

Welcome!

Translating workflows into Nextflow with Janis

Bioinformatics Workflows

Chain multiple tasks to perform an analysis.

Has **inputs**, produces **outputs**.

Can be written in essentially any language.



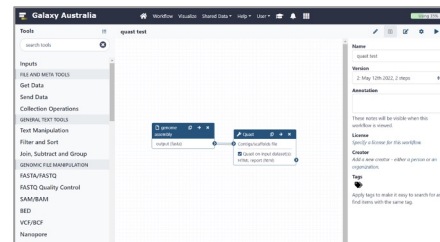
Choosing a Workflow Language

Each language has different *users & use-cases*

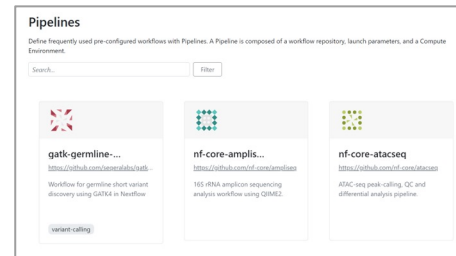
- Standard in field
- Properties / features
- Compute environments

Workflow languages are an *investment*

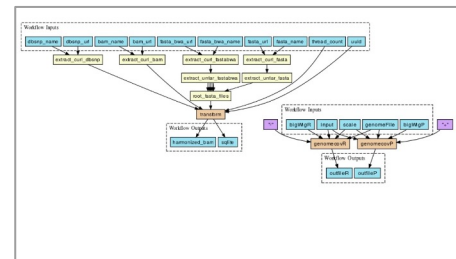
- Existing pool of **knowledge, talent** (researchers / staff), local legacy **workflows**
- Ongoing support & active community?
- Learning new language takes weeks / months



Galaxy Workflow Editor



Nextflow Tower

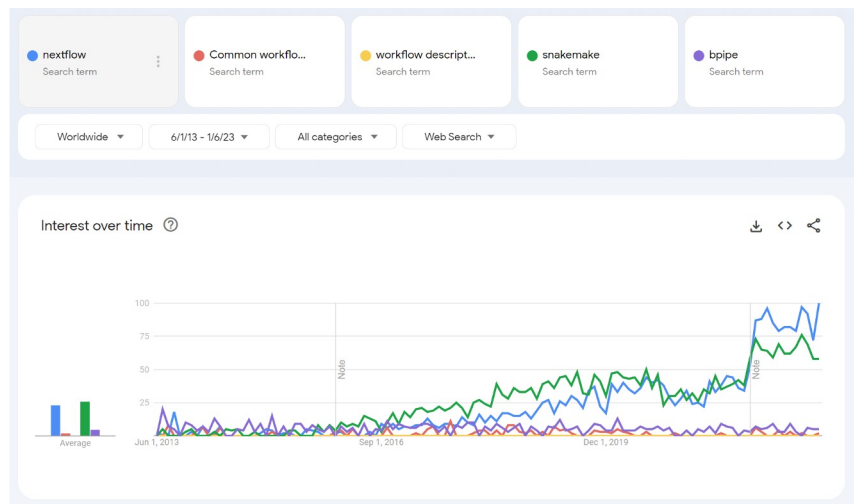


CWL viewer

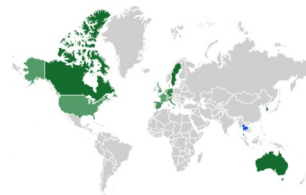
Choosing a Workflow Language

Workflow languages come and go over time.

- Can we be sure our investment will still be usable in 5 years?



2018 USA 



2023 USA 



Choosing a Workflow Language

Workflow space *highly divided*

No 'standard' workflow language

- New specs continually created
- Old specs become stale
- Can't know which spec will 'win'

Might not even be the correct question

- There are multiple programming languages
- Which will 'win'?

