:
        "samtools dict {input} > {output} "

rule genome_faidx:
    input:
        genome,
    output:
        genome_fai,
    # conda:
    #     "../envs/samtools.yaml"
    singularity: "docker://euformatics/samtools:1.19.2"
    shell:
        "samtools faidx {input}  "
```

https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/rules/common.smk

```
import glob
from os import path

from snakemake.utils import validate


# validate(config, schema="../schemas/fastq_schema.yaml")

# location of genome fasta file with .fa or .fasta ext
genome = config["ref"]["sequence"]

refFile = os.path.basename(genome)
refDir = os.path.dirname(genome)
genome_prefix = os.path.splitext(refFile)[0]
genome_dict = refDir + '/' + genome_prefix + '.dict'
genome_fai = refDir + '/' + refFile + '.fai'

optional_output = list()
qc_output = list()
```

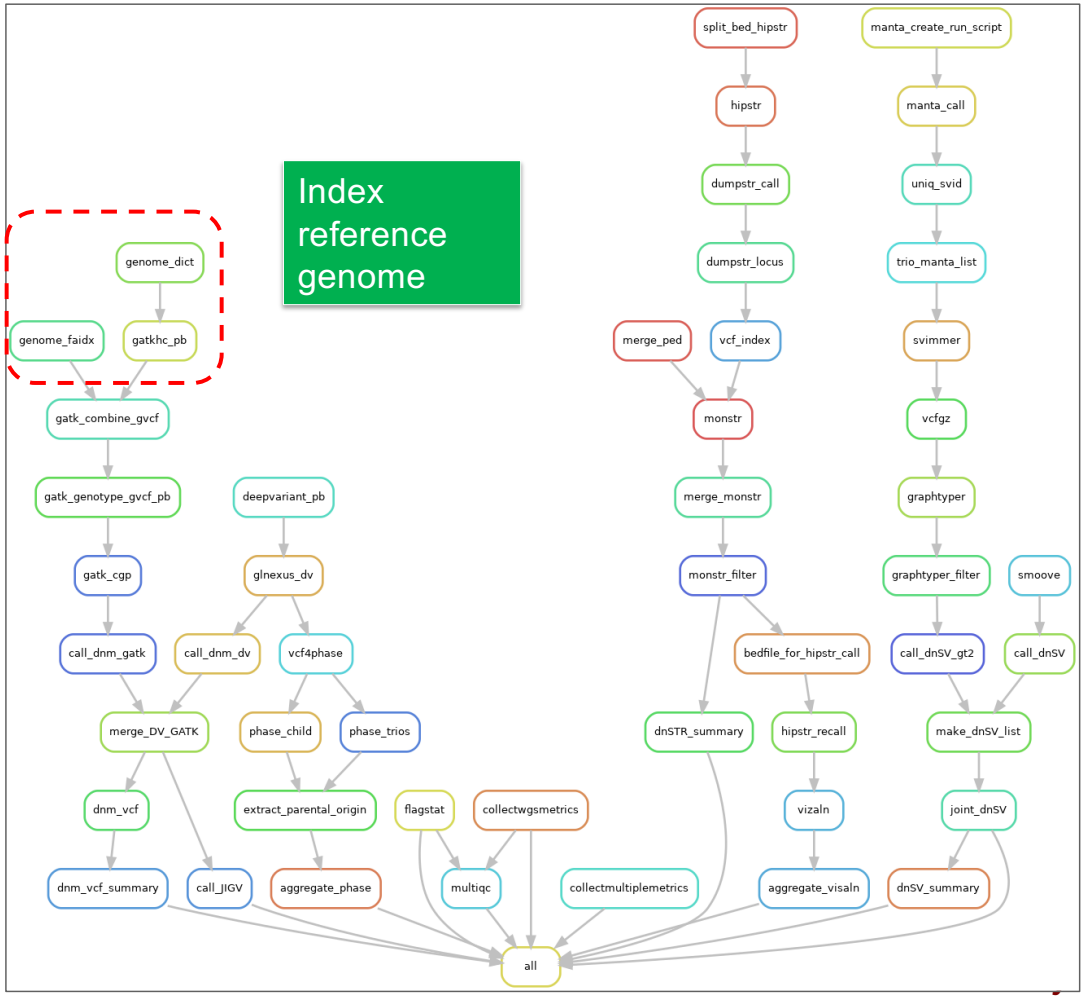
config.yaml

```
name: 8triosSplit

pepfile: "config/8triosSplit_pep.yaml"
pepschema: "../schemas/bam_schema.yaml"

output_dir: "output"


ref:
  sequence: "ref/Homo_sapiens_assembly38.fasta"
  build: "hg38"

