

Deploying Nextflow on CHTC

A Case Study for Serving Biology's Software Needs with AI-Assisted Engineering

Nick Minor • June 10th, 2026

HTC26

- ▶ The sciences' unmet need for software
- ▶ Two problems for researchers making software
- ▶ An attempt (by me) to solve those problems with AI
- ▶ A lukewarm but positive-leaning take on AI

*The sciences' have enormous
unmet need for software*

WILDLIFE BIOLOGY



What my friends think I do.



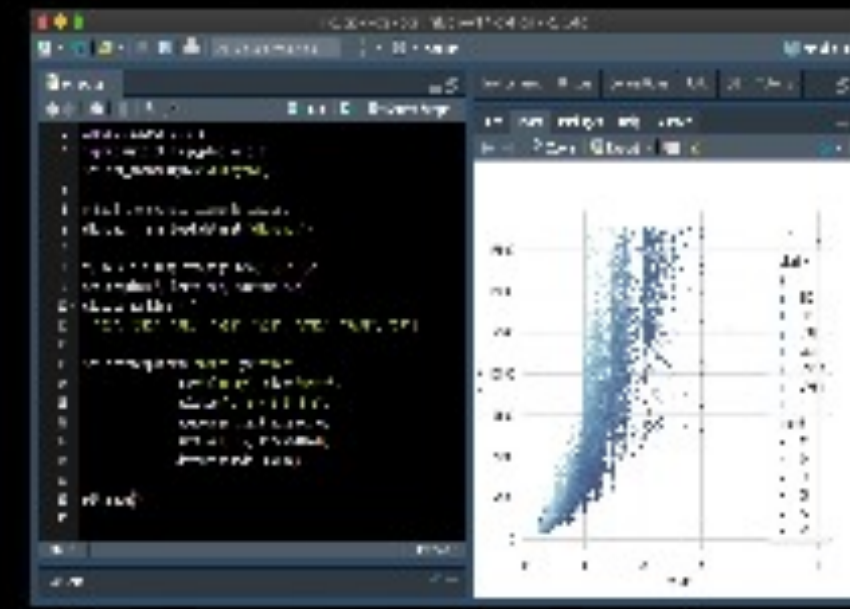
What my mom thinks I do.



What I think I do.



What society thinks I do.



What I really do.

imgflip.com

WeKnowMemes

HTC26

Researchers have at least two problems with doing software engineering:

1. a *time* problem
2. a *perceived agency* problem

CHANGE YOUR MINDSET!



