Jenkins Job Builder Documentation

Release 1.4.1.dev104

Jenkins Job Builder Maintainers

March 21, 2016

Contents

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:

```
$ sudo python setup.py install
```

Online documentation:

- http://docs.openstack.org/infra/jenkins-job-builder/

1.1 Developers

Bug report:

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

Repository:

- https://git.openstack.org/cgit/openstack-infra/jenkins-job-builder

Cloning:

```
git clone https://git.openstack.org/openstack-infra/jenkins-job-builder
```

Patches are submitted via Gerrit at:

- https://review.openstack.org/

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

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

- http://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 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 2

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/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/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/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.

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:

```
sudo python setup.py install
```

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`.

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 py27
```

- 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 in `/etc/jenkins_jobs/jenkins_jobs.ini` 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

[jenkins]
user=jenkins
password=1234567890abcdef1234567890abcdef
url=https://jenkins.example.com
query_plugins_info=False
##### This is deprecated, use job_builder section instead
#ignore_cache=True

[hipchat]
authtoken=dummy

[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-escaped'.

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.

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. True 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.

2.4 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.

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 definitions to Jenkins if needed. Jenkins Job Builder maintains, for each host, a cache [1] of previously configured jobs, 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 has 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, simply pass the job names as additional arguments after the job definition path. To update Foo1 and Foo2 run:

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

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
```

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.

2.5 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/
```

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

```
      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.

2.6 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/
      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.

2.6.1 Deleting Jobs

Jenkins Job Builder supports deleting jobs from Jenkins.

To delete a specific job:

```
jenkins-jobs delete Foo1
```

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

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

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

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

Obsolete jobs are *all* jobs not managed by JJB, even jobs which were *never* managed by JJB.

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

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

2.6.2 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 that only have 'foo' in their name:

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

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

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

2.6.3 Command Reference

```
usage: jenkins-jobs [-h] [--conf CONF] [-l LOG_LEVEL] [--ignore-cache]
                    [--flush-cache] [--version] [--allow-empty-variables]
                    {update,test,delete,delete-all} ...

positional arguments:
  {update,test,delete,delete-all}
                        update, test or delete job
    delete-all          delete *ALL* jobs from Jenkins server, including those
                        not managed by Jenkins Job Builder.

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

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

positional arguments:
  path                  colon-separated list of paths to YAML files or
                        directories
  name                  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.
  -p 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]
                           [--workers N_WORKERS]
                           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.
  --delete-old          delete obsolete jobs
  --workers N_WORKERS   number of workers to use, 0 for autodetection and 1
                        for just one worker.
```

```
usage: jenkins-jobs delete-all [-h]

optional arguments:
  -h, --help  show this help message and exit
```

```
usage: jenkins-jobs delete [-h] [-r] [-x EXCLUDE] [-p PATH] 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
```

2.7 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.

2.7.1 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
    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", "workflow" 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.
- **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
- **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.

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}'
```

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.

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.

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/>
            </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`.

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.

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

- 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
```

2.7.2 Custom Yaml Tags

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

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.

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:

```
- template-job:
    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.

2.7.3 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
```

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
```

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
```

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

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

- **execution-strategy (optional):**
  - **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
- **axes (*list*):**
  - **axis:**
    - **type** (*str*) – axis type, must be either 'label-expression', 'user-defined', 'slave' or 'jdk'.

* **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:
          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
```

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).
- **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).

- **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. It is possible to provide a ConfigFileProvider settings file as well see CFP Example below. (optional)
- **global-settings** (*str*): Path to custom maven global settings file. It is possible to provide a ConfigFileProvider settings file as well see CFP Example below. (optional)
- **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
      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.GlobalMavenSettingsConfig012345678901
      post-step-run-condition: SUCCESS
```

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
```

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"
```

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:
        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
- **artifacts** (*list*) – list of artifacts to resolve

  Artifact

  - **group-id** (*str*) – Group ID of the artifact
  - **artifact-id** (*str*) – Artifact ID of the artifact
  - **version** (*str*) – Version of the artifact
  - **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 '')

