

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

Table of contents

Overview	1
Quick Start	2
Detailed Setup Instructions	3
Configuration	3
Developer Setup	5
Contributing	6
Is it any good?	6
Citation	6

Overview

oneroof is a pipeline designed to take a common series of bioinformatic tasks (see below) and put them under “one roof”. We mean this quite literally: the pipeline will perform at its best when run on networked devices in the same building.

Note

Please note that oneroof is still under active development. Even the name of the pipeline is a prototype name that may change at any time. That said, the pipeline at its current state is relatively stable and working well for us in container-based and container-less use cases. The most active area of change at this stage is feature-additions, though in the near future, some of the more convoluted parts of the pipeline’s primer-handling will be replaced with a Rust-based toolkit under active development by our team, so stay tuned!

oneroof was originally developed in the early stages of the United States Bovine Highly Pathogenic Avian Influenza (HPAI) outbreak of 2024, when we wanted one, configurable, easy-to-run pipeline that would do all of the following:

1. Handle super-accuracy basecalling with GPU acceleration on pod5-formatted Nanopore signal files, working on GCP or AWS if need be.
2. Demultiplex BAM-formatted reads that come out of basecalling.
3. Perform the above two steps as signal files become available, either locally or remotely via an SSH client.
4. Accept raw read BAMs or FASTQs if basecalling and demultiplexing have already been performed elsewhere.
5. Accept paired Illumina reads in addition to Nanopore reads.

6. Use forward and reverse primer sequences to select only those reads that represent complete amplicons.
7. Trim away primers—and also any bases that are upstream of the forward primer or downstream of the reverse primer.
8. Align to a custom reference with the proper presets for the provided data.
9. Call variants and consensus sequences with appropriate settings for the provided data.
10. Perform tree building with `nextclade`, quality introspection with `multiQC`, and error correction based on the input sequence platform.

Though many excellent pipelines currently exist, e.g. `nf-core/viralrecon`, `epi2me-labs/wf-amplicon`, and `nf-core/nanoseq`, none of these pipelines quite handled all of the above. `oneroof` seeks to handle these requirements while remaining highly configurable for users, highly modular for developers, and easy to control in the command line for both.

Overall, `oneroof` can be summarized as a variant-calling pipeline written in and managed by Nextflow. Its software dependencies are provided through containers or through an environment assembled by `pixi`. To run it on your own Nanopore pod5s, simply run something like:

```
nextflow run nrminor/oneroof \
--pod5_dir my_pod5_dir \
--primer_bed my_primers.bed \
--refseq my_ref.fasta \
--ref_gbk my_ref.gbk \
--kit "SQK-NBD114-24"
```

These are the core elements required to run on Nanopore data: a directory of pod5 files, a BED file of primer coordinates, a reference sequence in FASTA and Genbank format, and the Nanopore barcoding kit used.

And for Illumina paired-end reads, it's even simpler:

```
nextflow run nrminor/oneroof \
--illumina_fastq_dir my_illumina_reads/
```

Quick Start

For most users, `oneroof` will have two core requirements: The Docker container engine, available [here](#), and Nextflow, available [here](#). For users interested in super-accuracy basecalling Nanopore signal files, an on-board GPU supported by the Dorado basecaller is also required.

All remaining software dependencies will be supplied through the pipeline's Docker image, which will be pulled and used to launch containers automatically.

From there, the pipeline's three data dependencies are sequence data in BAM, FASTQ, or POD5 format, a BED file of primer coordinates, and a reference sequence in FASTA and Genbank format. For Nanopore data, a barcoding kit identifier is also required. Simply plug in these files to a command like the above and hit enter!

Detailed Setup Instructions

Configuration

Most users should configure oneroof through the command line via the following parameters:

Note that oneroof checks for how to gather data in a particular order for Nanopore data, which mirrors the order they are listed in the table above:

1. First, oneroof checks if the user has supplied a remote location with `--remote_pod5_location`, at which point it will launch the file watcher module and begin transferring and basecalling batches of pod5 files. The size of these batches can be controlled with `--pod5_batch_size`, which defaults to basecalling batches of 100 pod5s.
2. If no remote directory is provided, oneroof checks for a local pod5 directory from `--pod5_dir`.
3. If the user doesn't provide a local `--pod5_dir`, oneroof assumes that pre-basecalled BAMs or FASTQs are being provided. These can either be watched for with the directory from `--precalled_staging`, or run immediately with the directory from `--prepped_data`.

Developer Setup

oneroof depends on software packages supplied through various conda registries as well as through PyPI, the Python Package Index. To unify these various channels, we used the relatively new pixi package and environment manager, which stores all dependencies from both locations in the file `pyproject.toml`.

To reproduce the environment required by this pipeline, make sure you are on a Mac with an x86_64 (Intel) processor, a 64-bit linux machine, or a 64-bit Windows machine using Windows Subsystem for Linux (Apple Silicon coming soon!). Then, to reproduce the environment, install pixi with:

```
curl -fsSL https://pixi.sh/install.sh | bash
```

Download the pipeline with:

```
git clone https://github.com/nrminor/oneroof.git && cd oneroof
```

And then open a pixi subshell within your terminal with:

```
pixi shell
```

As long as you are using a supported system, the pipeline should run within that subshell. You can also run the pipeline within that subshell without containers using the “containerless” profile:

```
nextflow run . \
-profile containerless \
--pod5_dir my_pod5_dir \
--primer_bed my_primers.bed \
--refseq my_ref.fasta \
--ref_gbk my_ref.gbk \
--kit "SQK-NBD114-24"
```

Especially on Macs, this will reduce the overhead of using the Docker Virtual Machine and allow the pipeline to invoke tools installed directly within the local project environment.

Note also that more information on the repo's files is available in our developer guide.

Contributing

Contributions, feature requests, improvement suggestions, and bug reports via GitHub issues are all welcome! For more information on how to work with the project and what all the repo files are, see our developer guide.

Is it any good?

Yes.

Citation

Coming soon!