I can't do it

It's too
difficult

It's all my
fault

There's a bug
in prod

Time to write
unit tests



USE AI



USE AI



USE AI



USE AI

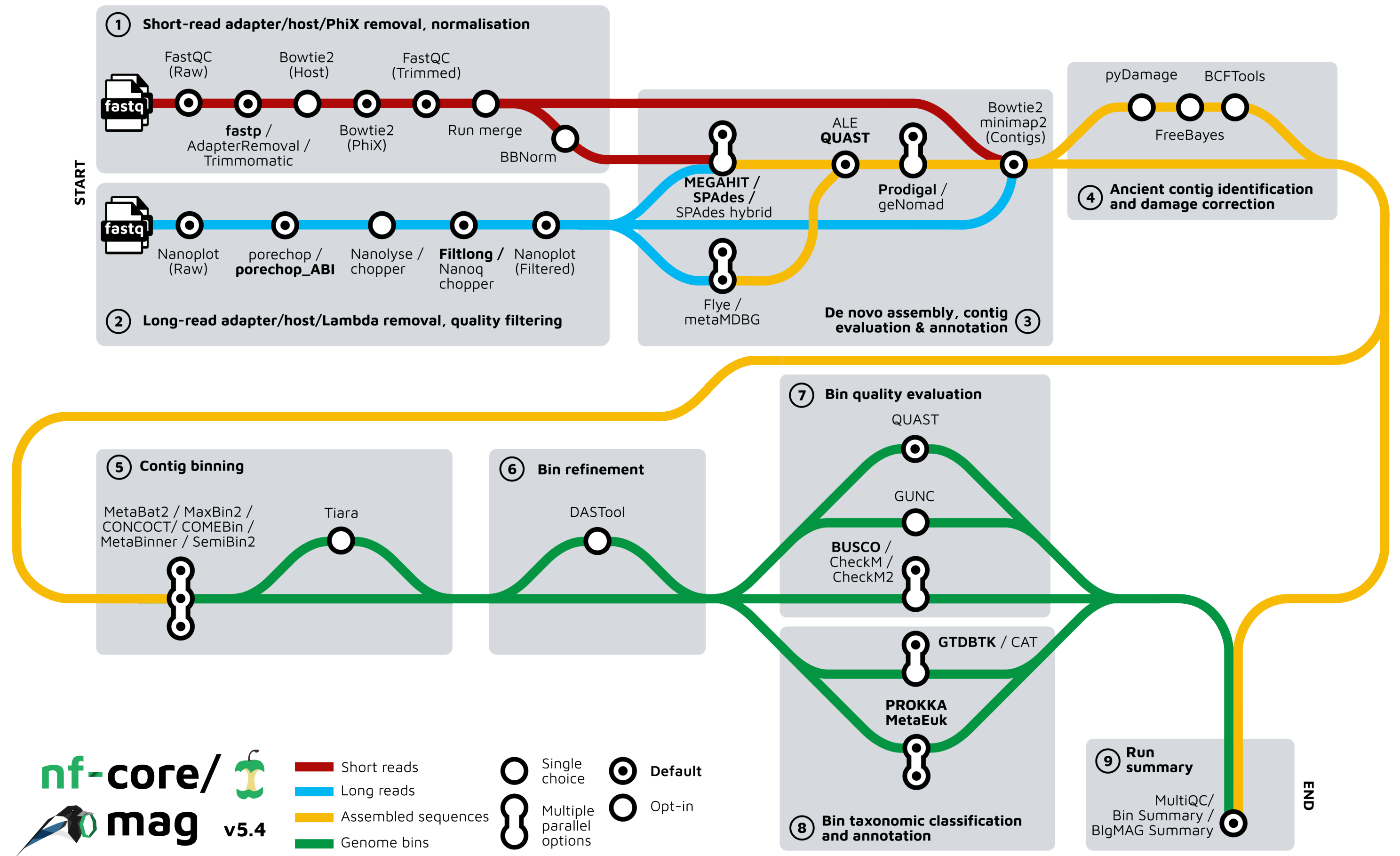


USE AI



nextflow

nf-core 



HTC26



Pipeline	Domain	GitHub Stars	GitHub Contributors
nf-core/rnaseq	RNA sequencing analysis	1,309	100
nf-core/sarek	germline or somatic mutation calling	574	81
nf-core/scrnaseq	Single-cell RNA expression	330	49
nf-core/mag	De novo assembly from metagenomes	302	48
nf-core/ampliseq	PCR amplicon sequencing	250	50



dholab / nvd



README GPL-3.0 license



NVD: Identify Low-Abundance Viruses in Big Environmental Metagenomes

Release passing release v3.0.0 license GPL-3.0

Overview

NVD is a Nextflow pipeline focused on finding human-infecting viruses in environmental metagenome samples. It leverages a fast *in silico* enrichment approach together with de novo assembly and BLAST verification to maximize recall and precision. It supports Illumina short-read as well as Nanopore long-read inputs and offers a configurable deduplication approach tuned for shotgun sequencing datasets.



nrminor / oneroof



README MIT license



OneRoof: Base-, Variant-, and Consensus-calling under One Proverbial Roof.