ped_dir: "ped"
```

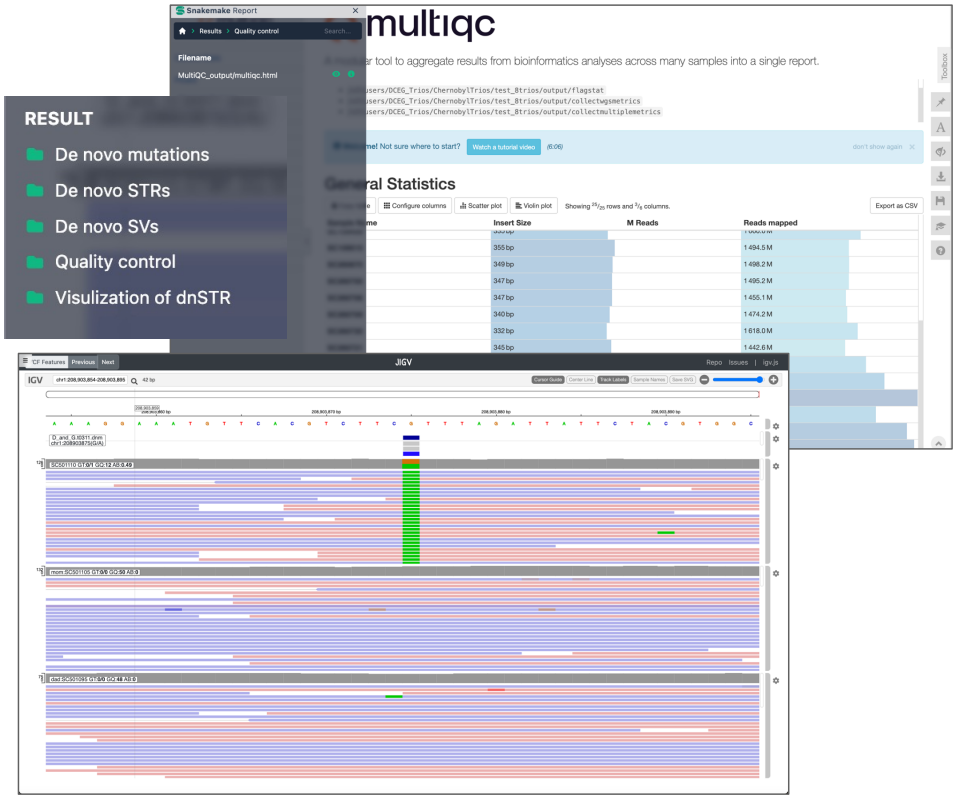
Cancer Genomics Research Laboratory

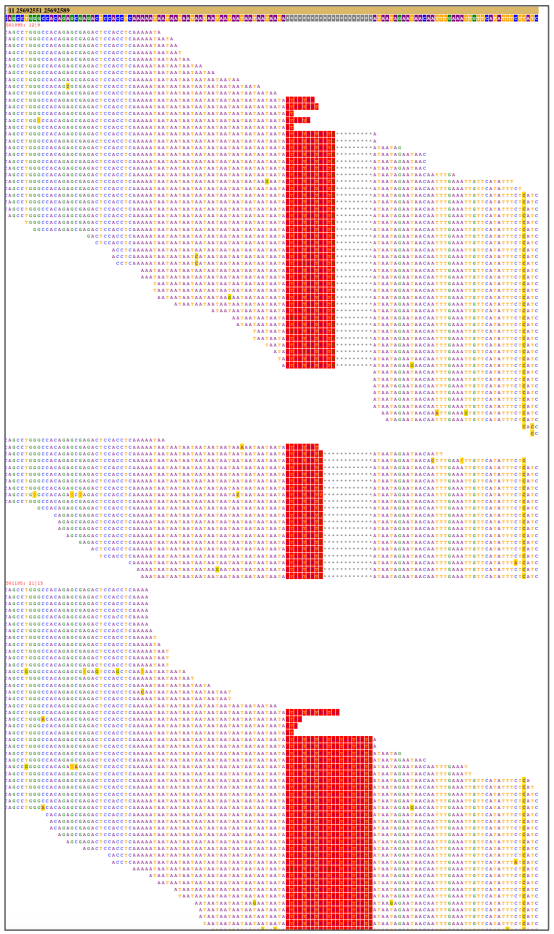
Snakemake workflow: TriosCompass

- https://github.com/NCI-CGR/TriosCompass_v2
- Call DNMs (de novo mutations) using *DeepVariant* and *GATK HaplotypeCaller*
- Phase DNMs using *whatshap*
- Call dnSTR (de novo simple tandem repeats) using *HipSTR* and *MonSTR*.
- Call dnSV (de novo structural variants) using *Manta*, *GraphType2* and *smoove*



Output of TriosCompass





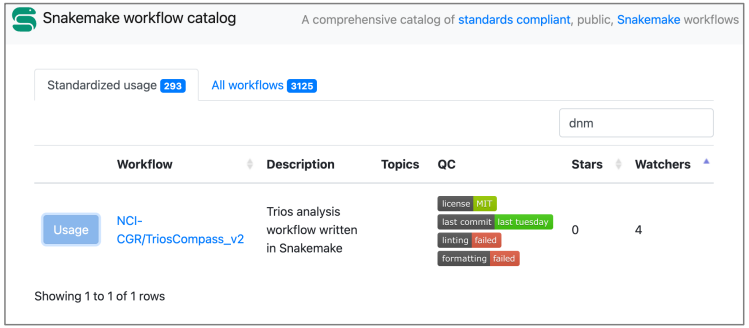
Cancer Genomics Research Laboratory

Findability

- Tip 1: Register the workflow
  - WorkflowHub
    - Common Workflow Language (CWL), Snakemake, Nextflow, and Galaxy
  - Dockstore
    - CWL, Workflow Description Language (WDL), Nextflow, and Galaxy
  - Workflow language-specific registries
    - nf-core
    - Galaxy
    - Snakemake workflow catalog
    - KNIMEhub

TriosCompass is automatically included by Snakemake workflow catalog

- https://snakemake.github.io/snakemake-workflow-catalog?rules=true
  - The workflow is contained in a public Github repository.
  - The repository has a readme, containing the words "snakemake" and "workflow" (case insensitive).
  - The repository contains a workflow definition named either Snakefile or workflow/Snakefile.
  - If the repository contains a folder rules or workflow/rules that folder must at least contain one file ending on .smk.
  - The repository is small enough to be cloned into a Github actions job (very large files should be handled via Git LFS, so that they can be stripped out during cloning).
  - The repository is not blacklisted here.



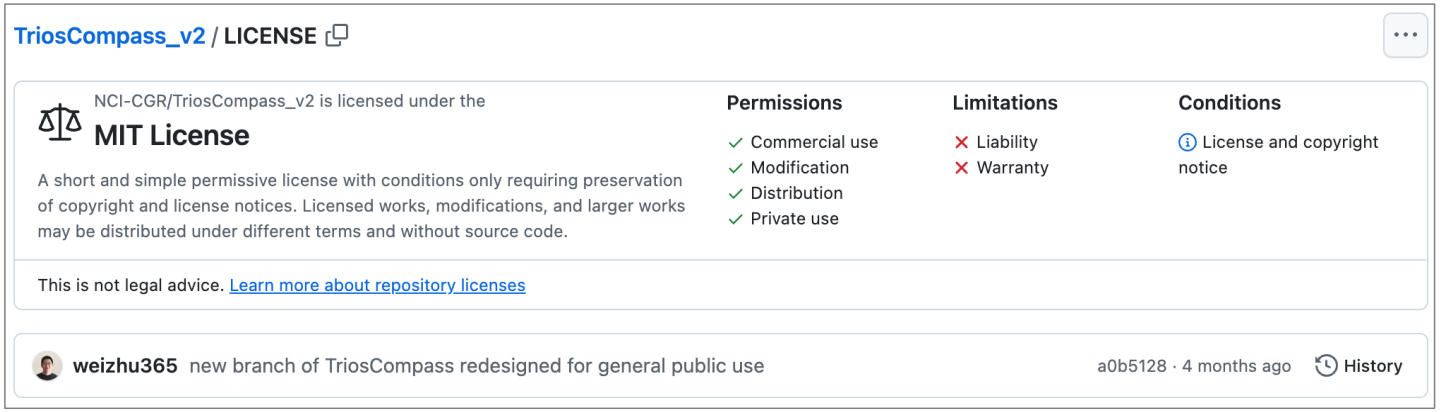
- Use Snakemake template to get started
  - https://github.com/snakemake-workflows/snakemake-workflow-template

Findability

- Tip 2: Describe the workflow with rich metadata
  - Rich meta data for all workflow components
    - Workflow language files
    - Scripts
    - Configuration files
    - Input data
    - Purposes, scopes, limitations, and etc

Accessibility

- Tip 3: Make source code available in a public code repository
  - GitHub, GitLab, and Bitbucket
  - The workflow's source code should be accompanied with clear open source licensing.
    - choosealicense.com



https://github.com/NCI-CGR/TriosCompass_v2/blob/main/LICENSE

Accessibility

- Tip 4: Provide example input data and results along with the workflow
  - Provide example data or give guidance on how to retrieve the data.
  - Generate synthetic data to protect privacy.
    - Random sampling of the original data's distribution.
  - All workflow results are best collected through comprehensive rendered reports.
    - Particularly useful for non-computational researchers.
  - Use test functions, unit tests to improve workflow development and workflow quality.

TriosCompass resource bundle: https://zenodo.org/uploads/13381196

Example data https://github.com/NCI-CGR/TriosCompass_v2/tree/main/.test/data

Report generated by TriosCompass

