ATLAS Documentation

Release 2.0

Joe Brown

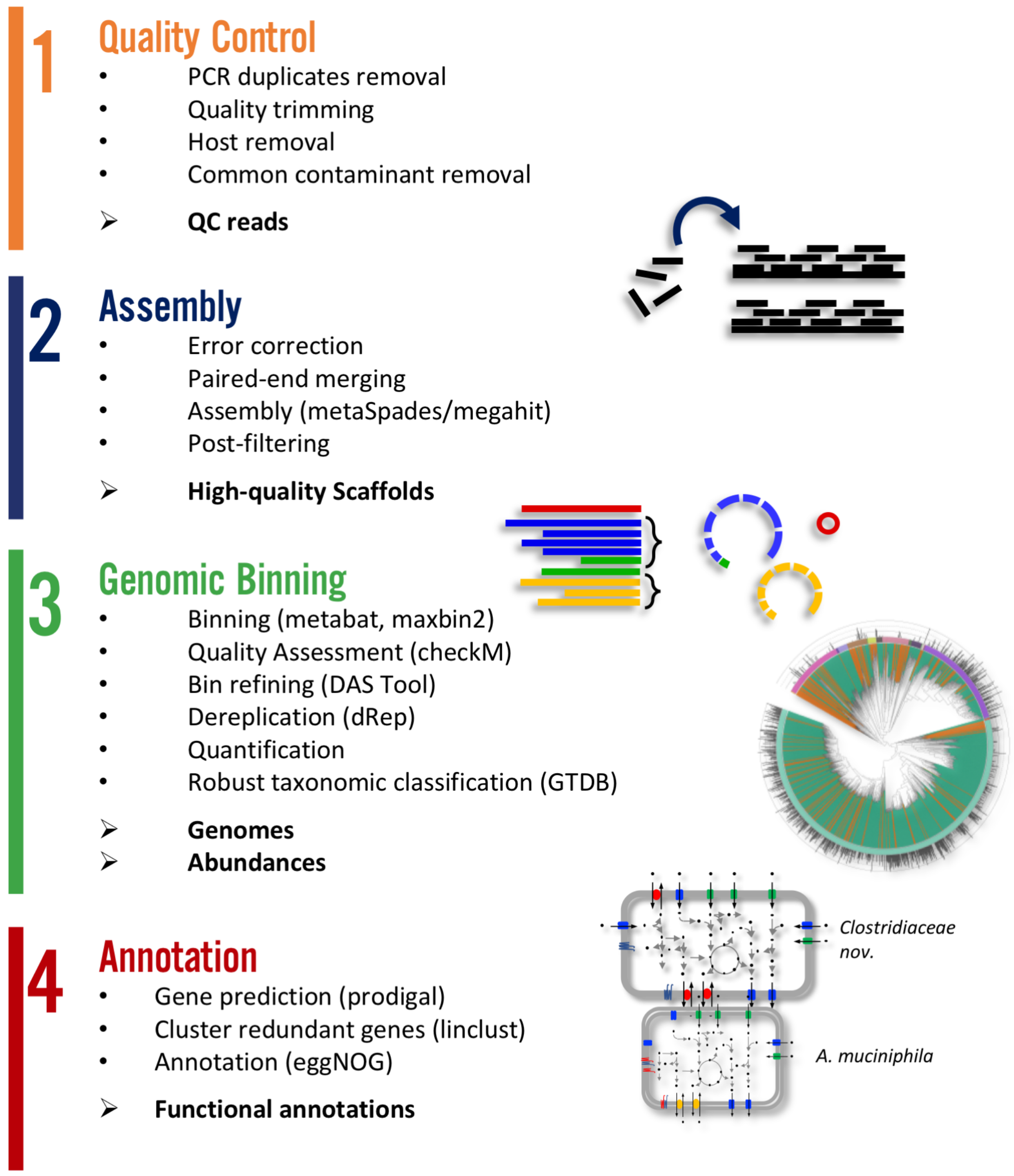
Jul 26, 2021

Documentation

Metagenome-Atlas is a easy-to-use metagenomic pipeline based on snakemake. It handles all steps from QC, Assembly, Binning, to Annotation.