nextflow DSL2 ≥22.10.1 run with docker run with singularity Docker CI passing

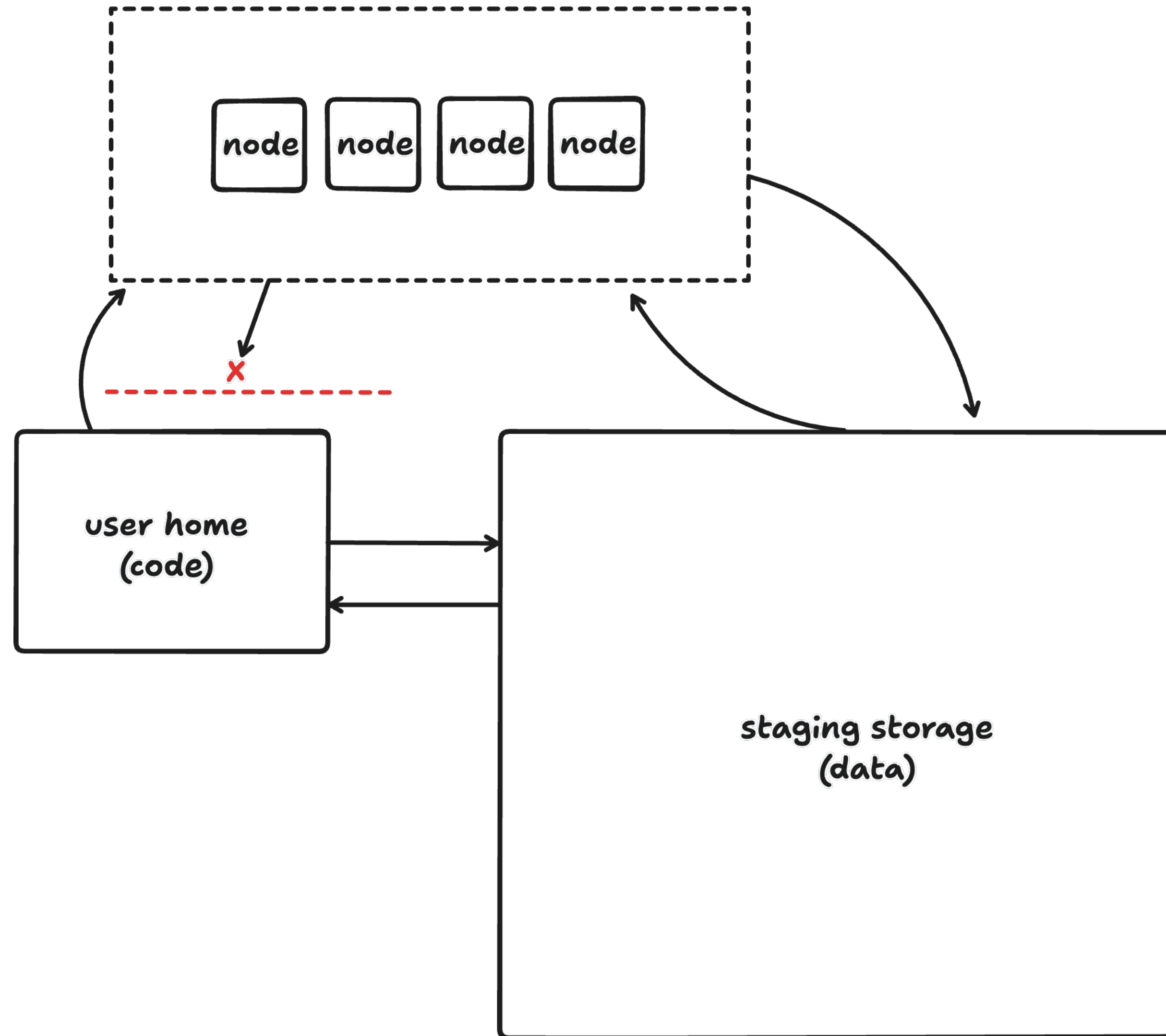
- [Overview](#)
 - [Analysis Presets](#)
- [Quick Start](#)
- [Detailed Setup Instructions](#)
 - [Configuration](#)
 - [Developer Setup](#)
 - [OneRoof CLI \(Power Users\)](#)
- [Testing](#)
 - [Running Tests](#)
 - [Test Structure](#)
- [Further Documentation](#)
- [Contributing](#)
- [Is it any good?](#)
- [Citation](#)

Overview

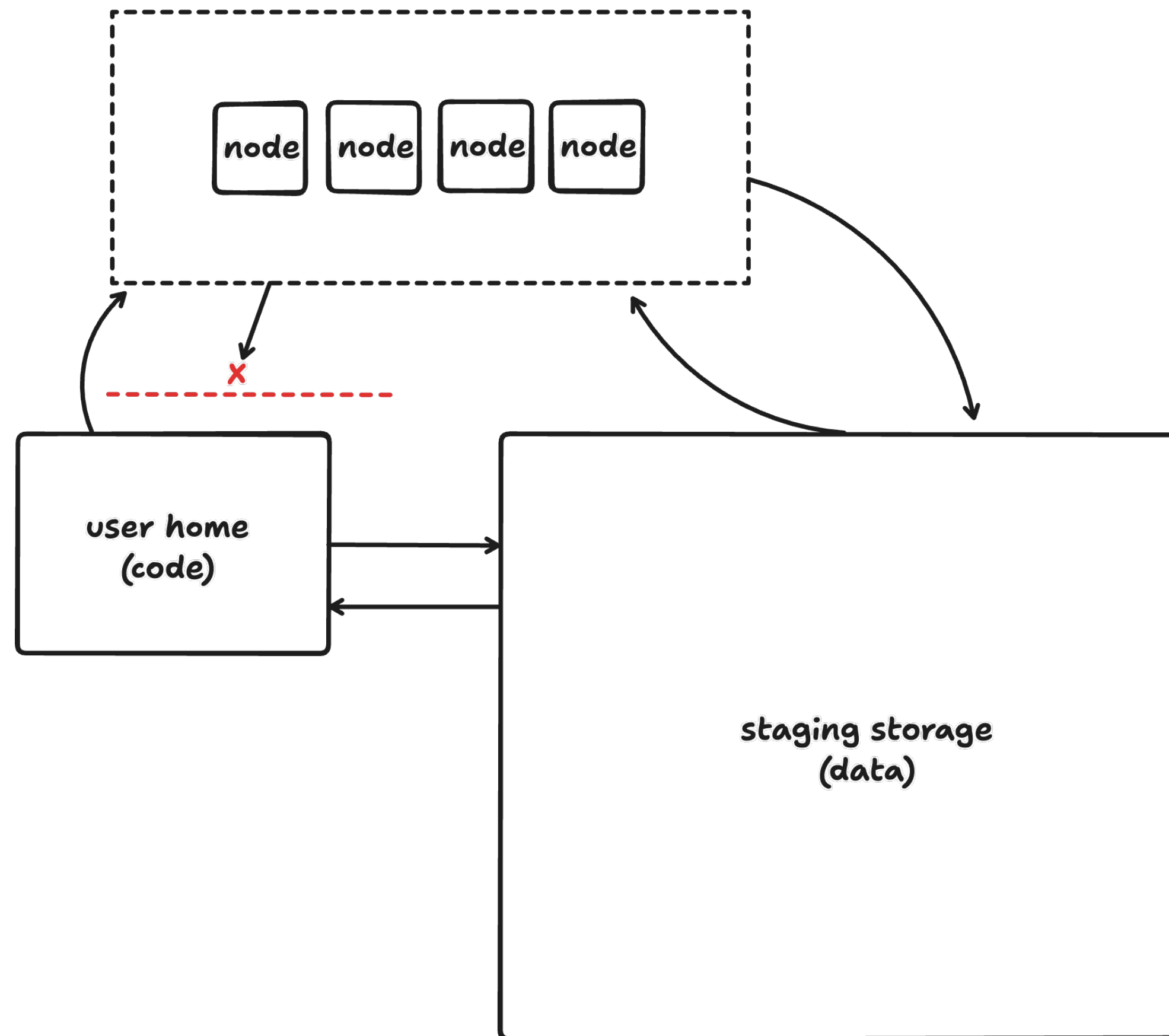
oneroof is a [Nextflow](#) pipeline designed to take a common series of bioinformatic tasks (see below) and put them under "one roof". We mean this literally as well as figuratively: the pipeline will perform certain tasks best when run on networked devices in the same building.