```
snakemake  \
      --report TriosCompass_full_report.zip \
      --profile TriosCompass_v2/workflow/profiles/slurm \
      --configfile TriosCompass_v2/config/config.yaml
```


Link to the report

Interoperability

- Tip 5: The tools integrated in a workflow should adhere to file format standards
  - Adopting standardized file formats increases interoperability.
    - SAM, BAM, UBAM, CRAM, VCF, MAF, GFF, etc.
    - Reuse individual workflow components in other workflows.
  - It is recommended to use file formats that are commonly used to read and write data frames in popular programming languages like R and Python, such as CSV and TSV.
  - For more complex data, file formats can be used that are more uncommon, but allow embedding of different data types, such as JSON, XML, or RDF.

Portable Encapsulated Project (PEP) format

- https://pep.databio.org/peppy/
- A PEP is a collection of metadata files conforming to a standardized structure, using simple YAML and TSV/CSV formats
  - *Portability* between computing environments
  - *Reusability* among different tools and project stages
  - *Durability* with respect to data movement

Using PEP to specify fastq input for TriosCompass

- fastq_pep.yaml

```yaml
pep_version: 2.0.0
sample_table: sample_fastq.csv

# In manifest file, Sample_ID + Flowcell should be unique
sample_modifiers:
  append:
    sample_name: "sn"
  derive:
    attributes: [sample_name]
    sources:
      sn: "{SAMPLE_ID}_{FLOWCELL}"
```

- sample_fastq.csv

	SAMPLE_ID	FLOWCELL	LANE	INDEX	R1	R2
2	HG002	BH2JWTDSX5	1	CGGTTGTT-GTGGTATG	data/fq/HG002_NA24385_son_80X_R1.fq.gz	data/fq/HG002_NA24385_son_80X_R2.fq.gz
3	HG003	BH2JWTDSX5	1	GCGTCATT-CAGACGTT	data/fq/HG003_NA24149_father_80X_R1.fq.gz	data/fq/HG003_NA24149_father_80X_R2.fq.gz
4	HG004	BH2JWTDSX5	1	CTGTTGAC-ACCTCAGT	data/fq/HG004_NA24143_mother_80X_R1.fq.gz	data/fq/HG004_NA24143_mother_80X_R2.fq.gz

SAMPLE_ID, FLOWCELL, LANE, INDEX, R1/2 are required

```
### One sample may have data from multiple flowcells
rule fq2bam:
    input:
        R1s=lambda w: expand(output_dir+"/fastp/{id}.R1.fastp.fastq.gz", id=[
w.SAMPLE_ID + '_'+ flowcell for flowcell in subj_flowcell_dict[w.SAMPLE_ID] ]),
        R2s=lambda w: expand(output_dir+"/fastp/{id}.R2.fastp.fastq.gz", id=[
w.SAMPLE_ID + '_'+ flowcell for flowcell in subj_flowcell_dict[w.SAMPLE_ID] ]),
        ref=genome,
        idx=rules.bwa_index.output
    output: output_dir + "/fq2bam/{SAMPLE_ID}.bam"
    threads: config["threads"]["fq2bam"]
    params:
       fqs=lambda w, input: " --in-fq ".join([r1+' '+r2+' ' + rg for r1,r2, rg in
zip(input.R1s, input.R2s, ["'@RG\\tPL:ILLUMINA\\tID:{FLOWCELL}_{LANE}\\tSM:
{SAMPLE_ID}\\tPU:{SAMPLE_ID}_{FLOWCELL}\\tLB:{SAMPLE_ID}_{INDEX}'" .format
(**samples.loc[id]) for id in list(compress(ids, [ id == w.SAMPLE_ID for id in
samples['SAMPLE_ID']]))  ])]),
       singularity_cmd = config["parabricks"]["singularity_cmd"]
    benchmark:
        output_dir + "/benchmark/fq2bam/{SAMPLE_ID}.tsv"
    shell: """
        {params.singularity_cmd} \
        pbrun fq2bam \
            --ref {input.ref} \
            --in-fq {params.fqs} \
            --out-bam {output}
    """