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>
```

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)

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** (*array*) – 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
        - 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 Hudson cmakebuilder Plugin.

Parameters

- **source-dir** (*str*) – the source code directory relative to the workspace directory. (required)
- **build-dir** (*str*) – The directory where the project will be built in. Relative to the workspace directory. (optional)
- **install-dir** (*list*) – 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)

  type Default types present in the CMake plugin

  - **Debug**
  - **Release**
  - **RelWithDebInfo**
  - **MinSizeRel**

- **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**

- **make-command** (*str*) – The make command (default "make").
- **install-command** (*str*) – The install command (default "make install").
- **preload-script** (*str*) – Path to a CMake preload script file. (optional)
- **other-arguments** (*str*) – Other arguments to be added to the CMake call. (optional)
- **custom-cmake-path** (*str*) – Path to cmake executable. (optional)
- **clean-build-dir** (*bool*) – If true, delete the build directory before each build (default false).
- **clean-install-dir** (*bool*) – If true, delete the install dir before each build (default false).

Example:

```
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** build was triggered via remote interface. **UPSTREAM_CAUSE** build was triggered by an upstream project. Following supported if XTrigger plugin installed: **FS_CAUSE** build was triggered by a file system change (FSTrigger Plugin). **URL_CAUSE** build was triggered by a URL change (URLTrigger

Example:

```
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"
        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 (optional)
- **variable** (*str*) – Define an environment variable to be used (optional)

Example:

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

```
                    target: target
                    variable: variable
```

copyartifact

Copy artifact from another project. Requires the Copy Artifact 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**
- **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** (*string*) – 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** (*string*) – Filter matching jobs based on these parameters (optional)

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
```

critical-block-end

Designate the end of a critical block. Must be used in conjuction 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 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 conjuction 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 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
```

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"
```

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 target is not specified)
- **target** (*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)

Example:

```
builders:
  - dsl:
      script-text: "job { name 'dsljob' }"
      ignore-existing: "true"
      removed-job-action: "DISABLE"
      removed-view-action: "DELETE"
      lookup-strategy: "SEED_JOB"
      additional-classpath: "*.jar"
```

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

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 (optional)
- **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}.

Example:

```
builders:
  - gradle:
      gradle-name: "gradle-1.2"
      wrapper: true
      executable: true
      use-root-dir: true
      root-build-script-dir: ${workspace}/tests
      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 (optional)
- **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
- **server-port** (*str*) – Specify a value for the server.port system property (optional)
- **work-dir** (*str*) – Specify a value for the grails.work.dir system property (optional)
- **project-dir** (*str*) – Specify a value for the grails.project.work.dir system property (optional)
- **base-dir** (*str*) – Specify a path to the root of the Grails project (optional)
- **properties** (*str*) – Additional system properties to set (optional)

- **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)

Example:

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

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. (optional)
- **script-parameters** (*str*) – These parameters will be passed to the script. (optional)
- **properties** (*str*) – Instead of passing properties using the -D parameter you can define them here. (optional)
- **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. (optional)
- **class-path** (*str*) – Specify script classpath here. Each line is one class path item. (optional)

Examples:

```
builders:
  - groovy:
      file: "test.groovy"
```

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

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
```

managed-script

This step allows to reference and execute a centrally managed script within your build. Requires the Jenkins Managed Script 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

Parameters

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

Requires the Jenkins Artifactory Plugin allows your build jobs to deploy artifacts automatically to Artifactory.

Example:

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

maven-target

Execute top-level Maven targets

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)
- **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)

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

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.GlobalMavenSettingsConfig0123
```

msbuild

Build .NET project using msbuild. Requires the Jenkins MSBuild Plugin.

Parameters

- **msbuild-version** (*str*) – which msbuild configured in Jenkins to use (optional)
- **solution-file** (*str*) – location of the solution file to build
- **extra-parameters** (*str*) – extra parameters to pass to msbuild (optional)
- **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)

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
```

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'. (default 'SUCCESSFUL')
- **projects** (*list*) – list of projects to include in the MultiJob phase

  Project

  – **name** (*str*) – Project name

  – **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)
- **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)
- **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)