Existing Workflow systems

Michael R. Crusoe edited this page 7 days ago · 320 revisions

Permalink: <https://s.apache.org/existing-workflow-systems>

Cite as (update dates):

Peter Amstutz, Maxim Mikheev, Michael R. Crusoe, Nebojša Tijanić, Samuel Lampa, et al. (2022): **Existing Workflow systems**. *Common Workflow Language wiki*, GitHub. <https://s.apache.org/existing-workflow-systems> updated 2022-06-20, accessed 2022-06-20.

Computational Data Analysis Workflow Systems

An incomplete list

Please add new entries at the bottom.

In addition to this list, actively developed free/open-source systems should be registered at <https://workflows.community/systems>

See also: <https://github.com/pditommaso/awesome-pipeline>

1. Arvados - CWL-based distributed computing platform for data analysis on massive data sets. <https://arvados.org/>
<https://github.com/arvados/arvados>
2. Apache Taverna <http://www.taverna.org.uk/> <https://taverna.incubator.apache.org/>
3. Galaxy <http://galaxyproject.org/>
4. SHIWA <https://www.shiwa-workflow.eu/>

• • •

312. SimTool/Sim2Ls: Jupyter notebook-based pipelines of Simulation Tools for the HUBzero platform lead by nanoHUB
<https://github.com/hubzero/simtool> <https://simtool.readthedocs.io/> <https://doi.org/10.1371/journal.pone.0264492>
313. SideIO: A Side I/O system framework for hybrid scientific workflow _(no project/source code available_)
<https://doi.org/10.1016/j.jpdc.2016.07.001>
314. Flyte <https://flyte.org/>
315. StreamFlow <https://streamflow.di.unito.it/>
316. Jupyter Workflow <https://jupyter-workflow.di.unito.it/>
317. Nnodes: a simple workflow manager for Python functions and command line tools <https://github.com/icui/nnodes>

Janis

Portable Pipelines Project

Janis was developed as part of the Portable Pipelines Project, a collaboration between:

- Peter MacCallum Cancer Centre
- Melbourne Bioinformatics (University of Melbourne)
- Walter and Eliza Hall Institute of Medical Research (WEHI)
- Australian BioCommons

Initial aim was to develop shared cancer bioinformatics pipelines across Parkville biomedical institutes.

But the question back then in 2018 was; **which workflow language should they be written in?**
[CWL/ WDL/ snakemake/ nextflow]



Australian
BioCommons



Peter Mac
Peter MacCallum Cancer Centre
Victoria Australia



WEHI
brighter together



Melbourne Bioinformatics
BIOINFORMATICS + DATA SERVICES + INFRASTRUCTURE, FOR LIFE SCIENCES TODAY

Janis

Instead of committing to a single language...

Built a framework which can **transpile** to CWL or WDL.