You can start using atlas with three commands:

```
conda install -c bioconda -c conda-forge metagenome-atlas
atlas init --db-dir databases path/to/fastq/files
atlas run
```



1 Quality Control

- PCR duplicates removal
- Quality trimming
- Host removal
- Common contaminant removal

➢ **QC reads**

2 Assembly

- Error correction
- Paired-end merging
- Assembly (metaSpades/megahit)
- Post-filtering

➢ **High-quality Scaffolds**

3 Genomic Binning

- Binning (metabat, maxbin2)
- Quality Assessment (checkM)
- Bin refining (DAS Tool)
- Dereplication (dRep)
- Quantification
- Robust taxonomic classification (GTDB)

➢ **Genomes**

➢ **Abundances**

4 Annotation

- Gene prediction (prodigal)
- Cluster redundant genes (linclust)
- Annotation (eggNOG)

➢ **Functional annotations**

CHAPTER 1

Publication

ATLAS: a Snakemake workflow for assembly, annotation, and genomic binning of metagenome sequence data. Kieser, S., Brown, J., Zdobnov, E. M., Trajkovski, M. & McCue, L. A. BMC Bioinformatics 21, 257 (2020). doi: 10.1186/s12859-020-03585-4

1.1 Getting Started

1.1.1 Setup

Conda package manager

Atlas has **one dependency**: conda. All databases and other dependencies are installed **on the fly**. Atlas is based on snakemake which allows to run steps of the workflow in parallel on a cluster.

If you want to try atlas and have a linux computer (OSX may also work), you can use our *example data* for testing.

For real metagenomic data atlas should be run on a _linux_ sytem, with enough memory (min ~50GB but assembly usually requires 250GB).

You need to install anaconda or miniconda. If you haven't done it already you need to configure conda with the bioconda-channel and the conda-forge channel. This are sources for packages beyond the default one.:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

The order is important by the way.

Install mamba

Conda can be a bit slow because there are so many packages. A good way around this is to use mamba (another snake).:

```
conda install mamba
```

From now on you can replace `conda install` with `mamba install` and see how much faster this snake is.

Install metagenome-atlas

We recommend you to install metagenome-atlas into a conda environment e.g. named `atlasenv`:

```
mamba create -y -n atlasenv metagenome-atlas
source activate atlasenv
```

Install metagenome-atlas from GitHub

Alternatively you can install metagenome Atlas directly form GitHub. This allows you to access versions that are not yet in the conda release, e.g. versions that are still in development.

```
git clone https://github.com/metagenome-atlas/atlas.git
cd atlas

# optional change to different branch
# git checkout branchname

# create dependencies for atlas
mamba env create -n atlas-dev --file atlasenv.yml
conda activate atlas-dev

# install atlas version. Changes in this files are directly available in the atlas␣
↪dev version
pip install --editable .
cd ..
```

1.1.2 Example Data

If you want to test atlas on a small example data here is a two sample, three genome minimal metagenome dataset, to test atlas. Even when atlas will run faster on the test data, it will anyway download all the databases and requirements, for the a complete run, which can take a certain amount of time and especially disk space (>100Gb).

The database dir of the test run should be the same as for the later atlas executions.

The example data can be downloaded as following:

```
wget https://zenodo.org/record/3992790/files/test_reads.tar.gz
tar -xzf test_reads.tar.gz
```

1.1.3 Usage

Start a new project

Let's apply atlas on your data or on our *example data*:

```
atlas init --db-dir databases path/to/fastq
```

This command creates a `samples.tsv` and a `config.yaml` in the working directory.

Have a look at them with a normal text editor and check if the samples names are inferred correctly. Samples should be alphanumeric names and cam be dash delimited. Underscores should be fine too. See the `example sample table`

The `BinGroup` parameter is used during the genomic binning. In short: all samples in which you expect the same strain to be found should belong to the same group, e.g. all metagenome samples from mice in the same cage or location. If you want to use *long reads* for a hybrid assembly, you can also specify them in the sample table.

You should also check the `config.yaml` file, especially:

- You may want to add ad *host genomes* to be removed.
- You may want to change the resources configuration, depending on the system you run atlas on.