def get_bam(subj):
    return output_dir +"/fq2bam/{}.bam".format(subj)
```

https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/rules/mapping.smk

- The combination of SAMPLE_ID and FLOWCELL should be unique.
- FLOWCELL, LANE, INDEX and the definition of RG follow the CGR convention.
- Schema to validate csv input
  - https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/schemas/fastq_schema.yaml

Using PEP to specify bam input for TriosCompass

- bam_pep.yaml

```yaml
pep_version: 2.0.0
sample_table: sample_bam.csv


sample_modifiers:
  append:
    sample_name: "sn"
  derive:
    attributes: [sample_name]
    sources:
      sn: "{SAMPLE_ID}"
```

- sample_bam.csv

SAMPLE_ID	BAM
HG002	sorted_bam/HG002_NA24385_son_80X.bam
HG003	sorted_bam/HG003_NA24149_father_80X.bam
HG004	sorted_bam/HG004_NA24143_mother_80X.bam

Interoperability

- Tip 6: Make the workflow portable
  - By utilizing workflow managers, workflows can achieve higher portability, allowing them to operate seamlessly across different types of computational environments.
  - Some workflow languages can be run with different workflow engines.
    - Workflows written in CWL can be run with Cromwell and Galaxy.
    - Workflows written in WDL can be run with Cromwell and miniWDL.

- Module is an easy solution for local development in short term, but not stable or portable.
- Conda is less portable than container.
  - A conda env can be easily converted to container.
- Docker is be the best choice for the long term.

Reusability

- Tip 7: Provide a reproducible computational environment to run the workflow
  - Irreproducible research results can be caused by small differences in computational environments.
    - For example, Python/R versions, library/packages versions, or operating systems.
  - Solutions
    - Package managers
      - Conda, venv, renv
    - Containers
      - Docker, Singularity/Apptainer, Podman
      - Docker is commonly supported by other container engines.

https://github.com/NCI-CGR/TriosCompass_v2/blob/main/environment.yaml

```
name: TriosCompassV2
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
  - snakemake==7.3.7
  - pip==22.3.1
  - python==3.10.6
  #
https://github.com/snakemake/snakem
ake/issues/1899
  - tabulate==0.8.10
  - pip:
    - eido==0.1.9
    - peds==1.3.2
    - peppy==0.35.4
```

```
mamba env create -f environment.yaml

conda activate TriosCompassV2
```

Reusability

- Tip 8: Add a configuration file with defaults
  - A config file can be used to fine-tune the workflow execution on different levels (software/ hardware), through which it can be run in different computational environments without the need to modify the workflow implementation.
    - Command-line parameters
    - Equally important for workflow reproducibility are hardware specifications in the con- fig file.
  - Configuration with default values.
    - Documentation for both workflow and the configuration options.

Resource settings

Original version

```
rule fq2bam:
    input:
        R1s=lambda w: expand(pep.config.output_dir+
        R2s=lambda w: expand(pep.config.output_dir+
        ref=pep.config.hg38_ref
    output: pep.config.output_dir + "/fq2bam/{CGF_I
    params:
       fqs=lambda w, input: " --in-fq ".join([r1+'
    benchmark:
        pep.config.output_dir + "/benchmark/fq2bam/
    threads: 48
    resources:
        threads=48,
        mem_mb = 360000,
        disk_mb = 1000000,
        runtime= "8h",
        partition="gpu",
        slurm="gres=gpu:v100x:2",
        tmpdir=pep.config.output_dir + "/TMP"
    envmodules: "parabricks/4.0.0"
    shell: """
        pbrun fq2bam \
            --ref {input.ref} \
            --in-fq {params.fqs} \
            --out-bam {output}
    """
```

Recent version

config/config.yaml

```
threads:
  bwa_index: 16
  fastp: 8
  fastq_screen: 8
  fq2bam: 48
  flagstat: 16
  collectmultiplemetrics: 32
  gatkhc_pb: 24
  gatk_combine_gvcf: 16
  gatk_cgp: 16
  gatk_genotype_gvcf_pb: 16
  deepvariant_pb: 48
  glnexus_dv: 8
```

workflow/rules/mapping.smk

```
rule fq2bam:
    input:
        R1s=lambda w: expand(output_dir+"/fastp/{id}.R1.fastp.fas
        R2s=lambda w: expand(output_dir+"/fastp/{id}.R2.fastp.fas
        ref=genome,
        idx=rules.bwa_index.output
    output: output_dir + "/fq2bam/{SAMPLE_ID}.bam"
    threads: config["threads"]["fq2bam"]
    params:
       fqs=lambda w, input: " --in-fq ".join([r1+' '+r2+' ' + rg
       singularity_cmd = config["parabricks"]["singularity_cmd"]
    benchmark:
        output_dir + "/benchmark/fq2bam/{SAMPLE_ID}.tsv"
    shell: """
        {params.singularity_cmd} \
        pbrun fq2bam \
            --ref {input.ref} \
            --in-fq {params.fqs} \
            --out-bam {output}
    """
```

```
# Define the number of threads used by rules
# buggy about this setting parse (see 121M)
# set-threads:
#   - "bwa_index=16"
```

profiles/biowulf/config.yaml

