

NextFlow pipeline for SARS-CoV-2 Illumina data

Table of contents

Quickstart	2
Pipeline overview	4
Hardware requirements	5
Software requirements	8
External databases updates	9
Updateing Nextclade database	12
Updateing Pangolin database	13
Updateing Kraken database	15
Updateing Freyja database	16
Pipeline description	17
Alphabetic list of modules	18

Quickstart

Installation and usage

1. Install Docker (<https://docs.docker.com/desktop/install/linux-install/>)
2. Install NextFlow

```
curl -s https://get.nextflow.io | bash
mv nextflow ~/bin
```

3. Copy `third-party/modeller/config.py.template` to `third-party/modeller/config.py` and replace the line

```
license = 'YOUR_MODELLEER_KEY'
```

with the actual Modeller key you own. If you don't have one, you can get a free academic license here (<https://salilab.org/modeller/registration.html>).

4. Build three containers:

```
docker build --target production -f Dockerfile-main -t
nf_illumina_sars-3.0-main .
docker build --target prodcuton -f Dockerfile-manta -t
nf_illumina_sars-3.0-manta .
docker build --target updater -f Dockerfile-main -t nf_illumina_sars-
3.0-updater:latest .
```

5. Download latest version of external databases:

In project root dir run:

```
./update_external_databases.sh
```

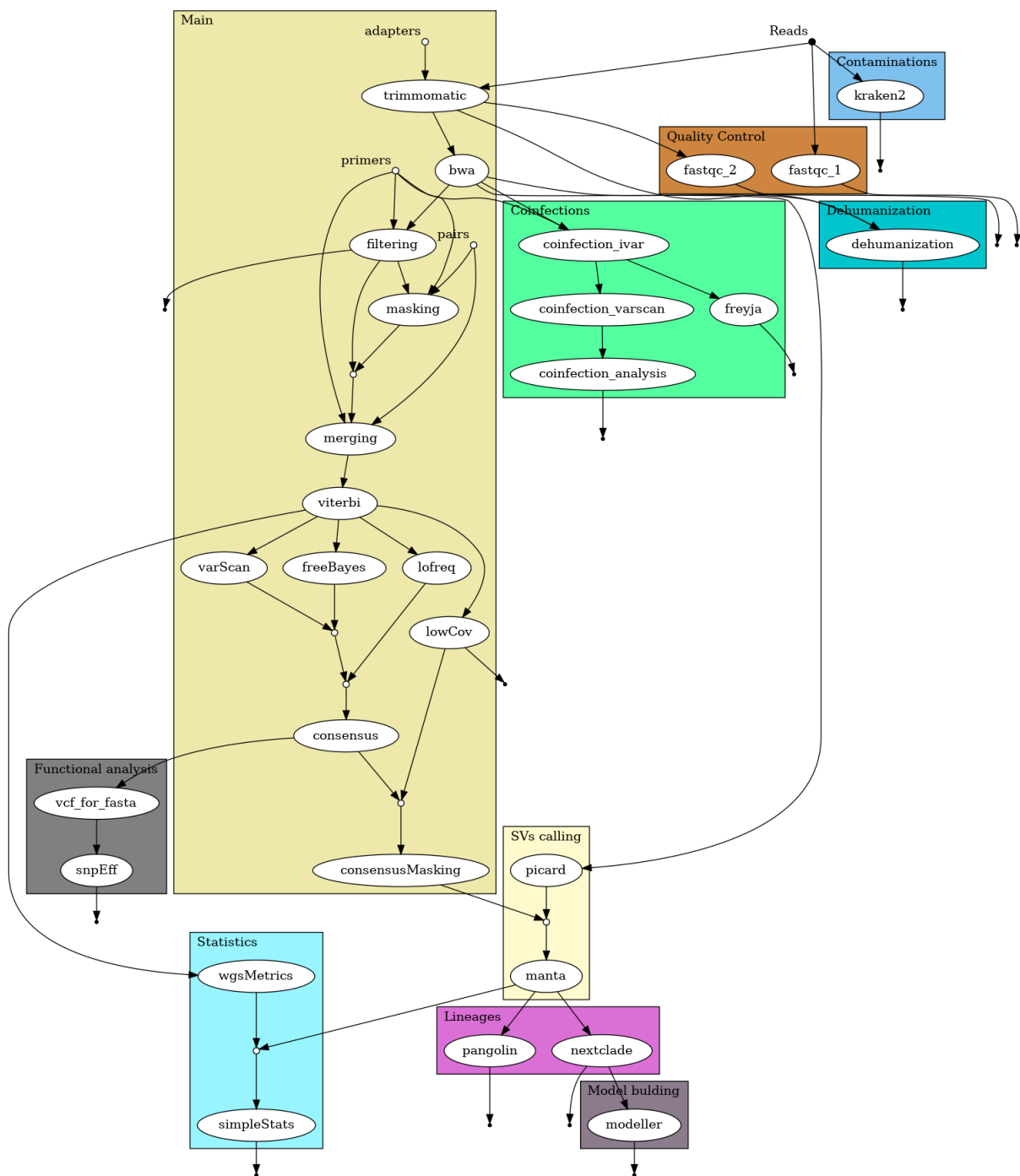
This should fill directories in `data/pangolin` and `data/nextclade`. For more details read the chapter [External databases updates](#).