Details about the parameters can be found in the section *Configure Atlas*

Keep in mind that all databases are installed in the directory specified with `--db-dir` so choose it wisely.

```
Usage: atlas init [OPTIONS] PATH_TO_FASTQ

  Write the file CONFIG and complete the sample names and paths for all
  FASTQ files in PATH.

  PATH is traversed recursively and adds any file with '.fastq' or '.fq' in
  the file name with the file name minus extension as the sample ID.

Options:
  -d, --db-dir PATH                location to store databases (need ~50GB)
                                   [default: /Users/silas/Documents/GitHub/atla
                                   s/databases]
  -w, --working-dir PATH           location to run atlas
  --assembler [megahit|spades]     assembler  [default: spades]
  --data-type [metagenome|metatranscriptome]
                                   sample data type  [default: metagenome]
  --interleaved-fastq              fastq files are paired-end in one files
                                   (interleaved)
  --threads INTEGER                number of threads to use per multi-threaded
                                   job
  --skip-qc                        Skip QC, if reads are already pre-processed
  -h, --help                       Show this message and exit.
```

Run atlas

```
atlas run all
```

`atlas run` need to know the working directory with a `samples.tsv` inside it.

Take note of the `--dryrun` parameter, see the section *Useful command line options* for other handy snakemake arguments.

We recommend to use atlas on a *Cluster execution* system, which can be set up in a view more commands.

```
Usage: atlas run [OPTIONS]
                 [[qc|assembly|binning|genomes|genecatalog|None|all]]
                 [SNAKEMAKE_ARGS]...

  Runs the ATLAS pipline
```

(continues on next page)

(continued from previous page)

```
  By default all steps are executed but a sub-workflow can be specified.
  Needs a config-file and expects to find a sample table in the working-
  directory. Both can be generated with 'atlas init'

  Most snakemake arguments can be appended to the command for more info see
  'snakemake --help'

  For more details, see: https://metagenome-atlas.readthedocs.io

Options:
  -w, --working-dir PATH  location to run atlas.
  -c, --config-file PATH  config-file generated with 'atlas init'
  -j, --jobs INTEGER      use at most this many jobs in parallel (see cluster
                          submission for mor details).  [default: 8]
  --profile TEXT          snakemake profile e.g. for cluster execution.
  -n, --dryrun            Test execution.  [default: False]
  -h, --help              Show this message and exit.
```

1.2 Execue Atlas

1.2.1 Cluster execution

Automatic submitting to cluster systems

Thanks to the underlying snakemake Atlas can submit parts of the pipeline automatically to a cluster system and define the appropriate resources. If one job has finished it launches the next one. This allows you use the full capacity of your cluster system. You even need to pay attention not to spam the other users of the cluster.

Thanks to the underlying snakemake system, atlas can submit parts of the pipeline to clusters and cloud systems. Instead of running all steps of the pipeline in one cluster job, atlas can automatically submit each step to your cluster system, specifying the necessary threads, memory, and runtime, based on the values in the config file. Atlas periodically checks the status of each cluster job and can re-run failed jobs or continue with other jobs.

See atlas scheduling jobs on a cluster in action https://asciinema.org/a/337467.

If you have a common cluster system (Slurm, LSF, PBS . . .) we have an easy set up (see below). Otherwise, if you have a different cluster system, file a GitHub issue (feature request) so we can help you bring the magic of atlas to your cluster system. For more information about cluster- and cloud submission, have a look at the snakemake cluster docs.

Set up of cluster execution

You need cookiecutter to be installed, which comes with atlas

Then run:

```
cookiecutter --output-dir ~/.config/snakemake https://github.com/metagenome-atlas/
↪clusterprofile.git
```

This opens a interactive shell dialog and ask you for the name of the profile and your cluster system. We recommend you keep the default name `cluster`. The profile was tested on `slurm`, `lsf` and `pbs`.

The resources (threads, memory and time) are defined in the atlas config file (hours and GB).

If you need to specify **queues or accounts** you can do this for all rules or for specific rules in the `~/.config/snakemake/cluster/cluster_config.yaml`. In addition, using this file you can overwrite the resources defined in the config file.

