Jenkins Job Builder Documentation

Release 5.0.5.dev1

Jenkins Job Builder Maintainers

Aug 19, 2023

CONTENTS

1 README **1**
- 1.1 Developers . . . 1
- 1.2 Writing a patch . . . 2
- 1.3 Unit Tests . . . 2
- 1.4 Installing without setup.py . . . 2

2 Contents **3**
- 2.1 Quick Start Guide . . . 3
  - 2.1.1 Use Case 1: Test a job definition . . . 3
  - 2.1.2 Use Case 2: Updating Jenkins Jobs . . . 3
  - 2.1.3 Use Case 3: Working with JSON job definitions . . . 4
  - 2.1.4 Use Case 4: Deleting a job . . . 4
  - 2.1.5 Use Case 5: Providing plugins info . . . 4
- 2.2 Installation . . . 4
  - 2.2.1 Documentation . . . 5
  - 2.2.2 Unit Tests . . . 5
  - 2.2.3 Test Coverage . . . 5
- 2.3 Configuration File . . . 5
  - 2.3.1 job_builder section . . . 6
  - 2.3.2 jenkins section . . . 7
  - 2.3.3 hipchat section . . . 7
  - 2.3.4 stash section . . . 7
  - 2.3.5 __future__ section . . . 8
- 2.4 Running . . . 8
  - 2.4.1 Test Mode . . . 8
  - 2.4.2 Updating Jobs . . . 8
  - 2.4.3 Passing Multiple Paths . . . 9
  - 2.4.4 Recursive Searching of Paths . . . 9
  - 2.4.5 Excluding Paths . . . 9
  - 2.4.6 Deleting Jobs/Views . . . 10
  - 2.4.7 Globbed Parameters . . . 11
  - 2.4.8 Providing Plugins Info . . . 11
  - 2.4.9 Command Reference . . . 11
- 2.5 Job Definitions . . . 14
  - 2.5.1 Definitions . . . 14
  - 2.5.2 Custom Yaml Tags . . . 32
  - 2.5.3 Modules . . . 42
  - 2.5.4 Module Execution . . . 360
- 2.6 Extending . . . 360
  - 2.6.1 The Builder . . . 360

2.6.2 XML Processing . . . 360
2.6.3 Modules . . . 361
2.6.4 Components . . . 362
2.6.5 Module Registry . . . 362

3 Indices and tables **363**

Python Module Index **365**

Index **367**

CHAPTER

ONE

README

Jenkins Job Builder takes simple descriptions of Jenkins jobs in YAML or JSON format and uses them to configure Jenkins. You can keep your job descriptions in human readable text format in a version control system to make changes and auditing easier. It also has a flexible template system, so creating many similarly configured jobs is easy.

To install:

```
$ pip install --user jenkins-job-builder
```

Online documentation:

- https://jenkins-job-builder.readthedocs.io/en/latest/

1.1 Developers

Bug report:

- https://storyboard.openstack.org/#!/project/723

Repository:

- https://opendev.org/jjb/jenkins-job-builder

Cloning:

```
git clone https://opendev.org/jjb/jenkins-job-builder.git
```

Install pre-commit from https://pre-commit.com/#intro in order to run some minimal testing on your commits.

A virtual environment is recommended for development. For example, Jenkins Job Builder may be installed from the top level directory:

```
$ virtualenv .venv
$ source .venv/bin/activate
$ pip install -r test-requirements.txt -e .
```

Patches are submitted via Gerrit at:

- https://review.opendev.org

Please do not submit GitHub pull requests, they will be automatically closed.

Mailing list:

- https://groups.google.com/forum/#!forum/jenkins-job-builder

IRC:

- `#openstack-jjb` on OFTC

More details on how you can contribute is available on our wiki at:

- https://docs.openstack.org/infra/manual/developers.html

1.2 Writing a patch

We ask that all code submissions be pep8 and pyflakes clean. The easiest way to do that is to run tox before submitting code for review in Gerrit. It will run `pep8` and `pyflakes` in the same manner as the automated test suite that will run on proposed patchsets.

When creating new YAML components, please observe the following style conventions:

- All YAML identifiers (including component names and arguments) should be lower-case and multiple word identifiers should use hyphens. E.g., "build-trigger".
- The Python functions that implement components should have the same name as the YAML keyword, but should use underscores instead of hyphens. E.g., "build_trigger".

This consistency will help users avoid simple mistakes when writing YAML, as well as developers when matching YAML components to Python implementation.

1.3 Unit Tests

Unit tests have been included and are in the `tests` folder. Many unit tests samples are included as examples in our documentation to ensure that examples are kept current with existing behaviour. To run the unit tests, execute the command:

```
tox -e py38
```

Unit tests could be run in parallel, using pytest-parallel pytest plugin:

```
tox -e py38 -- --workers=auto
```

- Note: View `tox.ini` to run tests on other versions of Python, generating the documentation and additionally for any special notes on running the test to validate documentation external URLs from behind proxies.

1.4 Installing without setup.py

For YAML support, you will need libyaml installed.

Mac OS X:

```
$ brew install libyaml
```

Then install the required python packages using pip:

```
$ sudo pip install PyYAML python-jenkins
```

CHAPTER

TWO

CONTENTS

2.1 Quick Start Guide

This guide was made with the impatient in mind so explanation is sparse. It will guide users through a set of typical use cases for JJB using the same job definitions we use to test JJB.

1. Clone the repository to get the JJB job definition examples
2. The *Installation* can be either from pypi (released version) or from the clone (master).

Usage of the commands below assumes that you are at the root of the cloned directory.

2.1.1 Use Case 1: Test a job definition

JJB creates Jenkins XML configuration file from a YAML/JSON definition file and just uploads it to Jenkins. JJB provides a convenient `test` command to allow you to validate the XML before you attempt to upload it to Jenkins.

Test a YAML job definition:

```
jenkins-jobs test tests/yamlparser/job_fixtures/templates002.yaml
```

The above command prints the generated Jenkins XML to the console. If you prefer to send it to a directory:

```
jenkins-jobs test -o output tests/yamlparser/job_fixtures/templates002.yaml
```

The *output* directory will contain files with the XML configurations.

2.1.2 Use Case 2: Updating Jenkins Jobs

Once you've tested your job definition and are happy with it then you can use the `update` command to deploy the job to Jenkins. The `update` command requires a configuration file. An example file is supplied in the etc folder, you should update it to match your Jenkins master:

```
jenkins-jobs --conf etc/jenkins_jobs.ini-sample update tests/yamlparser/job_fixtures/
↪templates002.yaml
```

The above command will update your Jenkins master with the generated jobs.

Caution: JJB caches Jenkins job information locally. Changes made using the Jenkins UI will not update that cache, which may lead to confusion. See *Updating Jobs* for more information.

2.1.3 Use Case 3: Working with JSON job definitions

You can also define your jobs in json instead of yaml:

```
jenkins-jobs --conf etc/jenkins_jobs.ini-sample update tests/jsonparser/fixtures/simple.
↪json
```

The above command just uses a simple job definition. You can also convert any of the YAML examples to JSON and feed that to JJB.

2.1.4 Use Case 4: Deleting a job

To delete a job:

```
jenkins-jobs --conf etc/jenkins_jobs.ini-sample delete simple
```

The above command deletes the job *simple* from the Jenkins master.

2.1.5 Use Case 5: Providing plugins info

To generate a plugins info, using an account with Administrator rights:

```
jenkins-jobs get-plugins-info -o plugins_info.yaml
```

To run JJB update using the plugins_info.yaml:

```
jenkins-jobs update -p plugins_info.yaml ./myjobs
```

Please refer to the jenkins-jobs *Command Reference* and the *Job Definitions* pages for more details.

2.2 Installation

To install Jenkins Job Builder from source, run:

```
pip install git+https://opendev.org/jjb/jenkins-job-builder/
```

A virtual environment is recommended for development. For example, Jenkins Job Builder may be installed from the top level directory:

```
$ virtualenv .venv
$ source .venv/bin/activate
$ pip install -r test-requirements.txt -e .
```

Alternatively, the current release can be installed from pypi:

```
sudo pip install jenkins-job-builder
```

The OpenStack project uses Puppet to manage its infrastructure systems, including Jenkins. If you use Puppet, you can use the OpenStack Jenkins module to install Jenkins Job Builder.

2.2.1 Documentation

Documentation is included in the `doc` folder. To generate docs locally execute the command:

```
tox -e docs
```

The generated documentation is then available under `doc/build/html/index.html`.

As over time URLs change or become stale there is also a testenv available to verify any links added. To run locally execute the command:

```
tox -e docs-linkcheck
```

- Note: When behind a proxy it is necessary to use `TOX_TESTENV_PASSENV` to pass any proxy settings for this test to be able to check links are valid.

2.2.2 Unit Tests

Unit tests have been included and are in the `tests` folder. We recently started including unit tests as examples in our documentation so to keep the examples up to date it is very important that we include unit tests for every module. To run the unit tests, execute the command:

```
tox -e py38
```

- Note: View `tox.ini` to run tests on other versions of Python.

2.2.3 Test Coverage

To measure test coverage, execute the command:

```
tox -e cover
```

2.3 Configuration File

After installation, you will need to create a configuration file. By default, `jenkins-jobs` looks for `~/.config/jenkins_jobs/jenkins_jobs.ini`, `<script directory>/jenkins_jobs.ini` or `/etc/jenkins_jobs/jenkins_jobs.ini` (in that order), but you may specify an alternative location when running `jenkins-jobs`. The file should have the following format:

```
[job_builder]
ignore_cache=True
keep_descriptions=False
include_path=.:scripts:~/git/
recursive=False
exclude=.*:manual:./development
allow_duplicates=False
update=all

[jenkins]
user=jenkins
password=1234567890abcdef1234567890abcdef
```

(continues on next page)

(continued from previous page)

```
url=https://jenkins.example.com
query_plugins_info=False
##### This is deprecated, use job_builder section instead
#ignore_cache=True

[plugin "hipchat"]
authtoken=dummy

[plugin "stash"]
username=user
password=pass
```

2.3.1 job_builder section

ignore_cache
(Optional) If set to True, Jenkins Job Builder won't use any cache.

keep_descriptions
By default *jenkins-jobs* will overwrite the jobs descriptions even if no description has been defined explicitly. When this option is set to True, that behavior changes and it will only overwrite the description if you specified it in the yaml. False by default.

include_path
(Optional) Can be set to a ':' delimited list of paths, which jenkins job builder will search for any files specified by the custom application yaml tags 'include', 'include-raw' and 'include-raw-escape'.

recursive
(Optional) If set to True, jenkins job builder will search for job definition files recursively.

exclude
(Optional) If set to a list of values separated by ':', these paths will be excluded from the list of paths to be processed when searching recursively. Values containing no / will be matched against directory names at all levels, those starting with / will be considered absolute, while others containing a / somewhere other than the start of the value will be considered relative to the starting path.

allow_duplicates
(Optional) By default *jenkins-jobs* will abort when a duplicate macro, template, job-group or job name is encountered as it cannot establish the correct one to use. When this option is set to True, only a warning is emitted.

allow_empty_variables
(Optional) When expanding strings, by default *jenkins-jobs* will raise an exception if there's a key in the string, that has not been declared in the input YAML files. Setting this option to True will replace it with the empty string, allowing you to use those strings without having to define all the keys it might be using.

print_job_urls
(Optional) If set to True it will print full jobs urls while updating jobs, so user can be sure which instance was updated. User may click the link to go directly to that job. False by default.

retain_anchors
(Optional) If set to True, YAML anchors will be retained across files, allowing jobs to be composed from bits of YAML defined in separate files. Note this means that the order of processing files matters - *jenkins-jobs* loads files in alphabetical order (all files in a dir are loaded before any files in subdirs). For example, if your anchors are in a file named *foo.yml* they will be accessible in *qux.yml* but not in *bar.yml*. They will also be accessible in *mydir/bar.yml* and *mydir/qux.yml*. False by default.

update
(Optional) If set, allows the user to specify if only "jobs" or "views" (or "all") are updated. Users can override the setting here by passing `--jobs-only` or `--views-only` CLI options. (Valid options: jobs, views, all)

2.3.2 jenkins section

user
This should be the name of a user previously defined in Jenkins. Appropriate user permissions must be set under the Jenkins security matrix: under the `Global` group of permissions, check `Read`, then under the `Job` group of permissions, check `Create`, `Delete`, `Configure` and finally `Read`.

password
The API token for the user specified. You can get this through the Jenkins management interface under `People` -> username -> `Configure` and then click the `Show API Token` button.

url
The base URL for your Jenkins installation.

timeout
(Optional) The connection timeout (in seconds) to the Jenkins server. By default this is set to the system configured socket timeout.

query_plugins_info
Whether to query the Jenkins instance for plugin info. If no configuration files are found (either in the default paths or given through the command-line), *jenkins-jobs* will skip querying for plugin information. False by default.

2.3.3 hipchat section

send-as
This is the hipchat user name that will be used when sending notifications.

authtoken
The API token necessary to send messages to hipchat. This can be generated in the hipchat web interface by a user with administrative access for your organization. This authtoken is set for each job individually; the JJB Hipchat Plugin does not currently support setting different tokens for different projects, so the token you use will have to be scoped such that it can be used for any room your jobs might be configured to notify. For more information on this topic, please see the Hipchat API Documentation

2.3.4 stash section

username
This is the stash user name that will be used to connect to stash when using the stash publisher plugin and not defining it in the yaml part.

password
This is the related password that will be used with the stash username when using the stash publisher plugin and not defining it in the yaml part.

2.3.5 __future__ section

This section is to control enabling of beta features or behaviour changes that deviate from previously released behaviour in ways that may require effort to convert existing JJB configs to adopt. This essentially will act as a method to share these new behaviours while under active development so they can be changed ahead of releases.

param_order_from_yaml
: Used to switch on using the order of the parameters are defined in yaml to control the order of corresponding XML elements being written out. This is intended as a global flag and can affect multiple modules.

2.4 Running

After it's installed and configured, you can invoke Jenkins Job Builder by running `jenkins-jobs`. You won't be able to do anything useful just yet without a configuration; that is discussed in the next section.

2.4.1 Test Mode

Once you have a configuration defined, you can run the job builder in test mode.

If you want to run a simple test with just a single YAML job definition file and see the XML output on stdout:

```
jenkins-jobs test /path/to/foo.yaml
```

You can also pass JJB a directory containing multiple job definition files:

```
jenkins-jobs test /path/to/defs -o /path/to/output
```

which will write XML files to the output directory for all of the jobs defined in the defs directory.

If you run:

```
jenkins-jobs test /path/to/defs -o /path/to/output --config-xml
```

the output directory will contain config.xml files similar to the internal storage format of Jenkins. This might allow you to more easily compare the output to an existing Jenkins installation.

2.4.2 Updating Jobs

When you're satisfied with the generated XML from the test, you can run:

```
jenkins-jobs update /path/to/defs
```

which will upload the job and view definitions to Jenkins if needed. Jenkins Job Builder maintains, for each host, a cache[1] of previously configured jobs and views, so that you can run that command as often as you like, and it will only update the jobs configurations in Jenkins if the defined definitions have changed since the last time it was run. Note: if you modify a job directly in Jenkins, jenkins-jobs will not know about it and will not update it.

To update a specific list of jobs/views, simply pass the job/view names as additional arguments after the job definition path. To update Foo1 and Foo2 run:

```
jenkins-jobs update /path/to/defs Foo1 Foo2
```

[1] The cache default location is at `~/.cache/jenkins_jobs`, which can be overridden by setting the `XDG_CACHE_HOME` environment variable.

You can also enable the parallel execution of the program passing the workers option with a value of 0, 2, or higher. Use 0 to run as many workers as cores in the host that runs it, and 2 or higher to specify the number of workers to use:

```
jenkins-jobs update --workers 0 /path/to/defs
```

To update only views or only jobs, simply add the argument –views-only or –jobs-only after the command:

```
jenkins-jobs update --views-only Foo-view
jenkins-jobs update --jobs-only Foo-job
```

2.4.3 Passing Multiple Paths

It is possible to pass multiple paths to JJB using colons as a path separator on *nix systems and semi-colons on Windows systems. For example:

```
jenkins-jobs test /path/to/global:/path/to/instance:/path/to/instance/project
```

This helps when structuring directory layouts as you may selectively include directories in different ways to suit different needs. If you maintain multiple Jenkins instances suited to various needs you may want to share configuration between those instances (global). Furthermore, there may be various ways you would like to structure jobs within a given instance.

2.4.4 Recursive Searching of Paths

In addition to passing multiple paths to JJB it is also possible to enable recursive searching to process all yaml files in the tree beneath each path. For example:

```
For a tree:
  /path/
    to/
      defs/
        ci_jobs/
        release_jobs/
      globals/
        macros/
        templates/

jenkins-jobs update -r /path/to/defs:/path/to/globals
```

JJB will search defs/ci_jobs, defs/release_jobs, globals/macros and globals/templates in addition to the defs and globals trees.

2.4.5 Excluding Paths

To allow a complex tree of jobs where some jobs are managed differently without needing to explicitly provide each path, the recursive path processing supports excluding paths based on absolute paths, relative paths and patterns. For example:

```
For a tree:
  /path/
    to/
```

(continues on next page)

(continued from previous page)

```
      defs/
        ci_jobs/
          manual/
        release_jobs/
          manual/
        qa_jobs/
      globals/
        macros/
        templates/
        special/

jenkins-jobs update -r -x man*:./qa_jobs -x /path/to/defs/globals/special \
  /path/to/defs:/path/to/globals
```

JJB will search the given paths, ignoring the directories qa_jobs, ci_jobs/manual, release_jobs/manual, and globals/special when building the list of yaml files to be processed. Absolute paths are denoted by starting from the root, relative by containing the path separator, and patterns by having neither. Patterns use simple shell globing to match directories.

2.4.6 Deleting Jobs/Views

Jenkins Job Builder supports deleting jobs and views from Jenkins.

To delete a specific job:

```
jenkins-jobs delete Foo1
```

To delete a list of jobs or views, simply pass them as additional arguments after the command:

```
jenkins-jobs delete Foo1 Foo2
```

To delete only views or only jobs, simply add the argument –views-only or –jobs-only after the command:

```
jenkins-jobs delete --views-only Foo1
jenkins-jobs delete --jobs-only Foo1
```

The `update` command includes a `delete-old` option to remove obsolete jobs:

```
jenkins-jobs update --delete-old /path/to/defs
```

Obsolete jobs are jobs once managed by JJB (as distinguished by a special comment that JJB appends to their description), that were not generated in this JJB run.

There is also a command to delete **all** jobs and/or views. **WARNING**: Use with caution.

To delete **all** jobs and views:

```
jenkins-jobs delete-all
```

TO delete **all** jobs:

```
jenkins-jobs delete-all --jobs-only
```

To delete **all** views:

```
jenkins-jobs delete-all --views-only
```

2.4.7 Globbed Parameters

Jenkins job builder supports globbed parameters to identify jobs from a set of definition files. This feature only supports JJB managed jobs.

To update jobs/views that only have 'foo' in their name:

```
jenkins-jobs update ./myjobs \*foo\*
```

To delete jobs/views that only have 'foo' in their name:

```
jenkins-jobs delete --path ./myjobs \*foo\*
```

2.4.8 Providing Plugins Info

With Jenkins LTS 1.651.1+ retrieving plugins info became a secure feature and now requires Administrator rights to use [#f2]. This causes JJB to no longer be able to work in situations where a user wants to publish jobs to Jenkins but is not able to receive the Administrator permissions. In this case we can provide a plugins_info.yaml file containing the plugin versions data needed by JJB to parse the job templates.

To generate a plugins info, using an account with Administrator rights:

    jenkins-jobs get-plugins-info -o plugins_info.yaml

To run JJB update using the plugins_info.yaml:

    jenkins-jobs update -p plugins_info.yaml ./myjobs

2.4.9 Command Reference

```
usage: jenkins-jobs [-h] [--conf CONF] [-l LOG_LEVEL] [--ignore-cache]
                    [--flush-cache] [--version] [--allow-empty-variables]
                    [--server SECTION] [--user USER] [--password PASSWORD]
                    {delete,delete-all,get-plugins-info,list,test,update} ...

positional arguments:
  {delete,delete-all,get-plugins-info,list,test,update}
                        update, test, list or delete job
    delete-all          delete *ALL* jobs from Jenkins server, including those
                        not managed by Jenkins Job Builder.
    get-plugins-info    get plugins info yaml by querying Jenkins server.
    list                List jobs

optional arguments:
  -h, --help            show this help message and exit
  --conf CONF           configuration file [JJB_CONF]
  -l LOG_LEVEL, --log_level LOG_LEVEL
                        log level (default: info) [JJB_LOG_LEVEL]
  --ignore-cache        ignore the cache and update the jobs anyhow (that will
                        only flush the specified jobs cache)
```

(continues on next page)

(continued from previous page)

```
  --flush-cache         flush all the cache entries before updating
  --version             show version
  --allow-empty-variables
                        Don't fail if any of the variables inside any string
                        are not defined, replace with empty string instead.
  --server SECTION, -s SECTION
                        The Jenkins server ini section to use. Defaults to
                        'jenkins' [JJB_SECTION]
  --user USER, -u USER  The Jenkins user to use for authentication. This
                        overrides the user specified in the configuration
                        file. [JJB_USER]
  --password PASSWORD, -p PASSWORD
                        Password or API token to use for authenticating
                        towards Jenkins. This overrides the password specified
                        in the configuration file. [JJB_PASSWORD]
```

```
usage: jenkins-jobs test [-h] [-r] [-x EXCLUDE] [--config-xml]
                         [-p PLUGINS_INFO_PATH] [-o OUTPUT_DIR]
                         [path] [names [names ...]]

positional arguments:
  path                  colon-separated list of paths to YAML files or
                        directories
  names                 name(s) of job(s)

optional arguments:
  -h, --help            show this help message and exit
  -r, --recursive       look for yaml files recursively
  -x EXCLUDE, --exclude EXCLUDE
                        paths to exclude when using recursive search, uses
                        standard globbing.
  --config-xml          use alternative output file layout using config.xml
                        files
  -p PLUGINS_INFO_PATH, --plugin-info PLUGINS_INFO_PATH
                        path to plugin info YAML file
  -o OUTPUT_DIR         path to output XML
```

```
usage: jenkins-jobs update [-h] [-r] [-x EXCLUDE] [--delete-old]
                           [-p PLUGINS_INFO_PATH] [--workers N_WORKERS]
                           [--existing-only] [--enabled-only] [-j | -v]
                           [path] [names [names ...]]

positional arguments:
  path                  colon-separated list of paths to YAML files or
                        directories
  names                 name(s) of job(s)

optional arguments:
  -h, --help            show this help message and exit
  -r, --recursive       look for yaml files recursively
  -x EXCLUDE, --exclude EXCLUDE
```

(continues on next page)

(continued from previous page)

```
                        paths to exclude when using recursive search, uses
                        standard globbing.
  --delete-old          delete obsolete jobs
  -p PLUGINS_INFO_PATH, --plugin-info PLUGINS_INFO_PATH
                        path to plugin info YAML file. Can be used to provide
                        previously retrieved plugins info when connecting
                        credentials don't have permissions to query.
  --workers N_WORKERS   number of workers to use, 0 for autodetection and 1
                        for just one worker.
  --existing-only       update existing jobs only
  --enabled-only        update enabled jobs only
  -j, --jobs-only       update only jobs
  -v, --views-only      update only views
```

```
usage: jenkins-jobs delete-all [-h] [-r] [-x EXCLUDE] [-j] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -r, --recursive       look for yaml files recursively
  -x EXCLUDE, --exclude EXCLUDE
                        paths to exclude when using recursive search, uses
                        standard globbing.
  -j, --jobs-only       delete only jobs
  -v, --views-only      delete only views
```

```
usage: jenkins-jobs delete [-h] [-r] [-x EXCLUDE] [-p PATH] [-j] [-v]
                           name [name ...]

positional arguments:
  name                  name of job

optional arguments:
  -h, --help            show this help message and exit
  -r, --recursive       look for yaml files recursively
  -x EXCLUDE, --exclude EXCLUDE
                        paths to exclude when using recursive search, uses
                        standard globbing.
  -p PATH, --path PATH  colon-separated list of paths to YAML files or
                        directories
  -j, --jobs-only       delete only jobs
  -v, --views-only      delete only views
```

```
usage: jenkins-jobs list [-h] [-r] [-x EXCLUDE] [-p PATH] [names [names ...]]

positional arguments:
  names                 name(s) of job(s)

optional arguments:
  -h, --help            show this help message and exit
  -r, --recursive       look for yaml files recursively
  -x EXCLUDE, --exclude EXCLUDE
```

(continues on next page)

(continued from previous page)

```
                        paths to exclude when using recursive search, uses
                        standard globbing.
  -p PATH, --path PATH  path to YAML file or directory
```

```
usage: jenkins-jobs get-plugins-info [-h] [-o PLUGINS_INFO_FILE]

optional arguments:
  -h, --help            show this help message and exit
  -o PLUGINS_INFO_FILE, --output-file PLUGINS_INFO_FILE
                        file to save output to.
```

2.5 Job Definitions

The job definitions for Jenkins Job Builder are kept in any number of YAML or JSON files, in whatever way you would like to organize them. When you invoke `jenkins-jobs` you may specify either the path of a single YAML file, or a directory. If you choose a directory, all of the .yaml/.yml or .json files in that directory will be read, and all the jobs they define will be created or updated.

Note: Jenkins Job Builder 2.x plugins are designed to default to generating the xml format for the latest supported version of JJB. This is a change in behaviour from 1.x and below which defaulted to the oldest supported plugin version.

2.5.1 Definitions

Jenkins Job Builder understands a few basic object types which are described in the next sections.

Job

The most straightforward way to create a job is simply to define a Job in YAML. It looks like this:

```
- job:
    name: job-name
```

That's not very useful, so you'll want to add some actions such as *Builders*, and perhaps *Publishers*. Those are described later.

These are job parameters that are common to every type of Jenkins job.

Example:

```
- job:
    name: job-name
    project-type: freestyle
    defaults: global
    description: 'Do not edit this job through the web!'
    disabled: false
    display-name: 'Fancy job name'
    concurrent: true
    workspace: /srv/build-area/job-name
```

(continues on next page)

(continued from previous page)

```
    quiet-period: 5
    block-downstream: false
    block-upstream: false
    retry-count: 3
    node: NodeLabel1 || NodeLabel2
    logrotate:
      daysToKeep: 3
      numToKeep: 20
      artifactDaysToKeep: -1
      artifactNumToKeep: -1
```

Job Parameters

- **project-type**: Defaults to "freestyle", but "maven" as well as "multijob", "flow", "pipeline" or "externaljob" can also be specified.
- **defaults**: Specifies a set of *Defaults* to use for this job, defaults to ''global". If you have values that are common to all of your jobs, create a `global` *Defaults* object to hold them, and no further configuration of individual jobs is necessary. If some jobs should not use the `global` defaults, use this field to specify a different set of defaults.
- **description**: The description for the job. By default, the description "!– Managed by Jenkins Job Builder" is applied.
- **disabled**: Boolean value to set whether or not this job should be disabled in Jenkins. Defaults to `false` (job will be enabled).
- **display-name**: Optional name shown for the project throughout the Jenkins web GUI in place of the actual job name. The jenkins_jobs tool cannot fully remove this trait once it is set, so use caution when setting it. Setting it to the same string as the job's name is an effective un-set workaround. Alternately, the field can be cleared manually using the Jenkins web interface.
- **concurrent**: Boolean value to set whether or not Jenkins can run this job concurrently. Defaults to `false`.
- **workspace**: Path for a custom workspace. Defaults to Jenkins default configuration.
- **folder**: The folder attribute provides an alternative to using '<path>/<name>' as the job name to specify which Jenkins folder to upload the job to.

  Requires the Jenkins CloudBees Folders Plugin.
- **child-workspace**: Path for a child custom workspace. Defaults to Jenkins default configuration. This parameter is only valid for matrix type jobs.
- **quiet-period**: Number of seconds to wait between consecutive runs of this job. Defaults to `0`.
- **block-downstream**: Boolean value to set whether or not this job must block while downstream jobs are running. Downstream jobs are determined transitively. Defaults to `false`.
- **block-upstream**: Boolean value to set whether or not this job must block while upstream jobs are running. Upstream jobs are determined transitively. Defaults to `false`.
- **auth-token**: Specifies an authentication token that allows new builds to be triggered by accessing a special predefined URL. Only those who know the token will be able to trigger builds remotely.
- **retry-count**: If a build fails to checkout from the repository, Jenkins will retry the specified number of times before giving up.
- **node**: Restrict where this job can be run. If there is a group of machines that the job can be built on, you can specify that label as the node to tie on, which will cause Jenkins to build the job on

any of the machines with that label. For matrix projects, this parameter will only restrict where the parent job will run.

- **logrotate**: The Logrotate section allows you to automatically remove old build history. It adds the `logrotate` attribute to the *Job* definition. All logrotate attributes default to "-1" (keep forever). **Deprecated on jenkins >=1.637**: use the `build-discarder` property instead
- **jdk**: The name of the jdk to use
- **raw**: If present, this section should contain a single **xml** entry. This XML will be inserted at the top-level of the *Job* definition.

Job Template

If you need several jobs defined that are nearly identical, except perhaps in their names, SCP targets, etc., then you may use a Job Template to specify the particulars of the job, and then use a *Project* to realize the job with appropriate variable substitution. Any variables not specified at the project level will be inherited from the *Defaults*.

A Job Template has the same syntax as a *Job*, but you may add variables anywhere in the definition. Variables are indicated by enclosing them in braces, e.g., `{name}` will substitute the variable *name*. When using a variable in a string field, it is good practice to wrap the entire string in quotes, even if the rules of YAML syntax don't require it because the value of the variable may require quotes after substitution. In the rare situation that you must encode braces within literals inside a template (for example a shell function definition in a builder), doubling the braces will prevent them from being interpreted as a template variable.

You must include a variable in the `name` field of a Job Template (otherwise, every instance would have the same name). For example:

```
- job-template:
    name: '{name}-unit-tests'
```

Will not cause any job to be created in Jenkins, however, it will define a template that you can use to create jobs with a *Project* definition. It's name will depend on what is supplied to the *Project*.

If you use the variable `{template-name}`, the name of the template itself (e.g. `{name}-unit-tests` in the above example) will be substituted in. This is useful in cases where you need to trace a job back to its template.

Sometimes it is useful to have the same job name format used even where the template contents may vary. *Ids* provide a mechanism to support such use cases in addition to simplifying referencing templates when the name contains the more complex substitution with default values.

Default Values for Template Variables

To facilitate reuse of templates with many variables that can be substituted, but where in most cases the same or no value is needed, it is possible to specify defaults for the variables within the templates themselves.

There are 2 ways JJB allows us to define defaults for a parameter in a job-template.

1. Defining the default variable value in the job-template itself

   With this method we declare the default value of the variable in the job-template itself just once. We can section off the job-template into two sections like this:

```
- job-template:
    name: '{project-name}-verify'

    ####################
```

(continues on next page)

(continued from previous page)

```
    # Variable Defaults #
    #####################

    branch: master

    #####################
    # Job Configuration #
    #####################

    parameters:
      - string:
          name: BRANCH
          default: '{branch}'

    scm:
      - git:
          refspec: 'refs/heads/{branch}'
```

In this case there is still two branch definitions for the job-template. However we also provide the default value for the {branch} variable at the top of the file. Just once. This will be the value that the job takes on if it is not passed in by a project using the template.

2. Using {var|default}

   In this method we can define the default with the definition of the variable. For example:

```
- job-template:
    name: '{project-name}-verify'
    parameters:
      - string:
          name: BRANCH
          default: '{branch|master}'
```

However where this method falls apart if we need to use the same JJB variable in more than one place as we will have multiple places to define the default value for the template. For example:

```
- job-template:
    name: '{project-name}-verify'
    parameters:
      - string:
          name: BRANCH
          default: '{branch|master}'

    scm:
      - git:
          refspec: 'refs/heads/{branch|master}'
```

We can see in this case the `{branch|master}` variable is defined in two places. Not ideal.

More complex example:

```
- project:
    name: template_variable_defaults
    jobs:
```

(continues on next page)

(continued from previous page)

```
        - 'template-variable-defaults-{num}':
            num: 1
            disabled_var: true
        - 'template-variable-defaults-{num}':
            test_var: Goodbye World
            num: 2

- job-template:
    # template specific defaults
    # empty value causes disabled_var to be ignored internally
    disabled_var:
    test_var: Hello World
    type: periodic

    # template settings
    name: 'template-variable-defaults-{num}-{type}'
    id: 'template-variable-defaults-{num}'
    disabled: '{obj:disabled_var}'
    builders:
      - shell: |
          echo "Job Name: template-variable-defaults-{num}-{type}"
          echo "Variable: {test_var}"
```

To use a default value for a variable used in the name would be uncommon unless it was in addition to another variable. However you can use *Ids* simplify such use cases.

Project

The purpose of a project is to collect related jobs together, and provide values for the variables in a *Job Template*. It looks like this:

```
- project:
    name: project-name
    jobs:
      - '{name}-unit-tests'
```

Any number of arbitrarily named additional fields may be specified, and they will be available for variable substitution in the job template. Any job templates listed under `jobs:` will be realized with those values. The example above would create the job called 'project-name-unit-tests' in Jenkins.

The `jobs:` list can also allow for specifying job-specific substitutions as follows:

```
- project:
    name: project-name
    jobs:
      - '{name}-unit-tests':
          mail-to: developer@nowhere.net
      - '{name}-perf-tests':
          mail-to: projmanager@nowhere.net
```

If a variable is a list, the job template will be realized with the variable set to each value in the list. Multiple lists will lead to the template being realized with the cartesian product of those values. Example:

```
- project:
    name: project-name
    pyver:
      - 26
      - 27
    jobs:
      - '{name}-{pyver}'
```

If there are templates being realized that differ only in the variable used for its name (thus not a use case for job-specific substitutions), additional variables can be specified for project variables. Example:

```
- job-template:
    name: '{name}-{pyver}'
    builders:
      - shell: 'git co {branch_name}'

- project:
   name: project-name
   pyver:
    - 26:
       branch_name: old_branch
    - 27:
       branch_name: new_branch
   jobs:
    - '{name}-{pyver}'
```

You can also specify some variable combinations to exclude from the matrix with the `exclude` keyword, to avoid generating jobs for those combinations. You can specify all the variables of the combination or only a subset, if you specify a subset, any value of the omited variable will match:

```
- project:
    name: project-name
    axe1:
      - axe1val1
      - axe1val2
    axe2:
      - axe2val1
      - axe2val2
    axe3:
      - axe3val1
      - axe3val2
    exclude:
      - axe1: axe1val1
        axe2: axe2val1
        axe3: axe3val2
      - axe2: axe2val2
        axe3: axe3val1
    jobs:
      - build-{axe1}-{axe2}-{axe3}

- job-template:
    name: build-{axe1}-{axe2}-{axe3}
    builders:
```

(continues on next page)

(continued from previous page)

```
      - shell: "echo Combination {axe1}:{axe2}:{axe3}"
```

The above example will omit the jobs:

- build-axe1val1-axe2val1-axe3val2
- build-axe1val1-axe2val2-axe3val1
- build-axe1val2-axe2val2-axe3val1

To achieve the same without the `exclude` tag one would have to do something a bit more complicated, that gets more complicated for each dimension in the combination, for the previous example, the counterpart would be:

```
- project:
    name: project-name_comb1
    axe1:
      - axe1val1
      - axe1val2
    axe2: axe2val1
    axe3: axe3val1
    jobs:
      - build-{axe1}-{axe2}-{axe3}

- project:
    name: project-name_comb2
    axe1:
      - axe1val1
      - axe1val2
    axe2: axe2val2
    axe3: axe3val2
    jobs:
      - build-{axe1}-{axe2}-{axe3}

- project:
    name: project-name_comb3
    axe1: axe1val2
    axe2: axe2val1
    axe3: axe3val2
    jobs:
      - build-{axe1}-{axe2}-{axe3}

- job-template:
    name: build-{axe1}-{axe2}-{axe3}
    builders:
      - shell: "echo Combination {axe1}:{axe2}:{axe3}"
```

Job Group

If you have several Job Templates that should all be realized together, you can define a Job Group to collect them. Simply use the Job Group where you would normally use a *Job Template* and all of the Job Templates in the Job Group will be realized. For example:

```
- job-template:
    name: '{name}-unit-tests'
    builders:
    - shell: unittest
    publishers:
    - email:
        recipients: '{mail-to}'

- job-template:
    name: '{name}-perf-tests'
    builders:
    - shell: perftest
    publishers:
    - email:
        recipients: '{mail-to}'

- job-group:
    name: '{name}-tests'
    jobs:
    - '{name}-unit-tests':
        mail-to: developer@nowhere.net
    - '{name}-perf-tests':
        mail-to: projmanager@nowhere.net

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

Would cause the jobs *project-name-unit-tests* and *project-name-perf-tests* to be created in Jenkins.

Views

A view is a particular way of displaying a specific set of jobs. To create a view, you must define a view in a YAML file and have a variable called view-type with a valid value. It looks like this:

```
- view:
    name: view-name
    view-type: list
```

Views are processed differently than Jobs and therefore will not work within a *Project* or a *Job Template*.

View Template

Allow views to also be configured via templates similar to job-templates. This is useful when you have multiple views defined that have similar configuration except for a few variables. View Templates can be passed variables to fill in sections automatically via a project configuration using the new 'views' key.

Minimal Example:

```
- view-template:
    name: '{name}-template-{seq}'
    description: 'testing view templates feature'
    view-type: list
    regex: 'test-view-.*'

- project:
    name: 'test-view'
    views:
        - '{name}-template-{seq}'
    seq:
        - a
        - b
        - c
```

Macro

Many of the actions of a *Job*, such as builders or publishers, can be defined as a Macro, and then that Macro used in the *Job* description. Builders are described later, but let's introduce a simple one now to illustrate the Macro functionality. This snippet will instruct Jenkins to execute "make test" as part of the job:

```
- job:
    name: foo-test
    builders:
      - shell: 'make test'
```

If you wanted to define a macro (which won't save much typing in this case, but could still be useful to centralize the definition of a commonly repeated task), the configuration would look like:

```
- builder:
    name: make-test
    builders:
      - shell: 'make test'

- job:
    name: foo-test
    builders:
      - make-test
```

This allows you to create complex actions (and even sequences of actions) in YAML that look like first-class Jenkins Job Builder actions. Not every attribute supports Macros, check the documentation for the action before you try to use a Macro for it.

Macros can take parameters, letting you define a generic macro and more specific ones without having to duplicate code:

```
# The 'add' macro takes a 'number' parameter and will creates a
# job which prints 'Adding ' followed by the 'number' parameter:
- builder:
    name: add
    builders:
     - shell: "echo Adding {number}"

# A specialized macro 'addtwo' reusing the 'add' macro but with
# a 'number' parameter hardcoded to 'two':
- builder:
    name: addtwo
    builders:
     - add:
         number: "two"

# Glue to have Jenkins Job Builder to expand this YAML example:
- job:
    name: "testingjob"
    builders:
     # The specialized macro:
     - addtwo
     # Generic macro call with a parameter
     - add:
         number: "ZERO"
     # Generic macro called without a parameter. Never do this!
     # See below for the resulting wrong output :(
     - add
```

Then <builders /> section of the generated job show up as:

```
<builders>
  <hudson.tasks.Shell>
    <command>echo Adding two</command>
  </hudson.tasks.Shell>
  <hudson.tasks.Shell>
    <command>echo Adding ZERO</command>
  </hudson.tasks.Shell>
  <hudson.tasks.Shell>
    <command>echo Adding {number}</command>
  </hudson.tasks.Shell>
</builders>
```

As you can see, the specialized macro addtwo reused the definition from the generic macro add.

Macro Notes

If a macro is not passed any parameters it will not have any expansion performed on it. Thus if you forget to provide *any* parameters to a macro that expects some, the parameter-templates (`{foo}`) will be left as is in the resulting output; this is almost certainly not what you want. Note if you provide an invalid parameter, the expansion will fail; the expansion will only be skipped if you provide *no* parameters at all.

Macros are expanded using Python string substitution rules. This can especially cause confusion with shell snippets that use { as part of their syntax. As described, if a macro has *no* parameters, no expansion will be performed and thus it is correct to write the script with no escaping, e.g.:

```
- builder:
  name: a_builder
  builders:
    - shell: |
        VARIABLE=${VARIABLE:-bar}
        function foo {
            echo "my shell function"
        }
```

However, if the macro *has* parameters, you must escape the { you wish to make it through to the output, e.g.:

```
- builder:
  name: a_builder
  builders:
     - shell: |
       PARAMETER={parameter}
       VARIABLE=${{VARIABLE:-bar}}
       function foo {{
            echo "my shell function"
       }}
```

Note that a `job-template` will have parameters by definition (at least a `name`). Thus embedded-shell within a `job-template` should always use {{ to achieve a literal {. A generic builder will need to consider the correct quoting based on its use of parameters.

Folders

Jenkins supports organising jobs, views, and slaves using a folder hierarchy. This allows for easier separation of access as well credentials and resources which can be assigned to only be available for a specific folder.

JJB has two methods of supporting uploading jobs to a specific folder:

- Name the job to contain the desired folder `<folder>/my-job-name`
- Use the `folder` attribute on a job definition, via a template, or through *Defaults*.

Supporting both an attributed and use of it directly in job names allows for teams to have all jobs using their defaults automatically use a top-level folder, while still allowing for them to additionally nest jobs for their own preferences.

Job Name Example:

```
- job:
    name: python-jobs/tox-py27
    builders:
```

(continues on next page)

(continued from previous page)

```
      - shell: |
          tox -e py27
```

Folder Attribute Example:

```
- defaults:
    name: team1
    folder: team1-jobs

- job:
    name: ruby-jobs/rspec
    defaults: team1
    builders:
      - shell: |
          rvm use --create ruby-2.3.0@rspec
          bundle install
          bundle exec rspec
```

Item ID's

It's possible to assign an *id* to any of the blocks and then use that to reference it instead of the name. This has two primary functions:

- A unique identifier where you wish to use the same naming format for multiple templates. This allows you to follow a naming scheme while still using multiple templates to handle subtle variables in job requirements.
- Provides a simpler name for a *job-template* where you have multiple variables including default values in the name and don't wish to have to include this information in every use. This also makes changing the template output name without impacting references.

Example:

```
- project:
    name: test_template_id
    jobs:
        - 'simple-template':
            test_var: Hello World
            type: periodic
            num: 1
        - 'not-as-simple-template':
            test_var: Goodbye World
            type: canary
            num: 2

- job-template:
    name: 'template-test-ids-{num}-{type}'
    id: simple-template
    builders:
      - shell: |
          echo "Template name: {template-name}"
          echo "Job name: template-test-ids-{num}-{type}"
          echo "{test_var}"
```

(continues on next page)

(continued from previous page)

```
- job-template:
    name: 'template-test-ids-{num}-{type}'
    id: not-as-simple-template
    builders:
      - shell: |
          echo "Template name: {template-name}"
          echo "Job name: template-test-ids-{num}-{type}"
      - shell: |
          echo "{test_var}"
```

Raw config

It is possible, but not recommended, to use *raw* within a module to inject raw xml into the job configs.

This is relevant in case there is no appropriate module for a Jenkins plugin or the module does not behave as you expect it to do.

For example:

```
wrappers:
  - raw:
      xml: |
        <hudson.plugins.xvnc.Xvnc>
          <takeScreenshot>true</takeScreenshot>
          <useXauthority>false</useXauthority>
        </hudson.plugins.xvnc.Xvnc>
```

Is the raw way of adding support for the *xvnc* wrapper.

To get the appropriate xml to use you would need to create/edit a job in Jenkins and grab the relevant raw xml segment from the *config.xml*.

The xml string can refer to variables just like anything else and as such can be parameterized like anything else.

You can use *raw* in most locations, the following example show them with arbitrary xml-data:

```
- project:
    name: complete002
    version:
        - 1.2
    jobs:
        - 'complete001_{version}'

- job-template:
    name: 'complete001_{version}'
    project-type: maven
    scm:
      - raw:
          xml: |
            <!-- <scm> for raw replaces the whole scm section.
             where as for others the raw part is added to the existing.
            -->
            <scm>
              <scmraw/>
```

(continues on next page)

(continued from previous page)

```
        </scm>
  triggers:
    - raw:
        xml: |
          <triggersraw/>
  wrappers:
    - raw:
        xml: |
          <wrappersraw/>
  builders:
    - raw:
        xml: |
          <buildersraw/>
  publishers:
    - raw:
        xml: |
          <publishersraw/>
  properties:
    - raw:
        xml: |
          <propertiesraw/>
  parameters:
    - raw:
        xml: |
          <parametersraw/>
  notifications:
    - raw:
        xml: |
          <metadataraw/>
  reporters:
    - raw:
        xml:
          <reportersraw/>
```

Note: If you have a need to use *raw* please consider submitting a patch to add or fix the module that will remove your need to use *raw*.

Defaults

Defaults collect job attributes (including actions) and will supply those values when the job is created, unless superseded by a value in the 'Job'_ definition. If a set of Defaults is specified with the name `global`, that will be used by all *Job* (and *Job Template*) definitions unless they specify a different Default object with the `defaults` attribute. For example:

```
- defaults:
    name: global
    description: 'Do not edit this job through the web!'
```

Will set the job description for every job created.

You can define variables that will be realized in a *Job Template*.

```
- defaults:
    name: global
    arch: 'i386'

- project:
    name: project-name
    jobs:
        - 'build-{arch}'
        - 'build-{arch}':
            arch: 'amd64'

- job-template:
    name: 'build-{arch}'
    builders:
        - shell: "echo Build arch {arch}."
```

Would create jobs `build-i386` and `build-amd64`.

You can also reference a variable `{template-name}` in any value and it will be subtitued by the name of the current job template being processed.

Variable References

If you want to pass an object (boolean, list or dict) to templates you can use an `{obj:key}` variable in the job template. This triggers the use of code that retains the original object type.

For example:

```
- project:
    name: test_custom_distri
    disabled: true
    distributions: !!python/tuple [precise, jessie]
    architectures: !!python/tuple &architectures
      - amd64
      - i386
    axis_a:
        type: user-defined
        name: architectures
        values: *architectures
    jobs:
      - '{name}-source'

- job-template:
      name: '{name}-source'
      project-type: matrix
      disabled: '{obj:disabled}'
      axes:
        - axis:
            type: user-defined
            name: distribution
            values: '{obj:distributions}'
        - axis: '{obj:axis_a}'
```

JJB also supports interpolation of parameters within parameters. This allows a little more flexibility when ordering template jobs as components in different projects and job groups.

For example:

```
- job-template:
    name: '{value-stream}_{project-id}_foo'
    display-name: '{value-stream} {project-id} foo'
    publishers:
        - trigger-parameterized-builds:
            - project: '{downstream}'
              current-parameters: False
              condition: ALWAYS
              git-revision: True

- job-template:
    name: '{value-stream}_{project-id}_bar'
    display-name: '{value-stream} {project-id} bar'
    publishers:
        - trigger-parameterized-builds:
            - project: '{downstream}'
              current-parameters: False
              condition: ALWAYS
              git-revision: True

- job-group:
    name: 'pipeline2'
    project-id: 'p2'
    jobs:
        - '{value-stream}_{project-id}_foo':
            downstream: '{value-stream}_{project-id}_bar'
        - '{value-stream}_{project-id}_bar':

- job-group:
    name: 'pipeline1'
    project-id: 'p1'
    jobs:
        - '{value-stream}_{project-id}_bar':
            downstream: '{value-stream}_{project-id}_foo'
        - '{value-stream}_{project-id}_foo':

- project:
    name: derp
    jobs:
        - 'pipeline1':
            value-stream: 'production'
        - 'pipeline2':
            value-stream: 'experimental'

- defaults:
    name: 'global'
    downstream: ''
```

By default JJB will fail if it tries to interpolate a variable that was not defined, but you can change that behavior and allow empty variables with the allow_empty_variables configuration option.

For example, having a configuration file with that option enabled:

```
[job_builder]
allow_empty_variables = True
```

Will prevent JJb from failing if there are any non-initialized variables used and replace them with the empty string instead.

Tip: Refer to *Default Values for Template Variables* for details on setting variable defaults.

Variable Inheritance

It is possible in JJB to define defaults for variables at different levels such that it is possible for users of job-templates to override variables defined in the job-template.

Variable priorities for each definition type are as follows:

1. job-group
2. project
3. job-template
4. defaults

From this list we can immediately see that if we want to make variables in job-templates override-able then using defaults configuration is useless as it has the lowest precedence when JJB is deciding where to pull from.

On the other side of the spectrum, job-groups has the highest precedence. Which unfortunately means if we define a variable in a job-group with the intention of overriding it at the project level then we are out of luck. For this reason avoid setting variables in job-groups unless we want to enforce a setting for a set of jobs and prevent projects from overriding it.

Declaring variable defaults

Refer to *Default Values for Template Variables* for details on how to declare variable defaults.

Overriding job-template variables

When a project wants to use a job-template it can use override it as follows:

```
- project:
    name: foo
    jobs:
      - '{project-name}-merge'
      - '{project-name}-verify'

    branch: master
```

This is the standard way that most folks use and it will set `branch: master` for every job-template in the list. However sometimes we may want to provide an alternative value for a specific job in the list. In this case the more specific declaration takes precedence:

```
- project:
    name: foo
    jobs:
      - '{project-name}-merge':
          branch: production
      - '{project-name}-verify'

    branch: master
```

In this case the verify job will get the value **master** but the merge job will instead get the branch value **production**.

Yaml Anchors & Aliases

The yaml specification supports anchors and aliases which means that JJB definitions allow references to variables in templates.

For example:

```
- _wrapper_defaults: &wrapper_defaults
    name: 'wrapper_defaults'
    wrappers:
      - timeout:
          timeout: 180
          fail: true
      - timestamps

- _job_defaults: &job_defaults
    name: 'defaults'
    <<: *wrapper_defaults

- job-template:
    name: 'myjob'
    <<: *job_defaults

- project:
    name: myproject
    jobs:
      - myjob
```

The anchors and aliases are expanded internally within JJB's yaml loading calls and are not limited to individual documents. That means you can't use the same anchor name in included files without collisions.

A simple example can be seen in the specs full length example with the following being more representative of usage within JJB:

```
- wrapper_defaults: &wrapper_defaults
    name: 'wrapper_defaults'
    wrappers:
      - timeout:
          timeout: 180
          fail: true
      - timestamps
```

(continues on next page)

(continued from previous page)

```
- job_defaults: &job_defaults
    name: 'defaults'
    <<: *wrapper_defaults

- job-template:
    name: 'myjob'
    <<: *job_defaults
```

Which will be expanded to the following yaml before being processed:

```
- wrapper_defaults:
    name: wrapper_defaults
    wrappers:
    - timeout:
        fail: true
        timeout: 180
    - timestamps
- job_defaults:
    name: defaults
    wrappers:
    - timeout:
        fail: true
        timeout: 180
    - timestamps
- job-template:
    name: myjob
    wrappers:
    - timeout:
        fail: true
        timeout: 180
    - timestamps
```

2.5.2 Custom Yaml Tags

Custom application specific yamls tags are supported to provide enhancements when reading yaml configuration.

Action Tags

These allow manipulation of data being stored in one layout in the source yaml for convenience and/or clarity, to another format to be processed by the targeted module instead of requiring all modules in JJB being capable of supporting multiple input formats.

The tag `!join:` will treat the first element of the following list as the delimiter to use, when joining the remaining elements into a string and returning a single string to be consumed by the specified module option.

This allows users to maintain elements of data in a list structure for ease of review/maintenance, and have the yaml parser convert it to a string for consumption as any argument for modules. The main expected use case is to allow for generic plugin data such as shell properties to be populated from a list construct which the yaml parser converts to a single string, instead of trying to support this within the module code which would require a templating engine similar to Jinja.

Generic Example:

```
- job:
    name: sample-job
    builders:
      - shell:
        - string-with-comma: !join:
            - ','
            -
                - item1
                - item2
                - item3

        - string-with-space: !join:
            - ' '
            -
                - item1
                - item2
                - item3
```

Environment Inject:

```
- project:
    name: string_join_example
    jobs:
        - 'string-join-data-{name}':
            name: set1
            files: !join:
              - ','
              -
                - /path/to/file1
                - /path/to/file2
                - /path/to/file3
                - /path/to/file4
                - /path/to/file5
                - /path/to/file6
                - /path/to/file7
                - /path/to/file8
                - /path/to/file9
                - /path/to/file10
                - /path/to/file11
                - /path/to/file12
                - /path/to/file13
                - /path/to/file14
                - /path/to/file15
                - /path/to/file16
                - /path/to/file17
                - /path/to/file18
                - /path/to/file19
                - /path/to/file20
        - 'string-join-data-{name}':
            name: set2
            files: !join:
              - ','
              -
```

(continues on next page)

(continued from previous page)

```
                - /another/different/path/to/file1
                - /another/different/path/to/file2
                - /another/different/path/to/file3
                - /another/different/path/to/file4
                - /another/different/path/to/file5
                - /another/different/path/to/file6
                - /another/different/path/to/file7
                - /another/different/path/to/file8
                - /another/different/path/to/file9
                - /another/different/path/to/file10
                - /another/different/path/to/file11
                - /another/different/path/to/file12
                - /another/different/path/to/file13
                - /another/different/path/to/file14
                - /another/different/path/to/file15
                - /another/different/path/to/file16
                - /another/different/path/to/file17
                - /another/different/path/to/file18
                - /another/different/path/to/file19
                - /another/different/path/to/file20

- job-template:
    name: 'string-join-data-{name}'
    properties:
      - inject:
          keep-system-variables: true
          properties-content: |
            FILE_LIST={files}
    builders:
      - shell: |
          echo "Template name: {template-name}"
          echo "Data to be processed:"
          echo "${{INPUT_DATA}}"
```

While this mechanism can also be used items where delimiters are supported by the module, that should be considered a bug that the existing code doesn't handle being provided a list and delimiter to perform the correct conversion for you. Should you discover a module that takes arguments with delimiters and the existing JJB codebase does not handle accepting lists, then this can be used as a temporary solution in place of using very long strings:

Extended Params Example:

```
parameters:
  - extended-choice:
      name: OPTIONS_VALUE
      description: "Available options"
      property-key: key
      quote-value: true
      type: multi-select
      value: "foo|bar|select"
      visible-items: 2
      multi-select-delimiter: '|'
      default-value: foo
      default-property-key: fookey
```

(continues on next page)

(continued from previous page)

```
- extended-choice:
    name: OPTIONS_FILE
    description: "Available options"
    property-file: /home/foo/property.prop
    property-key: key
    quote-value: true
    type: multi-select
    visible-items: 2
    multi-select-delimiter: '|'
    default-property-file: /home/property.prop
    default-property-key: fookey
- extended-choice:
    name: OPTIONS_CHECKBOX
    type: checkbox
    value: !join:
      - ','
      -
          - OptionA
          - OptionB
          - OptionC
    visible-items: 2
- extended-choice:
    name: MULTISELECTOPTIONS
    description: "Available options"
    property-key: key
    quote-value: true
    type: multi-select
    value: !join:
      - '|'
      -
          - foo
          - bar
          - select
    visible-items: 2
    multi-select-delimiter: '|'
    default-value: foo
- extended-choice:
    name: JSON
    type: json
    groovy-script: >-
        import net.sf.json.JSONObject;
        def jsonEditorOptions = JSONObject.fromObject(/{schema:
        {"type": "object", "title": "Name", "properties":
        {"name": {"type": "string", "propertyOrder" : 1}}}}/);
- extended-choice:
    name: MULTILEVELMULTISELECT
    type: multi-level-multi-select
    value: !join:
        - ','
        -
            - foo
            - bar
```

(continues on next page)

(continued from previous page)

```
            - baz
  - extended-choice:
      name: MULTILEVELSINGLESELECT
      type: multi-level-single-select
      value: foo
```

Inclusion Tags

These allow inclusion of arbitrary files as a method of having blocks of data managed separately to the yaml job configurations. A specific usage of this is inlining scripts contained in separate files, although such tags may also be used to simplify usage of macros or job templates.

The tag `!include:` will treat the following string as file which should be parsed as yaml configuration data.

Example:

```
- job:
    name: test-job-1
    builders:
      !include: include001.yaml.inc
```

contents of include001.yaml.inc:

```
- timeout-wrapper
- pre-scm-shell-ant
- copy-files
```

The tag `!include-raw:` will treat the given string or list of strings as filenames to be opened as one or more data blob, which should be read into the calling yaml construct without any further parsing. Any data in a file included through this tag, will be treated as string data.

Examples:

```
- job:
    name: test-job-include-raw-1
    builders:
      - shell:
          !include-raw: include-raw001-hello-world.sh
      - shell:
          !include-raw: include-raw001-vars.sh
```

contents of include-raw001-hello-world.sh:

```
#!/bin/bash
#
# Sample script showing how the yaml include-raw tag can be used
# to inline scripts that are maintained outside of the jenkins
# job yaml configuration.

echo "hello world"

exit 0
```

contents of include-raw001-vars.sh:

```
#!/bin/bash
#
# sample script to check that brackets aren't escaped
# when using the include-raw application yaml tag

VAR1="hello"
VAR2="world"
VAR3="${VAR1} ${VAR2}"

[[ -n "${VAR3}" ]] && {
    # this next section is executed as one
    echo "${VAR3}"
    exit 0
}
```

using a list of files:

```
- job:
    name: test-job-include-raw-1
    builders:
      - shell:
          !include-raw:
              - include-raw001-hello-world.sh
              - include-raw001-vars.sh
```

The tag `!include-raw-escape:` treats the given string or list of strings as filenames to be opened as one or more data blobs, which should be escaped before being read in as string data. This allows job-templates to use this tag to include scripts from files without needing to escape braces in the original file.

Warning: When used as a macro `!include-raw-escape:` should only be used if parameters are passed into the escaped file and you would like to escape those parameters. If the file does not have any jjb parameters passed into it then `!include-raw:` should be used instead otherwise you will run into an interesting issue where `include-raw-escape:` actually adds additional curly braces around existing curly braces. For example ${PROJECT} becomes ${{PROJECT}} which may break bash scripts.

Examples:

```
- job-template:
    name: test-job-include-raw-{num}
    builders:
      - shell:
          !include-raw-escape: include-raw001-hello-world.sh
      - shell:
          !include-raw-escape: include-raw001-vars.sh

- project:
    name: test-job-template-1
    num: 1
    jobs:
      - 'test-job-include-raw-{num}'
```

contents of include-raw001-hello-world.sh:

```
#!/bin/bash
#
# Sample script showing how the yaml include-raw tag can be used
# to inline scripts that are maintained outside of the jenkins
# job yaml configuration.

echo "hello world"

exit 0
```

contents of include-raw001-vars.sh:

```
#!/bin/bash
#
# sample script to check that brackets aren't escaped
# when using the include-raw application yaml tag

VAR1="hello"
VAR2="world"
VAR3="${VAR1} ${VAR2}"

[[ -n "${VAR3}" ]] && {
    # this next section is executed as one
    echo "${VAR3}"
    exit 0
}
```

using a list of files:

```
- job-template:
    name: test-job-include-raw-{num}
    builders:
      - shell:
          !include-raw-escape:
              - include-raw001-hello-world.sh
              - include-raw001-vars.sh

- project:
    name: test-job-template-1
    num: 1
    jobs:
      - 'test-job-include-raw-{num}'
```

For all the multi file includes, the files are simply appended using a newline character.

To allow for job templates to perform substitution on the path names, when a filename containing a python format placeholder is encountered, lazy loading support is enabled, where instead of returning the contents back during yaml parsing, it is delayed until the variable substitution is performed.

Example:

```
- wrapper:
    !include: lazy-load-jobs-timeout.yaml.inc
```

(continues on next page)

(continued from previous page)

```
- project:
    name: test
    version:
      - 1.1
    jobs:
      - 'build_myproject_{version}'

- job-template:
    name: 'build_myproject_{version}'
    wrappers:
      !include: lazy-load-wrappers-{version}.yaml.inc
    builders:
      - shell:
          !include-raw-escape: echo_vars_{version}.sh
```

using a list of files:

```
- wrapper:
    !include: lazy-load-jobs-timeout.yaml.inc

- project:
    name: test
    num: "002"
    version:
      - 1.1
    jobs:
      - 'build_myproject_{version}'

- job-template:
    name: 'build_myproject_{version}'
    wrappers:
      !include: lazy-load-wrappers-{version}.yaml.inc
    builders:
      - shell:
          !include-raw-escape:
            - lazy-load-scripts/echo_vars_{version}.sh
            - include-raw{num}-cool.sh
```

Note: Because lazy-loading involves performing the substitution on the file name, it means that jenkins-job-builder can not call the variable substitution on the contents of the file. This means that the `!include-raw:` tag will behave as though `!include-raw-escape:` tag was used instead whenever name substitution on the filename is to be performed.

Given the behaviour described above, when substitution is to be performed on any filename passed via `!include-raw-escape:` the tag will be automatically converted to `!include-raw:` and no escaping will be performed.

The tag `!include-jinja2:` will treat the given string or list of strings as filenames to be opened as Jinja2 templates, which should be rendered to a string and included in the calling YAML construct. (This is analogous to the templating that will happen with `!include-raw`.)

Examples:

```
- builder:
    name: test-builder
    builders:
        - shell:
            !include-jinja2: jinja01.yaml.inc

- job:
    name: test-job
    builders:
        - test-builder:
            var: "test variable"
            test_list:
                - a
                - b
                - c
```

contents of jinja01.yaml.inc:

```
{{ var }}
{% for item in test_list -%}
{{ item }}
{% endfor %}
```

The tag `!j2:` takes a string and treats it as a Jinja2 template. It will be rendered (with the variables in that context) and included in the calling YAML construct.

Examples:

```
- builder:
    name: test-builder
    builders:
        - shell:
            !j2: |
                {{ var }}
                {% for item in test_list -%}
                {{ item }}
                {% endfor %}

- job:
    name: test-job
    builders:
        - test-builder:
            var: "test variable"
            test_list:
                - a
                - b
                - c
```

The tag `!j2-yaml:` is similar to the `!j2:` tag, just that it loads the Jinja-rendered string as YAML and embeds it in the calling YAML construct. This provides a very flexible and convenient way of generating pieces of YAML structures. One of use cases is defining complex YAML structures with much simpler configuration, without any duplication.

Examples:

```
- job-template:
    name: test-job-template
    triggers:
    - gerrit:
        projects:
            !j2-yaml: |
                {% for item in triggers %}
                - branches:
                  - branch-compare-type: PLAIN
                    branch-pattern: '{{ item.branch }}'
                  project-compare-type: REG_EXP
                  project-pattern: '{{ item.repositories|join("|") }}'
                {% endfor %}

- project:
    name: test-job-project

    jobs:
    - test-job-template:
        triggers:
        - repositories:
            - a
            - b
            - c
          branch: master
        - repositories:
            - d
            - e
            - f
          branch: stable
```

Another use case is controlling lists dynamically, like conditionally adding list elements based on project configuration.

Examples:

```
- job-template:
    name: 'test-job-{variant}'
    properties: !j2-yaml: |
        - rebuild
        {% if discard_old_builds|default %}
        - build-discarder:
            days-to-keep: 7
        {% endif %}

- project:
    name: test-project

    jobs:
    - 'test-job-{variant}':
        variant: abc

    - 'test-job-{variant}':
        variant: def
        discard_old_builds: true
```

2.5.3 Modules

The bulk of the job definitions come from the following modules.

ExternalJob Project

The External Job Project module handles creating ExternalJob Jenkins projects. You may specify `externaljob` in the `project-type` attribute of the *Job* definition.

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Requires the Jenkins External Monitor Job Type Plugin.

Example:

```
name: openstack-infra
project-type: externaljob
```

class project_externaljob.**ExternalJob**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Flow Project

The flow Project module handles creating Jenkins flow projects. You may specify `flow` in the `project-type` attribute of the *Job* definition.

Requires the Jenkins Build Flow Plugin.

In order to use it for job-template you have to escape the curly braces by doubling them in the DSL: { -> {{ , otherwise it will be interpreted by the python str.format() command.

Job Parameters

- **dsl** (*str*): The DSL content. (optional)
- **needs-workspace** (*bool*): This build needs a workspace. (default false)
- **dsl-file** (*str*): Path to the DSL script in the workspace. Has effect only when *needs-workspace* is true. (optional)

Job example:

```
- job:
    name: test_job
    project-type: flow
    dsl: |
      build("job1")
      parallel (
        { build("job2a") },
        { build("job2b") }
      )
```

Job template example:

```
- job-template:
    name: '{name}-unit-tests'
    project-type: flow
    dsl: |
        build("job1")
        parallel (
            {{ build("job2a") }},
            {{ build("job2b") }}
        )
        build("job2c")
    builders:
    - shell: unittest
    publishers:
    - email:
        recipients: '{mail-to}'

- job-group:
    name: '{name}-tests'
    jobs:
    - '{name}-unit-tests':
        mail-to: developer@nowhere.net

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

Job example runninng a DSL file from the workspace:

```
- job:
     name: test_job
     project-type: flow
     needs-workspace: true
     dsl-file: script.groovy
```

class project_flow.**Flow**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Folder Project

The folder Project module handles creating Jenkins folder projects. You may specify `folder` in the `project-type` attribute of the *Job* definition.

Requires the Jenkins CloudBees Folders Plugin.

Job example:

```
- job:
    name: folder_test
    project-type: folder
```

Job template example:

```
- job-template:
    name: 'folder-{name}'
    project-type: folder

- project:
    name: test
    jobs:
    - 'folder-{name}'
```

class project_folder.**Folder**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Freestyle Project

The Freestyle Project module handles creating freestyle Jenkins projects (i.e., those that do not use Maven). You may specify `freestyle` in the `project-type` attribute to the *Job* definition if you wish, though it is the default, so you may omit `project-type` altogether if you are creating a freestyle project.

Example:

```
job:
  name: test_job
  project-type: freestyle
```

class project_freestyle.**Freestyle**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

GitHub Organization Project

The Github Organization project module handles creating Jenkins Github Organization jobs, which are made up of multibranch pipelines for all repositories containing the specified Jenkinsfile(s). You may specify `githuborg` in the `project-type` attribute of the *Job* definition.

Plugins required:

- GitHub Branch Source Plugin

Job Parameters

- **github-org** (*dict*): Refer to *github_org* for documentation.
- **periodic-folder-trigger** (*str*): How often to scan for new branches or pull/change requests. Valid values: 1m, 2m, 5m, 10m, 15m, 20m, 25m, 30m, 1h, 2h, 4h, 8h, 12h, 1d, 2d, 1w, 2w, 4w. (default none)
- **prune-dead-branches** (*bool*): If dead branches upon check should result in their job being dropped. (default true)
- **number-to-keep** (*int*): How many builds should be kept. (default '-1, all')

- **days-to-keep** (*int*): For how many days should a build be kept. (default '-1, forever')
- **script-path** (*str*): Path to Jenkinsfile, relative to workspace. (default 'Jenkinsfile')

Job examples:

```
name: github-org-minimal
project-type: githuborg
project: example-project
github-org:
    repo-owner: example-owner
```

```
name: githuborg-job-full
project-type: githuborg
project: example-project

periodic-folder-trigger: 2h
prune-dead-branches: false
number-to-keep: 10
days-to-keep: 90
script-path: some.Jenkinsfile

github-org:
    repo-owner: example-owner
```

class project_githuborg.**GithubOrganization**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

project_githuborg.**github_org**(*xml_parent*, *data*)

Configure GitHub Organization and SCM settings.

Parameters

- **repo-owner** (*str*) – Specify the name of the GitHub Organization or GitHub User Account. (required)
- **api-uri** (*str*) – The GitHub API uri for hosted / on-site GitHub. Must first be configured in Global Configuration. (default GitHub)
- **branch-discovery** (*str*) – Discovers branches on the repository. Valid options: no-pr, only-pr, all, false. (default 'no-pr')
- **build-strategies** (*list*) – Provides control over whether to build a branch (or branch like things such as change requests and tags) whenever it is discovered initially or a change from the previous revision has been detected. (optional) Refer to *~build_strategies*.
- **credentials-id** (*str*) – Credentials used to scan branches and pull requests, check out sources and mark commit statuses. (optional)
- **discover-pr-forks-strategy** (*str*) – Fork strategy. Valid options: merge-current, current, both, false. (default 'merge-current')
- **discover-pr-forks-trust** (*str*) – Discovers pull requests where the origin repository is a fork of the target repository. Valid options: contributors, everyone, permission or nobody. (default 'contributors')

- **discover-pr-origin** (*str*) – Discovers pull requests where the origin repository is the same as the target repository. Valid options: merge-current, current, both, false. (default 'merge-current')
- **discover-tags** (*bool*) – Discovers tags on the repository. (default false)
- **head-pr-filter-behaviors** (*list*) – Definition of Filter Branch PR behaviors. Requires the SCM Filter Branch PR Plugin. Refer to *~add_filter_branch_pr_behaviors*.
- **notification-context** (*dict*) – Change the default GitHub check notification context from "continuous-integration/jenkins/SUFFIX" to a custom label / suffix. (set a label and suffix to true or false, optional) Requires the Github Custom Notification Context SCM Behaviour. Refer to *~add_notification_context_trait*.
- **property-strategies** (*dict*) – Provides control over how to build a branch (like to disable SCM triggering or to override the pipeline durability) (optional) Refer to *~property_strategies*.
- **ssh-checkout** (*bool*) – Checkout over SSH.
  - **credentials ('str'): Credentials to use for** checkout of the repo over ssh.

Extensions

- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
  - **before (*dict*) - Clean the workspace before checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
- **depth** (*int*) - Set shallow clone depth (default 1)
- **disable-pr-notifications (*bool*) - Disable default github status** notifications on pull requests (default false) (Requires the GitHub Branch Source Plugin.)
- **do-not-fetch-tags (*bool*) - Perform a clone without tags** (default false)
- **lfs-pull (*bool*) - Call git lfs pull after checkout** (default false)
- **prune** (*bool*) - Prune remote branches (default false)
- **refspecs** (*list(str)*): Which refspecs to fetch.
- **shallow-clone** (*bool*) - Perform shallow clone (default false)
- **sparse-checkout (dict)**
  - **paths** (list) - List of paths to sparse checkout. (optional)
- **submodule (*dict*)**
  - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.

- **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
- **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
- **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
- **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
- **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).

- **timeout** (*str*) - Timeout for git commands in minutes (optional)
- **use-author (*bool*): Use author rather than committer in Jenkin's** build changeset (default false)
- **wipe-workspace (*bool*) - Wipe out workspace before build** (default true)

Job examples:

```
name: github-org-minimal
project-type: githuborg
project: example-project
github-org:
    repo-owner: example-owner
```

```
name: github-org-full
project-type: githuborg
github-org:
        api-uri: http://example.org/github
        ssh-checkout:
            credentials: 'ssh_secret'
        repo-owner: example-owner
        credentials-id: example-credential
        branch-discovery: all
        head-filter-regex: "(.*/master|.*/release/.*)"
        head-pr-filter-behaviors:
            head-pr-destined-regex:
                branch-regexp: "foo/.*"
                tag-regexp: "20\\..*"
            head-pr-destined-wildcard:
                branch-includes: "foo*"
                tag-includes: "qaz*"
                branch-excludes: "bar*"
                tag-excludes: "*baz"
            head-pr-originated-regex:
                branch-regexp: "(foo/.*|bar/.*)"
                tag-regexp: "1\\..*"
            head-pr-originated-wildcard:
                branch-includes: "qaz*"
                tag-includes: "bar*"
                branch-excludes: "baz*"
                tag-excludes: "*qaz"
```

(continues on next page)

(continued from previous page)

```
        discover-pr-forks-strategy: both
        discover-pr-forks-trust: everyone
        discover-pr-origin: both
        discover-tags: true
        notification-context:
            label: 'jenkins.example.com/my_context'
            suffix: false
        property-strategies:
            all-branches:
                - suppress-scm-triggering: true
                - pipeline-branch-durability-override: max-survivability
                - trigger-build-on-pr-comment: "Ci build!"
                - trigger-build-on-pr-review: true
                - trigger-build-on-pr-update: true
        build-strategies:
            - all-strategies-match:
                strategies:
                  - regular-branches: true
                  - skip-initial-build: true
            - any-strategies-match:
                strategies:
                  - change-request: {}
                  - tags: {}
            - tags:
                ignore-tags-newer-than: 1
                ignore-tags-older-than: 7
            - tags: {}
            - change-request:
                ignore-target-only-changes: true
            - change-request: {}
            - regular-branches: true
            - skip-initial-build: true
            - named-branches:
                - exact-name:
                    name: 'test'
                    case-sensitive: true
                - regex-name:
                    regex: 'test.*$'
                    case-sensitive: true
                - wildcards-name:
                    excludes: 'testexclude'
                    includes: 'testinclude'
            - named-branches:
                - exact-name: {}
                - regex-name: {}
                - wildcards-name: {}
        clean:
            after: true
            before: true
        committer:
            user: CI System
            email: no-reply@ci.example.com
```

(continues on next page)

(continued from previous page)

```
        prune: true
        local-branch: true
        sparse-checkout:
            paths:
                - "path1"
                - "path2"
                - "path3"
        shallow-clone: true
        depth: 3
        do-not-fetch-tags: true
        disable-pr-notifications: true
        refspecs:
          - '+refs/heads/*:refs/remotes/@{remote}/*'
        submodule:
            disable: false
            recursive: true
            parent-credentials: true
            timeout: 100
            threads: 1
        timeout: "100"
        skip-notifications: true
        use-author: true
        wipe-workspace: true
        lfs-pull: true
```

Matrix Project

The matrix project module handles creating Jenkins matrix projects. To create a matrix project specify `matrix` in the `project-type` attribute to the *Job* definition. Currently it supports four axes which share the same internal YAML structure:

- label expressions (`label-expression`)
- user-defined values (`user-defined`)
- slave name or label (`slave`)
- JDK name (`jdk`)

Requires the Jenkins Matrix Project Plugin.

The module also supports additional, plugin-defined axes:

- DynamicAxis (`dynamic`), requires the Jenkins DynamicAxis Plugin
- GroovyAxis (`groovy`), requires the Jenkins GroovyAxis Plugin
- YamlAxis (`yaml`), requires the Jenkins Yaml Axis Plugin

To tie the parent job to a specific node, you should use `node` parameter. On a matrix project, this will tie *only* the parent job. To restrict axes jobs, you can define a single value `slave` axis.

Job Parameters

> **Note:** You can only pick one of the strategies.

- **execution-strategy (optional, built in Jenkins):**
  - **combination-filter** (*str*): axes selection filter
  - **sequential** (*bool*): run builds sequentially (default false)
  - **touchstone (optional):**
    - **expr** (*str*) – selection filter for the touchstone build
    - **result** (*str*) – required result of the job: stable (default) or unstable
- **yaml-strategy** (optional, requires Yaml Axis Plugin):
  - **exclude-key** (*str*) – top key containing exclusion rules
  - Either one of:
  - **filename** (*str*) – Yaml file containing exclusions
  - **text** (*str*) – Inlined Yaml. Should be literal `text: | exclude:...`
- **axes (*list*):**
  - **axis:**
    - **type** (*str*) – axis type, must be either type defined by Matrix Project Plugin (`label-expression`, `user-defined`, `slave` or `jdk`) or a type defined by a plugin (see top of this document for a list of supported plugins).
    - **name** (*str*) – name of the axis
    - **values** (*list*) – values of the axis

The module supports also ShiningPanda axes:

Example:

```
name: matrix-test003
project-type: matrix
axes:
    - axis:
        type: python
        values:
        - python-2.6
        - python-2.7
        - python-3.4
    - axis:
        type: tox
        values:
        - py26
        - py27
        - py34
```

Requires the Jenkins ShiningPanda Plugin.

Example:

```
- job:
    name: matrix-test
    project-type: matrix
    execution-strategy:
```

(continues on next page)

(continued from previous page)

```
      combination-filter: |
        !(os=="fedora11" && arch=="amd64")
      sequential: true
      touchstone:
        expr: 'os == "fedora11"'
        result: unstable
    axes:
      - axis:
         type: label-expression
         name: os
         values:
          - ubuntu12.04
          - fedora11
      - axis:
         type: label-expression
         name: arch
         values:
          - amd64
          - i386
      - axis:
         type: slave
         name: nodes
         values:
          - node1
          - node2
      - axis:
         type: dynamic
         name: config
         values:
          - config_list
      - axis:
         type: user-defined
         name: database
         values:
          - mysql
          - postgresql
          - sqlite
      - axis:
         type: groovy
         name: foo
         command: return [one,two,three]
    builders:
      - shell: make && make check
```

Examples for yaml axis:

```
name: matrix-with-yaml-axis
project-type: matrix
axes:
    - axis:
        type: yaml
        filename: config.yaml
```

(continues on next page)

(continued from previous page)

```
        name: python
    - axis:
        type: yaml
        filename: config.yaml
        name: database
```

```
name: matrix-with-yaml-strategy-and-exclude-in-file
project-type: matrix
yaml-strategy:
    exclude-key: 'exclude'
    filename: 'exclude.yaml'
axes:
    - axis:
        type: yaml
        filename: 'config.yaml'
        name: python
    - axis:
        type: yaml
        filename: 'config.yaml'
        name: database
```

```
name: matrix-with-yaml-strategy-and-inlined-exclude
project-type: matrix
yaml-strategy:
    exclude-key: 'exclude'
    text: |
        exclude:
            - database: postgre
              python: py27
            - python: py35
              database: mysql
axes:
    - axis:
        type: yaml
        filename: config.yaml
        name: python
    - axis:
        type: yaml
        filename: config.yaml
        name: database
```

class project_matrix.**Matrix**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Maven Project

The Maven Project module handles creating Maven Jenkins projects.

To create a Maven project, specify `maven` in the `project-type` attribute to the *Job* definition. It also requires a `maven` section in the *Job* definition.

