Snakemake Documentation

Release 3.9.0

Johannes Koester

December 06, 2016

Contents

Build systems like GNU Make are frequently used to create complicated workflows, e.g. in bioinformatics. This project aims to reduce the complexity of creating workflows by providing a fast and comfortable execution environment, together with a clean and modern domain specific specification language (DSL) in python style.

Apart from being a command line tool, Snakemake can also be called from withing other python code and hence serve as a framework for organizing workflows withing your software. These pages describe the public parts of the Snakemake api.

For the user documentation and general information, see http://snakemake.bitbucket.org.

Contents:

CHAPTER 1

The Snakemake API

snakemake.**snakemake**(*snakefile*, *listrules=False*, *list_target_rules=False*, *cores=1*, *nodes=1*, *local_cores=1*, *resources={}*, *config={}*, *configfile=None*, *config_args=None*, *workdir=None*, *targets=None*, *dryrun=False*, *touch=False*, *forcetargets=False*, *forceall=False*, *forcerun=[]*, *until=[]*, *omit_from=[]*, *prioritytargets=[]*, *stats=None*, *printreason=False*, *printshellcmds=False*, *printdag=False*, *printrulegraph=False*, *printd3dag=False*, *nocolor=False*, *quiet=False*, *keepgoing=False*, *cluster=None*, *cluster_config=None*, *cluster_sync=None*, *drmaa=None*, *jobname='snakejob.{rulename}.{jobid}.sh'*, *immediate_submit=False*, *standalone=False*, *ignore_ambiguity=False*, *snakemakepath=None*, *lock=True*, *unlock=False*, *cleanup_metadata=None*, *force_incomplete=False*, *ignore_incomplete=False*, *list_version_changes=False*, *list_code_changes=False*, *list_input_changes=False*, *list_params_changes=False*, *list_resources=False*, *summary=False*, *detailed_summary=False*, *latency_wait=3*, *benchmark_repeats=1*, *wait_for_files=None*, *print_compilation=False*, *debug=False*, *notemp=False*, *keep_remote_local=False*, *nodeps=False*, *keep_target_files=False*, *keep_shadow=False*, *allowed_rules=None*, *jobscript=None*, *timestamp=False*, *greediness=None*, *no_hooks=False*, *overwrite_shellcmd=None*, *updated_files=None*, *log_handler=None*, *keep_logger=False*, *max_jobs_per_second=None*, *verbose=False*, *force_use_threads=False*, *use_conda=False*, *mode=0*, *wrapper_prefix=None*)

Run snakemake on a given snakefile.

This function provides access to the whole snakemake functionality. It is not thread-safe.

Parameters

- **snakefile** (*str*) – the path to the snakefile
- **listrules** (*bool*) – list rules (default False)
- **list_target_rules** (*bool*) – list target rules (default False)
- **cores** (*int*) – the number of provided cores (ignored when using cluster support) (default 1)
- **nodes** (*int*) – the number of provided cluster nodes (ignored without cluster support) (default 1)
- **local_cores** (*int*) – the number of provided local cores if in cluster mode (ignored without cluster support) (default 1)
- **resources** (*dict*) – provided resources, a dictionary assigning integers to resource names, e.g. {gpu=1, io=5} (default {})

- **config** (*dict*) – override values for workflow config
- **workdir** (*str*) – path to working directory (default None)
- **targets** (*list*) – list of targets, e.g. rule or file names (default None)
- **dryrun** (*bool*) – only dry-run the workflow (default False)
- **touch** (*bool*) – only touch all output files if present (default False)
- **forcetargets** (*bool*) – force given targets to be re-created (default False)
- **forceall** (*bool*) – force all output files to be re-created (default False)
- **forcerun** (*list*) – list of files and rules that shall be re-created/re-executed (default [])
- **prioritytargets** (*list*) – list of targets that shall be run with maximum priority (default [])
- **stats** (*str*) – path to file that shall contain stats about the workflow execution (default None)
- **printreason** (*bool*) – print the reason for the execution of each job (default false)
- **printshellcmds** (*bool*) – print the shell command of each job (default False)
- **printdag** (*bool*) – print the dag in the graphviz dot language (default False)
- **printrulegraph** (*bool*) – print the graph of rules in the graphviz dot language (default False)
- **printd3dag** (*bool*) – print a D3.js compatible JSON representation of the DAG (default False)
- **nocolor** (*bool*) – do not print colored output (default False)
- **quiet** (*bool*) – do not print any default job information (default False)
- **keepgoing** (*bool*) – keep goind upon errors (default False)
- **cluster** (*str*) – submission command of a cluster or batch system to use, e.g. qsub (default None)
- **cluster_config** (*str, list*) – configuration file for cluster options, or list thereof (default None)
- **cluster_sync** (*str*) – blocking cluster submission command (like SGE 'qsub -sync y') (default None)
- **drmaa** (*str*) – if not None use DRMAA for cluster support, str specifies native args passed to the cluster when submitting a job
- **jobname** (*str*) – naming scheme for cluster job scripts (default "snakejob.{rulename}.{jobid}.sh")
- **immediate_submit** (*bool*) – immediately submit all cluster jobs, regardless of dependencies (default False)
- **standalone** (*bool*) – kill all processes very rudely in case of failure (do not use this if you use this API) (default False) (deprecated)
- **ignore_ambiguity** (*bool*) – ignore ambiguous rules and always take the first possible one (default False)
- **snakemakepath** (*str*) – path to the snakemake executable (default None)
- **lock** (*bool*) – lock the working directory when executing the workflow (default True)