Example for `cluster_config.yaml` with queues defined:

```
__default__:
# default parameter for all rules
  queue: normal
  nodes: 1


# The following rules in atlas need need more time/memory.
# If you need to submit them to different queues you can configure this as outlined.

run_megahit:
  queue: bigmem
run_spades:
  queue: bigmem

This rules can take longer
run_checkm_lineage_wf:
  queue: long
```

Now, you can run atlas on a cluster with:

```
atlas run <options> --profile cluster
```

As the whole pipeline can take several days, I usually run this command in a screen on the head node, even when system administrators don't normally like that. On the head node atlas only schedules the jobs and combines tables, so it doesn't use many resources. You can also submit the atlas command as a long lasting job.

If a job fails, you will find the "external jobid" in the error message. You can investigate the job via this ID.

The atlas argument `--jobs` now becomes the number of jobs simultaneously submitted to the cluster system. You can set this as high as 99 if your colleagues don't mind you over-using the cluster system.

1.2.2 Single machine execution

If you cannot use the *automatic scheduling* you can still try to use atlas on a single machine (local execution) with a lot of memory and threads ideally. In this case I recommend you the following options. The same applies if you submit a single job to a cluster running atlas.

In theory you don't need to adapt the parameters in the config file. However you should tell atlas how many threads and how much memory (GB) you have available on our system so Atlas can take this into account.

For local execution the `--jobs` command line arguments defines the number of threads used in total. Set it to the number of processors available on your machine. If you have less core available than specified in the config file. The jobs are downscaled. If you have more Atlas tries to start multiple jobs, to optimally use the cores on you machine. The same applies for the memory.

For example on a machine with 16 processors and 250GB memory you might want to run:

```
atlas run all --resources mem=245 --jobs 16
```

The whole pipeline can take more than a day. If for any reason the pipeline stops you can just rerun the same command after having inspected the error.

1.2.3 Cloud execution

Atlas, like any other snakemake pipeline can also easily be submitted to cloud systems. I suggest looking at the snakemake doc. Keep in mind any snakemake comand line argument can just be appended to the atlas command.

1.2.4 Useful command line options

Atlas builds on snakemake. We designed the command line interface in a way that additional snakemake arguments can be added to an atlas run call.

For instance the `--profile` used for cluster execution. Other handy snakemake command line arguments include.

> `--keep-going`, which allows atlas in the case of a failed job to continue with independent steps.

For a full list of snakemake arguments see the snakemake doc.

1.3 Expected output

1.3.1 Quality control

```
atlas run qc
#or
atlas run all
```

Runs quality control of single or paired end reads and summarizes the main QC stats in reports/QC_report.html.

Per sample it generates:

- `{sample}/sequence_quality_control/{sample}_QC_{fraction}.fastq.gz`
- Various quality stats in `sample}/sequence_quality_control/read_stats`

Fractions:

When the input was paired end, we will put out three the reads in three fractions R1,R2 and se The se are the paired end reads which lost their mate during the filtering. The se are seamlessly integrated in the next steps.

1.3.2 Assembly

```
atlas run assembly
#or
atlas run all
```

Besides the reports/assembly_report.html this rule outputs the following files per sample:

- `{sample}/{sample}_contigs.fasta`
- `{sample}/sequence_alignment/{sample}.bam`
- `{sample}/assembly/contig_stats/final_contig_stats.txt`

1.3.3 Binning

```
atlas run binning
#or
atlas run all
```

When you use different binners (e.g. metabat, maxbin) and a binner-reconciliator (e.g. DAS Tool), then Atlas will produce for each binner and sample:

- `{sample}/binning/{binner}/cluster_attribution.tsv`

which shows the attribution of contigs to bins. For the final_binner it produces the

- `reports/bin_report_{binner}.html`

See an example as a summary of the quality of all bins.

1.3.4 Genomes

```
atlas run genomes
#or
atlas run all
```

As the binning can predict several times the same genome it is recommended to de-replicate these genomes. For now we use DeRep to filter and de-replicate the genomes. The Metagenome assembled genomes are then renamed, but we keep mapping files.

- `genomes/Dereplication`
- `genomes/clustering/contig2genome.tsv`
- `genomes/clustering/allbins2genome.tsv`

The fasta sequence of the dereplicated and renamed genomes can be found in `genomes/genomes` and their quality estimation are in `genomes/checkm/completeness.tsv`. The quantification of the genomes can be found in:

- `genomes/counts/median_coverage_genomes.tsv`
- `genomes/counts/raw_counts_genomes.tsv`

See in Atlas example how to analyze these abundances.