HTC26

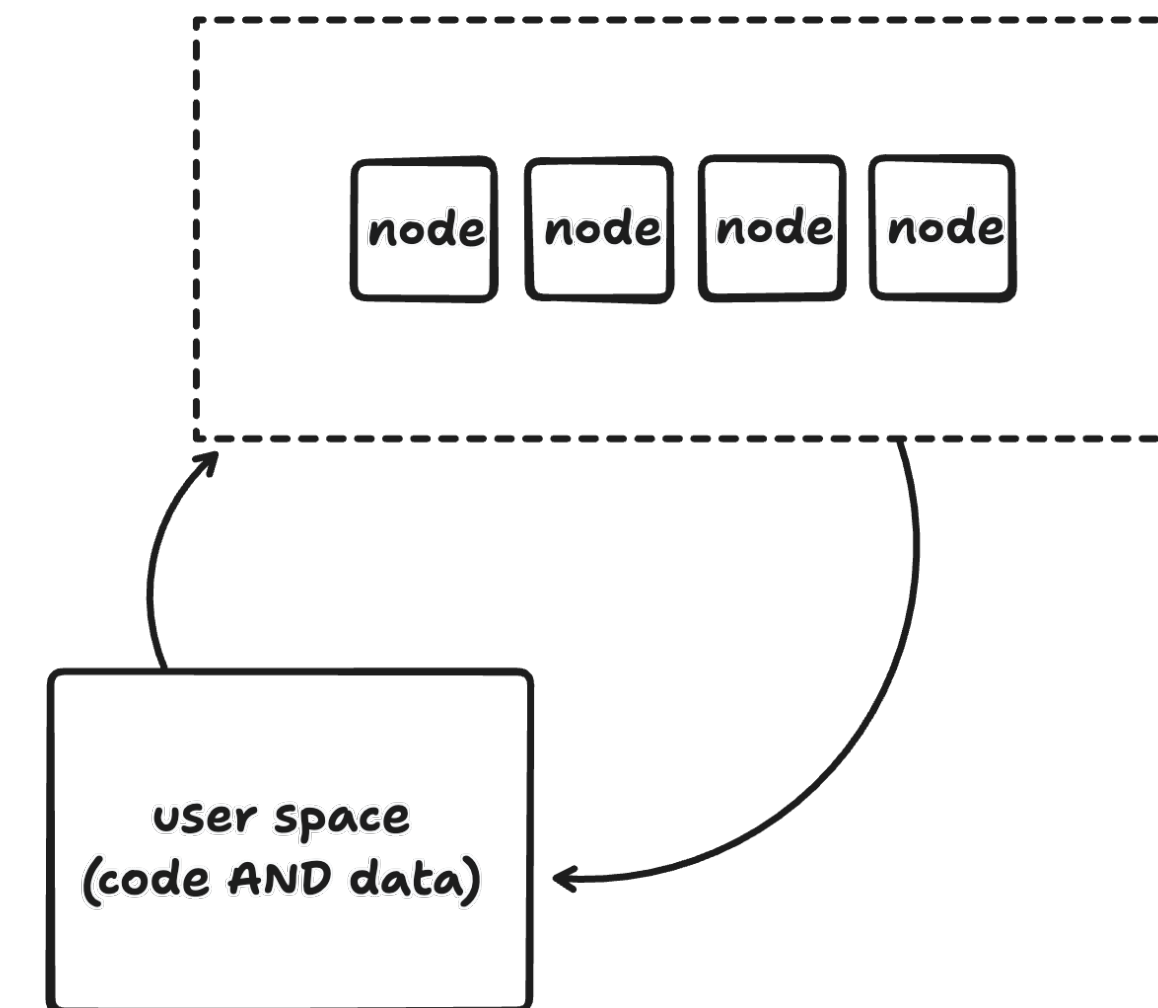
The Big Problem:
*Nextflow wants to get cozy with HTC...and
HTC's not interested*