6. Copy `run_nf_pipeline.sh.template` to `run_nf_pipeline.sh` and fill in the paths to the reads and output directory.

7. Run the pipeline:

```
./run_nf_pipeline.sh
```

Pipeline overview



overview of the pipeline

Hardware requirements

Platform

Pipeline is intended to run on Unix-like computing server with x86_64 architecture.

Memory and CPU

Pipeline consist of multiple steps (processes) that are run in separate containers possibly concurrently. Each process has its own hardware requirements. Proceses vary highly in their demands, from very low to very high. Some may benefit from multiple cores while others are single-threaded or fast enough to not require more than one core. Exact requirements depend on expected number of samples analyzed in parallel.

During our tests we run the pipeline for 32 samples in parallel on a machine with 96 cores (Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz) and 503 GiB of RAM, and we faced out-of-memory issues. We introduced limits on memory-intensive processes (BWA and Kraken2) to maximum 5 concurrent instances, to avoid OOM killer.

Single sample mode

In case of running the pipeline in single sample mode we recommend using at least 16 cores and 64 GiB of RAM.

Multiple samples mode

In case of running set of samples in parallel we recommended using 4 cores and ~100 GiB of RAM per sample.

Storage requirements

Total storage requirements is ~60 GiB of constant data, and further ~3.3 GiB per sample.

Performance

Many processes in the perform a lot of I/O operations, thus pipeline definitely can benefit from fast storage. We recommend store external databases, and temporary files in fast storage like NVMe SSD in RAID 0. It also may be beneficial to store it on in-memory filesystem like tmpfs, however no extensive tests were performed.

Docker images sizes:

Pipeline consist of three docker images two for computations and one is wrapper for external databases updates.

Image	Size
<code>nf_illumina_sars-3.0-main</code>	1.82 GiB
<code>nf_illumina_sars-3.0-updater</code>	257 MiB
<code>nf_illumina_sars-3.0-manta</code>	1.26 GiB
Total	3.33 GiB

Databases sizes:

Pipeline require access to external databases. Total size of databases is **~56 GiB**.

Database	Size
pangolin	~90 MiB
nextclade	~1.3 MiB
kraken	~55 GiB
freyja	~100 MiB
Total	~56 GiB

Temporary files and results

Pipeline generates a lot of temporary files, which are stored in `work` directory. According to our tests on EQA2023 dataset (32 samples) took ~105 GiB of disk space **~3.3 GiB per sample**. Please note that those tests are not representative, and real life data may vary significantly.

GPU requirements

Pipeline does not exploit GPU acceleration, so no GPU is required.

Software requirements

This pipeline is designed to run on a Unix-like operating system. The pipeline is written in Nextflow (<https://www.nextflow.io/docs/latest/index.html>), which is a language for writing bioinformatics pipelines. It is designed to be portable and scalable, and can be run on a variety of platforms, including local machines, clusters, and cloud computing environments.

Our pipeline is containerized using Docker (<https://www.docker.com/>), which is a platform for developing, shipping, and running applications in containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package. This makes it easy to deploy the application on any machine that supports Docker.

However, our containers are run differently than usual docker workflow. Containers are run by nextflow, instead of manual execution of docker. Nextflow take care of mounting volumes and deciding which container should be run and when.