The predicted genes and translated protein sequences are in `genomes/annotations/genes`.

Taxonomic adnnotation

```
annotations:
  - gtdb_tree
  - gtdb_taxonomy
  - checkm_tree
  - checkm_taxonomy
```

Different annotations can be turned on and off in the config file under the heading `annotations`: A taxonomy for the dereplicated genomes is proposed GTDB. The results can be found in `genomes/taxonomy`. The genomes are placed in a phylogenetic tree separately for bacteria and archaea (if there are any) using the GTDB markers. In addition a tree for bacteria and archaea can be generated based on the checkm markers. All trees are properly rooted using the midpoint. The files can be found in `genomes/tree`

1.3.5 Gene Catalog

```
atlas run all
# or
atlas run genecatalog
```

The gene catalog takes either genes predicted from the genomes or all genes predicted on the contigs and clusters them according to the configuration. This rule produces the following output file for the whole dataset.

- `Genecatalog/gene_catalog.fna`
- `Genecatalog/gene_catalog.faa`
- `Genecatalog/annotations/eggNog.tsv.gz`

1.3.6 All

```
atlas run all
```

1.4 Configure Atlas

1.4.1 Remove reads from Host

One of the most important steps in the Quality control is to remove host genome. You can add any number of genomes to be removed.

We recommend you to use genomes where repetitive sequences are masked. See here for more details human genome.

1.4.2 Long reads

Limitation: Hybrid assembly of long and short reads is supported with spades and metaSpades. However metaSpades needs a paired-end short-read library.

The path of the (preprocessed) long reads should be added manually to the the sample table under a new column heading 'longreads'.

In addition the type of the long reads should be defined in the config file: `longread_type` one of ["pacbio", "nanopore", "sanger", "trusted-contigs", "untrusted-contigs"]

1.4.3 Example config file

```
###################################################################
####                _______   _                    _____       ####
####        /\     |__   __| | |          /\      / ____|      ####
####       /  \       | |    | |         /  \    | (___        ####
####      / /\ \      | |    | |        / /\ \    \___ \       ####
####     / ____ \     | |    | |____   / ____ \   ____) |      ####
####    /_/    \_\    |_|    |______| /_/    \_\ |_____/       ####
####                                                           ####
###################################################################
```

(continues on next page)

(continued from previous page)

```
#  For more details about the config values see:
#  https://metagenome-atlas.rtfd.io


########################
# Execution parameters
########################
# threads and memory (GB) for most jobs especially from BBtools, which are memory␣
↪demanding
threads: 8
mem: 60

# threads and memory for jobs needing high amount of memory. e.g GTDB-tk,checkm or␣
↪assembly
large_mem: 250
large_threads: 8
assembly_threads: 8
assembly_memory: 250

#Runtime only for cluster execution
runtime: #in h
    default: 5
    assembly: 48
    long: 24

# Local directory for temp files, useful for cluster execution without shared file␣
↪system
tmpdir: /tmp
# directory where databases are downloaded with 'atlas download'
database_dir: databases

########################
# Quality control
########################
data_type: metagenome # metagenome or metatranscriptome
interleaved_fastqs: false

# remove (PCR)-duplicated reads using clumpify
deduplicate: true
duplicates_only_optical: false
duplicates_allow_substitutions: 2

# used to trim adapters from reads and read ends
preprocess_adapters: /path/to/databases/adapters.fa
preprocess_minimum_base_quality: 10
preprocess_minimum_passing_read_length: 51
# 0.05 requires at least 5 percent of each nucleotide per sequence
preprocess_minimum_base_frequency: 0.05
preprocess_adapter_min_k: 8
preprocess_allowable_kmer_mismatches: 1
preprocess_reference_kmer_match_length: 27
# error correction where PE reads overlap
error_correction_overlapping_pairs: true
#contamination references can be added such that -- key: /path/to/fasta
contaminant_references:
  PhiX: /path/to/databases/phiX174_virus.fa
#  host:/path/to/host_genome.fasta
```