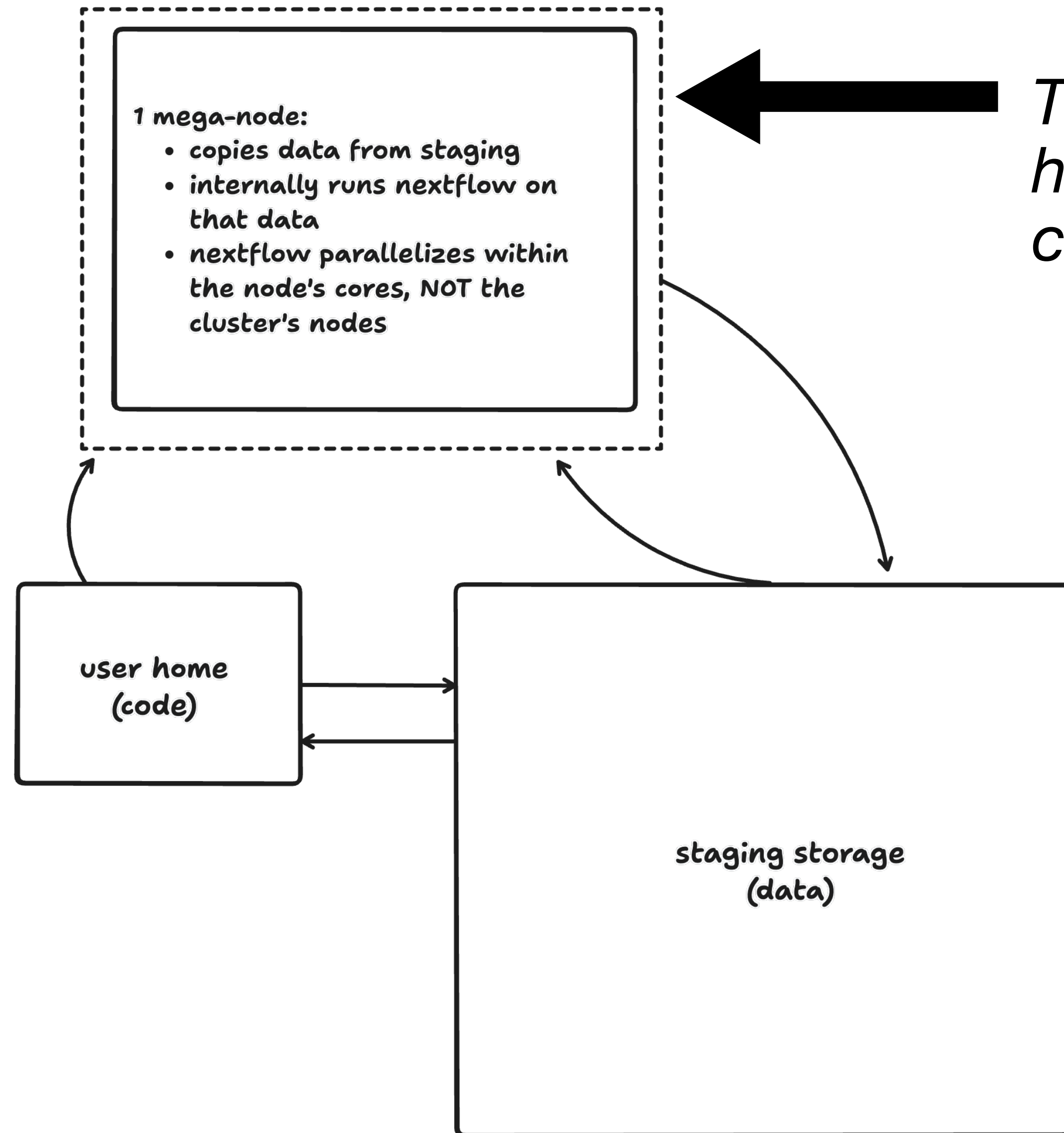


CHTC HTC Filesystem:



What Nextflow wants💔:





This is not the path to happiness on the HTC cluster

The Big Problem Refined:
*Nextflow-submitted nodes need access to
files in user home AND staging...*

The Big Problem Refined:
*...and to fix this, I need to modify the
source code for Nextflow itself*

Now *I* have our two problems with doing software engineering:

1. a *time* problem

- ▶ *I don't have the time to learn Groovy and learn the Nextflow codebase and development setup on top of the rest of my job.*

2. a *perceived agency* problem

- ▶ *I don't have the **ability** to learn Groovy and learn the Nextflow codebase and development setup on top of the rest of my job.*
- ▶ *Plus, good luck convincing the Nextflow maintainers to merge my PR!*

HTC26

project: pomarine %1
project: abagnale %2
project: nuclease %3
project: genotype %4
project: spillover %5
+

```

priority-0-projec... | START-HERE-intern...
36 chunk sorting rather than packed-key radix sorting. A6 added a private build-time
35 compatibility guard so `RadixThenRefine<N>` is accepted by the bio builder only
34 with matching forward `SequenceOrder<N>`. A7 strengthened pipeline evidence across
33 real spill boundaries with expected output computed from the intended comparators.
32
31 The semantic choice is no longer open: accelerated sequence order means packed
30 key first, then raw sequence bytes, then quality bytes. Name and length key
29 compatibility are tested. Exact raw sequence sorting is available through
28 `sort_by_sequence_exact()`. Reverse order is comparator-based by default rather
27 than silently radix-accelerated. Priority 0 is now sufficiently pinned to support
26 the next architecture workstream.
25
24 Do not use this priority section as an implementation spec. The detailed record
23 and reports live in `.agents/priority-0-projection-law-commit-plan.md`. Future
22 projection-law work should be treated as follow-up confidence work, such as
21 property tests with fixed small buffers that guarantee spill, not as a blocker for
20 the run-lifecycle refactor route below.
19
18 ## Priority 1: take the easy private cleanup, but do not confuse it for architecture
17
16 There is some stale private complexity that can be removed early and safely. For
15 example, core `Sorter::finish(self)` consumes the sorter, so `dedup: Option<D>`
14 and panic documentation about calling `finish` twice look like historical
13 scaffolding rather than live design. Removing that kind of noise is worthwhile
12 because it makes later diffs easier to review.
11
10 The `MergeConfig` buffering fields need a more careful audit. They are public
9 configuration, and bio currently accounts for `write_buffer_bytes` in arena
8 budget estimates, but core merge does not appear to consistently route these
7 fields through actual buffered I/O. If those fields are no-ops, they are
6 misleading. If they are meant to be real, wiring them through may affect codec
5 writer bounds and should not be treated as a trivial cleanup.
4
3 The reviewers were strongly aligned on one point: avoid helper traits or type
2 aliases that merely hide repeated bounds. A helper is useful only when it names a
1 real module concept, such as "can write this borrowed window item with keys." If
647 it only shortens signatures, it risks making compiler errors more mysterious
1 without improving locality.
2
3 ## Priority 2: deepen core around run accumulation
4
5 The main maintainability failure is that `spillover-bio` had to become a second
6 external sorter to support arena-backed ingest. Bio should own records, arenas,
7 handles, borrowed views, and genomics-specific memory estimates. Core should own
8 the lifecycle of sorted runs.
9
10 Today the embryo exists in `RunMerger::spill_sorted` and
11 `RunMerger::spill_sorted_with_keys`, but that interface is too low-level. It can
12 write one run, but bio still owns the run set and final merge flow:
13
14 ```text
15 bio arena backend today
16
17 arena + handles
18 |
19 | v
20 resolve views
21 |
22 | v
23 sort views
24 |
25 | v
26 RunMerger::spill_sorted(_with_keys)
27 |
28 | v
29 bio pushes SortedRun into Vec
30 |
31 | v
32 bio later builds RunMerger again and merges
33 ```
18 NORMAL 3cfd5 START-HERE-internal-complexity-reduction-audit.md

```

Hmm. Remember that always doing the fast easy thing can actually degrade a codebase when many such decisions build up in the long-term. What would be the best thing for the long-term health of the project here? What treats the disease rather than suppresses symptoms?