Example:

```
builders:
  - multijob:
      name: PhaseOne
      condition: SUCCESSFUL
      projects:
        - name: PhaseOneJobA
          current-parameters: true
          node-label-name: "vm_name"
          node-label: "agent-${BUILD_NUMBER}"
          git-revision: true
          abort-all-job: true
        - name: PhaseOneJobB
          current-parameters: true
          property-file: build.props
  - multijob:
      name: PhaseTwo
      condition: UNSTABLE
      projects:
        - name: PhaseTwoJobA
          current-parameters: true
          predefined-parameters: foo=bar
        - name: PhaseTwoJobB
          current-parameters: false
          kill-phase-on: UNSTABLE
          enable-condition: "${BUILDNUMBER} % 2 == 1"
          restrict-matrix-project: 'JVM_VARIANT == "server"'
```

`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. (optional)
- **verbose** (*str*) – 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. (optional)
- **commit-ID** (*str*) – The value here is what you supply with the –commit option when invoking the OpenShift *oc start-build* command. (optional)
- **verbose** (*str*) – 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. (optional)
- **show-build-logs** (*str*) – 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
      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. (optional)
- **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. (optional)
- **verbose** (*str*) – 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** (*str*) – 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. (optional)
- **verbose** (*str*) – 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. (optional)
- **verbose** (*str*) – 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. (optional)
- **verbose** (*str*) – 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. (optional)
- **verbose** (*str*) – 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
      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. (optional)
- **verbose** (*str*) – 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"
```

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)

Example:

```
builders:
  - runscope:
      test-trigger-url: "https://api.runscope.com/radar/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/tri
      access-token: "123456"
      timeout: 60
```

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"
```

shell

Execute a shell command.

Parameters **parameter** (*str*) – the shell command to execute

Example:

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

`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
            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.
- **task** (*str*) – Task to run. (optional)
- **project** (*str*) – Path to Sonar project properties file. (optional)
- **properties** (*str*) – Sonar configuration properties. (optional)
- **java-opts** (*str*) – Java options for Sonnar Runner. (optional)
- **jdk** (*str*) – JDK to use (inherited from the job if omitted). (optional)

Example:

```
    builders:
      - sonar:
          sonar-name: Sonar
          task: views
          project: sonar-project.properties
          properties: sonar.views.list=myview1,myview2
          java-opts: -Xmx512m
```

sonatype-clm

Requires the Jenkins Sonatype CLM Plugin.

Parameters

- **application-name** (*str*) – Determines the policy elements to associate with this build. (required)

- **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. (optional)
- **module-excludes** (*str*) – Pattern of files to exclude. (optional)
- **advanced-options** (*str*) – Options to be set on a case-by-case basis as advised by Sonatype Support. (optional)

Example:

```
builders:
    - sonatype-clm:
        application-name: jenkins-job-builder
        fail-on-clm-server-failure: true
        stage: release
        scan-targets: '**/*.jar'
        module-excludes: '**/my-module/target/**'
        advanced-options: 'test'
```

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 chose a script file instead)
- **bindings** (*str*) – Define variable bindings (in the properties file format). Specified variables can be addressed from the script. (optional)
- **class-path** (*str*) – Specify script classpath here. Each line is one class path item. (optional)

Examples:

```
builders:
  - system-groovy:
      file: "test.groovy"
```

```
builders:
  - system-groovy:
      command: "Some command"
      bindings: "Some bindings"
      class-path: "Some classpath"
```

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.
- **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)
- **svn-revision** (*bool*) – Whether to pass the svn revision to the triggered job (optional)
- **git-revision** (*bool*) – Whether to pass the git revision to the triggered job (optional)
- **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)

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
          name: parametername
```

```
              node-label: labelname
              ignore-offline-nodes: false
          block: true
```

`trigger-remote`

Trigger build of job on remote Jenkins instance.

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)
- **`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
      property-file: '.props'
      predefined-parameters: |
        FOO="bar"
        herp="derp"
```

Hipchat

Enable HipChat notifications of build execution.

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
  room: My 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
```

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
```

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

`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`** (*`str`*) – Timeout in milliseconds for sending notification request (30 seconds by default)
- **`log`** (*`str`*) – 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
```

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'."
```

`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
- **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'
```

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-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, radio, checkbox or textbox (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 ‘’)
- **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 ‘’)
- **multi-select-delimiter** (*str*) – value between selections when the parameter is a multi-select (optiona, default ‘,’)

Example:

```
parameters:
  - extended-choice:
      name: OPTIONS
      description: "Available options"
      type: 'PT_CHECKBOX'
      value: OptionA,OptionB,OptionC
      visible-item-count: "2"
```

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."
```