Job Parameters

- **root-module:**
  - **group-id** (*str*): GroupId.
  - **artifact-id** (*str*): ArtifactId.
- **root-pom** (*str*): The path to the pom.xml file. (default 'pom.xml')
- **goals** (*str*): Goals to execute. (required)
- **maven-opts** (*str*): Java options to pass to maven (aka MAVEN_OPTS)
- **maven-name** (*str*): Installation of maven which should be used. Not setting `maven-name` appears to use the first maven install defined in the global jenkins config.
- **private-repository** ('str'): Whether to use a private maven repository Possible values are *default*, *local-to-workspace* and *local-to-executor*.
- **ignore-upstream-changes** (*bool*): Do not start a build whenever a SNAPSHOT dependency is built or not. (default true)
- **incremental-build** (*bool*): Activate incremental build - only build changed modules (default false).
- **automatic-archiving** (*bool*): Activate automatic artifact archiving (default true).
- **automatic-site-archiving** (*bool*): Activate automatic site documentation artifact archiving (default true).
- **automatic-fingerprinting** (*bool*): Activate automatic fingerprinting of consumed and produced artifacts (default true).
- **per-module-email** (*bool*): Send an e-mail for each failed module (default true).
- **parallel-build-modules** (*bool*): Build modules in parallel (default false)
- **resolve-dependencies** (*bool*): Resolve Dependencies during Pom parsing (default false).
- **run-headless** (*bool*): Run headless (default false).
- **disable-downstream** (*bool*): Disable triggering of downstream projects (default false).
- **process-plugins** (*bool*): Process Plugins during Pom parsing (default false).
- **custom-workspace** (*str*): Path to the custom workspace. If no path is provided, custom workspace is not used. (optional)
- **settings** (*str*): Path to custom maven settings file. If settings type is 'file' then this is a Path. Otherwise it is the id for ConfigFileProvider. (optional)
- **settings-type** (*str*): Type of settings file file|cfp. (default file)
- **global-settings** (*str*): Path to custom maven global settings file. If settings type is 'file' then this is a Path. Otherwise it is the id for ConfigFileProvider. (optional)
- **global-settings-type** (*str*): Type of settings file file|cfp. (default file)

- **post-step-run-condition** (*str*): Run the post-build steps only if the build succeeds ('SUCCESS'), build succeeds or is unstable ('UNSTABLE'), regardless of build result ('FAILURE'). (default 'FAILURE').

Requires the Jenkins Config File Provider Plugin for the Config File Provider "settings" and "global-settings" config.

Example:

```
project-type: maven
maven:
  root-pom: pom.xml
  goals: deploy
  root-module:
    group-id: gabba.gabba
    artifact-id: hey
  settings: test
  global-settings: test
  incremental-build: true
  automatic-archiving: false
  automatic-site-archiving: false
  parallel-build-modules: true
  resolve-dependencies: true
  process-plugins: true
  run-headless: true
  disable-downstream: true
  custom-workspace: path/to/some/repository
```

CFP Example:

```
project-type: maven
maven:
  root-pom: pom.xml
  goals: deploy
  settings: org.jenkinsci.plugins.configfiles.maven.
↪MavenSettingsConfig0123456789012
  global-settings: org.jenkinsci.plugins.configfiles.maven.
↪GlobalMavenSettingsConfig0123456789012
  post-step-run-condition: SUCCESS
```

class project_maven.**Maven**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

MultiJob Project

The MultiJob Project module handles creating MultiJob Jenkins projects. You may specify `multijob` in the `project-type` attribute of the *Job* definition.

This project type may use `jenkins_jobs.modules.builders.multijob()` builders.

Requires the Jenkins Multijob Plugin.

Example:

```
job:
  name: test_job
  project-type: multijob
  builders:
    - multijob:
        name: PhaseOne
        condition: SUCCESSFUL
        projects:
          - name: PhaseOneJobA
            current-parameters: true
            git-revision: true
          - name: PhaseOneJobB
            current-parameters: true
            property-file: build.props
    - multijob:
        name: PhaseTwo
        condition: UNSTABLE
        projects:
          - name: PhaseTwoJobA
            current-parameters: true
            predefined-parameters: foo=bar
          - name: PhaseTwoJobB
            current-parameters: false
```

class project_multijob.**MultiJob**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Pipeline Project

The Pipeline Project module handles creating Jenkins Pipeline projects (formerly known as the Workflow projects). You may specify `pipeline` in the `project-type` attribute of the *Job* definition.

Requires the Jenkins Pipeline Plugin.

In order to write an inline script within a job-template you have to escape the curly braces by doubling them in the DSL: { -> {{ , otherwise it will be interpreted by the python str.format() command.

Job Parameters

- **sandbox** (*bool*): If the script should run in a sandbox (default false)
- **dsl** (*str*): The DSL content or,
- **pipeline-scm** (*str*): in case "Pipeline as code" feature is used. Then you should specify:

- **scm**: single `scm` component (or a reference) describing the source code repository
- **script-path**: path to the Groovy file containing the job's steps (optional, default: `Jenkinsfile`)
- **lightweight-checkout** (*bool*): If selected, try to obtain the Pipeline script contents directly from the SCM without performing a full checkout. (optional, default: `false`)

Note that `dsl` and `pipeline-scm` parameters are mutually exclusive.

Inline DSL job example:

```
- job:
    name: test_job
    project-type: pipeline
    dsl: |
      build job: "job1"
      parallel [
        2a: build job: "job2a",
        2b: node "dummynode" {
          sh "echo I'm alive!"
        }
      ]
```

Inline DSL job template example:

```
- job-template:
    name: '{name}-unit-tests'
    project-type: pipeline
    dsl: |
        build job: "job1"
        parallel [
          2a: build job: "job2a",
          2b: node "dummynode" {{
            sh "echo {isay}"
          }}
        ]

- job-group:
    name: '{name}-tests'
    jobs:
      - '{name}-unit-tests':
          isay: 'hello'

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

"Pipeline as code" example:

```
- job:
    name: test-job
    project-type: pipeline
    sandbox: true
    pipeline-scm:
```

(continues on next page)

(continued from previous page)

```
      scm:
        - hg:
            url: http://hg.example.org/test_job
            clean: true
      script-path: Jenkinsfile.groovy
      lightweight-checkout: true
```

"Pipeline as code" example using templates:

```
- scm:
    name: project-scm
    scm:
      - hg:
          url: http://hg.example.org/project
          clean: true

- job-template:
    name: '{name}-unit-tests'
    project-type: pipeline
    pipeline-scm:
      scm:
        - project-scm
    sandbox: true
    description: 'maintainer: {maintainer}'

- job-template:
    name: '{name}-perf-tests'
    project-type: pipeline
    pipeline-scm:
      scm:
        - project-scm
    sandbox: false
    description: 'maintainer: {maintainer}'

- job-group:
    name: '{name}-tests'
    jobs:
      - '{name}-unit-tests':
          maintainer: dev@example.org
      - '{name}-perf-tests':
          maintainer: qa@example.org

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

"Pipeline as nested stage" example :

```
- job-template:
    name: '{name}-unit-tests'
    project-type: pipeline
```

(continues on next page)

(continued from previous page)

```
    dsl: |
      stage('Build another job') {{
        build(job: "{isay}")
      }}

- job-group:
    name: '{name}-tests'
    jobs:
      - '{name}-unit-tests':
          isay: 'hello'

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

class project_pipeline.**Pipeline**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Workflow Project

Deprecated: please use *Pipeline Project* instead.

The workflow Project module handles creating Jenkins workflow projects. You may specify `workflow` in the `project-type` attribute of the *Job* definition. For now only inline scripts are supported.

Requires the Jenkins Workflow Plugin.

In order to use it for job-template you have to escape the curly braces by doubling them in the DSL: { -> {{ , otherwise it will be interpreted by the python str.format() command.

Job Parameters

- **dsl** (*str*): The DSL content.
- **sandbox** (*bool*): If the script should run in a sandbox (default false)

Job example:

```
- job:
    name: test_job
    project-type: workflow
    dsl: |
      build job: "job1"
      parallel [
        2a: build job: "job2a",
        2b: node "dummynode" {
          sh "echo I'm alive!"
        }
      ]
```

Job template example:

```
- job-template:
    name: '{name}-unit-tests'
    project-type: workflow
    dsl: |
        build job: "job1"
        parallel [
          2a: build job: "job2a",
          2b: node "dummynode" {{
            sh "echo {isay}"
          }}
        ]

- job-group:
    name: '{name}-tests'
    jobs:
      - '{name}-unit-tests':
          isay: 'hello'

- project:
    name: project-name
    jobs:
    - '{name}-tests'
```

class project_workflow.**Workflow**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Multibranch Pipeline Project

The Multibranch Pipeline project module handles creating Jenkins workflow projects. You may specify `multibranch` in the `project-type` attribute of the *Job* definition.

Multibranch Pipeline implementantion in JJB is marked as **experimental** which means that there is no guarantee that its behavior (or configuration) will not change, even between minor releases.

Plugins required:

- Workflow Plugin.
- Pipeline Multibranch Defaults Plugin (optional)
- Basic Branch Build Strategies Plugin (optional)

Job Parameters

- **scm** (*list*): The SCM definition.
  - **bitbucket** (*dict*): Refer to *~bitbucket_scm* for documentation.
  - **gerrit** (*dict*): Refer to *~gerrit_scm* for documentation.
  - **git** (*dict*): Refer to *~git_scm* for documentation.
  - **github** (*dict*): Refer to *~github_scm* for documentation.

- **periodic-folder-trigger** (*str*): How often to scan for new branches or pull/change requests. Valid values: 1m, 2m, 5m, 10m, 15m, 20m, 25m, 30m, 1h, 2h, 4h, 8h, 12h, 1d, 2d, 1w, 2w, 4w. (default none)
- **prune-dead-branches** (*bool*): If dead branches upon check should result in their job being dropped. (default true)
- **number-to-keep** (*int*): How many builds should be kept. (default '-1, all')
- **days-to-keep** (*int*): For how many days should a build be kept. (default '-1, forever')
- **abort-builds** (*bool*): Abort all pending or ongoing builds for removed SCM heads (i.e. deleted branches). (default false)
- **script-path** (*str*): Path to Jenkinsfile, relative to workspace. (default 'Jenkinsfile')
- **script-id** (*str*): Script id from the global Jenkins script store provided by the config-file provider plugin. Mutually exclusive with **script-path** option.
- **sandbox** (*bool*): This option is strongly recommended if the Jenkinsfile is using load to evaluate a groovy source file from an SCM repository. Usable only with **script-id** option. (default 'false')

Job examples:

```
name: 'demo-multibranch-defaults'
project-type: multibranch-defaults
script-id: my-pipeline
sandbox: true
scm:
    - github:
        repo: 'foo'
        repo-owner: 'johndoe'
        credentials-id: 'secret'
```

```
name: 'demo-multibranch-defaults'
project-type: multibranch-defaults
scm:
    - github:
        repo: 'foo'
        repo-owner: 'johndoe'
        credentials-id: 'secret'
```

```
name: 'demo-multibranch-multi-scm-full'
description: 'Workflow demo'

project-type: multibranch

periodic-folder-trigger: 1d
prune-dead-branches: True
number-to-keep: '10'
days-to-keep: '10'
abort-builds: True
script-path: 'some.Jenkinsfile'
scm:
    - bitbucket:
        repo-owner: 'SANDBOX'
```

(continues on next page)

(continued from previous page)

```
        repo: 'test'
        credentials-id: 'secret'
    - git:
        url: 'https://example.com/jonhndoe/keep-frontend.git'
        credentials-id: 'secret'
    - github:
        repo: 'foo'
        repo-owner: 'johndoe'
        credentials-id: 'secret'
```

class project_multibranch.**WorkflowMultiBranch**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

class project_multibranch.**WorkflowMultiBranchDefaults**(*registry*)

project_multibranch.**add_filter_branch_pr_behaviors**(*traits*, *data*)

Configure Filter Branch PR behaviors

Requires the SCM Filter Branch PR Plugin.

Parameters

head-pr-filter-behaviors (*list*) – Definition of filters.

- **head-pr-destined-regex (dict): Filter by name incl. PR destined to** this branch with regexp
  - **branch-regexp (str) Regular expression to filter branches and** PRs (optional, default ".*")
  - **tag-regexp (str) Regular expression to filter tags** (optional, default "(?!.*)")
- **head-pr-destined-wildcard (dict): Filter by name incl. PR** destined to this branch with wildcard
  - **branch-includes (str) Wildcard expression to include branches** and PRs (optional, default "*")
  - **tag-includes (str) Wildcard expression to include tags** (optional, default "")
  - **branch-excludes (str) Wildcard expression to exclude branches** and PRs (optional, default "")
  - **tag-excludes (str) Wildcard expression to exclude tags** (optional, default "*")
- **head-pr-originated-regex (dict): Filter by name incl. PR destined** to this branch with regexp
  - **branch-regexp (str) Regular expression to filter branches** and PRs (optional, default ".*")
  - **tag-regexp (str) Regular expression to filter tags** (optional, default "(?!.*)")
- **head-pr-originated-wildcard (dict): Filter by name incl. PR** destined to this branch with wildcard

- **branch-includes (str) Wildcard expression to include branches** and PRs (optional, default "*")
- **tag-includes (str) Wildcard expression to include tags** (optional, default "")
- **branch-excludes (str) Wildcard expression to exclude branches** and PRs (optional, default "")
- **tag-excludes (str) Wildcard expression to exclude tags** (optional, default "*")

project_multibranch.**add_filter_by_name_wildcard_behaviors**(*traits*, *data*)

Configure branch filtering behaviors.

Parameters

filter-by-name-wildcard (*dict*) – Enable filter by name with wildcards. Requires the SCM API Plugin.

- **includes ('str'): Space-separated list** of name patterns to consider. You may use * as a wildcard; for example: *master release**
- **excludes ('str'): Name patterns to** ignore even if matched by the includes list. For example: *release**

project_multibranch.**add_github_checks_traits**(*traits*, *data*)

Enable and configure the usage of GitHub Checks API

Requires the Github Checks plugin.

Parameters

status-checks (*dict*) –

- **name** (str): The text of the context label for GitHub Checks entry
- **skip** (bool): Skips publishing Checks (optional, default false)
- **skip-branch-source-notifications (bool): Disables the default** option of publishing statuses through Status API (optional, default false)
- **publish-unstable-as-neutral (bool): Publishes UNSTABLE builds** as neutral (not failed) checks (optional, default false)
- **suppress-log-output (bool): Suppresses sending build logs to GitHub** (optional, default false)
- **suppress-progress-updates (bool): Suppresses updating build progress** (optional, default false)
- **verbose-logs (bool): Enables sending build console logs to GitHub** (optional, default false)

project_multibranch.**add_notification_context_trait**(*traits*, *data*)

Change the default GitHub check notification context from "continuous-integration/jenkins/SUFFIX" to a custom label / suffix. (set a label and suffix to true or false, optional)

Requires the Github Custom Notification Context SCM Behaviour.

Parameters

notification-context (*dict*) – Definition of notification-context. A *label* must be specified. *suffix* may be specified with true / false, default being true.

- **label (str): The text of the context label for Github status** notifications.

- **suffix (bool): Appends the relevant suffix to the context label**
  based on the build type. '/pr-merge', '/pr-head' or '/branch' (optional, default true)

project_multibranch.**bitbucket_scm**(*xml_parent*, *data*)

Configure BitBucket scm

Requires the Bitbucket Branch Source Plugin.

Parameters

- **credentials-id** (*str*) – The credential to use to scan BitBucket. (required)
- **repo-owner** (*str*) – Specify the name of the Bitbucket Team or Bitbucket User Account. (required)
- **repo** (*str*) – The BitBucket repo. (required)
- **discover-tags** (*bool*) – Discovers tags on the repository. (default false)
- **lfs** (*bool*) – Git LFS pull after checkout. (default false)
- **server-url** (*str*) – The address of the bitbucket server. (optional)
- **head-filter-regex** (*str*) – A regular expression for filtering discovered source branches. Requires the SCM API Plugin.
- **head-pr-filter-behaviors** (*list*) – Definition of Filter Branch PR behaviors. Requires the SCM Filter Branch PR Plugin. Refer to *~add_filter_branch_pr_behaviors*.
- **discover-branch** (*str*) – Discovers branches on the repository. Valid options: ex-pr, only-pr, all. Value is not specified by default.
- **discover-pr-origin** (*str*) – Discovers pull requests where the origin repository is the same as the target repository. Valid options: mergeOnly, headOnly, mergeAndHead. Value is not specified by default.
- **discover-pr-forks-strategy** (*str*) – Fork strategy. Valid options: merge-current, current, both, false. (default 'merge-current')
- **discover-pr-forks-trust** (*str*) – Discovers pull requests where the origin repository is a fork of the target repository. Valid options: contributors, everyone, permission or nobody. (default 'contributors')
- **build-strategies** (*list*) – Provides control over whether to build a branch (or branch like things such as change requests and tags) whenever it is discovered initially or a change from the previous revision has been detected. (optional) Refer to *~build_strategies*.
- **property-strategies** (*dict*) – Provides control over how to build a branch (like to disable SCM triggering or to override the pipeline durability) (optional) Refer to *~property_strategies*.
- **local-branch** (*bool*) – Check out to matching local branch If given, checkout the revision to build as HEAD on this branch. If selected, then the branch name is computed from the remote branch without the origin. In that case, a remote branch origin/master will be checked out to a local branch named master, and a remote branch origin/develop/new-feature will be checked out to a local branch named develop/newfeature. Requires the Git Plugin.
- **refspecs** (*list(str)*) – Which refspecs to look for.
- **checkout-over-ssh** (*dict*) – Checkout repo over ssh.

- **credentials ('str'): Credentials to use for**
  checkout of the repo over ssh.

- `filter-by-name-wildcard` (*dict*) – Enable filter by name with wildcards. Requires the SCM API Plugin.
  - **includes ('str'): Space-separated list**
    of name patterns to consider. You may use * as a wildcard; for example: *master release**
  - **excludes ('str'): Name patterns to**
    ignore even if matched by the includes list. For example: *release**

Extensions

- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
  - **before (*dict*) - Clean the workspace before checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
- **prune** (*bool*) - Prune remote branches (default false)
- **shallow-clone** (*bool*) - Perform shallow clone (default false)
- **sparse-checkout (dict)**
  - **paths** (list) - List of paths to sparse checkout. (optional)
- **depth** (*int*) - Set shallow clone depth (default 1)
- **do-not-fetch-tags (*bool*) - Perform a clone without tags**
  (default false)
- **submodule (*dict*)**
  - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.
  - **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
  - **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
  - **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
  - **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
  - **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).
- **timeout** (*str*) - Timeout for git commands in minutes (optional)
- **use-author (*bool*): Use author rather than committer in Jenkin's**
  build changeset (default false)

- **wipe-workspace (*bool*) - Wipe out workspace before build**
  (default true)
- **lfs-pull (*bool*) - Call git lfs pull after checkout**
  (default false)

Minimal Example:

```
name: 'demo-multibranch-bitbucket-min'
project-type: multibranch
scm:
    - bitbucket:
        repo-owner: 'SANDBOX'
        repo: 'test'
```

Full Example:

```
name: 'demo-multibranch-bitbucket-min'
project-type: multibranch
script-path: 'some.Jenkinsfile'
scm:
    - bitbucket:
        credentials-id: 'secret'
        repo-owner: 'SANDBOX'
        repo: 'test'
        server-url: https://bitbucket.example.com:8080
        discover-tags: true
        lfs: true
        head-filter-regex: 'master|\d+\.\d+'
        head-pr-filter-behaviors:
            head-pr-destined-regex:
                branch-regexp: "foo/.*"
                tag-regexp: "20\\..*"
            head-pr-destined-wildcard:
                branch-includes: "foo*"
                tag-includes: "qaz*"
                branch-excludes: "bar*"
                tag-excludes: "*baz"
            head-pr-originated-regex:
                branch-regexp: "(foo/.*|bar/.*)"
                tag-regexp: "1\\..*"
            head-pr-originated-wildcard:
                branch-includes: "qaz*"
                tag-includes: "bar*"
                branch-excludes: "baz*"
                tag-excludes: "*qaz"
        discover-pr-origin: headOnly
        discover-branch: all
        discover-pr-forks-strategy: current
        discover-pr-forks-trust: everyone
        local-branch: true
        checkout-over-ssh:
            credentials: 'ssh_secret'
        filter-by-name-wildcard:
            includes: '*'
```

(continues on next page)

(continued from previous page)

```
            excludes: 'master'
        property-strategies:
            all-branches:
                - suppress-scm-triggering: true
                - pipeline-branch-durability-override: max-survivability
        refspecs:
            - '+refs/heads/*:refs/remotes/@{remote}/*'
            - '+refs/tags/*:refs/remotes/@{remote}/*'
        build-strategies:
            - all-strategies-match:
                strategies:
                  - regular-branches: true
                  - skip-initial-build: true
            - any-strategies-match:
                strategies:
                  - change-request: {}
                  - tags: {}
            - tags:
                ignore-tags-newer-than: 1
                ignore-tags-older-than: 7
            - tags: {}
            - change-request:
                ignore-target-only-changes: true
            - change-request: {}
            - regular-branches: true
            - skip-initial-build: true
            - named-branches:
                - exact-name:
                    name: 'test'
                    case-sensitive: true
                - regex-name:
                    regex: 'test.*$'
                    case-sensitive: true
                - wildcards-name:
                    excludes: 'testexclude'
                    includes: 'testinclude'
            - named-branches:
                - exact-name: {}
                - regex-name: {}
                - wildcards-name: {}
        clean:
            after: true
            before: true
        committer:
            user: CI System
            email: no-reply@ci.example.com
        prune: true
        sparse-checkout:
            paths:
                - "path1"
                - "path2"
                - "path3"
```

(continues on next page)

(continued from previous page)

```
        shallow-clone: true
        depth: 3
        do-not-fetch-tags: true
        submodule:
            disable: false
            recursive: true
            parent-credentials: true
            timeout: 100
            threads: 1
        timeout: "100"
        skip-notifications: true
        use-author: true
        wipe-workspace: true
        lfs-pull: true
```

project_multibranch.**build_strategies**(*xml_parent*, *data*)

Configure Basic Branch Build Strategies.

Requires the Basic Branch Build Strategies Plugin.

Other build strategies can be configured via raw XML injection.

Parameters

build-strategies (*list*) – Definition of build strategies.

- **all-strategies-match (dict): All sub strategies must match for** this strategy to match. * **strategies** (list): Sub strategies
- **any-strategies-match (dict): Builds whenever any of the sub** strategies match. * **strategies** (list): Sub strategies
- **tags (dict): Builds tags**
  - **ignore-tags-newer-than (int) The number of days since the tag** was created before it is eligible for automatic building. (optional, default -1)
  - **ignore-tags-older-than (int) The number of days since the tag** was created after which it is no longer eligible for automatic building. (optional, default -1)
- **change-request (dict): Builds change requests / pull requests**
  - **ignore-target-only-changes (bool) Ignore rebuilding merge** branches when only the target branch changed. (optional, default false)
- **regular-branches (bool): Builds regular branches whenever a** change is detected. (optional, default None)
- **skip-initial-build (bool): Skip initial build on first branch** indexing. (optional, default None)
- **named-branches** (list): Builds named branches whenever a change is detected.
  - **exact-name (dict) Matches the name verbatim.**
    - **name** (str) The name to match. (optional)
    - **case-sensitive (bool) Check this box if the name should** be matched case sensitively. (default false)
  - **regex-name** (dict) Matches the name against a regular expression.

        * **regex (str) A Java regular expression to restrict the** names. Names that do not match the supplied regular expression will be ignored. (default ^.*$)
        * **case-sensitive (bool) Check this box if the name should** be matched case sensitively. (default false)
      - **wildcards-name** (dict) Matches the name against an include/exclude set of wildcards.
        * **includes (str) Space-separated list of name patterns to** consider. You may use * as a wildcard; for example: *master release** (default *)
        * **excludes (str) Name patterns to ignore even if matched** by the includes list. For example: release (optional)
    - **raw (dict): Injects raw BuildStrategy XML to use other build** strategy plugins.

project_multibranch.**gerrit_scm**(*xml_parent*, *data*)

Configure Gerrit SCM

Requires the Gerrit Code Review Plugin.

Parameters

- **url** (*str*) – The git url. (required)
- **credentials-id** (*str*) – The credential to use to connect to the GIT URL.
- **ignore-on-push-notifications** (*bool*) – If a job should not trigger upon push notifications. (default false)
- **refspecs** (*list(str)*) – Which refspecs to look for. (default `['+refs/changes/*:refs/remotes/@{remote}/*', '+refs/heads/*:refs/remotes/@{remote}/*']`)
- **includes** (*str*) – Comma-separated list of branches to be included. (default '*')
- **excludes** (*str*) – Comma-separated list of branches to be excluded. (default '')
- **head-filter-regex** (*str*) – A regular expression for filtering discovered source branches. Requires the SCM API Plugin.
- **build-strategies** (*list*) – Provides control over whether to build a branch (or branch like things such as change requests and tags) whenever it is discovered initially or a change from the previous revision has been detected. (optional) Refer to *~build_strategies*.
- **property-strategies** (*dict*) – Provides control over how to build a branch (like to disable SCM triggering or to override the pipeline durability) (optional) Refer to *~property_strategies*.
- **filter-checks** (*dict*) – Enable the filtering by pending checks, allowing to discover the changes that need validation only. This feature is using the gerrit checks plugin. (optional) query-operator: Name of the query operator, supported values are: 'SCHEME' or 'ID'. query-string: Value of the query operator.
- **change-discovery** (*dict*) – Configure the query string in 'Discover open changes'. The default 'p:<project> status:open -age:24w' will be added prior to the query-string specified here. (optional) query-string: Value of the query operator.

Extensions

- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
  - **before (*dict*) - Clean the workspace before checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
- **prune** (*bool*) - Prune remote branches (default false)
- **shallow-clone** (*bool*) - Perform shallow clone (default false)
- **sparse-checkout (dict)**
  - **paths** (list) - List of paths to sparse checkout. (optional)
- **depth** (*int*) - Set shallow clone depth (default 1)
- **do-not-fetch-tags (*bool*) - Perform a clone without tags**
  (default false)
- **submodule (*dict*)**
  - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.
  - **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
  - **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
  - **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
  - **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
  - **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).
- **timeout** (*str*) - Timeout for git commands in minutes (optional)
- **use-author (*bool*): Use author rather than committer in Jenkin's**
  build changeset (default false)
- **wipe-workspace (*bool*) - Wipe out workspace before build**
  (default true)
- **lfs-pull (*bool*) - Call git lfs pull after checkout**
  (default false)

Minimal Example:

```
name: 'demo-multibranch-gerrit-min'
project-type: multibranch
scm:
    - gerrit:
        url: 'https://review.gerrithub.io/johndoe/foo'
```

Full Example:

```
name: 'demo-multibranch-gerrit-min'
project-type: multibranch
script-path: some.Jenkinsfile
scm:
    - gerrit:
        url: 'https://review.gerrithub.io/johndoe/foo'
        credentials-id: secret
        ignore-on-push-notifications: true
        refspecs: 'refs/heads/*'
        property-strategies:
            all-branches:
                - suppress-scm-triggering: true
                - pipeline-branch-durability-override: max-survivability
        filter-checks:
            query-operator: 'SCHEME'
            query-string: 'jenkins'
        build-strategies:
            - all-strategies-match:
                strategies:
                  - regular-branches: true
                  - skip-initial-build: true
            - any-strategies-match:
                strategies:
                  - change-request: {}
                  - tags: {}
            - tags:
                ignore-tags-newer-than: 1
                ignore-tags-older-than: 7
            - tags: {}
            - change-request:
                ignore-target-only-changes: true
            - change-request: {}
            - regular-branches: true
            - skip-initial-build: true
            - named-branches:
                - exact-name:
                    name: 'test'
                    case-sensitive: true
                - regex-name:
                    regex: 'test.*$'
                    case-sensitive: true
                - wildcards-name:
                    excludes: 'testexclude'
                    includes: 'testinclude'
            - named-branches:
                - exact-name: {}
                - regex-name: {}
                - wildcards-name: {}
        head-filter-regex: "(.*/master|.*/release/.*)"
        clean:
            after: true
            before: true
        prune: true
```

(continues on next page)

(continued from previous page)

```
        local-branch: true
        sparse-checkout:
            paths:
                - "path1"
                - "path2"
                - "path3"
        shallow-clone: true
        depth: 3
        do-not-fetch-tags: true
        submodule:
            disable: false
            recursive: true
            parent-credentials: true
            timeout: 100
            threads: 1
        timeout: "100"
        use-author: true
        wipe-workspace: true
        lfs-pull: true
```

project_multibranch.**git_scm**(*xml_parent*, *data*)

Configure Git SCM

Requires the Git Plugin.

Parameters

- **url** (*str*) – The git repo url. (required)
- **credentials-id** (*str*) – The credential to use to connect to the GIT repo. (default '')
- **discover-branches** (*bool*) – Discovers branches on the repository. (default true)
- **discover-tags** (*bool*) – Discovers tags on the repository. (default false)
- **ignore-on-push-notifications** (*bool*) – If a job should not trigger upon push notifications. (default false)
- **head-filter-regex** (*str*) – A regular expression for filtering discovered source branches. Requires the SCM API Plugin.
- **head-pr-filter-behaviors** (*list*) – Definition of Filter Branch PR behaviors. Requires the SCM Filter Branch PR Plugin. Refer to *~add_filter_branch_pr_behaviors*.
- **build-strategies** (*list*) – Provides control over whether to build a branch (or branch like things such as change requests and tags) whenever it is discovered initially or a change from the previous revision has been detected. (optional) Refer to *~build_strategies*.
- **property-strategies** (*dict*) – Provides control over how to build a branch (like to disable SCM triggering or to override the pipeline durability) (optional) Refer to *~property_strategies*.

Extensions

- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**

        * **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
      - **before (*dict*) - Clean the workspace before checkout**
        * **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
  - **prune** (*bool*) - Prune remote branches (default false)
  - **shallow-clone** (*bool*) - Perform shallow clone (default false)
  - **sparse-checkout (dict)**
      - **paths** (list) - List of paths to sparse checkout. (optional)
  - **depth** (*int*) - Set shallow clone depth (default 1)
  - **do-not-fetch-tags (*bool*) - Perform a clone without tags** (default false)
  - **submodule (*dict*)**
      - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.
      - **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
      - **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
      - **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
      - **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
      - **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).
  - **timeout** (*str*) - Timeout for git commands in minutes (optional)
  - **use-author (*bool*): Use author rather than committer in Jenkin's** build changeset (default false)
  - **wipe-workspace (*bool*) - Wipe out workspace before build** (default true)
  - **lfs-pull (*bool*) - Call git lfs pull after checkout** (default false)

Minimal Example:

```
name: 'demo-multibranch-git-min'
project-type: multibranch
scm:
    - git:
        url: 'https://example.com/jonhndoe/keep-frontend.git'
```

Full Example:

```yaml
name: 'demo-multibranch-git-min'
project-type: multibranch
script-path: some.Jenkinsfile
scm:
    - git:
        url: 'https://example.com/jonhndoe/keep-frontend.git'
        credentials-id: secret
        discover-branches: false
        ignore-on-push-notifications: true
        discover-tags: true
        head-filter-regex: 'master|\d+\.\d+'
        head-pr-filter-behaviors:
            head-pr-destined-regex:
                branch-regexp: "foo/.*"
                tag-regexp: "20\\..*"
            head-pr-destined-wildcard:
                branch-includes: "foo*"
                tag-includes: "qaz*"
                branch-excludes: "bar*"
                tag-excludes: "*baz"
            head-pr-originated-regex:
                branch-regexp: "(foo/.*|bar/.*)"
                tag-regexp: "1\\..*"
            head-pr-originated-wildcard:
                branch-includes: "qaz*"
                tag-includes: "bar*"
                branch-excludes: "baz*"
                tag-excludes: "*qaz"
        property-strategies:
            all-branches:
                - suppress-scm-triggering: true
                - pipeline-branch-durability-override: max-survivability
        build-strategies:
            - all-strategies-match:
                strategies:
                  - regular-branches: true
                  - skip-initial-build: true
            - any-strategies-match:
                strategies:
                  - change-request: {}
                  - tags: {}
            - tags:
                ignore-tags-newer-than: 1
                ignore-tags-older-than: 7
            - tags: {}
            - change-request:
                ignore-target-only-changes: true
            - change-request: {}
            - regular-branches: true
            - skip-initial-build: true
            - named-branches:
                - exact-name:
                    name: 'test'
```

(continues on next page)

(continued from previous page)

```
                    case-sensitive: true
                - regex-name:
                    regex: 'test.*$'
                    case-sensitive: true
                - wildcards-name:
                    excludes: 'testexclude'
                    includes: 'testinclude'
            - named-branches:
                - exact-name: {}
                - regex-name: {}
                - wildcards-name: {}
            - raw:
                xml: |
                  <com.igalg.jenkins.plugins.multibranch.buildstrategy.
↪IncludeRegionBranchBuildStrategy plugin="multibranch-build-strategy-extension">
                    <includedRegions>my/cool/project/*.cpp</includedRegions>
                  </com.igalg.jenkins.plugins.multibranch.buildstrategy.
↪IncludeRegionBranchBuildStrategy>
        clean:
            after: true
            before: true
        prune: true
        local-branch: true
        sparse-checkout:
            paths:
                - "path1"
                - "path2"
                - "path3"
        shallow-clone: true
        depth: 3
        do-not-fetch-tags: true
        submodule:
            disable: false
            recursive: true
            parent-credentials: true
            timeout: 100
            threads: 1
        timeout: "100"
        use-author: true
        wipe-workspace: true
        lfs-pull: true
```

project_multibranch.**github_scm**(*xml_parent*, *data*)

Configure GitHub SCM

Requires the GitHub Branch Source Plugin.

Parameters

- **api-uri** (*str*) – The GitHub API uri for hosted / on-site GitHub. Must first be configured in Global Configuration. (default GitHub)
- **ssh-checkout** (*bool*) – Checkout over SSH.
  - **credentials ('str'): Credentials to use for**

checkout of the repo over ssh.

- **credentials-id** (*str*) – Credentials used to scan branches and pull requests, check out sources and mark commit statuses. (optional)
- **repo-owner** (*str*) – Specify the name of the GitHub Organization or GitHub User Account. (required)
- **repo** (*str*) – The GitHub repo. (required)
- **branch-discovery** (*str*) – Discovers branches on the repository. Valid options: no-pr, only-pr, all, false. (default 'no-pr')
- **discover-pr-forks-strategy** (*str*) – Fork strategy. Valid options: merge-current, current, both, false. (default 'merge-current')
- **discover-pr-forks-trust** (*str*) – Discovers pull requests where the origin repository is a fork of the target repository. Valid options: contributors, everyone, permission or nobody. (default 'contributors')
- **discover-pr-origin** (*str*) – Discovers pull requests where the origin repository is the same as the target repository. Valid options: merge-current, current, both, false. (default 'merge-current')
- **discover-tags** (*bool*) – Discovers tags on the repository. (default false)
- **head-pr-filter-behaviors** (*list*) – Definition of Filter Branch PR behaviors. Requires the SCM Filter Branch PR Plugin. Refer to *~add_filter_branch_pr_behaviors*.
- **build-strategies** (*list*) – Provides control over whether to build a branch (or branch like things such as change requests and tags) whenever it is discovered initially or a change from the previous revision has been detected. (optional) Refer to *~build_strategies*.
- **notification-context** (*dict*) – Change the default GitHub check notification context from "continuous-integration/jenkins/SUFFIX" to a custom label / suffix. (set a label and suffix to true or false, optional) Requires the Github Custom Notification Context SCM Behaviour. Refer to *~add_notification_context_trait*.
- **property-strategies** (*dict*) – Provides control over how to build a branch (like to disable SCM triggering or to override the pipeline durability) (optional) Refer to *~property_strategies*.
- **filter-by-name-wildcard** (*dict*) – Enable filter by name with wildcards. Requires the SCM API Plugin.
  - **includes ('str'): Space-separated list**
    of name patterns to consider. You may use * as a wildcard; for example: *master release**
  - **excludes ('str'): Name patterns to**
    ignore even if matched by the includes list. For example: *release**

Extensions

- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**
    - **remove-stale-nested-repos (*bool*) - Deletes untracked**
      submodules and any other subdirectories which contain .git directories (default false)

      - **before** (*dict*) **- Clean the workspace before checkout**
        * **remove-stale-nested-repos** (*bool*) **- Deletes untracked** submodules and any other subdirectories which contain .git directories (default false)
    - **prune** (*bool*) - Prune remote branches (default false)
    - **shallow-clone** (*bool*) - Perform shallow clone (default false)
    - **sparse-checkout (dict)**
      - **paths** (list) - List of paths to sparse checkout. (optional)
    - **depth** (*int*) - Set shallow clone depth (default 1)
    - **do-not-fetch-tags** (*bool*) **- Perform a clone without tags** (default false)
    - **disable-pr-notifications** (*bool*) **- Disable default github status** notifications on pull requests (default false) (Requires the GitHub Branch Source Plugin.)
    - **refspecs** (*list(str)*): Which refspecs to fetch.
    - **submodule** (*dict*)
      - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.
      - **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
      - **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
      - **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
      - **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
      - **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).
    - **timeout** (*str*) - Timeout for git commands in minutes (optional)
    - **use-author** (*bool*)**: Use author rather than committer in Jenkin's** build changeset (default false)
    - **wipe-workspace** (*bool*) **- Wipe out workspace before build** (default true)
    - **lfs-pull** (*bool*) **- Call git lfs pull after checkout** (default false)

Minimal Example:

```
name: 'demo-multibranch-github-min'
project-type: multibranch
scm:
    - github:
        repo: 'foo'
        repo-owner: 'johndoe'
```

Full Example:

```
name: scm-github-full
project-type: multibranch
script-path: some.Jenkinsfile
scm:
    - github:
        api-uri: http://example.org/github
        ssh-checkout:
            credentials: 'ssh_secret'
        repo: example-repo
        repo-owner: example-owner
        credentials-id: example-credential
        branch-discovery: all
        head-filter-regex: "(.*/master|.*/release/.*)"
        head-pr-filter-behaviors:
            head-pr-destined-regex:
                branch-regexp: "foo/.*"
                tag-regexp: "20\\..*"
            head-pr-destined-wildcard:
                branch-includes: "foo*"
                tag-includes: "qaz*"
                branch-excludes: "bar*"
                tag-excludes: "*baz"
            head-pr-originated-regex:
                branch-regexp: "(foo/.*|bar/.*)"
                tag-regexp: "1\\..*"
            head-pr-originated-wildcard:
                branch-includes: "qaz*"
                tag-includes: "bar*"
                branch-excludes: "baz*"
                tag-excludes: "*qaz"
        discover-pr-forks-strategy: both
        discover-pr-forks-trust: everyone
        discover-pr-origin: both
        discover-tags: true
        notification-context:
            label: 'jenkins.example.com/my_context'
            suffix: false
        status-checks:
            name: my-checks
            skip: true
            skip-branch-source-notifications: true
            publish-unstable-as-neutral: true
            suppress-log-output: true
            suppress-progress-updates: true
            verbose-logs: true
        property-strategies:
            all-branches:
                - suppress-scm-triggering:
                    suppression-strategy: suppress-branch-indexing
                    branch-regex: ^.*test.*$
                - pipeline-branch-durability-override: max-survivability
```

(continues on next page)

(continued from previous page)

```
            - trigger-build-on-pr-comment:
                comment: "Ci build!"
                allow-untrusted-users: true
            - trigger-build-on-pr-review:
                allow-untrusted-users: true
            - trigger-build-on-pr-update:
                allow-untrusted-users: true
        build-strategies:
            - all-strategies-match:
                strategies:
                  - regular-branches: true
                  - skip-initial-build: true
            - any-strategies-match:
                strategies:
                  - change-request: {}
                  - tags: {}
            - tags:
                ignore-tags-newer-than: 1
                ignore-tags-older-than: 7
            - tags: {}
            - change-request:
                ignore-target-only-changes: true
            - change-request: {}
            - regular-branches: true
            - skip-initial-build: true
            - named-branches:
                - exact-name:
                    name: 'test'
                    case-sensitive: true
                - regex-name:
                    regex: 'test.*$'
                    case-sensitive: true
                - wildcards-name:
                    excludes: 'testexclude'
                    includes: 'testinclude'
            - named-branches:
                - exact-name: {}
                - regex-name: {}
                - wildcards-name: {}
        clean:
            after: true
            before: true
        committer:
            user: CI System
            email: no-reply@ci.example.com
        prune: true
        local-branch: true
        sparse-checkout:
            paths:
                - "path1"
                - "path2"
                - "path3"
```

(continues on next page)

(continued from previous page)

```
        shallow-clone: true
        depth: 3
        do-not-fetch-tags: true
        disable-pr-notifications: true
        refspecs:
          - '+refs/heads/*:refs/remotes/@{remote}/*'
        submodule:
            disable: false
            recursive: true
            parent-credentials: true
            timeout: 100
            threads: 1
        timeout: "100"
        skip-notifications: true
        use-author: true
        wipe-workspace: true
        lfs-pull: true
```

project_multibranch.**property_strategies**(*xml_parent*, *data*)

Configure Basic Branch Property Strategies.

Requires the Branch API Plugin.

Parameters

property-strategies (*dict*) – Definition of property strategies. Either *named-branches* or *all-branches* may be specified, but not both.

- **all-branches (list): A list of property strategy definitions** for use with all branches.
  - **suppress-scm-triggering (dict):** Suppresses automatic SCM triggering (optional).
    - **branch-regex (str):** Regex matching branch names. Only these branches will be affected by the selected suppression strategy. Default `^$`.
    - **suppress-strategy (str):** Select what to suppress for branches matching the regex. Valid values: `suppress-nothing` (default), `suppress-branch-indexing`, or `suppress-webhooks`.

      Note: Using `suppress-scm-triggering: true` is deprecated since Branch API 2.1045.v4ec3ed07b_e4f.
  - **pipeline-branch-durability-override (str): Set a custom** branch speed/durability level. Valid values: performance-optimized, survivable-nonatomic, or max-survivability (optional) Requires the Pipeline Multibranch Plugin
  - **trigger-build-on-pr-comment (str or dict): The comment body to** trigger a new build for a PR job when it is received. This is compiled as a case-insensitive regular expression, so use `".*"` to trigger a build on any comment whatsoever. (optional) If dictionary syntax is used, the option requires 2 fields: `comment` with the comment body and

`allow-untrusted-users` (bool) causing the plugin to skip checking if the comment author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin

- **trigger-build-on-pr-review (bool or dict): This property will** cause a job for a pull request (`PR-*`) to be triggered immediately when a review is made on the PR in GitHub. This has no effect on jobs that are not for pull requests. (optional) If dictionary syntax is used, the option requires `allow-untrusted-users` (bool) causing the plugin to skip checking if the review author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin
- **trigger-build-on-pr-update (bool or dict): This property will** cause a job for a pull request (`PR-*`) to be triggered immediately when the PR title or description is edited in GitHub. This has no effect on jobs that are not for pull requests. (optional) If dictionary syntax is used, the option requires `allow-untrusted-users` (bool) causing the plugin to skip checking if the update author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin

- **named-branches (dict): Named branches get different properties.** Comprised of a list of defaults and a list of property strategy exceptions for use with specific branches.
  - **defaults (list): A list of property strategy definitions** to be applied by default to all branches, unless overridden by an entry in *exceptions*
    - **suppress-scm-triggering (dict):** Suppresses automatic SCM triggering (optional).
      - **branch-regex (str):** Regex matching branch names. Only these branches will be affected by the selected suppression strategy. Default `^$`.
      - **suppress-strategy (str):** Select what to suppress for branches matching the regex. Valid values: `suppress-nothing` (default), `suppress-branch-indexing`, or `suppress-webhooks`.

        Note: Using `suppress-scm-triggering: true` is deprecated since Branch API 2.1045.v4ec3ed07b_e4f.

    - **pipeline-branch-durability-override (str): Set a custom** branch speed/durability level. Valid values: performance-optimized, survivable-nonatomic, or max-survivability (optional) Requires the Pipeline Multibranch Plugin
    - **trigger-build-on-pr-comment (str or dict): The comment body to** trigger a new build for a PR job when it is received. This is compiled as a case-insensitive regular expression, so use `".*"` to trigger a build on any comment whatsoever. (optional) If dictionary syntax is used, the option accepts 2 fields: `comment` (str, required) with the comment body and `allow-untrusted-users` (bool, optional) causing the plugin to skip checking if the comment author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin

    * **trigger-build-on-pr-review (bool or dict): This property will** cause a job for a pull request (`PR-*`) to be triggered immediately when a review is made on the PR in GitHub. This has no effect on jobs that are not for pull requests. (optional) If dictionary syntax is used, the option requires `allow-untrusted-users` (bool) causing the plugin to skip checking if the review author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin

    * **trigger-build-on-pr-update (bool or dict): This property will** cause a job for a pull request (`PR-*`) to be triggered immediately when the PR title or description is edited in GitHub. This has no effect on jobs that are not for pull requests. (optional) If dictionary syntax is used, the option requires `allow-untrusted-users` (bool) causing the plugin to skip checking if the update author is a collaborator of the GitHub project. Requires the GitHub PR Comment Build Plugin

- **exceptions (list): A list of branch names and the property** strategies to be used on that branch, instead of any listed in *defaults*.

    * **exception (dict): Defines exception**

        - **branch-name (str): Name of the branch to which these** properties will be applied.

        - **properties (list): A list of properties to apply to** this branch.

          suppress-scm-triggering (dict): Suppresses automatic SCM triggering (optional).

          branch-regex (str): Regex matching branch names. Only these branches will be affected by the selected suppression strategy. Default `^$`.

          suppress-strategy (str): Select what to suppress for branches matching the regex. Valid values: `suppress-nothing` (default), `suppress-branch-indexing`, or `suppress-webhooks`.

          Note: Using `suppress-scm-triggering: true` is deprecated since Branch API 2.1045.v4ec3ed07b_e4f.

          pipeline-branch-durability-override (str): Set a custom branch speed/durability level. Valid values: performance-optimized, survivable-nonatomic, or max-survivability (optional) Requires the Pipeline Multibranch Plugin

Delivery Pipeline View

The view delivery pipeline module handles creation of Delivery Pipeline views. To create a delivery pipeline view specify `delivery_pipeline` in the `view-type` attribute to the *Delivery Pipeline View* definition. Requires the Jenkins Delivery Pipeline Plugin.

View Parameters

- **name** (*str*): The name of the view.
- **view-type** (*str*): The type of view.
- **description** (*str*): A description of the view. (optional)
- **filter-executors** (*bool*): Show only executors that can execute the included views. (default false)
- **filter-queue** (*bool*): Show only included jobs in builder queue. (default false)
- **components (*list*):**
  - **name** (*str*): Name of the pipeline, usually the name of the component or product.
  - **initial-job** (*str*): First job in the pipeline.
  - **final-job** (*str*): Final job to display in the pipeline view regardless of its downstream jobs. (default '')
  - **show-upstream** (*bool*): Whether to show upstream. (default false)
- **regexps (*list*):**
  - **regexp** (*str*): Regular expression to find initial jobs.
  - **show-upstream** (*bool*): Whether to show upstream. (default false)
- **aggregated-changes-grouping-pattern** (*str*): Group changelog by regex pattern. (default '')
- **allow-abort** (*bool*): Allow cancelling a running job from the delivery pipeline view. (default false)
- **allow-manual-triggers** (*bool*): Displays a button in the pipeline view if a task is manual (Build other projects (manual step)) from Build Pipeline Plugin. (default false)
- **allow-pipeline-start** (*bool*): Allow starting a new pipeline run from the delivery pipeline view. (default false)
- **allow-rebuild** (*bool*): Allow rerunning a task from the delivery pipeline view. (default false)
- **link-relative** (*bool*): Use relative links for jobs in this pipeline view to allow for easier navigation. (default false)
- **link-to-console-log** (*bool*): Changes behaviour of task link in delivery pipeline view to go directly to the console log. (default false)
- **max-number-of-visible-pipelines** (*int*): Limits the number of pipelines shown in the view, regardless of how many pipelines are configured. A negative value will not enforce a limit.
- **no-of-columns** (*int*): Number of columns used for showing pipelines. Possible values are 1 (default), 2 and 3.
- **no-of-pipelines** (*int*): Number of pipelines instances shown for each pipeline. Possible values are numbers from 1 to 50 (default 3).

- **paging-enabled** (*bool*): Enable pagination in normal view, to allow navigation to older pipeline runs which are not displayed on the first page. Not available in full screen view. (default false)
- **show-absolute-date-time** (*bool*): Show dates and times as absolute values instead of as relative to the current time. (default false)
- **show-aggregated-changes** (*bool*): Show an aggregated changelog between different stages. (default false)
- **show-aggregated-pipeline** (*bool*): Show an aggregated view where each stage shows the latest version being executed. (default false)
- **show-avatars** (*bool*): Show avatars pictures instead of names of the people involved in a pipeline instance. (default false)
- **show-changes** (*bool*): Show SCM change log for the first job in the pipeline. (default false)
- **show-description** (*bool*): Show a build description connected to a specific pipeline task. (default false)
- **show-promotions** (*bool*): Show promotions from Promoted Builds Plugin. (default false)
- **show-static-analysis-results** (*bool*): Show different analysis results from Analysis Collector Plugin. (default false)
- **show-test-results** (*bool*): Show test results as pass/failed/skipped. (default false)
- **show-total-build-time** (*bool*): Show total build time for a pipeline run. (default false)
- **sorting** (*str*): How to sort the pipelines in the current view. Only applicable when multiple pipelines are configured in the same view. Possible values are 'none' (default), 'title' (sort by title), 'failed_last_activity' (sort by failed pipelines, then by last activity), 'last_activity' (sort by last activity).
- **update-interval** (*int*): How often the pipeline view will be updated. To be specified in seconds. (default 2)

Minimal Example:

```
name: Test pipeline
description: Test jobs created by JJB.
view-type: delivery_pipeline
components:
  - name: Test
    initial-job: Test-A
```

Full Example:

```
name: Test pipeline
description: Test jobs created by JJB.
view-type: delivery_pipeline
components:
  - name: Test
    initial-job: Test-A
regexps:
  - regexp: ^Test-A
no-of-pipelines: 1
allow-manual-triggers: yes
show-total-build-time: yes
```

(continues on next page)

(continued from previous page)

```
allow-rebuild: yes
allow-pipeline-start: yes
allow-abort: yes
paging-enabled: yes
link-to-console-log: yes
```

class view_delivery_pipeline.**DeliveryPipeline**(*registry*)

List View

The view list module handles creating Jenkins List views.

To create a list view specify `list` in the `view-type` attribute to the *List View* definition.

View Parameters

- **name** (*str*): The name of the view.
- **view-type** (*str*): The type of view.
- **description** (*str*): A description of the view. (default '')
- **filter-executors** (*bool*): Show only executors that can execute the included views. (default false)
- **filter-queue** (*bool*): Show only included jobs in builder queue. (default false)
- **job-name** (*list*): List of jobs to be included.
- **job-filters** (*dict*): Job filters to be included. Requires View Job Filters
  - **most-recent (*dict*)**

    most-recent

    * **max-to-include** (*int*): Maximum number of jobs to include. (default 0)
    * **check-start-time** (*bool*): Check job start time. (default false)
  - **build-duration (*dict*)**

    build-duration

    * **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
    * **build-duration-type** ('str'): Duration of the build. (default Latest)
    * **amount-type**: ('str'): Duration in hours, days or builds. (default Hours)
    * **amount**: ('int'): How far back to check. (default 0)
    * **less-than**: ('bool'): Check build duration less than or more than. (default True)
    * **build-duration-minutes**: ('int'): Build duration minutes. (default 0)
  - **build-trend (*dict*)**

build-trend

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **build-trend-type** ('str'): Duration of the build. (default Latest)
* **amount-type**: ('str'): Duration in hours, days or builds. (default Hours)
* **amount**: ('int'): How far back to check. (default 0)
* **status**: ('str'): Job status. (default Completed)

– **job-status** (*dict*)

job-status

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **unstable** ('bool'): Jobs with status unstable. (default False)
* **failed** ('bool'): Jobs with status failed. (default False)
* **aborted** ('bool'): Jobs with status aborted. (default False)
* **disabled** ('bool'): Jobs with status disabled. (default False)
* **stable** ('bool'): Jobs with status stable. (default False)

– **fallback** (*dict*)

fallback

* **fallback-type** ('str'): Fallback type to include/exclude for all jobs in a view, if no jobs have been included by previous filters. (default REMOVE_ALL_IF_ALL_INCLUDED)

– **build-status** (*dict*)

build-status

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **never-built** ('bool'): Jobs that are never built. (default False)
* **building** ('bool'): Jobs that are being built. (default False)
* **in-build-queue** ('bool'): Jobs that are in the build queue. (default False)

– **user-relevence** (*dict*)

user-relevence

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)

* **build-count** ('str'): Count of builds. (default AtLeastOne)
* **amount-type**: ('str'): Duration in hours, days or builds. (default Hours)
* **amount**: ('int'): How far back to check. (default 0)
* **match-user-id** ('bool'): Jobs matching user-id. (default False)
* **match-user-fullname** ('bool'): Jobs matching user fullname. (default False)
* **ignore-case** ('bool'): Ignore case. (default False)
* **ignore-whitespace** ('bool'): Ignore whitespace. (default False)
* **ignore-non-alphaNumeric** ('bool'): Ignore non-alphaNumeric. (default False)
* **match-builder** ('bool'): Jobs matching builder. (default False)
* **match-email** ('bool'): Jobs matching email. (default False)
* **match-scm-changes** ('bool'): Jobs matching scm changes. (default False)

– **regex-job (*dict*)**

regex-job

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **regex-name** ('str'): Regular expression name. (default '')
* **regex** ('str'): Regular expression. (default '')

– **job-tpye (*dict*)**

job-type

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **job-type** ('str'): Type of Job. (default hudson.model.FreeStyleProject)

– **parameter (*dict*)**

parameter

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **name** ('str'): Job name to match. (default '')
* **value** ('str'): Value to match. (default '')
* **desc** ('str'): Description to match. (default '')

* **use-default-value** ('bool'): Use default value. (default False)
* **match-builds-in-progress** ('bool'): Match build in progress. (default False)
* **match-all-builds** ('bool'): Match all builds. (default False)
* **max-builds-to-match** ('int'): Maximum builds to match. (default 0)

– **other-views (*dict*)**

other-views

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **view-name** ('str'): View name. (default select a view other than this one)

– **scm (*dict*)**

scm

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **scm-type** ('str'): Type of SCM. (default hudson.scm.NullSCM)

– **secured-job (*dict*)**

secured-job

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)

– **user-permissions (*dict*)**

user-permissions

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)
* **configure** ('bool'): User with configure permissions. (default false)
* **amount-type**: ('bool'): User with build permissions. (default false)
* **amount**: ('bool'): User with workspace permissions. (default false)
* **permission-check**: ('str'): Match user permissions. (default MustMatchAll)

– **upstream-downstream (*dict*)**

upstream-downstream

* **include-upstream** ('bool'): Jobs that match upstream. (default False)

* **include-downstream** ('bool'): Jobs that match downstream. (default False)
* **recursive** ('bool'): Jobs that are recursive. (default False)
* **exclude-originals** ('bool'): Jobs that are originals. (default False)

– **unclassified** (*dict*)

unclassified

* **match-type** ('str'): Jobs that match a filter to include. (default includeMatched)

- **columns** (*list*): List of columns to be shown in view.
- **regex** (*str*): . Regular expression for selecting jobs (optional)
- **recurse** (*bool*): Recurse in subfolders.(default false)
- **status-filter** (*bool*): Filter job list by enabled/disabled status. (optional)

Example:

```
name: list-view-name01
view-type: list
description: 'Sample description'
filter-executors: true
filter-queue: true
job-name:
    - job-name-1
    - job-name-3
    - job-name-2
    - Job-name-4
columns:
    - status
    - weather
    - job
    - last-success
    - last-failure
    - last-duration
    - build-button
    - last-stable
    - robot-list
    - find-bugs
    - jacoco
    - git-branch
    - favorite
    - schedule-build
    - priority-sorter
    - build-filter
    - desc
    - policy-violations
    - member-graph-view
    - built-on
    - extra-tests-total
```

(continues on next page)

(continued from previous page)

```
    - extra-tests-failed
    - extra-tests-passed
    - extra-tests-skipped
    - extra-tests-format-0
    - extra-tests-format-1
    - extra-build-parameters
    - extra-build-description
    - extra-last-user-name
    - extra-workspace-link
    - extra-configure-button
    - extra-last-output
recurse: true
status-filter: false
```

Example:

```
name: regex-example
view-type: list
description: 'description'
columns:
    - status
    - weather
    - job
    - last-success
    - last-failure
    - last-duration
    - extra-build-parameter: MY_PARAMETER
regex: (?!test.*).*
```

class view_list.**List**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Nested View

The view nested module handles creating Jenkins Nested views.

To create a nested view specify `nested` in the `view-type` attribute to the *Nested View* definition.

View Parameters

- **name** (*str*): The name of the view.
- **view-type** (*str*): The type of view.
- **description** (*str*): A description of the view. (default '')
- **filter-executors** (*bool*): Show only executors that can execute the included views. (default false)
- **filter-queue** (*bool*): Show only included jobs in builder queue. (default false)
- **views** (*list*): The views to nest.

- **default-view** (*str*): Name of the view to use as the default from the nested ones. (the first one by default)
- **columns** (*list*): List of columns to be shown in view. (default empty list)

Example:

```
name: NestedViewTest
view-type: nested
views:
  - name: All
    view-type: all
columns:
  - status
  - weather
```

class view_nested.**Nested**(*registry*)

Pipeline View

The view pipeline module handles creating Jenkins Build Pipeline views. To create a pipeline view specify `pipeline` in the `view-type` attribute to the *Pipeline View* definition.

Requires the Jenkins Build Pipeline Plugin.

View Parameters

- **name** (*str*): The name of the view.
- **view-type** (*str*): The type of view.
- **description** (*str*): A description of the view. (optional)
- **filter-executors** (*bool*): Show only executors that can execute the included views. (default false)
- **filter-queue** (*bool*): Show only included jobs in builder queue. (default false)
- **first-job** (*str*): Parent Job in the view.
- **no-of-displayed-builds** (*str*): Number of builds to display. (default 1)
- **title** (*str*): Build view title. (optional)
- **linkStyle** (*str*): Console output link style. Can be 'Lightbox', 'New Window', or 'This Window'. (default Lightbox)
- **css-Url** (*str*): Url for Custom CSS files (optional)
- **latest-job-only** (*bool*) Trigger only latest job. (default false)
- **manual-trigger** (*bool*) Always allow manual trigger. (default false)
- **show-parameters** (*bool*) Show pipeline parameters. (default false)
- **parameters-in-headers** (*bool*) Show pipeline parameters in headers. (default false)
- **starts-with-parameters** (*bool*) Use Starts with parameters. (default false)
- **refresh-frequency** (*str*) Frequency to refresh in seconds. (default '3')
- **definition-header** (*bool*) Show pipeline definition header. (default false)

Example:

```
name: testBPview
view-type: pipeline
description: 'This is a description'
filter-executors: false
filter-queue: false
first-job: job-one
no-of-displayed-builds: 5
title: Title
link-style: New Window
css-Url: fake.urlfor.css
latest-job-only: true
manual-trigger: true
show-parameters: true
parameters-in-headers: true
start-with-parameters: true
refresh-frequency: 3
definition-header: true
```

Example:

```
name: testBPview
view-type: pipeline
first-job: job-one
```

class view_pipeline.**Pipeline**(*registry*)

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Builders

Builders define actions that the Jenkins job should execute. Examples include shell scripts or maven targets. The `builders` attribute in the *Job* definition accepts a list of builders to invoke. They may be components defined below, locally defined macros (using the top level definition of `builder:`, or locally defined components found via the `jenkins_jobs.builders` entry point.

Component: builders

Macro
builder

Entry Point
jenkins_jobs.builders

Example:

```
job:
  name: test_job

  builders:
    - shell: "make test"
```

class builders.**Builders**(*registry*)

component_list_type = 'builders'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'builder'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 60

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

ansible-playbook()

This plugin allows you to execute Ansible tasks as a job build step.

Requires the Jenkins Ansible Plugin.

Parameters

- **playbook** (*str*) – Path to the ansible playbook file. The path can be absolute or relative to the job workspace. (required)
- **inventory-type** (*str*) – The inventory file form (default *path*)
  - **inventory-type values**
    - **path**
    - **content**
    - **do-not-specify**
- **inventory** (*dict*) – Inventory data, depends on inventory-type
  - **inventory-type == path**
    - **path** (*str*) – Path to inventory file.
  - **inventory-type == content**
    - **content** (*str*) – Content of inventory file.
    - **dynamic** (*bool*) – Dynamic inventory is used (default false)
- **hosts** (*str*) – Further limit selected hosts to an additional pattern (default '')
- **tags-to-run** (*str*) – Only run plays and tasks tagged with these values (default '')
- **tags-to-skip** (*str*) – Only run plays and tasks whose tags do not match these values (default '')
- **task-to-start-at** (*str*) – Start the playbook at the task matching this name (default '')
- **workers** (*int*) – Specify number of parallel processes to use (default 5)
- **credentials-id** (*str*) – The ID of credentials for the SSH connections. Only private key authentication is supported (default '')
- **vault-credentials-id** (*str*) – The ID of credentials for the vault decryption (default '')
- **sudo** (*bool*) – Run operations with sudo. It works only when the remote user is sudoer with nopasswd option (default false)
- **sudo-user** (*str*) – Desired sudo user. "root" is used when this field is empty. (default '')

- **unbuffered-output** (*bool*) – Skip standard output buffering for the ansible process. The ansible output is directly rendered into the Jenkins console. This option can be useful for long running operations. (default true)
- **colorized-output** (*bool*) – Check this box to allow ansible to render ANSI color codes in the Jenkins console. (default false)
- **disable-host-key-checking** (*bool*) – Check this box to disable the validation of the hosts SSH server keys. (>= 1.0) (default false)
- **additional-parameters** (*str*) – Any additional parameters to pass to the ansible command. (default '')
- **variables** (*list*) – List of extra variables to be passed to ansible. (optional)

  variable item

  - **name** (*str*) – Name of variable (required)
  - **value** (*str*) – Desired value (default '')
  - **hidden** (*bool*) – Hide variable in build log (default false)

Outdated Options for versions >= 1.0 of plugin:

Parameters

host-key-checking (*bool*) – Outdated, replaced with disable-host-key-checking. Check this box to enforce the validation of the hosts SSH server keys. (< 1.0) (default true)

Example:

```yaml
---
builders:
  - ansible-playbook:
      playbook: "path/to/playbook.yml"
      inventory-type: "path"
      inventory:
        path: "path/to/inventory-file"
      variables:
        - name: "my_variable"
          value: "my_value"
```

OR

```yaml
---
builders:
  - ansible-playbook:
      playbook: "path/to/playbook.yml"
      inventory-type: "content"
      inventory:
        content: |
          [all]
          machine01.example.com
          machine02.example.com
      hosts: "masters"
      tags-to-run: "ose"
      tags-to-skip: "ovirt"
      task-to-start-at: "Deploy application"
      workers: 2
      credentials-id: "0891c950-487b-4749-aa69-d87425e14459"
      vault-credentials-id: "0421b950-487b-4749-aa69-d87425e14459"
      sudo: true
      sudo-user: "cloud-user"
      unbuffered-output: false
      colorized-output: true
```

(continues on next page)

(continued from previous page)

```
      additional-parameters: "-vvv"
      variables:
        - name: "complete_var"
          value: "complete value"
          hidden: false
        - name: "empty_var"
        - name: "hidden_var"
          value: "Do not appear in console"
          hidden: true
```

Example(s) versions < 1.0:

```
---
builders:
  - ansible-playbook:
      playbook: "path/to/playbook.yml"
      inventory-type: "do-not-specify"
      become: "yes"
      become-user: "cloud-user"
      host-key-checking: false
```

ant()

Execute an ant target. Requires the Jenkins Ant Plugin.

To setup this builder you can either reference the list of targets or use named parameters. Below is a description of both forms:

1) Listing targets:

After the ant directive, simply pass as argument a space separated list of targets to build.

Parameter

space separated list of Ant targets

Example to call two Ant targets:

```
builders:
  - ant: "target1 target2"
```

The build file would be whatever the Jenkins Ant Plugin is set to use per default (i.e build.xml in the workspace root).

2) Using named parameters:

Parameters

- **targets** (*str*) – the space separated list of ANT targets.
- **buildfile** (*str*) – the path to the ANT build file.
- **properties** (*list*) – Passed to ant script using -Dkey=value (optional)
- **ant-name** (*str*) – the name of the ant installation, (default 'default') (optional)
- **java-opts** (*str*) – java options for ant, can have multiples, must be in quotes (optional)

Example specifying the build file too and several targets:

```
builders:
  - ant:
      targets: "debug test install"
      buildfile: "build.xml"
      properties:
```

(continues on next page)

(continued from previous page)

```
        builddir: "/tmp/"
        failonerror: true
    java-opts:
        - "-ea"
        - "-Xmx512m"
    ant-name: "Standard Ant"
```

artifact-resolver()

Allows one to resolve artifacts from a maven repository like nexus (without having maven installed)

Requires the Jenkins Repository Connector Plugin.

Parameters

- **fail-on-error** (*bool*) – Whether to fail the build on error (default false)
- **repository-logging** (*bool*) – Enable repository logging (default false)
- **target-directory** (*str*) – Where to resolve artifacts to (required)
- **artifacts** (*list*) – list of artifacts to resolve

  Artifact

  - **group-id** (*str*) – Group ID of the artifact (required)
  - **artifact-id** (*str*) – Artifact ID of the artifact (required)
  - **version** (*str*) – Version of the artifact (required)
  - **classifier** (*str*) – Classifier of the artifact (default '')
  - **extension** (*str*) – Extension of the artifact (default 'jar')
  - **target-file-name** (*str*) – What to name the artifact (default '')

Minimal Example:

```
builders:
  - artifact-resolver:
      target-directory: foo
      artifacts:
        - group-id: commons-logging
          artifact-id: commons-logging
          version: "1.1"
```

Full Example:

```
builders:
  - artifact-resolver:
      fail-on-error: true
      repository-logging: true
      target-directory: foo
      artifacts:
        - group-id: commons-logging
          artifact-id: commons-logging
          version: "1.1"
          classifier: src
          extension: jar
          target-file-name: comm-log.jar
        - group-id: commons-lang
          artifact-id: commons-lang
          version: "1.2"
```

batch()

Execute a batch command.

Parameter
the batch command to execute

Example:

```
builders:
  - batch: "foo/foo.bat"
```

beaker()

Execute a beaker build step.

Requires the Jenkins Beaker Builder Plugin.

Parameters
- **content** (*str*) – Run job from string (Alternative: you can choose a path instead)
- **path** (*str*) – Run job from file (Alternative: you can choose a content instead)
- **download-logs** (*bool*) – Download Beaker log files (default false)

Example:

```
builders:
    - beaker:
        path: 'test.xml'
        download-logs: true
```

```
builders:
    - beaker:
        content: |
            <job group='product-QA'>
                <whiteboard>
                    Apache 2.2 test
                </whiteboard>
            </job>
```

build-name-setter()

Define Build Name Setter options which allows your build name to be updated during the build process.

Requires the Jenkins Build Name Setter Plugin.

Parameters
- **name** (*str*) – Filename to use for Build Name Setter, only used if file bool is true. (default 'version.txt')
- **template** (*str*) – Macro Template string, only used if macro bool is true. (default '#${BUILD_NUMBER}')
- **file** (*bool*) – Read from named file (default false)
- **macro** (*bool*) – Read from macro template (default false)
- **macro-first** (*bool*) – Insert macro first (default false)

File Example:

```
builders:
      - build-name-setter:
          name: 'version.txt'
          file: true
```

Macro Example:

```
builders:
      - build-name-setter:
```

(continues on next page)

(continued from previous page)

```
        template: '#${BUILD_NUMBER}'
        macro: true
```

build-publish-docker-image()

Provides the ability to build projects with a Dockerfile and publish the resultant tagged image (repo) to the docker registry.

Requires the Jenkins CloudBees Docker Build and Publish plugin.

Parameters

- **docker-registry-url** (*str*) – URL to the Docker registry you are using (default '')
- **image** (*str*) – Repository name to be applied to the resulting image in case of success (default '')
- **docker-file-directory** (*str*) – Build step that sends a Dockerfile for building to docker host that used for this build run (default '')
- **push-on-success** (*bool*) – Resulting docker image will be pushed to the registry (or registries) specified within the "Image" field (default false)
- **push-credentials-id** (*str*) – Credentials to push to a private registry (default '')
- **clean-images** (*bool*) – Option to clean local images (default false)
- **jenkins-job-delete** (*bool*) – Attempt to remove images when jenkins deletes the run (default false)
- **cloud** (*str*) – Cloud to use to build image (default '')

Minimal example:

```
builders:
  - build-publish-docker-image
```

Full example:

```
builders:
  - build-publish-docker-image:
      docker-registry-url: Docker registry URL
      image: Image string
      docker-file-directory: Directory for Dockerfile
      push-on-success: true
      push-credentials-id: 71e4f29c-162b-40d0-85d9-3ddfba2911a0
      clean-images: true
      jenkins-job-delete: true
      cloud: cloud
```

builders-from()

Use builders from another project.

Requires the Jenkins Template Project Plugin.

Parameters

projectName (*str*) – the name of the other project

Example:

```
builders:
  - builders-from: "base-build"
```

change-assembly-version()

Change the assembly version. Requires the Jenkins Change Assembly Version.

Parameters

- **version** (*str*) – Set the new version number for replace (default 1.0.0)

- **assemblyFile** (*str*) – The file name to search (default AssemblyInfo.cs)

Minimal Example:

```
builders:
  - change-assembly-version
```

Full Example:

```
builders:
  - change-assembly-version:
        version: "1.2.3"
        assembly-file: "AFile"
```

cloudformation()

Create cloudformation stacks before running a build and optionally delete them at the end.

Requires the Jenkins AWS Cloudformation Plugin.

Parameters

- **name** (*list*) – The names of the stacks to create (required)
- **description** (*str*) – Description of the stack (optional)
- **recipe** (*str*) – The cloudformation recipe file (required)
- **parameters** (*list*) – List of key/value pairs to pass into the recipe, will be joined together into a comma separated string (optional)
- **timeout** (*int*) – Number of seconds to wait before giving up creating a stack (default 0)
- **access-key** (*str*) – The Amazon API Access Key (required)
- **secret-key** (*str*) – The Amazon API Secret Key (required)
- **sleep** (*int*) – Number of seconds to wait before continuing to the next step (default 0)
- **region** (*str*) – The region to run cloudformation in (required)

  region values

  - **us-east-1**
  - **us-west-1**
  - **us-west-2**
  - **eu-central-1**
  - **eu-west-1**
  - **ap-southeast-1**
  - **ap-southeast-2**
  - **ap-northeast-1**
  - **sa-east-1**

Example:

```
builders:
  - cloudformation:
    - name: "foo"
      description: "Build the foo stack"
      recipe: "foo.json"
      parameters:
        - "Key1=foo"
        - "Key2=fuu"
      timeout: 3600
      access-key: "$AWS_ACCESS_KEY"
      secret-key: "$AWS_SECRET_KEY"
      region: us-west-2
      sleep: 5
```

(continues on next page)

(continued from previous page)

```
    - name: "bar"
      description: "Build the bar stack"
      recipe: "bar.json"
      parameters:
        - "Key1=bar"
        - "Key2=baa"
      timeout: 3600
      access-key: "$AWS_ACCESS_KEY"
      secret-key: "$AWS_SECRET_KEY"
      region: us-west-1
```

cmake()

Execute a CMake target.

Requires the Jenkins CMake Plugin.

This builder is compatible with both versions 2.x and 1.x of the plugin. When specifying paramenters from both versions only the ones from the installed version in Jenkins will be used, and the rest will be ignored.

Parameters

- **source-dir** (*str*) – the source code directory relative to the workspace directory. (required)
- **build-type** (*str*) – Sets the "build type" option for CMake (default "Debug").
- **preload-script** (*str*) – Path to a CMake preload script file. (optional)
- **other-arguments** (*str*) – Other arguments to be added to the CMake call. (optional)
- **clean-build-dir** (*bool*) – If true, delete the build directory before each build (default false).
- **generator** (*list*) – The makefile generator (default "Unix Makefiles").

  type Possible generators

  - **Borland Makefiles**
  - **CodeBlocks - MinGW Makefiles**
  - **CodeBlocks - Unix Makefiles**
  - **Eclipse CDT4 - MinGW Makefiles**
  - **Eclipse CDT4 - NMake Makefiles**
  - **Eclipse CDT4 - Unix Makefiles**
  - **MSYS Makefiles**
  - **MinGW Makefiles**
  - **NMake Makefiles**
  - **Unix Makefiles**
  - **Visual Studio 6**
  - **Visual Studio 7 .NET 2003**
  - **Visual Studio 8 2005**
  - **Visual Studio 8 2005 Win64**
  - **Visual Studio 9 2008**
  - **Visual Studio 9 2008 Win64**
  - **Watcom WMake**

Version 2.x

Parameters that available only to versions 2.x of the plugin

- **working-dir** (*str*): The directory where the project will be built in. Relative to the workspace directory. (optional)
- **installation-name** (*str*): The CMake installation to be used on this builder. Use one defined in your Jenkins global configuration page (default "InSearchPath").
- **build-tool-invocations** (*list*): list of build tool invocations that will happen during the build:

Build tool invocations

- **use-cmake (*str*) – Whether to run the actual build tool** directly (by expanding `$CMAKE_BUILD_TOOL`) or to have cmake run the build tool (by invoking `cmake --build <dir>`) (default false).
- **arguments (*str*) – Specify arguments to pass to the** build tool or cmake (separated by spaces). Arguments may contain spaces if they are enclosed in double quotes. (optional)
- **environment-variables (*str*) – Specify extra** environment variables to pass to the build tool as key-value pairs here. Each entry must be on its own line, for example:

```
DESTDIR=${WORKSPACE}/
artifacts/dir

KEY=VALUE
```

Version 1.x

Parameters available only to versions 1.x of the plugin

- **build-dir** (*str*): The directory where the project will be built in. Relative to the workspace directory. (optional)
- **install-dir** (*str*): The directory where the project will be installed in, relative to the workspace directory. (optional)
- **build-type** (*list*): Sets the "build type" option. A custom type different than the default ones specified on the CMake plugin can also be set, which will be automatically used in the "Other Build Type" option of the plugin. (default "Debug")

  Default types present in the CMake plugin

  - **Debug**
  - **Release**
  - **RelWithDebInfo**
  - **MinSizeRel**
- **make-command** (*str*): The make command (default "make").
- **install-command** (*arg*): The install command (default "make install").
- **custom-cmake-path** (*str*): Path to cmake executable. (optional)
- **clean-install-dir** (*bool*): If true, delete the install dir before each build (default false).

Example (Versions 2.x):

```
builders:
  - cmake:
      source-dir: 'path/to/source'
      working-dir: 'path/to/build'
      install-dir: 'path/to/install'
      build-type: 'CustomReleaseType'
      generator: 'NMake Makefiles'
      installation-name: 'CMake custom install'
      preload-script: 'path/to/source/cmake.preload'
      other-arguments: '-DCMAKE_FIND_ROOT_PATH="path/to/something/else"'
      clean-build-dir: true
      build-tool-invocations:
        - use-cmake: true
          arguments: 'clean'
```

(continues on next page)

(continued from previous page)

```
        environment-variables: |
          DESTDIR=${WORKSPACE}/artifacts/dir
          URL=http://www.example.org/
      - use-cmake: false
        arguments: 'test'
        environment-variables: |
          RESTRICT="TRUE"
          TARGET="NONE"
```

Example (Versions 1.x):

```
builders:
  - cmake:
      source-dir: 'path/to/source'
      build-dir: 'path/to/build'
      install-dir: 'path/to/install'
      build-type: 'CustomReleaseType'
      generator: 'NMake Makefiles'
      make-command: '/usr/bin/make'
      install-command: 'make new-install'
      preload-script: 'path/to/source/cmake.preload'
      other-arguments: '-DCMAKE_FIND_ROOT_PATH="path/to/something/else"'
      custom-cmake-path: '/usr/bin/cmake'
      clean-build-dir: true
      clean-install-dir: true
```

conditional-step()

Conditionally execute some build steps.

Requires the Jenkins Conditional BuildStep Plugin.

Depending on the number of declared steps, a *Conditional step (single)* or a *Conditional steps (multiple)* is created in Jenkins.

Parameters

- **condition-kind** (*str*) – Condition kind that must be verified before the steps are executed. Valid values and their additional attributes are described in the *conditions* table.
- **on-evaluation-failure** (*str*) – What should be the outcome of the build if the evaluation of the condition fails. Possible values are *fail*, *mark-unstable*, *run-and-mark-unstable*, *run* and *dont-run*. (default 'fail').
- **steps** (*list*) – List of steps to run if the condition is verified. Items in the list can be any builder known by Jenkins Job Builder.

Condition kind	Description
always	Condition is always verified
never	Condition is never verified
boolean-expression	Run the step if the expression expends to a representation of true **condition-expression** Expression to expand (required)
build-cause	Run if the current build has a specific cause **cause** The cause why the build was triggered. Following causes are supported - **USER_CAUSE** build was triggered by a manual interaction. (default) **SCM_CAUSE** build was triggered by a SCM change. **TIMER_CAUSE** build was triggered by a timer. **CLI_CAUSE** build was triggered by via CLI interface **REMOTE_CAUSE**

Examples:

```
builders:
    - conditional-step:
        condition-kind: always
        steps:
            - shell: 'first command'
            - shell: 'second command'
```

```
builders:
      - conditional-step:
          condition-kind: current-status
          condition-worst: SUCCESS
          condition-best: FAILURE
          steps:
              - shell: "sl"
```

```
builders:
      - conditional-step:
          condition-kind: not
          condition-operand:
              condition-kind: file-exists
              condition-filename: mytestfile
              condition-basedir: workspace
          steps:
              - shell: "touch $WORKSPACE/mytestfile"
```

```
builders:
      - conditional-step:
          condition-kind: day-of-week
          day-selector: weekday
          use-build-time: true
          steps:
              - shell: "sl"
```