Wanted **type-safety** due to importance of file-types in bioinformatics

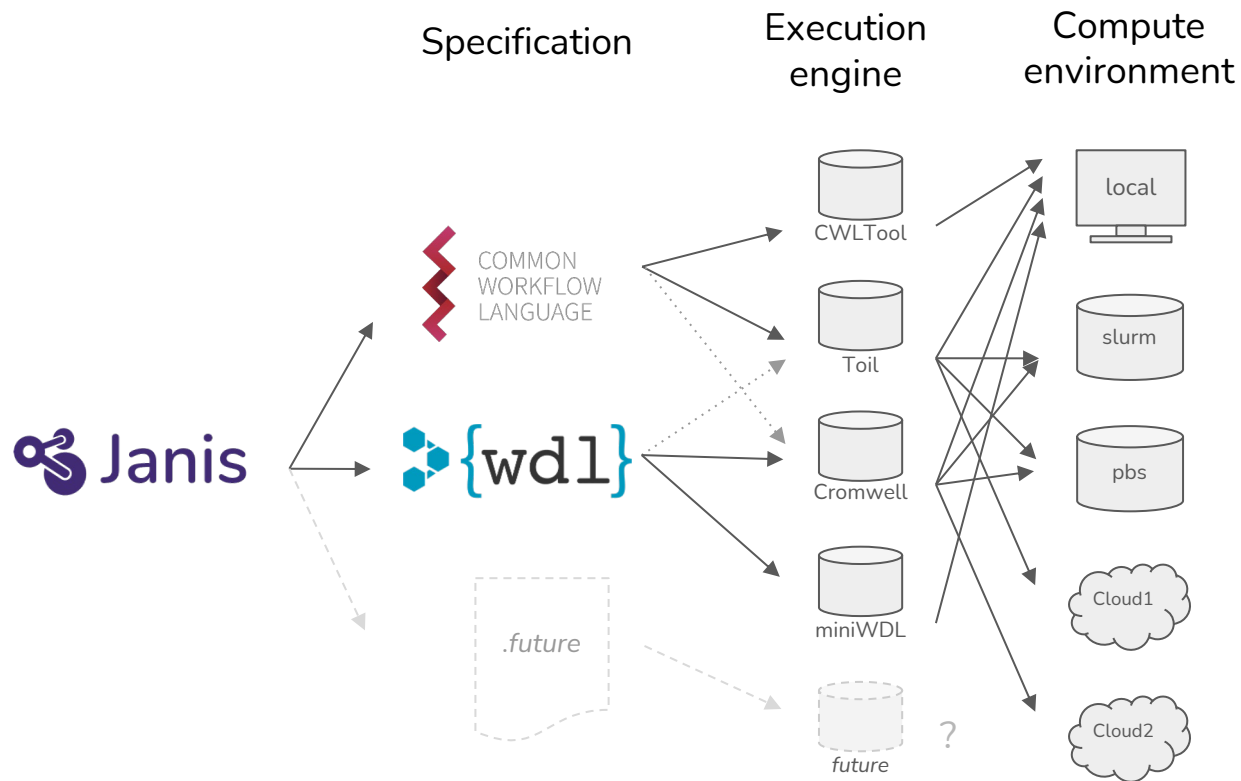
Would need to allow **flexible execution** across different compute environments of the respective stakeholders

If a new workflow system became popular in **future**, could add **transpilation** support for the format

```
class BcfToolsAnnotateBase(BcfToolsToolBase, ABC):  
  
    def tool(self):  
        return "bcftoolsAnnotate"  
  
    def base_command(self):  
        return ["bcftools", "annotate"]  
  
    def inputs(self):  
        return [  
            ToolInput("vcf", Vcf(), position=10),  
            ToolInput(  
                "outputFilename",  
                Filename(extension=".vcf"),  
                prefix="--output",  
                doc="[-o] see Common Options",  
            )  
        ]  
  
    def outputs(self):  
        return [ToolOutput("out", Vcf, glob=InputSelector("outputFilename"))]
```



First released in June 2019



Janis in 2021

Fast-forward to 2021.

Had built and run more than a dozen pipelines for real-world cancer analysis @ PeterMac.

Nextflow was booming. Name of the game had changed.

CWL, Snakemake and Nextflow all developed systems for flexible execution in different environments.

Reflected on our product & realised that going forward, **translation** was the unique value of Janis.

→ Started work on janis-translate

Janis in 2021

Fast-forward to 2021.

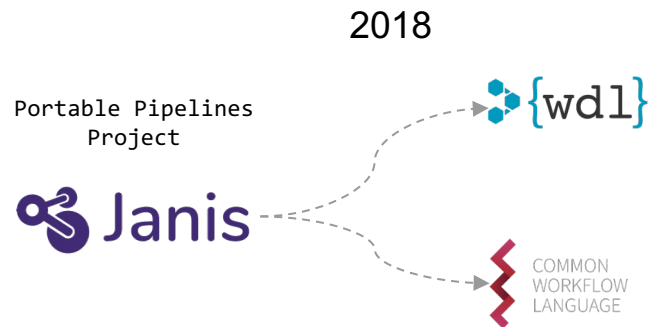
Had built and run more than a dozen pipelines for real-world cancer analysis @ PeterMac.

Nextflow was booming. Name of the game had changed.

CWL, Snakemake and Nextflow all developed systems for flexible execution in different environments.