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)

Example:

```
parameters:
  - label:
      name: node
      default: precise
      description: "The node on which to run the job"
```

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
```

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."
```

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)

Example:

```
parameters:
  - string:
      name: FOO
      default: bar
      description: "A parameter named FOO, defaults to 'bar'."
```

svn-tags

A svn tag parameter Requires the Jenkins Parameterized Trigger 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)
- **url** (*str*) – the url to list tags from
- **filter** (*str*) – the regular expression to filter tags

Example:

```
parameters:
  - svn-tags:
      name: BRANCH_NAME
      default: release
      description: A parameter named BRANCH_NAME default is release
      url: http://svn.example.com/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/
```

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

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-update**
- **scm-tag**

Example:

```
properties:
    - authorization:
        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-update
            - scm-tag
        anonymous:
            - job-read
            - 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"
```

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:

```
properties:
  - build-blocker:
      use-build-blocker: true
      blocking-jobs:
        - ".*-deploy"
        - "^maintenance.*"
      block_level: 'GLOBAL'
      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
```

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
```

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 '')

Example:

```
properties:
  - delivery-pipeline:
      stage: Stage
      task: Task
      description: Task-Description
```

extended-choice

Use of this config option is deprecated. You should use the *extended-choice* option in the parameter section of the job configuration instead.

github

Sets the GitHub URL for the project.

Parameters **url** (*str*) – the GitHub URL (required)

Example:

```
properties:
  - github:
      url: https://github.com/openstack-infra/jenkins-job-builder/
```

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 Env Inject 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)
- **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
```

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 100 as the standard priority. (required)

Example:

```
properties:
  - priority-sorter:
      priority: 150
```

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

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)

Example:

```
properties:
    - rebuild:
        auto-rebuild: true
        rebuild-disabled: true
```

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

As the Slack Plugin itself requires a publisher aswell as properties please note that you have to add the publisher to your job configuration aswell.

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)
- **notifiy-back-to-normal** (*bool*) – Send notification when job is succeeding again after being unstable or failed. (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 seperated list of rooms / channels to send the notifications to. (default: '')

Example:

```
properties:
    - slack:
        room: dummy, dummy2
        notify-start: true
        notify-success: true
```

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:
      slave-percentage: 40
      single-instance-per-slave: false
```

throttle

Throttles the number of builds for this job. Requires the Jenkins Throttle Concurrent Builds Plugin.

Parameters

- **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)
- **option** (*str*) – throttle *project* or *category*
- **categories** (*list*) – multiproject throttle categories
- **matrix-builds** (*bool*) – throttle matrix master builds (default true)
- **matrix-configs** (*bool*) – throttle matrix config builds (default false)

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

aggregate-flow-tests

Aggregate downstream test results in a Build Flow job. Requires the Jenkins Build Flow Test Aggregator Plugin.

Example:

```
publishers:
  - aggregate-flow-tests
```

aggregate-tests

Aggregate downstream test results

Parameters **include-failed-builds** (*bool*) – whether to include failed builds

Example:

```
publishers:
  - aggregate-tests:
      include-failed-builds: 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)
- **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 to enable or disable the default Ant exclusions. (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. :jenkins-wiki: 'Artifactory Plugin <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 commiters when build fails Requires the Jenkins Blame upstream commiters 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)

Example:

```
publishers:
  - build-publisher:
      publish-unstable-builds: true
      publish-failed-builds: true
      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.

The checkstyle 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-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:
 - checkstyle:
    pattern: '**/checkstyle-result.xml'
    healthy: 0
    unhealthy: 100
    health-threshold: 'high'
    thresholds:
        unstable:
            total-high: 10
        failed:
            total-high: 1
```

Full example:

```
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
```

`cifs`

Upload files via CIFS. Requires the Jenkins Publish over CIFS Plugin.