```
builders:
      - conditional-step:
          condition-kind: day-of-week
          day-selector: select-days
          days:
              MON: true
              FRI: true
          use-build-time: true
          steps:
              - shell: "sl"
```

```
builders:
    - conditional-step:
        condition-kind: time
        earliest-hour: "4"
        earliest-min: "15"
        latest-hour: "20"
```

(continues on next page)

(continued from previous page)

```
        latest-min: "30"
        steps:
            - shell: "sl"
```

```
builders:
    - conditional-step:
        condition-kind: regex-match
        regex: a*b
        label: cadaaab
        steps:
            - shell: "sl"
```

```
builders:
      - conditional-step:
          condition-kind: or
          condition-operands:
              - condition-kind: num-comp
                lhs: "2 + 5"
                rhs: "1 + 6"
                comparator: equal
                condition-basedir: "jenkins-home"
              - condition-kind: files-match
                include-pattern:
                    - "inc_pattern1"
                    - "inc_pattern2"
                exclude-pattern:
                    - "exc_pattern1"
                    - "exc_pattern2"
                condition-basedir: "jenkins-home"
          steps:
              - shell: "sl"
```

```
builders:
      - conditional-step:
          condition-kind: and
          condition-operands:
              - condition-kind: regex-match
                regex: "*abc*"
                label: "dabcddabc"
              - condition-kind: time
                earliest-hour: "2"
                earliest-min: "0"
                latest-hour: "23"
                latest-min: "40"
                use-build-time: true
          steps:
              - shell: "sl"
```

config-file-provider()

Provide configuration files (i.e., settings.xml for maven etc.) which will be copied to the job's workspace.

Requires the Jenkins Config File Provider Plugin.

Parameters

files (*list*) – List of managed config files made up of three parameters

files

- **file-id** (*str*) – The identifier for the managed config file
- **target** (*str*) – Define where the file should be created (default '')
- **variable** (*str*) – Define an environment variable to be used (default '')
- **replace-tokens** (*bool*) – Replace tokens in config file. For example "password: ${PYPI_JENKINS_PASS}" will be replaced with the global variable configured in Jenkins.

Full Example:

```
builders:
    - config-file-provider:
        files:
            - file-id: org.jenkinsci.plugins.configfiles.maven.
↪MavenSettingsConfig0123456789012
              target: target
              variable: variable
              replace-tokens: true
```

Minimal Example:

```
builders:
    - config-file-provider:
        files:
            - file-id: org.jenkinsci.plugins.configfiles.maven.
↪MavenSettingsConfig0123456789012
```

copyartifact()

Copy artifact from another project. Requires the Copy Artifact plugin.

Please note using the multijob-build for which-build argument requires the Multijob plugin

Parameters

- **project** (*str*) – Project to copy from
- **filter** (*str*) – what files to copy
- **target** (*str*) – Target base directory for copy, blank means use workspace
- **flatten** (*bool*) – Flatten directories (default false)
- **optional** (*bool*) – If the artifact is missing (for any reason) and optional is true, the build won't fail because of this builder (default false)
- **do-not-fingerprint** (*bool*) – Disable automatic fingerprinting of copied artifacts (default false)
- **which-build** (*str*) – which build to get artifacts from (optional, default last-successful)

which-build values

- **last-successful**
- **last-completed**
- **specific-build**
- **last-saved**
- **upstream-build**
- **permalink**
- **workspace-latest**
- **build-param**
- **downstream-build**
- **multijob-build**

- **build-number** (*str*) – specifies the build number to get when when specific-build is specified as which-build
- **permalink** (*str*) – specifies the permalink to get when permalink is specified as which-build

  permalink values
  - **last**
  - **last-stable**
  - **last-successful**
  - **last-failed**
  - **last-unstable**
  - **last-unsuccessful**
- **stable** (*bool*) – specifies to get only last stable build when last-successful is specified as which-build
- **fallback-to-last-successful** (*bool*) – specifies to fallback to last successful build when upstream-build is specified as which-build
- **param** (*str*) – specifies to use a build parameter to get the build when build-param is specified as which-build
- **upstream-project-name** (*str*) – specifies the project name of downstream when downstream-build is specified as which-build
- **upstream-build-number** (*str*) – specifies the number of the build to find its downstream build when downstream-build is specified as which-build
- **parameter-filters** (*str*) – Filter matching jobs based on these parameters (optional)
- **exclude** (*str*) – Specify paths or patterns of artifacts to exclude, even if specified in "Artifacts to copy". (default '')
- **result-var-suffix** (*str*) – The build number of the selected build will be recorded into the variable named COPYARTIFACT_BUILD_NUMBER_(SUFFIX) for later build steps to reference. (default '')

Example:

```
builders:
  - copyartifact:
      project: foo
      filter: "*.tar.gz"
      target: /home/foo
      which-build: specific-build
      build-number: "123"
      optional: true
      flatten: true
      do-not-fingerprint: true
      parameter-filters: PUBLISH=true
```

Multijob Example:

```
builders:
  - copyartifact:
      project: foo
      filter: "*.json"
      target: /home/foo
      which-build: multijob-build
      optional: true
      flatten: true
      parameter-filters: PUBLISH=true
      exclude: "*.txt"
```

(continues on next page)

(continued from previous page)

```
      result-var-suffix: "PROJECT_ABC"
```

critical-block-end()

Designate the end of a critical block. Must be used in conjunction with critical-block-start.

Must also add a build wrapper (exclusion), specifying the resources that control the critical block. Otherwise, this will have no effect.

Requires the Jenkins Exclusion Plugin.

Example:

```
- wrapper:
    name: critical-block-exclusion
    wrappers:
      - exclusion:
          resources:
            - myresource1

- job:
    name: critical-block-example
    project-type: freestyle
    wrappers:
      - critical-block-exclusion
    builders:
      - critical-block-start
      - shell:
          #!/bin/bash -ex
          rollback-my-data-base
      - critical-block-end
```

critical-block-start()

Designate the start of a critical block. Must be used in conjunction with critical-block-end.

Must also add a build wrapper (exclusion), specifying the resources that control the critical block. Otherwise, this will have no effect.

Requires the Jenkins Exclusion Plugin.

Example:

```
- wrapper:
    name: critical-block-exclusion
    wrappers:
      - exclusion:
          resources:
            - myresource1

- job:
    name: critical-block-example
    project-type: freestyle
    wrappers:
      - critical-block-exclusion
    builders:
      - critical-block-start
```

(continues on next page)

(continued from previous page)

```
      - shell:
          #!/bin/bash -ex
          rollback-my-data-base
      - critical-block-end
```

description-setter()

This plugin sets the description for each build, based upon a RegEx test of the build log file.

Requires the Jenkins Description Setter Plugin.

Parameters

- **regexp** (*str*) – A RegEx which is used to scan the build log file (default '')
- **description** (*str*) – The description to set on the build (optional)

Example:

```
builders:
  - description-setter:
      regexp: ".*(<a href=.*a>)"
      description: "some description"
```

docker-build-publish()

Provides the ability to build projects with a Dockerfile, and publish the resultant tagged image (repo) to the docker registry.

Requires the Jenkins Docker build publish Plugin.

Parameters

- **repo-name** (*str*) – Name of repository to push to.
- **repo-tag** (*str*) – Tag for image. (default '')
- **server** (*dict*) – The docker daemon (optional)
  - **uri** (str): Define the docker server to use. (optional)
  - **credentials-id** (str): ID of credentials to use to connect (optional)
- **registry** (*dict*) – Registry to push to
  - **url** (str) repository url to use (optional)
  - **credentials-id** (str): ID of credentials to use to connect (optional)
- **no-cache** (*bool*) – If build should be cached. (default false)
- **no-force-pull** (*bool*) – Don't update the source image before building when it exists locally. (default false)
- **skip-build** (*bool*) – Do not build the image. (default false)
- **skip-decorate** (*bool*) – Do not decorate the build name. (default false)
- **skip-tag-latest** (*bool*) – Do not tag this build as latest. (default false)
- **skip-push** (*bool*) – Do not push. (default false)
- **file-path** (*str*) – Path of the Dockerfile. (default '')
- **build-context** (*str*) – Project root path for the build, defaults to the workspace if not specified. (default '')
- **create-fingerprint** (*bool*) – If enabled, the plugin will create fingerprints after the build of each image. (default false)
- **build-args** (*str*) – Additional build arguments passed to docker build (default '')
- **force-tag** (*bool*) – Force tag replacement when tag already exists (default false)

Minimal example:

```
builders:
    - docker-build-publish:
        repo-name: 'test'
        repo-tag: 'test-tag'
```

(continues on next page)

(continued from previous page)

```
        no-cache: true
        no-force-pull: false
        skip-build: false
        skip-decorate: false
        skip-latest: false
        skip-tag: false
        file-path: '/tmp/'
        build-context: '/tmp/'
        create-fingerprint: true
        build-args: --build-arg https_proxy="http://some.proxy:port"
        force-tag: true
```

Full example:

```
builders:
    - docker-build-publish:
        repo-name: 'test'
        repo-tag: 'test-tag'
        no-cache: true
        no-force-pull: false
        skip-build: false
        skip-decorate: false
        skip-latest: false
        skip-tag: false
        file-path: '/tmp/'
        build-context: '/tmp/'
        create-fingerprint: true
        build-args: --build-arg https_proxy="http://some.proxy:port"
        force-tag: true
        registry:
            url: 'https://registry.example.org'
            credentials-id: 'registry-docker'
        server:
            uri: 'unix:///var/run/docker.sock'
            credentials-id: 'docker-server'
```

docker-pull-image()

Provides integration between Jenkins and Docker Hub, utilizing a Docker Hub hook to trigger one (or more) Jenkins job(s).

Requires the Jenkins CloudBees Docker Hub Notification.

Parameters

- **image** (*str*) – Image ID on DockerHub (default '')
- **docker-registry-url** (*str*) – URL to the Docker registry you are using (default '')
- **credentials-id** (*str*) – Registry credentials (default '')

Minimal example:

```
builders:
  - docker-pull-image
```

Full example:

```
builders:
  - docker-pull-image:
      image: test-image-id
      docker-registry-url: https://index.docker.io/v1/
      credentials-id: 71e4f29c-162b-40d0-85d9-3ddfba2911a0
```

doxygen()

Builds doxygen HTML documentation.

Requires the Jenkins Doxygen plugin.

Parameters

- **doxyfile** (*str*) – The doxyfile path (required)
- **install** (*str*) – The doxygen installation to use (required)
- **ignore-failure** (*bool*) – Keep executing build even on doxygen generation failure (default false)
- **unstable-warning** (*bool*) – Mark the build as unstable if warnings are generated (default false)

Example:

```
builders:
  - doxygen:
      doxyfile: Doxyfile
      install: doxygen
      ignore-failure: true
      unstable-warning: true
```

dsl()

Process Job DSL

Requires the Jenkins Job DSL plugin.

Parameters

- **script-text** (*str*) – dsl script which is Groovy code (Required if targets is not specified)
- **targets** (*str*) – Newline separated list of DSL scripts, located in the Workspace. Can use wildcards like 'jobs/*/*/*.groovy' (Required if script-text is not specified)
- **ignore-existing** (*str*) – Ignore previously generated jobs and views
- **removed-job-action** (*str*) – Specifies what to do when a previously generated job is not referenced anymore, can be 'IGNORE', 'DISABLE', or 'DELETE' (default 'IGNORE')
- **removed-view-action** (*str*) – Specifies what to do when a previously generated view is not referenced anymore, can be 'IGNORE' or 'DELETE'. (default 'IGNORE')
- **lookup-strategy** (*str*) – Determines how relative job names in DSL scripts are interpreted, can be 'JENKINS_ROOT' or 'SEED_JOB'. (default 'JENKINS_ROOT')
- **additional-classpath** (*str*) – Newline separated list of additional classpath entries for the Job DSL scripts. All entries must be relative to the workspace root, e.g. build/classes/main. (optional)
- **sandbox** (*bool*) – Execute script inside of groovy sandbox (default false)

Example:

```
builders:
  - dsl:
      script-text: "job { name 'dsljob' }"
      ignore-existing: "true"
      removed-job-action: "DISABLE"
```

(continues on next page)

(continued from previous page)

```
      removed-view-action: "DELETE"
      lookup-strategy: "SEED_JOB"
      additional-classpath: "*.jar"
      sandbox: "true"
```

```
builders:
  - dsl:
      target: "jobs/**/*.groovy"
      ignore-existing: "true"
      removed-job-action: "DISABLE"
      removed-view-action: "DELETE"
      lookup-strategy: "SEED_JOB"
      additional-classpath: "*.jar"
```

fingerprint()

Adds the ability to generate fingerprints as build steps instead of waiting for a build to complete.

Requires the Jenkins Fingerprint Plugin.

Parameters

targets (*str*) – Files to fingerprint (default '')

Full Example:

```
builders:
  - fingerprint:
      targets: module/dist/**/*.zip
```

Minimal Example:

```
builders:
  - fingerprint
```

github-notifier()

Set pending build status on Github commit.

Requires the Jenkins Github Plugin.

Example:

```
builders:
  - github-notifier
```

gradle()

Execute gradle tasks.

Requires the Jenkins Gradle Plugin.

Parameters

- **tasks** (*str*) – List of tasks to execute
- **gradle-name** (*str*) – Use a custom gradle name (default '')
- **wrapper** (*bool*) – use gradle wrapper (default false)
- **executable** (*bool*) – make gradlew executable (default false)
- **switches** (*list*) – Switches for gradle, can have multiples
- **use-root-dir** (*bool*) – Whether to run the gradle script from the top level directory or from a different location (default false)

- **root-build-script-dir** (*str*) – If your workspace has the top-level build.gradle in somewhere other than the module root directory, specify the path (relative to the module root) here, such as ${workspace}/parent/ instead of just ${workspace}.
- **build-file** (*str*) – name of gradle build script (default 'build.gradle')
- **pass-system-properties** (*bool*) – Pass all parameters as System properties (default false)
- **pass-project-properties** (*bool*) – Pass all parameters as Project properties (default false)

Example:

```
builders:
  - gradle:
      build-file: "build.gradle"
      gradle-name: "gradle-1.2"
      wrapper: true
      executable: true
      use-root-dir: true
      root-build-script-dir: ${workspace}/tests
      pass-system-properties: true
      pass-project-properties: true
      switches:
        - "-g /foo/bar/.gradle"
        - "-PmavenUserName=foobar"
      tasks: |
             init
             build
             tests
```

grails()

Execute a grails build step.

Requires the Jenkins Grails Plugin.

Parameters

- **use-wrapper** (*bool*) – Use a grails wrapper (default false)
- **name** (*str*) – Select a grails installation to use (default '(Default)')
- **force-upgrade** (*bool*) – Run 'grails upgrade –non-interactive' first (default false)
- **non-interactive** (*bool*) – append –non-interactive to all build targets (default false)
- **targets** (*str*) – Specify target(s) to run separated by spaces (required)
- **server-port** (*str*) – Specify a value for the server.port system property (default '')
- **work-dir** (*str*) – Specify a value for the grails.work.dir system property (default '')
- **project-dir** (*str*) – Specify a value for the grails.project.work.dir system property (default '')
- **base-dir** (*str*) – Specify a path to the root of the Grails project (default '')
- **properties** (*str*) – Additional system properties to set (default '')
- **plain-output** (*bool*) – append –plain-output to all build targets (default false)
- **stack-trace** (*bool*) – append –stack-trace to all build targets (default false)
- **verbose** (*bool*) – append –verbose to all build targets (default false)
- **refresh-dependencies** (*bool*) – append –refresh-dependencies to all build targets (default false)

Full Example:

```
builders:
  - grails:
      use-wrapper: true
```

(continues on next page)

(continued from previous page)

```
      name: grails-2.2.2
      force-upgrade: true
      non-interactive: true
      targets: war ear
      server-port: 8003
      work-dir: ./grails-work
      project-dir: ./project-work
      base-dir: ./grails/project
      properties: program.name=foo
      plain-output: true
      stack-trace: true
      verbose: true
      refresh-dependencies: true
```

Minimal Example:

```
builders:
  - grails:
      targets: foo
```

groovy()

Execute a groovy script or command.

Requires the Jenkins Groovy Plugin.

Parameters

- **file** (*str*) – Groovy file to run. (Alternative: you can chose a command instead)
- **command** (*str*) – Groovy command to run. (Alternative: you can chose a script file instead)
- **version** (*str*) – Groovy version to use. (default '(Default)')
- **parameters** (*str*) – Parameters for the Groovy executable. (default '')
- **script-parameters** (*str*) – These parameters will be passed to the script. (default '')
- **properties** (*str*) – Instead of passing properties using the -D parameter you can define them here. (default '')
- **java-opts** (*str*) – Direct access to JAVA_OPTS. Properties allows only -D properties, while sometimes also other properties like -XX need to be setup. It can be done here. This line is appended at the end of JAVA_OPTS string. (default '')
- **class-path** (*str*) – Specify script classpath here. Each line is one class path item. (default '')

Minimal Example:

```
builders:
  - groovy:
      command: "println Hello"
```

Full Example:

```
builders:
  - groovy:
      command: "Some command"
      version: "Groovy 1.2"
      parameters: "parameters"
      script-parameters: "script parameters"
```

(continues on next page)

(continued from previous page)

```
    properties: "properties"
    java-opts: "java opts"
```

http-request()

This plugin sends a http request to an url with some parameters.

Requires the Jenkins HTTP Request Plugin.

Parameters

- **url** (*str*) – Specify an URL to be requested (required)
- **mode** (*str*) – The http mode of the request (default GET)
  - **mode values**
    - **GET**
    - **POST**
    - **PUT**
    - **DELETE**
    - **HEAD**
- **content-type** (*str*) – Add 'Content-type: foo' HTTP request headers where foo is the http content-type the request is using. (default NOT_SET)
- **accept-type** (*str*) – Add 'Accept: foo' HTTP request headers where foo is the http content-type to accept (default NOT_SET)
  - **content-type and accept-type values**
    - **NOT_SET**
    - **TEXT_HTML**
    - **APPLICATION_JSON**
    - **APPLICATION_TAR**
    - **APPLICATION_ZIP**
    - **APPLICATION_OCTETSTREAM**
- **output-file** (*str*) – Name of the file in which to write response data (default '')
- **time-out** (*int*) – Specify a timeout value in seconds (default 0)
- **console-log** (*bool*) – This allows you to turn off writing the response body to the log (default false)
- **pass-build** (*bool*) – Should build parameters be passed to the URL being called (default false)
- **valid-response-codes** (*str*) – Configure response code to mark an execution as success. You can configure simple code such as "200" or multiple codes separated by comma(',') e.g. "200,404,500" Interval of codes should be in format From:To e.g. "100:399". The default (as if empty) is to fail to 4xx and 5xx. That means success from 100 to 399 "100:399" To ignore any response code use "100:599". (default '')
- **valid-response-content** (*str*) – If set response must contain this string to mark an execution as success (default '')
- **authentication-key** (*str*) – Authentication that will be used before this request. Authentications are created in global configuration under a key name that is selected here.
- **custom-headers** (*list*) – list of header parameters
  - **custom-header**
    - **name** (*str*) – Name of the header
    - **value** (*str*) – Value of the header

Example:

```
builders:
  - http-request:
      url: http://example.com/jenkinsTest
```

```
builders:
  - http-request:
      url: http://example.com/jenkinsTest
      mode: POST
      pass-build: true
      content-type: TEXT_HTML
      accept-type: TEXT_HTML
      output-file: response_file.txt
      authentication-key: authenticationkey
      console-log: true
      time-out: 10
      valid-response-codes: 100:399
      valid-response-content: foo
      custom-headers:
        - name: header
          value: value
        - name: header2
          value: value2
```

inject()

Inject an environment for the job.

Requires the Jenkins EnvInject Plugin.

Parameters

- **properties-file** (*str*) – the name of the property file (optional)
- **properties-content** (*str*) – the properties content (optional)
- **script-file** (*str*) – the name of a script file to run (optional)
- **script-content** (*str*) – the script content (optional)

Example:

```
builders:
  - inject:
      properties-file: example.prop
      properties-content: EXAMPLE=foo-bar
      script-file: script.sh
      script-content: script content
```

jenkins-jira-issue-updater()

Updates issues in Atlassian JIRA as part of a Jenkins job.

Requires the Jenkins Jira Issue Updater Plugin.

Parameters

- **base-url** (*str*) – The base url of the rest API. (default '')
- **username** (*str*) – The Jira username (required)
- **password** (*str*) – The Jira password (required)
- **jql** (*str*) – The JQL used to select the issues to update. (required)
- **workflow** (*str*) – The Name of the workflow action to be executed. (default '')
- **comment** (*str*) – The Jira comment to be added. (default '')
- **custom-Id** (*str*) – The Jira custom field to be edited. (default '')
- **custom-value** (*str*) – Jira custom field value. (default '')
- **fail-if-error** (*bool*) – Fail this build if JQL returns error. ((default false)
- **fail-if-no-match** (*bool*) – Fail this build if no issues are matched. (default false)
- **fail-if-no-connection** (*bool*) – Fail this build if can't connect to Jira. (default false)

Minimal Example:

```
builders:
  - jenkins-jira-issue-updater:
      username: 'Username'
      password: 'Password'
      jql: 'jql'
```

Full Example:

```
builders:
  - jenkins-jira-issue-updater:
      base-url: url
      username: your-username
      password: your-password
      jql: project-key
      workflow: workflow-name
      comment: comment
      custom-Id: ID
      custom-value: value
      fail-if-error: true
      fail-if-no-match: true
      fail-if-no-connection: true
```

jms-messaging()

The JMS Messaging Plugin provides the following functionality:

- A build trigger to submit jenkins jobs upon receipt of a matching message.
- A builder that may be used to submit a message to the topic upon the completion of a job
- A post-build action that may be used to submit a message to the topic upon the completion of a job

JMS Messaging provider types supported:

- ActiveMQ
- FedMsg

Requires the Jenkins JMS Messaging Plugin Pipeline Plugin.

Parameters

- **override-topic** (*str*) – If you need to override the default topic. (default '')
- **provider-name** (*str*) – Name of message provider setup in the global config. (default '')
- **msg-type** (*str*) – A message type (default 'CodeQualityChecksDone')
- **msg-props** (*str*) – Message header to publish. (default '')
- **msg-content** (*str*) – Message body to publish. (default '')

Full Example:

```
builders:
  - jms-messaging:
      override-topic: org.centos.stage.ci.pipeline.compose.complete
      provider-name: fedmsg
      msg-type: Custom
      msg-props: |
        topic=org.centos.prod.ci.pipeline.compose.complete
        username=fedora-atomic
      msg-content: |
        {
          "build_url": "${BUILD_URL}",
          "compose_url": "<full-url-to-compose>",
```

(continues on next page)

(continued from previous page)

```
          "build_id": "${BUILD_ID}",
          "ref": "fedora/rawhide/${basearch}/atomic-host",
          "rev": "<sha of the commit from dist-git>",
          "namespace": "rpms",
          "repo": "php-simplepie",
          "status": "<success/failure/aborted>",
          "test_guidance": "<comma-separated-list-of-test-suites-to-run>"
        }
```

Minimal Example:

```
builders:
  - jms-messaging:
      provider-name: fedmsg
      msg-type: CodeQualityChecksDone
      msg-props: test
      msg-content: test
```

kmap()

Publish mobile applications to your Keivox KMAP Private Mobile App Store.

Requires the Jenkins Keivox KMAP Private Mobile App Store Plugin.

Parameters

- **username** (*str*) – KMAP's user email with permissions to upload/publish applications to KMAP (required)
- **password** (*str*) – Password for the KMAP user uploading/publishing applications (required)
- **url** (*str*) – KMAP's url. This url must always end with "/kmap-client/". For example: http://testing.example.org/kmap-client/ (required)
- **categories** (*str*) – Categories' names. If you want to add the application to more than one category, write the categories between commas. (required)
- **file-path** (*str*) – Path to the application's file (required)
- **app-name** (*str*) – KMAP's application name (required)
- **bundle** (*str*) – Bundle indentifier (default '')
- **version** (*str*) – Application's version (required)
- **description** (*str*) – Application's description (default '')
- **icon-path** (*str*) – Path to the application's icon (default '')
- **publish-optional** (*bool*) – Publish application after it has been uploaded to KMAP (default false)

  publish-optional

  - **groups ('str') – groups' names to publish the application** (default '')
  - **users ('str') – users' names to publish the application** (default '')
  - **notify-users ('bool') – Send notifications to the users and** groups when publishing the application (default false)

Minimal Example:

```
builders:
  - kmap:
      username: user@user.com
      password: password
      url: http://foo.com/kmap-client/
```

(continues on next page)

(continued from previous page)

```
      categories: Productivity
      file-path: ${WORKSPACE}/path/to/file.extension
      app-name: AppName
      version: b${BUILD_NUMBER}_r${SVN_REVISION}
```

Full Example:

```
builders:
  - kmap:
      username: user@user.com
      password: password
      url: http://foo.com/kmap-client/
      categories: Productivity
      file-path: ${WORKSPACE}/path/to/file.extension
      app-name: AppName
      bundle: foo.apk
      version: b${BUILD_NUMBER}_r${SVN_REVISION}
      description: description
      icon-path: ${WORKSPACE}/target/application.png
      publish-optional: true
      groups: MobileUsers
      users: user@user.com
      notify-users: true
```

managed-script()

This step allows you to reference and execute a centrally managed script within your build.

Requires the Jenkins Managed Scripts Plugin.

Parameters

- **script-id** (*str*) – Id of script to execute (required)
- **type** (*str*) – Type of managed file (default script)
  - **type values**
    - **batch**: Execute managed windows batch
    - **script**: Execute managed script
- **args** (*list*) – Arguments to be passed to referenced script

Example:

```
builders:
  - managed-script:
      script-id: org.jenkinsci.plugins.managedscripts.ScriptConfig1401886156431
      type: script
      args:
        - arg1
        - arg2
```

```
builders:
  - managed-script:
      script-id: org.jenkinsci.plugins.managedscripts.WinBatchConfig1402391729132
      type: batch
      args:
        - arg1
        - arg2
```

maven-builder()

Execute Maven3 builder

Allows your build jobs to deploy artifacts automatically to Artifactory.

Requires the Jenkins Artifactory Plugin.

Parameters

- **name** (*str*) – Name of maven installation from the configuration (required)
- **pom** (*str*) – Location of pom.xml (default 'pom.xml')
- **goals** (*str*) – Goals to execute (required)
- **maven-opts** (*str*) – Additional options for maven (default '')

Example:

```
builders:
      - maven-builder:
          name: mvn3
          pom: modules/pom.xml
          goals: clean install
```

maven-target()

Execute top-level Maven targets.

Requires the Jenkins Config File Provider Plugin for the Config File Provider "settings" and "global-settings" config.

Parameters

- **goals** (*str*) – Goals to execute
- **properties** (*str*) – Properties for maven, can have multiples
- **pom** (*str*) – Location of pom.xml (default 'pom.xml')
- **private-repository** (*bool*) – Use private maven repository for this job (default false)
- **maven-version** (*str*) – Installation of maven which should be used (optional)
- **java-opts** (*str*) – java options for maven, can have multiples, must be in quotes (optional)
- **settings** (*str*) – Path to use as user settings.xml It is possible to provide a ConfigFileProvider settings file, such as see CFP Example below. (optional)
- **settings-type** (*str*) – Type of settings file file|cfp. (default file)
- **global-settings** (*str*) – Path to use as global settings.xml It is possible to provide a ConfigFileProvider settings file, such as see CFP Example below. (optional)
- **global-settings-type** (*str*) – Type of settings file file|cfp. (default file)

Example:

```
builders:
        - maven-target:
            maven-version: Maven3
            pom: parent/pom.xml
            goals: clean
            private-repository: true
            properties:
              - foo=bar
              - bar=foo
            java-opts:
              - "-Xms512m -Xmx1024m"
              - "-XX:PermSize=128m -XX:MaxPermSize=256m"
            settings: mvn/settings.xml
            global-settings: mvn/globalsettings.xml
```

CFP Example:

```
postbuilders:
      - maven-target:
          maven-version: mvn30
          goals: clean verify
          settings: org.jenkinsci.plugins.configfiles.maven.
↪MavenSettingsConfig0123456789012
          global-settings: org.jenkinsci.plugins.configfiles.maven.
↪GlobalMavenSettingsConfig0123456789012
```

msbuild()

Build .NET project using msbuild.

Requires the Jenkins :jenkins-plugins:'MSBuild Plugin <msbuild>`.

Parameters

- **msbuild-version** (*str*) – which msbuild configured in Jenkins to use (default '(Default)')
- **solution-file** (*str*) – location of the solution file to build (required)
- **extra-parameters** (*str*) – extra parameters to pass to msbuild (default '')
- **pass-build-variables** (*bool*) – should build variables be passed to msbuild (default true)
- **continue-on-build-failure** (*bool*) – should the build continue if msbuild returns an error (default false)
- **unstable-if-warnings** (*bool*) – If set to true and warnings on compilation, the build will be unstable (>=1.20) (default false)

Full Example:

```
builders:
  - msbuild:
      solution-file: "MySolution.sln"
      msbuild-version: "msbuild-4.0"
      extra-parameters: "/maxcpucount:4"
      pass-build-variables: False
      continue-on-build-failure: True
      unstable-if-warnings: True
```

Minimal Example:

```
builders:
  - msbuild:
      solution-file: MySolution.sln
```

multijob()

Define a multijob phase.

Requires the Jenkins Multijob Plugin.

This builder may only be used in `jenkins_jobs.modules.project_multijob.MultiJob` projects.

Parameters

- **name** (*str*) – MultiJob phase name
- **condition** (*str*) – when to trigger the other job. Can be: 'SUCCESSFUL', 'UNSTABLE', 'COMPLETED', 'FAILURE', 'ALWAYS'. (default 'SUCCESSFUL')
- **execution-type** (*str*) – Define how to run jobs in a phase: sequentially or parallel. Can be: 'PARALLEL', 'SEQUENTIALLY' (default 'PARALLEL')
- **projects** (*list*) – list of projects to include in the MultiJob phase

Project

- **name** (*str*) – Project name
- **alias** (*str*) – Project alias, which will be shown in MultiJob Overview. Helpful when working with the same project multiple times with different configurations
- **current-parameters** (*bool*) – Pass current build parameters to the other job (default false)
- **node-label-name** (*str*) – Define a list of nodes on which the job should be allowed to be executed on. Requires NodeLabel Parameter Plugin (optional)
- **node-label** (*str*) – Define a label of 'Restrict where this project can be run' on the fly. Requires NodeLabel Parameter Plugin (optional)
- **node-parameters** (*bool*) – Use the same Node for the triggered builds that was used for this build. (optional)
- **git-revision** (*bool*) – Pass current git-revision to the other job (default false)
- **property-file** (*str*) – Pass properties from file to the other job (optional)
- **predefined-parameters** (*str*) – Pass predefined parameters to the other job (optional)
- **abort-all-job** (*bool*) – Kill allsubs job and the phase job, if this subjob is killed (default false)
- **aggregate-results** (*bool*) – Aggregate test results. (default false)
- **enable-condition** (*str*) – Condition to run the job in groovy script format (optional)
- **kill-phase-on** (*str*) – Stop the phase execution on specific job status. Can be 'FAILURE', 'UNSTABLE', 'NEVER'. (optional)
- **restrict-matrix-project** (*str*) – Filter that restricts the subset of the combinations that the downstream project will run (optional)
- **retry (*dict*): Enable retry strategy (optional)**

  retry

  - **max-retry** (*int*) – Max number of retries (default 0)
  - **strategy-path** (*str*) – Parsing rules path (required)

Example:

```
builders:
  - multijob:
      name: PhaseOne
      condition: SUCCESSFUL
      execution-type: PARALLEL
      projects:
        - name: PhaseOneJobA
          current-parameters: true
          node-label-name: "vm_name"
          node-label: "agent-${BUILD_NUMBER}"
          git-revision: true
```

(continues on next page)

(continued from previous page)

```
        abort-all-job: true
      - name: PhaseOneJobB
        current-parameters: true
        property-file: build.props
- multijob:
    name: PhaseTwo
    condition: UNSTABLE
    execution-type: SEQUENTIALLY
    projects:
      - name: PhaseTwoJobA
        current-parameters: true
        predefined-parameters: foo=bar
        node-parameters: true
        aggregate-results: true
      - name: PhaseTwoJobB
        current-parameters: false
        kill-phase-on: UNSTABLE
        enable-condition: "${BUILDNUMBER} % 2 == 1"
        restrict-matrix-project: 'JVM_VARIANT == "server"'
- multijob:
    name: PhaseThree
    condition: ALWAYS
    projects:
      - name: PhaseThreeJobA
        current-parameters: true
        kill-phase-on: FAILURE
- multijob:
    name: PhaseFour
    execution-type: PARALLEL
    projects:
      - name: PhaseFourJobA
        retry:
            max-retry: 3
            strategy-path: "/PhaseFour/PhaseFourRetry.prop"
- multijob:
    name: PhaseFive
    projects:
      - name: PhaseFiveJobA
        alias: PhaseFiveJobRunA
      - name: PhaseFiveJobA
        alias: PhaseFiveJobRunB
```

nexus-artifact-uploader()

To upload result of a build as an artifact in Nexus without the need of Maven.

Requires the Jenkins Nexus Artifact Uploader Plugin.

Parameters

- **protocol** (*str*) – Protocol to use to connect to Nexus (default https)
- **nexus_url** (*str*) – Nexus url (without protocol) (default '')
- **nexus_user** (*str*) – Username to upload artifact to Nexus (default '')
- **nexus_password** (*str*) – Password to upload artifact to Nexus (default '')
- **group_id** (*str*) – GroupId to set for the artifact to upload (default '')
- **artifact_id** (*str*) – ArtifactId to set for the artifact to upload (default '')

- **version** (*str*) – Version to set for the artifact to upload (default '')
- **packaging** (*str*) – Packaging to set for the artifact to upload (default '')
- **type** (*str*) – Type to set for the artifact to upload (default '')
- **classifier** (*str*) – Classifier to set for the artifact to upload (default '')
- **repository** (*str*) – In which repository to upload the artifact (default '')
- **file** (*str*) – File which will be the uploaded artifact (default '')
- **credentials_id** (*str*) – Credentials to use (instead of password) (default '')

File Example:

```
builders:
  - nexus-artifact-uploader:
      nexus_url: 'nexus.org'
      group_id: 'com.example'
      artifact_id: 'artifact'
      version: '1.0'
      packaging: 'pom'
      type: 'zip'
      repository: 'my-hosted-repo'
      file: '/var/lib/jenkins/workspace/my_job/result.zip'
```

nexus-iq-policy-evaluator()

Integrates the Nexus Lifecycle into a Jenkins job. This function triggers 'Invokes Nexus Policy Evaluation'.

Requires the Jenkins Nexus Platform Plugin.

Parameters

- **stage** (*str*) – Controls the stage the policy evaluation will be run against on the Nexus IQ Server (required)
  - **stage values**
    - **build**
    - **stage-release**
    - **release**
    - **operate**
- **application-type** (*dict*) – Specifies an IQ Application (default manual)
  - **application-type values**
    - **manual**
    - **selected**
- **application-id** (*str*) – Specify the IQ Application ID (required)
- **scan-patterns** (*list*) – List of Ant-style patterns relative to the workspace root that denote files/archives to be scanned (default [])
- **fail-build-network-error** (*bool*) – Controls the build outcome if there is a failure in communicating with the Nexus IQ Server (default false)

Minimal Example:

```
builders:
  - nexus-iq-policy-evaluator:
      stage: 'build'
      application-id: 'nexus-iq-application-id001'
```

Full Example:

```
builders:
  - nexus-iq-policy-evaluator:
      stage: 'stage-release'
      application-type: 'selected'
```

(continues on next page)

(continued from previous page)

```
      application-id: 'nexus-iq-application-id002'
      scan-patterns:
        - '**/target/*.war'
        - '**/target/*.ear'
      fail-build-network-error: true
```

nexus-repo-manager()

Allows for artifacts selected in Jenkins packages to be available in Nexus Repository Manager.

Requires the Jenkins Nexus Platform Plugin.

Parameters

- **instance-id** (*str*) – The ID of the Nexus Instance (required)
- **repo-id** (*str*) – The ID of the Nexus Repository (required)

Minimal Example:

```
builders:
    - nexus-repo-manager:
        instance-id: Nexus-Repo-Instance
        repo-id: Releases
```

nodejs()

This plugin allows you to execute NodeJS scripts as a job build step.

Requires the Jenkins NodeJS Plugin.

Parameters

- **name** (*str*) – NodeJS installation name
- **script** (*str*) – NodeJS script (required)
- **config-id** (*str*) – ID of npmrc config file, which is the last field (a 32-digit hexadecimal code) of the path of URL visible after you clicked the file under Jenkins Managed Files.

Minimal Example:

```
builders:
  - nodejs:
      script: "console.log('Some output');"
```

Full Example:

```
builders:
  - nodejs:
      name: "NodeJS_8.1"
      script: "console.log('Some output');"
      config-id: "e3757442-7c21-4a65-a1ff-6c70f5c6df34"
```

openshift-build-verify()

Performs the equivalent of an ‘oc get builds` command invocation for the provided buildConfig key provided; once the list of builds are obtained, the state of the latest build is inspected for up to a minute to see if it has completed successfully.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default ‘https://openshift.default.svc.cluster.local’)

- **bld-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-build-verify:
      api-url: https://openshift.example.local.url/
      bld-cfg: front
      namespace: test-build
      auth-token: ose-key-buildv1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-build-verify
```

openshift-builder()

Perform builds in OpenShift for the job.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **bld-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **commit-ID** (*str*) – The value here is what you supply with the –commit option when invoking the OpenShift *oc start-build* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)
- **build-name** (*str*) – TThe value here is what you supply with the –from-build option when invoking the OpenShift *oc start-build* command. (default '')
- **show-build-logs** (*bool*) – Indicates whether the build logs get dumped to the console of the Jenkins build. (default false)

Full Example:

```
builders:
  - openshift-builder:
      api-url: https://openshift.example.local.url/
      bld-cfg: front
      namespace: test9
      auth-token: ose-builder1
      commit-ID: ae489f7d
      verbose: true
```

(continues on next page)

(continued from previous page)

```
    build-name: ose-test-build
    show-build-logs: true
```

Minimal Example:

```
builders:
  - openshift-builder
```

openshift-creator()

Performs the equivalent of an oc create command invocation; this build step takes in the provided JSON or YAML text, and if it conforms to OpenShift schema, creates whichever OpenShift resources are specified.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **jsonyaml** (*str*) – The JSON or YAML formatted text that conforms to the schema for defining the various OpenShift resources. (default '')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-creator:
      api-url: https://openshift.example.local.url/
      jsonyaml: 'front: back'
      namespace: test6
      auth-token: ose-key-creator1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-creator
```

openshift-dep-verify()

Determines whether the expected set of DeploymentConfig's, ReplicationController's, and active replicas are present based on prior use of the scaler (2) and deployer (3) steps

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default https://openshift.default.svc.cluster.local)
- **dep-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default frontend)
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default test)
- **replica-count** (*int*) – The value here should be whatever the number of pods you want started for the deployment. (default 0)

- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-dep-verify:
      api-url: https://openshift.example.local.url/
      dep-cfg: front
      namespace: test6
      replica-count: 4
      auth-token: ose-key-dep-verify1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-dep-verify
```

openshift-deployer()

Start a deployment in OpenShift for the job.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **dep-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-deployer:
      api-url: https://openshift.example.local.url/
      dep-cfg: front
      namespace: test3
      auth-token: ose-key-deployer1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-deployer
```

openshift-img-tagger()

Performs the equivalent of an oc tag command invocation in order to manipulate tags for images in OpenShift ImageStream's

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **test-tag** (*str*) – The equivalent to the name supplied to a *oc get service* command line invocation. (default 'origin-nodejs-sample:latest')
- **prod-tag** (*str*) – The equivalent to the name supplied to a *oc get service* command line invocation. (default 'origin-nodejs-sample:prod')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-img-tagger:
      api-url: https://openshift.example.local.url/
      test-tag: origin-nodejs-sample:test
      prod-tag: origin-nodejs-sample:production
      namespace: test5
      auth-token: ose-key-img1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-img-tagger
```

openshift-scaler()

Scale deployments in OpenShift for the job.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **dep-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **replica-count** (*int*) – The value here should be whatever the number of pods you want started for the deployment. (default 0)
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-scaler:
      api-url: https://openshift.example.local.url/
      dep-cfg: front
      namespace: test2
      replica-count: 4
```

(continues on next page)

(continued from previous page)

```
    auth-token: ose-key-scaler1
    verbose: true
```

Minimal Example:

```
builders:
  - openshift-scaler
```

openshift-svc-verify()

Verify a service is up in OpenShift for the job.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **svc-name** (*str*) – The equivalent to the name supplied to a *oc get service* command line invocation. (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
builders:
  - openshift-svc-verify:
      api-url: https://openshift.example.local.url/
      svc-name: front
      namespace: test4
      auth-token: ose-key-svc-verify1
      verbose: true
```

Minimal Example:

```
builders:
  - openshift-svc-verify
```

powershell()

Execute a powershell command.

Requires the Powershell Plugin.

Parameter

the powershell command to execute

Example:

```
builders:
  - powershell: "foo/foo.ps1"
```

publish-over-cifs()

Upload files via CIFS.

Requires the Jenkins Publish over CIFS Plugin.

Parameters

- **site** (*str*) – name of the ssh site
- **target** (*str*) – destination directory
- **target-is-date-format** (*bool*) – whether target is a date format. If true, raw text should be quoted (default false)
- **clean-remote** (*bool*) – should the remote directory be deleted before transferring files (default false)
- **source** (*str*) – source path specifier
- **excludes** (*str*) – excluded file pattern (optional)
- **remove-prefix** (*str*) – prefix to remove from uploaded file paths (optional)
- **fail-on-error** (*bool*) – fail the build if an error occurs (default false)
- **flatten** (*bool*) – only create files on the server, don't create directories (default false)

Example:

```
builders:
    - publish-over-cifs:
        site: 'cifs.share'
        target: 'dest/dir'
        source: 'base/source/dir/**'
        remove-prefix: 'base/source/dir'
        excludes: '**/*.excludedfiletype'
        flatten: true
```

publish-over-ssh()

Send files or execute commands over SSH.

Requires the Jenkins Publish over SSH Plugin.

Parameters

- **site** (*str*) – name of the ssh site
- **target** (*str*) – destination directory
- **target-is-date-format** (*bool*) – whether target is a date format. If true, raw text should be quoted (default false)
- **clean-remote** (*bool*) – should the remote directory be deleted before transferring files (default false)
- **source** (*str*) – source path specifier
- **command** (*str*) – a command to execute on the remote server (optional)
- **timeout** (*int*) – timeout in milliseconds for the Exec command (optional)
- **use-pty** (*bool*) – run the exec command in pseudo TTY (default false)
- **excludes** (*str*) – excluded file pattern (optional)
- **remove-prefix** (*str*) – prefix to remove from uploaded file paths (optional)
- **fail-on-error** (*bool*) – fail the build if an error occurs (default false)

Example:

```
builders:
    - publish-over-ssh:
        site: 'server.example.com'
        target: 'dest/dir'
        source: 'base/source/dir/**'
        timeout: 1800000
```

python()

Execute a python command. Requires the Jenkins Python plugin.

Parameters

parameter (*str*) – the python command to execute

Example:

```
builders:
  - python: 'import foobar'
```

runscope()

Execute a Runscope test.

Requires the Jenkins Runscope Plugin.

Parameters

- **test-trigger-url** (*str*) – Trigger URL for test. (required)
- **access-token** (*str*) – OAuth Personal Access token. (required)
- **timeout** (*int*) – Timeout for test duration in seconds. (default 60)

Minimal Example:

```
builders:
  - runscope:
      test-trigger-url: "https://api.runscope.com/radar/xxxxxxxx-xxxx-xxxx-xxxx-
↪xxxxxxxxxxxx/trigger"
      access-token: "123456"
```

Full Example:

```
builders:
  - runscope:
      test-trigger-url: "https://api.runscope.com/radar/xxxxxxxx-xxxx-xxxx-xxxx-
↪xxxxxxxxxxxx/trigger"
      access-token: "123456"
      timeout: 123
```

saltstack()

Send a message to Salt API.

Requires the Jenkins saltstack plugin.

Parameters

- **servername** (*str*) – Salt master server name (required)
- **authtype** (*str*) – Authentication type ('pam' or 'ldap', default 'pam')
- **credentials** (*str*) – Credentials ID for which to authenticate to Salt master (required)
- **target** (*str*) – Target minions (default '')
- **targettype** (*str*) – Target type ('glob', 'pcre', 'list', 'grain', 'pillar', 'nodegroup', 'range', or 'compound', default 'glob')
- **function** (*str*) – Function to execute (default '')
- **arguments** (*str*) – Salt function arguments (default '')
- **kwarguments** (*str*) – Salt keyword arguments (default '')
- **saveoutput** (*bool*) – Save Salt return data into environment variable (default false)
- **clientinterface** (*str*) – Client interface type ('local', 'local-batch', or 'runner', default 'local')
- **wait** (*bool*) – Wait for completion of command (default false)
- **polltime** (*str*) – Number of seconds to wait before polling job completion status (default '')
- **batchsize** (*str*) – Salt batch size, absolute value or %-age (default 100%)
- **mods** (*str*) – Mods to runner (default '')
- **setpillardata** (*bool*) – Set Pillar data (default false)
- **pillarkey** (*str*) – Pillar key (default '')
- **pillarvalue** (*str*) – Pillar value (default '')

Minimal Example:

```
builders:
  - saltstack:
      servername: '{{SALT_MASTER}}'
      credentials: 'credentials ID'
```

Full Example:

```
builders:
  - saltstack:
      servername: '{{SALT_MASTER}}'
      credentials: 'credentials ID'
      clientinterface: runner
      mods: runner_mods
      setpillardata: true
      pillarkey: pkey
      pillarvalue: pvalue
      wait: true
      polltime: 10
      target: '{{HOSTS}}'
      targettype: list
      function: pkg.update
      saveoutput: true
```

sbt()

Execute a sbt build step.

Requires the Jenkins Sbt Plugin.

Parameters

- **name** (*str*) – Select a sbt installation to use. If no name is provided, the first in the list of defined SBT builders will be used. (default to first in list)
- **jvm-flags** (*str*) – Parameters to pass to the JVM (default '')
- **actions** (*str*) – Select the sbt tasks to execute (default '')
- **sbt-flags** (*str*) – Add flags to SBT launcher (default '-Dsbt.log.noformat=true')
- **subdir-path** (*str*) – Path relative to workspace to run sbt in (default '')

Example:

```
builders:
  - sbt:
      name: "default"
      actions: "clean package"
      jvm-flags: "-Xmx8G"
```

scan-build()

This plugin allows you configure a build step that will execute the Clang scan-build static analysis tool against an XCode project.

The scan-build report has to be generated in the directory `${WORKSPACE}/clangScanBuildReports` for the publisher to find it.

Requires the Jenkins Clang Scan-Build Plugin.

Parameters

- **target** (*str*) – Provide the exact name of the XCode target you wish to have compiled and analyzed (required)
- **target-sdk** (*str*) – Set the simulator version of a currently installed SDK (default iphonesimulator)

- **config** (*str*) – Provide the XCode config you wish to execute scan-build against (default Debug)
- **clang-install-name** (*str*) – Name of clang static analyzer to use (default '')
- **xcode-sub-path** (*str*) – Path of XCode project relative to the workspace (default '')
- **workspace** (*str*) – Name of workspace (default '')
- **scheme** (*str*) – Name of scheme (default '')
- **scan-build-args** (*str*) – Additional arguments to clang scan-build (default –use-analyzer Xcode)
- **xcode-build-args** (*str*) – Additional arguments to XCode (default -derivedDataPath $WORKSPACE/build)
- **report-folder** (*str*) – Folder where generated reports are located (>=1.7) (default clangScanBuildReports)

Full Example:

```
builders:
  - scan-build:
      target: path/to/target
      target-sdk: iphonesimulator
      config: Debug
      clang-install-name: Analyzer
      xcode-sub-path: myProj/subfolder
      workspace: workspace
      scheme: SchemeName
      scan-build-args: --use-analyzer Xcode
      xcode-build-args: -derivedDataPath $WORKSPACE/build
      report-folder: clangScanBuildReports
```

Minimal Example:

```
builders:
  - scan-build:
      target: path/to/target
```

shell()

Execute a shell command.

There are two ways of configuring the builder, with a plain string to execute:

Parameters

parameter (*str*) – the shell command to execute

Or with a mapping that allows other parameters to be passed:

Parameters

- **command** (*str*) – the shell command to execute
- **unstable-return** (*int*) – the shell exit code to interpret as an unstable build result

Example:

```
builders:
  - shell: "make test"
```

```
builders:
  - shell:
      command: "make test"
      unstable-return: 3
```

shining-panda()

Execute a command inside various python environments.

Requires the Jenkins ShiningPanda plugin.

Parameters

build-environment (*str*) – Building environment to set up (required).

build-environment values

- **python**: Use a python installation configured in Jenkins.
- **custom**: Use a manually installed python.
- **virtualenv**: Create a virtualenv

For the **python** environment

Parameters

python-version (*str*) – Name of the python installation to use. Must match one of the configured installations on server configuration (default 'System-CPython-2.7')

For the **custom** environment:

Parameters

home (*str*) – path to the home folder of the custom installation (required)

For the **virtualenv** environment:

Parameters

- **python-version** (*str*) – Name of the python installation to use. Must match one of the configured installations on server configuration (default 'System-CPython-2.7')
- **name** (*str*) – Name of this virtualenv. Two virtualenv builders with the same name will use the same virtualenv installation (optional)
- **clear** (*bool*) – If true, delete and recreate virtualenv on each build. (default false)
- **use-distribute** (*bool*) – if true use distribute, if false use setuptools. (default true)
- **system-site-packages** (*bool*) – if true, give access to the global site-packages directory to the virtualenv. (default false)

Common to all environments:

Parameters

- **nature** (*str*) – Nature of the command field. (default shell)

  nature values

  - **shell**: execute the Command contents with default shell
  - **xshell**: like **shell** but performs platform conversion first
  - **python**: execute the Command contents with the Python executable
- **command** (*str*) – The command to execute
- **ignore-exit-code** (*bool*) – mark the build as failure if any of the commands exits with a non-zero exit code. (default false)

Examples:

```
builders:
    - shining-panda:
        build-environment: python
        python-version: System-CPython-2.7
        nature: python
        command: setup.py build
        ignore-exit-code: false
```

```
builders:
    - shining-panda:
        build-environment: custom
        home: /usr/local/lib/custom-python-27
        nature: xshell
        command: |
            cd $HOME/build
            python setup.py build
```

(continues on next page)

(continued from previous page)

```
        ignore-exit-code: true
```

```
builders:
    - shining-panda:
        build-environment: virtualenv
        python-version: System-CPython-2.7
        nature: shell
        command: python setup.py build
        name: virtvenv1
        clear: true
        use-distribute: true
        system-site-packages: true
        ignore-exit-code: true
```

sonar()

Invoke standalone Sonar analysis.

Requires the Jenkins Sonar Plugin.

Parameters

- **sonar-name** (*str*) – Name of the Sonar installation.
- **sonar-scanner** (*str*) – Name of the Sonar Scanner.
- **task** (*str*) – Task to run. (default '')
- **project** (*str*) – Path to Sonar project properties file. (default '')
- **properties** (*str*) – Sonar configuration properties. (default '')
- **java-opts** (*str*) – Java options for Sonnar Runner. (default '')
- **additional-arguments** (*str*) – additional command line arguments (default '')
- **jdk** (*str*) – JDK to use (inherited from the job if omitted). (optional)

Example:

```
builders:
  - sonar:
      sonar-name: Sonar
      scanner-name: scanner-3.x
      task: views
      project: sonar-project.properties
      properties: sonar.views.list=myview1,myview2
      java-opts: -Xmx512m
      additional-arguments: -X
```

sonatype-clm()

Requires the Jenkins Sonatype CLM Plugin.

WARNING: This plugin appears to be deprecated. There does not seem to be any place where it is available for download.

Try the *nexus-artifact-uploader* plugin instead.

Parameters

- **value** (*str*) – Select CLM application from a list of available CLM applications or specify CLM Application ID (default list)
- **application-name** (*str*) – Determines the policy elements to associate with this build. (required)
- **username** (*str*) – Username on the Sonatype CLM server. Leave empty to use the username configured at global level. (default '')

- **password** (*str*) – Password on the Sonatype CLM server. Leave empty to use the password configured at global level. (default '')
- **fail-on-clm-server-failure** (*bool*) – Controls the build outcome if there is a failure in communicating with the CLM server. (default false)
- **stage** (*str*) – Controls the stage the policy evaluation will be run against on the CLM server. Valid stages: build, stage-release, release, operate. (default 'build')
- **scan-targets** (*str*) – Pattern of files to include for scanning. (default '')
- **module-excludes** (*str*) – Pattern of files to exclude. (default '')
- **advanced-options** (*str*) – Options to be set on a case-by-case basis as advised by Sonatype Support. (default '')

Minimal Example:

```
builders:
    - sonatype-clm:
        application-name: jenkins-job-builder
```

Full Example:

```
builders:
    - sonatype-clm:
        value: manual
        application-name: jenkins-job-builder
        fail-on-clm-server-failure: true
        stage: release
        scan-targets: '**/*.jar'
        module-excludes: '**/my-module/target/**'
        advanced-options: 'test'
        username: bar
        password: 06XQY39LHGACt3r3kzSULg==
```

ssh-builder()

Executes command on remote host

Requires the Jenkins SSH plugin.

Parameters

- **ssh-user-ip** (*str*) – user@ip:ssh_port of machine that was defined in jenkins according to SSH plugin instructions
- **command** (*str*) – command to run on remote server

Example:

```
builders:
  - ssh-builder:
      ssh-user-ip: foo@bar:22
      command: echo foo
```

system-groovy()

Execute a system groovy script or command.

Requires the Jenkins Groovy Plugin.

Parameters

- **file** (*str*) – Groovy file to run. (Alternative: you can chose a command instead)
- **command** (*str*) – Groovy command to run. (Alternative: you can choose a script file instead)
- **sandbox** (*bool*) – Execute script inside of groovy sandbox (>=2.0) (default false)

- **bindings** (*str*) – Define variable bindings (in the properties file format). Specified variables can be addressed from the script. (optional)
- **class-path** (*list*) – List of script class paths. (optional)

Examples:

```
builders:
  - system-groovy:
      file: "test.groovy"
```

```
builders:
  - system-groovy:
      command: "println 'Hello'"
      bindings: "EXAMPLE=foo-bar"
      class-path: "file:/home/user/example.jar"
      sandbox: true
```

tox()

Use tox to build a multi-configuration project.

Requires the Jenkins ShiningPanda plugin.

Parameters

- **ini** (*str*) – The TOX configuration file path (default tox.ini)
- **recreate** (*bool*) – If true, create a new environment each time (default false)
- **toxenv-pattern** (*str*) – The pattern used to build the TOXENV environment variable. (optional)

Example:

```
builders:
  - tox:
      recreate: True
```

trigger-builds()

Trigger builds of other jobs.

Requires the Jenkins Parameterized Trigger Plugin.

Parameters

- **project** (*list*) – the Jenkins project to trigger
- **predefined-parameters** (*str*) – key/value pairs to be passed to the job (optional)
- **bool-parameters** (*list*) –
  Bool
  - **name** (*str*) – Parameter name
  - **value** (*bool*) – Value to set (default false)
- **property-file** (*str*) – Pass properties from file to the other job (optional)
- **property-file-fail-on-missing** (*bool*) – Don't trigger if any files are missing (default true)
- **current-parameters** (*bool*) – Whether to include the parameters passed to the current build to the triggered job (default false)
- **node-label-name** (*str*) – Define a name for the NodeLabel parameter to be set. Used in conjunction with node-label. Requires NodeLabel Parameter Plugin (optional)
- **node-label** (*str*) – Label of the nodes where build should be triggered. Used in conjunction with node-label-name. Requires NodeLabel Parameter Plugin (optional)
- **restrict-matrix-project** (*str*) – Filter that restricts the subset of the combinations that the triggered job will run (optional)
- **svn-revision** (*bool*) – Whether to pass the svn revision to the triggered job (optional)
- **git-revision** (*dict*) – Passes git revision to the triggered job (optional).

  - **combine-queued-commits** (bool): Whether to combine queued git hashes or not (default false)
- `block` (*bool*) – whether to wait for the triggered jobs to finish or not (default false)
- `block-thresholds` (*dict*) – Fail builds and/or mark as failed or unstable based on thresholds. Only apply if block parameter is true (optional)

  block-thresholds
  - **build-step-failure-threshold** (*str*) - build step failure threshold, valid values are 'never', 'SUCCESS', 'UNSTABLE', or 'FAILURE'. (default 'FAILURE')
  - **unstable-threshold** (*str*) - unstable threshold, valid values are 'never', 'SUCCESS', 'UNSTABLE', or 'FAILURE'. (default 'UNSTABLE')
  - **failure-threshold** (*str*) - overall failure threshold, valid values are 'never', 'SUCCESS', 'UNSTABLE', or 'FAILURE'. (default 'FAILURE')
- `same-node` (*bool*) – Use the same node for the triggered builds that was used for this build (optional)
- `parameter-factories` (*list*) – list of parameter factories

  Factory
  - **factory** (*str*) **filebuild** – For every property file, invoke one build
  - **file-pattern** (*str*) – File wildcard pattern
  - **no-files-found-action** (*str*) – Action to perform when no files found. Valid values 'FAIL', 'SKIP', or 'NOPARMS'. (default 'SKIP')

  Factory
  - **factory** (*str*) **binaryfile** – For every matching file, invoke one build
  - **file-pattern** (*str*) – Artifact ID of the artifact
  - **no-files-found-action** (*str*) – Action to perform when no files found. Valid values 'FAIL', 'SKIP', or 'NOPARMS'. (default 'SKIP')

  Factory
  - **factory** (*str*) **counterbuild** – Invoke i=0…N builds
  - **from** (*int*) – Artifact ID of the artifact
  - **to** (*int*) – Version of the artifact
  - **step** (*int*) – Classifier of the artifact
  - **parameters** (*str*) – KEY=value pairs, one per line (default '')
  - **validation-fail** (*str*) – Action to perform when stepping validation fails. Valid values 'FAIL', 'SKIP', or 'NOPARMS'. (default 'FAIL')

  Factory
  - **factory** (*str*) **allnodesforlabel** – Trigger a build on all nodes having specific label. Requires NodeLabel Parameter Plugin (optional)
  - **name** (*str*) – Name of the parameter to set (optional)
  - **node-label** (*str*) – Label of the nodes where build should be triggered
  - **ignore-offline-nodes** (*bool*) – Don't trigger build on offline nodes (default true)

  Factory
  - **factory** (*str*) **allonlinenodes** – Trigger a build on every online node. Requires NodeLabel Parameter Plugin (optional)

Examples:

Basic usage with yaml list of projects.

```
builders:
  - trigger-builds:
    - project:
        - "foo"
        - "bar"
        - "baz"
      current-parameters: true
```

Basic usage with passing svn revision through.

```
builders:
  - trigger-builds:
    - project: "build_started"
      predefined-parameters:
        FOO="bar"
      current-parameters: true
      svn-revision: true
      block: true
```

Basic usage with passing git revision through.

```
builders:
  - trigger-builds:
    - project: "build_started"
      predefined-parameters:
        FOO="bar"
      current-parameters: true
      node-label-name: NODE
      node-label: testnodes
      git-revision: true
      block: true
```

Example with all supported parameter factories.

```
builders:
  - trigger-builds:
    - project: "build_started"
      predefined-parameters:
        FOO="bar"
      current-parameters: true
      svn-revision: true
      parameter-factories:
        - factory: filebuild
          file-pattern: propfile*.txt
        - factory: binaryfile
          parameter-name: filename
          file-pattern: otherpropfile*.txt
        - factory: counterbuild
          from: 0
          to: 5
          step: 1
        - factory: allnodesforlabel
```

(continues on next page)

(continued from previous page)

```
        name: parametername
        node-label: labelname
        ignore-offline-nodes: false
      - factory: allonlinenodes
    block: true
```

trigger-remote()

Trigger build of job on remote Jenkins instance.

Requires the Jenkins Parameterized Remote Trigger Plugin

Please note that this plugin requires system configuration on the Jenkins Master that is unavailable from individual job views; specifically, one must add remote jenkins servers whose 'Display Name' field are what make up valid fields on the *remote-jenkins-name* attribute below.

Parameters

- **remote-jenkins-name** (*str*) – the remote Jenkins server (required)
- **job** (*str*) – the Jenkins project to trigger on the remote Jenkins server (required)
- **should-not-fail-build** (*bool*) – if true, remote job failure will not lead current job to fail (default false)
- **prevent-remote-build-queue** (*bool*) – if true, wait to trigger remote builds until no other builds (default false)
- **block** (*bool*) – whether to wait for the trigger jobs to finish or not (default true)
- **poll-interval** (*str*) – polling interval in seconds for checking statues of triggered remote job, only necessary if current job is configured to block (default 10)
- **connection-retry-limit** (*str*) – number of connection attempts to remote Jenkins server before giving up. (default 5)
- **enhanced-logging** (*bool*) – if this option is enabled, the console output of the remote job is also logged. (default false)
- **predefined-parameters** (*str*) – predefined parameters to send to the remote job when triggering it (optional)
- **property-file** (*str*) – file in workspace of current job containing additional parameters to be set on remote job (optional)

Example:

```
builders:
  - trigger-remote:
      remote-jenkins-name: "http://example.jenkinsmaster.lan"
      token: "BLAH"
      job: "build-things"
      should-fail-build: True
      prevent-remote-build-queue: True
      poll-interval: 5
      connection-retry-limit: 5
      block: true
      enhanced-logging: True
      property-file: '.props'
      predefined-parameters: |
        FOO="bar"
        herp="derp"
```

xcode()

This step allows you to execute an xcode build step.

Requires the Jenkins Xcode Plugin.

Parameters

- **developer-profile** (*str*) – the jenkins credential id for a ios developer profile. (optional)
- **clean-build** (*bool*) – if true will delete the build directories before invoking the build. (default false)
- **clean-test-reports** (*bool*) – UNKNOWN. (default false)
- **archive** (*bool*) – if true will generate an xcarchive of the specified scheme. A workspace and scheme are are also needed for archives. (default false)
- **configuration** (*str*) – This is the name of the configuration as defined in the Xcode project. (default 'Release')
- **configuration-directory** (*str*) – The value to use for CONFIGURATION_BUILD_DIR setting. (default '')
- **target** (*str*) – Leave empty for all targets. (default '')
- **sdk** (*str*) – Leave empty for default SDK. (default '')
- **symroot** (*str*) – Leave empty for default SYMROOT. (default '')
- **project-path** (*str*) – Relative path within the workspace that contains the xcode project file(s). (default '')
- **project-file** (*str*) – Only needed if there is more than one project file in the Xcode Project Directory. (default '')
- **build-arguments** (*str*) – Extra commandline arguments provided to the xcode builder. (default '')
- **schema** (*str*) – Only needed if you want to compile for a specific schema instead of a target. (default '')
- **workspace** (*str*) – Only needed if you want to compile a workspace instead of a project. (default '')
- **profile** (*str*) – The relative path to the mobileprovision to embed, leave blank for no embedded profile. (default '')
- **codesign-id** (*str*) – Override the code signing identity specified in the project. (default '')
- **allow-failing** (*bool*) – if true will prevent this build step from failing if xcodebuild exits with a non-zero return code. (default false)
- **version-technical** (*str*) – The value to use for CFBundleVersion. Leave blank to use project's technical number. (default '')
- **version-marketing** (*str*) – The value to use for CFBundleShortVersionString. Leave blank to use project's marketing number. (default '')
- **ipa-export-method** (*str*) – The export method of the .app to generate the .ipa file. Should be one in 'development', 'ad-hoc', 'enterprise', or 'app-store'. (default '')
- **ipa-version** (*str*) – A pattern for the ipa file name. You may use ${VERSION} and ${BUILD_DATE} (yyyy.MM.dd) in this string. (default '')
- **ipa-output** (*str*) – The output directory for the .ipa file, relative to the build directory. (default '')
- **compile-bitcode** (*bool*) – recompile from Bitcode when exporting the application to IPA. (default true)
- **upload-bitcode** (*bool*) – include Bitcode when exporting applications to IPA. (default true)
- **upload-symbols** (*bool*) – include symbols when exporting applications to IPA. (default true)
- **development-team-id** – The ID of the Apple development team to use to sign the IPA (default '')
- **keychain-name** (*str*) – The globally configured keychain to unlock for this build. (default '')
- **keychain-path** (*str*) – The path of the keychain to use to sign the IPA. (default '')
- **keychain-password** (*str*) – The password to use to unlock the keychain. (default '')
- **keychain-unlock** (*str*) – Unlocks the keychain during use. (default false)

- **bundle-id** (*str*) – The bundle identifier (App ID) for this provisioning profile (default '')
- **provisioning-profile-uuid** (*str*) – The UUID of the provisioning profile associated to this bundle identifier. (default '')

Example:

```
builders:
  - xcode
  - xcode:
      developer-profile: "849b07cd-ac61-4588-89c8-b6606ee84946"
      clean-build: true
      clean-test-reports: true
      configuration: Distribution
      target: TARGET
      sdk: iphonesimulator
      build-arguments: "test ONLY_ACTIVE_ARCH=NO -destination 'platform=iOS␣
↪Simulator,name=iPhone 6' -derivedDataPath ."
      schema: "UASDKInternal"
      workspace: "UA"
      profile: "PROFILE"
      codesign-id: "iPhone Distribution: MapMyFitness Inc."
      allow-failing: true
      version-technical: "TECHNICAL"
      version-marketing: "MARKETING"
      ipa-export-method: ad-hoc
      ipa-version: "${VERSION}"
      ipa-output: "/output"
      compile-bitcode: false
      upload-bitcode: false
      upload-symbols: false
      development-team-id: foo
      keychain-path: "/Users/jenkins/Library/Keychains/jenkins-uasdk-ios-pre_review"
      keychain-password: "testpass"
      keychain-unlock: true
      provisioning-profiles:
        - bundle-id: foo
          provisioning-profile-uuid: bar
        - bundle-id: foo2
          provisioning-profile-uuid: bar2
```

xunit()

Process tests results.

Requires the Jenkins xUnit Plugin.

Parameters

- **thresholdmode** (*str*) – Whether thresholds represents an absolute number of tests or a percentage. Either 'number' or 'percent'. (default 'number')
- **thresholds** (*list*) – Thresholds for both 'failed' and 'skipped' tests.
  - **threshold (*dict*)**
    Threshold values to set, where missing, xUnit should default to an internal value of 0. Each test threshold should contain the following:
    - **unstable** (*int*)
    - **unstablenew** (*int*)
    - **failure** (*int*)

– **failurenew** (*int*)

- **test-time-margin** (*int*) – Give the report time margin value in ms, before to fail if not new unless the option **requireupdate** is set for the configured framework. (default 3000)
- **types** (*list*) – Frameworks to configure, and options. Supports the following: `aunit`, `boosttest`, `checktype`, `cpptest`, `cppunit`, `ctest`, `dotnettest`, `embunit`, `fpcunit`, `gtest`, `junit`, `mstest`, `nunit`, `phpunit`, `tusar`, `unittest`, and `valgrind`.

  The 'custom' type is not supported.

  type (*dict*)

  each type can be configured using the following:

  – **pattern** (*str*): An Ant pattern to look for Junit result files, relative to the workspace root (default '')

  – **requireupdate** (*bool*): fail the build whenever fresh tests results have not been found (default true).

  – **deleteoutput** (*bool*): delete temporary JUnit files (default true).

  – **skip-if-no-test-files** (*bool*): Skip parsing this xUnit type report if there are no test reports files (default false).

  – **stoponerror** (*bool*): Fail the build whenever an error occur during a result file processing (default true).

Minimal Example:

```
builders:
  - xunit:
      types:
        - junit:
            pattern: "junit.xml"
```

Full Example:

```
builders:
  - xunit:
      thresholdmode: 'percent'
      thresholds:
          - failed:
                unstable: 0
                unstablenew: 0
                failure: 0
                failurenew: 0
          - skipped:
                unstable: 0
                unstablenew: 0
                failure: 0
                failurenew: 0
      test-time-margin: 5000
      types:
          - phpunit:
              pattern: "phpunit.log"
              requireupdate: true
              deleteoutput: true
              skip-if-no-test-files: false
              stoponerror: true
```

(continues on next page)

(continued from previous page)

```
        - cppunit:
            pattern: "cppunit.log"
            requireupdate: false
            deleteoutput: false
            skip-if-no-test-files: true
            stoponerror: false
        - gtest:
            pattern: "gtest.log"
```

Hipchat

Enable HipChat notifications of build execution.

Supports hipchat plugin versions < 1.9. Will automatically redirect to the publishers module for newer versions, but still recommended that you convert to the newer module.

Parameters

- **enabled** *(bool)*: general cut off switch. If not explicitly set to `true`, no hipchat parameters are written to XML. For Jenkins HipChat plugin of version prior to 0.1.5, also enables all build results to be reported in HipChat room. For later plugin versions, explicit notify-* setting is required (see below).
- **room** *(str)*: name of HipChat room to post messages to (default '')

  Deprecated since version 1.2.0: Please use 'rooms'.
- **rooms** *(list)*: list of HipChat rooms to post messages to (default empty)
- **start-notify** *(bool)*: post messages about build start event

  Deprecated since version 1.2.0: use notify-start parameter instead
- **notify-start** *(bool)*: post messages about build start event (default false)
- **notify-success** *(bool)*: post messages about successful build event (Jenkins HipChat plugin >= 0.1.5) (default false)
- **notify-aborted** *(bool)*: post messages about aborted build event (Jenkins HipChat plugin >= 0.1.5) (default false)
- **notify-not-built** *(bool)*: post messages about build set to NOT_BUILT status (Jenkins HipChat plugin >= 0.1.5). This status code is used in a multi-stage build (like maven2) where a problem in earlier stage prevented later stages from building. (default false)
- **notify-unstable** *(bool)*: post messages about unstable build event (Jenkins HipChat plugin >= 0.1.5) (default false)
- **notify-failure** *(bool)*: post messages about build failure event (Jenkins HipChat plugin >= 0.1.5) (default false)
- **notify-back-to-normal** *(bool)*: post messages about build being back to normal after being unstable or failed (Jenkins HipChat plugin >= 0.1.5) (default false)

Example:

```
hipchat:
  enabled: true
  rooms:
```

(continues on next page)

(continued from previous page)

```
      - My Room
      - Your Room
  notify-start: true
  notify-success: true
  notify-aborted: true
  notify-not-built: true
  notify-unstable: true
  notify-failure: true
  notify-back-to-normal: true
```

class hipchat_notif.**HipChat**(*registry*)

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 80

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

Metadata

The Metadata plugin module enables the ability to add metadata to the projects that can be exposed to job environment.

Requires the Jenkins Metadata Plugin.

Component: metadata

Macro
metadata

Entry Point
jenkins_jobs.metadata

Example:

```
metadata:
  - string:
      name: FOO
      value: bar
      expose-to-env: true
```

class metadata.**Metadata**(*registry*)

component_list_type = 'metadata'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'metadata'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 21

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

date()

A date metadata

Parameters

- **name** (*str*) – the name of the metadata
- **time** (*str*) – time value in millisec since 1970-01-01 00:00:00 UTC
- **timezone** (*str*) – time zone of the metadata
- **expose-to-env** (*bool*) – expose to environment (optional)

Example:

```
metadata:
  - date:
      name: FOO
      value: 1371708900268
      timezone: Australia/Melbourne
      expose-to-env: true
```

number()

A number metadata.

Parameters

- **name** (*str*) – the name of the metadata
- **value** (*str*) – the value of the metadata
- **expose-to-env** (*bool*) – expose to environment (optional)

Example:

```
metadata:
  - number:
      name: FOO
      value: 1
      expose-to-env: true
```

string()

A string metadata.

Parameters

- **name** (*str*) – the name of the metadata
- **value** (*str*) – the value of the metadata
- **expose-to-env** (*bool*) – expose to environment (optional)

Example:

```
metadata:
  - string:
      name: FOO
      value: bar
      expose-to-env: true
```

Notifications

The Notifications module allows you to configure Jenkins to notify other applications about various build phases. It requires the Jenkins notification plugin.

Component: notifications

Macro
notification

Entry Point
jenkins_jobs.notifications

`class notifications.Notifications(registry)`

`component_list_type = 'notifications'`

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

`component_type = 'notification'`

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

`gen_xml(xml_parent, data)`

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

`sequence = 22`

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

`http()`

Defines an HTTP notification endpoint.

Requires the Jenkins Notification Plugin.

Parameters

- **format** (*str*) – notification payload format, JSON (default) or XML
- **event** (*str*) – job events that trigger notifications: started, completed, finalized or all (default)
- **url** (*str*) – URL of the endpoint
- **timeout** (*int*) – Timeout in milliseconds for sending notification request (30 seconds by default)
- **retries** (*int*) – Nr of times to retry sending notification in case sending notification fails. (0 by default)

- **log** (*int*) – Number lines of log messages to send (0 by default). Use -1 for all (use with caution).

Example:

```
notifications:
  - http:
      url: http://example.com/jenkins_endpoint
      format: xml
      event: completed
      timeout: 40000
      log: -1
      retries: 5
```

Parameters

The Parameters module allows you to specify build parameters for a job.

Component: parameters

Macro
parameter

Entry Point
jenkins_jobs.parameters

Example:

```
job:
  name: test_job

  parameters:
    - string:
        name: FOO
        default: bar
        description: "A parameter named FOO, defaults to 'bar'."
```

class parameters.**Parameters**(*registry*)

component_list_type = 'parameters'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'parameter'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

`sequence = 21`

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

`active-choices()`

Active Choices Parameter

Requires the Jenkins Active Choices Plug-in.

Parameters

- **name** (*str*) – Name of the parameter (required).
- **description** (*str*) – Description of the parameter.
- **script** (*list*) – Use a Groovy script to define the parameter.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this** Groovy script in a sandbox with limited abilities (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **fallback-script** (*list*) – Use a Fallback script. If the script (specified above) fails, the fallback script will be used as a fallback.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this Groovy** script in a sandbox with limited abilities. (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **enable-filters** (*bool*) – If enabled a text box will appear next to this element and will permit the user to filter its entries. The list values never get re-evaluated (default False).
- **filter-starts-at** (*int*) – How many characters a user must enter before the filter is applied (default 1).
- **choice-type** (*str*) – type of the choices. (default 'single-select')
  Allowed Values
  - **single-select**
  - **multi-select**
  - **radio-buttons**
  - **checkboxes**

Minimal Example:

```
- job:
    name: active-choices-job
    parameters:
      - active-choices:
          name: lorem
```

Full Example:

```
- job:
    name: active-choices-job
    parameters:
      - active-choices:
          name: lorem
          description: ipsum
```

(continues on next page)

(continued from previous page)

```
        script:
            groovy: |-
                return [
                    'param1',
                    'param2'
                ]
            use-groovy-sandbox: false
            script-additional-classpath:
                - file:/jar-file-path
                - file:/jar-file-path2
        fallback-script:
            groovy: |-
                return [
                    'param3',
                    'param4'
                ]
            use-groovy-sandbox: false
            script-additional-classpath:
                - file:/jar-file-path
                - file:/jar-file-path2
        choice-type: multi-select
        enable-filters: true
        filter-starts-at: 1
```

`active-choices-reactive()`

Active Choices Reactive Parameter

Requires the Jenkins Active Choices Plug-in.

Parameters

- **name** (*str*) – Name of the parameter (required).
- **description** (*str*) – Description of the parameter.
- **script** (*list*) – Use a Groovy script to define the parameter.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this** Groovy script in a sandbox with limited abilities (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **fallback-script** (*list*) – Use a Fallback script. If the script (specified above) fails, the fallback script will be used as a fallback.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this Groovy** script in a sandbox with limited abilities. (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **enable-filters** (*bool*) – If enabled a text box will appear next to this element and will permit the user to filter its entries. The list values never get re-evaluated (default False).
- **filter-starts-at** (*int*) – How many characters a user must enter before the filter is applied (default 1).
- **choice-type** (*str*) – type of the choices. (default 'single-select')

Allowed Values

- **single-select**
- **multi-select**
- **radio-buttons**
- **checkboxes**

- **referenced-parameters** (*str*) – Comma separated list of other job parameters referenced in the uno-choice script

Minimal Example:

```
- job:
    name: active-choices-job
    parameters:
      - active-choices-reactive:
          name: foo
```

Full Example:

```
- job:
    name: active-choices-job
    parameters:
      - active-choices-reactive:
          name: lorem
          description: ipsum
          script:
              groovy: |-
                  return [
                      'param1',
                      'param2'
                  ]
              use-groovy-sandbox: false
              script-additional-classpath:
                  - file:/jar-file-path
                  - file:/jar-file-path2
          fallback-script:
              groovy: |-
                  return [
                      'param3',
                      'param4'
                  ]
              use-groovy-sandbox: false
              script-additional-classpath:
                  - file:/jar-file-path
                  - file:/jar-file-path2
          choice-type: multi-select
          enable-filters: true
          filter-starts-at: 1
          referenced-parameters: foo,bar
```

bool()

A boolean parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - bool:
      name: FOO
      default: false
      description: "A parameter named FOO, defaults to 'false'."
```

choice()

A single selection parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **choices** (*list*) – the available choices, first one is the default one.
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - choice:
      name: project
      choices:
        - nova
        - glance
      description: "On which project to run?"
```

copyartifact-build-selector()

Control via a build parameter, which build the copyartifact plugin should copy when it is configured to use 'build-param'.

Requires the Jenkins Copy Artifact plugin.

Parameters

- **name** (*str*) – name of the build parameter to store the selection in
- **description** (*str*) – a description of the parameter (optional)
- **which-build** (*str*) – which to provide as the default value in the UI. See `which-build` param of *copyartifact* from the builders module for the available values as well as options available that control additional behaviour for the selected value.

Example:

```
parameters:
  - copyartifact-build-selector:
      name: BUILD_SELECTOR
      which-build: workspace-latest
      description: 'Which build from upstream to copy artifacts from'
```

credentials()

A credentials selection parameter.