Reflected on our product & realised that going forward, **translation** was the unique value of Janis.

→ Started work on janis-translate



Janis in 2021

Fast-forward to 2021.

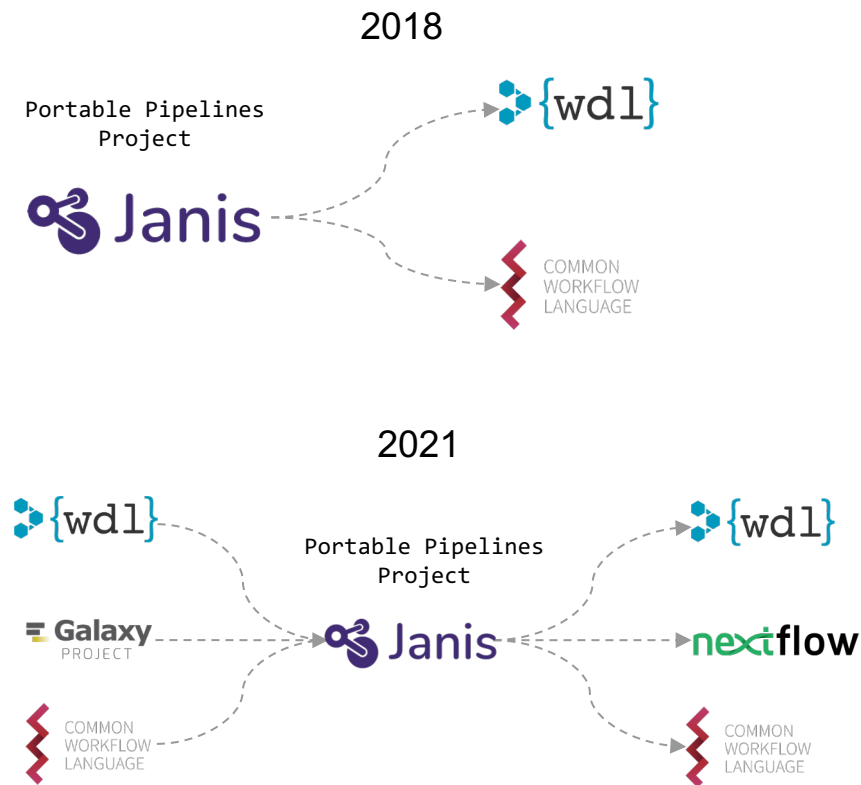
Had built and run more than a dozen pipelines for real-world cancer analysis @ PeterMac.

Nextflow was booming. Name of the game had changed.

CWL, Snakemake and Nextflow all developed systems for flexible execution in different environments.

Reflected on our product & realised that going forward, *translation* was the unique value of Janis.

→ Started work on janis-translate

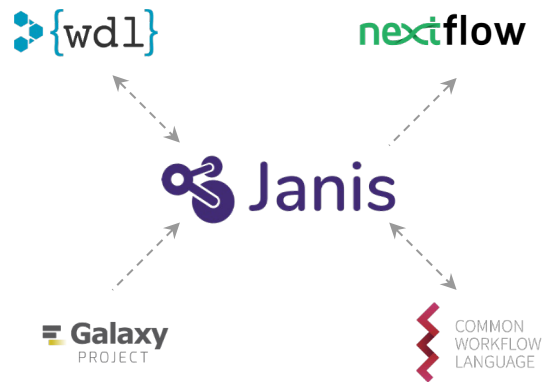


Janis Translate

This is our best attempt at realising our goal.

Janis-translate helps you migrate workflows / tools from one specification to another.

- Productivity tool to cut down boilerplate.
- Aims for human-readable translations.