Thought: Considering user questions · 2.8s

I need to answer the user's reflective question in a "grill-me" style, which feels like a challenge. They're asking whether it's better to make fields real or remove them for long-term benefit. I think I should discuss this based on evidence, even though I may not have tools for support. It could be useful to ask a follow-up question to encourage more thought from the user, so let's keep that in mind too!

For long-term health, I would not treat `MergeConfig` as "just docs." The disease is that `MergeConfig` currently exposes a public resource-control contract that the core engine does not seem to actually own.

That matters because `spillover` is fundamentally a memory-bounded external sorter. If it has public fields named `read_buffer_bytes` and `write_buffer_bytes`, users will reasonably believe those fields constrain or tune I/O buffering. If they are no-ops in core, then the API is teaching users a false model of the library. Documentation that says "these are only estimates" might reduce embarrassment, but it would still leave the design incoherent.

My recommended long-term answer: **make buffering an owned core responsibility, unless code exploration proves the codec layer must own it instead.**

Concretely, I'd frame the next Priority 1 work as:

Current smell:

```

MergeConfig promises read/write buffer sizing
├── max_fan_in: honored by core
├── temp_dir: honored by core
├── read_buffer_bytes: apparently not honored
└── write_buffer_bytes: apparently not honored, but bio budgets for it

```

Healthy target:

```

either core owns buffer sizing end-to-end
or the public API stops pretending it does

```

The "treat the disease" path is probably:

1. Clarify the ownership question first.

Is buffering supposed to be:

 - core-level I/O wrapping around files,
 - codec-level responsibility,
 - or not part of this crate's promise?
2. If core owns it, make the fields real.

That likely means core opens temp files and wraps them in `BufReader<File>` / `BufWriter<File>` sized from `MergeConfig`, then passes those wrappers into codec cursor/writer constructors.
3. If codec owns it, change the public contract.

Then `MergeConfig::{read_buffer_bytes, write_buffer_bytes}` are the wrong API surface, or at least wrongly documented. But because they're public, removal/deprecation is a bigger semver/API discussion.

My bias: core should own it, because buffering temp-run files is part of external-sort lifecycle, not domain serialization. Codecs should know how to encode/decode records; they should not have to know the global merge fan-in, temp-run lifecycle, or user memory budget.

Next grill-me question: are we willing to let the next commit touch the core codec/cursor/writer generic bounds if that is what making `MergeConfig` real requires?

My recommended answer: **yes, but only after a small design sketch**, because this may turn a "cleanup" into an API-shape change.