Requires the Jenkins Credentials Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **type** (*str*) – credential type (optional, default 'any')

  Allowed Values

  - **any** Any credential type (default)
  - **usernamepassword** Username with password
  - **sshkey** SSH Username with private key

- **secretfile** Secret file
- **secrettext** Secret text
- **certificate** Certificate

- **required** (*bool*) – whether this parameter is required (optional, default false)
- **default** (*str*) – default credentials ID (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - credentials:
      name: OS_CREDENTIALS
      type: usernamepassword
      default: "default-credentials-id"
      description: "Test credentials"
```

dynamic-choice()

Dynamic Choice Parameter

Requires the Jenkins Jenkins Dynamic Parameter Plug-in.

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **script** (*str*) – Groovy expression which generates the potential choices.
- **remote** (*bool*) – the script will be executed on the slave where the build is started (default false)
- **classpath** (*str*) – class path for script (optional)
- **read-only** (*bool*) – user can't modify parameter once populated (default false)

Example:

```
parameters:
  - dynamic-choice:
      name: OPTIONS
      description: "Available options"
      script: "['optionA', 'optionB']"
      remote: false
      read-only: false
```

dynamic-choice-scriptler()

Dynamic Choice Parameter (Scriptler)

Requires the Jenkins Jenkins Dynamic Parameter Plug-in.

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **script-id** (*str*) – Groovy script which generates the default value
- **parameters** (*list*) – parameters to corresponding script

  Parameter

  - **name** (*str*) Parameter name
  - **value** (*str*) Parameter value
- **remote** (*bool*) – the script will be executed on the slave where the build is started (default false)
- **read-only** (*bool*) – user can't modify parameter once populated (default false)

Example:

```
parameters:
  - dynamic-choice-scriptler:
      name: OPTIONS
      description: "Available options"
      script-id: "scriptid.groovy"
      parameters:
        - name: param1
          value: value1
        - name: param2
          value: value2
      remote: false
      read-only: false
```

dynamic-reference()

Active Choices Reactive Reference Parameter

Requires the Jenkins Active Choices Plug-in.

Parameters

- **name** (*str*) – Name of the parameter (required).
- **description** (*str*) – Description of the parameter.
- **script** (*list*) – Use a Groovy script to define the parameter.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this** Groovy script in a sandbox with limited abilities (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **fallback-script** (*list*) – Use a Fallback script. If the script (specified above) fails, the fallback script will be used as a fallback.
  Parameter
  - **groovy** (*str*) Groovy DSL Script
  - **use-groovy-sandbox (*bool*) To run this Groovy** script in a sandbox with limited abilities. (default True)
  - **script-additional-classpath (*list*) Additional** classpath entries accessible from the script.
- **omit-value-field** (*bool*) – By default Dynamic Reference Parameters always include a hidden input for the value. If your script creates an input HTML element, you can check this option and the value input field will be omitted (default False).
- **referenced-parameters** (*str*) – Comma separated list of other job parameters referenced in the uno-choice script. When any of the referenced parameters are updated, the Groovy script will re-evaluate the choice list using the updated values of referenced parameters.
- **choice-type** (*str*) – type of the choices. (default 'input-text-box')
  Allowed Values
  - **input-text-box**
  - **numbered-list**
  - **bullet-items-list**
  - **formatted-html**
  - **formatted-hidden-html**

Minimal Example:

```
- job:
```

(continues on next page)

(continued from previous page)

```
    name: dynamic-reference-job
    parameters:
      - dynamic-reference:
          name: lorem
```

Full Example:

```
- job:
    name: dynamic-reference-job
    folder: unochoice
    parameters:
      - dynamic-reference:
          name: lorem
          description: ipsum
          script:
              groovy: |-
                  return [
                      'param1',
                      'param2'
                  ]
              use-groovy-sandbox: false
              script-additional-classpath:
                  - file:/path
                  - file:/path2
          fallback-script:
              groovy: |-
                  return [
                      'param3',
                      'param4'
                  ]
              use-groovy-sandbox: false
              script-additional-classpath:
                  - file:/path
                  - file:/path2
          choice-type: numbered-list
          omit-value-field: True
          referenced-parameters: dolor
```

dynamic-string()

Dynamic Parameter

Requires the Jenkins Jenkins Dynamic Parameter Plug-in.

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **script** (*str*) – Groovy expression which generates the potential choices
- **remote** (*bool*) – the script will be executed on the slave where the build is started (default false)
- **classpath** (*str*) – class path for script (optional)
- **read-only** (*bool*) – user can't modify parameter once populated (default false)

Example:

```
parameters:
  - dynamic-string:
      name: FOO
      description: "A parameter named FOO, defaults to 'bar'."
      script: "bar"
      remote: false
      read-only: false
```

dynamic-string-scriptler()

Dynamic Parameter (Scriptler)

Requires the Jenkins Jenkins Dynamic Parameter Plug-in.

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **script-id** (*str*) – Groovy script which generates the default value
- **parameters** (*list*) – parameters to corresponding script
  - **Parameter**
    - **name** (*str*) Parameter name
    - **value** (*str*) Parameter value
- **remote** (*bool*) – the script will be executed on the slave where the build is started (default false)
- **read-only** (*bool*) – user can't modify parameter once populated (default false)

Example:

```
parameters:
  - dynamic-string-scriptler:
      name: FOO
      description: "A parameter named FOO, defaults to 'bar'."
      script-id: "scriptid.groovy"
      parameters:
        - name: param1
          value: value1
        - name: param2
          value: value2
      remote: false
      read-only: false
```

extended-choice()

Creates an extended choice parameter where values can be read from a file

Requires the Jenkins Extended Choice Parameter Plugin.

Parameters

- **name** (*str*) – name of the parameter
- **description** (*str*) – description of the parameter (optional, default '')
- **property-file** (*str*) – location of property file to read from (optional, default '')
- **property-key** (*str*) – key for the property-file (optional, default '')
- **quote-value** (*bool*) – whether to put quotes around the property when passing to Jenkins (optional, default false)
- **visible-items** (*str*) – number of items to show in the list (optional, default 5)
- **type** (*str*) – type of select, can be single-select, multi-select, multi-level-single-select, multi-level-multi-select, radio, checkbox, textbox, json (optional, default single-select)
- **value** (*str*) – comma separated list of values for the single select or multi-select box (optional, default '')

- **default-value** (*str*) – used to set the initial selection of the single-select or multi-select box (optional, default '')
- **value-description** (*str*) – comma separated list of value descriptions for the single select or multi-select box (optional, default '')
- **default-property-file** (*str*) – location of property file when default value needs to come from a property file (optional, default '')
- **default-property-key** (*str*) – key for the default property file (optional, default '')
- **description-property-file** (*str*) – location of property file when value description needs to come from a property file (optional, default '')
- **description-property-key** (*str*) – key for the value description property file (optional, default '')
- **multi-select-delimiter** (*str*) – value between selections when the parameter is a multi-select (optional, default ',')
- **groovy-script** (*str*) – the groovy script contents (optional, default ',')
- **groovy-script-file** (*str*) – location of groovy script file to generate parameters (optional, default '')
- **bindings** (*str*) – variable bindings for the groovy script (optional, default '')
- **classpath** (*str*) – the classpath for the groovy script (optional, default ',')
- **default-groovy-script** (*str*) – the default groovy script contents (optional, default '')
- **default-groovy-classpath** (*str*) – the default classpath for the groovy script (optional, default '')
- **description-groovy-script** (*str*) – location of groovy script when value description needs to come from a groovy script (optional, default '')
- **description-groovy-script-file** (*str*) – location of groovy script file when value description needs to come from a groovy script (optional, default '')
- **description-groovy-classpath** (*str*) – classpath for the value description groovy script (optional, default '')
- **javascript** (*str*) – the javascript script contents (optional, default '')
- **javascript-file** (*str*) – location of javasript script file to generate parameters (optional, default '')
- **save-json-parameter-to-file** (*bool*) – if json parameter should be saved to file (optional, default False)

Minimal Example:

```
parameters:
  - extended-choice:
      name: OPTIONS
      description: "Available options"
      type: 'PT_CHECKBOX'
      value: OptionA,OptionB,OptionC
```

Full Example:

```
parameters:
  - extended-choice:
      name: OPTIONS_VALUE
      description: "Available options"
      property-key: key
      quote-value: true
      type: multi-select
      value: "foo|bar|select"
      visible-items: 2
      multi-select-delimiter: '|'
```

(continues on next page)

(continued from previous page)

```
    default-value: foo
    default-property-key: fookey
- extended-choice:
    name: OPTIONS_FILE
    description: "Available options"
    property-file: /home/foo/property.prop
    property-key: key
    quote-value: true
    type: multi-select
    visible-items: 2
    multi-select-delimiter: '|'
    default-property-file: /home/property.prop
    default-property-key: fookey
- extended-choice:
    name: OPTIONS_CHECKBOX
    type: checkbox
    value: !join:
      - ','
      -
          - OptionA
          - OptionB
          - OptionC
    visible-items: 2
- extended-choice:
    name: MULTISELECTOPTIONS
    description: "Available options"
    property-key: key
    quote-value: true
    type: multi-select
    value: !join:
      - '|'
      -
          - foo
          - bar
          - select
    visible-items: 2
    multi-select-delimiter: '|'
    default-value: foo
- extended-choice:
    name: JSON
    type: json
    groovy-script: >-
        import net.sf.json.JSONObject;
        def jsonEditorOptions = JSONObject.fromObject(/{schema:
        {"type": "object", "title": "Name", "properties":
        {"name": {"type": "string", "propertyOrder" : 1}}}}/);
- extended-choice:
    name: MULTILEVELMULTISELECT
    type: multi-level-multi-select
    value: !join:
        - ','
        -
```

(continues on next page)

(continued from previous page)

```
            - foo
            - bar
            - baz
  - extended-choice:
      name: MULTILEVELSINGLESELECT
      type: multi-level-single-select
      value: foo
```

file()

A file parameter.

Parameters

- **name** (*str*) – the target location for the file upload
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - file:
      name: test.txt
      description: "Upload test.txt."
```

git-parameter()

This parameter allows you to select a git tag, branch or revision number as parameter in Parametrized builds.

Requires the Jenkins Git Parameter Plugin.

Parameters

- **name** (*str*) – Name of the parameter
- **description** (*str*) – Description of the parameter (default '')
- **type** (*str*) – The type of the list of parameters (default 'PT_TAG')

  Allowed Values

  - **PT_TAG list of all commit tags in repository -** returns Tag Name
  - **PT_BRANCH list of all branches in repository -** returns Branch Name
  - **PT_BRANCH_TAG list of all commit tags and all** branches in repository - returns Tag Name or Branch Name
  - **PT_REVISION list of all revision sha1 in repository** followed by its author and date - returns Tag SHA1
  - **PT_PULL_REQUEST**
- **branch** (*str*) – Name of branch to look in. Used only if listing revisions. (default '')
- **branchFilter** (*str*) – Regex used to filter displayed branches. If blank, the filter will default to ".*". Remote branches will be listed with the remote name first. E.g., "origin/master" (default '.*')
- **tagFilter** (*str*) – Regex used to filter displayed branches. If blank, the filter will default to ".*". Remote branches will be listed with the remote name first. E.g., "origin/master" (default '*')
- **sortMode** (*str*) – Mode of sorting. (default 'NONE')

  Allowed Values

  - **NONE**
  - **DESCENDING**
  - **ASCENDING**
  - **ASCENDING_SMART**
  - **DESCENDING_SMART**

- **defaultValue** (*str*) – This value is returned when list is empty. (default '')
- **selectedValue** (*str*) – Which value is selected, after loaded parameters. If you choose 'default', but default value is not present on the list, nothing is selected. (default 'NONE')

  Allowed Values
  - **NONE**
  - **TOP**
  - **DEFAULT**
- **useRepository** (*str*) – If in the task is defined multiple repositories parameter specifies which the repository is taken into account. If the parameter is not defined, is taken first defined repository. The parameter is a regular expression which is compared with a URL repository. (default '')
- **quickFilterEnabled** (*bool*) – When this option is enabled will show a text field. Parameter is filtered on the fly. (default false)

Minimal Example:

```
parameters:
  - git-parameter:
      name: Foo
```

Full Example:

```
parameters:
  - git-parameter:
      name: Foo
      description: Lorem ipsum dolor sit amet.
      type: PT_BRANCH_TAG
      branch: baz
      tagFilter: bam
      branchFilter: boo
      sortMode: ASCENDING
      defaultValue: bor
      selectedValue: TOP
      useRepository: buh
      quickFilterEnabled: true
```

hidden()

Allows you to use parameters hidden from the build with parameter page.

Requires the Jenkins Hidden Parameter Plugin.

Parameters
- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - hidden:
      name: FOO
      default: bar
      description: A parameter named FOO, defaults to 'bar'
```

label()

A node label parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **all-nodes** (*bool*) – to run job on all nodes matching label in parallel (default: false)
- **matching-label** (*str*) – to run all nodes matching label 'success', 'unstable' or 'allCases' (optional)
- **node-eligibility** (*str*) – all nodes, ignore temporary nodes or ignore temporary offline nodes (optional, default all nodes)

Example:

```
parameters:
  - label:
      name: EXAMPLE LABEL 1
      description: "EXAMPLE LABEL DESCRIPTION 1"
      matching-label: "success"
      node-eligibility: "all"
```

matrix-combinations()

Matrix combinations parameter

Requires the Jenkins Matrix Combinations Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **filter** (*str*) – Groovy expression to use filter the combination by default (optional)

Example:

```
parameters:
  - matrix-combinations:
      name: FOO
      description: "Select matrix combinations"
      filter: "platform == foo"
```

maven-metadata()

This parameter allows the resolution of maven artifact versions by contacting the repository and reading the maven-metadata.xml.

Requires the Jenkins Maven Metadata Plugin.

Parameters

- **name** (*str*) – Name of the parameter
- **description** (*str*) – Description of the parameter (optional)
- **repository-base-url** (*str*) – URL from where you retrieve your artifacts (default '')
- **repository-username** (*str*) – Repository's username if authentication is required. (default '')
- **repository-password** (*str*) – Repository's password if authentication is required. (default '')
- **artifact-group-id** (*str*) – Unique project identifier (default '')
- **artifact-id** (*str*) – Name of the artifact without version (default '')
- **packaging** (*str*) – Artifact packaging option. Could be something such as jar, zip, pom.... (default '')
- **versions-filter** (*str*) – Specify a regular expression which will be used to filter the versions which are actually displayed when triggering a new build. (default '')
- **default-value** (*str*) – For features such as SVN polling a default value is required. If job will only be started manually, this field is not necessary. (default '')

- **maximum-versions-to-display** (*str*) – The maximum number of versions to display in the drop-down. Any non-number value as well as 0 or negative values will default to all. (default 10)
- **sorting-order** (*str*) – ascending or descending (default descending)

Example:

```
parameters:
  - maven-metadata:
      name: 'maven metadata param'
      repository-base-url: 'http://nexus.example.com'
      repository-username: 'username'
      repository-password: 'password'
      artifact-group-id: 'com.example'
      artifact-id: 'example'
      packaging: 'jar'
      versions-filter: '[0-9]+'
      default-value: 'FIRST'
      maximum-versions-to-display: "5"
      sorting-order: "Ascending"
```

node()

Defines a list of nodes where this job could potentially be executed on. Restrict where this project can be run, If your using a node or label parameter to run your job on a particular node, you should not use the option "Restrict where this project can be run" in the job configuration - it will not have any effect to the selection of your node anymore!

Parameters

- **name** (*str*) – the name of the parameter
- **description** (*str*) – a description of the parameter (optional)
- **default-slaves** (*list*) – The nodes used when job gets triggered by anything else other than manually
- **allowed-slaves** (*list*) – The nodes available for selection when job gets triggered manually. Empty means 'All'.
- **ignore-offline-nodes** (*bool*) – Ignore nodes not online or not having executors (default false)
- **allowed-multiselect** (*bool*) – Allow multi node selection for concurrent builds - this option only makes sense (and must be selected!) in case the job is configured with: "Execute concurrent builds if necessary". With this configuration the build will be executed on all the selected nodes in parallel. (default false)

Example:

```
parameters:
  - node:
      name: SLAVE_NAME
      description: "Select slave"
      allowed-slaves:
        - slave001
        - slave002
        - slave003
      ignore-offline-nodes: true
      allowed-multiselect: true
```

parameter-separator()

A parameter separator.

Parameters

- **name** (*str*) – name of the separator (default "", the plugin will assign a randomly generated UUID if not specified)
- **separator-style** (*str*) – the style of the separator. Uses CSS. (default "")
- **section-header-text** (*str*) – the section header text of the separator (default "")
- **section-header-style** (*str*) – the section header style (CSS) of the separator. Uses CSS. (default "")

Example:

```
parameters:
  - parameter-separator:
      name: lorem
      separator-style: FOO
      section-header: bar
      section-header-style: font-weight:bold;z-index:10000
```

password()

A password parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - password:
      name: FOO
      default: 1HSC0Ts6E161FysGf+e1xasgsHkgleLh09JUTYnipPvw=
      description: "A parameter named FOO."
```

persistent-bool()

A persistent boolean parameter.

Requires the Jenkins Persistent Parameter Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **successfulOnly** (*bool*) – if true, then the value of the parameter gets persisted only between successful builds (optional, default: false)

Example:

```
parameters:
  - persistent-bool:
      name: FOO
      default: false
      description: "A persistent parameter named FOO, defaults to 'false'."
      successfulOnly: false
```

persistent-choice()

A persistent single selection parameter.

Requires the Jenkins Persistent Parameter Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **choices** (*list*) – the available choices, first one is the default one.

- **description** (*str*) – a description of the parameter (optional)
- **successfulOnly** (*bool*) – if true, then the value of the parameter gets persisted only between successful builds (optional, default: false)

Example:

```
parameters:
  - persistent-choice:
      name: project
      choices:
        - nova
        - glance
      description: "On which project to run?"
      successfulOnly: false
```

persistent-string()

A persistent string parameter.

Requires the Jenkins Persistent Parameter Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **trim** (*bool*) – strip whitespaces from the begnning and end of the string (optional, default: false)
- **successfulOnly** (*bool*) – if true, then the value of the parameter gets persisted only between successful builds (optional, default: false)

Example:

```
parameters:
  - persistent-string:
      name: FOO
      default: bar
      description: "A parameter named FOO, defaults to 'bar'."
      trim: false
      successfulOnly: false
```

persistent-text()

A persistent text parameter.

Requires the Jenkins Persistent Parameter Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **trim** (*bool*) – strip whitespaces from the begnning and end of the string (optional, default: false)
- **successfulOnly** (*bool*) – if true, then the value of the parameter gets persisted only between successful builds (optional, default: false)

Example:

```
parameters:
  - persistent-text:
      name: FOO
      default: bar
```

(continues on next page)

(continued from previous page)

```
    description: "A persistent parameter named FOO, defaults to 'bar'."
    successful0nly: false
```

promoted build()

A promoted build parameter.

Requires the Jenkins Promoted Builds Plugin.

Parameters

- **name** (*str*) – the name of the parameter (required)
- **project-name** (*str*) – the job from which the user can pick runs (required)
- **promotion-name** (*str*) – promotion process to choose from (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - promoted:
      name: FOO
      project-name: "foo-build"
      promotion-name: "passed-promotion"
      description: "Select a foo-build for promotion"
```

random-string()

This parameter generates a random string and passes it to the build, preventing Jenkins from combining queued builds.

Requires the Jenkins Random String Parameter Plugin.

Parameters

- **name** (*str*) – Name of the parameter
- **description** (*str*) – Description of the parameter (default '')
- **failed-validation-message** (*str*) – Failure message to display for invalid input (default '')

Example:

```
parameters:
  - random-string:
      name: job-string
      description: "A random string passed to the job"
      failed-validation-message: "Your input string is invalid"
```

run()

A run parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **project-name** (*str*) – the name of job from which the user can pick runs
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - run:
      name: FOO
      project-name: "foo-build"
      description: "Select a foo-build for promotion"
```

string()

A string parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **trim** (*bool*) – strip whitespaces from the begnning and end of the string (optional, default: false)

Example:

```
parameters:
  - string:
      name: FOO
      default: bar
      description: "A parameter named FOO, defaults to 'bar'."
      trim: true
```

svn-tags()

A svn tag parameter

Requires the Jenkins Parameterized Trigger Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **url** (*str*) – the url to list tags from
- **credentials-id** (*str*) – Credentials ID to use for authentication (default '')
- **filter** (*str*) – the regular expression to filter tags (default '')
- **default** (*str*) – the default value of the parameter (default '')
- **description** (*str*) – a description of the parameter (default '')
- **max-tags** (*int*) – the number of tags to display (default '100')
- **sort-newest-first** (*bool*) – sort tags from newest to oldest (default true)
- **sort-z-to-a** (*bool*) – sort tags in reverse alphabetical order (default false)

Example:

```
parameters:
  - svn-tags:
      name: BRANCH_NAME
      default: release
      description: A parameter named BRANCH_NAME default is release
      url: http://svn.example.org/repo
      filter: [A-za-z0-9]*
```

text()

A text parameter.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)

Example:

```
parameters:
  - text:
      name: FOO
      default: bar
      description: "A parameter named FOO, defaults to 'bar'."
```

validating-string()

A validating string parameter

Requires the Jenkins Validating String Plugin.

Parameters

- **name** (*str*) – the name of the parameter
- **default** (*str*) – the default value of the parameter (optional)
- **description** (*str*) – a description of the parameter (optional)
- **regex** (*str*) – a regular expression to validate the string
- **msg** (*str*) – a message to display upon failed validation

Example:

```
parameters:
  - validating-string:
      name: FOO
      default: bar
      description: "A parameter named FOO, defaults to 'bar'."
      regex: [A-Za-z]*
      msg: Your entered value failed validation
```

Properties

The Properties module supplies a wide range of options that are implemented as Jenkins job properties.

Component: properties

Macro
property

Entry Point
jenkins_jobs.properties

Example:

```
job:
  name: test_job

  properties:
    - github:
        url: https://github.com/openstack-infra/jenkins-job-builder/
```

class properties.**Properties**(*registry*)

component_list_type = 'properties'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'property'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML

output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 20

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

authenticated-build()

Specifies an authorization matrix where only authenticated users may trigger a build.

Deprecated since version 0.1.0.: Please use *authorization*.

Example:

```
properties:
  - authenticated-build
```

authorization()

Specifies an authorization matrix In 3.0 version of plugin was added support for explicitly assigning permissions to groups or users with a given name to prevent confusion when names match either.

For *matrix-auth >= 3.0*

Parameters

prefix:<name> (*list*) –

- ***prefix***
  - **GROUP**
  - **USER**
- <*name*> is the name of the group or user, containing

For *matrix-auth < 3.0*

Parameters

<name> (*list*) – <*name*> is the name of the group or user, containing the list of rights to grant.

<name> rights

- **credentials-create**
- **credentials-delete**
- **credentials-manage-domains**
- **credentials-update**
- **credentials-view**
- **job-build**
- **job-cancel**
- **job-configure**
- **job-delete**
- **job-discover**
- **job-extended-read**
- **job-move**
- **job-read**
- **job-status**
- **job-workspace**
- **ownership-jobs**
- **run-delete**
- **run-replay**
- **run-update**
- **scm-tag**

Example:

```
properties:
    - authorization:
        USER:admin:
            - credentials-create
            - credentials-delete
            - credentials-manage-domains
            - credentials-update
            - credentials-view
            - job-build
            - job-cancel
            - job-configure
            - job-delete
            - job-discover
            - job-move
            - job-read
            - job-status
            - job-workspace
            - ownership-jobs
            - run-delete
            - run-replay
            - run-update
            - scm-tag
        GROUP:anonymous:
            - job-read
            - job-extended-read
        authenticated:
            - job-read
            - job-discover
            - job-extended-read
```

batch-tasks()

Batch tasks can be tasks for events like releases, integration, archiving, etc. In this way, anyone in the project team can execute them in a way that leaves a record.

A batch task consists of a shell script and a name. When you execute a build, the shell script gets run on the workspace, just like a build. Batch tasks and builds "lock" the workspace, so when one of those activities is in progress, all the others will block in the queue.

Requires the Jenkins Batch Task Plugin.

Parameters

batch-tasks (*list*) – Batch tasks.

Tasks

- **name** (*str*) Task name.
- **script** (*str*) Task script.

Example:

```
properties:
  - batch-tasks:
      - name: release
        script: mvn -B release:prepare release:perform
      - name: say hello
        script: echo "Hello world"
```

branch-api()

Enforces a minimum time between builds based on the desired maximum rate.

Requires the Jenkins Branch API Plugin.

Parameters

- **number-of-builds** (*int*) – The maximum number of builds allowed within the specified time period. (default 1)
- **time-period** (*str*) – The time period within which the maximum number of builds will be enforced. (default 'hour')
  valid values
  second minute hour, **day**, **week**, **month**, **year**
- **skip-rate-limit** (*bool*) – Permit user triggered builds to skip the rate limit (default false)

Minimal Example:

```
properties:
  - branch-api
```

Full example:

```
properties:
  - branch-api:
      time-period: day
      number-of-builds: 5
      skip-rate-limit: true
```

build-blocker()

This plugin keeps the actual job in the queue if at least one name of currently running jobs is matching with one of the given regular expressions.

Requires the Jenkins Build Blocker Plugin.

Parameters

- **use-build-blocker** (*bool*) – Enable or disable build blocker (default true)
- **blocking-jobs** (*list*) – One regular expression per line to select blocking jobs by their names (required)
- **block-level** (*str*) – block build globally ('GLOBAL') or per node ('NODE') (default 'GLOBAL')
- **queue-scanning** (*str*) – scan build queue for all builds ('ALL') or only buildable builds ('BUILDABLE') (default 'DISABLED')

Example:

Minimal Example:

```
properties:
  - build-blocker:
      blocking-jobs:
        - ".*-deploy"
```

Full Example:

```
properties:
  - build-blocker:
      use-build-blocker: true
      blocking-jobs:
        - ".*-deploy"
        - "^maintenance.*"
```

(continues on next page)

(continued from previous page)

```
      block-level: 'NODE'
      queue-scanning: 'BUILDABLE'
```

build-discarder()

Parameters

- **days-to-keep** (*int*) – Number of days to keep builds for (default -1)
- **num-to-keep** (*int*) – Number of builds to keep (default -1)
- **artifact-days-to-keep** (*int*) – Number of days to keep builds with artifacts (default -1)
- **artifact-num-to-keep** (*int*) – Number of builds with artifacts to keep (default -1)

Example:

```
properties:
    - build-discarder:
        days-to-keep: 42
        num-to-keep: 43
        artifact-days-to-keep: 44
        artifact-num-to-keep: 45
```

```
properties:
    - build-discarder
```

build-failure-analyzer()

Controls failure cause analysis for builds. Requires the Jenkins Build Failure Analyzer Plugin.

Example:

```
properties:
  - build-failure-analyzer
```

```
properties:
  - build-failure-analyzer:
      disabled: true
```

builds-chain-fingerprinter()

Builds chain fingerprinter.

Requires the Jenkins Builds chain fingerprinter Plugin.

Parameters

- **per-builds-chain** (*bool*) – enable builds hierarchy fingerprinting (default false)
- **per-job-chain** (*bool*) – enable jobs hierarchy fingerprinting (default false)

Example:

```
properties:
  - builds-chain-fingerprinter:
      per-builds-chain: true
      per-job-chain: true
```

cachet-gating()

The Cachet Gating Plugin provides a gating mechanism based on the availability of resources.

Requires the Jenkins: Cachet Gate Plugin.

Parameters

- **required-resources** (*bool*) – Confirm availability of listed resources before building. Requires the list of resources to also be defined. (default true)
- **resources** (*list*) – which resources to gate

Example:

```
properties:
      - cachet-gating:
          required-resources: true
          resources:
              - beaker
              - brew
```

copyartifact()

Specify a list of projects that have access to copy the artifacts of this project.

Requires the Jenkins Copy Artifact plugin.

Parameters

projects (*str*) – comma separated list of projects that can copy artifacts of this project. Wild card character '*' is available.

Example:

```
properties:
  - copyartifact:
      projects: foo*
```

delivery-pipeline()

Requires the Jenkins Delivery Pipeline Plugin.

Parameters

- **stage** (*str*) – Name of the stage for this job (default '')
- **task** (*str*) – Name of the task for this job (default '')
- **description** (*str*) – task description template for this job (default '')

Minimal Example:

```
properties:
  - delivery-pipeline
```

Full Example:

```
properties:
  - delivery-pipeline:
      stage: Stage
      task: Task
      description: Task-Description
```

disable-resume()

Do not allow the pipeline to resume if the master restarts

Requires the Jenkins Pipeline Job Plugin.

Example:

```
properties:
  - disable-resume
```

disk-usage()

Enables the Disk Usage Plugin.

Requires the Jenkins Disk Usage Plugin.

Example:

```
properties:
  - disk-usage
```

docker-container()

Requires the Jenkins: Docker Plugin.

Parameters

- **docker-registry-url** (*str*) – URL of the Docker registry. (default '')
- **credentials-id** (*str*) – Credentials Id for the Docker registey. (default '')
- **commit-on-success** (*bool*) – When a job completes, the docker slave instance is committed with repository based on the job name and build number as tag. (default false)
- **additional-tag** (*str*) – Additional tag to apply to the docker slave instance when committing it. (default '')
- **push-on-success** (*bool*) – Also push the resulting image when committing the docker slave instance. (default false)
- **clean-local-images** (*bool*) – Clean images from the local daemon after building. (default true)

Minimal Example:

```
properties:
  - docker-container
```

Full Example:

```
properties:
  - docker-container:
      commit-on-success: true
      additional-tag: latest
      push-on-success: true
      clean-local-images: true
      docker-registry-url: https://index.docker.io/v1/
      credentials-id: 71e4f29c-162b-40d0-85d9-3ddfba2911a0
```

gitbucket()

Integrate GitBucket to Jenkins.

Requires the Jenkins GitBucket Plugin.

Parameters

- **url** (*str*) – GitBucket URL to issue (required)
- **link-enabled** (*bool*) – Enable hyperlink to issue (default false)

Minimal Example:

```
properties:
  - gitbucket:
      url: https://foo.com
```

Full Example:

```
properties:
  - gitbucket:
```

(continues on next page)

(continued from previous page)

```
      url: https://foo.com
      link-enabled: true
```

github()

Sets the GitHub URL for the project.

Parameters

- **url** (*str*) – the GitHub URL (required)
- **display-name** (*str*) – This value will be used as context name for commit status if status builder or status publisher is defined for this project. (>= 1.14.1) (default '')

Minimal Example:

```
properties:
  - github:
      url: https://github.com/openstack-infra/jenkins-job-builder/
```

Full Example:

```
properties:
  - github:
      url: https://github.com/openstack-infra/jenkins-job-builder/
      display-name: foo
```

gitlab()

Sets the GitLab connection for the project. Configured via Jenkins Global Configuration.

Requires the Jenkins GitLab Plugin.

Parameters

connection (*str*) – the GitLab connection name (required)

Example:

```
properties:
  - gitlab:
      connection: gitlab-connection
```

gitlab-logo()

Configures the GitLab-Logo Plugin.

Requires the Jenkins GitLab Logo Plugin.

Parameters

repository-name (*str*) – the GitLab repository name (required)

Example:

```
properties:
  - gitlab-logo:
      repository-name: gitlab-repository-name
```

gogs()

Sets the Gogs webhook properties for the project.

Requires the Jenkins Gogs Plugin.

Parameters

- **secret** (*str*) – webhook secret (default '')
- **branch-filter** (*str*) – filter which needs to match to trigger a job (default '')

Minimal Example:

```
properties:
  - gogs
```

Full Example:

```
properties:
  - gogs:
      branch-filter: 'master'
      secret: 'yoursecret'
```

groovy-label()

This plugin allows you to use Groovy script to restrict where this project can be run.

Requires the Jenkins Groovy Label Assignment Plugin.

Return value from Groovy script is treated as Label Expression. It is treated as followings:

- A non-string value will be converted to a string using toString()
- When null or blank string is returned, node restriction does not take effect (or is not overwritten).
- When exception occurred or Label Expression is not parsed correctly, builds are canceled.

Parameters

- **script** (*str*) – Groovy script (default '')
- **sandbox** (*bool*) – Use Groovy Sandbox. (default false) If checked, run this Groovy script in a sandbox with limited abilities. If unchecked, and you are not a Jenkins administrator, you will need to wait for an administrator to approve the script
- **classpath** (*list*) – Additional classpath entries accessible from the script, each of which should be an absolute local path or URL to a JAR file, according to "The file URI Scheme" (optional)

Minimal Example:

```
properties:
    - groovy-label
```

Full Example:

```
properties:
    - groovy-label:
        script: "$LABEL_NAME"
        sandbox: true
        classpath:
            - "file:/minimal/absolute/path/to/file.jar"
            - "file:///traditional/absolute/path/to/file.jar"
            - "http://example.org/path/to/file.jar"
            - "https://example.org/path/to/file.jar"
```

heavy-job()

This plugin allows you to define "weight" on each job, and making each job consume that many executors

Requires the Jenkins Heavy Job Plugin.

Parameters

weight (*int*) – Specify the total number of executors that this job should occupy (default 1)

Example:

```
properties:
  - heavy-job:
      weight: 2
```

inject()

Allows you to inject environment variables into the build.

Requires the Jenkins EnvInject Plugin.

Parameters

- **properties-file** (*str*) – file to read with properties (optional)
- **properties-content** (*str*) – key=value properties (optional)
- **script-file** (*str*) – file with script to run (optional)
- **script-content** (*str*) – script to run (optional)
- **groovy-content** (*str*) – groovy script to run (optional)
- **groovy-sandbox** (*bool*) – run groovy script in sandbox (default false)
- **load-from-master** (*bool*) – load files from master (default false)
- **enabled** (*bool*) – injection enabled (default true)
- **keep-system-variables** (*bool*) – keep system variables (default true)
- **keep-build-variables** (*bool*) – keep build variable (default true)
- **override-build-parameters** (*bool*) – override build parameters (default false)

Example:

```
properties:
  - inject:
      properties-content: |
        FOO=bar
        BAZ=foobar
```

least-load()

Enables the Least Load Plugin.

Requires the Jenkins Least Load Plugin.

Parameters

disabled (*bool*) – whether or not leastload is disabled (default true)

Example:

```
properties:
  - least-load:
      disabled: False
```

lockable-resources()

Requires the Jenkins Lockable Resources Plugin.

Parameters

- **resources** (*str*) – List of required resources, space separated. (required, mutual exclusive with label)
- **label** (*str*) – If you have created a pool of resources, i.e. a label, you can take it into use here. The build will select the resource(s) from the pool that includes all resources sharing the given label. (required, mutual exclusive with resources)
- **var-name** (*str*) – Name for the Jenkins variable to store the reserved resources in. Leave empty to disable. (default '')
- **number** (*int*) – Number of resources to request, empty value or 0 means all. This is useful, if you have a pool of similar resources, from which you want one or more to be reserved. (default 0)

- **match-script** (*str*) – Groovy script to reserve resource based on its properties. Leave empty to disable. (default None)
- **groovy-sandbox** (*bool*) – Execute the provided match-script in Groovy sandbox. Leave empty to disable. (default False)

Example:

```
---
properties:
  - lockable-resources:
      resources: "the-resource"
```

```
---
properties:
  - lockable-resources:
      label: "pool-1"
```

```
---
properties:
  - lockable-resources:
      resources: "the-resource"
      var-name: "RESOURCE_NAME"
      number: 10
```

```
---
properties:
  - lockable-resources:
      match-script: "resourceName == MY_VAR"
      groovy-sandbox: true
```

naginator-opt-out()

Lets you opt-out so no rebuild option for Naginator is added.

Requires the Jenkins Naginator Plugin.

Parameters

opt-out (*bool*) – disables the rebuild option if True (default False).

Example:

```
properties:
 - naginator-opt-out:
    opt-out: true
```

office-365-connector()

Used to send actionable messages to MS Outlook or Teams

Requires the Jenkins: :jenkins-plugins:` Office-365-Connector Plugin <Office-365-Connector>`.

Parameters

webhooks (*list*) – List of webhooks (required)

- **url** (srt): URL generated in the Office 365 Connectors page (required)
- **name (str): Allows to provide name fo the connection. Name is not** mandatory but helps managing when there are many connection assigned to the build (optional, default '')
- **start-notification (bool): If the notification should be sent on** start of build (optional, default False)

- **notify-success (bool): If the notification should be sent on** succeeded build (optional, default True)
- **notify-aborted (bool): If the notification should be sent on** aborted build (optional, default False)
- **notify-not-built (bool): If the notification should be sent on** not built build (optional, default False)
- **notify-unstable (bool): If the notification should be sent on** unstable build (optional, default True)
- **notify-failure (bool): If the notification should be sent on** failed build (optional, default True)
- **notify-back-to-normal (bool): If the notification should be sent on** back to normal build (optional, default True)
- **notify-repeated-failure (bool): If the notification should be sent on** repeated failures (optional, default False)
- **timeout (int): connection timeout (in milliseconds) for TCP and HTTP** (optional, default 30000)
- **macros** (list): List of macros
  - **template** (str) **value** (str)
- **fact-definitions** (list): List of fact definitions
  - **name** (str) **template** (str)

Example:

```
properties:
  - office-365-connector:
      webhooks:
        - url: http://outlook.office.com/webhook
          name: full
          start-notification: true
          notify-success: false
          notify-aborted: true
          notify-not-built: true
          notify-unstable: false
          notify-failure: false
          notify-back-to-normal: false
          notify-repeated-failure: true
          timeout: 30001
          macros:
            - template: macro1
              value: macro1_value
            - template: macro2
              value: macro2_value
          fact-definitions:
            - name: fd1
              template: fd1_value
            - name: fd2
              template: fd2_value
```

ownership()

Plugin provides explicit ownership for jobs and slave nodes.

Requires the Jenkins Ownership Plugin.

Parameters

- **enabled** (*bool*) – whether ownership enabled (default : true)
- **owner** (*str*) – the owner of job

- **co-owners** (*list*) – list of job co-owners

Example:

```
properties:
 - ownership:
    owner: foo
    co-owners:
     - bar
     - moo
```

priority-sorter()

Allows simple ordering of builds, using a configurable job priority.

Requires the Jenkins Priority Sorter Plugin.

Parameters

priority (*int*) – Priority of the job. Higher value means higher priority, with 3 as the default priority. (required)

Example:

```
properties:
  - priority-sorter:
      priority: 3
```

promoted-build()

Marks a build for promotion. A promotion process with an identical name must be created via the web interface in the job in order for the job promotion to persist. Promotion processes themselves cannot be configured by jenkins-jobs due to the separate storage of plugin configuration files.

Requires the Jenkins Promoted Builds Plugin.

Parameters

names (*list*) – the promoted build names (optional)

Example:

```
properties:
  - promoted-build:
      names:
        - "Release to QA"
        - "Jane Must Approve"
```

rebuild()

This plug-in allows the user to rebuild a parameterized build without entering the parameters again.It will also allow the user to edit the parameters before rebuilding.

Requires the Jenkins Rebuild Plugin.

Parameters

- **auto-rebuild** (*bool*) – Rebuild without asking for parameters (default false)
- **rebuild-disabled** (*bool*) – Disable rebuilding for this job (default false)

Minimal Example:

```
properties:
    - rebuild
```

Full Example:

```
properties:
    - rebuild:
        auto-rebuild: true
        rebuild-disabled: true
```

resource-gating()

Jenkins Gating enables requiring external resources to be available before build starts.

Requires the Jenkins: Jenkins Gating.

Parameters

resources (*list*) – Resource identifiers to be up before building

Example:

```
properties:
    - resource-gating:
        resources:
            - external/ci.example.com
            - foo/bar/baz
```

sidebar()

Allows you to add links in the sidebar. Requires the Jenkins Sidebar-Link Plugin.

Parameters

- **url** (*str*) – url to link to (optional)
- **text** (*str*) – text for the link (optional)
- **icon** (*str*) – path to icon (optional)

Example:

```
properties:
   - sidebar:
       url: https://jenkins.debian.net/userContent/about.html
       text: About jenkins.debian.net
       icon: /userContent/images/debian-swirl-24x24.png
   - sidebar:
       url: https://jenkins.debian.net/view/reproducible
       text: reproducible builds jobs
       icon: /userContent/images/debian-jenkins-24x24.png
```

slack()

Requires the Jenkins Slack Plugin.

When using Slack Plugin version < 2.0, Slack Plugin itself requires a publisher aswell as properties please note that you have to add the publisher to your job configuration aswell. When using Slack Plugin version >= 2.0, you should only configure the publisher.

Parameters

- **notify-start** (*bool*) – Send notification when the job starts (default false)
- **notify-success** (*bool*) – Send notification on success. (default false)
- **notify-aborted** (*bool*) – Send notification when job is aborted. (default false)
- **notify-not-built** (*bool*) – Send notification when job set to NOT_BUILT status. (default false)
- **notify-unstable** (*bool*) – Send notification when job becomes unstable. (default false)
- **notify-failure** (*bool*) – Send notification when job fails. (default false)
- **notify-back-to-normal** (*bool*) – Send notification when job is succeeding again after being unstable or failed. (default false)

- **'notify-repeated-failure'** (*bool*) – Send notification when job is still failing after last failure. (default false)
- **include-test-summary** (*bool*) – Include the test summary. (default False)
- **include-custom-message** (*bool*) – Include a custom message into the notification. (default false)
- **custom-message** (*str*) – Custom message to be included. (default '')
- **room** (*str*) – A comma separated list of rooms / channels to send the notifications to. (default '')

Example:

```
properties:
    - slack:
        room: dummy, dummy2
        notify-start: true
        notify-success: true
```

slave-prerequisites()

This plugin allows you to check prerequisites on slave before a job can run a build on it

Requires the Jenkins Slave Prerequisites Plugin.

Parameters

- **script** (*str*) – A script to be executed on slave node. If returning non 0 status, the node will be vetoed from hosting the build. (required)
- **interpreter** (*str*) – Command line interpreter to be used for executing the prerequisite script - either *shell* for Unix shell or *cmd* for Windows batch script. (default shell)

Example:

```
properties:
  - slave-prerequisites:
      script: |
        #!/bin/bash
        AVAILABLE=$(df -BG --output=avail / | tail -1)
        test ${AVAILABLE%G} -ge 10
```

```
properties:
  - slave-prerequisites:
      interpreter: cmd
      script: |
        @echo off
        if exist C:\Jenkins\workspace.lock exit /b 1
```

slave-utilization()

This plugin allows you to specify the percentage of a slave's capacity a job wants to use.

Requires the Jenkins Slave Utilization Plugin.

Parameters

- **slave-percentage** (*int*) – Specify the percentage of a slave's execution slots that this job should occupy (default 0)
- **single-instance-per-slave** (*bool*) – Control whether concurrent instances of this job will be permitted to run in parallel on a single slave (default false)

Example:

```
properties:
  - slave-utilization:
```

(continues on next page)

(continued from previous page)

```
      slave-percentage: 40
      single-instance-per-slave: false
```

speed-durability()

This setting allows users to change the default durability mode for running Pipelines.

Parameters

hint (*str*) – speed durability hint to be used, can be performance-optimized, survivable-non-atomic, max-survivability

Example:

```
properties:
  - speed-durability:
      hint: performance-optimized
```

throttle()

Throttles the number of builds for this job.

Requires the Jenkins Throttle Concurrent Builds Plugin.

Parameters

- **option** (*str*) – throttle *project* (throttle the project alone) or *category* (throttle the project as part of one or more categories)
- **max-per-node** (*int*) – max concurrent builds per node (default 0)
- **max-total** (*int*) – max concurrent builds (default 0)
- **enabled** (*bool*) – whether throttling is enabled (default true)
- **categories** (*list*) – multiproject throttle categories
- **matrix-builds** (*bool*) – throttle matrix master builds (default true)
- **matrix-configs** (*bool*) – throttle matrix config builds (default false)
- **parameters-limit** (*str*) – prevent jobs with matching parameters from running concurrently (default false)
- **parameters-check-list** (*list*) – Comma-separated list of parameters to use when comparing jobs (optional)

Example:

```
properties:
  - throttle:
      max-per-node: 2
      max-total: 4
      categories:
        - cat1
        - cat2
      option: category
      matrix-builds: false
      matrix-configs: true
```

zeromq-event()

This is a Jenkins plugin that will publish Jenkins Job run events (start, complete, finish) to a ZMQ PUB socket.

Requires the Jenkins ZMQ Event Publisher.

Example:

```
properties:
  - zeromq-event
```

Publishers

Publishers define actions that the Jenkins job should perform after the build is complete.

Component: publishers

Macro
publisher

Entry Point
jenkins_jobs.publishers

class publishers.**Publishers**(*registry*)

component_list_type = 'publishers'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'publisher'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 70

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

aggregate-flow-tests()

Aggregate downstream test results in a Build Flow job.

Requires the Jenkins Build Flow Test Aggregator Plugin.

Parameters
show-test-results-trend (*bool*) – whether to show test results trend graph (default true)

Example:

```
publishers:
  - aggregate-flow-tests:
      show-test-results-trend: false
```

aggregate-tests()

Aggregate downstream test results

Parameters
include-failed-builds (*bool*) – whether to include failed builds (default false)

Example:

```
publishers:
  - aggregate-tests:
      include-failed-builds: true
```

allure()

Publish Allure report for the build. Requires the Jenkins Allure Plugin.

Parameters

- **jdk** (*str*) – String identifier for a JDK installation in Jenkins
- **commandline** (*str*) – String identifier for a Allure-commandline tool installation
- **report-build-policy** (*str*) – String identifier for a report build policy enum. Possible values: 'ALWAYS', 'UNSTABLE', 'UNSUCCESSFUL'. (By default is 'ALWAYS')
- **include-properties** (*bool*) – Flag to include specified properties
- **results-paths** (*list*) – List of results directories
- **properties** (*list*) – List of key:value property pairs

Minimal Example:

```
publishers:
  - allure:
      results-paths:
       - path: 'build/allure-results'
```

Full Example:

```
publishers:
  - allure:
      results-paths:
       - path: 'build/allure-results1'
       - path: 'build/allure-results2'
      properties:
       - key: 'allure.link.issue.pattern'
         value: 'http://test.tms/{}'
      jdk: openjdk1.8
      commandline: allure2
      report-build-policy: UNSTABLE
      include-properties: true
```

archive()

Archive build artifacts

Parameters

- **artifacts** (*str*) – path specifier for artifacts to archive
- **excludes** (*str*) – path specifier for artifacts to exclude (optional)
- **latest-only** (*bool*) – only keep the artifacts from the latest successful build
- **allow-empty** (*bool*) – pass the build if no artifacts are found (default false)
- **follow-symlinks** (*bool*) – follow symbolic links during archiving (default false)
- **only-if-success** (*bool*) – archive artifacts only if build is successful (default false)
- **fingerprint** (*bool*) – fingerprint all archived artifacts (default false)
- **default-excludes** (*bool*) – This option allows you to enable or disable the default Ant exclusions. (default true)
- **case-sensitive** (*bool*) – Treat include and exclude patterns as case sensitive. (default true)

Example:

```
publishers:
  - archive:
      artifacts: '*.tar.gz'
      allow-empty: 'true'
      fingerprint: true
      default-excludes: false
```

artifact-deployer()

This plugin makes it possible to copy artifacts to remote locations.

Requires the Jenkins ArtifactDeployer Plugin.

Parameters

- **entries** (*list*) –
  - **entries**
    - **files** (*str*) - files to deploy
    - **basedir** (*str*) - the dir from files are deployed
    - **excludes** (*str*) - the mask to exclude files
    - **remote** (*str*) - a remote output directory
    - **flatten** (*bool*) - ignore the source directory structure (default false)
    - **delete-remote** (*bool*) - clean-up remote directory before deployment (default false)
    - **delete-remote-artifacts** (*bool*) - delete remote artifacts when the build is deleted (default false)
    - **fail-no-files** (*bool*) - fail build if there are no files (default false)
    - **groovy-script** (*str*) - execute a Groovy script before a build is deleted
- **deploy-if-fail** (*bool*) – Deploy if the build is failed (default false)

Example:

```
publishers:
  - artifact-deployer:
      entries:
        - files: '*.tar.gz'
          basedir: '/opt/data'
          excludes: '*tmp*'
          remote: '/home/test/'
          flatten: true
          delete-remote: true
          delete-remote-artifacts: true
          fail-no-files: true
          groovy-script: 'print 123'
      deploy-if-fail: true
```

artifactory()

Uses/requires the Artifactory plugin to deploy artifacts to Artifactory Server.

Requires the Jenkins Artifactory Plugin.

Parameters

- **url** (*str*) – Artifactory server url (default '')
- **name** (*str*) – Artifactory user with permissions use for connected to the selected Artifactory Server (default '')
- **release-repo-key** (*str*) – Release repository name (default '')
- **snapshot-repo-key** (*str*) – Snapshots repository name (default '')
- **publish-build-info** (*bool*) – Push build metadata with artifacts (default false)
- **discard-old-builds** (*bool*) – Remove older build info from Artifactory (default false)
- **discard-build-artifacts** (*bool*) – Remove older build artifacts from Artifactory (default false)
- **even-if-unstable** (*bool*) – Deploy artifacts even when the build is unstable (default false)

- **run-checks** (*bool*) – Run automatic license scanning check after the build is complete (default false)
- **include-publish-artifacts** (*bool*) – Include the build's published module artifacts in the license violation checks if they are also used as dependencies for other modules in this build (default false)
- **pass-identified-downstream** (*bool*) – When true, a build parameter named ARTIFACTORY_BUILD_ROOT with a value of ${JOB_NAME}-${BUILD_NUMBER} will be sent to downstream builds (default false)
- **license-auto-discovery** (*bool*) – Tells Artifactory not to try and automatically analyze and tag the build's dependencies with license information upon deployment (default true)
- **enable-issue-tracker-integration** (*bool*) – When the Jenkins JIRA plugin is enabled, synchronize information about JIRA issues to Artifactory and attach issue information to build artifacts (default false)
- **aggregate-build-issues** (*bool*) – When the Jenkins JIRA plugin is enabled, include all issues from previous builds up to the latest build status defined in "Aggregation Build Status" (default false)
- **allow-promotion-of-non-staged-builds** (*bool*) – The build promotion operation will be available to all successful builds instead of only staged ones (default false)
- **filter-excluded-artifacts-from-build** (*bool*) – Add the excluded files to the excludedArtifacts list and remove them from the artifacts list in the build info (default false)
- **scopes** (*str*) – A list of dependency scopes/configurations to run license violation checks on. If left empty all dependencies from all scopes will be checked (default '')
- **violation-recipients** (*str*) – Recipients that need to be notified of license violations in the build info (default '')
- **matrix-params** (*list*) – Semicolon-separated list of properties to attach to all deployed artifacts in addition to the default ones: build.name, build.number, and vcs.revision (default [])
- **black-duck-app-name** (*str*) – The existing Black Duck Code Center application name (default '')
- **black-duck-app-version** (*str*) – The existing Black Duck Code Center application version (default '')
- **black-duck-report-recipients** (*str*) – Recipients that will be emailed a report after the automatic Black Duck Code Center compliance checks finished (default '')
- **black-duck-scopes** (*str*) – A list of dependency scopes/configurations to run Black Duck Code Center compliance checks on. If left empty all dependencies from all scopes will be checked (default '')
- **black-duck-run-checks** (*bool*) – Automatic Black Duck Code Center compliance checks will occur after the build completes (default false)
- **black-duck-include-published-artifacts** (*bool*) – Include the build's published module artifacts in the license violation checks if they are also used as dependencies for other modules in this build (default false)
- **auto-create-missing-component-requests** (*bool*) – Auto create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory (default true)
- **auto-discard-stale-component-requests** (*bool*) – Auto discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory (default true)
- **deploy-artifacts** (*bool*) – Push artifacts to the Artifactory Server. Use deployment-include-patterns and deployment-exclude-patterns to filter deploy artifacts. (default true)
- **deployment-include-patterns** (*list*) – New line or comma separated mappings of build artifacts to published artifacts. Supports Ant-style wildcards mapping to target

directories. E.g.: /.zip=>dir (default [])
- **deployment-exclude-patterns** (*list*) – New line or comma separated patterns for excluding artifacts from deployment to Artifactory (default [])
- **env-vars-include** (*bool*) – Include all environment variables accessible by the build process. Jenkins-specific env variables are always included. Use env-vars-include-patterns and env-vars-exclude-patterns to filter variables to publish, (default false)
- **env-vars-include-patterns** (*list*) – Comma or space-separated list of environment variables that will be included as part of the published build info. Environment variables may contain the * and the ? wildcards (default [])
- **env-vars-exclude-patterns** (*list*) – Comma or space-separated list of environment variables that will be excluded from the published build info (default [])

Example:

```
publishers:
  - artifactory:
      url: http://artifactory.example.net/artifactory
      name: 'test'
      release-repo-key: libs-release-local
      snapshot-repo-key: libs-snapshot-local
```

```
publishers:
  - artifactory:
      url: http://artifactory.example.net/artifactory
      name: 'test'
      release-repo-key: libs-release-local
      snapshot-repo-key: libs-snapshot-local
      publish-build-info: true
      discard-old-builds: true
      discard-build-artifacts: true
      even-if-unstable: true
      run-checks: true
      include-publish-artifacts: true
      pass-identified-downstream: true
      license-auto-discovery: true
      aggregate-build-issues: true
      allow-promotion-of-non-staged-builds: true
      filter-excluded-artifacts-from-build: true
      violation-recipients: myfake@email.com
      matrix-params: []
      black-duck-app-name: myapp
      black-duck-app-version: '1.0'
      black-duck-report-recipients: myfake@email.com
      black-duck-scopes: []
      black-duck-run-checks: true
      black-duck-include-published-artifacts: true
      auto-create-missing-component-requests: false
      auto-discard-stale-component-requests: false
      deploy-artifacts: true
      deployment-include-patterns: []
      deployment-exclude-patterns: []
      env-vars-include: true
      env-vars-include-patterns: []
      env-vars-exclude-patterns: []
```

blame-upstream()

Notify upstream committers when build fails

Requires the Jenkins Blame Upstream Committers Plugin.

Example:

```
publishers:
  - blame-upstream
```

build-publisher()

This plugin allows records from one Jenkins to be published on another Jenkins.

Requires the Jenkins Build Publisher Plugin.

Parameters

- **publish-unstable-builds** (*bool*) – publish unstable builds (default true)
- **publish-failed-builds** (*bool*) – publish failed builds (default true)
- **days-to-keep** (*int*) – days to keep when publishing results (optional)
- **num-to-keep** (*int*) – number of jobs to keep in the published results (optional)

Minimal Example:

```
publishers:
  - build-publisher
```

Full Example:

```
publishers:
  - build-publisher:
      publish-unstable-builds: false
      publish-failed-builds: false
      days-to-keep: -1
      num-to-keep: 100
```

campfire()

Send build notifications to Campfire rooms. Requires the Jenkins Campfire Plugin.

Campfire notifications global default values must be configured for the Jenkins instance. Default values will be used if no specific values are specified for each job, so all config params are optional.

Parameters

- **subdomain** (*str*) – override the default campfire subdomain
- **token** (*str*) – override the default API token
- **ssl** (*bool*) – override the default 'use SSL'
- **room** (*str*) – override the default room name

Example:

```
publishers:
    - campfire:
        subdomain: 'sub'
        ssl: true
        token: 'TOKEN'
        room: 'room'
```

checkstyle()

Publish trend reports with Checkstyle.

Requires the Jenkins Checkstyle Plugin (https://github.com/jenkinsci/checkstyle-plugin).

The checkstyle component accepts a dictionary with the following values:

Parameters

- **pattern** (*str*) – Report filename pattern (default '')
- **can-run-on-failed** (*bool*) – Also runs for failed builds, instead of just stable or unstable builds (default false)
- **should-detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings (default false)
- **healthy** (*int*) – Sunny threshold (default '')
- **unhealthy** (*int*) – Stormy threshold (default '')
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high') (default 'low')
- **thresholds** (*dict*) – Mark build as failed or unstable if the number of errors exceeds a threshold. (optional)

  thresholds

  – **unstable (*dict*)**

  unstable

  * **total-all** (*int*)
  * **total-high** (*int*)
  * **total-normal** (*int*)
  * **total-low** (*int*)
  * **new-all** (*int*)
  * **new-high** (*int*)
  * **new-normal** (*int*)
  * **new-low** (*int*)

  – **failed (*dict*)**

  failed

  * **total-all** (*int*)
  * **total-high** (*int*)
  * **total-normal** (*int*)
  * **total-low** (*int*)
  * **new-all** (*int*)
  * **new-high** (*int*)
  * **new-normal** (*int*)
  * **new-low** (*int*)
- **default-encoding** (*str*) – Encoding for parsing or showing files (default '')
- **do-not-resolve-relative-paths** (*bool*) – (default false)
- **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
- **use-previous-build-as-reference** (*bool*) – determines whether to always use the previous build as the reference build (default false)
- **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **use-delta-values** (*bool*) – If set then the number of new warnings is calculated by subtracting the total number of warnings of the current build from the reference build. (default false)

Example:

```yaml
publishers:
 - checkstyle:
    pattern: '**/checkstyle-result.xml'
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    thresholds:
        unstable:
            total-high: 10
        failed:
            total-high: 1
```

Full example:

```yaml
publishers:
 - checkstyle:
    pattern: '**/checkstyle-result.xml'
    can-run-on-failed: true
    should-detect-modules: true
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    thresholds:
        unstable:
            total-all: 90
            total-high: 80
            total-normal: 70
            total-low: 60
            new-all: 50
            new-high: 40
            new-normal: 30
            new-low: 20
        failed:
            total-all: 91
            total-high: 81
            total-normal: 71
            total-low: 61
            new-all: 51
            new-high: 41
            new-normal: 31
            new-low: 21
    default-encoding: 'utf-8'
    do-not-resolve-relative-paths: true
    dont-compute-new: false
    use-stable-build-as-reference: true
    use-delta-values: true
```

chuck-norris()

Displays a picture of Chuck Norris (instead of Jenkins the butler) and a random Chuck Norris 'The Programmer' fact on each build page.

Requires the Jenkins ChuckNorris Plugin.

Example:

```
publishers:
  - chuck-norris
```

cifs()

Upload files via CIFS. Requires the Jenkins Publish over CIFS Plugin.

Parameters

- **site** (*str*) – name of the cifs site/share (required)
- **target** (*str*) – destination directory (required)
- **target-is-date-format** (*bool*) – whether target is a date format. If true, raw text should be quoted (default false)
- **clean-remote** (*bool*) – should the remote directory be deleted before transferring files (default false)
- **source** (*str*) – source path specifier (required)
- **excludes** (*str*) – excluded file pattern (default '')
- **remove-prefix** (*str*) – prefix to remove from uploaded file paths (default '')
- **fail-on-error** (*bool*) – fail the build if an error occurs (default false).
- **flatten** (*bool*) – only create files on the server, don't create directories (default false).
- **verbose** (*bool*) – adds lots of detail useful for debug to the console but generally should be left off (default false)
- **retries** (*int*) – the number of times to retry this server in the event of failure (optional)
- **retry-delay** (*int*) – the time to wait, in milliseconds, before attempting another transfer (default 10000)

Minimal Example:

```
publishers:
  - cifs:
      site: 'cifs.share'
      target: 'dest/dir'
      source: 'base/source/dir/**'
```

Full Example:

```
publishers:
  - cifs:
      site: 'cifs.share'
      target: "'dest/dir/'yyyyMMddHHmmss"
      target-is-date-format: true
      clean-remote: true
      source: 'base/source/dir/**'
      excludes: '**/*.excludedfiletype'
      remove-prefix: 'base/source/dir'
      fail-on-error: true
      flatten: true
      verbose: true
      retries: 99
      retry-delay: 12345
```

cigame()

This plugin introduces a game where users get points for improving the builds. Requires the Jenkins Continuous Integration Game plugin (https://github.com/jenkinsci/ci-game-plugin).

Example:

```
publishers:
  - cigame
```

claim-build()

Claim build failures Requires the Jenkins Claim Plugin.

Example:

```
publishers:
  - claim-build
```

clamav()

Check files with ClamAV, an open source antivirus engine. Requires the Jenkins ClamAV Plugin.

Parameters

- **includes** (*str*) – Comma separated list of files that should be scanned. Must be set for ClamAV to check for artifacts. (default '')
- **excludes** (*str*) – Comma separated list of files that should be ignored (default '')

Full Example:

```
publishers:
  - clamav:
      includes: '*.zip'
      excludes: 'foo.zip'
```

Minimal Example:

```
publishers:
  - clamav
```

clone-workspace()

Archive the workspace from builds of one project and reuse them as the SCM source for another project. Requires the Jenkins Clone Workspace SCM Plugin.

Parameters

- **workspace-glob** (*str*) – Files to include in cloned workspace (default '')
- **workspace-exclude-glob** (*str*) – Files to exclude from cloned workspace
- **criteria** (*str*) – Criteria for build to be archived. Can be 'any', 'not failed', or 'successful'. (default 'any')
- **archive-method** (*str*) – Choose the method to use for archiving the workspace. Can be 'tar' or 'zip'. (default 'tar')
- **override-default-excludes** (*bool*) – Override default ant excludes. (default false)

Minimal example:

```
publishers:
  - clone-workspace
```

Full example:

```
publishers:
  - clone-workspace:
      criteria: "Any"
      archive-method: "TAR"
      override-default-excludes: false
      workspace-glob: "**/*.zip"
      workspace-exclude-glob: "**/*.tgz"
```

`cloudfoundry()`

Pushes a project to Cloud Foundry or a CF-based platform (e.g. Stackato) at the end of a build. Requires the Jenkins Cloud Foundry Plugin.

Parameters

- **target** (*str*) – The API endpoint of the platform you want to push to. This is the URL you use to access the platform, possibly with ".api" added. (required)
- **organization** (*str*) – An org is a development account that an individual or multiple collaborators can own and use (required)
- **space** (*str*) – Provide users with access to a shared location for application development, deployment, and maintenance (required)
- **credentials-id** (*str*) – credentials-id of the user (required)
- **self-signed** (*bool*) – Allow self-signed SSL certificates from the target (default false)
- **reset-app** (*bool*) – Delete app before pushing app's configurations (default false)
- **plugin-timeout** (*int*) – The time in seconds before the Cloud Foundry plugin stops fetching logs and marks the build a failure (default 120)
- **create-services** (*list*) – Create services automatically (default '')
  - **create-services**
    - **name** ('str') – Service name (default '')
    - **type** ('str') – Service type (default '')
    - **plan** ('str') – Service plan (default '')
    - **reset-service ('bool') – Delete the service before creating** the new one (default false)
- **value** (*str*) – Select to read configuration from manifest file or to enter configuration in Jenkins (default 'manifestFile')
- **manifest-file** (*str*) – Path to manifest file (default 'manifest.yml')
- **app-name** (*str*) – The application's name. Default to Jenkins build name. (default '')
- **memory** (*int*) – The application's memory usage in MB (default 512)
- **host-name** (*str*) – The hostname of the URI to access your application. Default to app-name (default '')
- **instances** (*int*) – Number of instances of your application on creation (default 1)
- **manifest-timeout** (*int*) – The time in seconds before the health-manager gives up on starting the application (default 60)
- **no-route** (*bool*) – No URI path will be created to access the application (default false)
- **app-path** (*str*) – Path to application (default '')
- **build-pack** – If your application requires a custom buildpack, you can use this to specify its URL or name (default '')
- **stack** (*str*) – If your application requires a custom stack, you can use this to specify its name. (default '')
- **command** (*str*) – Set a custom start command for your application (default '')
- **domain** (*str*) – The domain of the URI to access your application (default '')
- **environment-variables** (*list*) – Inject environment variables
  - **environment-variables**
    - **key** ('str') – Environment variable key (default '')
    - **value** ('str') – Environment variable value (default '')
- **services-names** (*list*) – Name of service instances
  - **services-names**
    - **name** ('str') – Name of the service instance (default '')

Minimal example:

```
publishers:
  - cloudfoundry:
      target: https://api.stackato-rkw2.local
```

(continues on next page)

(continued from previous page)

```
      organization: AS
      space: SimpleSpace
      credentials-id: j89jk213
```

Full example:

```
publishers:
  - cloudfoundry:
      target: https://api.stackato-rkw2.local
      organization: AS
      space: SimpleSpace
      credentials-id: 123
      self-signed: true
      reset-app: true
      timeout: 240
      create-services:
        - name: foo-name
          type: foo-type
          plan: plan1
          reset-service: true
        - name: bar-name
          type: bar-type
          plan: plan2
          reset-service: false
      value: jenkinsConfig
      manifest-file: manifest.yml
      app-name: cloudfoundry
      memory: 1024
      host-name: cloudfoundry
      instances: 5
      manifest-timeout: 120
      no-route: true
      app-path: foo
      build-pack: custom-buildpack
      stack: custom-stack
      command: start
      domain: cloudfoundry.domain
      environment-variables:
        - key: key
          value: value
        - key: key2
          value: value2
      services-names:
        - name: service-name
        - name: service-name2
```

`cloudformation()`

Create cloudformation stacks before running a build and optionally delete them at the end. Requires the Jenkins AWS Cloudformation Plugin.

Parameters

- **create-stacks** (*list*) – List of stacks to create

  create-stacks attributes

  - **arg str name** - The name of the stack (Required)

- **arg str description** - Description of the stack (Optional)
- **arg str recipe** - The cloudformation recipe file (Required)
- **arg list parameters** - A list of key/value pairs, will be joined together into a comma separated string (Optional)
- **arg int timeout** - Number of seconds to wait before giving up creating a stack (default 0)
- **arg str access-key** - The Amazon API Access Key (Required)
- **arg str secret-key** - The Amazon API Secret Key (Required)
- **arg int sleep** - Number of seconds to wait before continuing to the next step (default 0)
- **arg array region** - The region to run cloudformation in. (Required)

  region values

  * **us-east-1**
  * **us-west-1**
  * **us-west-2**
  * **eu-central-1**
  * **eu-west-1**
  * **ap-southeast-1**
  * **ap-southeast-2**
  * **ap-northeast-1**
  * **sa-east-1**

- `delete-stacks` (*list*) – List of stacks to delete

  delete-stacks attributes

  - **arg list name** - The names of the stacks to delete (Required)
  - **arg str access-key** - The Amazon API Access Key (Required)
  - **arg str secret-key** - The Amazon API Secret Key (Required)
  - **arg bool prefix** - If selected the tear down process will look for the stack that Starts with the stack name with the oldest creation date and will delete it. (default false)
  - **arg array region** - The region to run cloudformation in. (Required)

    region values

    * **us-east-1**
    * **us-west-1**
    * **us-west-2**
    * **eu-central-1**
    * **eu-west-1**
    * **ap-southeast-1**
    * **ap-southeast-2**
    * **ap-northeast-1**
    * **sa-east-1**

Example:

```yaml
publishers:
  - cloudformation:
      create-stacks:
        - name: "foo"
          description: "Build the foo stack"
          recipe: "foo.json"
          parameters:
            - "Key1=foo"
            - "Key2=fuu"
          timeout: 3600
          access-key: "$AWS_ACCESS_KEY"
          secret-key: "$AWS_SECRET_KEY"
          region: us-west-2
          sleep: 5
        - name: "bar"
          description: "Build the bar stack"
          recipe: "bar.json"
          parameters:
            - "Key1=bar"
            - "Key2=baa"
          timeout: 3600
          access-key: "$AWS_ACCESS_KEY"
          secret-key: "$AWS_SECRET_KEY"
          region: us-west-1
      delete-stacks:
        - name: "foo"
          prefix: true
          region: us-west-2
          access-key: "$AWS_ACCESS_KEY"
          secret-key: "$AWS_SECRET_KEY"
        - name: "bar"
          region: us-west-1
          access-key: "$AWS_ACCESS_KEY"
          secret-key: "$AWS_SECRET_KEY"
```

cloverphp()

Capture code coverage reports from PHPUnit Requires the Jenkins Clover PHP Plugin.

Your job definition should pass to PHPUnit the –coverage-clover option pointing to a file in the workspace (ex: clover-coverage.xml). The filename has to be filled in the *xml-location* field.

Parameters

- **xml-location** (*str*) – Path to the coverage XML file generated by PHPUnit using –coverage-clover. Relative to workspace. (required)
- **html** (*dict*) – When existent, whether the plugin should generate a HTML report. Note that PHPUnit already provide a HTML report via its –cover-html option which can be set in your builder (optional):
  - **dir (str): Directory where HTML report will be generated relative** to workspace. (required in *html* dict).
  - **archive** (bool): Whether to archive HTML reports (default true).
- **metric-targets** (*list*) – List of metric targets to reach, must be one of **healthy**, **unhealthy** and **failing**. Each metric target can takes two parameters:
  - **method** Target for method coverage
  - **statement** Target for statements coverage

  Whenever a metric target is not filled in, the Jenkins plugin can fill in defaults for you

(as of v0.3.3 of the plugin the healthy target will have method: 70 and statement: 80 if both are left empty). Jenkins Job Builder will mimic that feature to ensure clean configuration diff.

Minimal example:

```
# Test for the defaults, only xml-location is required
publishers:
 - cloverphp:
       xml-location: 'build/clover.xml'
```

Full example:

```
# Exercise all options with non defaults values
publishers:
 - cloverphp:
       xml-location: 'build/clover.xml'
       html:
         dir: 'html'
         archive: false
       metric-targets:
        - healthy:
           method: 80
           statement: 90
        - unhealthy:
           method: 40
           statement: 50
        - failing:
           method: 10
           statement: 20
```

cobertura()

Generate a cobertura coverage report. Requires the Jenkins Cobertura Coverage Plugin.

Parameters

- **report-file** (*str*) – This is a file name pattern that can be used to locate the cobertura xml report files (optional)
- **only-stable** (*bool*) – Include only stable builds (default false)
- **fail-no-reports** (*bool*) – fail builds if no coverage reports are found (default false)
- **fail-unhealthy** (*bool*) – Unhealthy projects will be failed (default false)
- **fail-unstable** (*bool*) – Unstable projects will be failed (default false)
- **health-auto-update** (*bool*) – Auto update threshold for health on successful build (default false)
- **stability-auto-update** (*bool*) – Auto update threshold for stability on successful build (default false)
- **zoom-coverage-chart** (*bool*) – Zoom the coverage chart and crop area below the minimum and above the maximum coverage of the past reports (default false)
- **source-encoding** (*str*) – Override the source encoding (default ASCII)
- **targets** (*dict*) –

  targets

  (packages, files, classes, method, line, conditional)
  - **healthy** (*int*): Healthy threshold (default 0)
  - **unhealthy** (*int*): Unhealthy threshold (default 0)
  - **failing** (*int*): Failing threshold (default 0)

Example:

```
publishers:
  - cobertura:
      report-file: "/reports/cobertura/coverage.xml"
      only-stable: "true"
      fail-no-reports: "true"
      fail-unhealthy: "true"
      fail-unstable: "true"
      health-auto-update: "true"
      stability-auto-update: "true"
      zoom-coverage-chart: "true"
      source-encoding: "Big5"
      targets:
        - files:
            healthy: 10
            unhealthy: 20
            failing: 30
        - method:
            healthy: 50
            unhealthy: 40
            failing: 30
```

codecover()

This plugin allows you to capture code coverage report from CodeCover. Jenkins will generate the trend report of coverage. Requires the Jenkins CodeCover Plugin.

Parameters

- **include** (*str*) – Specify the path to the CodeCover HTML report file, relative to the workspace root (default '')
- **min-statement** (*int*) – Minimum statement threshold (default 0)
- **max-statement** (*int*) – Maximum statement threshold (default 90)
- **min-branch** (*int*) – Minimum branch threshold (default 0)
- **max-branch** (*int*) – Maximum branch threshold (default 80)
- **min-loop** (*int*) – Minimum loop threshold (default 0)
- **max-loop** (*int*) – Maximum loop threshold (default 50)
- **min-condition** (*int*) – Minimum condition threshold (default 0)
- **max-condition** (*int*) – Maximum conditon threshold (default 50)

Minimal Example:

```
publishers:
  - codecover
```

Full Example:

```
publishers:
  - codecover:
      include: ./path/report.html
      min-statement: 1
      max-statement: 100
      min-branch: 2
      max-branch: 90
      min-loop: 3
      max-loop: 80
      min-condition: 4
      max-condition: 70
```

conditional-publisher()

Conditionally execute some post-build steps. Requires the Jenkins Flexible Publish Plugin.

A Flexible Publish list of Conditional Actions is created in Jenkins.

Parameters

- **condition-kind** (*str*) – Condition kind that must be verified before the action is executed. Valid values and their additional attributes are described in the *conditions* table.
- **condition-aggregation** (*bool*) – If true Matrix Aggregation will be enabled. (default false)
- **condition-aggregation-kind** (*str*) – Condition Aggregation kind that must be verified before the action is executed. Valid values and their additional attributes are described in the *conditions* table.
- **on-evaluation-failure** (*str*) – What should be the outcome of the build if the evaluation of the condition fails. Possible values are *fail*, *mark-unstable*, *run-and-mark-unstable*, *run* and *dont-run*. Default is *fail*.
- **action** (*list*) – Action to run if the condition is verified. Item can be any publisher known by Jenkins Job Builder and supported by the Flexible Publish Plugin.

Condition kind	Description
always	Condition is always verified
never	Condition is never verified
boolean-expression	Run the action if the expression expands to a representation of true **condition-expression** Expression to expand
current-status	Run the action if the current build status is within the configured range **condition-worst** Accepted values are SUCCESS, UNSTABLE, FAILURE, NOT_BUILD, ABORTED **condition-best** Accepted values are SUCCESS, UNSTABLE, FAILURE, NOT_BUILD, ABORTED
shell	Run the action if the shell command succeeds **condition-command** Shell command to execute
windows-shell	Similar to shell, except that commands will be executed by cmd, under Windows **condition-command** Command to execute
regexp	Run the action if a regular expression matches **condition-expression** Regular Expression **condition-searchtext** Text to match against the regular expression
file-exists	Run the action if a file exists **condition-filename** Check existence of this file **condition-basedir** If condition-filename is relative, it will be considered relative to either *workspace*, *artifact-directory*, or *jenkins-home*. Default is *workspace*.

Single Conditional Action Example:

```
publishers:
    - conditional-publisher:
        - condition-kind: current-status
          condition-worst: FAILURE
          condition-best: SUCCESS
          action:
              - archive:
                  artifacts: '**/**'
                  allow-empty: 'true'
```

Multiple Conditional Actions Example (includes example of multiple actions per condition which requires v0.13 or higher of the Flexible Publish plugin):

```
publishers:
    - conditional-publisher:
        - condition-kind: always
          on-evaluation-failure: run-and-mark-unstable
          action:
              - archive:
                  artifacts: '**/**'
                  allow-empty: 'true'
              - aggregate-tests:
                  include-failed-builds: true
```

`Multiple Conditional Actions Example for pre-v0.13 versions`

copy-to-master()

Copy files to master from slave.

Requires the Jenkins Copy To Slave Plugin.

Parameters

- **includes** (*list*) – list of file patterns to copy
- **excludes** (*list*) – list of file patterns to exclude
- **destination** (*str*) – absolute path into which the files will be copied. If left blank they will be copied into the workspace of the current job (default '')
- **run-after-result** (*bool*) – If this is checked then copying files back to master will not run until the build result is finalized.(default true)

Example:

```
publishers:
  - copy-to-master:
      includes:
        - file1
        - file2*.txt
      excludes:
        - file2bad.txt
```

coverage()

WARNING: The coverage function is deprecated. Instead, use the cobertura function to generate a cobertura coverage report. Requires the Jenkins Cobertura Coverage Plugin.

Example:

```
publishers:
  - coverage
```

cppcheck()

Cppcheck result publisher Requires the Jenkins Cppcheck Plugin.

Parameters

- **pattern** (*str*) – File pattern for cppcheck xml report (required)
- **ignoreblankfiles** (*bool*) – Ignore blank files (default false)
- **allow-no-report** (*bool*) – Do not fail the build if the Cppcheck report is not found (default false)
- **thresholds** (*dict*) –
  thresholds
  Configure the build status and health. A build is considered as unstable or failure if the new or total number of issues exceeds the specified thresholds. The build health is also determined by thresholds. If the actual number of issues is between the provided thresholds, then the build health is interpolated.
  - **unstable** (*str*): Total number unstable threshold (default '')
  - **new-unstable** (*str*): New number unstable threshold (default '')
  - **failure** (*str*): Total number failure threshold (default '')
  - **new-failure** (*str*): New number failure threshold (default '')
  - **healthy** (*str*): Healthy threshold (default '')
  - **unhealthy** (*str*): Unhealthy threshold (default '')
- **severity** (*dict*) –
  severity
  Determines which severity of issues should be considered when evaluating the build status and health, default all true
  - **error** (*bool*): Severity error (default true)
  - **warning** (*bool*): Severity warning (default true)
  - **style** (*bool*): Severity style (default true)
  - **performance** (*bool*): Severity performance (default true)
  - **information** (*bool*): Severity information (default true)
  - **nocategory** (*bool*): Severity nocategory (default true)
  - **portability** (*bool*): Severity portability (default true)
- **graph** (*dict*) –
  graph
  Graph configuration
  - **xysize** (*array*): Chart width and height (default [500, 200])
  - **num-builds-in-graph** (*int*): Builds number in graph (default 0)

:arg dict display
display
which errors to display, default only sum

- **sum** (*bool*): Display sum of all issues (default true)
- **error** (*bool*): Display errors (default false)
- **warning** (*bool*): Display warnings (default false)
- **style** (*bool*): Display style (default false)
- **performance** (*bool*): Display performance (default false)
- **information** (*bool*): Display information (default false)
- **nocategory** (*bool*): Display no category (default false)
- **portability** (*bool*): Display portability (default false)

Minimal Example:

```yaml
publishers:
  - cppcheck:
      pattern: "**/cppcheck.xml"
```

Full Example:

```yaml
publishers:
  - cppcheck:
      pattern: "**/cppcheck.xml"
      # the rest is optional
      ignoreblankfiles: true
      allow-no-report: true
      # build status (new) error count thresholds
      thresholds:
        unstable: 5
        new-unstable: 5
        failure: 7
        new-failure: 3
        healthy: 5
        unhealthy: 10
        # severities which count towards the threshold, default all true
        severity:
          error: false
          warning: false
          style: false
          performance: false
          information: false
          nocategory: false
          portability: false
      graph:
        xysize: [600, 300]
        num-builds-in-graph: 10
        # which errors to display, default only sum
        display:
          sum: false
          error: true
          warning: true
          style: true
          performance: true
          information: true
          nocategory: true
          portability: true
```

cucumber-reports()

This plugin creates pretty cucumber-jvm html reports on jenkins.