```
# Job resources
set-resources:
  - bwa_index:mem_mb=16000
  - fastp:mem_mb=40000
  - fastqc:mem_mb=10000
  - fastq_screen:mem_mb=50000
  - fq2bam:mem_mb=40000
  - fq2bam:runtime="8h"
  - fq2bam:partition=gpu
  - fq2bam:slurm=gres=gpu:v100x:1
  - flagstat:mem_mb=80000
  - collectwgsmetrics:runtime="1d"
  - collectwgsmetrics:mem_mb=80000
  - collectmultiplemetrics:mem_mb=80000
```

- Set-threads does not work properly in Snakemake v7.3.7

TriosCompass documentation

- https://github.com/NCI-CGR/TriosCompass_v2/edit/main/README.md
- https://nci-cgr.github.io/TriosCompass_v2
  - Generate html pages using sphinx.
    - Convert Markdown to reStructuredText using *m2r2*
      - https://github.com/CrossNox/m2r2
  - Commit the changes to the branch gh-pages.
  - Publish the branch gh-pages to GitHub Pages

```
cd /Users/zhuw10/git/TriosCompass_v2
mkdir docs/

cd docs
sphinx-quickstart

pip install sphinxemoji


### after some configuration

cat ~/git/TriosCompass_v2/docs/readme.rst
 .. mdinclude:: ../README.md
    :start-line: 52


make html
```

Reusability

- Tip 9: Modularize the workflow
  - Building workflows in a modular structure.
    - nf-core-modules
    - Snakemake modules, wrappers (and meta-wrappers)
  - Another approach for workflow modularization is creating a software package that can be used outside of the workflow.

https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/Snakefile

```
include: "rules/common.smk"
include: "rules/pedigree.smk"
include: "rules/ref.smk"

if config["fastq_input"]["enable"]:
    include: "rules/premap.smk"
    include: "rules/mapping.smk"
else:
    include: "rules/bam_input.smk"

include: "rules/bam_qc.smk"
include: "rules/gatk_hc.smk"
include: "rules/deepvariant.smk"
include: "rules/call_dnm.smk"
include: "rules/jigv.smk"
include: "rules/phasing.smk"
include: "rules/dnSTR.smk"

if config["dnSV"]["enable"]:
    include: "rules/dnSV.smk"

if config["multiqc"]["enable"]:
    include: "rules/multiqc.smk"

rule all:
    input:
        output_dir + "/dnm_vcf_summary/DNM_summary.txt",
        expand(output_dir + "/call_JIGV/{fam}.JIGV.html", fam=fam_ids),
        expand(output_dir + "/phase_DNMs/{fam}.parental_origin.tab", fam=fam_ids),
        optional_output=optional_output,
```

Fastq/Bam input files

Optional components

Dynamic output

Reusability

- Tip 10: Provide clear and concise workflow documentation
  - Documentation can be provided in multiple forms.
    - README files
    - Workflow registries
    - Help pages
  - Schematic overview with a flowchart
  - The source code of the workflow can also function as documentation.
    - Snakemake is closed related with python.

```
import glob
from os import path

from snakemake.utils import validate


# validate(config, schema="../schemas/fastq_schema.yaml")

# location of genome fasta file with .fa or .fasta ext
genome = config["ref"]["sequence"]

refFile = os.path.basename(genome)
refDir = os.path.dirname(genome)
genome_prefix = os.path.splitext(refFile)[0]
genome_dict = refDir + '/' + genome_prefix + '.dict'
genome_fai = refDir + '/' + refFile + '.fai'

optional_output = list()
qc_output = list()
```

Use python package peds in TriosCompass

- https://github.com/jeremymcrae/peds
- https://github.com/NCI-CGR/TriosCompass_v2/blob/main/workflow/rules/pedigree.smk

```
import peds

### Define trios
ped_dir = config["ped_dir"]
ped_files = glob.glob(config["ped_dir"] + "/*.ped")

families = {}
for fn in ped_files:
    f=peds.open_ped(fn)[0]
    families[f.id]=f

fam_ids = list(families.keys())

child_ids = [[person.id for person in families[fid] if families[fid].get_father(person) ][
0] for fid in fam_ids]

final_subjs = list(set([p.id for f in fam_ids for p  in families[f] ]))

CHILD_DICT=dict(zip(fam_ids,child_ids))
```

Use input functions to take a variety of inputs

- https://snakemake.readthedocs.io/en/stable/snakefiles/rules.html#input-functions
- 3 likely bam sources in TriosCompass
  - fq2bam with fastq input
  - Bam input
    - From sample sheet directly
    - After reset RG
- Define input function ***get_bam***

```
rule hipstr:
    input:
        bams = [get_bam(subj) for subj in final_subjs],
        ref=genome,
        reg = output_dir+"/splitted_panel/hipstr_{chunk}.bed"
    output:
```

```
def get_bam(subj):
    return output_dir +"/fq2bam/{}.bam".format(subj)
```

```
if config["bam_input"]["reset_RG"]:
    rule replace_rg:
        input: lambda w: sample_bam_dict[w.subj]
        output:
            output_dir + "/fixed-rg/{subj}.bam"
        params:
            extra="--RGLB lib1 --RGPL illumina --RGPU {subj} --RGSM {subj} --CREATE_INDEX true "
        benchmark:
            output_dir +"/benchmark/replace_rg/{subj}.tsv"
        threads: config["threads"]["replace_rg"]
        wrapper:
            "v1.25.0/bio/picard/addorreplacereadgroups"

    def get_bam(subj):
        return output_dir +"/fixed-rg/{}.bam".format(subj)


else:
    def get_bam(subj):
        return sample_bam_dict[subj]
```

```
def get_bam_by_subj(wildcards):
    return get_bam(wildcards.subj)

def get_bams_by_family(wildcards):
    rv = [get_bam(person.id) for person in families[wildcards.fam]]
    return rv