Parameters

- **`site`** (*`str`*) – name of the cifs site/share
- **`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:

```
publishers:
  - cifs:
      site: 'cifs.share'
      target: 'dest/dir'
      source: 'base/source/dir/**'
      remove-prefix: 'base/source/dir'
      excludes: '**/*.excludedfiletype'
      flatten: true
```

cigame

This plugin introduces a game where users get points for improving the builds. Requires the Jenkins The Continuous Integration 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*) – Files that should be scanned. (default "")
- **excludes** (*str*) – Files that should be ignored. (default "")

Example:

```
publishers:
  - clamav:
      includes: '*.zip'
      excludes: 'foo.zip'
```

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
- **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"
```

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:

```
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
```

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.

- **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`** (*`string`*) – absolute path into which the files will be copied. If left blank they will be copied into the workspace of the current job

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 for more optional parameters see the example

Example:

```
publishers:
  - cppcheck:
      pattern: "**/cppcheck.xml"
      # the rest is optional
      # build status (new) error count thresholds
      thresholds:
        unstable: 5
        new-unstable: 5
        failure: 7
        new-failure: 3
        # severities which count towards the threshold, default all true
        severity:
          error: true
          warning: true
          information: false
      graph:
        xysize: [500, 200]
        # which errors to display, default only sum
        display:
          sum: false
          error: 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)

Example:

```
publishers:
- cucumber-reports:
    plugin-url-path: http://example.com/
```

```
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
```

`cucumber-testresult`

Publish cucumber test results. Requires the Jenkins cucumber testresult.

Parameters **`results`** (*str*) – results filename (required)

Example:

```
publishers:
- cucumber-testresult:
    results: nosetests.xml
```

`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
- **`regexp-for-failed`** (*str*) – A RegEx which is used for failed builds (optional)
- **`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)

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
```

display-upstream-changes

Display SCM changes of upstream jobs. Requires the Jenkins Display Upstream Changes Plugin.

Example:

```
publishers:
  - display-upstream-changes
```

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** (*string*) – comparison condition used for the criteria. One of 'equal-or-over', 'equal-or-under', 'equal' (default: 'equal-or-over')
- **criteria** (*string*) – 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
- **slave** (*str*) – The node or label to pull the doxygen HTML files from
- **keep-all** (*bool*) – Retain doxygen generation for each successful build (default: false)
- **folder** (*str*) – Folder where you run doxygen (default: '')

Example:

```
publishers:
  - doxygen:
      doxyfile: "Doxyfile"
      slave: "doxygen-slave"
```

```
        keep-all: false
        folder: "build"
```

dry

Publish trend reports with DRY. Requires the Jenkins DRY Plugin.

The DRY 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')
- **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. (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-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
            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.

Parameters

- **recipients** (*str*) – Recipient email addresses (optional)
- **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 bar@example.com
```

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 emails
- **reply-to** (*str*) – Comma separated list of emails that should be in the Reply-To header for this project (default $DEFAULT_REPLYTO)
- **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
- **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
- **attach-build-log** (*bool*) – Include build log in the email (default false)
- **attachments** (*str*) – pattern of files to include as attachment (optional)
- **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)
- **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)
- **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)

- **`presend-script`** (*`str`*) – A Groovy script executed prior sending the mail. (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)

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
      attachments: "*/foo*.log"
      always: true
      unstable: true
      first-failure: true
      not-built: true
      aborted: true
      regression: true
      failure: true
      second-failure: true
      improvement: true
      still-failing: true
      success: true
      fixed: true
      still-unstable: true
      pre-build: true
      matrix-trigger: only-configurations
      presend-script: "cancel=true"
      save-output: true
      send-to:
        - developers
        - requester
        - culprits
        - recipients
```

`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.

Parameters

- **`pattern`** (*str*) – specifies the generated raw FindBugs XML report files, such as **/findbugs.xml or **/findbugsXml.xml. (Optional)
- **`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)
- **`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')
- **`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:

```
publishers:
    - findbugs:
        pattern: '**/findbugs.xml'
        rank-priority: true
        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

Parameters

- **`files`** (`str`) – files to fingerprint, follows the @includes of Ant fileset (default blank)
- **`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 incude 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
- **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:

```
publishers:
- ftp:
    site: 'ftp.example.com'
    target: 'dest/dir'
    source: 'base/source/dir/**'
    remove-prefix: 'base/source/dir'
    excludes: '**/*.excludedfiletype'
    flatten: 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
- **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)

Example:

```
publishers:
  - git:
      push-merge: true
      push-only-if-success: false
      tags:
          - tag:
              remote: tagremotename
              name: tagname
              message: "some tag message"
              create-tag: true
              update-tag: true
      branches:
          - branch:
              remote: branchremotename
              name: "some/branch"
      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** (*bool*) – Comment to set on the merge commit (optional)
- **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:
      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
```

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)

    * **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)
    * **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'
                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 is "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
```

hipchat

Publisher that sends hipchat notifications on job events Requires the Jenkins Hipchat Plugin version >=1.9

Please see documentation for older plugin version http://docs.openstack.org/infra/jenkins-job-builder/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
```

html-publisher

This plugin publishes HTML reports.

Requires the Jenkins HTML Publisher Plugin.

Parameters

- **name** (*str*) – Report name
- **dir** (*str*) – HTML directory to archive
- **files** (*str*) – Specify the pages to display
- **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** (*string*) – Colour of building state (default 'blue')
- **good-build** (*string*) – Colour of succesful state (default 'green')
- **unstable-build** (*string*) – Colour of unstable state (default 'yellow')
- **bad-build** (*string*) – Colour of unsuccessful state (default 'red')

Example:

```
publishers:
  - hue-light:
      light-id: 123
```

```
publishers:
  - hue-light:
      light-id: 123
      pre-build: blue
      good-build: green
      unstable-build: yellow
      bad-build: red
```

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
```

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** (*string*) – 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`** (*`string`*) – 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`** (*`string`*) –

  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**

Example:

```
publishers:
  - ircbot:
      strategy: all
      notify-start: false
      notify-committers: false
      notify-culprits: false
      notify-upstream: false
      notify-fixers: false
      message-type: summary-scm
      channels:
          - name: '#jenkins-channel1'
            password: secrete
            notify-only: false
          - name: '#jenkins-channel2'
            notify-only: true
      matrix-notifier: only-configurations
```

`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
  - **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

Example:

```
publishers:
  - jabber:
      notify-on-build-start: true
      group-targets:
        - "foo-room@conference-2-fooserver.foo.com"
      individual-targets:
        - "foo-user@conference-2-fooserver.foo.com"
      strategy: all
      message: summary-scm
```

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`)
- **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 (optional)
- **exclusion-pattern** (*str*) – This is a file name pattern that can be used to exclude certain class files (optional)
- **targets** (*dict*) –

targets (instruction, branch, complexity, line, method, class)

- **healthy** (*int*): Healthy threshold (default 0)
- **unhealthy** (*int*): Unhealthy threshold (default 0)

Example:

```
publishers:
  - jacoco:
      exec-pattern: "**/**.exec"
      class-pattern: "**/classes"
      source-pattern: "**/src/main/java"
      status-update: true
      targets:
        - branch:
            healthy: 10
            unhealthy: 20
        - method:
            healthy: 50
            unhealthy: 40
```

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:

```
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:
```

`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
```

`join-trigger`

Trigger a job after all the immediate downstream jobs have completed

Parameters

- **`even-if-unstable`** (*bool*) – if true jobs will trigger even if some downstream jobs are marked as unstable (default false)
- **`projects`** (*list*) – list of projects to trigger
- **`publishers`** (*list*) – list of triggers from publishers module that defines projects that need to be triggered

Example:

```
publishers:
  - join-trigger:
      projects:
        - project-one
        - project-two
      even-if-unstable: true
      publishers:
        - trigger-parameterized-builds:
          - project: archive
            current-parameters: true
            trigger-with-no-params: true
          - project: cleanup
            current-parameters: true
            trigger-with-no-params: false
```

`junit`

Publish JUnit test results.

Parameters

- **`results`** (*str*) – results filename
- **`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)
- **`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..

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
    test-stability: true
    claim-build: true
    measurement-plots: true
```

logparser

Requires the Jenkins Log Parser Plugin.

Parameters

- **parse-rules** (*str*) – full path to parse rules
- **unstable-on-warning** (*bool*) – mark build unstable on warning
- **fail-on-error** (*bool*) – mark build failed on error

Example:

```
publishers:
  - logparser:
      parse-rules: "/path/to/parserules"
      unstable-on-warning: true
      fail-on-error: true
```

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** (*num*) – 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
      deploy-unstable: false
      release-env-var: TIMER
```

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)
- **fixed-delay** (*int*) – Fixed delay 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 before retrying build increment (cannot be used when fixed-delay is being used) (Default 0)
- **progressive-delay-maximum** (*int*) – Progressive delay 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:
      rerun-unstable-builds: 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. (optional)

- **verbose** (*str*) – 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. (optional)
- **verbose** (*str*) – 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
```