Compatibility

Pipeline was tested with following software versions:

- **Operating system:**
 - Ubuntu 20.04.06 LTS
 - Debian 12.5
- **Docker:**
 - 24.0.7
 - 26.0.2
- **Nextflow:**
 - 23.10.1.5891

It is known that pipeline will not work with docker 20.10.5.

External databases updates

Some components of the pipeline require access to their two databases, which are updated roughly once every two weeks. Different software pieces need to be updated in different ways. To make this process as smooth and painless as possible, we prepared a dedicated Docker container exactly for this task, along with a bash script for running it with appropriate volume mounts. The script should be placed in either the cron or systemd timer and run on a weekly basis.

Updates procedure

Build the dedicated container:

```
docker build --target updater -f Dockerfile-main -t nf_illumina_sars-3.0-updater:latest .
```

Run the updater script. The working dir must be in project root directory.

```
update_external_databases.sh nextclade
update_external_databases.sh pangolin
update_external_databases.sh kraken
update_external_databases.sh freyja
```

Total size of downloads is ~55 GiB.

Database	Size
pangolin	~90 MiB
nextclade	~1.3 MiB
kraken	~55 GiB
freyja	~100 MiB

If everything work fine in directories `data\pangolin` and `data\nextclade` you should see downloaded content like below:

```

data/nextclade/
└─ sars-cov-2.zip

data/pangolin/
├─ bin
├─ pangolin_data
└─ pangolin_data-1.25.1.dist-info

data/kraken
└─ k2_standard_20240112.tar.gz

data/freyja/
├─ curated_lineages.json
├─ lineages.yml
└─ usher_barcodes.csv

```

It is recommended to put the following in crontab or equivalently systemd timer.

```

0 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh nextclade
5 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh pangolin
10 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh freyja
15 3 1 */3 * cd /path/to/sars-illumina &&
bin/update_external_databases.sh kraken

```

Updates internals

The following section contain information what and how is updated. Unless you need to debug or refactor the code, and you followed guides in chapter ["Updates procedure" in "External databases updates"](#) you can safely skip it.

List of components that require updates

- Nextclade
- Pangolin

- Kraken
- Freyja

Updateing Nextclade database

Nextclade (<https://docs.nextstrain.org/projects/nextclade/>) is software for assigning evolutionary lineage to SARS-Cov2. To make it work properly, it requires a database which is updated roughly once every two weeks.

The recommended way of downloading dataset is using `nextclade` tool.

```
nextclade dataset get --name sars-cov-2 --output-zip sars-cov-2.zip
```

Detailed manual is available

<https://docs.nextstrain.org/projects/nextclade/en/stable/user/datasets.html>. Nextclade is downloading `index.json` from site: <https://data.clades.nextstrain.org/v3/index.json>, and based on that files it decide what to download and from where. Probably the same data are available directly on GitHub:

https://github.com/nextstrain/nextclade_data/tree/master/data/nextstrain/sars-cov-2/wuhan-hu-1/orfs.

The command above will download a `sars-cov-2.zip` file in desired destination (default: `data/nextclade`). That directory have to be mounted inside `main` container. It is done by Nextflow in the `modules/nextclade.nf` module.

```
process nextclade {  
    (...)  
    containerOptions "--volume  
    ${params.nextclade_db_absolute_path_on_host}:/home/SARS-  
    CoV2/nextclade_db"  
    (...)
```

Updateing Pangolin database

Pangolin (<https://github.com/cov-lineages/pangolin>) (Phylogenetic Assignment of Named Global Outbreak LINEages) is alternative to Nextclade software for assigning evolutionary lineage to SARS-Cov2.

To make it work properly, it requires a database that is stored in the Git repository pangolin-data (<https://github.com/cov-lineages/pangolin-data>).

Pangolin-data is actually a regular python package. Normal update procedure is via command: `pangolin --update-data`. It also can be installed by `pip` command. Keeping it inside main container is slightly tricky. We don't want to rebuild entire container just to update the database. We also don't want to keep the database inside the container, because it would force us to run the update before every pipeline run, which is stupid. The best solution is to mount the database from the host.