Requires the Jenkins cucumber reports.

Parameters

- **json-reports-path** (*str*) – The path relative to the workspace of the json reports generated by cucumber-jvm e.g. target - leave empty to scan the whole workspace (default '')
- **file-include-pattern** (*str*) – Include pattern (default '')
- **file-exclude-pattern** (*str*) – Exclude pattern (default '')

- **plugin-url-path** (*str*) – The path to the jenkins user content url e.g. `http://host:port[/jenkins/]plugin` - leave empty if jenkins url root is host:port (default '')
- **skipped-fails** (*bool*) – Skipped steps to cause the build to fail (default false)
- **pending-fails** (*bool*) – Pending steps to cause the build to fail (default false)
- **undefined-fails** (*bool*) – Undefined steps to cause the build to fail (default false)
- **missing-fails** (*bool*) – Missing steps to cause the build to fail (default false)
- **no-flash-charts** (*bool*) – Use javascript charts instead of flash charts (default false)
- **ignore-failed-tests** (*bool*) – Entire build to fail when these tests fail (default false)
- **parallel-testing** (*bool*) – Run same test in parallel for multiple devices (default false)
- **failed-steps-number** (*int*) – Maximum number of failed steps above which build result is changed (default 0)
- **skipped-steps-number** (*int*) – Maximum number of skipped steps above which build result is changed (default 0)
- **pending-steps-number** (*int*) – Maximum number of pending steps above which build result is changed (default 0)
- **undefined-steps-number** (*int*) – Maximum number of undefined steps above which build result is changed (default 0)
- **failed-scenarios-number** (*int*) – Maximum number of failed scenarios above which build result is changed (default 0)
- **failed-features-number** (*int*) – Maximum number of failed features above which build result is changed (default 0)
- **build-status** (*list*) – Build result to which the build should be set when the report becomes failed or unstable (default '')
- **trends-limit** (*int*) – Number of past reports that should be presented. Zero means unlimited number of builds (default 0)
- **sorting-method** (*list*) – Result sorting order (default 'NATURAL')

Full example:

```
publishers:
  - cucumber-reports:
      json-reports-path: path
      plugin-url-path: http://example.com/
      file-include-pattern: '**/*.json'
      file-exclude-pattern: badfile.txt
      skipped-fails: true
      pending-fails: true
      undefined-fails: true
      missing-fails: true
      no-flash-charts: true
      ignore-failed-tests: true
      parallel-testing: true
      failed-steps-number: 1
      skipped-steps-number: 2
      pending-steps-number: 3
      undefined-steps-number: 4
      failed-scenarios-number: 5
      failed-features-number: 6
      build-status: UNSTABLE
      trends-limit: 7
      sorting-method: ALPHABETICAL
      sorting-values:
```

(continues on next page)

(continued from previous page)

```
        - key-value-pair:
            key: classification key 1
            value: classification value 1
        - key-value-pair:
            key: classification key 2
            value: classification value 2
```

Minimal Example:

```
publishers:
  - cucumber-reports
```

cucumber-testresult()

Publish cucumber test results. Requires the Jenkins Cucumber testresult.

Parameters

- **results** (*str*) – Results filename (required)
- **ignore-bad-steps** (*bool*) – Ignore not existed step results (default false)

Minimal example:

```
publishers:
- cucumber-testresult:
    results: nosetests.xml
```

Full Example:

```
publishers:
- cucumber-testresult:
    results: nosetests.xml
    ignore-bad-steps: true
```

dependency-check()

Dependency-Check is an open source utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

Requires the Jenkins OWASP Dependency-Check Plugin.

Parameters

- **pattern** (*str*) – Report filename pattern (optional)
- **can-run-on-failed** (*bool*) – Also runs for failed builds, instead of just stable or unstable builds (default false)
- **should-detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings (default false)
- **healthy** (*int*) – Sunny threshold (optional)
- **unhealthy** (*int*) – Stormy threshold (optional)
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high', defaulted to 'low')
- **thresholds** (*dict*) – Mark build as failed or unstable if the number of errors exceeds a threshold. (optional)

  thresholds

  – **unstable (*dict*)**

  unstable

  * **total-all** (*int*)
  * **total-high** (*int*)

* **total-normal** (*int*)
* **total-low** (*int*)
* **new-all** (*int*)
* **new-high** (*int*)
* **new-normal** (*int*)
* **new-low** (*int*)

– **failed** (*dict*)
failed

* **total-all** (*int*)
* **total-high** (*int*)
* **total-normal** (*int*)
* **total-low** (*int*)
* **new-all** (*int*)
* **new-high** (*int*)
* **new-normal** (*int*)
* **new-low** (*int*)

- **default-encoding** (*str*) – Encoding for parsing or showing files (optional)
- **do-not-resolve-relative-paths** (*bool*) – (default false)
- **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
- **use-previous-build-as-reference** (*bool*) – determines whether to always use the previous build as the reference build (default false)
- **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **use-delta-values** (*bool*) – If set then the number of new warnings is calculated by subtracting the total number of warnings of the current build from the reference build. (default false)

Minimal Example:

```
publishers:
  - dependency-check
```

Full Example:

```
publishers:
  - dependency-check:
      pattern: '**/dependency-check-report.xml'
      can-run-on-failed: true
      should-detect-modules: true
      healthy: 0
      unhealthy: 100
      health-threshold: 'high'
      thresholds:
          unstable:
              total-all: 90
              total-high: 80
```

(continues on next page)

(continued from previous page)

```
            total-normal: 70
            total-low: 60
            new-all: 50
            new-high: 40
            new-normal: 30
            new-low: 20
        failed:
            total-all: 91
            total-high: 81
            total-normal: 71
            total-low: 61
            new-all: 51
            new-high: 41
            new-normal: 31
            new-low: 21
    default-encoding: 'utf-8'
    do-not-resolve-relative-paths: true
    dont-compute-new: false
    use-previous-build-as-reference: true
    use-stable-build-as-reference: true
    use-delta-values: true
```

description-setter()

This plugin sets the description for each build, based upon a RegEx test of the build log file.

Requires the Jenkins Description Setter Plugin.

Parameters

- **regexp** (*str*) – A RegEx which is used to scan the build log file (default '')
- **regexp-for-failed** (*str*) – A RegEx which is used for failed builds (default '')
- **description** (*str*) – The description to set on the build (optional)
- **description-for-failed** (*str*) – The description to set on the failed builds (optional)
- **set-for-matrix** (*bool*) – Also set the description on a multi-configuration build (default false)

Minimal Example:

```
publishers:
  - description-setter
```

Full Example:

```
publishers:
  - description-setter:
      regexp: ".*(<a href=.*a>)"
      regexp-for-failed: ".*(<a href=.*a>)"
      description: "some description"
      description-for-failed: "another description"
      set-for-matrix: true
```

disable-failed-job()

Automatically disable failed jobs.

Requires the Jenkins Disable Failed Job Plugin.

Parameters

- **when-to-disable** (*str*) – The condition to disable the job. (required) Possible values are
  - **Only Failure**
  - **Failure and Unstable**
  - **Unstable**
- **no-of-failures** (*int*) – Number of consecutive failures to disable the job. (optional)

Example:

```
publishers:
  - disable-failed-job:
      when-to-disable: 'Failure and Unstable'
      no-of-failures: 3
```

discord-notifier()

This plugin allows for a job to publish results to discord Requires the Jenkins https://plugins.jenkins.io/discord-notifier.

Parameters

webhook-url (*str*) – Discord webhook URL (required)

Minimal Example:

```
publishers:
  - discord-notifier:
      webhook-url: "https://discord.com/api/webhooks/token"
```

Full Example:

```
publishers:
  - discord-notifier:
      webhook-url: "https://discord.com/api/webhooks/token"
      branch-name: "branchName"
      status-title: "statusTitle"
      thumbnail-url: "thumbnailURL"
      notes: "notes"
      custom-avatar-url: "customAvatarUrl"
      custom-username: "customUsername"
      send-on-state-change: true
      enable-url-linking: true
      enable-artifact-list: true
      enable-footer-info: true
      show-changeset: true
      send-log-file: true
      send-start-notification: true
```

display-upstream-changes()

Display SCM changes of upstream jobs. Requires the Jenkins Display Upstream Changes Plugin.

Example:

```
publishers:
  - display-upstream-changes
```

docker-stop-container()

This plugin allows removing stopped docker containers. It requires the Docker build step plugin.

Parameters

remove-stopped-containers (*bool*) – Boolean value to remove stopped docker containers (default False)

Minimal Example: .. literalinclude:: /../../tests/ publishers/fixtures/docker-stop-container-minimal.yaml

Full Example: .. literalinclude:: /../../tests/ publishers/fixtures/docker-stop-container-full.yaml

downstream-ext()

Trigger multiple downstream jobs when a job is completed and condition is met.

Requires the Jenkins Downstream-Ext Plugin.

Parameters

- **projects** (*list*) – Projects to build (required)
- **condition** (*str*) – comparison condition used for the criteria. One of 'equal-or-over', 'equal-or-under', 'equal' (default 'equal-or-over')
- **criteria** (*str*) – Trigger downstream job if build results meets condition. One of 'success', 'unstable', 'failure' or 'aborted' (default 'success')
- **only-on-scm-change** (*bool*) – Trigger only if downstream project has SCM changes (default false)
- **only-on-local-scm-change** (*bool*) – Trigger only if current project has SCM changes (default false)

Example:

```
publishers:
  - downstream-ext:
      projects:
        - foo
        - bar
      only-on-scm-change: true
      criteria: unstable
      condition: equal
```

doxygen()

This plugin parses the Doxygen descriptor (Doxyfile) and provides a link to the generated Doxygen documentation.

Requires the Jenkins Doxygen Plugin.

Parameters

- **doxyfile** (*str*) – The doxyfile path (required)
- **slave** (*str*) – The node or label to pull the doxygen HTML files from (default '')
- **keep-all** (*bool*) – Retain doxygen generation for each successful build (default false)
- **folder** (*str*) – Folder where you run doxygen (default '')

Minimal Example:

```
publishers:
  - doxygen:
      doxyfile: "Doxyfile"
```

Full Example:

```
publishers:
  - doxygen:
      doxyfile: "Doxyfile"
      slave: "doxygen-slave"
      keep-all: true
      folder: "build"
```

dry()

Publish trend reports with DRY.

Requires the Jenkins DRY Plugin (https://github.com/jenkinsci/dry-plugin).

The DRY component accepts a dictionary with the following values:

Parameters

- **pattern** (*str*) – Report filename pattern (default '')
- **can-run-on-failed** (*bool*) – Also runs for failed builds, instead of just stable or unstable builds (default false)
- **should-detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings (default false)
- **healthy** (*int*) – Sunny threshold (default '')
- **unhealthy** (*int*) – Stormy threshold (default '')
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high', defaulted to 'low')
- **high-threshold** (*int*) – Minimum number of duplicated lines for high priority warnings. (default 50)
- **normal-threshold** (*int*) – Minimum number of duplicated lines for normal priority warnings. (default 25)
- **thresholds** (*dict*) – Mark build as failed or unstable if the number of errors exceeds a threshold. (default '')

  thresholds

  - **unstable (*dict*)**

    unstable

    - **total-all** (*int*)
    - **total-high** (*int*)
    - **total-normal** (*int*)
    - **total-low** (*int*)
    - **new-all** (*int*)
    - **new-high** (*int*)
    - **new-normal** (*int*)
    - **new-low** (*int*)
  - **failed (*dict*)**

    failed

    - **total-all** (*int*)
    - **total-high** (*int*)
    - **total-normal** (*int*)
    - **total-low** (*int*)
    - **new-all** (*int*)
    - **new-high** (*int*)
    - **new-normal** (*int*)
    - **new-low** (*int*)
- **default-encoding** (*str*) – Encoding for parsing or showing files (optional)
- **do-not-resolve-relative-paths** (*bool*) – (default false)

- **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
- **use-previous-build-as-reference** (*bool*) – determines whether to always use the previous build as the reference build (default false)
- **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **use-delta-values** (*bool*) – If set then the number of new warnings is calculated by subtracting the total number of warnings of the current build from the reference build. (default false)

Example:

```
publishers:
 - dry:
    pattern: '**/cpd-result.xml'
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    high-threshold: 50
    normal-threshold: 25
    thresholds:
        unstable:
            total-high: 10
        failed:
            total-high: 1
```

Full example:

```
publishers:
 - dry:
    pattern: '**/cpd-result.xml'
    can-run-on-failed: true
    should-detect-modules: true
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    high-threshold: 20
    normal-threshold: 10
    thresholds:
        unstable:
            total-all: 90
            total-high: 80
            total-normal: 70
            total-low: 60
            new-all: 50
            new-high: 40
            new-normal: 30
            new-low: 20
        failed:
            total-all: 91
            total-high: 81
            total-normal: 71
            total-low: 61
            new-all: 51
```

(continues on next page)

(continued from previous page)

```
            new-high: 41
            new-normal: 31
            new-low: 21
    default-encoding: 'utf-8'
    do-not-resolve-relative-paths: true
    dont-compute-new: false
    use-stable-build-as-reference: true
    use-delta-values: true
```

email()

Email notifications on build failure. Requires the Jenkins Mailer Plugin.

Parameters

- **recipients** (*str*) – Space separated list of recipient email addresses (required)
- **notify-every-unstable-build** (*bool*) – Send an email for every unstable build (default true)
- **send-to-individuals** (*bool*) – Send an email to the individual who broke the build (default false)

Example:

```
publishers:
  - email:
      recipients: foo@example.com
```

```
publishers:
  - email:
      recipients: foo@example.com bar@example.com
      notify-every-unstable-build: false
      send-to-individuals: true
```

email-ext()

Extend Jenkin's built in email notification Requires the Jenkins Email-ext Plugin.

Parameters

- **disable-publisher** (*bool*) – Disable the publisher, while maintaining the settings. The usage model for this is when you want to test things out in the build, not send out e-mails during the testing. A message will be printed to the build log saying that the publisher is disabled. (default false)
- **recipients** (*str*) – Comma separated list of recipient email addresses (default '$DEFAULT_RECIPIENTS')
- **reply-to** (*str*) – Comma separated list of email addresses that should be in the Reply-To header for this project (default '$DEFAULT_REPLYTO')
- **from** (*str*) – Email address that should be in the From header for this project (default '')
- **content-type** (*str*) – The content type of the emails sent. If not set, the Jenkins plugin uses the value set on the main configuration page. Possible values: 'html', 'text', 'both-html-text' or 'default' (default 'default')
- **subject** (*str*) – Subject for the email, can include variables like ${BUILD_NUMBER} or even groovy or javascript code (default '$DEFAULT_SUBJECT')
- **body** (*str*) – Content for the body of the email, can include variables like ${BUILD_NUMBER}, but the real magic is using groovy or javascript to hook into the Jenkins API itself (default '$DEFAULT_CONTENT')
- **attach-build-log** (*bool*) – Include build log in the email (default false)

- **compress-log** (*bool*) – Compress build log in the email (default false)
- **attachments** (*str*) – pattern of files to include as attachment (default '')
- **always** (*bool*) – Send an email for every result (default false)
- **unstable** (*bool*) – Send an email for an unstable result (default false)
- **first-failure** (*bool*) – Send an email for just the first failure (default false)
- **first-unstable** (*bool*) – Send an email for just the first unstable build (default false)
- **not-built** (*bool*) – Send an email if not built (default false)
- **aborted** (*bool*) – Send an email if the build is aborted (default false)
- **regression** (*bool*) – Send an email if there is a regression (default false)
- **failure** (*bool*) – Send an email if the build fails (default true)
- **second-failure** (*bool*) – Send an email for the second failure (default false)
- **improvement** (*bool*) – Send an email if the build improves (default false)
- **still-failing** (*bool*) – Send an email if the build is still failing (default false)
- **success** (*bool*) – Send an email for a successful build (default false)
- **fixed** (*bool*) – Send an email if the build is fixed (default false)
- **fixed-unhealthy** (*bool*) – Send an email if the build status changes from "Failure" or "Unstable" to "Success". Intermediate "Aborted" builds are ignored. (default false)
- **still-unstable** (*bool*) – Send an email if the build is still unstable (default false)
- **pre-build** (*bool*) – Send an email before the build (default false)
- **trigger-script** (*str*) – A Groovy script used to determine if an email should be sent.
- **presend-script** (*str*) – A Groovy script executed prior sending the mail. (default '')
- **postsend-script** (*str*) – A Goovy script executed after sending the email. (default '')
- **save-output** (*bool*) – Save email content to workspace (default false)
- **matrix-trigger** (*str*) – If using matrix projects, when to trigger

  matrix-trigger values

  - **both**
  - **only-parent**
  - **only-configurations**
- **send-to** (*list*) – list of recipients from the predefined groups

  send-to values

  - **developers** (disabled by default)
  - **requester** (disabled by default)
  - **culprits** (disabled by default)
  - **recipients** (enabled by default)
  - **upstream-committers** (>=2.39) (disabled by default)
  - **failing-test-suspects-recipients** (>=2.39) (disabled by default)
  - **first-failing-build-suspects-recipients** (>=2.39) (disabled by default)

Example:

```
publishers:
  - email-ext:
      recipients: foo@example.com, bar@example.com
      reply-to: foo@example.com
      content-type: html
      subject: Subject for Build ${BUILD_NUMBER}
      body: The build has finished
      attach-build-log: false
      compress-log: false
```

(continues on next page)

(continued from previous page)

```
      attachments: "*/foo*.log"
      always: true
      unstable: true
      first-failure: true
      first-unstable: true
      not-built: true
      aborted: true
      regression: true
      failure: true
      second-failure: true
      improvement: true
      still-failing: true
      success: true
      fixed: true
      fixed-unhealthy: true
      still-unstable: true
      pre-build: true
      matrix-trigger: only-configurations
      presend-script: "cancel=true"
      postsend-script: "cancel=true"
      save-output: true
      send-to:
        - developers
        - requester
        - culprits
        - recipients
        - upstream-committers
```

emotional-jenkins()

Emotional Jenkins. This funny plugin changes the expression of Mr. Jenkins in the background when your builds fail.

Requires the Jenkins Emotional Jenkins Plugin.

Example:

```
publishers:
  - emotional-jenkins
```

findbugs()

FindBugs reporting for builds

Requires the Jenkins FindBugs Plugin (https://github.com/jenkinsci/findbugs-plugin).

Parameters

- **pattern** (*str*) – specifies the generated raw FindBugs XML report files, such as **/findbugs.xml or **/findbugsXml.xml. (default '')
- **rank-priority** (*bool*) – Use rank as priority (default false)
- **include-files** (*str*) – Comma separated list of files to include. (default '')
- **exclude-files** (*str*) – Comma separated list of files to exclude. (default '')
- **can-run-on-failed** (*bool*) – Weather or not to run plug-in on failed builds (default false)
- **should-detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings. (default false)
- **healthy** (*int*) – Sunny threshold (default '')

- **unhealthy** (*int*) – Stormy threshold (default '')
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high', defaulted to 'low')
- **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
- **use-delta-values** (*bool*) – Use delta for new warnings. (default false)
- **use-previous-build-as-reference** (*bool*) – If set then the number of new warnings will always be calculated based on the previous build. Otherwise the reference build. (default false)
- **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **thresholds** (*dict*) –

  thresholds

  – **unstable (*dict*)**

  unstable

  * **total-all** (*int*)
  * **total-high** (*int*)
  * **total-normal** (*int*)
  * **total-low** (*int*)
  * **new-all** (*int*)
  * **new-high** (*int*)
  * **new-normal** (*int*)
  * **new-low** (*int*)

  – **failed (*dict*)**

  failed

  * **total-all** (*int*)
  * **total-high** (*int*)
  * **total-normal** (*int*)
  * **total-low** (*int*)
  * **new-all** (*int*)
  * **new-high** (*int*)
  * **new-normal** (*int*)
  * **new-low** (*int*)

Minimal Example:

```
publishers:
  - findbugs
```

Full Example:

```
publishers:
    - findbugs:
        pattern: '**/findbugs.xml'
        rank-priority: true
```

(continues on next page)

(continued from previous page)

```
        include-files: 'f,d,e,.*'
        exclude-files: 'a,c,d,.*'
        can-run-on-failed: true
        should-detect-modules: true
        healthy: 80
        unhealthy: 10
        use-delta-values: true
        health-threshold: 'high'
        thresholds:
            unstable:
                total-all: 90
                total-high: 80
                total-normal: 50
                total-low: 20
                new-all: 95
                new-high: 85
                new-normal: 55
                new-low: 25
            failed:
                total-all: 80
                total-high: 70
                total-normal: 40
                total-low: 10
                new-all: 85
                new-high: 75
                new-normal: 45
                new-low: 15
        dont-compute-new: false
        use-delta-values: true
        use-previous-build-as-reference: true
        use-stable-build-as-reference: true
```

fingerprint()

Fingerprint files to track them across builds. Requires the Jenkins Fingerprint Plugin.

Parameters

- **files** (*str*) – files to fingerprint, follows the @includes of Ant fileset (default '')
- **record-artifacts** (*bool*) – fingerprint all archived artifacts (default false)

Example:

```
publishers:
  - fingerprint:
      files: builddir/test*.xml
      record-artifacts: false
```

fitnesse()

Publish Fitnesse test results

Requires the Jenkins Fitnesse plugin.

Parameters

results (*str*) – path specifier for results files

Example:

```
publishers:
  - fitnesse:
      results: "fitnesse-results/**/*.xml"
```

flowdock()

This plugin publishes job build results to a Flowdock flow.

Requires the Jenkins Flowdock Plugin.

Parameters

- **token** (*str*) – API token for the targeted flow. (required)
- **tags** (*str*) – Comma-separated list of tags to include in message (default “”)
- **chat-notification** (*bool*) – Send chat notification when build fails (default true)
- **notify-success** (*bool*) – Send notification on build success (default true)
- **notify-failure** (*bool*) – Send notification on build failure (default true)
- **notify-fixed** (*bool*) – Send notification when build is fixed (default true)
- **notify-unstable** (*bool*) – Send notification when build is unstable (default false)
- **notify-aborted** (*bool*) – Send notification when build was aborted (default false)
- **notify-notbuilt** (*bool*) – Send notification when build did not occur (default false)

Example:

```
publishers:
  - flowdock:
      token: abcdefghijklmnopqrstuvwxyzabcdef
```

Full example:

```
publishers:
  - flowdock:
      token: abcdefghijklmnopqrstuvwxyzabcdef
      tags: jenkins,ci
      chat-notification: true
      notify-success: true
      notify-failure: true
      notify-fixed: true
      notify-unstable: false
      notify-aborted: false
      notify-notbuilt: false
```

ftp()

Upload files via FTP. Requires the Jenkins Publish over FTP Plugin.

Parameters

- **site** (*str*) – name of the ftp site (required)
- **target** (*str*) – destination directory (required)
- **target-is-date-format** (*bool*) – whether target is a date format. If true, raw text should be quoted (default false)
- **clean-remote** (*bool*) – should the remote directory be deleted before transferring files (default false)
- **source** (*str*) – source path specifier (required)
- **excludes** (*str*) – excluded file pattern (optional)
- **remove-prefix** (*str*) – prefix to remove from uploaded file paths (optional)
- **fail-on-error** (*bool*) – fail the build if an error occurs (default false).
- **flatten** (*bool*) – only create files on the server, don’t create directories (default false).
- **verbose** (*bool*) – adds lots of detail useful for debug to the console but generally should be left off (default false)

- **retries** (*int*) – the number of times to retry this server in the event of failure (optional)
- **retry-delay** (*int*) – the time to wait, in milliseconds, before attempting another transfer (default 10000)

Minimal Example:

```
publishers:
  - ftp:
      site: 'ftp.example.com'
      target: 'dest/dir'
      source: 'base/source/dir/**'
```

Full Example:

```
publishers:
  - ftp:
      site: 'ftp.example.com'
      target: "'dest/dir/'yyyyMMddHHmmss"
      target-is-date-format: true
      clean-remote: true
      source: 'base/source/dir/**'
      excludes: '**/*.excludedfiletype'
      remove-prefix: 'base/source/dir'
      fail-on-error: true
      flatten: true
      verbose: true
      retries: 99
      retry-delay: 12345
```

ftp-publisher()

This plugin can be used to upload project artifacts and whole directories to an ftp server. Requires the Jenkins FTP-Publisher Plugin.

Parameters

- **uploads** (*list*) – List of files to upload
  - **uploads**
    - **file-path ('str') – Destination folder. It will be created** if doesn't exists. Created relative to ftp root directory. (default '')
    - **source-file ('str') – Source files which will be uploaded** (default '')
- **site-name** (*str*) – Name of FTP server to upload to (required)
- **use-timestamps** (*bool*) – Use timestamps in the FTP directory path (default false)
- **flatten-files** (*bool*) – Flatten files on the FTP host (default false)
- **skip-publishing** (*bool*) – Skip publishing (default false)

Minimal Example:

```
publishers:
  - ftp-publisher:
      site-name: foo
```

Full Example:

```
publishers:
  - ftp-publisher:
      uploads:
```

(continues on next page)

(continued from previous page)

```
        - file-path: destination/folder
          source-file: folder/dist/*.jar
        - file-path: foo/bar
          source-file: foo/bar/*.ear
      site-name: foo
      use-timestamps: true
      flatten-files: true
      skip-publishing: true
```

gatling()

Publish gatling results as a trend graph Requires the Jenkins Gatling Plugin.

Example:

```
publishers:
  - gatling
```

git()

This plugin will configure the Jenkins Git plugin to push merge results, tags, and/or branches to remote repositories after the job completes.

Requires the Jenkins Git Plugin.

Parameters

- **push-merge** (*bool*) – push merges back to the origin specified in the pre-build merge options (default false)
- **push-only-if-success** (*bool*) – Only push to remotes if the build succeeds - otherwise, nothing will be pushed. (default true)
- **force-push** (*bool*) – Add force option to git push (default false)
- **tags** (*list*) – tags to push at the completion of the build
  - **tag**
    - **remote** (*str*) remote repo name to push to (default 'origin')
    - **name** (*str*) name of tag to push
    - **message** (*str*) message content of the tag
    - **create-tag** (*bool*) whether or not to create the tag after the build, if this is False then the tag needs to exist locally (default false)
    - **update-tag** (*bool*) whether to overwrite a remote tag or not (default false)
- **branches** (*list*) – branches to push at the completion of the build
  - **branch**
    - **remote** (*str*) remote repo name to push to (default 'origin')
    - **name** (*str*) name of remote branch to push to
    - **rebase** (*bool*) whether or not to rebase before push (default false)
- **notes** (*list*) – notes to push at the completion of the build
  - **note**
    - **remote** (*str*) remote repo name to push to (default 'origin')
    - **message** (*str*) content of the note
    - **namespace** (*str*) namespace of the note (default master)
    - **replace-note** (*bool*) whether to overwrite a note or not (default false)

Minimal Example:

```
publishers:
  - git
```

Full Example:

```
publishers:
  - git:
      push-merge: true
      push-only-if-success: false
      force-push: true
      tags:
          - tag:
              remote: tagremotename
              name: tagname
              message: "some tag message"
              create-tag: true
              update-tag: true
      branches:
          - branch:
              remote: branchremotename
              name: "some/branch"
              rebase: true
      notes:
          - note:
              remote: remotename
              message: "some note to push"
              namespace: notenamespace
              replace-note: true
```

github-notifier()

Set build status on Github commit. Requires the Jenkins Github Plugin.

Example:

```
publishers:
  - github-notifier
```

github-pull-request-merge()

This action merges the pull request that triggered the build (see the github pull request trigger)

Requires the Jenkins GitHub pull request builder plugin.

Parameters

- **only-admins-merge** (*bool*) – if *true* only administrators can merge the pull request, (default false)
- **disallow-own-code** (*bool*) – if *true* will allow merging your own pull requests, in opposite to needing someone else to trigger the merge. (default false)
- **merge-comment** (*str*) – Comment to set on the merge commit (default '')
- **fail-on-non-merge** (*bool*) – fail the job if the merge was unsuccessful (default false)
- **delete-on-merge** (*bool*) – Delete the branch of the pull request on successful merge (default false)

Full Example:

```
publishers:
  - github-pull-request-merge:
```

(continues on next page)

(continued from previous page)

```
      only-admins-merge: true
      disallow-own-code: true
      merge-comment: 'my fancy commit message'
      fail-on-non-merge: true
      delete-on-merge: true
```

Minimal Example:

```
publishers:
  - github-pull-request-merge
```

gitlab-message()

Add note with build status on GitLab merge request. Requires the Jenkins GitLab Plugin.

Parameters

- **failure-only** (*bool*) – make a comment only on failure (default false)
- **success-note** (*bool*) – make a comment on GitLab Merge Request if build succeeds (default false)
- **failure-note** (*bool*) – make a comment on GitLab Merge Request if build failed (default false)
- **abort-note** (*bool*) – make a comment on GitLab Merge Request if build aborted (default false)
- **unstable-note** (*bool*) – make a comment on GitLab Merge Request if build unstable (default false)
- **success-note-text** (*str*) – text of comment on success build (default '')
- **failure-note-text** (*str*) – text of comment on failed build (default '')
- **abort-note-text** (*str*) – text of comment on aborted build (default '')
- **unstable-note-text** (*str*) – text of comment on unstable build (default '')

Minimal Example:

```
publishers:
  - gitlab-message
```

Full Example:

```
publishers:
  - gitlab-message:
      failure-only: true
      success-note: true
      success-note-text: "SUCCESS"
      failure-note: true
      failure-note-text: "Build was failed. See log on Jenkins"
      abort-note: true
      abort-note-text: "Build was aborted"
      unstable-note: true
      unstable-note-text: "The build is unstable"
```

gitlab-notifier()

Set build status on GitLab commit. Requires the Jenkins GitLab Plugin.

Parameters

- **name** (*str*) – The name of the build in GitLab. With this you can distinguish different Jenkins jobs for the same commit in GitLab. (default 'jenkins')
- **mark-unstable-as-success** (*bool*) – (default false)

Minimal Example:

```
publishers:
  - gitlab-notifier
```

Full Example:

```
publishers:
  - gitlab-notifier:
      name: foobar-jenkins
      mark-unstable-as-success: true
```

gitlab-vote()

Set vote for build status on GitLab merge request. Requires the Jenkins GitLab Plugin.

Example:

```
publishers:
  - gitlab-vote
```

google-cloud-storage()

Upload build artifacts to Google Cloud Storage. Requires the Jenkins Google Cloud Storage plugin.

Apart from the Google Cloud Storage Plugin itself, installation of Google OAuth Credentials and addition of required credentials to Jenkins is required.

Parameters

- **credentials-id** (*str*) – The set of Google credentials registered with the Jenkins Credential Manager for authenticating with your project. (required)
- **uploads** (*list*) –

  uploads

  – **expiring-elements (*dict*)**

  params

  * **bucket-name** (*str*) bucket name to upload artifacts (required)
  * **days-to-retain** (*int*) days to keep artifacts (required)

  – **build-log (*dict*)**

  params

  * **log-name** (*str*) name of the file that the Jenkins console log to be named (required)
  * **storage-location** (*str*) bucket name to upload artifacts (required)
  * **share-publicly** (*bool*) whether to share uploaded share uploaded artifacts with everyone (default false)
  * **upload-for-failed-jobs** (*bool*) whether to upload artifacts even if the build fails (default false)

* **show-inline** (*bool*) whether to show uploaded build log inline in web browsers, rather than forcing it to be downloaded (default true)
* **strip-prefix** (*str*) strip this prefix off the file names (default not set)

– **classic** (*dict*)
params

* **file-pattern** (*str*) ant style globs to match the files to upload (required)
* **storage-location** (*str*) bucket name to upload artifacts (required)
* **share-publicly** (*bool*) whether to share uploaded share uploaded artifacts with everyone (default false)
* **upload-for-failed-jobs** (*bool*) whether to upload artifacts even if the build fails (default false)
* **show-inline** (*bool*) whether to show uploaded artifacts inline in web browsers, rather than forcing them to be downloaded (default false)
* **strip-prefix** (*str*) strip this prefix off the file names (default not set)

Example:

```
publishers:
    - google-cloud-storage:
        credentials-id: 'myCredentials'
        uploads:
            - expiring-elements:
                bucket-name: 'gs://myBucket'
                days-to-retain: 7
```

Full example:

```
publishers:
    - google-cloud-storage:
        credentials-id: 'myCredentials'
        uploads:
            - expiring-elements:
                bucket-name: 'gs://myBucket'
```

(continues on next page)

(continued from previous page)

```
            days-to-retain: 7
        - build-log:
            log-name: 'console.log'
            storage-location: 'gs://myBucket'
            upload-for-failed-jobs: true
            share-publicly: true
        - classic:
            file-pattern: 'target/*.war'
            storage-location: 'gs://myBucket'
            upload-for-failed-jobs: true
        - classic:
            file-pattern: '**/build/*.iso'
            storage-location: 'gs://myBucket/artifacts/'
            share-publicly: true
            strip-prefix: 'path/to/'
```

groovy-postbuild()

Execute a groovy script. Requires the Jenkins Groovy Postbuild Plugin.

Please pay attention on version of plugin you have installed. There were incompatible changes between 1.x and 2.x. Please see home page of this plugin for full information including migration process.

Parameters

- **script** (*str*) – The groovy script to execute
- **classpath** (*list*) – List of additional classpaths (>=1.6)
- **on-failure** (*str*) – In case of script failure leave build as it is for "nothing" option, mark build as unstable for "unstable" and mark job as failure for "failed" (default 'nothing')
- **matrix-parent** (*bool*) – Run script for matrix parent only (>=1.9) (default false)
- **sandbox** (*bool*) – Execute script inside of groovy sandbox (>=2.0) (default false)

Example:

```
publishers:
    - groovy-postbuild:
        script: "manager.buildFailure()"
        classpath:
            - "file:///path/to/your/lib"
            - "file:///path/to/your/lib"
        on-failure: "failed"
        matrix-parent: true
```

growl()

Push notifications to growl client. Requires the Jenkins Growl Plugin.

Parameters

- **ip** (*str*) – IP address to send growl notifications to (required)
- **notify-only-on-fail-or-recovery** (*bool*) – send a growl only when build fails or recovers from a failure (default false)

Minimal Example:

```
publishers:
  - growl:
      ip: foo.ip.address
```

Full Example:

```
publishers:
  - growl:
      ip: foo.ip.address
      notify-only-on-fail-or-recovery: true
```

hipchat()

Publisher that sends hipchat notifications on job events Requires the Jenkins Hipchat Plugin version >=1.9

Please see documentation for older plugin version https://jenkins-job-builder.readthedocs.io/en/latest/hipchat.html

Parameters

- **token** (*str*) – This will override the default auth token (optional)
- **rooms** (*list*) – list of HipChat rooms to post messages to, overrides global default (optional)
- **notify-start** (*bool*) – post messages about build start event (default false)
- **notify-success** (*bool*) – post messages about successful build event (default false)
- **notify-aborted** (*bool*) – post messages about aborted build event (default false)
- **notify-not-built** (*bool*) – post messages about build set to NOT_BUILT. This status code is used in a multi-stage build where a problem in earlier stage prevented later stages from building. (default false)
- **notify-unstable** (*bool*) – post messages about unstable build event (default false)
- **notify-failure** (*bool*) – post messages about build failure event (default false)
- **notify-back-to-normal** (*bool*) – post messages about build being back to normal after being unstable or failed (default false)
- **start-message** (*str*) – This will override the default start message (optional)
- **complete-message** (*str*) – This will override the default complete message (optional)

Example:

```
publishers:
  - hipchat:
      token: auth
      rooms:
        - room1
        - room2
      notify-start: true
      notify-aborted: true
      start-message: job started
      complete-message: job completed
```

hp-alm()

Publish test results to HP-ALM.

Requires the Jenkins Micro Focus Application Automation Tools.

Parameters

- **server-name** (*str*) – The name of the ALM Server. (required)
- **credentials-id** (*str*) – credentials-id of the user (default '')
- **domaine** (*str*) – The Domain of the project to be used. (required)
- **client-type** (*str*) – Client type is required for some ALM above 12.60 in authentication.(default '')
- **project** (*str*) – The project to be used. (required)
- **testing-framework** (*str*) – The testing framework that is used when generate the testing result file. (default Junit)
- **testing-tool** (*str*) – The testing tool that is used when generate the testing result file. (default '')

- **folder** (*str*) – The path of the test folder that will contain the uploaded test. The path doesn't include the Root test folder (Subject). For example, sampletestfolder/subfolder means, the tests will be uploaded to test folder named 'subfolder', which is under the test folder named 'sampletestfolder', and 'sampletestfolder' is under the root test folder 'Subject'. (required)
- **set-folder** (*str*) – The path of the testset folder that will contain the uploaded testset. The path doesn't include the Root testset folder. For example, sampletestsetfolder/subfolder means, the testsets will be uploaded to testset folder named 'subfolder', which is under the testset folder named 'sampletestsetfolder', and 'sampletestsetfolder' is under the root testset folder 'Root'. (required)
- **testing-result-file** (*str*) – The condition to find the testing result file, start from the root path of the job. For example, `**/junitResult.xml` to find testing result file for Junit Plugin, `**/testng-results.xml` to find testing result file for TestNG plugin. (required)
- **jenkins-server-url** (*str*) – The HTTP URL of the Jenkins Server, form example, http://jenkins.example.org:8080 . (optional)

Minimal example using defaults:

```
publishers:
  - hp-alm:
      server-name: HP-ALM
      domain: FOO_COMPANY
      project: foo_project
      folder: 'ALM/foo/release1/test_case1'
      set-folder: 'ALM/foo/release1/test_case1/$env'
      testing-result-file: '**/junitResult.xml'
```

Full example:

```
publishers:
  - hp-alm:
      server-name: HP-ALM
      credentials-id: cba09876-4321-4567-890a-bcde12345678
      domain: FOO_COMPANY
      project: foo_project
      client-type: foo_client
      testing-framework: JUnit
      testing-tool: foo_tool
      folder: 'ALM/foo/release1/test_case1'
      set-folder: 'ALM/foo/release1/test_case1/$env'
      testing-result-file: '**/junitResult.xml'
      jenkins-server-url: 'http://myjenkinsserver.test.com:8080'
```

html-publisher()

This plugin publishes HTML reports.

Requires the Jenkins HTML Publisher Plugin.

Parameters

- **name** (*str*) – Report name (required)
- **dir** (*str*) – HTML directory to archive (required)
- **files** (*str*) – Specify the pages to display (required)
- **keep-all** (*bool*) – keep HTML reports for each past build (default false)
- **allow-missing** (*bool*) – Allow missing HTML reports (default false)
- **link-to-last-build** (*bool*) – If this and 'keep-all' both are true, it publishes the link on project level even if build failed. (default false)

Example:

```
publishers:
  - html-publisher:
      name: "some name"
      dir: "path/"
      files: "index.html"
      keep-all: true
      allow-missing: true
      link-to-last-build: true
```

hue-light()

This plugin shows the state of your builds using the awesome Philips hue lights.

Requires the Jenkins hue-light Plugin.

Parameters

- **light-id** (*int*) – ID of light. Define multiple lights by a comma as a separator (required)
- **pre-build** (*str*) – Colour of building state (default 'blue')
- **good-build** (*str*) – Colour of successful state (default 'green')
- **unstable-build** (*str*) – Colour of unstable state (default 'yellow')
- **bad-build** (*str*) – Colour of unsuccessful state (default 'red')

Full Example:

```
publishers:
  - hue-light:
      light-id: 123
      pre-build: yellow
      good-build: red
      unstable-build: blue
      bad-build: green
```

Minimal Example:

```
publishers:
  - hue-light:
      light-id: 123
```

image-gallery()

Produce an image gallery using Javascript library. Requires the Jenkins Image Gallery Plugin.

Parameters

- **gallery-type** (*str*) –
  gallery-type values
  - **archived-images-gallery** (default)
  - **in-folder-comparative-gallery**
  - **multiple-folder-comparative-gallery**
- **title** (*str*) – gallery title (optional)
- **image-width** (*int*) – width of the image (optional)
- **unstable-if-no-artifacts** (*bool*) – mark build as unstable if no archived artifacts were found (default false)
- **includes** (*str*) – include pattern (valid for archived-images-gallery gallery)
- **base-root-folder** (*str*) – base root dir (valid for comparative gallery)
- **image-inner-width** (*int*) – width of the image displayed in the inner gallery popup (valid for comparative gallery, optional)

Example:

```
publishers:
    - image-gallery:
        - gallery-type: archived-images-gallery
          title: Gallery 1
          includes: path/images
          image-width: 100
          unstable-if-no-artifacts: true
        - gallery-type: in-folder-comparative-gallery
          title: Gallery 2
          base-root-folder: path/images2
          image-width: 321
          image-inner-width: 111
          unstable-if-no-artifacts: false
        - gallery-type: multiple-folder-comparative-gallery
          title: Gallery 3
          base-root-folder: path/images3
          image-width: 222
          image-inner-width: 1
```

influx-db()

Requires the Jenkins :jenkins-plugins: *Influx DB <influxdb>*.

ircbot()

ircbot enables Jenkins to send build notifications via IRC and lets you interact with Jenkins via an IRC bot.

Requires the Jenkins IRC Plugin.

Parameters

- **strategy** (*str*) – When to send notifications
  - **strategy values**
    - **all** always (default)
    - **any-failure** on any failure
    - **failure-and-fixed** on failure and fixes
    - **new-failure-and-fixed** on new failure and fixes
    - **statechange-only** only on state change
- **notify-start** (*bool*) – Whether to send notifications to channels when a build starts (default false)
- **notify-committers** (*bool*) – Whether to send notifications to the users that are suspected of having broken this build (default false)
- **notify-culprits** (*bool*) – Also send notifications to 'culprits' from previous unstable/failed builds (default false)
- **notify-upstream** (*bool*) – Whether to send notifications to upstream committers if no committers were found for a broken build (default false)
- **notify-fixers** (*bool*) – Whether to send notifications to the users that have fixed a broken build (default false)
- **message-type** (*str*) – Channel Notification Message.
  - **message-type values**
    - **summary-scm** for summary and SCM changes (default)
    - **summary** for summary only
    - **summary-params** for summary and build parameters
    - **summary-scm-fail** for summary, SCM changes, failures)
- **channels** (*list*) – list channels definitions If empty, it takes channel from Jenkins configuration. (default empty) WARNING: the IRC plugin requires the channel to be configured in the system wide configuration or the jobs will fail to emit notifications to the channel

Channel

- **name** (*str*) Channel name
- **password** (*str*) Channel password (optional)
- **notify-only** (*bool*) Set to true if you want to disallow bot commands (default false)

- **matrix-notifier** (*str*) – notify for matrix projects instant-messaging-plugin injects an additional field in the configuration form whenever the project is a multi-configuration project

  matrix-notifier values

  - **all**
  - **only-configurations** (default)
  - **only-parent**

Minimal Example:

```
publishers:
  - ircbot
```

Full Example:

```
publishers:
  - ircbot:
      strategy: failure-and-fixed
      notify-start: true
      notify-committers: true
      notify-culprits: true
      notify-upstream: true
      notify-fixers: true
      message-type: summary
      channels:
          - name: '#jenkins-channel1'
            password: secrete
            notify-only: false
          - name: '#jenkins-channel2'
            notify-only: true
      matrix-notifier: all
```

`jabber()`

Integrates Jenkins with the Jabber/XMPP instant messaging protocol Requires the Jenkins Jabber Plugin.

Parameters

- **notify-on-build-start** (*bool*) – Whether to send notifications to channels when a build starts (default false)
- **notify-scm-committers** (*bool*) – Whether to send notifications to the users that are suspected of having broken this build (default false)
- **notify-scm-culprits** (*bool*) – Also send notifications to 'culprits' from previous unstable/failed builds (default false)
- **notify-upstream-committers** (*bool*) – Whether to send notifications to upstream committers if no committers were found for a broken build (default false)
- **notify-scm-fixers** (*bool*) – Whether to send notifications to the users that have fixed a broken build (default false)
- **group-targets** (*list*) – List of group targets to notify
- **individual-targets** (*list*) – List of individual targets to notify
- **strategy** (*dict*) – When to send notifications (default all)

  strategy values

  - **all** – Always

- **failure** – On any failure
- **failure-fixed** – On failure and fixes
- **new-failure-fixed** – On new failure and fixes
- **change** – Only on state change

- **message** (*dict*) – Channel notification message (default summary-scm)

  message values

  - **summary-scm** – Summary + SCM changes
  - **summary** – Just summary
  - **summary-build** – Summary and build parameters
  - **summary-scm-fail** – Summary, SCM changes, and failed tests

Minimal Example:

```
publishers:
  - jabber
```

Full Example:

```
publishers:
  - jabber:
      notify-on-build-start: true
      notify-scm-committers: true
      notify-scm-culprits: true
      notify-upstream-committers: true
      notify-scm-fixers: true
      group-targets:
        - "foo-room@conference-2-fooserver.foo.com"
      individual-targets:
        - "foo-user@conference-2-fooserver.foo.com"
      strategy: new-failure-fixed
      message: summary
```

jacoco()

Generate a JaCoCo coverage report. Requires the Jenkins JaCoCo Plugin.

Parameters

- **exec-pattern** (*str*) – This is a file name pattern that can be used to locate the jacoco report files (default `**/**.exec`)
- **class-pattern** (*str*) – This is a file name pattern that can be used to locate class files (default `**/classes`)
- **source-pattern** (*str*) – This is a file name pattern that can be used to locate source files (default `**/src/main/java`)
- **source-inclusion-pattern** (*str*) – This is a file name pattern that can be used to include certain source files (default `**/*.java`)
- **update-build-status** (*bool*) – Update the build according to the results (default false)
- **inclusion-pattern** (*str*) – This is a file name pattern that can be used to include certain class files (default '')
- **exclusion-pattern** (*str*) – This is a file name pattern that can be used to exclude certain class files (default '')
- **targets** (*dict*) –

  targets

  (instruction, branch, complexity, line, method, class)

  - **healthy** (*int*): Healthy threshold (default 0)
  - **unhealthy** (*int*): Unhealthy threshold (default 0)

Minimal Example:

```yaml
publishers:
  - jacoco
```

Full Example:

```yaml
publishers:
  - jacoco:
      exec-pattern: '**/**.exec'
      class-pattern: '**/classes'
      source-pattern: '**/src/main/java'
      source-inclusion-pattern: '**/*.java,**/*.kt'
      update-build-status: true
      inclusion-pattern: '**/*.class'
      exclusion-pattern: '**/*Test*.class'
      targets:
        - instruction:
            healthy: 7
            unhealthy: 1
        - branch:
            healthy: 8
            unhealthy: 2
        - complexity:
            healthy: 9
            unhealthy: 3
        - line:
            healthy: 10
            unhealthy: 4
        - method:
            healthy: 11
            unhealthy: 5
        - class:
            healthy: 12
            unhealthy: 6
```

javadoc()

Publish Javadoc Requires the Jenkins Javadoc Plugin.

Parameters

- **directory** (*str*) – Directory relative to the root of the workspace, such as 'myproject/build/javadoc' (optional)
- **keep-all-successful** (*bool*) – When true, it will retain Javadoc for each successful build. This allows you to browse Javadoc for older builds, at the expense of additional disk space requirement. If false, it will only keep the latest Javadoc, so older Javadoc will be overwritten as new builds succeed. (default false)

Example:

```yaml
publishers:
  - javadoc:
      directory: myproject/build/javadoc
      keep-all-successful: true
```

jclouds()

JClouds Cloud Storage Settings provides a way to store artifacts on JClouds supported storage providers. Requires the Jenkins JClouds Plugin.

JClouds Cloud Storage Settings must be configured for the Jenkins instance.

Parameters

- **profile** (*str*) – preconfigured storage profile (required)
- **files** (*str*) – files to upload (regex) (required)
- **basedir** (*str*) – the source file path (relative to workspace, Optional)
- **container** (*str*) – the destination container name (required)
- **hierarchy** (*bool*) – keep hierarchy (default false)

Example:

```
publishers:
  - jclouds:
      profile: hp
      files: '*.tar.gz'
      container: jenkins
      basedir: test base dir
```

jdepend()

Publish jdepend report Requires the JDepend Plugin.

Parameters

file (*str*) – path to jdepend file (required)

Example:

```
publishers:
  - jdepend:
      file: build/jdepend/main.xml
```

jira()

Update relevant JIRA issues Requires the Jenkins JIRA Plugin.

Example:

```
publishers:
  - jira
```

jms-messaging()

The JMS Messaging Plugin provides the following functionality:

- A build trigger to submit jenkins jobs upon receipt of a matching message.
- A builder that may be used to submit a message to the topic upon the completion of a job
- A post-build action that may be used to submit a message to the topic upon the completion of a job

JMS Messaging provider types supported:

- ActiveMQ
- FedMsg

Requires the Jenkins JMS Messaging Plugin Pipeline Plugin.

Parameters

- **override-topic** (*str*) – If you need to override the default topic. (default '')
- **provider-name** (*str*) – Name of message provider setup in the global config. (default '')
- **msg-type** (*str*) – A message type (default 'CodeQualityChecksDone')
- **msg-props** (*str*) – Message header to publish. (default '')
- **msg-content** (*str*) – Message body to publish. (default '')

Full Example:

```
publishers:
  - jms-messaging:
```

(continues on next page)

(continued from previous page)

```
      override-topic: org.centos.stage.ci.pipeline.compose.complete
      provider-name: fedmsg
      msg-type: Custom
      msg-props: |
        topic=org.centos.prod.ci.pipeline.compose.complete
        username=fedora-atomic
      msg-content: |
        {
          "build_url": "${BUILD_URL}",
          "compose_url": "<full-url-to-compose>",
          "build_id": "${BUILD_ID}",
          "ref": "fedora/rawhide/${basearch}/atomic-host",
          "rev": "<sha of the commit from dist-git>",
          "namespace": "rpms",
          "repo": "php-simplepie",
          "status": "<success/failure/aborted>",
          "test_guidance": "<comma-separated-list-of-test-suites-to-run>"}
```

Minimal Example:

```
publishers:
  - jms-messaging:
      provider-name: fedmsg
      msg-type: CodeQualityChecksDone
      msg-props: test
      msg-content: test
```

join-trigger()

Trigger a job after all the immediate downstream jobs have completed. Requires the Jenkins Join Plugin.

Parameters

- **projects** (*list*) – list of projects to trigger
- **publishers** (*list*) – list of triggers from publishers module that defines projects that need to be triggered
- **threshold** (*str*) – result threshold to trigger jobs (optional). Valid values are "success", "unstable", "failure", and "aborted".
- **even-if-unstable** (*bool*) – if true jobs will trigger even if some downstream jobs are marked as unstable (default false) (DEPRECATED)

Example:

```
publishers:
  - join-trigger:
      projects:
        - project-one
        - project-two
      threshold: unstable
      publishers:
        - trigger-parameterized-builds:
          - project: archive
            current-parameters: true
            trigger-from-child-projects: true
            trigger-with-no-params: true
          - project: cleanup
            current-parameters: true
```

(continues on next page)

(continued from previous page)

```
            trigger-with-no-params: false
```

junit()

Publish JUnit test results. Requires the Jenkins JUnit Plugin.

Parameters

- **results** (*str*) – results filename (required)
- **keep-long-stdio** (*bool*) – Retain long standard output/error in test results (default true).
- **health-scale-factor** (*float*) – Amplification factor to apply to test failures when computing the test result contribution to the build health score. (default 1.0)
- **allow-empty-results** (*bool*) – Do not fail the build on empty test results (default false)
- **skip-publishing-checks** (*bool*) – Do not publish issues to SCM provider platforms (default false)
- **skip-marking-build-unstable** (*bool*) – Do not mark build as unstable on test failure (default false)
- **test-stability** (*bool*) – Add historical information about test results stability (default false). Requires the Jenkins Test stability Plugin.
- **claim-build** (*bool*) – Allow claiming of failed tests (default false) Requires the Jenkins Claim Plugin.
- **measurement-plots** (*bool*) – Create measurement plots (default false) Requires the Jenkins Measurement Plots Plugin.
- **flaky-test-reports** (*bool*) – Publish flaky test reports (default false). Requires the Jenkins Flaky Test Handler Plugin.
- **junit-attachments** (*bool*) – Publish test attachments (default false). Requires the Jenkins JUnit Attachments Plugin.

Minimal example using defaults:

```
publishers:
- junit:
    results: nosetests.xml
```

Full example:

```
publishers:
- junit:
    results: nosetests-example.xml
    keep-long-stdio: false
    health-scale-factor: 2.0
    allow-empty-results: true
    skip-publishing-checks: true
    skip-marking-build-unstable: true
    test-stability: true
    claim-build: true
    measurement-plots: true
    flaky-test-reports: true
    junit-attachments: true
```

logparser()

Requires the Jenkins Log Parser Plugin.

Parameters

- **parse-rules** (*str*) – full path to parse rules (default '')
- **use-project-rules** (*bool*) – use project rules instead of global (default true)

- **unstable-on-warning** (*bool*) – mark build unstable on warning (default false)
- **fail-on-error** (*bool*) – mark build failed on error (default false)
- **show-graphs** (*bool*) – show parser trend graphs (default true)

Minimal Example:

```
publishers:
  - logparser:
      parse-rules: "project-log-parser-rules.txt"
```

Full Example:

```
publishers:
  - logparser:
      use-project-rules: false
      parse-rules: "/path/to/global-rules"
      unstable-on-warning: true
      fail-on-error: true
      show-graphs: false
```

logstash()

Send job's console log to Logstash for processing and analyis of your job data. Also stores test metrics from Junit. Requires the Jenkins Logstash Plugin.

Parameters

- **max-lines** (*int*) – The maximum number of log lines to send to Logstash. (default 1000)
- **fail-build** (*bool*) – Mark build as failed if this step fails. (default false)

Minimal Example:

```
publishers:
  - logstash
```

Full Example:

```
publishers:
  - logstash:
      max-lines: 2000
      fail-build: true
```

maven-deploy()

Deploy artifacts to Maven repository.

Parameters

- **id** (*str*) – Repository ID
- **url** (*str*) – Repository URL (optional)
- **unique-version** (*bool*) – Assign unique versions to snapshots (default true)
- **deploy-unstable** (*bool*) – Deploy even if the build is unstable (default false)
- **release-env-var** (*str*) – If the given variable name is set to "true", the deploy steps are skipped. (optional)

Example:

```
publishers:
  - maven-deploy:
      id: example
      url: http://repo.example.com/maven2/
      unique-version: true
```

(continues on next page)

(continued from previous page)

```
    deploy-unstable: false
    release-env-var: TIMER
```

mqtt()

This plugin lets you send build notifications to a MQTT message queue. Requires the MQTT Notification Plugin.

Parameters

- **broker-url** (*str*) – the broker URL, as protocol://address:port (required)
- **credentials-id** (*str*) – credentials to use to connect to the broker (optional)
- **topic** (*str*) – the message topic (default "jenkins/$PROJECT_URL")
- **message** (*str*) – the message itself (default "$BUILD_RESULT")
- **qos** (*str*) – one of AT_MOST_ONCE, AT_LEAST_ONCE, or EXACTLY_ONCE (default AT_MOST_ONCE)
- **retain-message** (*bool*) – whether to resend message or not when a new client connects (default false)

Minimal Example:

```
publishers:
  - mqtt:
      broker-url: tcp://localhost:1883
```

Full Example:

```
publishers:
  - mqtt:
      broker-url: tcp://localhost:1883
      topic: hello
      message: world
      qos: EXACTLY_ONCE
      retain-message: true
      credentials-id: abcde
```

naginator()

Automatically reschedule a build after a build failure Requires the Jenkins Naginator Plugin.

Parameters

- **rerun-unstable-builds** (*bool*) – Rerun build for unstable builds as well as failures (default false)
- **rerun-matrix-part** (*bool*) – Rerun build only for failed parts on the matrix (>=1.12) (default false)
- **fixed-delay** (*int*) – Fixed delay in seconds before retrying build (cannot be used with progressive-delay-increment or progressive-delay-maximum. This is the default delay type. (default 0)
- **progressive-delay-increment** (*int*) – Progressive delay in seconds before retrying build increment (cannot be used when fixed-delay is being used) (default 0)
- **progressive-delay-maximum** (*int*) – Progressive delay in seconds before retrying maximum delay (cannot be used when fixed-delay is being used) (default 0)
- **max-failed-builds** (*int*) – Maximum number of successive failed builds (default 0)
- **regular-expression** (*str*) – Only rerun build if regular expression is found in output (default '')

Example:

```
publishers:
  - naginator:
```

(continues on next page)

(continued from previous page)

```
    rerun-unstable-builds: true
    rerun-matrix-part: true
    progressive-delay-increment: 5
    progressive-delay-maximum: 15
    max-failed-builds: 6
    regular-expression: "foo"
```

openshift-build-canceller()

This action is intended to provide cleanup for a Jenkins job which failed because a build is hung (instead of terminating with a failure code); this step will allow you to perform the equivalent of a oc cancel-build for the provided build config; any builds under that build config which are not previously terminated (either successfully or unsuccessfully) or cancelled will be cancelled.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **bld-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default 'frontend')
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
publishers:
  - openshift-build-canceller:
      api-url: https://openshift.example.local.url/
      bld-cfg: front
      namespace: test-build
      auth-token: ose-key-canceller1
      verbose: true
```

Minimal Example:

```
publishers:
  - openshift-build-canceller
```

openshift-deploy-canceller()

This action is intended to provide cleanup for any OpenShift deployments left running when the Job completes; this step will allow you to perform the equivalent of a oc deploy –cancel for the provided deployment config.

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **api-url** (*str*) – this would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default 'https://openshift.default.svc.cluster.local')
- **dep-cfg** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on (default frontend)
- **namespace** (*str*) – If you run *oc get bc* for the project listed in "namespace", that is the value you want to put here. (default 'test')
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')

- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
publishers:
  - openshift-deploy-canceller:
      api-url: https://openshift.example.local.url/
      dep-cfg: front
      namespace: test6
      auth-token: ose-key-dep-canceller1
      verbose: true
```

Minimal Example:

```
publishers:
  - openshift-deploy-canceller
```

opsgenie()

OpsGenie notification on build completion, Requires the OpsGenie Notifier Plugin.

Parameters

- **enable-sending-alerts** (*bool*) – Send alerts to opsgenie. (default false)
- **notify-build-start** (*bool*) – Send a notification when the build starts. (default false)
- **api-key** (*str*) – This token is used to verify requests between OpsGenie and Jenkins. You can copy this key from your OpsGenie-Jenkins Integration page. (default '')
- **tags** (*str*) – Comma-separated list of tags you want to add on alert. (default '')
- **teams** (*str*) – Comma-separated list of teams that get notified from alert. (default '')
- **priority** (*str*) – Set the priority of the alert that's going to be created at OpsGenie, if job's build fails. (default 'P3')
- **build-starts-alerts-priority** (*str*) – Set the priority of the build started alert that's going to be created at OpsGenie. (default 'P3')
- **api-url** (*str*) – Api url that collects the webhook. (default '')

Minimal example:

```
publishers:
- opsgenie
```

Full Example:

```
publishers:
- opsgenie:
    enable-sending-alerts: true
    notify-build-start: true
    api-key: "test api key"
    priority: "P1"
    build-starts-alerts-priority: "P2"
    api-url: "another-opsgenie-instance.com"
    tags: "a,b,c"
    teams: "team a, team b"
```

packer()

This plugin allows for a job to publish an image generated Packer Requires the Jenkins Packer Plugin.

Parameters

- **name** (*str*) – Name of the packer installation (required)
- **json-template** (*str*) – Path to a Packer JSON template file (default '')

- **json-template-text** (*str*) – Text of Packer JSON template (default '')
- **add-params** (*str*) – Specify which additional parameters to pass to packer (default '')
- **use-debug** (*bool*) – adds -debug argument when packer executes (default false)
- **change-dir** (*str*) – If set, the current directory will be changed to this before starting packer (default '')
- **template-mode** (*str*) – Packer template option used (default global)

  template-mode values
  - **global**
  - **file**
  - **text**
- **file-entries** (*list*) – File entries for the packer configuration (default [])
- **variable-name** (*str*) – Variable name for a file to used in the configuration JSON (default '')
- **contents** (*str*) – File contents of the configuration JSON (default '')

Minimal Example:

```
publishers:
  - packer:
      name: test name
```

Full Example:

```
publishers:
  - packer:
      name: test name
      json-template: test template
      json-template-text: test template text
      add-params: additional params
      use-debug: true
      change-dir: change to directory
      template-mode: global
      file-entries:
        - files:
            variable-name: test var
            contents: test content
        - files:
            variable-name: test var 2
            contents: test content 2
```

performance()

Publish performance test results from jmeter and junit. Requires the Jenkins Performance Plugin.

Parameters

- **failed-threshold** (*int*) – Specify the error percentage threshold that set the build failed. A negative value means don't use this threshold (default 0)
- **unstable-threshold** (*int*) – Specify the error percentage threshold that set the build unstable. A negative value means don't use this threshold (default 0)
- **unstable-response-time-threshold** (*str*) – Average response time threshold (default '')
- **failed-threshold-positive** (*float*) – Maximum failed percentage for build comparison (default 0.0)
- **failed-threshold-negative** (*float*) – Minimum failed percentage for build comparison (default 0.0)
- **unstable-threshold-positive** (*float*) – Maximum unstable percentage for build comparison (default 0.0)

- **unstable-threshold-negative** (*float*) – Minimum unstable percentage for build comparison (default 0.0)
- **nth-build-number** (*int*) – Build number for build comparison (default 0)
- **mode-relative-thresholds** (*bool*) – Relative threshold mode (default false)
- **config-type** (*str*) – Compare based on (default 'ART')
  config-type values
  - **ART** – Average Response Time
  - **MRT** – Median Response Time
  - **PRT** – Percentile Response Time
- **mode-of-threshold** (*bool*) – Mode of threshold, true for relative threshold and false for error threshold (default false)
- **fail-build** (*bool*) – Fail build when result files are not present (default false)
- **compare-build-previous** (*bool*) – Compare with previous build (default false)
- **mode-performance-per-test-case** (*bool*) – Performance Per Test Case Mode (default true)
- **mode-thoughput** (*bool*) – Show Throughput Chart (default false)
- **report** (*dict*) –
  (jmeter or junit)
  (*dict* or *str*): Specify a custom report file (optional; jmeter default **/*.jtl, junit default* */TEST-*.xml)

Minimal Example:

```
publishers:
  - performance
```

Full Example:

```
publishers:
  - performance:
      failed-threshold: 85
      unstable-threshold: -1
      unstable-response-time-threshold: "JMeterResultsOrders.jtl:2000"
      failed-threshold-positive: 90.0
      failed-threshold-negative: 10.0
      unstable-threshold-positive: 80.0
      unstable-threshold-negative: 20.0
      nth-build-number: 10
      mode-relative-thresholds: true
      config-type: "PRT"
      mode-of-threshold: true
      fail-build: true
      compare-build-previous: true
      mode-performance-per-test-case: false
      mode-thoughput: true
      report:
        - jmeter: "/special/file.jtl"
        - junit: "/special/file.xml"
        - jmeter
        - junit
```

phabricator()

Integrate with Phabricator

Requires the Jenkins Phabricator Plugin.

Parameters

- **comment-on-success** (*bool*) – Post a *comment* when the build succeeds. (optional)
- **uberalls-enabled** (*bool*) – Integrate with uberalls. (optional)
- **comment-file** (*str*) – Include contents of given file if commenting is enabled. (optional)
- **comment-size** (*int*) – Maximum comment character length. (optional)
- **comment-with-console-link-on-failure** (*bool*) – Post a *comment* when the build fails. (optional)

Example:

```
publishers:
  - phabricator:
      comment-on-success: false
      uberalls-enabled: false
      comment-with-console-link-on-failure: false
```

pipeline()

Specify a downstream project in a pipeline. Requires the Jenkins Build Pipeline Plugin.

Use of the *node-label-name* or *node-label* parameters requires the Jenkins NodeLabel Parameter Plugin. Note: 'node-parameters' overrides the Node that the triggered project is tied to.

Parameters

- **projects** (*list*) – list the jobs to trigger, will generate comma-separated string containing the named jobs.
- **predefined-parameters** (*str*) – parameters to pass to the other job (optional)
- **current-parameters** (*bool*) – Whether to include the parameters passed to the current build to the triggered job (optional)
- **node-parameters** (*bool*) – Use the same Node for the triggered builds that was used for this build. (optional)
- **svn-revision** (*bool*) – Pass svn revision to the triggered job (optional)
- **include-upstream** (*bool*) – Include/pass through Upstream SVN Revisons. Only valid when 'svn-revision' is true. (default false)
- **git-revision** (*dict*) – Passes git revision to the triggered job (optional).
  - **combine-queued-commits** (bool): Whether to combine queued git hashes or not (default false)
- **boolean-parameters** (*dict*) – Pass boolean parameters to the downstream jobs. Specify the name and boolean value mapping of the parameters. (optional)
- **property-file** (*str*) – Use properties from file (optional)
- **fail-on-missing** (*bool*) – Blocks the triggering of the downstream jobs if any of the property files are not found in the workspace. Only valid when 'property-file' is specified. (default false)
- **file-encoding** (*str*) – Encoding of contents of the files. If not specified, default encoding of the platform is used. Only valid when 'property-file' is specified. (optional)
- **restrict-matrix-project** (*str*) – Filter that restricts the subset of the combinations that the downstream project will run (optional)

Example:

```
publishers:
  - pipeline:
      project: test_project
      current-parameters: true
      predefined-parameters: foo=bar
```

```
publishers:
  - pipeline:
```

(continues on next page)

(continued from previous page)

```
      projects:
        - test_project
      predefined-parameters: BUILD_NUM=${BUILD_NUMBER}
      current-parameters: true
      property-file: vars.txt
      git-revision:
        combine-queued-commits: true
      fail-on-missing: true
      file-encoding: UTF-8
      boolean-parameters:
          p1: true
          p2: false
      svn-revision: true
      include-upstream: true
```

You can build pipeline jobs that are re-usable in different pipelines by using a *Job Template* to define the pipeline jobs, and variable substitution to specify the name of the downstream job in the pipeline. Job-specific substitutions are useful here (see *Project*).

See 'samples/pipeline.yaml' for an example pipeline implementation.

plot()

Plot provides generic plotting (or graphing).

Requires the Jenkins Plot Plugin.

Parameters

- **title** (*str*) – title for the graph (default '')
- **yaxis** (*str*) – title of Y axis (default '')
- **width** (*int*) – the width of the plot in pixels (default 750)
- **height** (*int*) – the height of the plot in pixels (default 450)
- **group** (*str*) – name of the group to which the plot belongs (required)
- **num-builds** (*int*) – number of builds to plot across (default plot all builds)
- **style** (*str*) – Specifies the graph style of the plot Can be: area, bar, bar3d, line, line3d, stackedArea, stackedbar, stackedbar3d, waterfall (default 'line')
- **use-description** (*bool*) – When false, the X-axis labels are formed using build numbers and dates, and the corresponding tooltips contain the build descriptions. When enabled, the contents of the labels and tooltips are swapped, with the descriptions used as X-axis labels and the build number and date used for tooltips. (default false)
- **exclude-zero-yaxis** (*bool*) – When false, Y-axis contains the value zero even if it is not included in the data series. When true, the value zero is not automatically included. (default false)
- **logarithmic-yaxis** (*bool*) – When true, the Y-axis will use a logarithmic scale. By default, the Y-axis uses a linear scale. (default false)
- **keep-records** (*bool*) – When true, show all builds up to 'Number of builds to include'. (default false)
- **csv-file-name** (*str*) – Use for choosing the file name in which the data will be persisted. If none specified and random name is generated as done in the Jenkins Plot plugin. (default random generated .csv filename, same behaviour as the Jenkins Plot plugin)
- **series** (*list*) – list data series definitions

  Series

  - **file** (*str*) : files to include
  - **inclusion-flag** filtering mode for CSV files. Possible values are:

        * **off** (default)
        * **include-by-string**
        * **exclude-by-string**
        * **include-by-column**
        * **exclude-by-column**
    - **exclude** (*str*) : exclude pattern for CSV file.
    - **url** (*str*) : for 'csv' and 'xml' file types used when you click on a point (default empty)
    - **display-table** (*bool*) : for 'csv' file type if true, original CSV will be shown above plot (default false)
    - **label** (*str*) : used by 'properties' file type Specifies the legend label for this data series. (default empty)
    - **format** (*str*) : Type of file where we get datas. Can be: properties, csv, xml
    - **xpath-type** (*str*) : The result type of the expression must be supplied due to limitations in the java.xml.xpath parsing. The result can be: node, nodeset, boolean, string, or number. Strings and numbers will be converted to double. Boolean will be converted to 1 for true, and 0 for false. (default 'node')
    - **xpath** (*str*) : used by 'xml' file type Xpath which selects the values that should be plotted.

Minimal Example:

```
publishers:
  - plot:
      - yaxis: ''
        group: 'bench'
        series:
            - file: 'data.csv'
              format: 'csv'
```

Full Example:

```
publishers:
  - plot:
    - title: MyPlot
      yaxis: Y
      width: 900
      height: 1000
      csv-file-name: myplot.csv
      group: PlotGroup
      num-builds: '1'
      style: line
      exclude-zero-yaxis: true
      logarithmic-yaxis: true
      use-description: true
      keep-records: true
      series:
        - file: graph-me-second.properties
          label: MyLabel
          format: properties
        - file: graph-me-first.csv
          url: 'http://srv1'
          inclusion-flag: 'include-by-string'
```

(continues on next page)

(continued from previous page)

```
        exclude: 'Column 1,Column 2,Column 3'
        display-table: true
        format: csv
    - title: MyPlot2
      yaxis: Y
      csv-file-name: myplot2.csv
      group: PlotGroup
      style: bar
      use-description: false
      series:
        - file: graph-me-third.xml
          url: 'http://srv2'
          format: xml
          xpath-type: 'string'
          xpath: '/*'
```

pmd()

Publish trend reports with PMD.

Requires the Jenkins PMD Plugin (https://github.com/jenkinsci/pmd-plugin).

The PMD component accepts a dictionary with the following values:

Parameters

- **pattern** (*str*) – Report filename pattern (optional)
- **can-run-on-failed** (*bool*) – Also runs for failed builds, instead of just stable or unstable builds (default false)
- **should-detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings (default false)
- **healthy** (*int*) – Sunny threshold (optional)
- **unhealthy** (*int*) – Stormy threshold (optional)
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high', defaulted to 'low')
- **thresholds** (*dict*) – Mark build as failed or unstable if the number of errors exceeds a threshold. (optional)

  thresholds

  - **unstable (*dict*)**

    unstable

    - **total-all** (*int*)
    - **total-high** (*int*)
    - **total-normal** (*int*)
    - **total-low** (*int*)
    - **new-all** (*int*)
    - **new-high** (*int*)
    - **new-normal** (*int*)
    - **new-low** (*int*)
  - **failed (*dict*)**

    failed

    - **total-all** (*int*)

        * **total-high** (*int*)
        * **total-normal** (*int*)
        * **total-low** (*int*)
        * **new-all** (*int*)
        * **new-high** (*int*)
        * **new-normal** (*int*)
        * **new-low** (*int*)
    - **default-encoding** (*str*) – Encoding for parsing or showing files (optional)
    - **do-not-resolve-relative-paths** (*bool*) – (default false)
    - **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
    - **use-previous-build-as-reference** (*bool*) – determines whether to always use the previous build as the reference build (default false)
    - **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
    - **use-delta-values** (*bool*) – If set then the number of new warnings is calculated by subtracting the total number of warnings of the current build from the reference build. (default false)

Example:

```
publishers:
 - pmd:
    pattern: '**/pmd-result.xml'
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    thresholds:
        unstable:
            total-high: 10
        failed:
            total-high: 1
```

Full example:

```
publishers:
 - pmd:
    pattern: '**/pmd-result.xml'
    can-run-on-failed: true
    should-detect-modules: true
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    thresholds:
        unstable:
            total-all: 90
            total-high: 80
            total-normal: 70
            total-low: 60
        failed:
            total-all: 90
```

(continues on next page)

(continued from previous page)

```
            total-high: 80
            total-normal: 70
            total-low: 60
    default-encoding: 'utf-8'
```

post-tasks()

Adds support to post build task plugin

Requires the Jenkins Post Build Task plugin.

Parameters

- **task** (*dict*) – Post build task definition
- **task[matches]** (*list*) – list of matches when to run the task
- **task[matches][*]** (*dict*) – match definition
- **task[matches][*][log-text]** (*str*) – text to match against the log
- **task[matches][*][operator]** (*str*) – operator to apply with the next match

  task[matches][*][operator] values (default 'AND')

  - **AND**
  - **OR**
- **task[escalate-status]** (*bool*) – Escalate the task status to the job (default 'false')
- **task[run-if-job-successful]** (*bool*) – Run only if the job was successful (default 'false')
- **task[script]** (*str*) – Shell script to run (default '')

Example:

```
publishers:
  - post-tasks:
    - matches:
      - log-text: line to match
        operator: AND
      - log-text: line to match
        operator: OR
      - log-text: line to match
        operator: AND
      escalate-status: true
      run-if-job-successful: true
      script: |
        echo "Here goes the task script"
```

postbuildscript()

Executes additional builders, script or Groovy after the build is complete.

Requires the Jenkins Post Build Script plugin.

Parameters

- **generic-script** (*list*) – Series of Batch/Shell scripts to to run

  generic-script

  - **file-path** (*str*) - Path to Batch/Shell scripts
  - **role** (*str*) - Execute scripts on. One of MASTER / SLAVE / BOTH. (default 'BOTH')
  - **build-on** (*list*) - Build statuses which trigger the scripts. Valid options: SUCCESS / UNSTABLE / FAILURE / NOT_BUILT / ABORTED (default 'SUCCESS')
  - ** execute-on (*str*) - For matrix projects, scripts can be run after each axis is built (*axes*), after all axis of the matrix are built

(*matrix*) or after each axis AND the matrix are built (*both*). (default *both*)

- **groovy-script** (*list*) – Paths to Groovy scripts
  groovy-script
  - **file-path** (*str*) - Path to Groovy scripts
  - **role** (*str*) - Execute scripts on. One of MASTER / SLAVE / BOTH. (default 'BOTH')
  - **build-on** (*list*) - Build statuses which trigger the scripts. Valid options: SUCCESS / UNSTABLE / FAILURE / NOT_BUILT / ABORTED (default 'SUCCESS')
  - ** execute-on (*str*) - For matrix projects, scripts can be run after each axis is built (*axes*), after all axis of the matrix are built (*matrix*) or after each axis AND the matrix are built (*both*). (default *both*)
- **groovy** (*list*) – Inline Groovy
  groovy
  - **content** (*str*) - Inline Groovy script.
  - **role** (*str*) - Execute scripts on. One of MASTER / SLAVE / BOTH. (default 'BOTH')
  - **build-on** (*list*) - Build statuses which trigger the scripts. Valid options: SUCCESS / UNSTABLE / FAILURE / NOT_BUILT / ABORTED (default 'SUCCESS')
  - ** execute-on (*str*) - For matrix projects, scripts can be run after each axis is built (*axes*), after all axis of the matrix are built (*matrix*) or after each axis AND the matrix are built (*both*). (default *both*)
- **builders** (*list*) – Execute any number of supported Jenkins builders.
  builders
  - **build-steps** (*str*) - Any supported builders, see *Builders*.
  - **role** (*str*) - Execute scripts on. One of MASTER / SLAVE / BOTH. (default 'BOTH')
  - **build-on** (*list*) - Build statuses which trigger the scripts. Valid options: SUCCESS / UNSTABLE / FAILURE / NOT_BUILT / ABORTED (default 'SUCCESS')
  - ** execute-on (*str*) - For matrix projects, scripts can be run after each axis is built (*axes*), after all axis of the matrix are built (*matrix*) or after each axis AND the matrix are built (*both*). (default *both*)
- **mark-unstable-if-failed** (*bool*) – Build will be marked unstable if job will be successfully completed but publishing script will return non zero exit code (default false)

Deprecated Options for versions < 2.0 of plugin:

Parameters

- **onsuccess** (*bool*) – Deprecated, replaced with script-only-if-succeeded
- **script-only-if-succeeded** (*bool*) – Scripts and builders are run only if the build succeeded (default true)
- **onfailure** (*bool*) – Deprecated, replaced with script-only-if-failed
- **script-only-if-failed** (*bool*) – Scripts and builders are run only if the build failed (default false)
- **execute-on** (*str*) – For matrix projects, scripts can be run after each axis is built (*axes*), after all axis of the matrix are built (*matrix*) or after each axis AND the matrix are built (*both*).

The *script-only-if-succeeded* and *bool script-only-if-failed* options are confusing. If you want the post build to always run regardless of the build status, you should set them both to *false*.