Janis Translate

```
{
  "position": {
    "bottom": 365.32854738315707,
    "height": 190.4874725243797,
    "left": 1868.781888777551,
    "right": 1582.7818502688736,
    "top": 166.83387476897738,
    "width": 133.9999999942388,
    "x": 1386.7818887775517,
    "y": 166.83387476897738
  },
  "post_job_actions": [],
  "tool_id": "toolshed.g2.bx.psu.edu/repos/devteam/fastqc/fastqc/0.72/galaxy1",
  "tool_shed_repository": {
    "changeup_revision": "e782208dfeaa",
    "name": "fastqc",
    "owner": "devteam",
    "tool_shed": "toolshed.g2.bx.psu.edu"
  },
  "tool_state": {
    "adapters": {
      "\V_class_\V": "\RuntimeValue()", "\contains_\V": {
        "\V_class_\V": "\RuntimeValue()", "\V_supers_file": {
          "\V_class_\V": "\RuntimeValue()", "\V_supers_value": "\V_value_\V", "\V_items": {
            "\V_class_\V": "\RuntimeValue()", "\min_length": null, "\nngroup": "\V_false_\V", "\V_page_\V": null, "\V_recur_remap_job_id_\V": null, "\V_tool_version": "\V_7.2/galaxy1"
          }
        }
      },
    "type": "tool",
    "uuid": "48e16881-e8f6-4889-9502-c57aa0b1a36",
    "workflow_outputs": [
      {
        "label": null,
        "output_name": "html_file",
        "source": "\c822feef-2298-4f46-960c-8ebcafe6180a"
      },
      {
        "label": null,
        "output_name": "text_file",
        "source": "\480bae2e-8a9f-4d69-99fe-62c8c4b6c78f"
      }
    ]
  },
  "tags": [
    "assembly"
  ],
  "uuid": "\5999f86-86f9-4733-bd98-95918092856d"
}
```



```
# WORKFLOW DECLARATION
w = WorkflowBuilder(
  "unicycler_training_imported_from_uploaded_file",
  doc="Unicycler Assembly"
)

# INPUTS
w.input("in_forward_reads", File)
w.input("in_reverse_reads", File)
w.input("in_long_reads", File)

# STEPS
# -----
# STEPS: FASTQC
# -----
w.input("fastqc_adapters", Tabular)
w.input("fastqc_contaminants", Tabular)
w.input("fastqc_limits", TextFile)
w.input("fastqc_outdir", File)
w.input("fastqc_option_F", File)
w.step(
  "fastqc",
  fastqc(
    input_files.in_forward_reads, # positional [Fastq]
    adapters=fastqc_adapters, # RUNTIME VALUE - adapters [Tabular]
    contaminants=fastqc_contaminants, # RUNTIME VALUE - contaminants [Tabular]
    limits=fastqc_limits, # RUNTIME VALUE - limits [TextFile]
    option_f=fastqc_option_F, # RUNTIME VALUE - F [File]
   outdir=fastqc_outdir, # RUNTIME VALUE -outdir [File]
    nngroup=fastqc, # nngroup [Boolean]
    min_length=min, # min_length [Int]
    extract_rows, # --extract [Boolean] [GALAXY DEFAULT]
    quiet=true, # --quiet [Boolean] [GALAXY DEFAULT]
    kmers=k, # --kmers [Int] [GALAXY DEFAULT]
  )
)

# OUTPUTS
w.output(
  "fastqc_out_html_file",
  Html,
  source=w.fastqc, "out_html_file"
)
w.output(
  "fastqc_out_text_file",
  TextFile,
  source=w.fastqc, "out_text_file"
)

# METADATA
w.metadata(
  "fastqc",
  base_command["fastqc",
  "-a", "\adapters",
  "-c", "\contaminants",
  "-l", "\limits",
  "-o", "\outdir",
  "-F", "\option_f",
  "-n", "\nngroup",
  "-m", "\min_length",
  "--extract",
  "--quiet",
  "--kmers", "\kmers"
  ]
)
if __name__ == "__main__":
  FastQC().translate(
    "fastqc", "\toolshed.g2.bx.psu.edu/repos/devteam/fastqc/fastqc/0.72/galaxy1"
  )
```



```
nextflow-e8a614-651c2
include { FASTQC as FASTQC } from './modules/FASTQC'
include { FASTQC as FASTQC } from './modules/FASTQC'
include { UNICYCLER } from './modules/unicycler'
include { NANOPILOT } from './modules/nanopilot'
include { QUAST } from './modules/quast'
include { RUCSAC } from './modules/rucsaac'

// data which will be passed as variables
fastqc_adapters = file( params.fastqc_adapters )
fastqc_contaminants = file( params.fastqc_contaminants )
fastqc_limits = file( params.fastqc_limits )
fastqc_adapters = file( params.fastqc_adapters )
fastqc_contaminants = file( params.fastqc_contaminants )
fastqc_limits = file( params.fastqc_limits )
in_long = file( params.in_long )
in_short_r1 = file( params.in_short_r1 )
in_short_r2 = file( params.in_short_r2 )
unicycler_lr_align_contamination = file( params.unicycler_lr_align_contamination )
unicycler_rot_start_genes = file( params.unicycler_rot_start_genes )

workflow {
  FASTQC(
    in_short_r1, // input file
    fastqc_adapters, // adapters
    fastqc_contaminants, // contaminants
    fastqc_limits, // limits
    params.fastqc_format_string, // format_string
    params.fastqc_outdir, // outdir
  )
  FASTQC(
    in_short_r2, // input file
    fastqc_adapters, // adapters
    fastqc_contaminants, // contaminants
    fastqc_limits, // limits
    params.fastqc_format_string, // format_string
    params.fastqc_outdir, // outdir
  )
  UNICYCLER(
    in_short_r1, // paired_unpaired_fastq_inpath1
    in_short_r2, // paired_unpaired_fastq_inpath2
    in_long, // long
    unicycler_lr_align_contamination, // lr_align_contamination
    unicycler_rot_start_genes, // rotation_start_genes
    params.unicycler_pilon_path, // pilon_path
  )
}
```



```
janis translate --from galaxy --to nextflow ./unicycler_wf.ga
```