def get_child_bam_by_family(wildcards):
    return get_bam(CHILD_DICT[wildcards.fam])
```

y

Summary

- The tips are valuable for workflow developers and users.
- Two essential messages from the 10 tips.
  - Use workflow management systems.
  - Documentation, documentation, and documentation!
- Pros of Snakemake
  - Easier for persons with python programing knowledge
    - Treat the workflow as White Box: more interoperable and reusable.
  - Other features: Benchmarking, modularization, cluster/cloud execution, report, etc.
- A community/forum may promote both development and application of FAIR workflows.
  - Developers ⇔ Developers, Developers ⇔ Users.

References

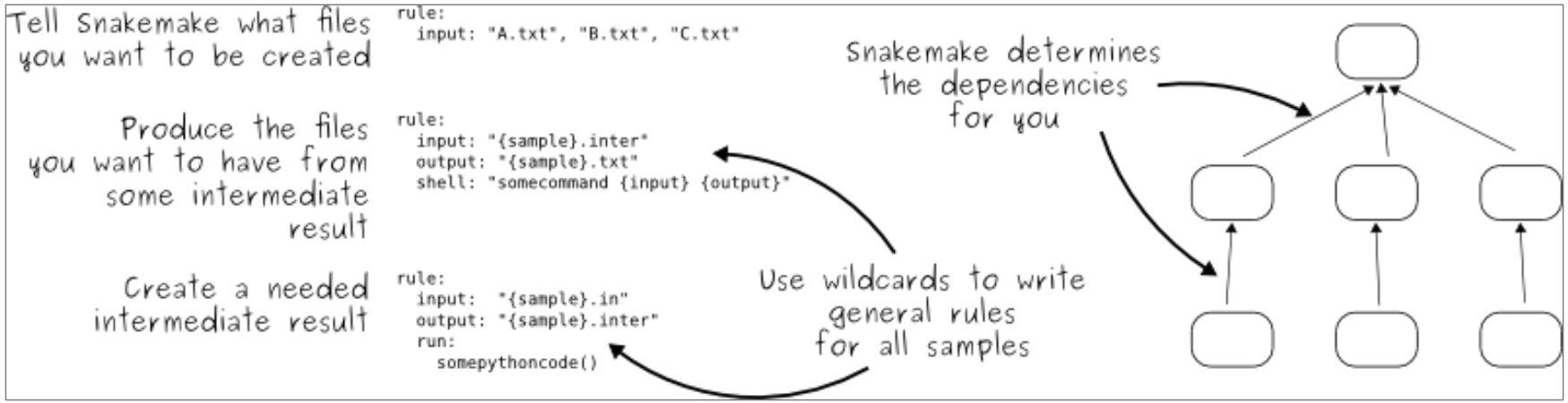
- Visser, C. de, Johansson, L. F., Kulkarni, P., Mei, H., Neerincx, P., Joeri van der Velde, K., Horvatovich, P., van Gool, A. J., Swertz, M. A., t Hoen, P. A. C., & Niehues, A. (2023). Ten quick tips for building FAIR workflows. *PLoS Computational Biology*, *19*(9 September), 1–13. https://doi.org/10.1371/journal.pcbi.1011369
- Wratten, L., Wilm, A., & Göke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature Methods*, *18*(10), 1161–1168. https://doi.org/10.1038/s41592-021-01254-9
- Wratten, L., Wilm, A., & Göke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nature Methods*, *18*(10), 1161–1168. https://doi.org/10.1038/s41592-021-01254-9
- Tutorials for building reproducible snakemake workflows
  - https://ifb-elixirfr.github.io/IFB-FAIR-bioinfo-training/assets/pdf/Session2020/session_03/03_workflow.pdf
  - https://sib-swiss.github.io/containers-snakemake-training/2023.10/
  - https://nbis-reproducible-research.readthedocs.io/en/course_1905/

Q & A

Draft slides

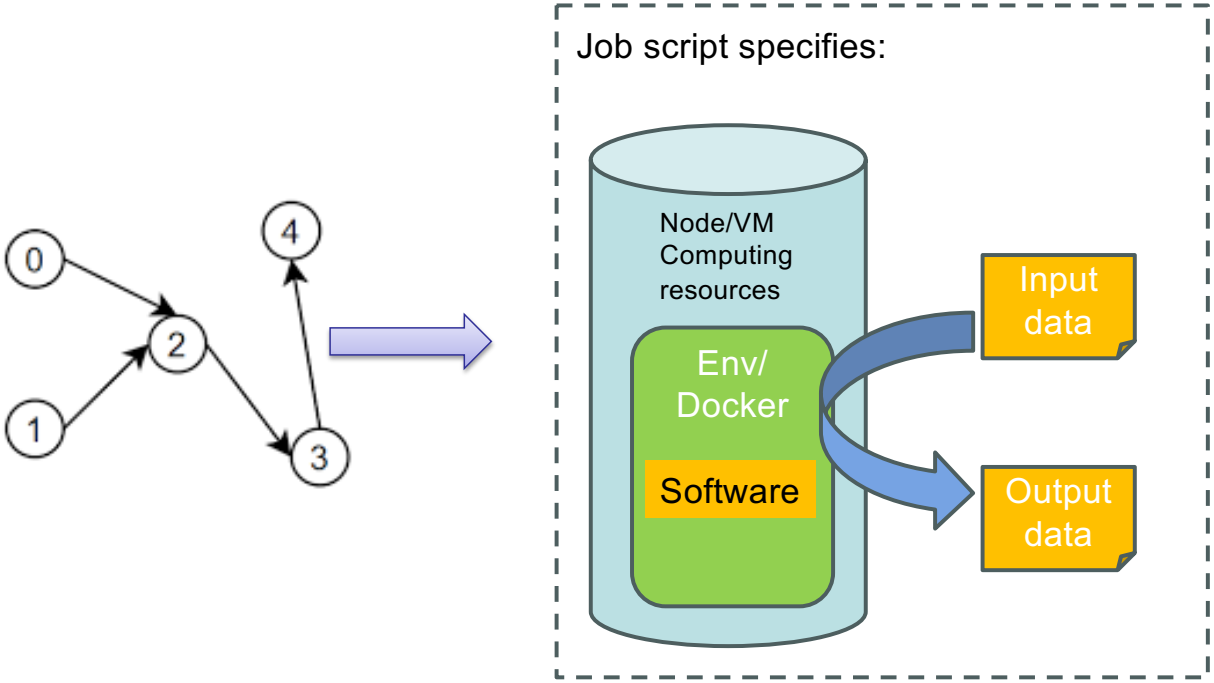
- TriosCompass FAIR
  - Configure files
    - Bam, fastq input files
  - Resource bundle
  - Modules
  - Report
  - Function as input
  - Use python in Snakemake
  - Data-dependent conditional execution

Snakemake workflow



https://snakemake.readthedocs.io/en/latest/project_info/faq.html

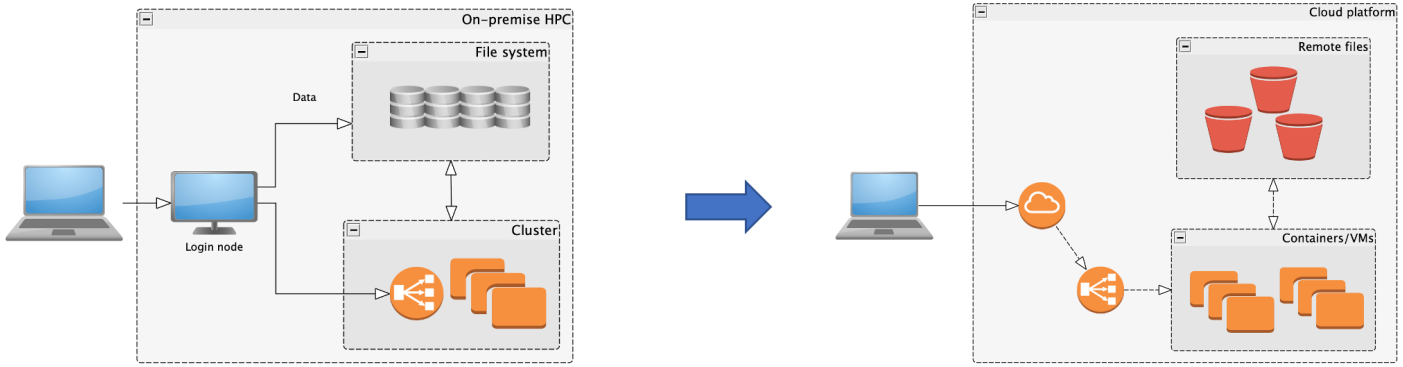
A workflow management system wraps a task into a job script



The jobscript generated by Snakemake