Minimal Example:

```
publishers:
    - postbuildscript
```

Full Example:

```
publishers:
    - postbuildscript:
        mark-unstable-if-failed: true
        generic-script:
            - file-path: '/fakepath/generic'
              role: MASTER
              build-on:
                  - SUCCESS
                  - UNSTABLE
            - file-path: '/fakepath/generic-two'
              role: SLAVE
              build-on:
                  - NOT_BUILT
                  - ABORTED
                  - FAILURE
              execute-on: matrix
        groovy-script:
            - file-path: '/fakepath/groovy'
              role: MASTER
              build-on:
                  - SUCCESS
                  - UNSTABLE
              execute-on: axes
            - file-path: '/fakepath/groovy-too'
              role: SLAVE
              build-on:
                  - NOT_BUILT
                  - ABORTED
                  - FAILURE
        groovy:
            - role: MASTER
              build-on:
                  - SUCCESS
                  - UNSTABLE
              execute-on: matrix
              content: 'println "Hello world!"'
            - role: SLAVE
              build-on:
                  - NOT_BUILT
                  - ABORTED
                  - FAILURE
              content: |
                  println "Hello world!"
                  println "Multi-line script"

        builders:
            - role: MASTER
```

(continues on next page)

(continued from previous page)

```
            build-on:
                - SUCCESS
                - UNSTABLE
            execute-on: axes
            build-steps:
                - shell: 'echo "Hello world!"'
          - role: SLAVE
            build-on:
                - NOT_BUILT
                - ABORTED
                - FAILURE
            execute-on: both
            build-steps:
                - shell: 'echo "Hello world!"'
                - shell: 'echo "Goodbye world!"'
```

Example(s) versions < 2.0:

```
publishers:
    - postbuildscript:
        generic-script:
            - '/tmp/one.sh'
            - '/tmp/two.sh'
        groovy-script:
            - '/tmp/one.groovy'
            - '/tmp/two.groovy'
        groovy:
            - "/** This is some inlined groovy */"
            - "/** Some more inlined groovy */"
        script-only-if-succeeded: False
        script-only-if-failed: True
        mark-unstable-if-failed: True
```

You can also execute *builders*:

```
publishers:
    - postbuildscript:
        builders:
            - shell: 'echo "Shell execution"'
            - ant: 'ant_target'
```

Run once after the whole matrix (all axes) is built:

```
publishers:
    - postbuildscript:
        execute-on: 'matrix'
        builders:
            - shell: 'echo "Shell execution"'
```

publishers-from()

Use publishers from another project. Requires the Jenkins Template Project Plugin.

Parameters

project-name (*str*) – The name of the other project.

Example:

```
publishers:
  - publishers-from:
      project-name: base-build
```

rich-text-publisher()

This plugin puts custom rich text message to the Build pages and Job main page.

Requires the Jenkins Rich Text Publisher Plugin.

Parameters

- **stable-text** (*str*) – The stable text (required)
- **unstable-text** (*str*) – The unstable text if different from stable (default '')
- **unstable-as-stable** (*bool*) – The same text block is used for stable and unstable builds (default true)
- **failed-text** (*str*) – The failed text if different from stable (default '')
- **failed-as-stable** (*bool*) – The same text block is used for stable and failed builds (default true)
- **parser-name** (*str*) – HTML, Confluence or WikiText (default 'WikiText')

Minimal Example:

```
publishers:
    - rich-text-publisher:
        stable-text: testing
```

Full Example:

```
publishers:
    - rich-text-publisher:
        stable-text: the stable text
        unstable-text: the unstable text
        failed-text: the failed text
        unstable-as-stable: false
        failed-as-stable: false
        parser-name: HTML
```

robot()

Adds support for the Robot Framework Plugin

Requires the Jenkins Robot Framework Plugin.

Parameters

- **output-path** (*str*) – Path to directory containing robot xml and html files relative to build workspace. (required)
- **log-file-link** (*str*) – Name of log or report file to be linked on jobs front page (default '')
- **report-html** (*str*) – Name of the html file containing robot test report (default 'report.html')
- **log-html** (*str*) – Name of the html file containing detailed robot test log (default 'log.html')
- **output-xml** (*str*) – Name of the xml file containing robot output (default 'output.xml')
- **pass-threshold** (*str*) – Minimum percentage of passed tests to consider the build successful (default 0.0)
- **unstable-threshold** (*str*) – Minimum percentage of passed test to consider the build as not failed (default 0.0)

- **only-critical** (*bool*) – Take only critical tests into account when checking the thresholds (default true)
- **other-files** (*list*) – list other files to archive (default '')
- **archive-output-xml** (*bool*) – Archive output xml file to server (default true)
- **enable-cache** (*bool*) – Enable cache for test results (default true)

Minimal Example:

```
publishers:
  - robot:
      output-path: reports/robot
```

Full Example:

```
publishers:
  - robot:
      output-path: reports/robot
      log-file-link: report.html
      report-html: custom-report.html
      log-html: custom-log.html
      output-xml: custom-output.xml
      pass-threshold: 80.0
      unstable-threshold: 60.0
      only-critical: false
      enable-cache: false
      other-files:
        - extra-file1.html
        - extra-file2.txt
      archive-output-xml: false
```

rocket()

RocketChat notification on build completion, Requires the RocketChat Notifier Plugin.

Parameters

- **channel** (*str*) – Comma separated list of rooms (e.g. #project) or persons (e.g. @john)
- **abort** (*bool*) – Notify abort (default false)
- **start** (*bool*) – Notify start (default false)
- **success** (*bool*) – Notify success (default false)
- **failure** (*bool*) – Notify failure (default false)
- **repeated-failure** (*bool*) – Notify repeated failure (default false)
- **back-to-normal** (*bool*) – Notify back to normal (default false)
- **not-built** (*bool*) – Notify if not built (default false)
- **unstable** (*bool*) – Notify if unstable (default false)
- **webhook-token** (*str*) – Webhook token for posting messages. Overrides global authentication data and channel
- **commit-info** (*str*) – What commit information to include into notification message.
  commit-info values
  - **none**
  - **authors**
  - **authors-and-titles**
- **custom-message** (*str*) – Custom text message (default '')
- **trust-ssl** (*bool*) – Trust RocketChat server certificate, if checked, the SSL certificate will not be checked (default false)
- **build-server** (*str*) – Build Server URL
- **attachments** (*list*) – Optional list of attachments
  attachments

- **title** (*str*) Attachment title (required)
- **title-link** (*str*)
- **title-link-download** (*bool*)
- **color** (*str*)
- **text** (*str*)
- **collapsed** (*bool*)
- **message-link** (*str*)
- **author-name** (*str*)
- **author-link** (*str*)
- **author-icon** (*str*)
- **thumb** (*str*) Thumb URL
- **image** (*str*) Image URL
- **audio** (*str*) Audio URL
- **video** (*str*) Video URL

Minimal example using defaults:

```
publishers:
- rocket:
    channel: ''
```

Full example:

```
publishers:
- rocket:
    channel: '#channel,@user'
    abort: true
    back-to-normal: true
    failure: true
    not-built: true
    repeated-failure: true
    start: true
    success: true
    unstable: true
    trust-ssl: true
    build-server: 'http://localhost:8080/'
    webhook-token: 'non-secure-webhook-token'
    webhook-token-credential-id: 'secret-text-id'
    commit-info: 'authors-and-titles'
    include-custom-message: true
    include-test-log: true
    include-test-summary: true
    custom-message: 'Hello World!'
    raw-message: true
    attachments:
        - title: The Black
        - title: The Red
          color: red
        - title: The Blue
          color: blue
          text: The navy blue
        - title: The White
          title-link: title_link
          title-link-download: true
          message-link: message_link
```

(continues on next page)

(continued from previous page)

```
          color: white
          text: 'All&all'
          collapsed: true
          author-name: author_name
          author-link: author_link
          author-icon: author_icon
          thumb: 'http://hostname/thumb.png'
          image: 'http://hostname/image.jpg'
          audio: 'http://hostname/audio.mp3'
          video: 'http://hostname/video.avi'
```

ruby-metrics()

Rcov plugin parses rcov html report files and shows it in Jenkins with a trend graph.

Requires the Jenkins Ruby metrics plugin.

Parameters

- **report-dir** (*str*) – Relative path to the coverage report directory
- **targets** (*dict*) –

  targets

  (total-coverage, code-coverage)

  - **healthy** (*int*): Healthy threshold
  - **unhealthy** (*int*): Unhealthy threshold
  - **unstable** (*int*): Unstable threshold

Example:

```
publishers:
  - ruby-metrics:
      report-dir: "coverage/rcov"
      target:
        - total-coverage:
            healthy: 80
            unhealthy: 0
            unstable: 0
        - code-coverage:
            healthy: 80
            unhealthy: 0
            unstable: 0
```

rundeck()

Trigger a rundeck job when the build is complete.

Requires the Jenkins RunDeck Plugin.

Parameters

- **job-id** (*str*) – The RunDeck job identifier. (required) This could be: * ID example : “42” * UUID example : “2027ce89-7924-4ecf-a963-30090ada834f” * reference, in the format : “project:group/job”
- **options** (*str*) – List of options for the Rundeck job, in Java-Properties format: key=value (default “”)
- **node-filters** (*str*) – List of filters to optionally filter the nodes included by the job. (default “”)
- **tag** (*str*) – Used for on-demand job scheduling on rundeck: if a tag is specified, the job will only execute if the given tag is present in the SCM changelog. (default “”)
- **wait-for-rundeck** (*bool*) – If true Jenkins will wait for the job to complete, if false the job will be started and Jenkins will move on. (default false)

- **fail-the-build** (*bool*) – If true a RunDeck job failure will cause the Jenkins build to fail. (default false)

Example:

```
publishers:
 - rundeck:
    job-id: testproject:group/jobname
```

Full example:

```
publishers:
 - rundeck:
    job-id: testproject:group/jobname
    options: |
      STUFF_FOR_THE_JOB=stuff
      ANOTHER_VAR=more_stuff
    node-filters: dev
    tag: master
    wait-for-rundeck: true
    fail-the-build: true
```

s3()

Upload build artifacts to Amazon S3.

Requires the Jenkins S3 plugin.

Parameters

- **s3-profile** (*str*) – Globally-defined S3 profile to use
- **dont-wait-for-concurrent-builds** (*bool*) – Don't wait for completion of concurrent builds before publishing to S3 (default false)
- **entries** (*list*) –

  entries

  - **destination-bucket** (*str*) - Destination S3 bucket
  - **source-files** (*str*) - Source files (Ant glob syntax)
  - **storage-class** (*str*) - S3 storage class; one of "STANDARD" or "REDUCED_REDUNDANCY"
  - **bucket-region** (*str*) - S3 bucket region (capitalized with underscores)
  - **upload-on-failure** (*bool*) - Upload files even if the build failed (default false)
  - **upload-from-slave** (*bool*) - Perform the upload directly from the Jenkins slave rather than the master node. (default false)
  - **managed-artifacts** (*bool*) - Let Jenkins fully manage the published artifacts, similar to when artifacts are published to the Jenkins master. (default false)
  - **s3-encryption** (*bool*) - Use S3 AES-256 server side encryption support. (default false)
  - **flatten** (*bool*) - Ignore the directory structure of the artifacts in the source project and copy all matching artifacts directly into the specified bucket. (default false)
- **metadata-tags** (*list*) –

  metadata-tags

  - **key** Metadata key for files from this build. It will be prefixed by "x-amz-meta-" when uploaded to S3. Can contain macros (e.g. environment variables).

- **value** Metadata value associated with the key. Can contain macros.

Example:

```
publishers:
  - s3:
      s3-profile: banana
      dont-wait-for-concurrent-builds: true
      entries:
        - destination-bucket: herp-derp
          source-files: 'bargle_${BUILD_ID}.tgz'
          storage-class: STANDARD
          bucket-region: US_EAST_1
          upload-on-failure: false
          upload-from-slave: true
          managed-artifacts: true
          s3-encryption: true
          flatten: true
      metadata-tags:
        - key: warbl ${garbl}
          value: herp derp weevils
        - key: hurrdurr
          value: wharrgarbl blee ${FANCY_VARIABLE}
```

scan-build()

Publishes results from the Clang scan-build static analyzer.

The scan-build report has to be generated in the directory `${WORKSPACE}/clangScanBuildReports` for the publisher to find it.

Requires the Jenkins Clang Scan-Build Plugin.

Parameters

- **mark-unstable** (*bool*) – Mark build as unstable if the number of bugs exceeds a threshold (default false)
- **threshold** (*int*) – Threshold for marking builds as unstable (default 0)
- **exclude-paths** (*str*) – Comma separated paths to exclude from reports (>=1.5) (default '')
- **report-folder** (*str*) – Folder where generated reports are located (>=1.7) (default 'clangScanBuildReports')

Full Example:

```
publishers:
  - scan-build:
      mark-unstable: true
      threshold: 50
      exclude-paths: external-lib
      report-folder: scan-build-report
```

Minimal Example:

```
publishers:
  - scan-build
```

scoverage()

Publish scoverage results as a trend graph. Requires the Jenkins Scoverage Plugin.

Parameters

- **report-directory** (*str*) – This is a directory that specifies the locations where the xml scoverage report is generated (required)
- **report-file** (*str*) – This is a file name that is given to the xml scoverage report (required)

Example:

```
publishers:
  - scoverage:
      report-directory: target/scala-2.10/scoverage-report/
      report-file: scoverage.xml
```

scp()

Upload files via SCP Requires the Jenkins SCP Plugin.

When writing a publisher macro, it is important to keep in mind that Jenkins uses Ant's SCP Task via the Jenkins SCP Plugin which relies on FileSet and DirSet patterns. The relevant piece of documentation is excerpted below:

> Source points to files which will be uploaded. You can use ant includes syntax, eg. `folder/dist/*.jar`. Path is constructed from workspace root. Note that you cannot point files outside the workspace directory. For example providing: `../myfile.txt` won't work... Destination points to destination folder on remote site. It will be created if doesn't exists and relative to root repository path. You can define multiple blocks of source/destination pairs.

This means that absolute paths, e.g., `/var/log/**` will not work and will fail to compile. All paths need to be relative to the directory that the publisher runs and the paths have to be contained inside of that directory. The relative working directory is usually:

```
/home/jenkins/workspace/${JOB_NAME}
```

Parameters

- **site** (*str*) – name of the scp site (required)
- **target** (*str*) – destination directory (required)
- **source** (*str*) – source path specifier (default '')
- **keep-hierarchy** (*bool*) – keep the file hierarchy when uploading (default false)
- **copy-after-failure** (*bool*) – copy files even if the job fails (default false)
- **copy-console** (*bool*) – copy the console log (default false); if specified, omit 'source'

Example:

```
publishers:
  - scp:
      site: 'example.com'
      files:
        - target: 'dest/dir'
          source: 'base/source/dir/**'
          keep-hierarchy: true
          copy-after-failure: true
```

shining-panda()

Publish coverage.py results. Requires the Jenkins ShiningPanda Plugin.

Parameters

html-reports-directory (*str*) – path to coverage.py html results (optional)

Example:

```
publishers:
  - shining-panda:
      html-reports-directory: foo/bar/coveragepy_html_report/
```

sitemonitor()

This plugin checks the availability of an url.

It requires the sitemonitor plugin.

Parameters

sites (*list*) – List of URLs to check

Minimal Example:

```
publishers:
  - sitemonitor
```

Full Example:

```
publishers:
  - sitemonitor:
      sites:
        - url: http://foo.example.com
        - url: http://bar.example.com:8080/
```

slack()

Publisher that sends slack notifications on job events.

Requires the Jenkins Slack Plugin

When using Slack Plugin version < 2.0, Slack Plugin itself requires a publisher as well as properties please note that you have to create those too. When using Slack Plugin version >= 2.0, you should only configure the publisher.

For backward compatibility, the publisher needs to query version of the Slack Plugin. Hence the `query_plugins_info` parameter shouldn't be set to `False` in the `jenkins` section of the configuration file.

Parameters

- **team-domain** (*str*) – Your team's domain at slack. (default '')
- **auth-token** (*str*) – The integration token to be used when sending notifications. (default '')
- **auth-token-id** (*str*) – Allows credentials to be stored in Jenkins. (default '')
- **build-server-url** (*str*) – Specify the URL for your server installation. (default '/')
- **room** (*str*) – A comma separated list of rooms / channels to post the notifications to. (default '')
- **notify-start** (*bool*) – Send notification when the job starts (>=2.0). (default false)
- **notify-success** (*bool*) – Send notification on success (>=2.0). (default false)
- **notify-aborted** (*bool*) – Send notification when job is aborted (>=2.0). (default false)
- **notify-not-built** (*bool*) – Send notification when job set to NOT_BUILT status (>=2.0). (default false)
- **notify-unstable** (*bool*) – Send notification when job becomes unstable (>=2.0). (default false)
- **notify-failure** (*bool*) – Send notification when job fails for the first time (previous build was a success) (>=2.0). (default false)
- **notify-every-failure** (*bool*) – Send notification every time a job fails (>=2.23). (default false)
- **notify-back-to-normal** (*bool*) – Send notification when job is succeeding again after being unstable or failed (>=2.0). (default false)

- **notify-repeated-failure** (*bool*) – Send notification when job fails successively (previous build was also a failure) (>=2.0). (default false)
- **notify-regression** (*bool*) – Send notification when number of failed tests increased or the failed tests are different than previous build (>=2.2). (default false)
- **include-failed-tests** (*bool*) – includes all failed tests when some tests failed. does nothing if no failed tests were found (>=2.2). (default false)
- **include-test-summary** (*bool*) – Include the test summary (>=2.0). (default false)
- **commit-info-choice** (*str*) – What commit information to include into notification message, "NONE" includes nothing about commits, "AUTHORS" includes commit list with authors only, and "AUTHORS_AND_TITLES" includes commit list with authors and titles (>=2.0). (default "NONE")
- **include-custom-message** (*bool*) – Include a custom message into the notification (>=2.0). (default false)
- **custom-message** (*str*) – Custom message to be included for all statuses (>=2.0). (default '')
- **custom-message-success** (*str*) – Custom message for succesful builds (>=2.10). (default '')
- **custom-message-aborted** (*str*) – Custom message for aborted builds (>=2.10). (default '')
- **custom-message-not-built** (*str*) – Custom message for not-built (>=2.10). (default '')
- **custom-message-unstable** (*str*) – Custom message for unstable builds (>=2.10). (default '')
- **custom-message-failure** (*str*) – Custom message for failed builds (>=2.10). (default '')
- **auth-token-credential-id** (*str*) – The ID for the integration token from the Credentials plugin to be used to send notifications to Slack. (>=2.1) (default '')
- **bot-user** (*bool*) – This option indicates the token belongs to a bot user in Slack. (>=2.2) (default False)
- **base-url** (*str*) – Your Slack compatible Base URL. `bot-user` is not supported with Base URL. (>=2.2) (default '')

Example (version < 2.0):

```
publishers:
  - slack:
      room: '#builds'
      team-domain: 'teamname'
      auth-token: 'yourauthtoken'
      auth-token-id: 'yourauthtokenid'
```

Minimal example (version >= 2.0):

```
publishers:
  - slack
```

Full example (version >= 2.10):

```
publishers:
  - slack:
      team-domain: 'teamname'
      auth-token: 'yourauthtoken'
      auth-token-id: 'yourauthtokenid'
      build-server-url: 'http://localhost:8081'
      room: '#builds'
```

(continues on next page)

(continued from previous page)

```
    notify-start: True
    notify-success: True
    notify-aborted: True
    notify-not-built: True
    notify-unstable: True
    notify-failure: True
    notify-every-failure: True
    notify-back-to-normal: True
    notify-repeated-failure: True
    notify-regression: True
    include-test-summary: True
    include-failed-tests: True
    commit-info-choice: 'AUTHORS_AND_TITLES'
    include-custom-message: True
    custom-message: 'A custom message.'
    custom-message-success: 'A custom message for sucessful builds.'
    custom-message-aborted: 'A custom message for aborted builds.'
    custom-message-not-built: 'A custom message for not-built.'
    custom-message-unstable: 'A custom message for unstable builds.'
    custom-message-failure: 'A custom message for failed builds.'
    auth-token-credential-id: yourauthtoken
    bot-user: True
    base-url: https://hooks.slack.com/services/
```

sloccount()

Generates the trend report for SLOCCount

Requires the Jenkins SLOCCount Plugin.

Parameters

- **report-files** (*str*) – Setting that specifies the generated raw SLOCCount report files. Be sure not to include any non-report files into this pattern. The report files must have been generated by sloccount using the "–wide –details" options. (default '**/sloccount.sc')
- **charset** (*str*) – The character encoding to be used to read the SLOCCount result files. (default 'UTF-8')
- **builds-in-graph** (*int*) – Maximal number of last successful builds, that are displayed in the trend graphs. (default 0)
- **comment-is-code** (*bool*) – This option is considered only in the cloc report parser and is ignored in the SLOCCount one. (default false)
- **ignore-build-failure** (*bool*) – Try to process the report files even if the build is not successful. (default false)

Minimal Example:

```
publishers:
  - sloccount
```

Full Example:

```
publishers:
  - sloccount:
      report-files: sloccount.sc
      charset: latin-1
      builds-in-graph: 1
```

(continues on next page)

(continued from previous page)

```
    comment-is-code: true
    ignore-build-failure: true
```

sonar()

Sonar plugin support. Requires the Jenkins Sonar Plugin.

Parameters

- **installation-name** (*str*) – name of the Sonar instance to use (optional)
- **jdk** (*str*) – JDK to use (inherited from the job if omitted). (optional)
- **branch** (*str*) – branch onto which the analysis will be posted (default '')
- **language** (*str*) – source code language (default '')
- **root-pom** (*str*) – Root POM (default 'pom.xml')
- **private-maven-repo** (*bool*) – If true, use private Maven repository. (default false)
- **maven-opts** (*str*) – options given to maven (default '')
- **additional-properties** (*str*) – sonar analysis parameters (default '')
- **maven-installation-name** (*str*) – the name of the Maven installation to use (optional)
- **skip-global-triggers** (*dict*) –

  Triggers

  - **skip-when-scm-change** (*bool*): skip analysis when build triggered by scm (default false)
  - **skip-when-upstream-build** (*bool*): skip analysis when build triggered by an upstream build (default false)
  - **skip-when-envvar-defined** (*str*): skip analysis when the specified environment variable is set to true (default '')
- **settings** (*str*) – Path to use as user settings.xml. It is possible to provide a ConfigFileProvider settings file, see Example below. (optional)
- **global-settings** (*str*) – Path to use as global settings.xml. It is possible to provide a ConfigFileProvider settings file, see Example below. (optional)

Requires the Jenkins Config File Provider Plugin for the Config File Provider "settings" and "global-settings" config.

This publisher supports the post-build action exposed by the Jenkins Sonar Plugin, which is triggering a Sonar Analysis with Maven.

Minimal Example:

```
publishers:
  - sonar
```

Full Example:

```
publishers:
  - sonar:
      installation-name: MySonar
      jdk: MyJdk
      branch: myBranch
      language: java
      root-pom: mypom.xml
      private-maven-repo: true
      maven-installation-name: Maven3.3.3
      maven-opts: -DskipTests
      additional-properties: -DsonarHostURL=http://example.com/
      skip-global-triggers:
        skip-when-scm-change: true
```

(continues on next page)

(continued from previous page)

```
        skip-when-upstream-build: true
        skip-when-envvar-defined: SKIP_SONAR
      settings: org.jenkinsci.plugins.configfiles.maven.
↪MavenSettingsConfig0123456789012
      global-settings: org.jenkinsci.plugins.configfiles.maven.
↪GlobalMavenSettingsConfig0123456789012
```

sounds()

Play audio clips locally through sound hardware, remotely by piping them through an operating system command, or simultaneously through all browsers on a Jenkins page.

Requires the Jenkins Jenkins Sounds plugin

Parameters

- **success** (*dict*) – Play on success

  success

  - **sound** (*str*) - Sound name
  - **from (*list*) - Previous build result (default is all)**

    from values

    - **success**
    - **unstable**
    - **failure**
    - **not_build**
    - **aborted**

- **unstable** (*dict*) – Play on unstable. Specifying sound and conditions see *above*.
- **failure** (*dict*) – Play on failure. Specifying sound and conditions see *above*.
- **not_build** (*dict*) – Play on not build. Specifying sound and conditions see *above*.
- **aborted** (*dict*) – Play on aborted. Specifying sound and conditions see *above*.

Minimal example using defaults:

```
publishers:
- sounds:
    failure:
      sound: EXPLODE
```

Full example:

```
publishers:
- sounds:
    failure:
      sound: EXPLODE
      from:
        - not_build
        - aborted
    success:
      sound: Yahoo.Yodel
      from:
        - unstable
        - success
        - failure
```

ssh()

Upload files via SCP. Requires the Jenkins Publish over SSH Plugin.

Parameters

- **site** (*str*) – name of the ssh site
- **target** (*str*) – destination directory
- **target-is-date-format** (*bool*) – whether target is a date format. If true, raw text should be quoted (default false)
- **clean-remote** (*bool*) – should the remote directory be deleted before transferring files (default false)
- **source** (*str*) – source path specifier
- **command** (*str*) – a command to execute on the remote server (optional)
- **timeout** (*int*) – timeout in milliseconds for the Exec command (optional)
- **use-pty** (*bool*) – run the exec command in pseudo TTY (default false)
- **excludes** (*str*) – excluded file pattern (optional)
- **remove-prefix** (*str*) – prefix to remove from uploaded file paths (optional)
- **fail-on-error** (*bool*) – fail the build if an error occurs (default false).
- **always-publish-from-master** (*bool*) – transfer the files through the master before being sent to the remote server (defaults false)
- **flatten** (*bool*) – only create files on the server, don't create directories (default false).
- **verbose** (*bool*) – adds lots of detail useful for debug to the console but generally should be left off (default false)
- **retries** (*int*) – the number of times to retry this server in the event of failure (optional)
- **retry-delay** (*int*) – the time to wait, in milliseconds, before attempting another transfer (default 10000)

Minimal Example:

```yaml
publishers:
  - ssh:
      site: 'server.example.com'
      target: 'dest/dir/'
      source: 'base/source/dir/**'
```

Full Example:

```yaml
publishers:
  - ssh:
      site: 'server.example.com'
      target: "'dest/dir/'yyyyMMddHHmmss"
      target-is-date-format: true
      clean-remote: true
      source: 'base/source/dir/**'
      command: 'rm -r jenkins_$BUILD_NUMBER'
      timeout: 1800000
      use-pty: true
      excludes: '**/*.excludedfiletype'
      remove-prefix: 'base/source/dir'
      fail-on-error: true
      always-publish-from-master: true
      flatten: true
      verbose: true
      retries: 99
      retry-delay: 12345
```

stash()

This plugin will configure the Jenkins BitBucket Server Notifier plugin to notify Atlassian BitBucket after job

completes.

Requires the Jenkins Bitbucket Server Notifier Plugin.

Parameters

- **url** (*str*) – Base url of Stash Server (default "")
- **username** (*str*) – Username of Stash Server (default "")
- **password** (*str*) – Password of Stash Server (default "")
- **credentials-id** (*str*) – Credentials of Stash Server (optional)
- **ignore-ssl** (*bool*) – Ignore unverified SSL certificate (default false)
- **commit-sha1** (*str*) – Commit SHA1 to notify (default "")
- **include-build-number** (*bool*) – Include build number in key (default false)

Minimal Example:

```
publishers:
  - stash
```

Full Example:

```
publishers:
  - stash:
      url: "https://mystash"
      username: a
      password: b
      ignore-ssl: true
      commit-sha1: c
      include-build-number: true
```

tap()

Adds support to TAP test result files

Requires the Jenkins TAP Plugin.

Parameters

- **results** (*str*) – TAP test result files (required)
- **fail-if-no-results** (*bool*) – Fail if no result (default false)
- **failed-tests-mark-build-as-failure** (*bool*) – Mark build as failure if test fails (default false)
- **output-tap-to-console** (*bool*) – Output tap to console (default true)
- **enable-subtests** (*bool*) – Enable subtests (default true)
- **discard-old-reports** (*bool*) – Discard old reports (default false)
- **todo-is-failure** (*bool*) – Handle TODO's as failures (default true)
- **include-comment-diagnostics** (*bool*) – Include comment diagnostics (#) in the results table (>=1.12) (default false)
- **validate-tests** (*bool*) – Validate number of tests (>=1.13) (default false)
- **plan-required** (*bool*) – TAP plan required? (>=1.17) (default true)
- **verbose** (*bool*) – Print a message for each TAP stream file (>=1.17) (default true)
- **show-only-failures** (*bool*) – show only test failures (>=1.17) (default false)

Full Example:

```
publishers:
  - tap:
      results: puiparts.tap
      fail-if-no-results: true
      failed-tests-mark-build-as-failure: true
      output-tap-to-console: false
      enable-subtests: false
```

(continues on next page)

(continued from previous page)

```
      discard-old-reports: true
      todo-is-failure: false
      include-comment-diagnostics: true
      validate-tests: true
      plan-required: false
      verbose: false
      show-only-failures: true
```

Minimal Example:

```
publishers:
  - tap:
      results: puiparts.tap
```

tasks()

Scans the workspace files for open tasks and generates a trend report.

Requires the Jenkins Task Scanner Plugin (https://github.com/jenkinsci/tasks-plugin).

Parameters

- **files-to-scan** (*list*) – Fileset includes setting that specifies the workspace files to scan for tasks, such as `**/*.java`. Basedir of the fileset is the workspace root. (default '`**/*.java`')
- **files-to-exclude** (*list*) – Fileset excludes setting that specifies the workspace files to exclude scanning for tasks, such as library source files. Basedir of the fileset is the workspace root. (default '')
- **tasks-tags-high** (*list*) – Tags identifiers for high priority that should be looked for in the workspace files. Only alphanumerical characters are allowed as tags as these strings are pasted into a regular expression. (default '')
- **tasks-tags-normal** (*list*) – Tags identifiers for normal priority that should be looked for in the workspace files. Only alphanumerical characters are allowed as tags as these strings are pasted into a regular expression. (default '')
- **tasks-tags-low** (*list*) – Tags identifiers for low priority that should be looked for in the workspace files. Only alphanumerical characters are allowed as tags as these strings are pasted into a regular expression. (default '')
- **ignore-case** (*bool*) – Ignore the case of the the tag identifiers. (default false)
- **regular-expression** (*bool*) – Treat the tag identifiers as regular expression. Note that the regular expression must contain two capturing groups, the first one is interpreted as tag name, the second one as message. An example of such a regular expression would be `^.*(TODO(?:[0-9]*))(.*)$`. (default false)
- **run-always** (*bool*) – By default, this plug-in runs only for stable or unstable builds, but not for failed builds. If this plug-in should run even for failed builds then activate this check box. (default false)
- **detect-module** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings. Activating this option may increase your build time since the detector scans the whole workspace for `build.xml` or `pom.xml` files in order to assign the correct module names. (default false)
- **health-thresholds-100** (*int*) – Configure the upper thresholds for the build health. If left empty then no health report is created. If the actual number of warnings is between the provided thresholds then the build health is interpolated. (default '')
- **health-thresholds-0** (*str*) – Configure the lower thresholds for the build health. If left empty then no health report is created. If the actual number of warnings is between the provided thresholds then the build health is interpolated. (default '')
- **health-priorities** (*str*) – Determines which warning priorities should be consid-

ered when evaluating the build health. Can be `high` (only priority high), `normal` (priorities high and normal) or `low` (all priorities). (default 'low')

- `status-thresholds` (*dict*) – Configure the build status and health. If the number of total or new warnings is greater than one of these thresholds then a build is considered as unstable or failed, respectively. I.e., a value of 0 means that the build status is changed if there is at least one warning found. Leave this field empty if the state of the build should not depend on the number of warnings. Note that for new warnings, you need to enable the next option (`compute-new-warnings`).

  status-thresholds

  - **unstable-total-all** (*str*): Total number for all priorities, unstable threshold (default '')
  - **unstable-total-high** (*str*): Total number for high priority, unstable threshold (default '')
  - **unstable-total-normal** (*str*): Total number for normal priority, unstable threshold (default '')
  - **unstable-total-low** (*str*): Total number for low priority, unstable threshold (default '')
  - **failed-total-all** (*str*): Total number for all priorities, failure threshold (default '')
  - **failed-total-high** (*str*): Total number for high priority, failure threshold (default '')
  - **failed-total-normal** (*str*): Total number for normal priority, failure threshold (default '')
  - **failed-total-low** (*str*): Total number for low priority, failure threshold (default '')
  - **unstable-new-all** (*str*): New number for all priorities, unstable threshold (default '')
  - **unstable-new-high** (*str*): New number for high priority, unstable threshold (default '')
  - **unstable-new-normal** (*str*): New number for normal priority, unstable threshold (default '')
  - **unstable-new-low** (*str*): New number for low priority, unstable threshold (default '')
  - **failed-new-all** (*str*): New number for all priorities, failure threshold (default '')
  - **failed-new-high** (*str*): New number for high priority, failure threshold (default '')
  - **failed-new-normal** (*str*): New number for normal priority, failure threshold (default '')
  - **failed-new-low** (*str*): New number for low priority, failure threshold (default '')

- `compute-new-warnings` (*bool*) – Compute new warnings (based on the last successful build unless another reference build is chosen below). (default false)
- `use-delta` (*bool*) – If set the number of new warnings is computed by subtracting the total number of warnings of the reference build from the total number of warnings of the current build. This may lead to wrong results if you have both fixed and new warnings in a build. If unset the number of new warnings is computed by a more sophisticated algorithm: instead of using totals an asymmetric set difference of the warnings in the current build and the warnings in the reference build is used. This will find all new warnings even if the number of total warnings has decreased. Note that sometimes false positives will be reported due to minor changes in a warning (e.g. refactoring of variables or method names). It is recommended to uncheck this option in order to get the most accurate results for new warnings. Depends on `compute-new-warnings` option. (default false)

- **use-prev-build-as-ref** (*bool*) – If set the number of new warnings will always be computed based on the previous build, even if that build is unstable (due to a violated warning threshold). Otherwise the last build that did not violate any given threshold will be used as reference. It is recommended to uncheck this option if the plug-in should ensure that all new warnings will be finally fixed in subsequent builds. Depends on `compute-new-warnings` option. (default false)
- **only-use-stable-as-ref** (*bool*) – Use the last stable build as the reference to compute the number of new warnings against. This allows you to ignore interim unstable builds for which the number of warnings decreased. Note that the last stable build is evaluated only by inspecting the unit test failures. The static analysis results are not considered. Depends on `compute-new-warnings` option. (default false)
- **default-encoding** (*str*) – Default encoding when parsing or showing files. Leave this field empty to use the default encoding of the platform. (default '')

Minimal Example:

```
publishers:
  - tasks
```

Full Example:

```
publishers:
  - tasks:
      files-to-scan:
        - "**/*.java"
        - "**/*.c"
      files-to-exclude:
        - "specific/file.java"
      tasks-tags-high:
        - "FIXME"
        - "XXX"
      tasks-tags-normal:
        - "TODO"
      tasks-tags-low:
        - "NICETOHAVE"
      ignore-case: false
      regular-expression: false
      run-always: false
      detect-module: false
      health-thresholds-100: 100
      health-thresholds-0: 0
      health-priorities: 'low'
      status-thresholds:
        unstable-total-all: 0
        unstable-total-high: 0
        unstable-total-normal: 0
        unstable-total-low: 0
        failed-total-all: 0
        failed-total-high: 0
        failed-total-normal: 0
        failed-total-low: 0
        unstable-new-all: 0
        unstable-new-high: 0
        unstable-new-normal: 0
        unstable-new-low: 0
```

(continues on next page)

(continued from previous page)

```
        failed-new-all: 0
        failed-new-high: 0
        failed-new-normal: 0
        failed-new-low: 0
      compute-new-warnings: false
      use-delta: false
      use-prev-build-as-ref: false
      only-use-stable-as-ref: false
      default-encoding: "UTF-8"
```

test-fairy()

This plugin helps you to upload Android APKs or iOS IPA files to www.testfairy.com.

Requires the Jenkins Test Fairy Plugin.

Parameters

platform (*str*) – Select platform to upload to, **android** or **ios** (required)

Android Only:

Parameters

- **proguard-file** (*str*) – Path to Proguard file. Path of mapping.txt from your proguard output directory. (default '')
- **storepass** (*str*) – Password for the keystore (default android)
- **alias** (*str*) – alias for key (default androiddebugkey)
- **keypass** (*str*) – password for the key (default '')
- **keystorepath** (*str*) – Path to Keystore file (required)

IOS Only:

Parameters

dSYM-file (*str*) – Path to .dSYM.zip file (default '')

All:

Parameters

- **apikey** (*str*) – TestFairy API_KEY. Find it in your TestFairy account settings (required)
- **appfile** (*str*) – Path to App file (.apk) or (.ipa). For example: $WORKSPACE/[YOUR_FILE_NAME].apk or full path to the apk file. (required)
- **tester-groups** (*str*) – Tester groups to notify (default '')
- **notify-testers** (*bool*) – Send email with changelogs to testers (default false)
- **autoupdate** (*bool*) – Automatic update (default false)
- **max-duration** (*str*) – Duration of the session (default 10m)

  max-duration values

  - **10m**
  - **60m**
  - **300m**
  - **1440m**
- **record-on-background** (*bool*) – Record on background (default false)
- **data-only-wifi** (*bool*) – Record data only in wifi (default false)
- **video-enabled** (*bool*) – Record video (default true)
- **screenshot-interval** (*int*) – Time interval between screenshots (default 1)

  screenshot-interval values

  - **1**
  - **2**
  - **5**
- **video-quality** (*str*) – Video quality (default high)

  video-quality values

  - **high**

  - **medium**
  - **low**
- **cpu** (*bool*) – Enable CPU metrics (default true)
- **memory** (*bool*) – Enable memory metrics (default true)
- **logs** (*bool*) – Enable logs metrics (default true)
- **network** (*bool*) – Enable network metrics (default false)
- **phone-signal** (*bool*) – Enable phone signal metrics (default false)
- **wifi** (*bool*) – Enable wifi metrics (default false)
- **gps** (*bool*) – Enable gps metrics (default false)
- **battery** (*bool*) – Enable battery metrics (default false)
- **opengl** (*bool*) – Enable opengl metrics (default false)

Example:

```
publishers:
  - test-fairy:
      platform: android
      apikey: apikey
      appfile: /tmp/appfile.apk
      keystorepath: /tmp/keystorefile
```

```
publishers:
  - test-fairy:
      platform: android
      apikey: apikey
      appfile: /tmp/appfile.apk
      proguard-file: /tmp/proguardfile
      keystorepath: /tmp/keystorefile
      storepass: android
      alias: androiddebugkey
      keypass: keypass
      tester-groups: testergroups
      notify-testers: true
      autoupdate: true
```

```
publishers:
  - test-fairy:
      platform: ios
      apikey: apikey
      appfile: /tmp/appfile.ipa
```

```
publishers:
  - test-fairy:
      platform: ios
      apikey: apikey
      appfile: /tmp/appfile.ipa
      dSYM-file: /tmp/dsym.zip
      tester-groups: testergroups
      notify-testers: true
      autoupdate: true
```

testng()

This plugin publishes TestNG test reports.

Requires the Jenkins TestNG Results Plugin.

Parameters

- **pattern** (*str*) – filename pattern to locate the TestNG XML report files (required)
- **escape-test-description** (*bool*) – escapes the description string associated with the test method while displaying test method details (default true)
- **escape-exception-msg** (*bool*) – escapes the test method's exception messages. (default true)
- **fail-on-failed-test-config** (*bool*) – Allows for a distinction between failing tests and failing configuration methods (>=1.10) (default false)
- **show-failed-builds** (*bool*) – include results from failed builds in the trend graph (>=1.6) (default false)
- **unstable-skips** (*int*) – Build is marked UNSTABLE if the number/percentage of skipped tests exceeds the specified threshold (>=1.11) (default 100)
- **unstable-fails** (*int*) – Build is marked UNSTABLE if the number/percentage of failed tests exceeds the specified threshold (>=1.11) (default 0)
- **failed-skips** (*int*) – Build is marked FAILURE if the number/percentage of skipped tests exceeds the specified threshold (>=1.11) (default 100)
- **failed-fails** (*int*) – Build is marked FAILURE if the number/percentage of failed tests exceeds the specified threshold (>=1.11) (default 100)
- **threshold-mode** (*str*) – Interpret threshold as number of tests or percentage of tests (>=1.11) (default percentage)

Full Example:

```
publishers:
    - testng:
        pattern: "**/testng-results.xml"
        escape-test-description: false
        escape-exception-msg: false
        fail-on-failed-test-config: true
        show-failed-builds: true
        unstable-skips: 50
        unstable-fails: 25
        failed-skips: 75
        failed-fails: 66
        threshold-mode: number
```

Minimal Example:

```
publishers:
  - testng:
      pattern: "**/testng-results.xml"
```

testselector()

This plugin allows you to choose specific tests you want to run.

Requires the Jenkins Tests Selector Plugin.

Parameters

- **name** (*str*) – Environment variable in which selected tests are saved (required)
- **description** (*str*) – Description (default "")
- **properties-file** (*str*) – Contain all your tests (required)
- **enable-field** (*str*) – Imply if the test is enabled or not (default "")
- **groupby** (*str*) – Plugin will group the tests by (default "")
- **field-sperator** (*str*) – Separate between the fields in the tests tree (default "")
- **show-fields** (*str*) – Shown in the tests tree (default "")
- **multiplicity-field** (*str*) – Number of times the test should run (default "")

Example:

```yaml
publishers:
  - testselector:
      name: tests
      description: integration
      properties-file: example.properties
      enable-field: enabled
      groupby: testgroup
      field-separator: .
      show-fields: testsuite,testcase
      multiplicity-field: multiplicity
```

text-finder()

This plugin lets you search keywords in the files you specified and additionally check build status

Requires the Jenkins Text-finder Plugin.

Parameters

- **regexp** (*str*) – Specify a regular expression (required)
- **fileset** (*str*) – Specify the path to search (optional)
- **also-check-console-output** (*bool*) – Search the console output (default false)
- **succeed-if-found** (*bool*) – Force a build to succeed if a string was found (default false)
- **unstable-if-found** (*bool*) – Set build unstable instead of failing the build (default false)
- **not-built-if-found** (*bool*) – Set build to "Not Built" instead of failing the build (default false)

Example:

```yaml
publishers:
  - text-finder:
      regexp: "some string"
      fileset: "file.txt"
      also-check-console-output: true
      succeed-if-found: false
      unstable-if-found: false
      not-built-if-found: false
```

trigger()

Trigger non-parametrised builds of other jobs.

Parameters

- **project** (*str*) – name of the job to trigger
- **threshold** (*str*) – when to trigger the other job (default 'SUCCESS'), alternatives: SUCCESS, UNSTABLE, FAILURE

Example:

```yaml
publishers:
  - trigger:
      project: other_job
      threshold: SUCCESS
```

trigger-parameterized-builds()

Trigger parameterized builds of other jobs. Requires the Jenkins Parameterized Trigger Plugin.

Use of the *node-label-name* or *node-label* parameters requires the Jenkins NodeLabel Parameter Plugin. Note: 'node-parameters' overrides the Node that the triggered project is tied to.

Parameters

- **project** (*list*) – list the jobs to trigger, will generate comma-separated string containing the named jobs.
- **predefined-parameters** (*str*) – parameters to pass to the other job (optional)
- **current-parameters** (*bool*) – Whether to include the parameters passed to the current build to the triggered job (optional)
- **node-parameters** (*bool*) – Use the same Node for the triggered builds that was used for this build. (optional)
- **svn-revision** (*bool*) – Pass svn revision to the triggered job (optional)
- **include-upstream** (*bool*) – Include/pass through Upstream SVN Revisons. Only valid when 'svn-revision' is true. (default false)
- **git-revision** (*dict*) – Passes git revision to the triggered job (optional).
  - **combine-queued-commits** (bool): Whether to combine queued git hashes or not (default false)
- **combine-queued-commits** (*bool*) – Combine Queued git hashes. Only valid when 'git-revision' is true. (default false)

  Deprecated since version 1.5.0: Please use *combine-queued-commits* under the *git-revision* argument instead.
- **boolean-parameters** (*dict*) – Pass boolean parameters to the downstream jobs. Specify the name and boolean value mapping of the parameters. (optional)
- **condition** (*str*) – when to trigger the other job. Can be: 'SUCCESS', 'UNSTABLE', 'FAILED_OR_BETTER', 'UNSTABLE_OR_BETTER', 'UNSTABLE_OR_WORSE', 'FAILED', 'ALWAYS'. (default 'ALWAYS')
- **property-file** (*str*) – Use properties from file (optional)
- **fail-on-missing** (*bool*) – Blocks the triggering of the downstream jobs if any of the property files are not found in the workspace. Only valid when 'property-file' is specified. (default 'False')
- **property-multiline** (*bool*) – When enabled properties containing newline character(s) are propagated as TextParameterValue which is a specialized StringParameterValue commonly used for handling multi-line strings in Jenkins. When disabled (default) all properties are propagated as StringParameterValue. (default 'False') (>=2.35.2)
- **trigger-from-child-projects** (*bool*) – Trigger build from child projects. Used for matrix projects. (default 'False')
- **use-matrix-child-files** (*bool*) – Use files in workspaces of child builds (default 'False')
- **matrix-child-combination-filter** (*str*) – A Groovy expression to filter the child builds to look in for files
- **only-exact-matrix-child-runs** (*bool*) – Use only child builds triggered exactly by the parent.
- **file-encoding** (*str*) – Encoding of contents of the files. If not specified, default encoding of the platform is used. Only valid when 'property-file' is specified. (optional)
- **trigger-with-no-params** (*bool*) – Trigger a build even when there are currently no parameters defined (default 'False')
- **restrict-matrix-project** (*str*) – Filter that restricts the subset of the combinations that the downstream project will run (optional)
- **node-label-name** (*str*) – Specify the Name for the NodeLabel parameter. (optional)
- **node-label** (*str*) – Specify the Node for the NodeLabel parameter. (optional)

Example:

```
publishers:
  - trigger-parameterized-builds:
    - project:
```

(continues on next page)

(continued from previous page)

```
        - other_job
        - foo
        - bar
      predefined-parameters: |
        foo=bar
        bar=foo
    - project: other_job1, other_job2
      predefined-parameters: BUILD_NUM=${BUILD_NUMBER}
      git-revision: true
      property-file: version.prop
      fail-on-missing: true
      property-multiline: true
    - project: yet_another_job
      predefined-parameters: foo=bar
      git-revision:
        combine-queued-commits: true
      restrict-matrix-project: label=="x86"
    - project: yet_another_job_2
      node-label-name: foo
    - project: yet_another_job_3
      node-label: node-label-foo || node-label-bar
    - project: 'test-project-same-node'
      node-parameters: true
      current-parameters: true
```

```
publishers:
  - trigger-parameterized-builds:
    - project:
        - other_job
        - foo
        - bar
      boolean-parameters:
          p1: true
          p2: false
      svn-revision: true
      include-upstream: true
      git-revision: true
      combine-queued-commits: true
      property-file: version.prop
      file-encoding: UTF-8
```

valgrind()

This plugin publishes Valgrind Memcheck XML results.

Requires the Jenkins Valgrind Plugin.

Parameters

- **pattern** (*str*) – Filename pattern to locate the Valgrind XML report files (required)
- **thresholds** (*dict*) – Mark build as failed or unstable if the number of errors exceeds a threshold. All threshold values are optional.

  thresholds

  – **unstable (*dict*)**

  unstable

* **invalid-read-write** (*int*)
* **definitely-lost** (*int*)
* **total** (*int*)

– **failed** (*dict*)

failed

* **invalid-read-write** (*int*)
* **definitely-lost** (*int*)
* **total** (*int*)

- **fail-no-reports** (*bool*) – Fail build if no reports are found (default false)
- **fail-invalid-reports** (*bool*) – Fail build if reports are malformed (default false)
- **publish-if-aborted** (*bool*) – Publish results for aborted builds (default false)
- **publish-if-failed** (*bool*) – Publish results for failed builds (default false)

Example:

```
publishers:
  - valgrind:
      pattern: "test.xml"
      thresholds:
        unstable:
          invalid-read-write: 1
          definitely-lost: 2
          total: 3
        failed:
          invalid-read-write: 4
          definitely-lost: 5
          total: 6
      fail-no-reports: true
      fail-invalid-reports: true
      publish-if-aborted: true
      publish-if-failed: true
```

violations()

Publish code style violations. Requires the Jenkins Violations Plugin.

The violations component accepts any number of dictionaries keyed by the name of the violations system. The dictionary has the following values:

Parameters

- **min** (*int*) – sunny threshold
- **max** (*int*) – stormy threshold
- **unstable** (*int*) – unstable threshold
- **pattern** (*str*) – report filename pattern

Any system without a dictionary provided will use default values.

Valid systems are:

checkstyle, codenarc, cpd, cpplint, csslint, findbugs, fxcop, gendarme, jcreport, jslint, pep8, perlcritic, pmd, pylint, simian, stylecop

Example:

```
publishers:
  - violations:
      pep8:
        min: 0
```

(continues on next page)

(continued from previous page)

```
        max: 1
        unstable: 1
        pattern: '**/pep8.txt'
```

warnings()

Generate trend report for compiler warnings in the console log or in log files.

Requires the Jenkins Warnings Plugin (https://github.com/jenkinsci/warnings-plugin).

Parameters

- **console-log-parsers** (*list*) – The parser to use to scan the console log (default '')
- **workspace-file-scanners** (*dict*) –
  workspace-file-scanners
  - **file-pattern (*str*) – Fileset 'includes' setting that** specifies the files to scan for warnings (required)
  - **scanner (*str*) – The parser to use to scan the files** provided in workspace-file-pattern (default '')
- **files-to-include** (*str*) – Comma separated list of regular expressions that specifies the files to include in the report (based on their absolute filename). By default all files are included
- **files-to-ignore** (*str*) – Comma separated list of regular expressions that specifies the files to exclude from the report (based on their absolute filename). (default '')
- **messages-to-ignore** (*str*) – Newline separated list of regular expressions that specifies the warning messages to exclude form the report (based on the warning messages). By default all warning messages are included
- **categories-to-ignore** (*str*) – Newline separated list of regular expressions that specifies the warning messages to exclude form the report (based on the warning categories). By default all warning categories are included
- **run-always** (*bool*) – By default, this plug-in runs only for stable or unstable builds, but not for failed builds. Set to true if the plug-in should run even for failed builds. (default false)
- **detect-modules** (*bool*) – Determines if Ant or Maven modules should be detected for all files that contain warnings. Activating this option may increase your build time since the detector scans the whole workspace for 'build.xml' or 'pom.xml' files in order to assign the correct module names. (default false)
- **resolve-relative-paths** (*bool*) – Determines if relative paths in warnings should be resolved using a time expensive operation that scans the whole workspace for matching files. Deactivate this option if you encounter performance problems. (default false)
- **health-threshold-high** (*int*) – The upper threshold for the build health. If left empty then no health report is created. If the actual number of warnings is between the provided thresholds then the build health is interpolated (default '')
- **health-threshold-low** (*int*) – The lower threshold for the build health. See health-threshold-high. (default '')
- **health-priorities** (*dict*) – Determines which warning priorities should be considered when evaluating the build health (default all-priorities)
  health-priorities values
  - **priority-high** – Only priority high
  - **high-and-normal** – Priorities high and normal
  - **all-priorities** – All priorities
- **total-thresholds** (*dict*) – If the number of total warnings is greater than one of these thresholds then a build is considered as unstable or failed, respectively. (default '')
  total-thresholds
  - **unstable (*dict*)**

unstable

* **total-all** (*int*)
* **total-high** (*int*)
* **total-normal** (*int*)
* **total-low** (*int*)

– **failed (*dict*)**

failed

* **total-all** (*int*)
* **total-high** (*int*)
* **total-normal** (*int*)
* **total-low** (*int*)

- **new-thresholds** (*dict*) – If the specified number of new warnings exceeds one of these thresholds then a build is considered as unstable or failed, respectively. (default '')

new-thresholds

– **unstable (*dict*)**

unstable

* **new-all** (*int*)
* **new-high** (*int*)
* **new-normal** (*int*)
* **new-low** (*int*)

– **failed (*dict*)**

failed

* **new-all** (*int*)
* **new-high** (*int*)
* **new-normal** (*int*)
* **new-high** (*int*)

- **use-delta-for-new-warnings** (*bool*) – If set then the number of new warnings is calculated by subtracting the total number of warnings of the current build from the reference build. This may lead to wrong results if you have both fixed and new warnings in a build. If not set, then the number of new warnings is calculated by an asymmetric set difference of the warnings in the current and reference build. This will find all new warnings even if the number of total warnings is decreasing. However, sometimes false positives will be reported due to minor changes in a warning (refactoring of variable of method names, etc.) (default false)
- **use-previous-build-as-reference** (*bool*) – If set the number of new warnings will always be computed based on the previous build, even if that build is unstable (due to a violated warning threshold). Otherwise the last build that did not violate any given threshold will be used as reference. It is recommended to uncheck this option if the plug-in should ensure that all new warnings will be finally fixed in subsequent builds. (default false)
- **only-use-stable-builds-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **default-encoding** (*str*) – Default encoding when parsing or showing files Leave empty to use default encoding of platform (default '')

Minimal Example:

```yaml
publishers:
  - warnings
```

Full Example:

```yaml
publishers:
  - warnings:
      console-log-parsers:
        - FxCop
        - CodeAnalysis
      workspace-file-scanners:
        - file-pattern: '**/*.out'
          scanner: 'AcuCobol Compiler'
        - file-pattern: '**/*.warnings'
          scanner: FxCop
      files-to-include: '[a-zA-Z]\.java,[a-zA-Z]\.cpp'
      files-to-ignore: '[a-zA-Z]\.html,[a-zA-Z]\.js'
      messages-to-ignore: |-
        ^Test results:.*
        No sources found skipping Kotlin compile
      categories-to-ignore: |-
        WARN.*
        ERROR
      run-always: true
      detect-modules: true
      resolve-relative-paths: true
      health-threshold-high: 50
      health-threshold-low: 25
      health-priorities: high-and-normal
      total-thresholds:
        unstable:
          total-all: 90
          total-high: 90
          total-normal: 40
          total-low: 30
        failed:
          total-all: 100
          total-high: 100
          total-normal: 50
          total-low: 40
      new-thresholds:
        unstable:
          new-all: 100
          new-high: 50
          new-normal: 30
          new-low: 10
        failed:
          new-all: 100
          new-high: 60
          new-normal: 50
          new-low: 40
      use-delta-for-new-warnings: true
```

(continues on next page)

(continued from previous page)

```
      use-previous-build-as-reference: true
      only-use-stable-builds-as-reference: true
      default-encoding: ISO-8859-9
```

whitesource()

This plugin brings automatic open source management to Jenkins users.

Requires the Jenkins Whitesource Plugin.

Parameters

- **product-token** (*str*) – Product name or token to update (default '')
- **version** (*str*) – Product version (default '')
- **override-token** (*str*) – Override the api token from the global config (default '')
- **project-token** (*str*) – Token uniquely identifying the project to update (default '')
- **includes** (*list*) – list of libraries to include (default '[]')
- **excludes** (*list*) – list of libraries to exclude (default '[]')
- **policies** (*str*) – Whether to override the global settings. Valid values: global, enable, disable (default 'global')
- **requester-email** (*str*) – Email of the WhiteSource user that requests to update WhiteSource (>=1.5.1) (default '')

Full Example:

```
publishers:
  - whitesource:
      product-token: abcdefghijklmnopqrstuvwxyzabcdef
      version: 1.0.17
      policies: enable
      override-token: "1231424523412"
      project-token: sd;fkljsdfkljasdfkj
      requester-email: foo@email.com
      includes:
        - lib/*.jar
        - test/lib/*.jar
      excludes:
        - lib/ant*.jar
        - test/lib/ant*.jar
```

Minimal Example:

```
publishers:
  - whitesource
```

workspace-cleanup (post-build)()

Requires the Jenkins Workspace Cleanup Plugin.

The pre-build workspace-cleanup is available as a wrapper.

Parameters

- **include** (*list*) – list of files to be included
- **exclude** (*list*) – list of files to be excluded
- **dirmatch** (*bool*) – Apply pattern to directories too (default false)
- **clean-if** (*list*) – clean depending on build status

  clean-if values

  - **success** (*bool*) (default true)
  - **unstable** (*bool*) (default true)
  - **failure** (*bool*) (default true)

- **aborted** (*bool*) (default true)
- **not-built** (*bool*) (default true)

- **fail-build** (*bool*) – Fail the build if the cleanup fails (default true)
- **clean-parent** (*bool*) – Cleanup matrix parent workspace (default false)
- **external-deletion-command** (*str*) – external deletion command to run against files and directories
- **disable-deferred-wipeout** (*bool*) – Disable improved deferred wipeout method (default false)

Minimal Example:

```
publishers:
  - workspace-cleanup
```

Full Example:

```
publishers:
  - workspace-cleanup:
      include:
        - "*.zip"
      exclude:
        - "*.txt"
      clean-if:
        - success: false
        - unstable: false
        - failure: false
        - aborted: false
        - not-built: false
      dirmatch: true
      fail-build: false
      clean-parent: true
      external-deletion-command: 'command'
      disable-deferred-wipeout: true
```

xml-summary()

Adds support for the Summary Display Plugin

Requires the Jenkins Summary Display Plugin.

Parameters

- **files** (*str*) – Files to parse (required)
- **shown-on-project-page** (*bool*) – Display summary on project page (default false)

Minimal Example:

```
publishers:
  - xml-summary:
      files: '*_summary_report.xml'
```

Full Example:

```
publishers:
  - xml-summary:
      files: '*_summary_report.xml'
      shown-on-project-page: true
```

xunit()

Publish tests results. Requires the Jenkins xUnit Plugin.

Parameters

- **thresholdmode** (*str*) – Whether thresholds represents an absolute number of tests or a percentage. Either 'number' or 'percent'. (default 'number')
- **thresholds** (*list*) – Thresholds for both 'failed' and 'skipped' tests.
  threshold (*dict*)
  Threshold values to set, where missing, xUnit should default to an internal value of 0. Each test threshold should contain the following:
  - **unstable** (*int*)
  - **unstablenew** (*int*)
  - **failure** (*int*)
  - **failurenew** (*int*)
- **test-time-margin** (*int*) – Give the report time margin value in ms, before to fail if not new unless the option **requireupdate** is set for the configured framework. (default 3000)
- **types** (*list*) – Frameworks to configure, and options. Supports the following: `aunit`, `boosttest`, `checktype`, `cpptest`, `cppunit`, `ctest`, `dotnettest`, `embunit`, `fpcunit`, `gtest`, `junit`, `mstest`, `nunit`, `phpunit`, `tusar`, `unittest`, and `valgrind`.

  The 'custom' type is not supported.
  type (*dict*)
  each type can be configured using the following:
  - **pattern** (*str*): An Ant pattern to look for Junit result files, relative to the workspace root (default '')
  - **requireupdate** (*bool*): fail the build whenever fresh tests results have not been found (default true).
  - **deleteoutput** (*bool*): delete temporary JUnit files (default true).
  - **skip-if-no-test-files** (*bool*): Skip parsing this xUnit type report if there are no test reports files (default false).
  - **stoponerror** (*bool*): Fail the build whenever an error occur during a result file processing (default true).

Example:

```
publishers:
  - xunit:
      thresholdmode: 'percent'
      thresholds:
          - failed:
                unstable: 0
                unstablenew: 0
                failure: 0
                failurenew: 0
          - skipped:
                unstable: 0
                unstablenew: 0
                failure: 0
                failurenew: 0
      types:
          - phpunit:
              pattern: "junit.log"
              stoponerror: true
          - cppunit:
              pattern: "cppunit.log"
```

(continues on next page)

(continued from previous page)

```
        - gtest:
            pattern: "gtest.log"
```

zulip()

Set build status on zulip. Requires the Jenkins Humbug Plugin.

Example:

```
publishers:
  - zulip
```

Reporters

Reporters are like publishers but only applicable to Maven projects.

Component: reporters

Macro
reporter

Entry Point
jenkins_jobs.reporters

Example:

```
job:
  name: test_job
  project-type: maven

  reporters:
    - email:
        recipients: breakage@example.com
```

class reporters.**Reporters**(*registry*)

component_list_type = 'reporters'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'reporter'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 55

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

email()

Email notifications on build failure.

Parameters

- **recipients** (*str*) – Recipient email addresses
- **notify-every-unstable-build** (*bool*) – Send an email for every unstable build (default true)
- **send-to-individuals** (*bool*) – Send an email to the individual who broke the build (default false)
- **notify-for-each-module** (*bool*) – Send an email for each module (e.g. failed, unstable). (default true)

Example:

```
reporters:
  - email:
      recipients: breakage@example.com
```

findbugs()

FindBugs reporting for builds

Requires the Jenkins FindBugs Plugin (https://github.com/jenkinsci/findbugs-plugin).

Parameters

- **rank-priority** (*bool*) – Use rank as priority (default false)
- **include-files** (*str*) – Comma separated list of files to include. (Optional)
- **exclude-files** (*str*) – Comma separated list of files to exclude. (Optional)
- **can-run-on-failed** (*bool*) – Weather or not to run plug-in on failed builds (default false)
- **healthy** (*int*) – Sunny threshold (optional)
- **unhealthy** (*int*) – Stormy threshold (optional)
- **health-threshold** (*str*) – Threshold priority for health status ('low', 'normal' or 'high', defaulted to 'low')
- **dont-compute-new** (*bool*) – If set to false, computes new warnings based on the reference build (default true)
- **use-delta-values** (*bool*) – Use delta for new warnings. (default false)
- **use-previous-build-as-reference** (*bool*) – If set then the number of new warnings will always be calculated based on the previous build. Otherwise the reference build. (default false)
- **use-stable-build-as-reference** (*bool*) – The number of new warnings will be calculated based on the last stable build, allowing reverts of unstable builds where the number of warnings was decreased. (default false)
- **thresholds** (*dict*) –

  thresholds

  – **unstable (*dict*)**

  unstable

  * **total-all** (*int*)
  * **total-high** (*int*)
  * **total-normal** (*int*)
  * **total-low** (*int*)
  * **new-all** (*int*)
  * **new-high** (*int*)
  * **new-normal** (*int*)

* **new-low** (*int*)

– **failed** (*dict*)

failed

* **total-all** (*int*)
* **total-high** (*int*)
* **total-normal** (*int*)
* **total-low** (*int*)
* **new-all** (*int*)
* **new-high** (*int*)
* **new-normal** (*int*)
* **new-low** (*int*)

Minimal Example:

```
project-type: maven
reporters:
    - findbugs
```

Full Example:

```
project-type: maven
reporters:
    - findbugs:
        rank-priority: true
        include-files: 'f,d,e,.*'
        exclude-files: 'a,c,d,.*'
        can-run-on-failed: true
        healthy: 80
        unhealthy: 10
        use-delta-values: true
        health-threshold: 'high'
        thresholds:
            unstable:
                total-all: 90
                total-high: 80
                total-normal: 50
                total-low: 20
                new-all: 95
                new-high: 85
                new-normal: 55
                new-low: 25
            failed:
                total-all: 80
                total-high: 70
                total-normal: 40
                total-low: 10
                new-all: 85
                new-high: 75
                new-normal: 45
                new-low: 15
        dont-compute-new: false
```

(continues on next page)

(continued from previous page)

```
        use-delta-values: true
        use-previous-build-as-reference: true
        use-stable-build-as-reference: true
```

SCM

The SCM module allows you to specify the source code location for the project. It adds the `scm` attribute to the *Job* definition, which accepts any number of scm definitions. It is also possible to pass `[]` to the `scm` attribute. This is useful when a set of configs has a global default `scm` and you want to a particular job to override that default with no SCM.

Component: scm

Macro
scm

Entry Point
jenkins_jobs.scm

The scm module allows referencing multiple repositories in a Jenkins job. Note: Adding more than one scm definition requires the Jenkins Multiple SCMs plugin.

Example of multiple repositories in a single job:

```
- scm:
    name: first-scm
    scm:
      - git:
          url: ssh://jenkins@review.openstack.org:29418/first.git
          branches:
           - origin/master

- scm:
    name: second-scm
    scm:
      - git:
          url: ssh://jenkins@review.openstack.org:29418/second.git
          branches:
           - origin/master

- scm:
    name: first-and-second
    scm:
      - first-scm
      - second-scm

- job:
    name: my-job
    scm:
      - first-and-second
```

Example of an empty `scm`:

```
scm: []
```

class scm.PipelineSCM(*registry*)

component_list_type = 'pipeline-scm'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'pipeline-scm'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 30

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

class scm.SCM(*registry*)

component_list_type = 'scm'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'scm'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 30

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

accurev()

Specifies the AccuRev SCM repository for this job.

Requires the Jenkins AccuRev Plugin.

Parameters

- **depot** (*str*) – Depot you want to use for the current job (optional)
- **stream** (*str*) – Stream where the build will be generated from (optional)

- **server-name** (*str*) – AccuRev server you are using for your builds (required)
- **ignore-parent-changes** (*bool*) – Ignore possibility of changes in the parent stream (default false)
- **clean-reference-tree** (*bool*) – Deletes any external files in reference tree (default false)
- **build-from-snapshot** (*bool*) – Creates snapshot of the target stream, then populates and builds from that snapshot (default false)
- **do-not-pop-content** (*bool*) – If checkbox is on, elements are not populating vice versa (default false)
- **workspace** (*str*) – Name of existing workspace (optional)
- **reference-tree** (*str*) – Name of the reference tree (optional)
- **directory-offset** (*str*) – Relative directory path from the default Jenkins workspace location where the files from the stream, workspace, or reference tree should be retrieved from. (optional)
- **sub-path** (*str*) – Makes a "best effort" to ensure that only the sub-path is populated (optional)
- **filter-poll-scm** (*str*) – Specify directories or files you want Jenkins to check before starting a build (optional)
- **snapshot-name-format** (*str*) – Naming conventions for the snapshot in this field (optional)

Example:

```
scm:
  - accurev:
      depot: Test depot
      stream: Test stream
      server-name: Test server name
      ignore-parent-changes: true
      clean-reference-tree: true
      build-from-snapshot: true
      do-not-pop-content: true
      workspace: Test workspace
      reference-tree: Test reference tree
      directory-offset: Test directory offset
      sub-path: Test sub path
      filter-poll-scm: Test filter
      snapshot-name-format: Test snapshot name format
```

bzr()

Specifies the bzr SCM repository for this job.

Requires the Jenkins Bazaar Plugin.

Parameters

- **url** (*str*) – URL of the bzr branch (required)
- **clean-tree** (*bool*) – Clean up the workspace (using bzr) before pulling the branch (default false)
- **lightweight-checkout** (*bool*) – Use a lightweight checkout instead of a full branch (default false)
- **browser** (*str*) – The repository browser to use.

  browsers supported

  - **auto** - (default)
  - **loggerhead** - as used by Launchpad
  - **opengrok** - https://opengrok.github.io/OpenGrok/
- **browser-url** (*str*) – URL for the repository browser (required if browser is set).

- **opengrok-root-module** (*str*) – Root module for OpenGrok (required if browser is opengrok).

Example:

```
scm:
  - bzr:
      url: lp:test_project
```

cvs()

Specifies the CVS SCM repository for this job.

Requires the Jenkins CVS Plugin.

Parameters

- **repos** (*list*) – List of CVS repositories. (required)

  Repos

  - **root** (*str*) – The CVS connection string Jenkins uses to connect to the server. The format is :protocol:user@host:path (required)
  - **locations** (*list*) – List of locations. (required)

    Locations

    * **type** (*str*) – Type of location.

      supported values

      - **HEAD** - (default)
      - **BRANCH**
      - **TAG**

    * **name** (*str*) – Name of location. Only valid in case of 'BRANCH' or 'TAG' location type. (default '')
    * **use-head** (*bool*) – Use Head if not found. Only valid in case of 'BRANCH' or 'TAG' location type. (default false)
    * **modules** (*list*) – List of modules. (required)

      Modules

      - **re-**

mote – The name of the module in the repository at CVSROOT. (required)

.

localname – The name to be applied to this module in the local workspace. If blank, the remote module name will be used. (default '')

– **excluded-regions** (*list str*) – Patterns for excluding regions. (optional)

  - **compression-level** (*int*) – Compression level. Must be a number between -1 and 9 inclusive. Choose -1 for System Default. (default -1)
- **use-update** (*bool*) – If true, Jenkins will use 'cvs update' whenever possible for builds. This makes a build faster. But this also causes the artifacts from the previous build to remain in the file system when a new build starts, making it not a true clean build. (default true)
- **prune-empty** (*bool*) – Remove empty directories after checkout using the CVS '-P' option. (default true)
- **skip-changelog** (*bool*) – Prevent the changelog being generated after checkout has completed. (default false)
- **show-all-output** (*bool*) – Instructs CVS to show all logging output. CVS normally runs in quiet mode but this option disables that. (default false)
- **clean-checkout** (*bool*) – Perform clean checkout on failed update. (default false)
- **clean-copy** (*bool*) – Force clean copy for locally modified files. (default false)

Example

```
scm:
  - cvs:
      repos:
          - root: ":protocol:user@host1:path"
            locations:
                - modules:
                    - remote: "remote1"
                    - remote: "remote2"
          - root: ":protocol:user@host2:path"
            locations:
                - modules:
                    - remote: "remote1"
```

```
scm:
  - cvs:
      repos:
          - root: ":protocol:user@host1:path"
            compression-level: "1"
            locations:
                - type: TAG
                  name: "tag name"
                  use-head: false
                  modules:
                    - remote: "remote1"
                      local-name: "localName"
                    - remote: "remote2"
            excluded-regions:
                - "pattern1"
                - "pattern2"
          - root: ":protocol:user@host2:path"
            locations:
                - modules:
                    - remote: "remote1"
      use-update: false
      prune-empty: false
      skip-changelog: true
```

(continues on next page)

(continued from previous page)

```
      show-all-output: true
      clean-checkout: true
      clean-copy: true
```

dimensions()

Specifies the Dimensions SCM repository for this job.

Requires Jenkins Dimensions Plugin.

Parameters

- **project** (*str*) – Project name of format PRODUCT_ID:PROJECT_NAME (required)
- **permissions** (*str*) – Default Permissions for updated files (default: DEFAULT)
  - **Permissions**
    - **DEFAULT**
    - **READONLY**
    - **WRITABLE**
- **eol** (*str*) – End of line (default: DEFAULT)
  - **End of line**
    - **DEFAULT**
    - **UNIX**
    - **WINDOWS**
    - **UNCHANGED**
- **folders** (*list*) – Folders to monitor (default /)
- **exclude** (*list*) – Paths to exclude from monitor
- **username** (*str*) – Repository username for this job
- **password** (*str*) – Repository password for this job
- **server** (*str*) – Dimensions server for this job
- **database** (*str*) – Dimensions database for this job. Format must be database@dsn
- **update** (*bool*) – Use update (default false)
- **clear-workspace** (*bool*) – Clear workspace prior to build (default false)
- **force-build** (*bool*) – Force build even if the repository SCM checkout operation fails (default false)
- **overwrite-modified** (*bool*) – Overwrite files in worspace from repository files (default false)
- **expand-vars** (*bool*) – Expand substitution variables (default false)
- **no-metadata** (*bool*) – Checkout files with no metadata (default false)
- **maintain-timestamp** (*bool*) – Maintain file timestamp from Dimensions (default false)
- **slave-checkout** (*bool*) – Force slave based checkout (default false)
- **timezone** (*str*) – Server timezone
- **web-url** (*str*) – Dimensions Web URL

Examples:

```
scm:
  - dimensions:
      project: myProduct:myProject
```

```
scm:
  - dimensions:
      project: myProduct:myProject
      permissions: WRITABLE
      eol: UNIX
      folders:
```

(continues on next page)

(continued from previous page)

```
      - src
      - test
    exclude:
      - excluded_dir
      - excluded_other_dir
    username: johnd
    password: passw0rd
    server: my.dmscm.server:1234
    database: myDatabase@myDsn
    update: true
    clear-workspace: true
    force-build: true
    overwrite-modified: true
    expand-vars: true
    no-metadata: true
    maintain-timestamp: true
    slave-checkout: true
    timezone: Europe/Berlin
    web-url: https://my.dmscm.weburl
```

git()

Specifies the git SCM repository for this job.

Requires the Jenkins Git Plugin.

Parameters

- **url** (*str*) – URL of the git repository
- **credentials-id** (*str*) – ID of credential to use to connect, which is the last field (a 32-digit hexadecimal code) of the path of URL visible after you clicked the credential under Jenkins Global credentials. (optional)
- **refspec** (*str*) – refspec to fetch (default '+refs/heads/*:refs/remotes/remoteName/*')
- **name** (*str*) – name to fetch (default 'origin')
- **remotes** (*list(str)*) – list of remotes to set up (optional, only needed if multiple remotes need to be set up)

  Remote

  - **url** (*string*) - url of remote repo
  - **refspec** (*string*) - refspec to fetch (optional)
  - **credentials-id** - ID of credential to use to connect, which is the last field of the path of URL (a 32-digit hexadecimal code) visible after you clicked credential under Jenkins Global credentials. (optional)
- **branches** (*list(str)*) – list of branch specifiers to build (default '**')
- **skip-tag** (*bool*) – Skip tagging (default true)

  Deprecated since version 2.0.0.: Please use per-build-tag extension, which has the inverse meaning.
- **clean** (*bool*) – Clean after checkout (default false)

  Deprecated since version 1.1.1.: Please use clean extension format.
- **fastpoll** (*bool*) – Use fast remote polling (default false)
- **disable-submodules** (*bool*) – Disable submodules (default false)

  Deprecated since version 1.1.1.: Please use submodule extension.
- **recursive-submodules** (*bool*) – Recursively update submodules (default false)

  Deprecated since version 1.1.1.: Please use submodule extension.

- **git-tool** (*str*) – The name of the Git installation to use (default 'Default')
- **reference-repo** (*str*) – Path of the reference repo to use during clone (optional)
- **browser** (*str*) – what repository browser to use.

  browsers supported
  - **auto** - (default)
  - **assemblaweb** - https://www.assembla.com/home
  - **bitbucketweb** - https://bitbucket.org/
  - **cgit** - https://git.zx2c4.com/cgit/about/
  - **fisheye** - https://www.atlassian.com/software/fisheye
  - **gitblit** - http://gitblit.com/
  - **githubweb** - https://github.com/
  - **gitiles** - https://code.google.com/archive/p/gitiles/
  - **gitlab** - https://about.gitlab.com/
  - **gitlist** - http://gitlist.org/
  - **gitoriousweb** - https://gitorious.org/
  - **gitweb** - https://git-scm.com/docs/gitweb
  - **kiln** - https://www.fogbugz.com/version-control
  - **microsoft-tfs-2013** - https://azure.microsoft.com/en-us/services/devops/server
  - **phabricator** - https://www.phacility.com/
  - **redmineweb** - https://www.redmine.org/
  - **rhodecode** - https://rhodecode.com/
  - **stash** - https://www.atlassian.com/software/bitbucket/enterprise/data-center
  - **viewgit**
- **browser-url** (*str*) – url for the repository browser (required if browser is not 'auto', no default)
- **browser-version** (*str*) – version of the repository browser (GitLab only, default '0.0')
- **project-name** (*str*) – project name in Gitblit and ViewGit repobrowser (optional)
- **repo-name** (*str*) – repository name in phabricator repobrowser (optional)
- **git-config-name** (*str*) – Configure name for Git clone (optional)
- **git-config-email** (*str*) – Configure email for Git clone (optional)

Extensions

- **basedir (*string*) - Location relative to the workspace root to** clone to (default workspace)
- **changelog-against (*dict*)**
  - **remote** (*string*) - name of repo that contains branch to create changelog against (default 'origin')
  - **branch** (*string*) - name of the branch to create changelog against (default 'master')
- **choosing-strategy: (*string*) - Jenkins class for selecting what** to build. Can be one of *default*,`inverse`, or *gerrit* (default 'default')
- **clean (*dict*)**
  - **after (*dict*) - Clean the workspace after checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
  - **before (*dict*) - Clean the workspace before checkout**
    - **remove-stale-nested-repos** (*bool*) - Deletes untracked submodules and any other subdirectories which contain .git directories (default false)
- **committer (*dict*)**
  - **name** (*str*) - Name to use as author of new commits

  - **email** (*str*) - E-mail address to use for new commits
- **excluded-users: (*list(string)*) - list of users to ignore**
  revisions from when polling for changes. (if polling is enabled, optional)
- **included-regions: (*list(string)*) - list of file/folders to**
  include (optional)
- **excluded-regions: (*list(string)*) - list of file/folders to**
  exclude (optional)
- **ignore-commits-with-messages (*list(str)*) - Revisions committed**
  with messages matching these patterns will be ignored. (optional)
- **ignore-notify: (*bool*) - Ignore notifyCommit URL accesses**
  (default false)
- **force-polling-using-workspace (*bool*) - Force polling using**
  workspace (default false)
- **local-branch (*string*) - Checkout/merge to local branch**
  (optional)
- **merge (*dict*)**
  - **remote** (*string*) - name of repo that contains branch to merge to (default 'origin')
  - **branch** (*string*) - name of the branch to merge to
  - **strategy** (*string*) - merge strategy. Can be one of 'default', 'resolve', 'recursive', 'octopus', 'ours', 'subtree'. (default 'default')
  - **fast-forward-mode** (*string*) - merge fast-forward mode. Can be one of 'FF', 'FF_ONLY' or 'NO_FF'. (default 'FF')
- **per-build-tag (*bool*) - Create a tag in the workspace for every**
  build. (default is inverse of skip-tag if set, otherwise false)
- **prune** (*bool*) - Prune remote branches (default false)
- **scm-name (*string*) - The unique scm name for this Git SCM**
  (optional)
- **shallow-clone** (*bool*) - Perform shallow clone (default false)
- **depth** (*int*) - Set shallow clone depth (default 1)
- **do-not-fetch-tags (*bool*) - Perform a clone without tags**
  (default false)
- **honor-refspec (*bool*) - Perform initial clone using the refspec**
  defined for the repository (default false)
- **skip-notifications (*bool*) - Skip build status notifications**
  (default false). Requires the Jenkins Skip Notifications Trait Plugin.
- **sparse-checkout (*dict*)**
  - **paths** (*list*) - List of paths to sparse checkout. (optional)
- **submodule (*dict*)**
  - **disable** (*bool*) - By disabling support for submodules you can still keep using basic git plugin functionality and just have Jenkins to ignore submodules completely as if they didn't exist.
  - **recursive** (*bool*) - Retrieve all submodules recursively (uses '–recursive' option which requires git>=1.6.5)
  - **tracking** (*bool*) - Retrieve the tip of the configured branch in .gitmodules (Uses '--remote' option which requires git>=1.8.2)
  - **parent-credentials** (*bool*) - Use credentials from default remote of parent repository (default false).
  - **reference-repo** (*str*) - Path of the reference repo to use during clone (optional)
  - **timeout** (*int*) - Specify a timeout (in minutes) for submodules operations (default 10).
  - **threads** (*int*) - Number of parallel processes to be used when updating submodules. Default is to use a single thread for submodule updates.