To achieve this goal we install the package externally to the container in designated path using host native `pip`.

```
pip install \
  --target data/pangolin \
  --upgrade \
  git+https://github.com/cov-lineages/pangolin-data.git@v1.25.1
```

i Make sure you entered proper version in the end of git url. The version number is also git tag. List of available tags with their release dates is here (<https://github.com/cov-lineages/pangolin-data/tags>).

Then that dir is mounted as docker volume inside the container (which is done automagically in the Nextflow module file):

```
process variantIdentification {
  containerOptions "--volume
  ${params.pangolin_db_absolute_path_on_host}:/home/SARS-CoV2/pangolin"
  (...)
```

During container build the `$PYTHONPATH` environment variable is set to indicate proper dir.

```
(...)  
ENV PYTHONPATH="/home/SARS-CoV2/pangolin"  
(...)
```

So the manual download consist of two steps:

1. Install the desired version of `pangolin-data` package in `data/pangolin` directory.
2. Provide absolute path to that dir during starting pipeline

```
--pangolin_db_absolute_path_on_host /absolute/path/to/data/pangolin
```

Updateing Kraken database

Kraken 2 (<https://ccb.jhu.edu/software/kraken2/>) is a taxonomic classification system using exact k-mer matches. The pipeline utilizes it to detect contamination in samples. It requires a database of approximately 55 GiB, which could potentially be reduced to 16 GiB or 8 GiB, albeit at the cost of sensitivity and accuracy. It updated roughly quarterly. Skipping updates may result in skipping newer taxa.

Database may be built for user own (not recommended) or be downloaded from aws s3. DB is maintained by Kraken 2 maintainers. Here you can find more here (<https://github.com/BenLangmead/aws-indexes>), and here (<https://benlangmead.github.io/aws-indexes/>).

Downloading is via aws cli (`apt install awscli`), python library `boto3` or HTTP protocol. There are several types of databases, that differ with set of organism. We use and recommend using `standard` db, which is quite complete. There is also `nt` db which contain all RefSeq and GenBank sequences, but it's size and processing time is too high for routine surveillance.

Links for http downloads are available here (<https://benlangmead.github.io/aws-indexes/k2>). They are in form of: `https://genome-idx.s3.amazonaws.com/kraken/k2_DBNAME_YYYYMMDD.tar.gz`

where DB name is one of following: `standard`, `standard_08gb`, `standard_16gb`, `viral`, `minusb`, `pluspf`, `pluspf_08gb`, `pluspf_16gb`, `pluspfp`, `pluspfp_08gb`, `pluspfp_16gb`, `nt`, `eupathdb48`.

Refer to official docs for more details.

In case of our pipeline we use python script that is using `boto3` library. It is part of `nf_illumina_sars-3.0-updater` container. For running this script simply pass `kraken` to the container during running.

We recommend using for this dedicated script: `update_external_databases.sh`.

Kraken DB will not be updated if local path already contain the file with the same name.

Updateing Freyja database

Freyja (<https://andersen-lab.github.io/Freyja/index.html>) is a tool to recover relative lineage abundances from mixed SARS-CoV-2 samples from a sequencing dataset.

Natively Freyja updates require installing Freyja python package (by default with conda) and running `freyja update` command. This command will download the latest version of curated lineages and usher barcodes. However, in our pipeline we do it simpler way. We download the files directly from the dedicated GitHub repository andersen-lab/Freyja-data (<https://github.com/andersen-lab/Freyja-data>) using regular `wget`, which is implemented in the `update_external_databases.sh` script, and `updater` container.

Pipeline description

Start typing here...

Alphabetic list of modules

- bwa.nf
- coinfection_analysis.nf
- coinfection_ivar.nf
- coinfection_varscan.nf
- consensus.nf
- consensusAnalysis.nf
- consensusMasking.nf
- dehumanization.nf
- fastqc.nf
- filtering.nf
- freeBayes.nf
- freyja.nf
- functionalAnalysis.nf
- indexGenome.nf
- kraken2.nf
- lofreq.nf
- lowCov.nf
- manta.nf
- masking.nf
- merging.nf
- modeller.nf
- nextclade.nf

- pangolin.nf
- picard.nf
- simpleStats.nf
- trimmomatic.nf
- varscan.nf
- vcf_for_fasta.nf
- viterbi.nf
- wgsMetrics.nf