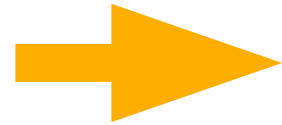
Careful · GPT-5.5 · 14.3s

Careful · GPT-5.5 OpenAI · low

647:61 76% 845L markdown
101.5K (25%) ctrl+p commands



dholab / nvd



```
NEXTFLOW ~ version 25.10.2
OSPool executor plugin started
Launching `/home/nrminor/.nvd/v2.4.0/main.nf` [cheeky_wright] DSL2 - revision: a76b1c2c37

executor > local (1), ospool (129)
[-] process > GATHER_READS:FETCH_FASTQ
[80/7c92cb] process > GATHER_READS:INTERLEAVE_PAIRS (School029-20260105_S32_L004) [100%] 61 of 61, cached: 61 ✓
[b5/068e1d] process > PREPROCESS_READS:DEDUP_WITH_CLUMPIFY (School025-20260105_S28_L004) [100%] 61 of 61, cached: 61 ✓
[3c/35e27d] process > PREPROCESS_READS:SCRUB_HOST_READS (School026-20260105_S29_L004) [ 96%] 59 of 61, cached: 59
[ff/94ac53] process > PREPROCESS_READS:REPAIR_PAIRS (School020-20260105_S23_L004) [100%] 59 of 59, cached: 56
[c2/2a94f8] process > STAT_BLAST_WORKFLOW:CHECK_RUN_STATE (1) [100%] 1 of 1 ✓
[6c/b4f673] process > STAT_BLAST_WORKFLOW:VALIDATE_LK_BLAST:VALIDATE_BLAST_HITS_LIST [100%] 1 of 1, cached: 1 ✓
[b4/3b7d81] process > STAT_BLAST_WORKFLOW:VALIDATE_LK_BLAST:VALIDATE_BLAST_FASTA_LIST [100%] 1 of 1, cached: 1 ✓
[0e/05249e] process > STAT_BLAST_WORKFLOW:COUNT_READS (School020-20260105_S23_L004) [100%] 59 of 59, cached: 56
[66/7c016d] process > STAT_BLAST_WORKFLOW:PREPROCESS_CONTIGS:EXTRACT_HUMAN_VIRUS_READS (School030-20260105_S33_L004) [ 61%] 36 of 59, cached: 19
[40/e3f82b] process > STAT_BLAST_WORKFLOW:PREPROCESS_CONTIGS:RUN_SPADES (School030-20260105_S33_L004) [ 26%] 8 of 30, cached: 4
[62/8130d7] process > STAT_BLAST_WORKFLOW:PREPROCESS_CONTIGS:MASK_LOW_COMPLEXITY (NTC-Duck_S46_L004) [ 75%] 6 of 8
[30/389bf9] process > STAT_BLAST_WORKFLOW:PREPROCESS_CONTIGS:FILTER_SHORT_CONTIGS (Buff-W05_S51_L004) [100%] 6 of 6
[6a/7638b6] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:CLASSIFY_CONTIGS_FIRST_PASS (Buff-W05_S51_L004) [100%] 5 of 5
[f1/cd94e6] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:GENERATE_CONTIGS_TAXA_LIST (Buff-G01_S52_L004) [100%] 5 of 5
[8e/c0be67] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:CLASSIFY_CONTIGS_SECOND_PASS (Buff-G01_S52_L004) [100%] 5 of 5
[1b/b74e38] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:GENERATE_STAT_CONTIG_REPORT (Buff-G01_S52_L004) [100%] 5 of 5
[7e/724463] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:IDENTIFY_HUMAN_VIRUS_FAMILY_CONTIGS (Clinic002-20251103_S2_L004) [100%] 5 of 5
[1a/9b3c73] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:EXTRACT_HUMAN_VIRUS_CONTIGS (Clinic002-20251103_S2_L004) [100%] 4 of 4
[8c/32ed63] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:MAP_READS_TO_CONTIGS (Clinic002-20251103_S2_L004) [100%] 4 of 4
[9d/5a9a4e] process > STAT_BLAST_WORKFLOW:EXTRACT_HUMAN_VIRUSES:COUNT_MAPPED_READS (Clinic002-20251103_S2_L004) [100%] 4 of 4
[7a/eb3f09] process > STAT_BLAST_WORKFLOW:CLASSIFY_WITH_MEGABLAST:MEGABLAST (Clinic002-20251103_S2_L004) [ 0%] 0 of 4
```

HTC26



nrminor / nf-ospool



README Apache-2.0 license

nf-ospool

Nextflow plugin for the Open Science Pool (OSPool) HTCondor environment.

Overview

This plugin provides a custom Nextflow executor (`ospool`) specifically designed for the Open Science Pool HTCondor environment at UW-Madison CHTC and similar infrastructures. It addresses unique constraints in OSPool environments:

- **Submit location restrictions:** HTCondor policy blocks `condor_submit` from running in certain directories (e.g., `/staging/`)
- **Non-shared filesystem:** Compute nodes use isolated sandboxes with different filesystem topology than submit nodes
- **Path normalization:** Symlinked paths must be normalized for container bind mounts
- **Directory staging:** Project directories and secrets may not be accessible from compute nodes

seqera × nextflow

Plugins

Modules

nf-ospool @ 0.1.0

Provider: University of Wisconsin - Madison

Claimed: Jan 11

Description: Nextflow executor for the Open Science Pool HTCondor environment

Project URL: <https://github.com/nrminor/nf-ospool>

Latest version: 0.1.0

Total downloads: 127 [View trends](#)

Overall benefits?

- ▶ Nodes match in minutes instead of hours -> faster overall turnaround
- ▶ Nodes use less resources -> good citizen effects
- ▶ Full Nextflow resumeability -> better error resilience
- ▶ Nextflow UX mirrors desktop UX -> easier onboarding

Reprise!

Researchers have at least two problems with doing software engineering:

1. a *time* problem
2. a *perceived agency* problem

Reprise!

Researchers have at least two problems with doing software engineering:

1. *a time* problem
 - Time was saved *but was still nontrivial*
2. *a perceived agency* problem

Reprise!

Researchers have at least two problems with doing software engineering:

1. *a time* problem

- Time was saved *but was still nontrivial*

2. *a perceived agency* problem

- AI-assisted development has radically expanded the software problems my research group is comfortable with taking on

My lukewarm take on AI-assisted development:

Easier problems resulting from knowledge gaps are now much easier to solve and we should be eager to solve them.

Hard problems are still hard.

Open Questions

These questions are socio-technical; they involve the technology themselves as well as the social norms around it

- what should our standards be for reviewing code in tech that's outside our expertise?
- what does code ownership and maintenance look like for AI-generated code?
- what are the replicability/reproducibility implications?
- should more software become interactive with a graphical user interface instead of being chiefly command-line-driven?

Thanks and Acknowledgements

Research software dev requires many guinea pigs!