(continues on next page)

(continued from previous page)

```
# We won't allow large indels
contaminant_max_indel: 20
contaminant_min_ratio: 0.65
contaminant_kmer_length: 13
contaminant_minimum_hits: 1
contaminant_ambiguous: best


########################
# Pre-assembly-processing
########################

error_correction_before_assembly : true

# join R1 and R2 at overlap; unjoined reads are still utilized
merge_pairs_before_assembly : true
merging_k: 62
# extend reads while merging to this many nucleotides
merging_extend2: 40
# Iterations are performed until extend2 x iterations
merging_flags: "ecct iterations=5"

########################
# Assembly
########################
# megahit OR spades
assembler: spades

# Megahit
#-----------
# 2 is for metagenomes, 3 for genomes with 30x coverage
megahit_min_count: 2
megahit_k_min: 21
megahit_k_max: 121
megahit_k_step: 20
megahit_merge_level: 20,0.98
megahit_prune_level: 2
megahit_low_local_ratio: 0.2
# ['default','meta-large','meta-sensitive']
megahit_preset: default

# Spades
#------------
spades_skip_BayesHammer: true
spades_use_scaffolds: true # if false use contigs
#Comma-separated list of k-mer sizes to be used (all values must be odd, less than␣
↪128 and listed in ascending order).
spades_k: auto
spades_preset: meta    # meta, ,normal, rna  single end libraries doesn't work for␣
↪metaspades
spades_extra: ""
longread_type: none # [none,"pacbio", "nanopore", "sanger", "trusted-contigs",
↪"untrusted-contigs"]
# Preprocessed long reads can be defined in the sample table with 'longreads' , for␣
↪more info see the spades manual
```

(continues on next page)

(continued from previous page)

```
# Filtering
#-------------
# filter out assembled noise
# this is more important for assemblys from megahit
filter_contigs: true
prefilter_minimum_contig_length: 300
# trim contig tips
contig_trim_bp: 0
# require contigs to have read support
minimum_average_coverage: 1
minimum_percent_covered_bases: 20
minimum_mapped_reads: 0
# after filtering
minimum_contig_length: 500


########################
# Quantification
########################

# Mapping reads to contigs
#--------------------------
contig_min_id: 0.9
contig_map_paired_only: true
contig_max_distance_between_pairs: 1000
maximum_counted_map_sites: 10

########################
# Binning
########################

final_binner: DASTool             # [DASTool or one of the binner, e.g. maxbin]

binner:                           # If DASTool is used as final_binner, use␣
↪predictions of this binners
  - metabat
  - maxbin


metabat:
  sensitivity: sensitive
  min_contig_length: 1500 # metabat needs >1500

maxbin:
  max_iteration: 50
  prob_threshold: 0.9
  min_contig_length: 1000

DASTool:
  search_engine: 'diamond'
  score_threshold: 0.5              #Score threshold until selection algorithm will␣
↪keep selecting bins [0..1].

genome_dereplication:
  ANI: 0.95
  overlap: 0.6
  opt_parameters: ""
```

(continues on next page)

(continued from previous page)

```
  filter:
      noFilter: false
      length: 5000
      completeness: 50
      contamination: 10
  score:
      completeness: 1
      contamination: 5
      N50: 0.5
      length: 0

rename_mags_contigs: true         #Rename contigs of representative MAGs

########################
# Annotations
#######################

annotations:
  - gtdb_tree
  - gtdb_taxonomy
  - genes
#   - checkm_taxonomy
#   - checkm_tree

########################
# Gene catalog
#######################
genecatalog:
  source: contigs                # [contigs, genomes] Predict genes from all contigs␣
↪or only from the representative genomes
  clustermethod: linclust        # [cd-hit-est or mmseqs or linclust] see mmseqs for␣
↪more details
  minlength_nt: 100
  minid: 0.95                    # min id for gene clustering for the main gene␣
↪catalog used for annotation
  coverage: 0.9
  extra: ""
  SubsetSize: 500000

eggNOG_use_virtual_disk: false    # coping the eggNOG DB to a virtual disk can sppeed␣
↪up the annotation
virtual_disk: "/dev/shm"          # But you need 37G extra ram
```

1.4.4 Detailed configuration