OneRoof Pipeline Architecture

OneRoof Development Team

2025-07-08

Table of contents

This document provides a comprehensive map of the OneRoof Nextflow pipeline structure, including workflow dependencies, data flow, and critical testing points.

1 Main Workflow Entry Point

1.1 main.nf

- **Purpose**: Central orchestrator that routes to platform-specific workflows
- **Key Functions**:
  - Platform detection (Nanopore vs Illumina) based on input parameters
  - Input channel initialization for all required files
  - Workflow selection and invocation
  - Email notification on completion

Input Channels:

- `ch_primer_bed`: Optional primer BED file
- `ch_refseq`: Required reference FASTA
- `ch_ref_gbk`: Optional GenBank file for annotation
- `ch_contam_fasta`: Optional contamination sequences
- `ch_metagenomics_ref`: Optional metagenomics reference

- `ch_snpeff_config`: Optional SnpEff configuration
- `ch_primer_tsv`: Optional primer TSV file
- `ch_sylph_tax_db`: Optional Sylph taxonomy database

2 Platform-Specific Workflows

2.1 NANOPORE Workflow (workflows/nanopore.nf)

Workflow DAG:

```
graph TD
    A[GATHER_NANOPORE] --> B[PRIMER_HANDLING]
    B --> C[ALIGNMENT]
    C --> D[CONSENSUS]
    C --> E[VARIANTS]
    C --> F[HAPLOTYPING]
    C --> G[METAGENOMICS]
    D --> H[PHYLO]
    E --> I[SLACK_ALERT]

    style B fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5 5
```

> **Note**
>
> Dashed boxes indicate optional workflow components

Key Parameters:

- `platform = "ont"`
- `min_variant_frequency = 0.2`
- `min_qual = 10`

2.2 ILLUMINA Workflow (workflows/illumina.nf)

Workflow DAG:

```
graph TD
    A[GATHER_ILLUMINA] --> B[ILLUMINA_CORRECTION]
    B --> C[PRIMER_HANDLING]
    C --> D[ALIGNMENT]
```

```
    D --> E[CONSENSUS]
    D --> F[VARIANTS]
    D --> G[PHYLO]
    D --> H[METAGENOMICS]
    E --> I[SLACK_ALERT]
    F --> I

    style C fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5 5
```

Key Parameters:

- `platform = "illumina"`
- `min_variant_frequency = 0.05`
- `min_qual = 20`

3 Sub-workflows and Dependencies

3.1 Data Gathering Workflows

3.1.1 GATHER_NANOPORE (subworkflows/gather_nanopore.nf)

Purpose: Handle multiple Nanopore input formats

Input Options:

1. Remote POD5 monitoring (`remote_pod5_location`)
2. Local POD5 directory (`pod5_dir`)
3. Pre-called staging directory (`precalled_staging`)
4. Pre-processed data directory (`prepped_data`)

Process Flow:

```
graph LR
    A[POD5 Input] --> B[DOWNLOAD_MODELS]
    B --> C[BASECALL]
    C --> D[MERGE_BAMS]
    D --> E[DEMULTIPLEX]

    F[Pre-called Input] --> G[VALIDATE_NANOPORE]
    E --> G
    G --> H[FILTER_WITH_CHOPPER]
    H --> I[COMPRESS_TO_SORTED_FASTA]
```

```
    I --> J[FAIDX]
    J --> K[EARLY_RASUSA_READ_DOWNSAMPLING]
```

3.1.2 GATHER_ILLUMINA (subworkflows/gather_illumina.nf)

Purpose: Process paired-end Illumina FASTQ files

Process Flow:

```
graph LR
    A[Paired FASTQs] --> B[VALIDATE_ILLUMINA]
    B --> C[MERGE_READ_PAIRS]
```

3.2 Processing Workflows

3.2.1 ILLUMINA_CORRECTION (subworkflows/illumina_correction.nf)

Purpose: Quality control and decontamination for Illumina reads

Process Flow:

```
graph TD
    A[CORRECT_WITH_FASTP] --> B[DECONTAMINATE]
    B --> C[FASTQC]
    C --> D[MULTIQC]
    B --> E[COMPRESS_TO_SORTED_FASTA]
    E --> F[FAIDX]
    F --> G[EARLY_RASUSA_READ_DOWNSAMPLING]

    style B fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5 5
```

3.2.2 PRIMER_HANDLING (subworkflows/primer_handling.nf)

Purpose: Validate primers and extract complete amplicons

Input Options:

1. Primer BED file
2. Primer TSV file

Process Flow:

```
graph TD
    A[ORIENT_READS] --> B[GET_PRIMER_PATTERNS]
    B --> C[FIND_COMPLETE_AMPLICONS]
    B --> D[TRIM_ENDS_TO_PRIMERS]
    D --> E[PER_AMPLICON_FILTERS]
    E --> F[MERGE_BY_SAMPLE]
```

3.2.3 ALIGNMENT (subworkflows/alignment.nf)

Purpose: Map reads to reference and generate coverage statistics

Process Flow:

```
graph TD
    A[ALIGN_WITH_PRESET] --> B[CONVERT_AND_SORT]
    B --> C[RASUSA_ALN_DOWNSAMPLING]
    C --> D[SORT_BAM]
    D --> E[INDEX]
    E --> F[MOSDEPTH]
    F --> G[PLOT_COVERAGE]
    G --> H[COVERAGE_SUMMARY]
```