- **unlock** (*bool*) – just unlock the working directory (default False)
- **cleanup_metadata** (*bool*) – just cleanup metadata of output files (default False)
- **force_incomplete** (*bool*) – force the re-creation of incomplete files (default False)
- **ignore_incomplete** (*bool*) – ignore incomplete files (default False)
- **list_version_changes** (*bool*) – list output files with changed rule version (default False)
- **list_code_changes** (*bool*) – list output files with changed rule code (default False)
- **list_input_changes** (*bool*) – list output files with changed input files (default False)
- **list_params_changes** (*bool*) – list output files with changed params (default False)
- **summary** (*bool*) – list summary of all output files and their status (default False)
- **latency_wait** (*int*) – how many seconds to wait for an output file to appear after the execution of a job, e.g. to handle filesystem latency (default 3)
- **benchmark_repeats** (*int*) – number of repeated runs of a job if declared for benchmarking (default 1)
- **wait_for_files** (*list*) – wait for given files to be present before executing the workflow
- **list_resources** (*bool*) – list resources used in the workflow (default False)
- **summary** – list summary of all output files and their status (default False). If no option is specified a basic summary will be ouput. If 'detailed' is added as an option e.g –summary detailed, extra info about the input and shell commands will be included
- **detailed_summary** (*bool*) – list summary of all input and output files and their status (default False)
- **print_compilation** (*bool*) – print the compilation of the snakefile (default False)
- **debug** (*bool*) – allow to use the debugger within rules
- **notemp** (*bool*) – ignore temp file flags, e.g. do not delete output files marked as temp after use (default False)
- **keep_remote_local** (*bool*) – keep local copies of remote files (default False)
- **nodeps** (*bool*) – ignore dependencies (default False)
- **keep_target_files** (*bool*) – Do not adjust the paths of given target files relative to the working directory.
- **keep_shadow** (*bool*) – Do not delete the shadow directory on snakemake startup.
- **allowed_rules** (*set*) – Restrict allowed rules to the given set. If None or empty, all rules are used.
- **jobscript** (*str*) – path to a custom shell script template for cluster jobs (default None)
- **timestamp** (*bool*) – print time stamps in front of any output (default False)
- **greediness** (*float*) – set the greediness of scheduling. This value between 0 and 1 determines how careful jobs are selected for execution. The default value (0.5 if prioritytargets are used, 1.0 else) provides the best speed and still acceptable scheduling quality.
- **overwrite_shellcmd** (*str*) – a shell command that shall be executed instead of those given in the workflow. This is for debugging purposes only.

- **updated_files** (*list*) – a list that will be filled with the files that are updated or created during the workflow execution
- **verbose** (*bool*) – show additional debug output (default False)
- **log_handler** (*function*) – redirect snakemake output to this custom log handler, a function that takes a log message dictionary (see below) as its only argument (default None). The log message dictionary for the log handler has to following entries:
- **max_jobs_per_second** – maximal number of cluster/drmaa jobs per second, None to impose no limit (default None)
- **force_use_threads** – whether to force use of threads over processes. helpful if shared memory is full or unavailable (default False)
- **use_conda** (*bool*) – create conda environments for each job (defined with conda directive of rules)
- **mode** (*snakemake.common.Mode*) – Execution mode
- **wrapper_prefix** (*str*) – Prefix for wrapper script URLs (default None)

  level the log level (“info”, “error”, “debug”, “progress”, “job_info”)

  level=”info”, “error” or “debug”

  msg the log message

  level=”progress”

  done number of already executed jobs

  total number of total jobs

  level=”job_info”

  input list of input files of a job

  output list of output files of a job

  log path to log file of a job

  local whether a job is executed locally (i.e. ignoring cluster)

  msg the job message

  reason the job reason

  priority the job priority

  threads the threads of the job

Returns True if workflow execution was successful.

Return type bool

CHAPTER 2

Additional utils

class snakemake.utils.**AlwaysQuotedFormatter**(*quote_func=<function quote>*, **args*, ***kwargs*)

Subclass of QuotedFormatter that always quotes.

Usage is identical to QuotedFormatter, except that it *always* acts like "q" was appended to the format spec.

class snakemake.utils.**QuotedFormatter**(*quote_func=<function quote>*, **args*, ***kwargs*)

Subclass of string.Formatter that supports quoting.