- **timeout** (*str*) - Timeout for git commands in minutes (optional)
- **use-author (*bool*): Use author rather than committer in Jenkin's** build changeset (default false)
- **wipe-workspace (*bool*) - Wipe out workspace before build** (default true)
- **lfs-pull (*bool*) - Call git lfs pull after checkout** (default false)

Example:

```
scm:
  - git:
      url: https://example.com/project.git
      branches:
        - master
        - stable
      browser: githubweb
      browser-url: http://github.com/foo/example.git
      timeout: 20
```

hg()

Specifies the mercurial SCM repository for this job.

Requires the Jenkins Mercurial Plugin.

Parameters

- **url** (*str*) – URL of the hg repository (required)
- **credentials-id** (*str*) – ID of credentials to use to connect (optional)
- **revision-type** (*str*) – revision type to use (default 'branch')
- **revision** (*str*) – the branch or tag name you would like to track (default 'default')
- **modules** (*list(str)*) – reduce unnecessary builds by specifying a list of "modules" within the repository. A module is a directory name within the repository that this project lives in. (default '')
- **clean** (*bool*) – wipe any local modifications or untracked files in the repository checkout (default false)
- **subdir** (*str*) – check out the Mercurial repository into this subdirectory of the job's workspace (optional)
- **disable-changelog** (*bool*) – do not calculate the Mercurial changelog for each build (default false)
- **browser** (*str*) – what repository browser to use

  browsers supported

  - **auto** - (default)
  - **bitbucketweb** - https://bitbucket.org/
  - **fisheye** - https://www.atlassian.com/software/fisheye
  - **googlecode** - https://code.google.com/
  - **hgweb** - https://www.mercurial-scm.org/repo/hg/help/hgweb
  - **kilnhg** - https://www.fogbugz.com/version-control
  - **rhodecode** - https://rhodecode.com/ (versions >= 1.2)
  - **rhodecode-pre-1.2.0** - https://rhodecode.com/ (versions < 1.2)
- **browser-url** (*str*) – url for the repository browser (required if browser is set)

Example:

```
scm:
  - hg:
     revision: feature
```

(continues on next page)

(continued from previous page)

```
    url: ssh://hg@hg/repo
    credentials-id: "abcdef01234567890"
    modules:
      - module1
      - module2
    clean: true
    subdir: my/sources
    disable-changelog: true
    browser: hgweb
    browser-url: http://hg/repo
```

openshift-img-streams()

Rather than a Build step extension plugin, this is an extension of the Jenkins SCM plugin, where this baked-in polling mechanism provided by Jenkins is leveraged by exposing some of the common semantics between OpenShift ImageStreams (which are abstractions of Docker repositories) and SCMs - versions / commit IDs of related artifacts (images vs. programmatics files)

Requires the Jenkins OpenShift Pipeline Plugin.

Parameters

- **image-stream-name** (*str*) – The name of the ImageStream is what shows up in the NAME column if you dump all the ImageStream's with the *oc get is* command invocation. (default nodejs-010-centos7)
- **tag** (*str*) – The specific image tag within the ImageStream to monitor. (default latest)
- **api-url** (*str*) – This would be the value you specify if you leverage the –server option on the OpenShift *oc* command. (default https://openshift.default.svc.cluster.local)
- **namespace** (*str*) – The value here should be whatever was the output form *oc project* when you created the BuildConfig you want to run a Build on. (default test)
- **auth-token** (*str*) – The value here is what you supply with the –token option when invoking the OpenShift *oc* command. (default '')
- **verbose** (*bool*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (default false)

Full Example:

```
scm:
  - openshift-img-streams:
      image-stream-name: nodejs-010-fedora
      tag: prod
      api-url: https://openshift.example.local.url/
      namespace: test-scm
      auth-token: ose-key-img-streams1
      verbose: true
```

Minimal Example:

```
scm:
  - openshift-img-streams
```

p4()

Specifies the Perforce (P4) repository for this job.

Requires the Jenkins P4 Plugin.

repo()

Specifies the repo SCM repository for this job.

Requires the Jenkins Repo Plugin.

Parameters

- **manifest-url** (*str*) – URL of the repo manifest (required)
- **manifest-branch** (*str*) – The branch of the manifest to use (optional)
- **manifest-file** (*str*) – Initial manifest file to use when initialising (optional)
- **manifest-group** (*str*) – Only retrieve those projects in the manifest tagged with the provided group name (optional)
- **ignore-projects** (*list(str)*) – a list of projects in which changes would not be considered to trigger a build when pooling (optional)
- **destination-dir** (*str*) – Location relative to the workspace root to clone under (optional)
- **repo-url** (*str*) – custom url to retrieve the repo application (optional)
- **mirror-dir** (*str*) – Path to mirror directory to reference when initialising (optional)
- **jobs** (*int*) – Number of projects to fetch simultaneously (default 0)
- **depth** (*int*) – Specify the depth in history to sync from the source. The default is to sync all of the history. Use 1 to just sync the most recent commit (default 0)
- **current-branch** (*bool*) – Fetch only the current branch from the server (default true)
- **reset-first** (*bool*) – Remove any commits that are not on the repositories by running the following command before anything else (default false): `repo forall -c "git reset --hard"`
- **quiet** (*bool*) – Make repo more quiet (default true)
- **force-sync** (*bool*) – Continue sync even if a project fails to sync (default false)
- **no-tags** (*bool*) – Don't fetch tags (default false)
- **trace** (*bool*) – Trace git command execution into the build logs. (default false)
- **show-all-changes** (*bool*) – When this is checked –first-parent is no longer passed to git log when determining changesets (default false)
- **local-manifest** (*str*) – Contents of .repo/local_manifest.xml, written prior to calling sync (optional)

Example:

```
scm:
  - repo:
      manifest-url: https://example.com/project/
      manifest-branch: stable
      manifest-file: repo.xml
      manifest-group: drivers
      ignore-projects:
        - static-project
        - unimportant-project
      destination-dir: build
      repo-url: https://internal.net/projects/repo
      mirror-dir: ~/git/project/
      jobs: 3
      current-branch: false
      reset-first: true
      quiet: false
      force-sync: true
      no-tags: true
      trace: true
      show-all-changes: true
      local-manifest: |
        <?xml version="1.0" encoding="UTF-8"?>
        <manifest>
```

(continues on next page)

(continued from previous page)

```
        <project path="external/project" name="org/project"
          remote="gerrit" revision="master" />
      </manifest>
```

store()

Specifies the Visualworks Smalltalk Store repository for this job.

Requires the Jenkins Visualworks Smalltalk Store Plugin.

Parameters

- **script** (*str*) – name of the Store script to run
- **repository** (*str*) – name of the Store repository
- **version-regex** (*str*) – regular expression that specifies which pundle versions should be considered (optional)
- **minimum-blessing** (*str*) – minimum blessing level to consider (optional)
- **parcel-builder-file** (*str*) – name of the file to generate as input to a later parcel building step (optional - if not specified, then no parcel builder file will be generated)
- **pundles** (*list*) –

  (package or bundle)

  (*dict*): A package or bundle to check

Example:

```
scm:
  - store:
      script: someStoreScript
      repository: StoreRepository
      version-regex: "[0-9]+"
      minimum-blessing: Integrated
      parcel-builder-file: parcelBuilderInput
      pundles:
        - package: SomePackage
        - package: AnotherPackage
        - bundle: SomeBundle
```

svn()

Specifies the svn SCM repository for this job.

Parameters

- **url** (*str*) – URL of the svn repository
- **basedir** (*str*) – location relative to the workspace root to checkout to (default '.')
- **credentials-id** (*str*) – optional argument to specify the ID of credentials to use
- **repo-depth** (*str*) – Repository depth. Can be one of 'infinity', 'empty', 'files', 'immediates' or 'unknown'. (default 'infinity')
- **ignore-externals** (*bool*) – Ignore Externals. (default false)
- **workspaceupdater** (*str*) – optional argument to specify
- **workspaceupdater** – optional argument to specify how to update the workspace (default wipeworkspace)

  supported values

  - **wipeworkspace** - deletes the workspace before checking out
  - **revertupdate** - do an svn revert then an svn update
  - **emulateclean** - delete unversioned/ignored files then update
  - **update** - do an svn update as much as possible
- **excluded-users** (*list(str)*) – list of users to ignore revisions from when polling for changes (if polling is enabled; parameter is optional)
- **included-regions** (*list(str)*) – list of file/folders to include (optional)

- **excluded-regions** (*list(str)*) – list of file/folders to exclude (optional)
- **excluded-commit-messages** (*list(str)*) – list of commit messages to exclude (optional)
- **exclusion-revprop-name** (*str*) – revision svn-property to ignore (optional)
- **ignore-property-changes-on-directories** (*bool*) – ignore svn-property only changes of directories (default false)
- **filter-changelog** (*bool*) – If set Jenkins will apply the same inclusion and exclusion patterns for displaying changelog entries as it does for polling for changes (default false)
- **repos** (*list*) – list of repositories to checkout (optional)
- **additional-credentials** (*list*) – list of additional credentials (optional) :Additional-Credentials:
  - **realm** (*str*) – realm to use
  - **credentials-id** (*str*) – optional ID of credentials to use
- **viewvc-url** (*str*) – URL of the svn web interface (optional)

  Repo
  - **url** (*str*) – URL for the repository
  - **basedir** (*str*) – Location relative to the workspace root to checkout to (default '.')
  - **credentials-id** - optional ID of credentials to use
  - **repo-depth** - Repository depth. Can be one of 'infinity', 'empty', 'files', 'immediates' or 'unknown'. (default 'infinity')
  - **ignore-externals** - Ignore Externals. (default false)

Multiple repos example:

```
scm:
  - svn:
      workspaceupdater: update
      repos:
        - url: http://svn.example.com/repo
          basedir: .
          credentials-id: "abcdef01234567890"
          repo-depth: files
          ignore-externals: true
        - url: http://svn.example.com/repo2
          basedir: repo2
```

Advanced commit filtering example:

```
scm:
  - svn:
      url: http://svn.apache.org/repos/asf/spamassassin/trunk
      credentials-id: "abcdef01234567890"
      repo-depth: empty
      ignore-externals: true
      workspaceupdater: wipeworkspace
      included-regions:
          - /region1/.*\.cpp
          - /region2
      excluded-regions:
          - /region3/.*\.jpg
          - /region4
      excluded-users:
          - user1
```

(continues on next page)

(continued from previous page)

```
        - user2
    excluded-commit-messages:
        - test-msg
        - test-msg2
    exclusion-revprop-name: propname
    filter-changelog: true
    ignore-property-changes-on-directories: true
    viewvc-url: http://svn.apache.org/viewvc/spamassassin/trunk
```

`tfs()`

Specifies the Team Foundation Server repository for this job.

Requires the Jenkins Team Foundation Server Plugin (https://github.com/jenkinsci/tfs-plugin).

NOTE: TFS Password must be entered manually on the project if a user name is specified. The password will be overwritten with an empty value every time the job is rebuilt with Jenkins Job Builder.

Parameters

- **server-url** (*str*) – The name or URL of the team foundation server. If the server has been registered on the machine then it is only necessary to enter the name.
- **project-path** (*str*) – The name of the project as it is registered on the server.
- **login** (*str*) – The user name that is registered on the server. The user name must contain the name and the domain name. Entered as domain\\user or user@domain (optional). **NOTE**: You must enter in at least two slashes for the domain\\user format in JJB YAML. It will be rendered normally.
- **use-update** (*str*) – If true, Hudson will not delete the workspace at end of each build. This causes the artifacts from the previous build to remain when a new build starts. (default true)
- **local-path** (*str*) – The folder where all files will be retrieved into. The folder name is a relative path, under the workspace of the current job. (default .)
- **workspace** (*str*) – The name of the workspace under which the source should be retrieved. This workspace is created at the start of a download, and deleted at the end. You can normally omit the property unless you want to name a workspace to avoid conflicts on the server (i.e. when you have multiple projects on one server talking to a Team Foundation Server). (default Hudson-${JOB_NAME}-${NODE_NAME})

  The TFS plugin supports the following macros that are replaced in the workspace name:
  - ${JOB_NAME} - The name of the job.
  - **${USER_NAME} - The user name that the Hudson server or slave is** running as.
  - **${NODE_NAME} - The name of the node/slave that the plugin currently** is executed on. Note that this is not the hostname, this value is the Hudson configured name of the slave/node.
  - ${ENV} - The environment variable that is set on the master or slave.
- **web-access** (*dict*) – Adds links in "changes" views within Jenkins to an external system for browsing the details of those changes. The "Auto" selection attempts to infer the repository browser from other jobs, if supported by the SCM and a job with matching SCM details can be found. (optional, default Auto).

  web-access value
  - **web-url** – Enter the URL to the TSWA server. The plugin will strip the last path (if any) of the URL when building URLs for change set pages and other pages. (optional, default uses server-url)

Examples:

```
scm:
  - tfs:
      server-url: "tfs.company.com"
      project-path: "$/myproject"
      login: "mydomain\\\\jane"
      use-update: false
      local-path: "../foo/"
      workspace: "Hudson-${JOB_NAME}"
      web-access:
        - web-url: "http://TFSMachine:8080"
```

```
scm:
  - tfs:
      server-url: "tfs.company.com"
      project-path: "$/myproject"
      login: "jane@mydomain"
      use-update: false
      local-path: "../foo/"
      workspace: "Hudson-${JOB_NAME}"
      web-access:
```

url()

Watch for changes in, and download an artifact from a particular url.

Requires the Jenkins URL SCM.

Parameters

- **url-list** (*list*) – List of URLs to watch. (required)
- **clear-workspace** (*bool*) – If set to true, clear the workspace before downloading the artifact(s) specified in url-list. (default false)

Examples:

```
scm:
  - url:
      url-list:
        - 'http://jenkins.domain.local/jnlpJars/jenkins-cli.jar'
```

```
scm:
  - url:
      url-list:
        - 'http://jenkins.domain.local/jnlpJars/jenkins-cli.jar'
        - 'http://code.jquery.com/jquery-1.11.3.min.js'
      clear-workspace: true
```

workspace()

Specifies the cloned workspace for this job to use as a SCM source.

Requires the Jenkins Clone Workspace SCM Plugin.

The job the workspace is cloned from must be configured with an clone-workspace publisher

Parameters

- **parent-job** (*str*) – The name of the parent job to clone the workspace from.
- **criteria** (*str*) – Set the criteria to determine what build of the parent project to use. Can be one of 'Any', 'Not Failed' or 'Successful'. (default Any)

Example:

```
scm:
  - workspace:
      parent-job: my-upstream-job
      criteria: Any
```

Triggers

Triggers define what causes a Jenkins job to start building.

Component: triggers

Macro
trigger

Entry Point
jenkins_jobs.triggers

Example:

```
job:
  name: test_job

  triggers:
    - timed: '@daily'
```

class triggers.**Triggers**(*registry*)

component_list_type = 'triggers'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'trigger'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 50

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

artifactory()

Artifactory trigger. Trigger if files are added or modified in configured path(s) to watch on chosen Artifactory server.

Requires the Jenkins Artifactory Plugin.

Parameters

- **artifactory-server** (*str*) – Artifactory server where the configured path(s) are monitored from. Available Artifactory servers must be configured on Jenkins Global Configuration in advance. (default '')
- **schedule** (*str*) – cron syntax of when to poll. (default '')
- **paths** (*str*) – Paths in Artifactory to poll for changes. Multiple paths can be configured by the ';' separator. (default '')

Example with Single Path to Monitor:

```
triggers:
  - artifactory:
        artifactory-server: my-artifactory
        schedule: H/15 * * * *
        paths: myrepo/myfolder/latest
```

Example with Multiple Paths to Monitor:

```
triggers:
  - artifactory:
        artifactory-server: my-artifactory
        schedule: H 0,8,16 * * *
        paths: myrepo/myfolder/latest;myrepo/myfolder/commit
```

bitbucket()

Trigger a job when bitbucket repository is pushed to.

Requires the Jenkins BitBucket Plugin.

Example:

```
triggers:
  - bitbucket
```

build-result()

Configure jobB to monitor jobA build result. A build is scheduled if there is a new build result that matches your criteria (unstable, failure, ...).

Requires the Jenkins BuildResultTrigger Plugin.

Parameters

- **groups** (*list*) – List groups of jobs and results to monitor for
- **jobs** (*list*) – The jobs to monitor (required)
- **results** (*list*) – Build results to monitor for (default success)
- **combine** (*bool*) – Combine all job information. A build will be scheduled only if all conditions are met (default false)
- **cron** (*str*) – The cron syntax with which to poll the jobs for the supplied result (default '')

Full Example:

```
triggers:
  - build-result:
      cron: H/15 * * * *
      combine: true
      groups:
        - jobs:
            - test
            - test2
```

(continues on next page)

(continued from previous page)

```
          results:
            - success
            - not-built
        - jobs:
            - test3
          results:
            - unstable
        # Use default results (success):
        - jobs:
            - test4
```

Minimal Example:

```
triggers:
  - build-result:
      groups:
       - jobs:
           - test
         results:
           - aborted
```

dockerhub-notification()

The job will get triggered when Docker Hub/Registry notifies that Docker image(s) used in this job has been rebuilt.

Requires the Jenkins CloudBees Docker Hub Notification.

Parameters

- **referenced-image** (*bool*) – Trigger the job based on repositories used by any compatible docker plugin in this job. (default true)
- **repositories** (*list*) – Specified repositories to trigger the job. (default [])

Minimal Example:

```
triggers:
  - dockerhub-notification
```

Full Example:

```
triggers:
  - dockerhub-notification:
      referenced-image: true
      repositories:
        - repo1
        - repo2
```

generic-webhook-trigger()

Generic webhook trigger. Trigger when a set of parameters are submitted.

Requires the Jenkins Generic Webhook Trigger.

Parameters

- **token** (*str*) – A token to use to trigger the job. (default '')
- **token-credential-id** (*str*) – A token credential id to use to trigger the job. (default '')
- **print-post-content** (*bool*) – Print post content in job log.
- **print-contrib-var** (*bool*) – Print contributed variables in job log.

- **silent-response** (*bool*) – Avoid responding with information about triggered jobs.
- **cause** (*str*) – This will be displayed in any triggered job.
- **regex-filter-expression** (*str*) – Regular expression to test on the evaluated text specified in regex-filter-text
- **regex-filter-text** (*str*) – Text to test for the given regexp-filter-expression.
- **post-content-params** (*list*) – Parameters to use from posted JSON/XML
  post-content-params
  - **type** (*str*) – JSONPath or XPath
  - **key** (*str*) – Variable name
  - **value** (*str*) – Expression to evaluate in POST content. Use JSONPath for JSON or XPath for XML.
  - **regex-filter** (*str*) – Anything in the evaluated value, matching this regular expression, will be removed. (optional)
  - **default-value** (*str*) – This value will be used if expression does not match anything. (optional)
- **request-params** (*list*) – Parameters to use passed in as request arguments
  request-params
  - **key** (*str*) – Name of request parameter
  - **regex-filter** (*str*) – Anything in the evaluated value, matching this regular expression, will be removed. (optional)
- **header-params** (*list*) – Parameters to use passed in as headers
  header-params
  - **key** (*str*) – Name of request header in lowercase. Resulting variable name has '_' instead of '-' characters.
  - **regex-filter** (*str*) – Anything in the evaluated value, matching this regular expression, will be removed. (optional)

Example:

```
triggers:
  - generic-webhook-trigger:
        post-content-params:
            - type: JSONPath
              key: action
              value: $.action
              regex-filter: value_filter
              default-value: default_value
            - type: XPath
              key: blah
              value: whooga
              regex-filter: value_filer
              default-value: default_something
        regex-filter-text: $action
        regex-filter-expression: ^(opened|reopened|synchronize)$
        request-params:
            - key: request_test_1
              regex-filter: requesT_value_1
        header-params:
            - key: header_test1
              regex-filter: header_value1
        print-post-content: true
        print-contrib-var: true
        cause: Generic Cause
        token: blah
```

(continues on next page)

(continued from previous page)

```
        token-credential-id: token_credential_id
        silent-response: true
```

gerrit()

Trigger on a Gerrit event.

Requires the Jenkins Gerrit Trigger Plugin version >= 2.6.0.

Parameters

- **trigger-on** (*list*) – Events to react on. Please use either the new **trigger-on**, or the old **trigger-on-*** events definitions. You cannot use both at once.

  Trigger on

  - **patchset-created-event** (*dict*) – Trigger upon patchset creation.

    Patchset created

    - **exclude-drafts** (*bool*) – exclude drafts (default false)
    - **exclude-trivial-rebase** (*bool*) – exclude trivial rebase (default false)
    - **exclude-no-code-change** (*bool*) – exclude no code change (default false)
    - **exclude-private** (*bool*) – exclude private change (default false)
    - **exclude-wip** (*bool*) – exclude wip change (default false)
    - **commit-message-contains-regex** (*str*) – Commit message contains regular expression. (default '') Requires Gerrit Trigger Plugin >= 2.32.0

    exclude-private|exclude-wip needs Gerrit Trigger v2.29.0 Exclude drafts|trivial-rebase|no-code-change needs Gerrit Trigger v2.12.0
  - **patchset-uploaded-event** – Trigger upon patchset creation (this is a alias for *patchset-created-event*).

    Deprecated since version 1.1.0: Please use *trigger-on*.
  - **change-abandoned-event** – Trigger on patchset abandoned. Requires Gerrit Trigger Plugin version >= 2.8.0.
  - **change-merged-event** – Trigger on change merged
  - **change-restored-event** – Trigger on change restored. Requires Gerrit Trigger Plugin version >= 2.8.0
  - **draft-published-event** – Trigger on draft published event.
  - **ref-updated-event** – Trigger on ref-updated. Gerrit Trigger Plugin version >= 2.29.0
  - **topic-changed-event** – Trigger on topic-changed. Gerrit Trigger Plugin version >= 2.26.0
  - **private-state-changed-event** – Trigger on private state changed event.

  - **wip-state-changed-event** – Trigger on wip state changed event. Gerrit Trigger Plugin version >= 2.8.0
  - **comment-added-event** (*dict*) – Trigger on comment added.
    Comment added
    - **approval-category** (*str*) – Approval (verdict) category (for example 'APRV', 'CRVW', 'VRIF' – see Gerrit access control
    - **approval-value** – Approval value for the comment added.
  - **comment-added-contains-event** (*dict*) – Trigger on comment added contains Regular Expression.
    Comment added contains
    - **comment-contains-value** (*str*) – Comment contains Regular Expression value.
- **trigger-on-patchset-uploaded-event** (*bool*) – Trigger on patchset upload.

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-on-change-abandoned-event** (*bool*) – Trigger on change abandoned. Requires Gerrit Trigger Plugin version >= 2.8.0

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-on-change-merged-event** (*bool*) – Trigger on change merged

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-on-change-restored-event** (*bool*) – Trigger on change restored. Requires Gerrit Trigger Plugin version >= 2.8.0

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-on-comment-added-event** (*bool*) – Trigger on comment added

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-on-draft-published-event** (*bool*) – Trigger on draft published event

  Deprecated since version 1.1.0: Please use *trigger-on*.
- **trigger-on-ref-updated-event** (*bool*) – Trigger on ref-updated

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-approval-category** (*str*) – Approval category for comment added

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **trigger-approval-value** (*int*) – Approval value for comment added

  Deprecated since version 1.1.0.: Please use *trigger-on*.
- **override-votes** (*bool*) – Override default vote values
- **gerrit-build-started-verified-value** (*int*) – Started ''Verified" value
- **gerrit-build-successful-verified-value** (*int*) – Successful ''Verified" value
- **gerrit-build-failed-verified-value** (*int*) – Failed ''Verified" value
- **gerrit-build-unstable-verified-value** (*int*) – Unstable ''Verified" value
- **gerrit-build-notbuilt-verified-value** (*int*) – Not built ''Verified" value
- **gerrit-build-aborted-verified-value** (*int*) – Aborted ''Verified" value Requires Gerrit Trigger Plugin version >= 2.31.0
- **gerrit-build-started-codereview-value** (*int*) – Started ''CodeReview" value
- **gerrit-build-successful-codereview-value** (*int*) – Successful ''CodeReview" value
- **gerrit-build-failed-codereview-value** (*int*) – Failed ''CodeReview" value

- **gerrit-build-unstable-codereview-value** (*int*) – Unstable ''CodeReview" value
- **gerrit-build-notbuilt-codereview-value** (*int*) – Not built ''CodeReview" value
- **gerrit-build-aborted-codereview-value** (*int*) – Aborted ''CodeReview" value Requires Gerrit Trigger Plugin version >= 2.31.0
- **failure-message** (*str*) – Message to leave on failure (default '')
- **successful-message** (*str*) – Message to leave on success (default '')
- **unstable-message** (*str*) – Message to leave when unstable (default '')
- **notbuilt-message** (*str*) – Message to leave when not built (default '')
- **aborted-message** (*str*) – Message to leave when aborted (default '')
- **failure-message-file** (*str*) – Sets the filename within the workspace from which to retrieve the unsuccessful review message. (optional)
- **projects** (*list*) – list of projects to match

  Project

  - **project-compare-type** (*str*) – ''PLAIN", ''ANT" or ''REG_EXP"
  - **project-pattern** (*str*) – Project name pattern to match
  - **branch-compare-type** (*str*) – ''PLAIN", ''ANT" or ''REG_EXP" (not used if *branches* list is specified)

    Deprecated since version 1.1.0: Please use *branches*.
  - **branch-pattern** (*str*) – Branch name pattern to match (not used if *branches* list is specified)

    Deprecated since version 1.1.0: Please use *branches*.
  - **branches** (*list*) – List of branches to match (optional)

    Branch

    * **branch-compare-type** (*str*) – ''PLAIN", ''ANT" or ''REG_EXP" (optional) (default ''PLAIN")
    * **branch-pattern** (*str*) – Branch name pattern to match
  - **file-paths** (*list*) – List of file paths to match (optional)

    File Path

    * **compare-type** (*str*) – ''PLAIN", ''ANT" or ''REG_EXP" (optional) (default ''PLAIN")
    * **pattern** (*str*) – File path pattern to match
  - **forbidden-file-paths** (*list*) – List of file paths to skip triggering (optional)

    Forbidden File Path

    * **compare-type** (*str*) – ''PLAIN", ''ANT" or ''REG_EXP" (optional) (default ''PLAIN")
    * **pattern** (*str*) – File path pattern to match
  - **topics** (*list*) – List of topics to match (optional)

    Topic

    * **compare-type** (*str*) – ''PLAIN",

''ANT" or ''REG_EXP" (optional) (default ''PLAIN")

* **pattern** (*str*) – Topic name pattern to match

– **disable-strict-forbidden-file-verification** (***bool***) – Enabling this option will allow an event to trigger a build if the event contains BOTH one or more wanted file paths AND one or more forbidden file paths. In other words, with this option, the build will not get triggered if the change contains only forbidden files, otherwise it will get triggered. Requires plugin version >= 2.16.0 (default false)

- **skip-vote** (*dict*) – map of build outcomes for which Jenkins must skip vote. Requires Gerrit Trigger Plugin version >= 2.7.0

  Outcome

  – **successful** (*bool*)
  – **failed** (*bool*)
  – **unstable** (*bool*)
  – **notbuilt** (*bool*)
  – **aborted** (*bool*) – Requires Gerrit Trigger Plugin version >= 2.31.0
- **silent** (*bool*) – When silent mode is on there will be no communication back to Gerrit, i.e. no build started/failed/successful approve messages etc. If other non-silent jobs are triggered by the same Gerrit event as this job, the result of this job's build will not be counted in the end result of the other jobs. (default false)
- **silent-start** (*bool*) – Sets silent start mode to on or off. When silent start mode is on there will be no 'build started' messages sent back to Gerrit. (default false)
- **escape-quotes** (*bool*) – escape quotes in the values of Gerrit change parameters (default true)
- **build-cancellation-policy** (*dict*) – If used, rules regarding cancellation of builds can be set with this option when patchsets of the same change comes in. This setting overrides global server configuration. If build-cancellation-policy is not present in YAML the global server configuration is used. Requires Gerrit Trigger Plugin version >= 2.32.0

  Options

  – **abort-new-patchsets** (*bool*) – Only running jobs will be cancelled if a new patch version is pushed over (default false).
  – **abort-manual-patchsets** (*bool*) – Builds triggered manually will be aborted when a new patch set arrives (default false).
  – **abort-same-topic** (*bool*) – Builds triggered with topic will be aborted when a new patch set with the same topic arrives (default false).
- **no-name-and-email** (*bool*) – Do not pass compound 'name and email' parameters (default false)

  Deprecated since version 3.5.0: Please use *name-and-email-parameter-mode* parameter.
- **readable-message** (*bool*) – If parameters regarding multiline text, e.g. commit message, should be as human readable or not. If false, those parameters are Base64 encoded to keep environment variables clean. (default false)

  Deprecated since version 3.5.0: Please use *commit-message-parameter-mode* parameter.
- **name-and-email-parameter-mode** (*str*) – The parameter mode for the compound

"name and email" parameters (like GERRIT_PATCHSET_UPLOADER or GERRIT_CHANGE_OWNER). This can either be 'NONE' to avoid passing the parameter all together, 'PLAIN' to pass the parameter in human readable form, or 'BASE64' to pass the parameter in base64 encoded form (default 'PLAIN'). Requires Gerrit Trigger Plugin version >= 2.18.0.

- **commit-message-parameter-mode** (*str*) – The parameter mode for the GERRIT_CHANGE_COMMIT_MESSAGE parameter. This can either be 'NONE' to avoid passing the parameter all together, 'PLAIN' to pass the parameter in human readable form, or 'BASE64' to pass the parameter in base64 encoded form (default 'BASE64'). Requires Gerrit Trigger Plugin version >= 2.18.0.
- **change-subject-parameter-mode** (*str*) – The parameter mode for the GERRIT_CHANGE_SUBJECT parameter. This can either be 'NONE' to avoid passing the parameter all together, 'PLAIN' to pass the parameter in human readable form, or 'BASE64' to pass the parameter in base64 encoded form (default 'PLAIN'). Requires Gerrit Trigger Plugin version >= 2.18.0.
- **comment-text-parameter-mode** (*str*) – The parameter mode for the GERRIT_EVENT_COMMENT_TEXT parameter. This can either be 'NONE' to avoid passing the parameter all together, 'PLAIN' to pass the parameter in human readable form, or 'BASE64' to pass the parameter in base64 encoded form (default 'BASE64'). Requires Gerrit Trigger Plugin version >= 2.18.0.
- **dependency-jobs** (*str*) – All jobs on which this job depends. If a commit should trigger both a dependency and this job, the dependency will be built first. Use commas to separate job names. Beware of cyclic dependencies. (optional)
- **notification-level** (*str*) – Defines to whom email notifications should be sent. This can either be nobody ('NONE'), the change owner ('OWNER'), reviewers and change owner ('OWNER_REVIEWERS'), all interested users i.e. owning, reviewing, watching, and starring ('ALL') or server default ('SERVER_DEFAULT'). (default 'SERVER_DEFAULT')
- **dynamic-trigger-enabled** (*bool*) – Enable/disable the dynamic trigger (default false)
- **dynamic-trigger-url** (*str*) – if you specify this option, the Gerrit trigger configuration will be fetched from there on a regular interval
- **trigger-for-unreviewed-patches** (*bool*) – trigger patchset-created events for changes that were uploaded while connection to Gerrit was down (default false). Requires Gerrit Trigger Plugin version >= 2.11.0.

  Deprecated since version 3.5.0: Supported for Gerrit Trigger Plugin versions < 2.14.0. See Missed Events Playback Feature.
- **custom-url** (*str*) – Custom URL for a message sent to Gerrit. Build details URL will be used if empty. (default '')
- **server-name** (*str*) – Name of the server to trigger on, or ''__ANY__" to trigger on any configured Gerrit server (default '__ANY__'). Requires Gerrit Trigger Plugin version >= 2.11.0

You may select one or more Gerrit events upon which to trigger. You must also supply at least one project and branch, optionally more. If you select the comment-added trigger, you should also indicate which approval category and value you want to trigger the job.

Until version 0.4.0 of Jenkins Job Builder, camelCase keys were used to configure Gerrit Trigger Plugin, instead of hyphenated-keys. While still supported, camedCase keys are deprecated and should not be used. Support for this will be removed after 1.0.0 is released.

Example:

```
triggers:
  - gerrit:
```

(continues on next page)

(continued from previous page)

```
      trigger-on:
        - patchset-created-event:
            exclude-drafts: true
            exclude-trivial-rebase: true
            exclude-no-code-change: true
            exclude-private: true
            exclude-wip: true
            commit-message-contains-regex: "regex"
        - comment-added-event:
            approval-category: 'APRV'
            approval-value: 1
      projects:
        - project-compare-type: 'PLAIN'
          project-pattern: 'test-project'
          branches:
            - branch-compare-type: 'PLAIN'
              branch-pattern: 'master'
            - branch-compare-type: 'PLAIN'
              branch-pattern: 'stable'
          file-paths:
              - compare-type: ANT
                pattern: subdirectory/**
          topics:
              - compare-type: ANT
                pattern: refactor-xy**
      skip-vote:
          successful: true
          failed: true
          unstable: true
          notbuilt: true
          aborted: true
      build-cancellation-policy:
        abort-new-patchsets: false
        abort-manual-patchsets: true
        abort-same-topic: true
      silent: false
      silent-start: true
      escape-quotes: false
      dependency-jobs: 'job1, job2'
      name-and-email-parameter-mode: PLAIN
      notification-level: ALL
      dynamic-trigger-enabled: true
      dynamic-trigger-url: http://myhost/mytrigger
      server-name: my-server
      failure-message-file: path/to/filename
```

github()

Trigger a job when github repository is pushed to.

Requires the Jenkins GitHub Plugin.

Example:

```
triggers:
  - github
```

github-pull-request()

Build pull requests in github and report results.

Requires the Jenkins GitHub Pull Request Builder Plugin.

Parameters

- **admin-list** (*list*) – the users with admin rights (optional)
- **white-list** (*list*) – users whose pull requests build (optional)
- **org-list** (*list*) – orgs whose users should be white listed (optional)
- **allow-whitelist-orgs-as-admins** (*bool*) – members of white listed orgs will have admin rights. (default false)
- **cron** (*str*) – cron syntax of when to run (optional)
- **trigger-phrase** (*str*) – when filled, commenting this phrase in the pull request will trigger a build (optional)
- **only-trigger-phrase** (*bool*) – only commenting the trigger phrase in the pull request will trigger a build (default false)
- **skip-build-phrase** (*str*) – when filled, adding this phrase to the pull request title or body will not trigger a build (optional)
- **black-list-commit-author** (*list*) – When filled, pull request commits from this user(s) will not trigger a build (optional)
- **black-list-labels** (*str*) – list of GitHub labels for which the build should not be triggered (optional)
- **white-list-labels** (*str*) – list of GitHub labels for which the build should only be triggered. (Leave blank for 'any') (optional)
- **github-hooks** (*bool*) – use github hook (default false)
- **permit-all** (*bool*) – build every pull request automatically without asking (default false)
- **auto-close-on-fail** (*bool*) – close failed pull request automatically (default false)
- **display-build-errors-on-downstream-builds** (*bool*) – Display build errors on downstream builds (default false)
- **white-list-target-branches** (*list*) – Adding branches to this whitelist allows you to selectively test pull requests destined for these branches only. Supports regular expressions (e.g. 'master', 'feature-.*'). (optional)
- **black-list-target-branches** (*list*) – Adding branches to this blacklist allows you to selectively prevent pull requests builds destined for these branches. Supports regular expressions (e.g. 'master', 'feature-.*'). (optional)
- **auth-id** (*str*) – the auth id to use (optional)
- **build-desc-template** (*str*) – the template for build descriptions in jenkins (optional)
- **status-context** (*str*) – the context to include on PR status comments (optional)
- **triggered-status** (*str*) – the status message to set when the build has been triggered (optional)
- **started-status** (*str*) – the status comment to set when the build has been started (optional)
- **status-url** (*str*) – the status URL to set (optional)
- **status-add-test-results** (*bool*) – add test result one-liner to status message (optional)
- **success-status** (*str*) – the status message to set if the job succeeds (optional)
- **failure-status** (*str*) – the status message to set if the job fails (optional)
- **error-status** (*str*) – the status message to set if the job errors (optional)
- **success-comment** (*str*) – comment to add to the PR on a successful job (optional)
- **failure-comment** (*str*) – comment to add to the PR on a failed job (optional)

- **error-comment** (*str*) – comment to add to the PR on an errored job (optional)
- **cancel-builds-on-update** (*bool*) – cancel existing builds when a PR is updated (optional)
- **comment-file** (*str*) – Extends the standard build comment message on github with a custom message file. (optional)
- **no-commit-status** (*bool*) – Enables "Do not update commit status"
- **included-regions** (*list*) – Each inclusion uses regular expression pattern matching, and must be separated by a new line. An empty list implies that everything is included. (optional)
- **excluded-regions** (*list*) – Each exclusion uses regular expression pattern matching, and must be separated by a new line. Exclusions take precedence over inclusions, if there is an overlap between included and excluded regions. (optional)

Full Example:

```yaml
triggers:
  - github-pull-request:
      admin-list:
        - user1
        - user2
      white-list:
        - user3
        - user4
      org-list:
        - org1
        - org2
      white-list-labels:
        - label1
        - label2
      black-list-labels:
        - label3
        - label4
      cron: '* * * * *'
      build-desc-template: "build description"
      trigger-phrase: 'retest this please'
      skip-build-phrase: 'no tests'
      black-list-commit-author:
        - blacklist
        - commit
        - author
      only-trigger-phrase: true
      github-hooks: true
      permit-all: true
      auto-close-on-fail: false
      display-build-errors-on-downstream-builds: true
      allow-whitelist-orgs-as-admins: true
      white-list-target-branches:
        - master
        - testing
      black-list-target-branches:
        - master
        - testing
      auth-id: '123-456-789'
      status-context: "status context"
```

(continues on next page)

(continued from previous page)

```
      triggered-status: "triggered status message"
      started-status: "started"
      status-url: "url/to/status"
      status-add-test-results: false
      success-status: "success message"
      failure-status: "failure message"
      error-status: "error message"
      success-comment: "success comment"
      failure-comment: "failure comment"
      error-comment: "error-comment"
      cancel-builds-on-update: true
      comment-file: "/tmp/path"
      no-commit-status: true
      included-regions:
          - include
          - region
      excluded-regions:
          - exclude
          - region
```

Minimal Example:

```
triggers:
  - github-pull-request
```

gitlab()

Makes Jenkins act like a GitLab CI server.

Requires the Jenkins GitLab Plugin.

Parameters

- **trigger-push** (*bool*) – Build on Push Events (default true)
- **trigger-merge-request** (*bool*) – Build on Merge Request Events (default true)
- **trigger-accepted-merge-request** (*bool*) – Build on Accepted Merge Request Events (>= 1.4.6) (default false)
- **trigger-closed-merge-request** (*bool*) – Build on Closed Merge Request Events (>= 1.4.6) (default false)
- **trigger-open-merge-request-push** (*str*) – Rebuild open Merge Requests on Push Events.
  - **trigger-open-merge-request-push values (< 1.1.26)**
    - **true** (default)
    - **false**
  - **trigger-open-merge-request-push values (>= 1.1.26)**
    - **never** (default)
    - **source**
    - **both**
- **trigger-only-if-new-commits-pushed** (*bool*) – Trigger a build on commits pushed only, but not trigger on another MR changes(label, edit, assign, etc) (>=1.5.17)(default false)
- **trigger-note** (*bool*) – Build when comment is added with defined phrase (>= 1.2.4) (default true)
- **note-regex** (*str*) – Phrase that triggers the build (>= 1.2.4) (default 'Jenkins please retry a build')
- **ci-skip** (*bool*) – Enable skipping builds of commits that contain [ci-skip] in the com-

mit message (default true)
- **wip-skip** (*bool*) – Enable skipping builds of WIP Merge Requests (>= 1.2.4) (default true)
- **set-build-description** (*bool*) – Set build description to build cause (eg. Merge request or Git Push) (default true)
- **cancel-pending-builds-on-update** (*bool*) – Cancel pending merge request builds on update (default false)
- **pending-build-name** (*str*) – Set the pending merge request build name (optional)
- **add-note-merge-request** (*bool*) – Add note with build status on merge requests (default true)
- **add-vote-merge-request** (*bool*) – Vote added to note with build status on merge requests (>= 1.1.27) (default true)
- **accept-merge-request-on-success** (*bool*) – Automatically accept the Merge Request if the build is successful (>= 1.1.27) (default false)
- **add-ci-message** (*bool*) – Add CI build status (1.1.28 - 1.2.0) (default false)
- **allow-all-branches** (*bool*) – Allow all branches (Ignoring Filtered Branches) (< 1.1.29) (default false)
- **branch-filter-type** (*str*) – Filter branches that can trigger a build. Valid values and their additional attributes are described in the *branch filter type* table (>= 1.1.29) (default 'All').
- **include-branches** (*list*) – Defined list of branches to include (default [])
- **exclude-branches** (*list*) – Defined list of branches to exclude (default [])
- **source-branch-regex** (*str*) – Regular expression to select branches
- **target-branch-regex** (*str*) – Regular expression to select branches
- **secret-token** (*str*) – Secret token for build trigger
- **merge-request-label-filter-config** (*dict*) – If used allow merge requests filtering by labels

  Options
  - **include** (*str*) Run for specified labels.
  - **exclude** (*str*) Do not run for specified labels.

Branch filter type	Description
All	All branches are allowed to trigger this job.
Name-Based-Filter	Filter branches by name. List source branches that are allowed to trigger a build from a Push event or a Merge Request event. If both fields are left empty, all branches are allowed to trigger this job. For Merge Request events only the target branch name is filtered out by the **include-branches** and **exclude-branches** lists.
RegexBasedFilter	Filter branches by regex The target branch regex allows you to limit the execution of this job to certain branches. Any branch matching the specified pattern in **target-branch-regex** and **source-branch-regex** triggers the job. No filtering is performed if the field is left empty.

Example (version < 1.1.26):

```
triggers:
  - gitlab:
      trigger-push: true
      trigger-merge-request: true
      trigger-open-merge-request-push: true
      ci-skip: true
      set-build-description: true
      add-note-merge-request: true
```

(continues on next page)

(continued from previous page)

```
      add-vote-merge-request: true
      add-ci-message: true
      allow-all-branches: true
      include-branches:
        - 'master'
        - 'master2'
        - 'local-test'
      exclude-branches:
        - 'broken-test'
        - 'master-foo'
```

Minimal example (version >= 1.1.26):

```
triggers:
  - gitlab
```

Full example (version >= 1.1.26):

```
triggers:
  - gitlab:
      trigger-push: false
      trigger-merge-request: false
      trigger-only-if-new-commits-pushed: false
      trigger-open-merge-request-push: both
      ci-skip: false
      set-build-description: false
      add-note-merge-request: false
      add-vote-merge-request: false
      add-ci-message: true
      allow-all-branches: true
      include-branches:
        - 'master'
        - 'master2'
        - 'local-test'
      exclude-branches:
        - 'broken-test'
        - 'master-foo'
      pending-build-name: 'test'
```

gitlab-merge-request()

Build merge requests in gitlab and report results.

Requires the Jenkins Gitlab MergeRequest Builder Plugin.

Parameters

- **cron** (*str*) – Cron syntax of when to run (required)
- **project-path** (*str*) – Gitlab-relative path to project (required)
- **target-branch-regex** (*str*) – Allow execution of this job for certain branches only (default ‘’). Requires Gitlab MergeRequest Builder Plugin >= 2.0.0
- **use-http-url** (*str*) – Use the HTTP(S) URL to fetch/clone repository (default false)
- **assignee-filter** (*str*) – Only MRs with this assigned user will trigger the build automatically (default ‘jenkins’)
- **tag-filter** (*str*) – Only MRs with this label will trigger the build automatically (default ‘Build’)

- **trigger-comment** (*str*) – Force build if this comment is the last in merge reguest (default '')
- **publish-build-progress-messages** (*str*) – Publish build progress messages (except build failed) (default true)

  Deprecated since version 2.0.0.
- **auto-close-failed** (*str*) – On failure, auto close the request (default false)
- **auto-merge-passed** (*str*) – On success, auto merge the request (default false)

Example (version < 2.0.0):

```
triggers:
  - gitlab-merge-request:
      cron: '* * * * *'
      project-path: 'test/project'
      use-http-url: false
      assignee-filter: 'jenkinsbot'
      tag-filter: 'fix'
      trigger-comment: 'rebuild'
      publish-build-progress-messages: true
      auto-close-failed: false
      auto-merge-passed: false
```

Example (version >= 2.0.0):

```
triggers:
  - gitlab-merge-request:
      cron: '* * * * *'
      project-path: 'test/project'
      target-branch-regex: '(.*release.*|.*hotfix.*)'
      use-http-url: false
      assignee-filter: 'jenkinsbot'
      tag-filter: 'fix'
      trigger-comment: 'rebuild'
      auto-close-failed: false
      auto-merge-passed: false
```

gogs()

Trigger a job when gogs repository is pushed to.

Requires the Jenkins Gogs Plugin.

Example:

```
triggers:
  - gogs
```

groovy-script()

Triggers the job using a groovy script.

Requires the Jenkins ScriptTrigger Plugin.

Parameters

- **system-script** (*bool*) – If true, run the groovy script as a system script, the script will have access to the same variables as the Groovy Console. If false, run the groovy script on the executor node, the script will not have access to the hudson or job model. (default false)

- **script** (*str*) – Content of the groovy script. If the script result is evaluated to true, a build is scheduled. (default '')
- **script-file-path** (*str*) – Groovy script path. (default '')
- **property-file-path** (*str*) – Property file path. All properties will be set as parameters for the triggered build. (default '')
- **enable-concurrent** (*bool*) – Enable concurrent build. (default false)
- **label** (*str*) – Restrict where the polling should run. (default '')
- **cron** (*str*) – cron syntax of when to run (default '')

Full Example:

```
triggers:
  - groovy-script:
        script: groovy-content
        script-file-path: path/to/filename
        property-file-path: /path/to/properties/file
        cron: H/15 * * * *
        enable-concurrent: true
        label: master
        system-script: true
```

Minimal Example:

```
triggers:
  - groovy-script
```

ivy()

Poll with an Ivy script.

Requires the Jenkins IvyTrigger Plugin.

Parameters

- **path** (*str*) – Path of the ivy file. (optional)
- **settings-path** (*str*) – Ivy Settings Path. (optional)
- **properties-file** (*list*) – List of properties file path. Properties will be injected as variables in the ivy settings file. (optional)
- **properties-content** (*str*) – Properties content. Properties will be injected as variables in the ivy settings file. (optional)
- **debug** (*bool*) – Active debug mode on artifacts resolution. (default false)
- **download-artifacts** – Download artifacts for dependencies to see if they have changed. (default true)
- **enable-concurrent** (*bool*) – Enable Concurrent Build. (default false)
- **label** (*str*) – Restrict where the polling should run. (default '')
- **cron** (*str*) – cron syntax of when to run (default '')

Example:

```
triggers:
  - ivy:
        path: path/to/file
        settings-path: path/to/settings/file
        properties-file:
            - 'filename1'
            - 'filename2'
        debug: true
        cron: 'H/15 * * * *'
        enable-concurrent: False
```

(continues on next page)

(continued from previous page)

```
        label: master
```

jira-changelog()

Sets up a trigger that listens to JIRA issue changes.

Requires the Jenkins JIRA Trigger Plugin.

Parameters

- **jql-filter** (*str*) – Must match updated issues to trigger a build. (default '')
- **changelog-matchers** (*list*) –
  - **Custom Field Matcher**
    - **custom-field-name (*str*) – The custom field** name that has been changed during the issue update. (default '')
    - **compare-new-value (*bool*) – Compare the** new value of the updated field. (default false)
    - **new-value (*str*) – The new value of the updated field.** (default '')
    - **compare-old-value (*bool*) – Compare the** old value of the updated field. (default false)
    - **old-value (*str*) – The value** before the field is updated. (default '')
  - **JIRA Field Matcher**
    - **jira-field-ID (*str*) – The JIRA Field ID that** has been changed during the issue update. (default '')
    - **compare-new-value (*bool*) – Compare the new value** of the updated field. (default false)
    - **new-value (*str*) – The new value of the updated field.** (default '')
    - **compare-old-value (*bool*) – Compare the old value** of the updated field. (default false)
    - **old-value (*str*) – The value before** the field is updated. (default '')
- **parameter-mapping** (*list*) –
  - **Issue Attribute Path**
    - **jenkins-parameter (*str*) – Jenkins parameter name** (default '')
    - **issue-attribute-path** (*str*) – Attribute path (default '')

Minimal Example:

```
triggers:
  - jira-changelog
```

Full Example:

```
triggers:
  - jira-changelog:
      jql-filter: filter
      changelog-matchers:
          - field-type: 'CUSTOM'
            field: name
            new-value: val1
            old-value: val2
            compare-new-value: true
```

(continues on next page)

(continued from previous page)

```
            compare-old-value: true
          - field-type: 'JIRA'
            field: versions
            new-value: val3
            old-value: val4
            compare-new-value: true
            compare-old-value: true
      parameter-mapping:
          - jenkins-parameter: param
            issue-attribute-path: path
```

jira-comment-trigger()

Trigger builds when a comment is added to JIRA.

Requires the Jenkins JIRA Trigger Plugin.

Parameters

- **jql-filter** (*str*) – Must match updated issues to trigger a build. (default '')
- **comment-pattern** (*str*) – Triggers build only when the comment added to JIRA matches pattern (default '(?i)build this please')
- **parameter-mapping** (*list*) –

  Issue Attribute Path

  - **jenkins-parameter (*str*) – Jenkins parameter name** (default '')
  - **issue-attribute-path** (*str*) – Attribute path (default '')

Minimal Example:

```
triggers:
  - jira-comment-trigger
```

Full Example:

```
triggers:
  - jira-comment-trigger:
      jql-filter: filter
      comment-pattern: comment
      parameter-mapping:
        - jenkins-parameter: param1
          issue-attribute-path: 'path/to/attribute'
```

jms-messaging()

The JMS Messaging Plugin provides the following functionality:

- A build trigger to submit jenkins jobs upon receipt of a matching message.
- A builder that may be used to submit a message to the topic upon the completion of a job
- A post-build action that may be used to submit a message to the topic upon the completion of a job

JMS Messaging provider types supported:

- ActiveMQ
- FedMsg

Requires the Jenkins JMS Messaging Plugin.

Parameters

- **no-squash** (*bool*) – true = schedule a new job for every triggering message. (default false) Normally if a job is queued and another triggering message is received, a new job is not submitted and the job is "squashed" into the job already queued. Setting this option to 'True' forces a new job to be submitted for every triggering message that is

received.
- **override-topic** (*str*) – If you need to override the default topic. (default '')
- **selector** (*str*) – The JSON or YAML formatted text that conforms to the schema for defining the various OpenShift resources. (default '') note: topic needs to be in double quotes ex. topic = "org.fedoraproject.prod.fedimg.image.upload"
- **provider-name** (*str*) – Name of message provider setup in the global config. (default '')
- **checks** (*list*) – List of checks to monitor. (default [])
- **field** (*str*) – Check the body of messages for a field. (default '')
- **expected-value** (*str*) – Expected value for the field. regex (default '')

Full Example:

```
triggers:
  - jms-messaging:
      no-squash: True
      selector: topic = "org.fedoraproject.prod.fedimg.image.upload"
      provider-name: fedmsg
      checks:
        - field: compose
          expected-value: .+compose_id.+Fedora-Atomic.+
        - field: image_name
          expected-value: .+Fedora-Atomic.+
```

Minimal Example:

```
triggers:
  - jms-messaging:
      selector: topic = "org.fedoraproject.prod.fedimg.image.upload"
      provider-name: fedmsg
```

monitor-files()

Configure Jenkins to monitor files. Requires the Jenkins Filesystem Trigger Plugin.

Parameters

- **files** (*list*) – List of files to monitor

  File

  - **path** (*str*) – File path to monitor. You can use a pattern that specifies a set of files if you don't know the real file path. (required)
  - **strategy** (*str*) – Choose your strategy if there is more than one matching file. Can be one of Ignore file ('IGNORE') or Use the most recent ('LATEST'). (default 'LATEST')
  - **check-content** (*list*) – List of content changes of the file to monitor

    Content Nature

    * **simple** (*bool*) – Trigger on change in content of the specified file (whatever the type file). (default false)
    * **jar** (*bool*) – Trigger on change in content of the specified JAR file. (default false)

* **tar** (*bool*) – Trigger on change in content of the specified Tar file. (default false)
* **zip** (*bool*) – Trigger on change in content of the specified ZIP file. (default false)
* **source-manifest** (*list*) – Trigger on change to MANIFEST files.

  MANIFEST File

  - **keys** (*list*) – List of keys to inspect. (optional)
  - **all-keys** (*bool*) – If true, take into account all keys. (default true)

* **jar-manifest** (*list*) – Trigger on change to MANIFEST files (contained in jar files).

  MANIFEST File

  - **keys** (*list*) –

List of keys to inspect. (optional)

- **all-keys** (*bool*) – If true, take into account all keys. (default true)

* **properties** (*list*) – Monitor the contents of the properties file.

  Properties File

  - **keys** (*list*) – List of keys to inspect. (optional)
  - **all-keys** (*bool*) – If true, take into account

all keys. (default true)

* **xml** (*list str*) – Trigger on change to the listed XPath expressions.
* **text** (*list str*) – Trigger on change to the listed regular expressions.

– **ignore-modificaton-date** (*bool*) – If true, ignore the file modification date. Only valid when content changes of the file are being monitored. (default true)

- **cron** (*str*) – cron syntax of when to run (default '')

Minimal Example:

```
triggers:
  - monitor-files:
      files:
          - path: 'path1'
```

Full Example:

```
triggers:
  - monitor-files:
      cron: '* * * * *'
      files:
          - path: 'path1'
            strategy: 'IGNORE'
          - path: 'path2'
            check-content:
                - simple: true
                - jar: true
                - tar: true
                - zip: true
                - source-manifest:
                    - all-keys: false
                      keys:
                          - key1
                          - key2
                - jar-manifest:
                    - keys:
                          - key1
                          - key2
                - properties:
                    - all-keys: false
                      keys:
                          - prop1
                          - prop2
                - xml:
                    - 'xpath1'
```

(continues on next page)

(continued from previous page)

```
                - 'xpath2'
            - text:
                - 'regex1'
        ignore-modificaton-date: false
```

monitor-folders()

Configure Jenkins to monitor folders.

Requires the Jenkins Filesystem Trigger Plugin.

Parameters

- **path** (*str*) – Folder path to poll. (default '')
- **includes** (*list*) – Fileset includes setting that specifies the list of includes files. Basedir of the fileset is relative to the workspace root. If no value is set, all files are used. (default '')
- **excludes** (*str*) – The 'excludes' pattern. A file that matches this mask will not be polled even if it matches the mask specified in 'includes' section. (default '')
- **check-modification-date** (*bool*) – Check last modification date. (default true)
- **check-content** (*bool*) – Check content. (default true)
- **check-fewer** (*bool*) – Check fewer files (default true)
- **cron** (*str*) – cron syntax of when to run (default '')

Full Example:

```
triggers:
  - monitor-folders:
      path: 'pathname'
      includes:
        - 'pattern1'
        - 'pattern2'
      excludes: 'pattern1'
      check-modification-date: false
      check-content: false
      check-fewer: false
      cron: H/15 * * * *
```

Minimal Example:

```
triggers:
  - monitor-folders
```

parameterized-timer()

Trigger builds with parameters at certain times. Requires the Jenkins Parameterized Scheduler Plugin.

Parameters

cron (*str*) – cron syntax of when to run and with which parameters (required)

Example:

```
triggers:
  - parameterized-timer:
      cron: "@midnight % PARAM=value"
```

pollscm()

Poll the SCM to determine if there has been a change.

Parameter

the polling interval (cron syntax)

Deprecated since version 1.3.0.: Please use *cron*.

Parameters

- **cron** (*str*) – the polling interval (cron syntax, required)
- **ignore-post-commit-hooks** (*bool*) – Ignore changes notified by SCM post-commit hooks. The subversion-plugin supports this since version 1.44. (default false)

Example:

```
triggers:
  - pollscm:
      cron: "*/30 * * * *"
      ignore-post-commit-hooks: True
```

pollurl()

Trigger when the HTTP response from a URL changes. Requires the Jenkins URLTrigger Plugin.

Parameters

- **cron** (*str*) – cron syntax of when to run (default '')
- **polling-node** (*str*) – Restrict where the polling should run. (optional)
- **urls** (*list*) – List of URLs to monitor

  URL

  - **url** (*str*) – URL to monitor for changes (required)
  - **proxy** (*bool*) – Activate the Jenkins proxy (default false)
  - **timeout** (*int*) – Connect/read timeout in seconds (default 300)
  - **username** (*str*) – User name for basic authentication (optional)
  - **password** (*str*) – Password for basic authentication (optional)
  - **check-status** (*int*) – Check for a specific HTTP status code (optional)
  - **check-etag** (*bool*) – Check the HTTP ETag for changes (default false)
  - **check-date** (*bool*) – Check the last modification date of the URL (default false)
  - **check-content** (*list*) – List of content type changes to monitor

    Content Type

    * **simple** (*bool*) – Trigger on any change to the content of the URL (default false)
    * **json** (*list*) – Trigger on any change to the listed JSON paths
    * **text** (*list*) – Trigger on any change to the listed regular expressions
    * **xml** (*list*) – Trigger on any change to the listed XPath expressions

Example:

```
triggers:
  - pollurl:
      cron: '* * * * *'
      polling-node: 'label expression'
      urls:
        - url: 'http://example.com/url1'
          proxy: false
```

(continues on next page)

(continued from previous page)

```
          timeout: 442
          username: username
          password: sekr3t
          check-status: 202
          check-etag: false
          check-date: true
          check-content:
            - simple: true
            - json:
              - '$..author'
              - '$.store..price'
        - url: 'http://example.com/url2'
          proxy: true
          check-etag: true
          check-content:
            - simple: false
            - xml:
              - '//author'
              - '/store//price'
            - text:
              - '\d+'
```

rabbitmq()

This plugin triggers build using remote build message in RabbitMQ queue.

Requires the Jenkins RabbitMQ Build Trigger Plugin.

Parameters

- **token** (*str*) – the build token expected in the message queue (required)
- **filters** (*list*) – list of filters to apply (optional)

  Filter

  - **field** (*str*) - Some field in message (required)
  - **value** (*str*) - value of specified field (required)

Example:

```
triggers:
  - rabbitmq:
      token: 'build_trigger_token'
```

Example with filters:

```
triggers:
  - rabbitmq:
      token: 'build_trigger_token'
      filters:
        - field: 'field1'
          value: 'value1'
        - field: 'field2'
          value: 'value2'
```

reverse()

This trigger can be configured in the UI using the checkbox with the following text: 'Build after other projects are built'.

Set up a trigger so that when some other projects finish building, a new build is scheduled for this project. This

is convenient for running an extensive test after a build is complete, for example.

This configuration complements the "Build other projects" section in the "Post-build Actions" of an upstream project, but is preferable when you want to configure the downstream project.

Parameters

- **jobs** (*str*) – List of jobs to watch. Can be either a comma separated list or a list.
- **result** (*str*) – Build results to monitor for between the following options: success, unstable and failure. (default 'success').

Example:

```
triggers:
  - reverse:
        jobs: 'Fantastic-job'
        result: 'failure'
```

Example List:

```
triggers:
  - reverse:
        jobs:
            - 'a'
            - 'b'
            - 'c'
        result: 'failure'
```

script()

Triggers the job using shell or batch script.

Requires the Jenkins ScriptTrigger Plugin.

Parameters

- **label** (*str*) – Restrict where the polling should run. (default '')
- **script** (*str*) – A shell or batch script. (default '')
- **script-file-path** (*str*) – A shell or batch script path. (default '')
- **cron** (*str*) – cron syntax of when to run (default '')
- **enable-concurrent** (*bool*) – Enables triggering concurrent builds. (default false)
- **exit-code** (*int*) – If the exit code of the script execution returns this expected exit code, a build is scheduled. (default 0)

Full Example:

```
triggers:
  - script:
        script: 'exit 0'
        script-file-path: '$WORKSPACE/scripts'
        cron: 'H/15 * * * *'
        enable-concurrent: true
        label: master
        exit-code: 0
```

Minimal Example:

```
triggers:
  - script
```

stash-pull-request()

Trigger builds via Stash/Bitbucket Server Pull Requests.

Requires the Jenkins Stash Pull Request Builder Plugin.

arg str cron
cron syntax of when to run (required)
arg str stash-host
The HTTP or HTTPS URL of the Stash host (NOT ssh). e.g.: https://example.com (required)
arg str credentials-id
Jenkins credential set to use. (required)
arg str project
Abbreviated project code. e.g.: PRJ or ~user (required)
arg str repository
Stash Repository Name. e.g.: Repo (required)
arg str ci-skip-phrases
CI Skip Phrases. (default 'NO TEST')
arg str ci-build-phrases
CI Build Phrases. (default 'test this please')
arg str target-branches
Target branches to filter. (default '')
arg bool ignore-ssl
Ignore SSL certificates for Stash host. (default false)
arg bool check-destination-commit
Rebuild if destination branch changes. (default false)
arg bool check-mergable
Build only if PR is mergeable. (default false)
arg bool merge-on-success
Merge PR if build is successful. (default false)
arg bool check-not-conflicted
Build only if Stash reports no conflicts. (default false)
arg bool only-build-on-comment
Only build when asked (with test phrase). (default false)
arg bool delete-previous-build-finish-comments
Keep PR comment only for most recent Build. (default false)
arg bool cancel-outdated-jobs
Cancel outdated jobs. (default false)

Minimal Example:

```
triggers:
  - stash-pull-request:
      cron: "* * * * *"
      stash-host: "https://stash-host.com"
      credentials-id: default-stash-credentials
      project: stash-project
      repository: stash-repo
```

Full Example:

```
triggers:
  - stash-pull-request:
      cron: "H 1 2 3 4"
      stash-host: "https://stash-host.com"
      credentials-id: default-stash-credentials
      project: stash-project
      repository: stash-repo
```

(continues on next page)

(continued from previous page)

```
      ci-skip-phrases: "test skip phrase"
      ci-build-phrases: "test build phrase"
      target-branches: "master"
      ignore-ssl: true
      check-destination-commit: true
      check-mergable: true
      merge-on-success: true
      check-not-conflicted: false
      only-build-on-comment: true
      delete-previous-build-finish-comments: true
      cancel-outdated-jobs: true
```

timed()

Trigger builds at certain times.

Parameter
when to run the job (cron syntax)

Example:

```
triggers:
  - timed: "@midnight"
```

Wrappers

Wrappers can alter the way the build is run as well as the build output.

Component: wrappers

Macro
wrapper

Entry Point
jenkins_jobs.wrappers

class wrappers.**Wrappers**(*registry*)

component_list_type = 'wrappers'

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = 'wrapper'

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 80

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

android-emulator()

Automates many Android development tasks including SDK installation, build file generation, emulator creation and launch, APK (un)installation...

Requires the Jenkins Android Emulator Plugin.

Parameters

- **avd** (*str*) – Enter the name of an existing Android emulator configuration. If this is exclusive with the 'os' arg.
- **os** (*str*) – Can be an OS version, target name or SDK add-on
- **screen-density** (*str*) – Density in dots-per-inch (dpi) or as an alias, e.g. "160" or "mdpi". (default mdpi)
- **screen-resolution** (*str*) – Can be either a named resolution or explicit size, e.g. "WVGA" or "480x800". (default WVGA)
- **locale** (*str*) – Language and country pair. (default en_US)
- **target-abi** (*str*) – Name of the ABI / system image to be used. (optional)
- **sd-card** (*str*) – sd-card size e.g. "32M" or "10240K". (optional)
- **wipe** (*bool*) – if true, the emulator will have its user data reset at start-up (default false)
- **show-window** (*bool*) – if true, the Android emulator user interface will be displayed on screen during the build. (default false)
- **snapshot** (*bool*) – Start emulator from stored state (default false)
- **delete** (*bool*) – Delete Android emulator at the end of build (default false)
- **startup-delay** (*int*) – Wait this many seconds before attempting to start the emulator (default 0)
- **commandline-options** (*str*) – Will be given when starting the Android emulator executable (optional)
- **exe** (*str*) – The emulator executable. (optional)
- **hardware-properties** (*list*) – Dictionary of hardware properties. Allows you to override the default values for an AVD. (optional)

Example:

```
wrappers:
  - android-emulator:
      os: android-19
      target-abi: x86
      sd-card: 16MB
      hardware-properties:
        hw.accelerometer: 100
      wipe: true
      show-window: true
      snapshot: true
      delete: true
      startup-delay: 10
      commandline-options: "-gpu on -no-audio"
      exe: emulator-arm
```

ansicolor()

Translate ANSI color codes to HTML in the console log.

Requires the Jenkins Ansi Color Plugin.

Parameters

colormap (*str*) – Color mapping to use (default xterm)

Minimal Example:

```
wrappers:
  - ansicolor
```

Full Example:

```
wrappers:
  - ansicolor:
      colormap: "gnome-terminal"
```

artifactory-generic()

Wrapper for non-Maven projects.

Requires the Jenkins Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. https://jfrog.com/artifactory/ (default '')
- **name** (*str*) – Artifactory user with permissions use for connected to the selected Artifactory Server (default '')
- **repo-key** (*str*) – Release repository name (plugin < 2.3.0) (default '')
- **snapshot-repo-key** (*str*) – Snapshots repository name (plugin < 2.3.0) (default '')
- **key-from-select** (*str*) – Repository key to use (plugin >= 2.3.0) (default '')
- **key-from-text** (*str*) – Repository key to use that can be configured dynamically using Jenkins variables (plugin >= 2.3.0) (default '')
- **upload-spec** (*str*) – File Spec schema for uploading files is as follows (default '')
- **download-spec** (*str*) – File Spec schema for downloading files is as follows (default '')
- **upload-spec-file** (*str*) – File location for uploading Spec schema (default '')
- **download-spec-file** (*str*) – File location for downloading Spec schema (default '')
- **deploy-pattern** (*list*) – List of patterns for mappings build artifacts to published artifacts. Supports Ant-style wildcards mapping to target directories. E.g.: /.zip=>dir (default [])
- **resolve-pattern** (*list*) – List of references to other artifacts that this build should use as dependencies.
- **matrix-params** (*list*) – List of properties to attach to all deployed artifacts in addition to the default ones: build.name, build.number, and vcs.revision (default [])
- **deploy-build-info** (*bool*) – Deploy jenkins build metadata with artifacts to Artifactory (default false)
- **env-vars-include** (*bool*) – Include environment variables accessible by the build process. Jenkins-specific env variables are always included. Use the env-vars-include-patterns and env-vars-exclude-patterns to filter the environment variables published to artifactory. (default false)
- **env-vars-include-patterns** (*list*) – List of environment variable patterns for including env vars as part of the published build info. Environment variables may contain the * and the ? wildcards (default [])
- **env-vars-exclude-patterns** (*list*) – List of environment variable patterns that determine the env vars excluded from the published build info (default [])
- **discard-old-builds** (*bool*) – Remove older build info from Artifactory (default false)
- **discard-build-artifacts** (*bool*) – Remove older build artifacts from Artifactory (default false)

Example:

```
wrappers:
  - artifactory-generic:
      url: http://artifactory.example.net/artifactory
      name: 'test'
      deploy-build-info: true
      repo-key: 'release-repo'
      snapshot-repo-key: 'snapshot-repo'
      deploy-pattern:
      - '*.zip=>results'
      resolve-pattern:
      - 'libs-release-local:prod/*=>prod-jars'
      matrix-params:
      - 'custom_prop=${PROJECT_ENV_VAR}'
      env-vars-include: true
      env-vars-include-patterns:
      - 'PROJECT_*'
      - 'ORG_*'
      discard-old-builds: true
      discard-build-artifacts: true
```

artifactory-maven()

Wrapper for non-Maven projects.

Requires the Jenkins Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. https://jfrog.com/artifactory/ (default '')
- **name** (*str*) – Artifactory user with permissions use for connected to the selected Artifactory Server (default '')
- **repo-key** (*str*) – Name of the repository to search for artifact dependencies. Provide a single repo-key or provide separate release-repo-key and snapshot-repo-key.
- **release-repo-key** (*str*) – Release repository name. Value of repo-key take priority over release-repo-key if provided.
- **snapshot-repo-key** (*str*) – Snapshots repository name. Value of repo-key take priority over release-repo-key if provided.

Example:

```
wrappers:
  - artifactory-maven:
      url: http://artifactory.example.net/artifactory
      name: 'test'
      repo-key: repo
```

artifactory-maven-freestyle()

Wrapper for Free Stype projects.

Requires the Jenkins Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. https://jfrog.com/artifactory/ (default '')
- **name** (*str*) – Artifactory user with permissions use for connected to the selected Artifactory Server (default '')
- **release-repo-key** (*str*) – Release repository name (default '')
- **snapshot-repo-key** (*str*) – Snapshots repository name (default '')

- **publish-build-info** (*bool*) – Push build metadata with artifacts (default false)
- **discard-old-builds** (*bool*) – Remove older build info from Artifactory (default true)
- **discard-build-artifacts** (*bool*) – Remove older build artifacts from Artifactory (default false)
- **include-env-vars** (*bool*) – Include all environment variables accessible by the build process. Jenkins-specific env variables are always included (default false)
- **run-checks** (*bool*) – Run automatic license scanning check after the build is complete (default false)
- **include-publish-artifacts** (*bool*) – Include the build's published module artifacts in the license violation checks if they are also used as dependencies for other modules in this build (default false)
- **license-auto-discovery** (*bool*) – Tells Artifactory not to try and automatically analyze and tag the build's dependencies with license information upon deployment (default true)
- **enable-issue-tracker-integration** (*bool*) – When the Jenkins JIRA plugin is enabled, synchronize information about JIRA issues to Artifactory and attach issue information to build artifacts (default false)
- **aggregate-build-issues** (*bool*) – When the Jenkins JIRA plugin is enabled, include all issues from previous builds up to the latest build status defined in "Aggregation Build Status" (default false)
- **filter-excluded-artifacts-from-build** (*bool*) – Add the excluded files to the excludedArtifacts list and remove them from the artifacts list in the build info (default false)
- **scopes** (*str*) – A list of dependency scopes/configurations to run license violation checks on. If left empty all dependencies from all scopes will be checked (default '')
- **violation-recipients** (*str*) – Recipients that need to be notified of license violations in the build info (default '')
- **matrix-params** (*list*) – List of properties to attach to all deployed artifacts in addition to the default ones: build.name, build.number, and vcs.revision (default '')
- **black-duck-app-name** (*str*) – The existing Black Duck Code Center application name (default '')
- **black-duck-app-version** (*str*) – The existing Black Duck Code Center application version (default '')
- **black-duck-report-recipients** (*str*) – Recipients that will be emailed a report after the automatic Black Duck Code Center compliance checks finished (default '')
- **black-duck-scopes** (*str*) – A list of dependency scopes/configurations to run Black Duck Code Center compliance checks on. If left empty all dependencies from all scopes will be checked (default '')
- **black-duck-run-checks** (*bool*) – Automatic Black Duck Code Center compliance checks will occur after the build completes (default false)
- **black-duck-include-published-artifacts** (*bool*) – Include the build's published module artifacts in the license violation checks if they are also used as dependencies for other modules in this build (default false)
- **auto-create-missing-component-requests** (*bool*) – Auto create missing components in Black Duck Code Center application after the build is completed and deployed in Artifactory (default true)
- **auto-discard-stale-component-requests** (*bool*) – Auto discard stale components in Black Duck Code Center application after the build is completed and deployed in Artifactory (default true)
- **deploy-artifacts** (*bool*) – Push artifacts to the Artifactory Server. The specific artifacts to push are controlled using the deployment-include-patterns and deployment-exclude-patterns. (default true)
- **deployment-include-patterns** (*list*) – List of patterns for including build arti-

facts to publish to artifactory. (default[]')
- **deployment-exclude-patterns** (*list*) – List of patterns for excluding artifacts from deployment to Artifactory (default [])
- **env-vars-include** (*bool*) – Include environment variables accessible by the build process. Jenkins-specific env variables are always included. Environment variables can be filtered using the env-vars-include-patterns nad env-vars-exclude-patterns. (default false)
- **env-vars-include-patterns** (*list*) – List of environment variable patterns that will be included as part of the published build info. Environment variables may contain the * and the ? wildcards (default [])
- **env-vars-exclude-patterns** (*list*) – List of environment variable patterns that will be excluded from the published build info (default [])

Example:

```
wrappers:
  - artifactory-maven-freestyle:
      url: http://artifactory.example.net/artifactory
      name: 'test'
      repo-key: repo
      matrix-params:
      - 'custom_prop=${PROJECT_ENV_VAR}'
      deployment-include-patterns:
      - '*.zip=>results'
      env-vars-include: true
      env-vars-include-patterns:
      - 'PROJECT_*'
      - 'ORG_*'
```

build-keeper()

Keep builds based on specific policy.

Requires the Jenkins Build Keeper Plugin.

Parameters

- **policy** (*str*) – Policy to keep builds.
  - **policy values**
    - **by-day**
    - **keep-since**
    - **build-number**
    - **keep-first-failed**
    - **run-condition**
- **build-period** (*int*) – Number argument to calculate build to keep, depends on the policy. (default 0)
- **dont-keep-failed** (*bool*) – Flag to indicate if to keep failed builds. (default false)
- **number-of-fails** (*int*) – number of consecutive failed builds in order to mark first as keep forever, only applies to keep-first-failed policy (default 0)
- **keep-build** (*bool*) – Build will be kept if there is a problem evaluating the RunCondition (default false)
- **token** (*str*) – Token value for the boolean condition (default '')
- **build-cause** (*list*) – The cause why the build was triggered (default USER_CAUSE)
- **exclusive-cause** (*bool*) – Cause must be the only one causing this build to be triggered (default False)
- **command** (*str*) – Contents of your shell script (default '')
- **allowed-nodes** (*str*) – Node to be executed on (default '')
- **expression** (*str*) – The regular expression used to match the label (default '')

- **label** (*str*) – The label that will be tested by the regular expression (default '')
- **arg1** (*str*) – First string argument for strings-match condition (default '')
- **arg2** (*str*) – Second string argument for strings-match condition (default '')
- **ignore-case** (*bool*) – Ignore the case of the strings when matching the two string arguments (default False)

Example:

```
wrappers:
  - build-keeper:
      policy: 'by-day'
      build-period: 10
      dont-keep-failed: true
```

```
wrappers:
  - build-keeper:
      policy: 'keep-first-failed'
      number-of-fails: 1
```

build-name()

Set the name of the build.

Requires the Jenkins Build Name Setter Plugin.

Parameters

- **name** (*str*) – Name for the build. Typically you would use a variable from Jenkins in the name. The syntax would be ${FOO} for the FOO variable.
- **description** (*str*) – Build description for the build (Optional).
- **run-at-start** (*str*) – Set build name before build starts (Optional, default: True).
- **run-at-end** (*str*) – Set build name after build ends (Optional, default: True).

Example:

```
wrappers:
  - build-name:
      name: Build-${FOO}
      description: lorem ipsum dolor
      run-at-start: true
      run-at-end: false
```

build-user-vars()

Set environment variables to the value of the user that started the build.

Requires the Jenkins Build User Vars Plugin.

Example:

```
wrappers:
  - build-user-vars
```

ci-skip()

Skip making a build for certain push. Just add [ci skip] into your commit's message to let Jenkins know, that you do not want to perform build for the next push.

Requires the Jenkins Ci Skip Plugin.

Example:

```
wrappers:
  - ci-skip
```

config-file-provider()

Provide configuration files (i.e., settings.xml for maven etc.) which will be copied to the job's workspace.

Requires the Jenkins Config File Provider Plugin.

Parameters

files (*list*) – List of managed config files made up of three parameters

files

- **file-id** (*str*) – The identifier for the managed config file
- **target** (*str*) – Define where the file should be created (default '')
- **variable** (*str*) – Define an environment variable to be used (default '')
- **replace-tokens** (*bool*) – Replace tokens in config file. For example "password: ${PYPI_JENKINS_PASS}" will be replaced with the global variable configured in Jenkins.

Full Example:

```
wrappers:
  - config-file-provider:
      files:
        - file-id: org.jenkinsci.plugins.configfiles.custom.CustomConfig1234
        - file-id: org.jenkinsci.plugins.configfiles.custom.CustomConfig5678
          target: /foo.txt
          variable: varName
          replace-tokens: true
```

Minimal Example:

```
wrappers:
  - config-file-provider:
      files:
        - file-id: org.jenkinsci.plugins.configfiles.custom.CustomConfig1234
```

copy-to-slave()

Copy files to slave before build.

Requires the Jenkins Copy To Slave Plugin.

Parameters

- **includes** (*list*) – list of file patterns to copy (optional)
- **excludes** (*list*) – list of file patterns to exclude (optional)
- **flatten** (*bool*) – flatten directory structure (default false)
- **relative-to** (*str*) – base location of includes/excludes, must be home ($JENKINS_HOME), somewhereElse ($JENKINS_HOME/copyToSlave), userContent ($JENKINS_HOME/userContent) or workspace (default userContent)
- **include-ant-excludes** (*bool*) – exclude ant's default excludes (default false)

Minimal Example:

```
wrappers:
    - copy-to-slave
```

Full Example:

```
wrappers:
    - copy-to-slave:
        includes:
            - 'file1'
            - 'file2*.txt'
        excludes:
            - 'file2bad.txt'
        flatten: True
        relative-to: 'somewhereElse'
        include-ant-excludes: True
```

credentials-binding()

Binds credentials to environment variables using the credentials binding plugin for jenkins.

Requires the Jenkins Credentials Binding Plugin version 1.1 or greater.

Parameters

binding-type (*list*) – List of each bindings to create. Bindings may be of type *zip-file*, *file*, *username-password*, *text*, *username-password-separated* or *amazon-web-services*. username-password sets a variable to the username and password given in the credentials, separated by a colon. username-password-separated sets one variable to the username and one variable to the password given in the credentials. amazon-web-services sets one variable to the access key and one variable to the secret access key. Requires the AWS Credentials Plugin .