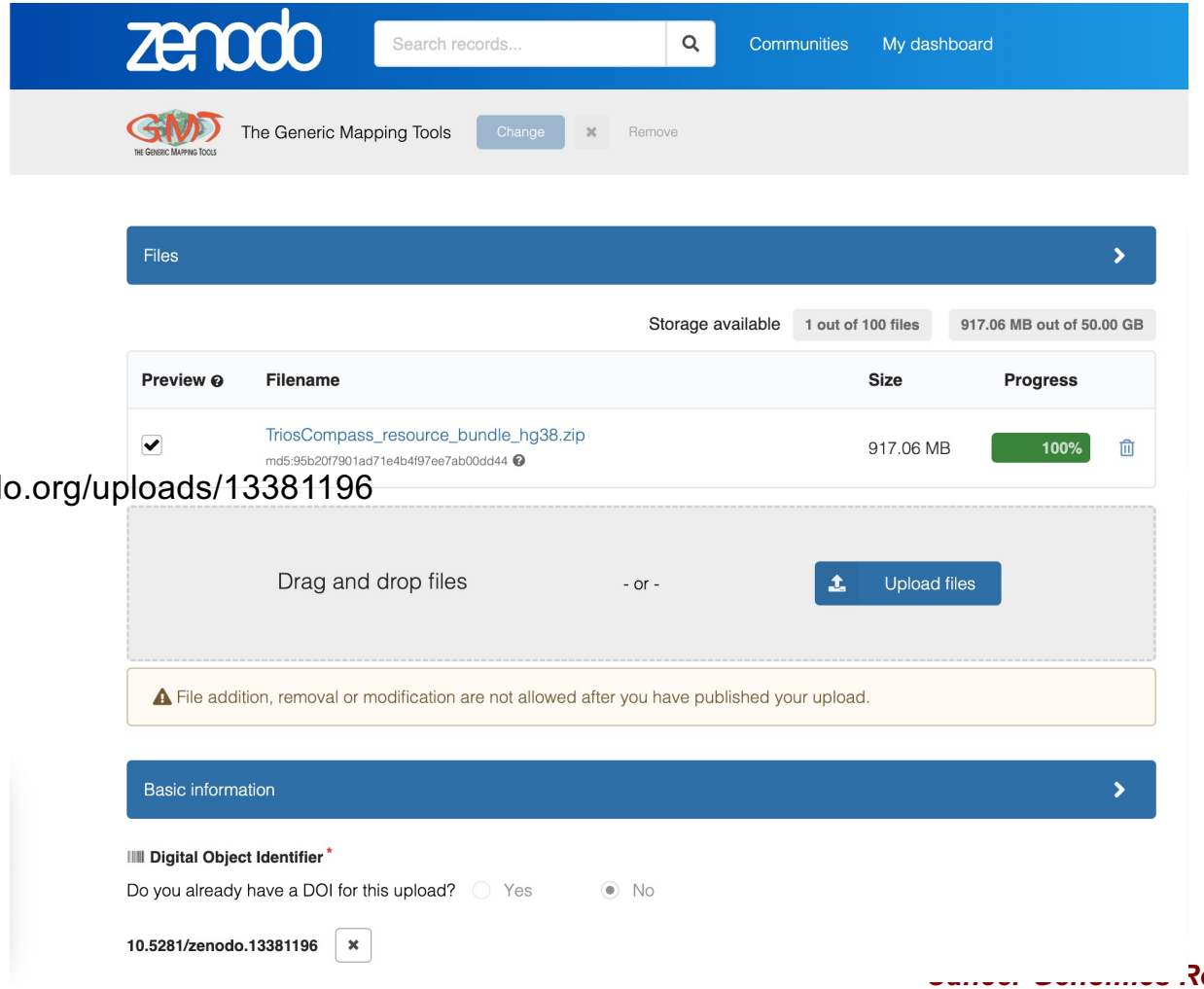
```
#!/bin/bash
# properties = {"type": "single", "rule": "deepvariant_pb", "local": false, "input": ["output/fixed-rg/SC260720_2.bam",
"ref/Homo_sapiens_assembly38.fasta"], "output": ["output/deepvariant_pb/SC260720_2.deepvariant.g.vcf",
"output/deepvariant_pb/SC260720_2.deepvariant.g.vcf.gz", "output/deepvariant_pb/SC260720_2.deepvariant.g.vcf.gz.tbi"], "wildcards":
{"subj": "SC260720_2"}, "params": {"singularity_cmd": " singularity exec --pwd /mnt --nv -B `pwd`:/mnt
docker://nvcr.io/nvidia/clara/clara-parabricks:4.0.0-1 "}, "log": [], "threads": 48, "resources": {"mem_mb": 180000, "disk_mb": 183656,
"tmpdir": "/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios/TMP", "runtime": "\"20h\"", "slurm": "gres=gpu:v100x:1", "partition": "gpu"}, "jobid":
25, "cluster": {}}
cd '/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios' && /data/zhuw10/conda_envs/TriosCompassV2/bin/python3.10 -m snakemake --snakefile
'/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios/TriosCompass_v2/workflow/Snakefile' 'output/deepvariant_pb/SC260720_2.deepvariant.g.vcf.gz' --
allowed-rules 'deepvariant_pb' --cores 'all' --attempt 1 --force-use-threads  --wait-for-files
'/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios/.snakemake/tmp.k3mjps6o' 'output/fixed-rg/SC260720_2.bam' 'ref/Homo_sapiens_assembly38.fasta'
'/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios/.snakemake/conda/65ed72cde0f39be10bd6ec1f2cfdc7a8' --force --keep-target-files --keep-remote --
max-inventory-time 0 --nocolor --notemp --no-hooks --nolock --ignore-incomplete --use-conda  --conda-frontend 'mamba' --conda-base-path
'/data/zhuw10/miniconda3' --use-singularity  --use-envmodules  --wrapper-prefix 'https://github.com/snakemake/snakemake-wrappers/raw/' --
configfiles '/gpfs/gsfs10/users/PLCO/zhuw10/8X2trios/TriosCompass_v2/config/8triosSplit_config.yaml' --printshellcmds  --latency-wait 60 -
-scheduler 'ilp' --scheduler-solver-path '/data/zhuw10/conda_envs/TriosCompassV2/bin' --set-resources 'bwa_index:mem_mb=16000'
'fastp:mem_mb=20000' 'fastp:runtime="10h"' 'fastqc:mem_mb=10000' 'fastq_screen:mem_mb=50000' 'fq2bam:mem_mb=40000' 'fq2bam:runtime="8h"'