Using this formatter, any field can be quoted after formatting by appending "q" to its format string. By default, shell quoting is performed using "shlex.quote", but you can pass a different quote_func to the constructor. The quote_func simply has to take a string argument and return a new string representing the quoted form of the input string.

Note that if an element after formatting is the empty string, it will not be quoted.

snakemake.utils.**R**(*code*)

Execute R code

This function executes the R code given as a string. The function requires rpy2 to be installed.

Parameters **code** (*str*) – R code to be executed

class snakemake.utils.**SequenceFormatter**(*separator=' '*, *element_formatter=<string.Formatter object>*, **args*, ***kwargs*)

string.Formatter subclass with special behavior for sequences.

This class delegates formatting of individual elements to another formatter object. Non-list objects are formatted by calling the delegate formatter's "format_field" method. List-like objects (list, tuple, set, frozenset) are formatted by formatting each element of the list according to the specified format spec using the delegate formatter and then joining the resulting strings with a separator (space by default).

format_element(*elem*, *format_spec*)

Format a single element

For sequences, this is called once for each element in a sequence. For anything else, it is called on the entire object. It is intended to be overridden in subclases.

snakemake.utils.**available_cpu_count**()

Return the number of available virtual or physical CPUs on this system. The number of available CPUs can be smaller than the total number of CPUs when the cpuset(7) mechanism is in use, as is the case on some cluster systems.

Adapted from http://stackoverflow.com/a/1006301/715090

snakemake.utils.**format**(*_pattern*, **args*, *stepout=1*, *_quote_all=False*, ***kwargs*)

Format a pattern in Snakemake style.

This means that keywords embedded in braces are replaced by any variable values that are available in the current namespace.

snakemake.utils.**linecount**(*filename*)
Return the number of lines of given file.

Parameters **filename** (*str*) – the path to the file

snakemake.utils.**listfiles**(*pattern*, *restriction=None*, *omit_value=None*)
Yield a tuple of existing filepaths for the given pattern.

Wildcard values are yielded as the second tuple item.

Parameters

- **pattern** (*str*) – a filepattern. Wildcards are specified in snakemake syntax, e.g. "{id}.txt"
- **restriction** (*dict*) – restrict to wildcard values given in this dictionary
- **omit_value** (*str*) – wildcard value to omit

Yields *tuple* – The next file matching the pattern, and the corresponding wildcards object

snakemake.utils.**makedirs**(*dirnames*)
Recursively create the given directory or directories without reporting errors if they are present.

snakemake.utils.**min_version**(*version*)
Require minimum snakemake version, raise workflow error if not met.

snakemake.utils.**read_job_properties**(*jobscript*, *prefix='# properties'*, *pattern=re.compile('# properties = (.*)')*)
Read the job properties defined in a snakemake jobscript.

This function is a helper for writing custom wrappers for the snakemake –cluster functionality. Applying this function to a jobscript will return a dict containing information about the job.

snakemake.utils.**report**(*text*, *path*, *stylesheet='/home/docs/checkouts/readthedocs.org/user_builds/snakemake/checkouts/v3.9.*
defaultenc='utf8', *template=None*, *metadata=None*, ***files*)
Create an HTML report using python docutils.

Attention: This function needs Python docutils to be installed for the python installation you use with Snakemake.

All keywords not listed below are intepreted as paths to files that shall be embedded into the document. They keywords will be available as link targets in the text. E.g. append a file as keyword arg via F1=input[0] and put a download link in the text like this:

```
report('''
==============
Report for ...
==============

Some text. A link to an embedded file: F1_.

Further text.
''', outputpath, F1=input[0])

Instead of specifying each file as a keyword arg, you can also expand
the input of your rule if it is completely named, e.g.:

report('''
Some text...
''', outputpath, **input)
```

Parameters

- **text** (*str*) – The "restructured text" as it is expected by python docutils.
- **path** (*str*) – The path to the desired output file
- **stylesheet** (*str*) – An optional path to a css file that defines the style of the document. This defaults to <your snakemake install>/report.css. Use the default to get a hint how to create your own.
- **defaultenc** (*str*) – The encoding that is reported to the browser for embedded text files, defaults to utf8.
- **template** (*str*) – An optional path to a docutils HTML template.
- **metadata** (*str*) – E.g. an optional author name or email address.

snakemake.utils.**set_protected_output**(**rules*)
Set the output of rules to protected

snakemake.utils.**set_temporary_output**(**rules*)
Set the output of rules to temporary

snakemake.utils.**update_config**(*config*, *overwrite_config*)
Recursively update dictionary config with overwrite_config.

See http://stackoverflow.com/questions/3232943/update-value-of-a-nested-dictionary-of-varying-depth for details.

Parameters

- **config** (*dict*) – dictionary to update
- **overwrite_config** (*dict*) – dictionary whose items will overwrite those in config

S

A

F

L

M

Q

R

S

U