Parameters

- **credential-id** (*str*) UUID of the credential being referenced
- **variable** (*str*) Environment variable where the credential will be stored
- **username** (*str*) Environment variable for the username (Required for binding-type username-password-separated)
- **password** (*str*) Environment variable for the password (Required for binding-type username-password-separated)
- **access-key** (*str*) Environment variable for the access key (Required for binding-type amazon-web-services)
- **secret-key** (*str*) Environment variable for the access secret key (Required for binding-type amazon-web-services)
- **key-file-variable** (*str*) Environment variable to be set to the temporary path of the SSH key file during the build.
- **username-variable** (*str*) Environment variable to be set to the username during the build. (optional)
- **passphrase-variable** (*str*) Environment variable to be set to the password during the build. (optional)
- **keystore-variable** (*str*) Environment variable to be set to the temporary keystore location during the build.
- **password-variable** (*str*) Environment variable to be set to the password during the build.
- **alias-variable** (*str*) Environment variable to be set to the keystore alias name of the certificate during the build.

Example:

```
wrappers:
 - credentials-binding:
     - zip-file:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caa6
        variable: CONFIG_ZIP
     - file:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caab
        variable: config_file
     - username-password:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caac
        variable: config_username_password
     - text:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caad
        variable: config_text
     - docker-server-creds-binding:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caae
        variable: config_docker_server
 - credentials-binding:
     - username-password-separated:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caag
        username: myUsername
        password: myPassword
 - credentials-binding:
     - amazon-web-services:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caaf
        access-key: AWS_ACCESS_KEY_ID
        secret-key: AWS_SECRET_ACCESS_KEY
 - credentials-binding:
     - ssh-user-private-key:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caah
        key-file-variable: KEY_FILE_VARIABLE
        username-variable: USER_NAME_VARIABLE
        passphrase-variable: PASSPHRASE_VARIABLE
 - credentials-binding:
     - cert-multi-binding:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caaj
        keystore-variable: KEYSTORE_VARIABLE
        password-variable: PASSWORD_VARIABLE
        alias-variable: ALIAS_VARIABLE
```

custom-tools()

Requires the Jenkins Custom Tools Plugin.

Parameters

- **tools** (*list*) – List of custom tools to add (optional)
- **skip-master-install** (*bool*) – skips the install in top level matrix job (default 'false')
- **convert-homes-to-upper** (*bool*) – Converts the home env vars to uppercase (default 'false')

Example:

```
wrappers:
  - custom-tools:
      tools:
```

(continues on next page)

(continued from previous page)

```
    - my_custom_tool
    skip-master-install: true
    convert-homes-to-upper: true
```

delivery-pipeline()

If enabled the job will create a version based on the template. The version will be set to the environment variable PIPELINE_VERSION and will also be set in the downstream jobs.

Requires the Jenkins Delivery Pipeline Plugin.

Parameters

- **version-template** (*str*) – Template for generated version e.g 1.0.${BUILD_NUMBER} (default '')
- **set-display-name** (*bool*) – Set the generated version as the display name for the build (default false)

Minimal Example:

```
wrappers:
    - delivery-pipeline
```

Full Example:

```
wrappers:
    - delivery-pipeline:
        version-template: 1.0.0-${BUILD_NUMBER}
        set-display-name: true
```

docker-custom-build-env()

Allows the definition of a build environment for a job using a Docker container.

Requires the Jenkins CloudBees Docker Custom Build Environment Plugin.

Parameters

- **image-type** (*str*) – Docker image type. Valid values and their additional attributes described in the *image_types* table
- **docker-tool** (*str*) – The name of the docker installation to use (default 'Default')
- **host** (*str*) – URI to the docker host you are using
- **credentials-id** (*str*) – Argument to specify the ID of credentials to use for docker host (optional)
- **registry-credentials-id** (*str*) – Argument to specify the ID of credentials to use for docker registry (optional)
- **volumes** (*list*) – Volumes to bind mound from slave host into container

  volume

  - **host-path** (*str*) Path on host
  - **path** (*str*) Path inside container
- **verbose** (*bool*) – Log docker commands executed by plugin on build log (default false)
- **privileged** (*bool*) – Run in privileged mode (default false)
- **force-pull** (*bool*) – Force pull (default false)
- **group** (*str*) – The user to run build has to be the same as the Jenkins slave user so files created in workspace have adequate owner and permission set
- **command** (*str*) – Container start command (default '/bin/cat')
- **net** (*str*) – Network bridge (default 'bridge')
- **memory-limit** (*str*) – Configure the limit memory constraint (default '')
- **cpu-shares** (*str*) – Configure the CPU shares constraint (default '')

Image Type	Description
dockerfile	Build docker image from a Dockerfile in project workspace. With this option, project can define the build environment as a Dockerfile stored in SCM with project source code **context-path** (str) Path to docker context (default '.') **dockerfile** (str) Use an alternate Dockerfile to build the container hosting this build (default 'Dockerfile')
pull	Pull specified docker image from Docker repository **image** (str) Image id/tag

Example:

```
wrappers:
  - docker-custom-build-env:
      image-type: 'pull'
      image: 'centos:7'
      force-pull: true
      privileged: true
      verbose: true
      group: jenkins
      command: /bin/cat
      net: bridge
      memory-limit: memory=L<inf, memory-swap=inf
      cpu-shares: 2
```

env-file()

Add or override environment variables to the whole build process.

Requires the Jenkins Environment File Plugin.

Parameters

properties-file (*str*) – path to the properties file (optional)

Example:

```
wrappers:
  - env-file:
      properties-file: ${WORKSPACE}/foo
```

env-script()

Add or override environment variables to the whole build process.

Requires the Jenkins Environment Script Plugin.

Parameters

- **script-content** – The script to run (default '')
- **script-type** (*str*) – The script type.

script-types supported
- **unix-script** (default)
- **power-shell**
- **batch-script**

- **only-run-on-parent** – Only applicable for Matrix Jobs. If true, run only on the matrix parent job (default false)

Example:

```
wrappers:
  - env-script:
      script-content: 'echo foo=bar'
      only-run-on-parent: true
```

exclusion()

Add a resource to use for critical sections to establish a mutex on. If another job specifies the same resource, the second job will wait for the blocked resource to become available.

Requires the Jenkins Exclusion Plugin.

Parameters

resources (*list*) – List of resources to add for exclusion

Example:

```
wrappers:
  - exclusion:
      resources:
          - myresource1
          - myresource2
```

github-pull-request()

Set GitHub commit status with custom context and message.

Requires the Jenkins GitHub Pull Request Builder Plugin.

Parameters

- **show-matrix-status** (*bool*) – Only post commit status of parent matrix job (default false)
- **status-context** (*str*) – The context to include on PR status comments (default '')
- **triggered-status** (*str*) – The status message to set when the build has been triggered (default '')
- **started-status** (*str*) – The status message to set when the build has been started (default '')
- **status-url** (*str*) – The status URL to set (default '')
- **status-add-test-results** (*bool*) – Add test result one-liner to status message (default false)
- **statuses** (*list*) – List of custom statuses on the commit for when a build is completed

  Status

  - **message** (*str*) – The message that is appended to a comment when a build finishes with the desired build status. If no status updates should be made when a build finishes with the indicated build status, use "–none–" to alert the trigger. (required)
  - **result** (*str*) – Build result. Can be one of 'SUCCESS', 'ERROR' or 'FAILURE'. (required)

Minimal Example:

```
wrappers:
  - github-pull-request
```

Full Example:

```
wrappers:
  - github-pull-request:
      show-matrix-status: true
      status-context: "my-build"
      triggered-status: "build was triggered"
      started-status: "build was started"
      status-url: "http://1.2.3.4"
      status-add-test-results: true
      statuses:
        - message: "build has succeeded"
          result: SUCCESS
        - message: "build has failed"
          result: ERROR
```

inject()

Add or override environment variables to the whole build process.

Requires the Jenkins EnvInject Plugin.

Parameters

- **properties-file** (*str*) – path to the properties file (optional)
- **properties-content** (*str*) – key value pair of properties (optional)
- **script-file** (*str*) – path to the script file (optional)
- **script-content** (*str*) – contents of a script (optional)
- **load-from-master** (*bool*) – load files from master (default false)
- **groovy-script** (*str*) – contents of the groovy script (optional)
- **groovy-sandbox** (*bool*) – use groovy sandbox (default false)

Minimal Example:

```
wrappers:
    - inject
```

Full Example:

```
wrappers:
    - inject:
        properties-file: example.prop full
        properties-content: EXAMPLE=foo-bar full
        script-file: scriptfull.sh
        script-content: test script content full
        groovy-script: test groovy-script location full
        groovy-sandbox: true
```

inject-ownership-variables()

Inject ownership variables to the build as environment variables.

Requires the Jenkins EnvInject Plugin and Jenkins Ownership plugin.

Parameters

- **job-variables** (*bool*) – inject job ownership variables to the job (default false)
- **node-variables** (*bool*) – inject node ownership variables to the job (default false)

Example:

```
wrappers:
  - inject-ownership-variables:
```

(continues on next page)

(continued from previous page)

```
    job-variables: true
    node-variables: true
```

inject-passwords()

Inject passwords to the build as environment variables.

Requires the Jenkins EnvInject Plugin.

Parameters

- **global** (*bool*) – inject global passwords to the job
- **mask-password-params** (*bool*) – mask password parameters
- **job-passwords** (*list*) – key value pair of job passwords

  Parameter

  - **name** (*str*) Name of password
  - **password** (*str*) Encrypted password

Example:

```
wrappers:
    - inject-passwords:
        global: true
        mask-password-params: true
        job-passwords:
            - name: ADMIN
              password: 0v8ZCNaHwq1hcx+sHwRLdg9424uBh4Pin0zO4sBIb+U=
```

jclouds()

Uses JClouds to provide slave launching on most of the currently usable Cloud infrastructures.

Requires the Jenkins JClouds Plugin.

Parameters

- **single-use** (*bool*) – Whether or not to terminate the slave after use (default false).
- **instances** (*list*) – The name of the jclouds template to create an instance from, and its parameters.
- **cloud-name** (*str*) – The name of the jclouds profile containing the specified template.
- **count** (*int*) – How many instances to create (default 1).
- **stop-on-terminate** (*bool*) – Whether or not to suspend instead of terminate the instance (default false).

Example:

```
wrappers:
   - jclouds:
       single-use: True
       instances:
         - jenkins-dev-slave:
             cloud-name: mycloud1
             count: 1
             stop-on-terminate: True
         - jenkins-test-slave:
             cloud-name: mycloud2
             count: 2
             stop-on-terminate: False
```

job-log-logger()

Enable writing the job log to the underlying logging system.

Requires the Jenkins Job Log Logger plugin.

Parameters

suppress-empty (*bool*) – Suppress empty log messages (default true)

Example:

```
wrappers:
  - job-log-logger:
      suppress-empty: false
```

live-screenshot()

Show live screenshots of running jobs in the job list.

Requires the Jenkins Live-Screenshot Plugin.

Parameters

- **full-size** (*str*) – name of screenshot file (default 'screenshot.png')
- **thumbnail** (*str*) – name of thumbnail file (default 'screenshot-thumb.png')

File type must be .png and they must be located inside the $WORKDIR.

Full Example:

```
wrappers:
    - live-screenshot:
        full-size: my_screenshot.png
        thumbnail: my_screenshot-thumb.png
```

Minimal Example:

```
wrappers:
    - live-screenshot
```

locks()

Control parallel execution of jobs.

Requires the Jenkins Locks and Latches Plugin.

Arg

list of locks to use

Example:

```
wrappers:
    - locks:
        - FOO
        - FOO2
```

logfilesize()

Abort the build if its logfile becomes too big.

Requires the Jenkins Logfilesizechecker Plugin.

Parameters

- **set-own** (*bool*) – Use job specific maximum log size instead of global config value (default false).
- **fail** (*bool*) – Make builds aborted by this wrapper be marked as "failed" (default false).
- **size** (*int*) – Abort the build if logfile size is bigger than this value (in MiB, default 128). Only applies if set-own is true.

Full Example:

```
wrappers:
  - logfilesize:
      set-own: true
      size: 1024
      fail: true
```

Minimal Example:

```
wrappers:
  - logfilesize
```

logstash build wrapper()

Dump the Jenkins console output to Logstash.

Requires the Jenkins logstash plugin.

Parameters

- **use-redis** – Boolean to use Redis. (default true)
- **redis** – Redis config params

  Parameter

  – **host** (*str*) Redis hostname (default 'localhost')

  Parameter

  – **port** (*int*) Redis port number (default 6397)

  Parameter

  – **database-number** (*int*) Redis database number (default 0)

  Parameter

  – **database-password** (*str*) Redis database password (default '')

  Parameter

  – **data-type** (*str*) Redis database type (default 'list')

  Parameter

  – **key** (*str*) Redis key (default 'logstash')

Example:

```
wrappers:
  - logstash:
      use-redis: True
      redis:
        host: 'localhost'
        port: 6379
        database-number: 0
        database-password: 'password'
        data-type: 'list'
        key: 'logstash'
```

m2-repository-cleanup()

Configure M2 Repository Cleanup.

Requires the Jenkins M2 Repository Cleanup.

Parameters

patterns (*list*) – List of patterns for artifacts to cleanup before building. (optional)

This plugin allows you to configure a maven2 job to clean some or all of the artifacts from the repository before it runs.

Example:

```
wrappers:
    - m2-repository-cleanup:
        patterns:
            - com/ibm/**
            - com/microsoft/**
```

mask-passwords()

Hide passwords in the console log.

Requires the Jenkins Mask Passwords Plugin.

Example:

```
wrappers:
  - mask-passwords
```

matrix-tie-parent()

Tie parent to a node.

Requires the Jenkins Matrix Tie Parent Plugin.

Note that from Jenkins version 1.532 this plugin's functionality is available under the "advanced" option of the matrix project configuration. You can use the top level **node** parameter to control where the parent job is tied in Jenkins 1.532 and higher.

Parameters

node (*str*) – Name of the node (required)

Example:

```
project-type: matrix
wrappers:
  - matrix-tie-parent:
      node: Unix
```

maven-release()

Wrapper for Maven projects

Requires the Jenkins M2 Release Plugin

Parameters

- **release-goals** (*str*) – Release goals and options (default '')
- **dry-run-goals** (*str*) – DryRun goals and options (default '')
- **num-successful-builds** (*int*) – Number of successful release builds to keep (default 1)
- **select-custom-scm-comment-prefix** (*bool*) – Preselect 'Specify custom SCM comment prefix' (default false)
- **select-append-jenkins-username** (*bool*) – Preselect 'Append Jenkins Username' (default false)
- **select-scm-credentials** (*bool*) – Preselect 'Specify SCM login/password' (default false)
- **release-env-var** (*str*) – Release environment variable (default '')
- **scm-user-env-var** (*str*) – SCM username environment variable (default '')
- **scm-password-env-var** (*str*) – SCM password environment variable (default '')

Example:

```
wrappers:
  - maven-release:
```

(continues on next page)

(continued from previous page)

```
    release-goals: -Dresume=false release:prepare release:perform
    dry-run-goals: -Dresume=false -DdryRun=true release:prepare
    num-successful-builds: 1
    select-custom-scm-comment-prefix: false
    select-append-jenkins-username: false
    select-scm-credentials: false
    release-env-var: IS_M2RELEASEBUILD
    scm-user-env-var: SCM_USER
```

mongo-db build wrapper()

Initializes a MongoDB database while running the build.

Requires the Jenkins MongoDB plugin.

Parameters

- **name** (*str*) – The name of the MongoDB install to use (required)
- **data-directory** (*str*) – Data directory for the server (default '')
- **port** (*int*) – Port for the server (default '')
- **startup-params** (*str*) – Startup parameters for the server (default '')
- **start-timeout** (*int*) – How long to wait for the server to start in milliseconds. 0 means no timeout. (default 0)

Full Example:

```
wrappers:
  - mongo-db:
      name: 2.4.6
      data-directory: /var/tmp/mongo
      port: 5555
      startup-params: "--bind_ip 127.0.0.1"
      start-timeout: 5000
```

Minimal Example:

```
wrappers:
  - mongo-db:
      name: 2.4.6
```

nodejs-installator()

Provides Jenkins integration for NodeJS & npm packages.

Requires the Jenkins NodeJS Plugin.

Parameters

name (*str*) – nodejs installation name (required)

Example:

```
wrappers:
  - nodejs-installator:
      name: "latest node"
```

openstack()

Provision slaves from OpenStack on demand.

Requires the Jenkins Openstack Cloud Plugin.

Parameters

- **instances** (*list*) – List of instances to be launched at the beginning of the build.

instances

- **cloud-name** (*str*) – The name of the cloud profile which contains the specified cloud instance template (required).
- **template-name** (*str*) – The name of the cloud instance template to create an instance from(required).
- **manual-template** (*bool*) – If True, instance template name will be put in 'Specify Template Name as String' option. Not specifying or specifying False, instance template name will be put in 'Select Template from List' option. To use parameter replacement, set this to True. (default false)
- **count** (*int*) – How many instances to create (default 1).

- **single-use** (*bool*) – Whether or not to terminate the slave after use (default false).

Example:

```
wrappers:
  - openstack:
      instances:
        - cloud-name: mycloud1
          template-name: jenkins-dev-slave
          count: 1
        - cloud-name: mycloud2
          template-name: jenkins-test-slave
          manual-template: True
          count: 2
      single-use: True
```

pathignore()

This plugin allows SCM-triggered jobs to ignore build requests if only certain paths have changed.

Requires the Jenkins Pathignore Plugin.

Parameters

ignored (*str*) – A set of patterns to define ignored changes

Example:

```
wrappers:
  - pathignore:
      ignored: "docs, tests"
```

port-allocator()

Assign unique TCP port numbers.

Requires the Jenkins Port Allocator Plugin.

Parameters

- **name** (*str*) – Deprecated, use names instead
- **names** (*list*) – Variable list of names of the port or list of specific port numbers

Example:

```
wrappers:
    - port-allocator:
        names:
            - SERVER_PORT
            - SERVER_PORT2
```

pre-scm-buildstep()

Execute a Build Step before running the SCM.

Requires the Jenkins Pre SCM BuildStep.

Parameters

- **failOnError** (*str*) – Specifies if the job should fail on error (plugin >= 0.3) (default false).
- **buildsteps** (*list*) – List of build steps to execute

  Buildstep

  Any acceptable builder, as seen in the example

Example:

```
wrappers:
  - pre-scm-buildstep:
      failOnError: true
      buildsteps:
        - shell: |
            #!/bin/bash
            echo "Doing something cool"
        - shell: |
            #!/bin/zsh
            echo "Doing something cool with zsh"
        - ant: "target1 target2"
          ant-name: "Standard Ant"
        - inject:
            properties-file: example.prop
            properties-content: EXAMPLE=foo-bar
```

rbenv()

Set the rbenv implementation.

Requires the Jenkins rbenv plugin.

All parameters are optional.

Parameters

- **ruby-version** (*str*) – Version of Ruby to use (default 1.9.3-p484)
- **ignore-local-version** (*bool*) – If true, ignore local Ruby version (defined in the ".ruby-version" file in workspace) even if it has been defined (default false)
- **preinstall-gem-list** (*str*) – List of gems to install (default 'bundler,rake')
- **rbenv-root** (*str*) – RBENV_ROOT (default $HOME/.rbenv)
- **rbenv-repo** (*str*) – Which repo to clone rbenv from (default https://github.com/rbenv/rbenv)
- **rbenv-branch** (*str*) – Which branch to clone rbenv from (default master)
- **ruby-build-repo** (*str*) – Which repo to clone ruby-build from (default https://github.com/rbenv/ruby-build)
- **ruby-build-branch** (*str*) – Which branch to clone ruby-build from (default master)

Example:

```
wrappers:
  - rbenv:
      ruby-version: 2.0.0-p353
      ignore-local-version: false
      preinstall-gem-list: "bundler,rake"
      rbenv-root: "$HOME/.rbenv"
      rbenv-repo: "https://github.com/sstephenson/rbenv.git"
      rbenv-branch: "master"
      ruby-build-repo: "https://github.com/sstephenson/ruby-build.git"
      ruby-build-branch: "master"
```

release()

Add release build configuration.

Requires the Jenkins Release Plugin.

Parameters

- **keep-forever** (*bool*) – Keep build forever (default true)
- **override-build-parameters** (*bool*) – Enable build-parameter override (default false)
- **version-template** (*str*) – Release version template (default '')
- **parameters** (*list*) – Release parameters (see the *Parameters* module)
- **pre-build** (*list*) – Pre-build steps (see the *Builders* module)
- **post-build** (*list*) – Post-build steps (see *Builders*)
- **post-success** (*list*) – Post successful-build steps (see *Builders*)
- **post-failed** (*list*) – Post failed-build steps (see *Builders*)

Example:

```
wrappers:
  - release:
      keep-forever: false
      parameters:
          - string:
              name: RELEASE_BRANCH
              default: ''
              description: Git branch to release from.
          - bool:
              name: FOO
              default: false
              description: "A parameter named FOO, defaults to 'false'."
      post-success:
          - shell: |
              #!/bin/bash
              copy_build_artefacts.sh
```

rvm-env()

Set the RVM implementation.

Requires the Jenkins Rvm Plugin.

Parameters

implementation (*str*) – Type of implementation. Syntax is RUBY[@GEMSET], such as '1.9.3' or 'jruby@foo'.

Example:

```
wrappers:
  - rvm-env:
      implementation: 1.9.3
```

sauce-ondemand()

Allows you to integrate Sauce OnDemand with Jenkins. You can automate the setup and tear down of Sauce Connect and integrate the Sauce OnDemand results videos per test.

Requires the Jenkins Sauce OnDemand Plugin.

Parameters

- **enable-sauce-connect** (*bool*) – launches a SSH tunnel from their cloud to your private network (default false)

- **sauce-host** (*str*) – The name of the selenium host to be used. For tests run using Sauce Connect, this should be localhost. ondemand.saucelabs.com can also be used to connect directly to Sauce OnDemand, The value of the host will be stored in the SAUCE_ONDEMAND_HOST environment variable. (default '')
- **sauce-port** (*str*) – The name of the Selenium Port to be used. For tests run using Sauce Connect, this should be 4445. If using ondemand.saucelabs.com for the Selenium Host, then use 4444. The value of the port will be stored in the SAUCE_ONDEMAND_PORT environment variable. (default '')
- **override-username** (*str*) – If set then api-access-key must be set. Overrides the username from the global config. (default '')
- **override-api-access-key** (*str*) – If set then username must be set. Overrides the api-access-key set in the global config. (default '')
- **starting-url** (*str*) – The value set here will be stored in the SELENIUM_STARTING_ULR environment variable. Only used when type is selenium. (default '')
- **type** (*str*) – Type of test to run (default selenium)

  type values
  - **selenium**
  - **webdriver**
- **platforms** (*list*) – The platforms to run the tests on. Platforms supported are dynamically retrieved from sauce labs. The format of the values has only the first letter capitalized, no spaces, underscore between os and version, underscore in internet_explorer, everything else is run together. If there are not multiple version of the browser then just the first version number is used. Examples: Mac_10.8iphone5.1 or Windows_2003firefox10 or Windows_2012internet_explorer10 (default '')
- **launch-sauce-connect-on-slave** (*bool*) – Whether to launch sauce connect on the slave. (default false)
- **https-protocol** (*str*) – The https protocol to use (default '')
- **sauce-connect-options** (*str*) – Options to pass to sauce connect (default '')

Example:

```
wrappers:
  - sauce-ondemand:
      enable-sauce-connect: true
      sauce-host: foo
      sauce-port: 8080
      override-username: foo
      override-api-access-key: 123lkj123kh123l;k12323
      type: webdriver
      platforms:
        - Linuxandroid4
        - Linuxfirefox10
        - Linuxfirefox11
      launch-sauce-connect-on-slave: true
```

sonar()

Wrapper for SonarQube Plugin.

Requires SonarQube plugin

Parameters

install-name (*str*) – Release goals and options (default '')

Minimal Example:

```
wrappers:
```

(continues on next page)

(continued from previous page)

```
  - sonar
```

Full Example:

```
wrappers:
  - sonar:
      install-name: test-sonar-installation
```

ssh-agent-credentials()

Sets up the user for the ssh agent plugin for jenkins.

Requires the Jenkins SSH-Agent Plugin.

Parameters

- **users** (*list*) – A list of Jenkins users credential IDs (required)
- **user** (*str*) – The user id of the jenkins user credentials (deprecated)
- **ignore-missing-credentials** (*bool*) – Specifies the option to ignore missing credentials (default false)

Example:

```
wrappers:
  - ssh-agent-credentials:
      users:
          - '44747833-247a-407a-a98f-a5a2d785111c'
          - 'f1c0f777-7ac6-43fd-b5c7-68b420aa1392'
          - 'dd647a01-be21-402b-bfc5-a4e89be7d0c4'
```

if both users and user parameters specified, users will be
preferred, **user** will be ignored.

Example:

```
wrappers:
    - ssh-agent-credentials:
        user: '49d20745-9889-4c02-b286-fc6fb89c36bd'
        users:
            - '44747833-247a-407a-a98f-a5a2d785111c'
            - 'dd647a01-be21-402b-bfc5-a4e89be7d0c4'
```

Example:

```
wrappers:
  - ssh-agent-credentials:
      users:
          - '49d20745-9889-4c02-b286-fc6fb89c36bd'
```

equals to:

```
wrappers:
  - ssh-agent-credentials:
      user: '49d20745-9889-4c02-b286-fc6fb89c36bd'
```

timeout()

Abort the build if it runs too long.

Requires the Jenkins Build Timeout Plugin.

Parameters

- **fail** (*bool*) – Mark the build as failed (default false)
- **abort** (*bool*) – Mark the build as aborted (default false)
- **abort-and-restart** (*bool*) – Mark the build as aborted, then restart. Count of restarts can be set via *max-restarts* (default false) (Version >= 1.17).
- **write-description** (*bool*) – Write a message in the description (default false)
- **max-restarts** (*int*) – Count of maximum restarts. 0 means without a limit (default 0) (Version >= 1.17).
- **timeout** (*int*) – Abort the build after this number of minutes (default 3)
- **timeout-var** (*str*) – Export an environment variable to reference the timeout value (optional)
- **type** (*str*) – Timeout type to use (default absolute)
- **elastic-percentage** (*int*) – Percentage of the three most recent builds where to declare a timeout, only applies to **elastic** type. (default 0)
- **elastic-number-builds** (*int*) – Number of builds to consider computing average duration, only applies to **elastic** type. (default 3)
- **elastic-default-timeout** (*int*) – Timeout to use if there were no previous builds, only applies to **elastic** type. (default 3)
- **deadline-time** (*str*) – Build terminate automatically at next deadline time (HH:MM:SS), only applies to **deadline** type. (default 0:00:00)
- **deadline-tolerance** (*int*) – Period in minutes after deadline when a job should be immediately aborted, only applies to **deadline** type. (default 1)

Example (Version < 1.14):

```
wrappers:
  - timeout:
      timeout: 90
      timeout-var: 'BUILD_TIMEOUT'
      fail: true
      type: absolute
```

```
wrappers:
  - timeout:
      fail: false
      type: likely-stuck
```

```
 wrappers:
    - timeout:
        timeout-var: 'BUILD_TIMEOUT'
        fail: true
        elastic-percentage: 150
        elastic-default-timeout: 90
        type: elastic
```

Example (Version >= 1.14):

```
wrappers:
  - timeout:
      timeout: 90
      timeout-var: 'BUILD_TIMEOUT'
      fail: true
      type: absolute
```

```
wrappers:
  - timeout:
      timeout: 5
      timeout-var: 'BUILD_TIMEOUT'
      type: no-activity
      abort: true
      write-description: "Blah Blah Blah"
```

```
wrappers:
  - timeout:
      timeout: 90
      timeout-var: 'BUILD_TIMEOUT'
      abort: true
      type: likely-stuck
```

```
wrappers:
  - timeout:
      elastic-percentage: 150
      elastic-default-timeout: 3
      elastic-number-builds: 14
      timeout-var: 'BUILD_TIMEOUT'
      abort: true
      type: elastic
```

```
 wrappers:
    - timeout:
        deadline-time: '0:00:00'
        deadline-tolerance: 1
        timeout-var: 'BUILD_TIMEOUT'
        type: deadline
```

timestamps()

Add timestamps to the console log.

Requires the Jenkins Timestamper Plugin.

Example:

```
wrappers:
  - timestamps
```

vault-secrets()

Inject environment variables from a HashiCorp Vault secret.

Secrets are generally masked in the build log.

Requires the Jenkins HashiCorp Vault Plugin.

Parameters

- **vault-url** (*str*) – Vault URL
- **credentials-id** (*str*) – Vault Credential
- **engine-version** (*str*) – Vault K/V Engine version
- **fail-if-not-found** (*bool*) – Fail if the secret path is not found
- **skip-ssl-verification** (*bool*) – Skip verification of SSL certs
- **secrets** (*list*) – List of secrets

secrets

- **secret-path** (*str*) – The path of the secret in the vault server
- **engine-version** (*str*) – Vault K/V Engine version

secret-values

- **secret-values** (*list*) – List of key / value pairs
  - **env-var** (*str*) – The environment variable to set with the value of the vault key
  - **vault-key** (*str*) – The vault key whose value with populate the environment variable

Minimal Example:

```
wrappers:
  - vault-secrets:
      vault-url: 'http://127.0.0.1:8200'
      credentials-id: 'myCredentials'
      secrets:
        - secret-path: 'secret/my-token'
          secret-values:
            - env-var: 'TOKEN'
              vault-key: 'token'
```

Full Example:

```
wrappers:
  - vault-secrets:
      vault-url: 'http://127.0.0.1:8200'
      credentials-id: 'myCredentials'
      fail-if-not-found: 'false'
      skip-ssl-verification: 'true'
      engine-version: '2'
      secrets:
        - secret-path: 'secret/my-secret'
          secret-values:
            - env-var: 'USERNAME'
              vault-key: 'username'
            - env-var: 'PASSWORD'
              vault-key: 'password'
        - secret-path: 'secret/my-secret2'
          engine-version: '2'
          secret-values:
            - env-var: 'USERNAME2'
              vault-key: 'username2'
            - env-var: 'PASSWORD2'
              vault-key: 'password2'
```

version-number()

Generate a version number for the build using a format string. See the wiki page for more detailed descriptions of options.

Requires the Jenkins Version number plugin.

Parameters

- **variable-name** (*str*) – Name of environment variable to assign version number to (required)
- **format-string** (*str*) – Format string used to generate version number (required)
- **prefix-variable** (*str*) – Variable that contains version number prefix (optional)
- **skip-failed-builds** (*bool*) – If the build fails, DO NOT increment any auto-incrementing component of the version number (default: false)
- **display-name** (*bool*) – Use the version number for the build display name (default: false)
- **start-date** (*str*) – The date the project began as a UTC timestamp (default 1970-1-1 00:00:00.0 UTC)
- **builds-today** (*int*) – The number of builds that have been executed today (optional)
- **builds-this-month** (*int*) – The number of builds that have been executed since the start of the month (optional)
- **builds-this-year** (*int*) – The number of builds that have been executed since the start of the year (optional)
- **builds-all-time** (*int*) – The number of builds that have been executed since the start of the project (optional)

Example:

```
wrappers:
  - version-number:
      variable-name: relVersion
      prefix-variable: relVersion
      format-string: ${BUILD_DATE_FORMATTED, "yy.M"}.${BUILDS_THIS_MONTH_Z}
```

workspace-cleanup (pre-build)()

Requires the Jenkins Workspace Cleanup Plugin.

The post-build workspace-cleanup is available as a publisher.

Parameters

- **include** (*list*) – list of files to be included
- **exclude** (*list*) – list of files to be excluded
- **dirmatch** (*bool*) – Apply pattern to directories too (default false)
- **check-parameter** (*str*) – boolean environment variable to check to determine whether to actually clean up
- **external-deletion-command** (*str*) – external deletion command to run against files and directories
- **disable-deferred-wipeout** (*bool*) – Disable improved deferred wipeout method (default false)

Full Example:

```
wrappers:
  - workspace-cleanup:
      include:
        - "_generated.py"
      exclude:
        - "*.py"
      dirmatch: true
      check-parameter: "DO_WS_CLEANUP"
      external-deletion-command: "shred -u %s"
      disable-deferred-wipeout: true
```

Minimal Example:

```
wrappers:
  - workspace-cleanup
```

xvfb()

Enable xvfb during the build.

Requires the Jenkins Xvfb Plugin.

Parameters

- **installation-name** (*str*) – The name of the Xvfb tool installation (default 'default')
- **auto-display-name** (*bool*) – Uses the -displayfd option of Xvfb by which it chooses it's own display name (default false)
- **display-name** (*str*) – Ordinal of the display Xvfb will be running on, if left empty chosen based on current build executor number (default '')
- **assigned-labels** (*str*) – If you want to start Xvfb only on specific nodes specify its name or label (default '')
- **parallel-build** (*bool*) – When running multiple Jenkins nodes on the same machine this setting influences the display number generation (default false)
- **timeout** (*int*) – A timeout of given seconds to wait before returning control to the job (default 0)
- **screen** (*str*) – Resolution and color depth. (default '1024x768x24')
- **display-name-offset** (*int*) – Offset for display names. (default 1)
- **additional-options** (*str*) – Additional options to be added with the options above to the Xvfb command line (default '')
- **debug** (*bool*) – If Xvfb output should appear in console log of this job (default false)
- **shutdown-with-build** (*bool*) – Should the display be kept until the whole job ends (default false)

Full Example:

```
wrappers:
  - xvfb:
      installation-name: default
      auto-display-name: false
      display-name: 123
      assigned-labels: nodes-xxx
      parallel-build: false
      timeout: 10
      screen: 1024x768x16
      display-name-offset: 100
      additional-options: -fbdir /tmp
      debug: true
      shutdown-with-build: false
```

Minimal Example:

```
wrappers:
  - xvfb
```

xvnc()

Enable xvnc during the build.

Requires the Jenkins xvnc plugin.

Parameters

- **screenshot** (*bool*) – Take screenshot upon build completion (default false)
- **xauthority** (*bool*) – Create a dedicated Xauthority file per build (default true)

Full Example:

```
wrappers:
  - xvnc:
      screenshot: true
      xauthority: false
```

Minimal Example:

```
wrappers:
  - xvnc
```

Zuul

The Zuul module adds jobs parameters to manually run a build as Zuul would have. It is entirely optional, Zuul 2.0+ pass the parameters over Gearman.

class zuul.**Zuul**(*registry*)

amend_job_dict(*job*)

This method is called before any XML is generated. By overriding this method, a module may arbitrarily modify a job data structure which will probably be the JJB Job intermediate data dict representation. If it has changed the data structure at all, it must return `True`, otherwise, it must return `False`.

Parameters

job (*dict*) – the intermediate representation of job data loaded from JJB Yaml files after variables interpolation and other yaml expansions.

Return type

bool

sequence = 0

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

zuul()

Configure this job to be triggered by Zuul.

Adds parameters describing the change triggering the build such as the branch name, change number and patchset.

See parameters expected by Zuul.

Example:

```
triggers:
  - zuul
```

zuul-post()

Configure this post-merge job to be triggered by Zuul.

Adds parameters describing the reference update triggering the build, which are the previous and next revisions in full (40 hexadecimal sha1) and short form.

See parameters expected by Zuul.

Example:

```
triggers:
  - zuul-post
```

2.5.4 Module Execution

The jenkins job builder modules are executed in sequence.

Generally the sequence is:

1. parameters/properties
2. scm
3. triggers
4. wrappers
5. prebuilders (maven only, configured like *Builders*)
6. builders (maven, freestyle, matrix, etc..)
7. postbuilders (maven only, configured like *Builders*)
8. publishers/reporters/notifications

2.6 Extending

Jenkins Job Builder is quite modular. It is easy to add new attributes to existing components, a new module to support a Jenkins plugin, or include locally defined methods to deal with an idiosyncratic build system.

2.6.1 The Builder

The `Builder` class manages Jenkins jobs. It's responsible for creating/deleting/updating jobs and can be called from your application. You can pass it a filename or an open file-like object that represents your YAML configuration. See the `jenkins_jobs/builder.py` file for more details.

2.6.2 XML Processing

Most of the work of building XML from the YAML configuration file is handled by individual functions that implement a single characteristic. For example, see the `jenkins_jobs/modules/builders.py` file for the Python module that implements the standard Jenkins builders. The `shell` function at the top of the file implements the standard *Execute a shell* build step. All of the YAML to XML functions in Jenkins Job Builder have the same signature:

`component`(*xml_parent*, *data*)

:arg class:*xml.etree.ElementTree* xml_parent: this attribute's parent XML element :arg dict data: the YAML data structure for this attribute and below

The function is expected to examine the YAML data structure and create new XML nodes and attach them to the xml_parent element. This general pattern is applied throughout the included modules.

2.6.3 Modules

Nearly all of Jenkins Job Builder is implemented in modules. The main program has no concept of builders, publishers, properties, or any other aspects of job definition. Each of those building blocks is defined in a module, and due to the use of setuptools entry points, most modules are easily extensible with new components.

To add a new module, define a class that inherits from *jenkins_jobs.modules.base.Base*, and add it to the `jenkins_jobs.modules` entry point in your setup.py.

class jenkins_jobs.modules.base.**Base**(*registry*)

A base class for a Jenkins Job Builder Module.

The module is initialized before any YAML is parsed.

Parameters

registry (ModuleRegistry) – the global module registry.

amend_job_dict(*job*)

This method is called before any XML is generated. By overriding this method, a module may arbitrarily modify a job data structure which will probably be the JJB Job intermediate data dict representation. If it has changed the data structure at all, it must return `True`, otherwise, it must return `False`.

Parameters

job (*dict*) – the intermediate representation of job data loaded from JJB Yaml files after variables interpolation and other yaml expansions.

Return type

bool

component_list_type = None

The component list type will be used to look up possible implementations of the component type via entry points (entry points provide a list of components, so it should be plural). Set both component_type and component_list_type to None if module doesn't have components.

component_type = None

The component type for components of this module. This will be used to look for macros (they are defined singularly, and should not be plural). Set both component_type and component_list_type to None if module doesn't have components.

gen_xml(*xml_parent*, *data*)

Update the XML element tree based on YAML data. Override this method to add elements to the XML output. Create new Element objects and add them to the xml_parent. The YAML data structure must not be modified.

:arg class:*xml.etree.ElementTree* xml_parent: the parent XML element :arg dict data: the YAML data structure

sequence = 10

The sequence number for the module. Modules are invoked in the order of their sequence number in order to produce consistently ordered XML output.

2.6.4 Components

Most of the standard modules supply a number of components, and it's easy to provide your own components for use by those modules. For instance, the Builders module provides several builders, such as the *shell* builder as well as the *trigger_builds* builder. If you wanted to add a new builder, all you need to do is write a function that conforms to the *Component Interface*, and then add that function to the appropriate entry point (via a setup.py file).

2.6.5 Module Registry

All modules and their associated components are registered in the module registry. It can be accessed either from modules via the registry field, or via the parser parameter of components.

class jenkins_jobs.registry.**ModuleRegistry**(*jjb_config*, *plugins_list=None*)

dispatch(*component_type*, *xml_parent*, *component*, *template_data={}*, *job_data=None*)

This is a method that you can call from your implementation of Base.gen_xml or component. It allows modules to define a type of component, and benefit from extensibility via Python entry points and Jenkins Job Builder *Macros*.

Parameters

- **component_type** (*str*) – the name of the component (e.g., *builder*)
- **xml_parent** – the parent XML element
- **component** – component definition
- **template_data** (*dict*) – values that should be interpolated into the component definition
- **job_data** (*dict*) – full job definition

See *jenkins_jobs.modules.base.Base* for how to register components of a module.

See the Publishers module for a simple example of how to use this method.

get_plugin_info(*plugin_name*)

Provide information about plugins within a module's impl of Base.gen_xml.

The return value is a dictionary with data obtained directly from a running Jenkins instance. This allows module authors to differentiate generated XML output based on information such as specific plugin versions.

Parameters

plugin_name (*str*) – Either the shortName or longName of a plugin as see in a query that looks like: `http://<jenkins-hostname>/pluginManager/api/json?pretty&depth=2`

During a 'test' run, it is possible to override JJB's query to a live Jenkins instance by passing it a path to a file containing a YAML list of dictionaries that mimics the plugin properties you want your test output to reflect:

```
jenkins-jobs test -p /path/to/plugins-info.yaml
```

Below is example YAML that might be included in /path/to/plugins-info.yaml.

```
- longName: 'Jenkins HipChat Plugin'
  shortName: 'hipchat'
  version: "0.1.8"
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

builders, 91

h

hipchat_notif, 145

j

jenkins_jobs.modules.general, 14
jenkins_jobs.yaml_objects, 32

m

metadata, 146

n

notifications, 148

p

parameters, 149
project_externaljob, 42
project_flow, 42
project_folder, 43
project_freestyle, 44
project_githuborg, 44
project_matrix, 49
project_maven, 53
project_multibranch, 59
project_multijob, 55
project_pipeline, 55
project_workflow, 58
properties, 168
publishers, 184

r

reporters, 282

s

scm, 285

t

triggers, 302

v

view_delivery_pipeline, 82
view_list, 84
view_nested, 89
view_pipeline, 90

w

wrappers, 330

z

zuul, 359

INDEX

A

accurev (*in module scm*), 286
accurev() (*in module scm*), 286
active-choices (*in module parameters*), 150
active-choices() (*in module parameters*), 150
active-choices-reactive (*in module parameters*), 151
active-choices-reactive() (*in module parameters*), 151
add_filter_branch_pr_behaviors() (*in module project_multibranch*), 61
add_filter_by_name_wildcard_behaviors() (*in module project_multibranch*), 62
add_github_checks_traits() (*in module project_multibranch*), 62
add_notification_context_trait() (*in module project_multibranch*), 62
aggregate-flow-tests (*in module publishers*), 184
aggregate-flow-tests() (*in module publishers*), 184
aggregate-tests (*in module publishers*), 184
aggregate-tests() (*in module publishers*), 184
allure (*in module publishers*), 184
allure() (*in module publishers*), 184
amend_job_dict() (*jenkins_jobs.modules.base.Base method*), 361
amend_job_dict() (*zuul.Zuul method*), 359
android-emulator (*in module wrappers*), 331
android-emulator() (*in module wrappers*), 331
ansible-playbook (*in module builders*), 92
ansible-playbook() (*in module builders*), 92
ansicolor (*in module wrappers*), 331
ansicolor() (*in module wrappers*), 331
ant (*in module builders*), 94
ant() (*in module builders*), 94
archive (*in module publishers*), 185
archive() (*in module publishers*), 185
artifact-deployer (*in module publishers*), 185
artifact-deployer() (*in module publishers*), 185
artifact-resolver (*in module builders*), 95
artifact-resolver() (*in module builders*), 95
artifactory (*in module publishers*), 186
artifactory (*in module triggers*), 302
artifactory() (*in module publishers*), 186
artifactory() (*in module triggers*), 302
artifactory-generic (*in module wrappers*), 332
artifactory-generic() (*in module wrappers*), 332
artifactory-maven (*in module wrappers*), 333
artifactory-maven() (*in module wrappers*), 333
artifactory-maven-freestyle (*in module wrappers*), 333
artifactory-maven-freestyle() (*in module wrappers*), 333
authenticated-build (*in module properties*), 169
authenticated-build() (*in module properties*), 169
authorization (*in module properties*), 169
authorization() (*in module properties*), 169

B

Base (*class in jenkins_jobs.modules.base*), 361
batch (*in module builders*), 95
batch() (*in module builders*), 95
batch-tasks (*in module properties*), 170
batch-tasks() (*in module properties*), 170
beaker (*in module builders*), 96
beaker() (*in module builders*), 96
bitbucket (*in module triggers*), 303
bitbucket() (*in module triggers*), 303
bitbucket_scm() (*in module project_multibranch*), 63
blame-upstream (*in module publishers*), 188
blame-upstream() (*in module publishers*), 188
bool (*in module parameters*), 152
bool() (*in module parameters*), 152
branch-api (*in module properties*), 170
branch-api() (*in module properties*), 170
build_strategies() (*in module project_multibranch*), 67
build-blocker (*in module properties*), 171
build-blocker() (*in module properties*), 171
build-discarder (*in module properties*), 172
build-discarder() (*in module properties*), 172
build-failure-analyzer (*in module properties*), 172
build-failure-analyzer() (*in module properties*), 172
build-keeper (*in module wrappers*), 335

build-keeper() (*in module wrappers*), 335
build-name (*in module wrappers*), 336
build-name() (*in module wrappers*), 336
build-name-setter (*in module builders*), 96
build-name-setter() (*in module builders*), 96
build-publish-docker-image (*in module builders*), 97
build-publish-docker-image() (*in module builders*), 97
build-publisher (*in module publishers*), 189
build-publisher() (*in module publishers*), 189
build-result (*in module triggers*), 303
build-result() (*in module triggers*), 303
build-user-vars (*in module wrappers*), 336
build-user-vars() (*in module wrappers*), 336
builders
    module, 91
Builders (*class in builders*), 91
builders-from (*in module builders*), 97
builders-from() (*in module builders*), 97
builds-chain-fingerprinter (*in module properties*), 172
builds-chain-fingerprinter() (*in module properties*), 172
bzr (*in module scm*), 287
bzr() (*in module scm*), 287

C

cachet-gating (*in module properties*), 172
cachet-gating() (*in module properties*), 172
campfire (*in module publishers*), 189
campfire() (*in module publishers*), 189
change-assembly-version (*in module builders*), 97
change-assembly-version() (*in module builders*), 97
checkstyle (*in module publishers*), 189
checkstyle() (*in module publishers*), 189
choice (*in module parameters*), 153
choice() (*in module parameters*), 153
chuck-norris (*in module publishers*), 191
chuck-norris() (*in module publishers*), 191
ci-skip (*in module wrappers*), 336
ci-skip() (*in module wrappers*), 336
cifs (*in module publishers*), 192
cifs() (*in module publishers*), 192
cigame (*in module publishers*), 192
cigame() (*in module publishers*), 192
claim-build (*in module publishers*), 193
claim-build() (*in module publishers*), 193
clamav (*in module publishers*), 193
clamav() (*in module publishers*), 193
clone-workspace (*in module publishers*), 193
clone-workspace() (*in module publishers*), 193
cloudformation (*in module builders*), 98
cloudformation (*in module publishers*), 195
cloudformation() (*in module builders*), 98
cloudformation() (*in module publishers*), 195
cloudfoundry (*in module publishers*), 193
cloudfoundry() (*in module publishers*), 193
cloverphp (*in module publishers*), 197
cloverphp() (*in module publishers*), 197
cmake (*in module builders*), 99
cmake() (*in module builders*), 99
cobertura (*in module publishers*), 198
cobertura() (*in module publishers*), 198
codecover (*in module publishers*), 199
codecover() (*in module publishers*), 199
component_list_type (*builders.Builders attribute*), 91
component_list_type (*jenkins_jobs.modules.base.Base attribute*), 361
component_list_type (*metadata.Metadata attribute*), 146
component_list_type (*notifications.Notifications attribute*), 148
component_list_type (*parameters.Parameters attribute*), 149
component_list_type (*properties.Properties attribute*), 168
component_list_type (*publishers.Publishers attribute*), 184
component_list_type (*reporters.Reporters attribute*), 282
component_list_type (*scm.PipelineSCM attribute*), 286
component_list_type (*scm.SCM attribute*), 286
component_list_type (*triggers.Triggers attribute*), 302
component_list_type (*wrappers.Wrappers attribute*), 330
component_type (*builders.Builders attribute*), 92
component_type (*jenkins_jobs.modules.base.Base attribute*), 361
component_type (*metadata.Metadata attribute*), 146
component_type (*notifications.Notifications attribute*), 148
component_type (*parameters.Parameters attribute*), 149
component_type (*properties.Properties attribute*), 168
component_type (*publishers.Publishers attribute*), 184
component_type (*reporters.Reporters attribute*), 282
component_type (*scm.PipelineSCM attribute*), 286
component_type (*scm.SCM attribute*), 286
component_type (*triggers.Triggers attribute*), 302
component_type (*wrappers.Wrappers attribute*), 330
conditional-publisher (*in module publishers*), 199
conditional-publisher() (*in module publishers*), 199
conditional-step (*in module builders*), 101
conditional-step() (*in module builders*), 101
config-file-provider (*in module builders*), 105

config-file-provider (*in module wrappers*), 337
config-file-provider() (*in module builders*), 105
config-file-provider() (*in module wrappers*), 337
copy-to-master (*in module publishers*), 203
copy-to-master() (*in module publishers*), 203
copy-to-slave (*in module wrappers*), 337
copy-to-slave() (*in module wrappers*), 337
copyartifact (*in module builders*), 106
copyartifact (*in module properties*), 173
copyartifact() (*in module builders*), 106
copyartifact() (*in module properties*), 173
copyartifact-build-selector (*in module parameters*), 153
copyartifact-build-selector() (*in module parameters*), 153
coverage (*in module publishers*), 203
coverage() (*in module publishers*), 203
cppcheck (*in module publishers*), 204
cppcheck() (*in module publishers*), 204
credentials (*in module parameters*), 153
credentials() (*in module parameters*), 153
credentials-binding (*in module wrappers*), 338
credentials-binding() (*in module wrappers*), 338
critical-block-end (*in module builders*), 108
critical-block-end() (*in module builders*), 108
critical-block-start (*in module builders*), 108
critical-block-start() (*in module builders*), 108
cucumber-reports (*in module publishers*), 205
cucumber-reports() (*in module publishers*), 205
cucumber-testresult (*in module publishers*), 207
cucumber-testresult() (*in module publishers*), 207
custom-tools (*in module wrappers*), 339
custom-tools() (*in module wrappers*), 339
cvs (*in module scm*), 288
cvs() (*in module scm*), 288

D

date (*in module metadata*), 147
date() (*in module metadata*), 147
delivery-pipeline (*in module properties*), 173
delivery-pipeline (*in module wrappers*), 340
delivery-pipeline() (*in module properties*), 173
delivery-pipeline() (*in module wrappers*), 340
DeliveryPipeline (*class in view_delivery_pipeline*), 84
dependency-check (*in module publishers*), 207
dependency-check() (*in module publishers*), 207
description-setter (*in module builders*), 109
description-setter (*in module publishers*), 209
description-setter() (*in module builders*), 109
description-setter() (*in module publishers*), 209
dimensions (*in module scm*), 291
dimensions() (*in module scm*), 291
disable-failed-job (*in module publishers*), 209
disable-failed-job() (*in module publishers*), 209
disable-resume (*in module properties*), 173
disable-resume() (*in module properties*), 173
discord-notifier (*in module publishers*), 210
discord-notifier() (*in module publishers*), 210
disk-usage (*in module properties*), 173
disk-usage() (*in module properties*), 173
dispatch() (*jenkins_jobs.registry.ModuleRegistry method*), 362
display-upstream-changes (*in module publishers*), 210
display-upstream-changes() (*in module publishers*), 210
docker-build-publish (*in module builders*), 109
docker-build-publish() (*in module builders*), 109
docker-container (*in module properties*), 174
docker-container() (*in module properties*), 174
docker-custom-build-env (*in module wrappers*), 340
docker-custom-build-env() (*in module wrappers*), 340
docker-pull-image (*in module builders*), 110
docker-pull-image() (*in module builders*), 110
docker-stop-container (*in module publishers*), 210
docker-stop-container() (*in module publishers*), 210
dockerhub-notification (*in module triggers*), 304
dockerhub-notification() (*in module triggers*), 304
downstream-ext (*in module publishers*), 211
downstream-ext() (*in module publishers*), 211
doxygen (*in module builders*), 111
doxygen (*in module publishers*), 211
doxygen() (*in module builders*), 111
doxygen() (*in module publishers*), 211
dry (*in module publishers*), 211
dry() (*in module publishers*), 211
dsl (*in module builders*), 111
dsl() (*in module builders*), 111
dynamic-choice (*in module parameters*), 154
dynamic-choice() (*in module parameters*), 154
dynamic-choice-scriptler (*in module parameters*), 154
dynamic-choice-scriptler() (*in module parameters*), 154
dynamic-reference (*in module parameters*), 155
dynamic-reference() (*in module parameters*), 155
dynamic-string (*in module parameters*), 156
dynamic-string() (*in module parameters*), 156
dynamic-string-scriptler (*in module parameters*), 157
dynamic-string-scriptler() (*in module parameters*), 157

E

email (*in module publishers*), 214

email (*in module reporters*), 283
email() (*in module publishers*), 214
email() (*in module reporters*), 283
email-ext (*in module publishers*), 214
email-ext() (*in module publishers*), 214
emotional-jenkins (*in module publishers*), 216
emotional-jenkins() (*in module publishers*), 216
env-file (*in module wrappers*), 341
env-file() (*in module wrappers*), 341
env-script (*in module wrappers*), 341
env-script() (*in module wrappers*), 341
exclusion (*in module wrappers*), 342
exclusion() (*in module wrappers*), 342
extended-choice (*in module parameters*), 157
extended-choice() (*in module parameters*), 157
ExternalJob (*class in project_externaljob*), 42

F

file (*in module parameters*), 160
file() (*in module parameters*), 160
findbugs (*in module publishers*), 216
findbugs (*in module reporters*), 283
findbugs() (*in module publishers*), 216
findbugs() (*in module reporters*), 283
fingerprint (*in module builders*), 112
fingerprint (*in module publishers*), 218
fingerprint() (*in module builders*), 112
fingerprint() (*in module publishers*), 218
fitnesse (*in module publishers*), 218
fitnesse() (*in module publishers*), 218
Flow (*class in project_flow*), 43
flowdock (*in module publishers*), 219
flowdock() (*in module publishers*), 219
Folder (*class in project_folder*), 44
Freestyle (*class in project_freestyle*), 44
ftp (*in module publishers*), 219
ftp() (*in module publishers*), 219
ftp-publisher (*in module publishers*), 220
ftp-publisher() (*in module publishers*), 220

G

gatling (*in module publishers*), 221
gatling() (*in module publishers*), 221
gen_xml() (*builders.Builders method*), 92
gen_xml() (*hipchat_notif.HipChat method*), 146
gen_xml() (*jenkins_jobs.modules.base.Base method*), 361
gen_xml() (*metadata.Metadata method*), 147
gen_xml() (*notifications.Notifications method*), 148
gen_xml() (*parameters.Parameters method*), 149
gen_xml() (*properties.Properties method*), 168
gen_xml() (*publishers.Publishers method*), 184
gen_xml() (*reporters.Reporters method*), 282
gen_xml() (*scm.PipelineSCM method*), 286
gen_xml() (*scm.SCM method*), 286
gen_xml() (*triggers.Triggers method*), 302
gen_xml() (*wrappers.Wrappers method*), 330
generic-webhook-trigger (*in module triggers*), 304
generic-webhook-trigger() (*in module triggers*), 304
gerrit (*in module triggers*), 306
gerrit() (*in module triggers*), 306
gerrit_scm() (*in module project_multibranch*), 68
get_plugin_info() (*jenkins_jobs.registry.ModuleRegistry method*), 362
git (*in module publishers*), 221
git (*in module scm*), 292
git() (*in module publishers*), 221
git() (*in module scm*), 292
git_scm() (*in module project_multibranch*), 71
git-parameter (*in module parameters*), 160
git-parameter() (*in module parameters*), 160
gitbucket (*in module properties*), 174
gitbucket() (*in module properties*), 174
github (*in module properties*), 175
github (*in module triggers*), 311
github() (*in module properties*), 175
github() (*in module triggers*), 311
github_org() (*in module project_githuborg*), 45
github_scm() (*in module project_multibranch*), 74
github-notifier (*in module builders*), 112
github-notifier (*in module publishers*), 222
github-notifier() (*in module builders*), 112
github-notifier() (*in module publishers*), 222
github-pull-request (*in module triggers*), 312
github-pull-request (*in module wrappers*), 342
github-pull-request() (*in module triggers*), 312
github-pull-request() (*in module wrappers*), 342
github-pull-request-merge (*in module publishers*), 222
github-pull-request-merge() (*in module publishers*), 222
GithubOrganization (*class in project_githuborg*), 45
gitlab (*in module properties*), 175
gitlab (*in module triggers*), 314
gitlab() (*in module properties*), 175
gitlab() (*in module triggers*), 314
gitlab-logo (*in module properties*), 175
gitlab-logo() (*in module properties*), 175
gitlab-merge-request (*in module triggers*), 316
gitlab-merge-request() (*in module triggers*), 316
gitlab-message (*in module publishers*), 223
gitlab-message() (*in module publishers*), 223
gitlab-notifier (*in module publishers*), 223
gitlab-notifier() (*in module publishers*), 223
gitlab-vote (*in module publishers*), 224
gitlab-vote() (*in module publishers*), 224

gogs (*in module properties*), 175
gogs (*in module triggers*), 317
gogs() (*in module properties*), 175
gogs() (*in module triggers*), 317
google-cloud-storage (*in module publishers*), 224
google-cloud-storage() (*in module publishers*), 224
gradle (*in module builders*), 112
gradle() (*in module builders*), 112
grails (*in module builders*), 113
grails() (*in module builders*), 113
groovy (*in module builders*), 114
groovy() (*in module builders*), 114
groovy-label (*in module properties*), 176
groovy-label() (*in module properties*), 176
groovy-postbuild (*in module publishers*), 226
groovy-postbuild() (*in module publishers*), 226
groovy-script (*in module triggers*), 317
groovy-script() (*in module triggers*), 317
growl (*in module publishers*), 226
growl() (*in module publishers*), 226

H

heavy-job (*in module properties*), 176
heavy-job() (*in module properties*), 176
hg (*in module scm*), 295
hg() (*in module scm*), 295
hidden (*in module parameters*), 161
hidden() (*in module parameters*), 161
HipChat (*class in hipchat_notif*), 146
hipchat (*in module publishers*), 227
hipchat() (*in module publishers*), 227
hipchat_notif
    module, 145
hp-alm (*in module publishers*), 227
hp-alm() (*in module publishers*), 227
html-publisher (*in module publishers*), 228
html-publisher() (*in module publishers*), 228
http (*in module notifications*), 148
http() (*in module notifications*), 148
http-request (*in module builders*), 115
http-request() (*in module builders*), 115
hue-light (*in module publishers*), 229
hue-light() (*in module publishers*), 229

I

image-gallery (*in module publishers*), 229
image-gallery() (*in module publishers*), 229
influx-db (*in module publishers*), 230
influx-db() (*in module publishers*), 230
inject (*in module builders*), 116
inject (*in module properties*), 177
inject (*in module wrappers*), 343
inject() (*in module builders*), 116
inject() (*in module properties*), 177
inject() (*in module wrappers*), 343
inject-ownership-variables (*in module wrappers*), 343
inject-ownership-variables() (*in module wrappers*), 343
inject-passwords (*in module wrappers*), 344
inject-passwords() (*in module wrappers*), 344
ircbot (*in module publishers*), 230
ircbot() (*in module publishers*), 230
ivy (*in module triggers*), 318
ivy() (*in module triggers*), 318

J

jabber (*in module publishers*), 231
jabber() (*in module publishers*), 231
jacoco (*in module publishers*), 232
jacoco() (*in module publishers*), 232
javadoc (*in module publishers*), 233
javadoc() (*in module publishers*), 233
jclouds (*in module publishers*), 233
jclouds (*in module wrappers*), 344
jclouds() (*in module publishers*), 233
jclouds() (*in module wrappers*), 344
jdepend (*in module publishers*), 234
jdepend() (*in module publishers*), 234
jenkins_jobs.modules.general
    module, 14
jenkins_jobs.yaml_objects
    module, 32
jenkins-jira-issue-updater (*in module builders*), 116
jenkins-jira-issue-updater() (*in module builders*), 116
jira (*in module publishers*), 234
jira() (*in module publishers*), 234
jira-changelog (*in module triggers*), 319
jira-changelog() (*in module triggers*), 319
jira-comment-trigger (*in module triggers*), 320
jira-comment-trigger() (*in module triggers*), 320
jms-messaging (*in module builders*), 117
jms-messaging (*in module publishers*), 234
jms-messaging (*in module triggers*), 320
jms-messaging() (*in module builders*), 117
jms-messaging() (*in module publishers*), 234
jms-messaging() (*in module triggers*), 320
job-log-logger (*in module wrappers*), 344
job-log-logger() (*in module wrappers*), 344
join-trigger (*in module publishers*), 235
join-trigger() (*in module publishers*), 235
junit (*in module publishers*), 236
junit() (*in module publishers*), 236

K

kmap (*in module builders*), 118

kmap() (*in module builders*), 118

L

label (*in module parameters*), 161
label() (*in module parameters*), 161
least-load (*in module properties*), 177
least-load() (*in module properties*), 177
List (*class in view_list*), 89
live-screenshot (*in module wrappers*), 345
live-screenshot() (*in module wrappers*), 345
lockable-resources (*in module properties*), 177
lockable-resources() (*in module properties*), 177
locks (*in module wrappers*), 345
locks() (*in module wrappers*), 345
logfilesize (*in module wrappers*), 345
logfilesize() (*in module wrappers*), 345
logparser (*in module publishers*), 236
logparser() (*in module publishers*), 236
logstash (*in module publishers*), 237
logstash build wrapper (*in module wrappers*), 346
logstash build wrapper() (*in module wrappers*), 346
logstash() (*in module publishers*), 237

M

m2-repository-cleanup (*in module wrappers*), 346
m2-repository-cleanup() (*in module wrappers*), 346
managed-script (*in module builders*), 119
managed-script() (*in module builders*), 119
mask-passwords (*in module wrappers*), 347
mask-passwords() (*in module wrappers*), 347
Matrix (*class in project_matrix*), 52
matrix-combinations (*in module parameters*), 162
matrix-combinations() (*in module parameters*), 162
matrix-tie-parent (*in module wrappers*), 347
matrix-tie-parent() (*in module wrappers*), 347
Maven (*class in project_maven*), 54
maven-builder (*in module builders*), 119
maven-builder() (*in module builders*), 119
maven-deploy (*in module publishers*), 237
maven-deploy() (*in module publishers*), 237
maven-metadata (*in module parameters*), 162
maven-metadata() (*in module parameters*), 162
maven-release (*in module wrappers*), 347
maven-release() (*in module wrappers*), 347
maven-target (*in module builders*), 120
maven-target() (*in module builders*), 120
metadata
    module, 146
Metadata (*class in metadata*), 146
module
    builders, 91
    hipchat_notif, 145
    jenkins_jobs.modules.general, 14
    jenkins_jobs.yaml_objects, 32
    metadata, 146
    notifications, 148
    parameters, 149
    project_externaljob, 42
    project_flow, 42
    project_folder, 43
    project_freestyle, 44
    project_githuborg, 44
    project_matrix, 49
    project_maven, 53
    project_multibranch, 59
    project_multijob, 55
    project_pipeline, 55
    project_workflow, 58
    properties, 168
    publishers, 184
    reporters, 282
    scm, 285
    triggers, 302
    view_delivery_pipeline, 82
    view_list, 84
    view_nested, 89
    view_pipeline, 90
    wrappers, 330
    zuul, 359
ModuleRegistry (*class in jenkins_jobs.registry*), 362
mongo-db build wrapper (*in module wrappers*), 348
mongo-db build wrapper() (*in module wrappers*), 348
monitor-files (*in module triggers*), 321
monitor-files() (*in module triggers*), 321
monitor-folders (*in module triggers*), 325
monitor-folders() (*in module triggers*), 325
mqtt (*in module publishers*), 238
mqtt() (*in module publishers*), 238
msbuild (*in module builders*), 121
msbuild() (*in module builders*), 121
MultiJob (*class in project_multijob*), 55
multijob (*in module builders*), 121
multijob() (*in module builders*), 121

N

naginator (*in module publishers*), 238
naginator() (*in module publishers*), 238
naginator-opt-out (*in module properties*), 178
naginator-opt-out() (*in module properties*), 178
Nested (*class in view_nested*), 90
nexus-artifact-uploader (*in module builders*), 123
nexus-artifact-uploader() (*in module builders*), 123
nexus-iq-policy-evaluator (*in module builders*), 124

nexus-iq-policy-evaluator() (*in module builders*), 124
nexus-repo-manager (*in module builders*), 125
nexus-repo-manager() (*in module builders*), 125
node (*in module parameters*), 163
node() (*in module parameters*), 163
nodejs (*in module builders*), 125
nodejs() (*in module builders*), 125
nodejs-installator (*in module wrappers*), 348
nodejs-installator() (*in module wrappers*), 348
notifications
    module, 148
Notifications (*class in notifications*), 148
number (*in module metadata*), 147
number() (*in module metadata*), 147

O

office-365-connector (*in module properties*), 178
office-365-connector() (*in module properties*), 178
openshift-build-canceller (*in module publishers*), 239
openshift-build-canceller() (*in module publishers*), 239
openshift-build-verify (*in module builders*), 125
openshift-build-verify() (*in module builders*), 125
openshift-builder (*in module builders*), 126
openshift-builder() (*in module builders*), 126
openshift-creator (*in module builders*), 127
openshift-creator() (*in module builders*), 127
openshift-dep-verify (*in module builders*), 127
openshift-dep-verify() (*in module builders*), 127
openshift-deploy-canceller (*in module publishers*), 239
openshift-deploy-canceller() (*in module publishers*), 239
openshift-deployer (*in module builders*), 128
openshift-deployer() (*in module builders*), 128
openshift-img-streams (*in module scm*), 296
openshift-img-streams() (*in module scm*), 296
openshift-img-tagger (*in module builders*), 128
openshift-img-tagger() (*in module builders*), 128
openshift-scaler (*in module builders*), 129
openshift-scaler() (*in module builders*), 129
openshift-svc-verify (*in module builders*), 130
openshift-svc-verify() (*in module builders*), 130
openstack (*in module wrappers*), 348
openstack() (*in module wrappers*), 348
opsgenie (*in module publishers*), 240
opsgenie() (*in module publishers*), 240
ownership (*in module properties*), 179
ownership() (*in module properties*), 179

P

p4 (*in module scm*), 296
p4() (*in module scm*), 296
packer (*in module publishers*), 240
packer() (*in module publishers*), 240
parameter-separator (*in module parameters*), 163
parameter-separator() (*in module parameters*), 163
parameterized-timer (*in module triggers*), 325
parameterized-timer() (*in module triggers*), 325
parameters
    module, 149
Parameters (*class in parameters*), 149
password (*in module parameters*), 164
password() (*in module parameters*), 164
pathignore (*in module wrappers*), 349
pathignore() (*in module wrappers*), 349
performance (*in module publishers*), 241
performance() (*in module publishers*), 241
persistent-bool (*in module parameters*), 164
persistent-bool() (*in module parameters*), 164
persistent-choice (*in module parameters*), 164
persistent-choice() (*in module parameters*), 164
persistent-string (*in module parameters*), 165
persistent-string() (*in module parameters*), 165
persistent-text (*in module parameters*), 165
persistent-text() (*in module parameters*), 165
phabricator (*in module publishers*), 242
phabricator() (*in module publishers*), 242
Pipeline (*class in project_pipeline*), 58
Pipeline (*class in view_pipeline*), 91
pipeline (*in module publishers*), 243
pipeline() (*in module publishers*), 243
PipelineSCM (*class in scm*), 286
plot (*in module publishers*), 244
plot() (*in module publishers*), 244
pmd (*in module publishers*), 246
pmd() (*in module publishers*), 246
pollscm (*in module triggers*), 325
pollscm() (*in module triggers*), 325
pollurl (*in module triggers*), 326
pollurl() (*in module triggers*), 326
port-allocator (*in module wrappers*), 349
port-allocator() (*in module wrappers*), 349
post-tasks (*in module publishers*), 248
post-tasks() (*in module publishers*), 248
postbuildscript (*in module publishers*), 248
postbuildscript() (*in module publishers*), 248
powershell (*in module builders*), 130
powershell() (*in module builders*), 130
pre-scm-buildstep (*in module wrappers*), 349
pre-scm-buildstep() (*in module wrappers*), 349
priority-sorter (*in module properties*), 180
priority-sorter() (*in module properties*), 180
project_externaljob
    module, 42
project_flow

module, 42
project_folder
module, 43
project_freestyle
module, 44
project_githuborg
module, 44
project_matrix
module, 49
project_maven
module, 53
project_multibranch
module, 59
project_multijob
module, 55
project_pipeline
module, 55
project_workflow
module, 58
promoted build (*in module parameters*), 166
promoted build() (*in module parameters*), 166
promoted-build (*in module properties*), 180
promoted-build() (*in module properties*), 180
properties
module, 168
Properties (*class in properties*), 168
property_strategies() (*in module project_multibranch*), 79
publish-over-cifs (*in module builders*), 130
publish-over-cifs() (*in module builders*), 130
publish-over-ssh (*in module builders*), 131
publish-over-ssh() (*in module builders*), 131
publishers
module, 184
Publishers (*class in publishers*), 184
publishers-from (*in module publishers*), 251
publishers-from() (*in module publishers*), 251
python (*in module builders*), 131
python() (*in module builders*), 131

R

rabbitmq (*in module triggers*), 327
rabbitmq() (*in module triggers*), 327
random-string (*in module parameters*), 166
random-string() (*in module parameters*), 166
rbenv (*in module wrappers*), 350
rbenv() (*in module wrappers*), 350
rebuild (*in module properties*), 180
rebuild() (*in module properties*), 180
release (*in module wrappers*), 350
release() (*in module wrappers*), 350
repo (*in module scm*), 296
repo() (*in module scm*), 296
reporters
module, 282
Reporters (*class in reporters*), 282
resource-gating (*in module properties*), 181
resource-gating() (*in module properties*), 181
reverse (*in module triggers*), 327
reverse() (*in module triggers*), 327
rich-text-publisher (*in module publishers*), 252
rich-text-publisher() (*in module publishers*), 252
robot (*in module publishers*), 252
robot() (*in module publishers*), 252
rocket (*in module publishers*), 253
rocket() (*in module publishers*), 253
ruby-metrics (*in module publishers*), 255
ruby-metrics() (*in module publishers*), 255
run (*in module parameters*), 166
run() (*in module parameters*), 166
rundeck (*in module publishers*), 255
rundeck() (*in module publishers*), 255
runscope (*in module builders*), 132
runscope() (*in module builders*), 132
rvm-env (*in module wrappers*), 351
rvm-env() (*in module wrappers*), 351

S

s3 (*in module publishers*), 256
s3() (*in module publishers*), 256
saltstack (*in module builders*), 132
saltstack() (*in module builders*), 132
sauce-ondemand (*in module wrappers*), 351
sauce-ondemand() (*in module wrappers*), 351
sbt (*in module builders*), 133
sbt() (*in module builders*), 133
scan-build (*in module builders*), 133
scan-build (*in module publishers*), 257
scan-build() (*in module builders*), 133
scan-build() (*in module publishers*), 257
scm
module, 285
SCM (*class in scm*), 286
scoverage (*in module publishers*), 257
scoverage() (*in module publishers*), 257
scp (*in module publishers*), 258
scp() (*in module publishers*), 258
script (*in module triggers*), 328
script() (*in module triggers*), 328
sequence (*builders.Builders attribute*), 92
sequence (*hipchat_notif.HipChat attribute*), 146
sequence (*jenkins_jobs.modules.base.Base attribute*), 361
sequence (*metadata.Metadata attribute*), 147
sequence (*notifications.Notifications attribute*), 148
sequence (*parameters.Parameters attribute*), 149
sequence (*project_externaljob.ExternalJob attribute*), 42

sequence (*project_flow.Flow attribute*), 43
sequence (*project_folder.Folder attribute*), 44
sequence (*project_freestyle.Freestyle attribute*), 44
sequence (*project_githuborg.GithubOrganization attribute*), 45
sequence (*project_matrix.Matrix attribute*), 52
sequence (*project_maven.Maven attribute*), 54
sequence (*project_multibranch.WorkflowMultiBranch attribute*), 61
sequence (*project_multijob.MultiJob attribute*), 55
sequence (*project_pipeline.Pipeline attribute*), 58
sequence (*project_workflow.Workflow attribute*), 59
sequence (*properties.Properties attribute*), 169
sequence (*publishers.Publishers attribute*), 184
sequence (*reporters.Reporters attribute*), 282
sequence (*scm.PipelineSCM attribute*), 286
sequence (*scm.SCM attribute*), 286
sequence (*triggers.Triggers attribute*), 302
sequence (*view_list.List attribute*), 89
sequence (*view_pipeline.Pipeline attribute*), 91
sequence (*wrappers.Wrappers attribute*), 330
sequence (*zuul.Zuul attribute*), 359
shell (*in module builders*), 134
shell() (*in module builders*), 134
shining-panda (*in module builders*), 134
shining-panda (*in module publishers*), 258
shining-panda() (*in module builders*), 134
shining-panda() (*in module publishers*), 258
sidebar (*in module properties*), 181
sidebar() (*in module properties*), 181
sitemonitor (*in module publishers*), 259
sitemonitor() (*in module publishers*), 259
slack (*in module properties*), 181
slack (*in module publishers*), 259
slack() (*in module properties*), 181
slack() (*in module publishers*), 259
slave-prerequisites (*in module properties*), 182
slave-prerequisites() (*in module properties*), 182
slave-utilization (*in module properties*), 182
slave-utilization() (*in module properties*), 182
sloccount (*in module publishers*), 261
sloccount() (*in module publishers*), 261
sonar (*in module builders*), 136
sonar (*in module publishers*), 262
sonar (*in module wrappers*), 352
sonar() (*in module builders*), 136
sonar() (*in module publishers*), 262
sonar() (*in module wrappers*), 352
sonatype-clm (*in module builders*), 136
sonatype-clm() (*in module builders*), 136
sounds (*in module publishers*), 263
sounds() (*in module publishers*), 263
speed-durability (*in module properties*), 183
speed-durability() (*in module properties*), 183
ssh (*in module publishers*), 263
ssh() (*in module publishers*), 263
ssh-agent-credentials (*in module wrappers*), 353
ssh-agent-credentials() (*in module wrappers*), 353
ssh-builder (*in module builders*), 137
ssh-builder() (*in module builders*), 137
stash (*in module publishers*), 264
stash() (*in module publishers*), 264
stash-pull-request (*in module triggers*), 328
stash-pull-request() (*in module triggers*), 328
store (*in module scm*), 298
store() (*in module scm*), 298
string (*in module metadata*), 147
string (*in module parameters*), 166
string() (*in module metadata*), 147
string() (*in module parameters*), 166
svn (*in module scm*), 298
svn() (*in module scm*), 298
svn-tags (*in module parameters*), 167
svn-tags() (*in module parameters*), 167
system-groovy (*in module builders*), 137
system-groovy() (*in module builders*), 137

T

tap (*in module publishers*), 265
tap() (*in module publishers*), 265
tasks (*in module publishers*), 266
tasks() (*in module publishers*), 266
test-fairy (*in module publishers*), 269
test-fairy() (*in module publishers*), 269
testng (*in module publishers*), 270
testng() (*in module publishers*), 270
testselector (*in module publishers*), 271
testselector() (*in module publishers*), 271
text (*in module parameters*), 167
text() (*in module parameters*), 167
text-finder (*in module publishers*), 272
text-finder() (*in module publishers*), 272
tfs (*in module scm*), 300
tfs() (*in module scm*), 300
throttle (*in module properties*), 183
throttle() (*in module properties*), 183
timed (*in module triggers*), 330
timed() (*in module triggers*), 330
timeout (*in module wrappers*), 353
timeout() (*in module wrappers*), 353
timestamps (*in module wrappers*), 355
timestamps() (*in module wrappers*), 355
tox (*in module builders*), 138
tox() (*in module builders*), 138
trigger (*in module publishers*), 272
trigger() (*in module publishers*), 272
trigger-builds (*in module builders*), 138
trigger-builds() (*in module builders*), 138

trigger-parameterized-builds (*in module publishers*), 272
trigger-parameterized-builds() (*in module publishers*), 272
trigger-remote (*in module builders*), 141
trigger-remote() (*in module builders*), 141
triggers
    module, 302
Triggers (*class in triggers*), 302

U

url (*in module scm*), 301
url() (*in module scm*), 301

V

valgrind (*in module publishers*), 274
valgrind() (*in module publishers*), 274
validating-string (*in module parameters*), 167
validating-string() (*in module parameters*), 167
vault-secrets (*in module wrappers*), 355
vault-secrets() (*in module wrappers*), 355
version-number (*in module wrappers*), 356
version-number() (*in module wrappers*), 356
view_delivery_pipeline
    module, 82
view_list
    module, 84
view_nested
    module, 89
view_pipeline
    module, 90
violations (*in module publishers*), 275
violations() (*in module publishers*), 275

W

warnings (*in module publishers*), 276
warnings() (*in module publishers*), 276
whitesource (*in module publishers*), 279
whitesource() (*in module publishers*), 279
Workflow (*class in project_workflow*), 59
WorkflowMultiBranch (*class in project_multibranch*), 61
WorkflowMultiBranchDefaults (*class in project_multibranch*), 61
workspace (*in module scm*), 301
workspace() (*in module scm*), 301
workspace-cleanup (post-build) (*in module publishers*), 279
workspace-cleanup (post-build)() (*in module publishers*), 279
workspace-cleanup (pre-build) (*in module wrappers*), 357
workspace-cleanup (pre-build)() (*in module wrappers*), 357
wrappers
    module, 330
Wrappers (*class in wrappers*), 330

X

xcode (*in module builders*), 141
xcode() (*in module builders*), 141
xml-summary (*in module publishers*), 280
xml-summary() (*in module publishers*), 280
xunit (*in module builders*), 143
xunit (*in module publishers*), 280
xunit() (*in module builders*), 143
xunit() (*in module publishers*), 280
xvfb (*in module wrappers*), 358
xvfb() (*in module wrappers*), 358
xvnc (*in module wrappers*), 358
xvnc() (*in module wrappers*), 358

Z

zeromq-event (*in module properties*), 183
zeromq-event() (*in module properties*), 183
zulip (*in module publishers*), 282
zulip() (*in module publishers*), 282
zuul
    module, 359
Zuul (*class in zuul*), 359
zuul (*in module zuul*), 359
zuul() (*in module zuul*), 359
zuul-post (*in module zuul*), 359
zuul-post() (*in module zuul*), 359