Janis Tools (translated)

Nextflow Processes (translated)

Today's Session

Hands-on experience with workflow migration using Janis Translate.

- CWL → Nextflow
- Galaxy → Nextflow

Aim is to:

- Get familiar with Janis translation output
- Run the translations with test data
- Do some proper Nextflow debugging to bring translations to a finished, runnable state

(we are here) →

Description	Est. Duration	Real time (finish)
Housekeeping	5 min	1:10pm
Intro to the session & Janis	10 min	1:20pm
CWL → NEXTFLOW TRANSLATIONS		
Intro to CWL	10 min	1:30pm
Samtools Flagstat Tool	20 min	1:50pm
GATK HaplotypeCaller Tool	20 min	2:10pm
Align Sort Markdup Workflow	20 min	2:30pm
Break	10 min	2:40pm
GALAXY → NEXTFLOW TRANSLATIONS		
Intro to Galaxy	10 min	2:50pm
Samtools Flagstat Tool	10 min	3:00pm
Limma Voom Tool	20 min	3:20pm
RNA-Seq reads to counts workflow	30 min	3:50pm
Wrap up	10 min	4:00pm
EXTENSION		
Participant workflow translations	30 min	4:30pm

Acknowledgements

Current Janis team:

- Grace Hall (Janis)
- Richard Lupat (Peter Mac)
- Daniel Park (UoM)
- Bernie Pope (UoM)
- Lisa Phippard (BioCommons)
- Evan Thomas (WEHI)

Previous team members of Portable Pipelines Project:

- Michael Franklin (Janis)
- Juny Kesumadewi (Janis)
- Jiaan Yu (Peter Mac)
- Xinzhe Li (Peter Mac)

Initial proof-of-concept

- Mohammad Bhuyan (WEHI)

Early adopters / community support

- Sebastian Hollizeck (Peter Mac)
- Tom Conway (Peter Mac)
- Kersten Bruer (DKFZ)
- Michael Crusoe (CWL)

Workshop Setup:

- Matthew Downton (NCI)
- Melissa Burke (BioCommons)
- Alex Ip (AARNET)
- Georgie Samaha (SIH)
- Audrey Stott (Pawsey)