'fq2bam:partition=gpu' 'fq2bam:slurm=gres=gpu:v100x:1' 'flagstat:mem_mb=80000' 'collectwgsmetrics:runtime="1d"'
'collectwgsmetrics:mem_mb=80000' 'collectmultiplemetrics:mem_mb=80000' 'collectmultiplemetrics:runtime="1d"'
'collectmultiplemetrics:partition=gpu' 'collectmultiplemetrics:slurm=gres=gpu:v100x:1' 'gatkhc_pb:mem_mb=200000' 'gatkhc_pb:runtime="8h"'
'gatkhc_pb:partition=gpu' 'gatkhc_pb:slurm=gres=gpu:v100x:1' 'gatk_combine_gvcf:mem_mb=10000' 'gatk_combine_gvcf:runtime="3h"'
'gatk_cgp:mem_mb=10000' 'gatk_genotype_gvcf_pb:partition=gpu' 'gatk_genotype_gvcf_pb:slurm=gres=gpu:v100x:1'
'deepvariant_pb:mem_mb=180000' 'deepvariant_pb:runtime="20h"' 'deepvariant_pb:slurm=gres=gpu:v100x:1' 'deepvariant_pb:partition=gpu'
'glnexus_dv:mem_mb=100000' 'glnexus_dv:runtime="1d"' 'call_JIGV:mem_mb=60000' 'replace_rg:mem_mb=40000' 'replace_rg:runtime="1d"'
'phase_trios:run_time="12h"' 'split_bed_hipstr:mem_mb=10000' 'hipstr:mem_mb=40000' 'hipstr:runtime="18h"'
'dumpstr_call_hipstr:mem_mb=40000' 'dumpstr_locus:mem_mb=40000' 'dumpstr_locus:runtime="20h"' 'monstr:mem_mb=40000' 'monstr:runtime="20h"'
'merge_monstr:mem_mb=20000' 'merge_monstr:runtime="20h"' 'monstr_filter:mem_mb=10000' 'manta_create_run_script:mem_mb=10000'
'manta_call:mem_mb=60000' 'manta_call:runtime="3d"' 'uniq_svid:mem_mb=40000' 'svimmer:runtime="24h"' 'svimmer:mem_mb=60000'
'graphtyper:runtime="72h"' 'graphtyper:mem_mb=120000' 'graphtyper_filter:mem_mb=20000' 'smoove:runtime="24h"' 'smoove:mem_mb=100000' --
default-resources 'mem_mb=2000' 'disk_mb=max(2*input.size_mb, 1000)' 'tmpdir=system_tmpdir' 'runtime="10:00:00"' --mode 2 && exit 0 ||
exit 1
```

From cluster execution to cloud execution



- Job scheduler
  - Slurm, SGE
- Shared file system
  - Input/output files
  - Software environment
    - Locally installed software
    - Using environment modules
    - Using conda
    - Using container

- Job scheduler/Executor
  - Kubernetes, Tibanna, Google-life-science
- Shared file system
  - Input/output files at Cloud Storage
  - Software environment
    - Using conda
    - Using container



https://zenodo.org/uploads/13381196