3.2.4 VARIANTS (subworkflows/variant_calling.nf)

Purpose: Call and annotate variants

Process Flow:

```
graph TD
    A[CALL_VARIANTS] --> B[CONVERT_TO_VCF]
    B --> C[ANNOTATE_VCF]
    C --> D[EXTRACT_FIELDS]
    D --> E[MERGE_VCF_FILES]
```

3.2.5 CONSENSUS (subworkflows/consensus_calling.nf)

Purpose: Generate consensus sequences

Process Flow:

```
graph LR
    A[CALL_CONSENSUS] --> B[CONCAT]
```

3.3 Optional Feature Workflows

3.3.1 PHYLO (subworkflows/phylo.nf)

Purpose: Phylogenetic analysis using Nextclade

Process Flow:

```
graph LR
    A[CHECK_DATASET] --> B[DOWNLOAD_DATASET]
    B --> C[RUN_NEXTCLADE]
```

3.3.2 METAGENOMICS (subworkflows/metagenomics.nf)

Purpose: Metagenomic classification using Sylph

Process Flow:

```
graph TD
    A[SKETCH_DATABASE_KMERS] --> C[CLASSIFY_SAMPLE]
    B[SKETCH_SAMPLE_KMERS] --> C
    C --> D[OVERLAY_TAXONOMY]
    D --> E[MERGE_TAXONOMY]

    style D fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5 5
    style E fill:#f9f,stroke:#333,stroke-width:2px,stroke-dasharray: 5 5
```

3.3.3 HAPLOTYPING (subworkflows/haplotyping.nf)

Purpose: Viral haplotype reconstruction (Nanopore only)

Condition: Number of reference sequences equals number of amplicons

4 Key Modules/Processes

4.1 Critical Processes for Testing

4.1.1 dorado.nf

- `DOWNLOAD_MODELS`: Model caching
- `BASECALL`: GPU-based basecalling
- `DEMULTIPLEX`: Barcode demultiplexing

4.1.2 minimap2.nf

- `ALIGN_WITH_PRESET`: Platform-specific alignment

4.1.3 ivar.nf

- `CALL_VARIANTS`: Variant detection
- `CALL_CONSENSUS`: Consensus generation
- `CONVERT_TO_VCF`: Format conversion

4.1.4 samtools.nf

- `CONVERT_AND_SORT`: BAM processing
- `INDEX`: BAM indexing

4.1.5 validate.nf

- `VALIDATE_NANOPORE`: Input validation
- `VALIDATE_ILLUMINA`: Paired-end validation
- `VALIDATE_PRIMER_BED`: Primer validation

5 Critical Testing Paths

5.1 Minimal Test Path (No Primers)

1. **Nanopore**: POD5/FASTQ → Basecall → Align → Consensus/Variants
2. **Illumina**: Paired FASTQs → Merge → Align → Consensus/Variants

5.2 Full Test Path (With Primers)

1. Input validation
2. Primer handling and amplicon extraction
3. Alignment and coverage analysis
4. Variant calling and annotation
5. Consensus generation
6. Optional: Phylogenetics, metagenomics, haplotyping

5.3 Key Input Requirements

Minimal Requirements

- Reference FASTA (`--refseq`)
- Sequencing data:
  - Nanopore: POD5 files + kit name OR pre-called BAM/FASTQ
  - Illumina: Paired-end FASTQ directory

Full Feature Requirements

- Primer BED file (`--primer_bed`) or TSV (`--primer_tsv`)
- Reference GenBank (`--ref_gbk`) for annotation
- SnpEff config for variant annotation
- Contamination FASTA for decontamination
- Metagenomics database for classification

6 Output Structure

6.1 Nanopore Output Tree

```
nanopore/
  01_basecalled_demuxed/
  02_primer_handling/
  03_alignments/
  04_consensus_seqs/
  05_variants/
  06_QC/
  07_phylo/
```

```
  metagenomics/
  haplotyping/
```

6.2 Illumina Output Tree

```
illumina/
  01_merged_reads/
  02_primer_handling/
  03_alignments/
  04_consensus_seqs/
  05_variants/
  06_QC/
  07_phylo/
  metagenomics/
```

7 Configuration and Parameters

7.1 Platform-Specific Defaults

Parameter	Nanopore	Illumina
`min_variant_frequency`	0.2	0.05
`min_qual`	10	20
Alignment preset	`map-ont`	`sr`

7.2 Resource Management

- `pod5_batch_size`: Controls GPU memory usage
- `downsample_to`: Coverage depth limiting
- `basecall_max`: Parallel basecalling instances
- `low_memory`: Resource-constrained mode

7.3 Key Process Labels

- `big_mem`: Memory-intensive processes (variant calling, consensus)
- GPU requirements: Dorado basecalling

8 Error Handling and Retry Strategy

Most processes implement:

```
errorStrategy { task.attempt < 3 ? 'retry' : 'ignore' }
maxRetries 2
```

This provides resilience against transient failures while preventing infinite loops.

9 Testing Considerations

9.1 Critical Validation Points

1. Input file validation (exists, correct format)
2. Primer validation (coordinates, sequences)
3. Read count filtering (empty file handling)
4. Platform-specific parameter application
5. Optional workflow branching

9.2 Edge Cases to Test

- Empty input files
- No reads passing filters
- Missing optional inputs
- Primer mismatches
- Low coverage regions
- Multiple reference sequences
- Remote file watching timeout

9.3 Integration Test Scenarios

1. **Minimal run**: reference + reads only
2. **Full featured run**: all optional inputs
3. **Real-time processing**: file watching
4. **Multi-sample processing**
5. **Platform switching**: same data, different platforms