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)
- **report** (*dict*) –

(jmeter or junit) (*dict* or *str*): Specify a custom report file (optional; jmeter default **/*.jtl, junit default* */TEST-*.xml)

Examples:

```
publishers:
  - performance:
      failed-threshold: 85
      unstable-threshold: -1
      report:
        - jmeter: "/special/file.jtl"
        - junit: "/special/file.xml"
```

```
publishers:
  - performance:
      failed-threshold: 85
      unstable-threshold: -1
      report:
        - jmeter
        - junit
```

```
publishers:
  - performance:
      failed-threshold: 85
      unstable-threshold: -1
      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.

Parameters

- **project** (*str*) – the name of the downstream project
- **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)

Example:

```
publishers:
  - pipeline:
      project: test_project
      current-parameters: true
      predefined-parameters: foo=bar
```

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
- **`group`** (*`str`*) – name of the group to which the plot belongs
- **`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

  Serie

  - **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.

Example:

```
publishers:
  - plot:
    - title: MyPlot
      yaxis: Y
      csv-file-name: myplot.csv
      group: PlotGroup
      num-builds: ''
      style: line
      exclude-zero-yaxis: true
      logarithmic-yaxis: true
      use-description: false
      series:
        - file: graph-me-second.properties
          label: MyLabel
          format: properties
        - file: graph-me-first.csv
          url: 'http://srv1'
          inclusion-flag: 'off'
          display-table: true
          format: csv
    - title: MyPlot2
      yaxis: Y
      csv-file-name: myplot2.csv
      group: PlotGroup
      style: line
      use-description: false
      series:
        - file: graph-me-third.xml
          url: 'http://srv2'
          format: xml
          xpath-type: 'node'
          xpath: '/*'
```

```
publishers:
  - plot:
      - title: 'Sample graph'
        yaxis: ''
        csv-file-name: 'persisted.csv'
        group: 'bench'
        numbuilds: '1'
        style: 'line'
        use-description: false
        series:
          - file: 'data.csv'
            format: 'csv'
            inclusion-flag: 'include-by-string'
            exclude: 'Column 1,Column 2,Column 3'
```

pmd

Publish trend reports with PMD. Requires the Jenkins 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-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
            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*) – Paths to Batch/Shell scripts
- **groovy-script** (*list*) – Paths to Groovy scripts
- **groovy** (*list*) – Inline Groovy
- **builders** (*list*) – Any supported builders, see Builders.
- **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)
- **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)
- **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*.

Example:

```
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"'
```

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
- **unstable-text** (*str*) – The unstable text if different from stable (default '')
- **failed-text** (*str*) – The failed text if different from stable (default '')
- **parser-name** (*str*) – HTML, Confluence or WikiText

Example:

```
publishers:
    - rich-text-publisher:
        stable-text: testing
        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. (default '')
- **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)

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
      other-files:
        - extra-file1.html
        - extra-file2.txt
      archive-output-xml: false
```

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
- **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
      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** (*string*) – Comma separated paths to exclude from reports (default: '')

Example:

```
publishers:
  - scan-build:
      mark-unstable: true
      threshold: 0
      exclude-paths: external-lib
```

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
- **report-file** (*str*) – This is a file name that is given to the xml scoverage report.

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
- **target** (*str*) – destination directory
- **source** (*str*) – source path specifier
- **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

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

As the Slack Plugin itself requires a publisher aswell as properties please note that you have to create those too.

Parameters

- **team-domain** (*str*) – Your team's domain at slack. (default: '')
- **auth-token** (*str*) – The integration token to be used when sending notifications. (default: '')
- **build-server-url** (*str*) – Specify the URL for your server installation. (default: '/')
- **room** (*str*) – A comma seperated list of rooms / channels to post the notifications to. (default: '')

Example:

```
publishers:
  - slack:
      room: '#builds'
      team-domain: 'teamname'
      auth-token: 'yourauthtoken'
```

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')

Example:

```
publishers:
  - sloccount:
      report-files: sloccount.sc
      charset: latin-1
```

sonar

Sonar plugin support. Requires the Jenkins Sonar Plugin.

Parameters

- **jdk** (*str*) – JDK to use (inherited from the job if omitted). (optional)
- **branch** (*str*) – branch onto which the analysis will be posted (optional)
- **language** (*str*) – source code language (optional)
- **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 (optional)
- **additional-properties** (*str*) – sonar analysis parameters (optional)
- **skip-global-triggers** (*dict*) –
  Triggers
  - **skip-when-scm-change** (*bool*): skip analysis when build triggered by scm
  - **skip-when-upstream-build** (*bool*): skip analysis when build triggered by an upstream build
  - **skip-when-envvar-defined** (*str*): skip analysis when the specified environment variable is set to true
- **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.

Example:

```
publishers:
  - sonar:
      jdk: MyJdk
      branch: myBranch
      language: java
      maven-opts: -DskipTests
      additional-properties: -DsonarHostURL=http://example.com/
      skip-global-triggers:
        skip-when-scm-change: true
        skip-when-upstream-build: true
        skip-when-envvar-defined: SKIP_SONAR
```

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).

Example:

```
publishers:
  - ssh:
      site: 'server.example.com'
      target: 'dest/dir'
      source: 'base/source/dir/**'
      remove-prefix: 'base/source/dir'
      excludes: '**/*.excludedfiletype'
      use-pty: true
      command: 'rm -r jenkins_$BUILD_NUMBER'
      timeout: 1800000
      flatten: true
```

stash

This plugin will configure the Jenkins Stash Notifier plugin to notify Atlassian Stash after job completes.

Requires the Jenkins StashNotifier Plugin.

Parameters

- **url** (*string*) – Base url of Stash Server (Default: "")
- **username** (*string*) – Username of Stash Server (Default: "")
- **password** (*string*) – Password of Stash Server (Default: "")
- **credentials-id** (*string*) – Credentials of Stash Server (optional)
- **ignore-ssl** (*bool*) – Ignore unverified SSL certificate (Default: False)
- **commit-sha1** (*string*) – Commit SHA1 to notify (Default: "")
- **include-build-number** (*bool*) – Include build number in key (Default: False)

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
- **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)

Example:

```
publishers:
  - tap:
      results: puiparts.tap
      todo-is-failure: false
```

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
- **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)

Example:

```
publishers:
    - testng:
        pattern: "**/target/surefire-reports/testng-results.xml"
        escape-test-description: false
        escape-exception-msg: true
```

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*) – Amount of times the test should run (default "")

Example:

```
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
- **fileset** (*str*) – Specify the path to search
- **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)

Example:

```
publishers:
  - text-finder:
      regexp: "some string"
      fileset: "file.txt"
      also-check-console-output: true
      succeed-if-found: false
      unstable-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:

```
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** (*bool*) – Pass git revision to the other job (optional)

- **combine-queued-commits** (*bool*) – Combine Queued git hashes. Only valid when 'git-revision' is true. (default false)
- **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')
- **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:
        - other_job
        - foo
        - bar
      predefined-parameters: |
        foo=bar
        bar=foo
    - project: other_job1, other_job2
      predefined-parameters: BUILD_NUM=${BUILD_NUMBER}
      property-file: version.prop
      fail-on-missing: true
    - project: yet_another_job
      predefined-parameters: foo=bar
      git-revision: 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, perl-critic, pmd, pylint, simian, stylecop

Example:

```
publishers:
  - violations:
      pep8:
        min: 0
        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.

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
  - **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 '')
- **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 '')

Example:

```
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'
      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
      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')

Example:

```
publishers:
  - whitesource:
      product-token: abcdefghijklmnopqrstuvwxyzabcdef
      version: 1.0.17
      policies: enable
      override-token: "1231424523412"
      project-token: sd;fkljsdfkljasdfkj
      includes:
        - lib/*.jar
        - test/lib/*.jar
      excludes:
        - lib/ant*.jar
        - test/lib/ant*.jar
```

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)

Example:

```
publishers:
  - workspace-cleanup:
      include:
        - "*.zip"
      clean-if:
        - success: true
        - not-built: false
```

xml-summary

Adds support for the Summary Display Plugin

Requires the Jenkins Summary Display Plugin.

Parameters

- **files** (*str*) – Files to parse (default '')
- **shown-on-project-page** (*bool*) – Display summary on project page (default 'false')

Example:

```
publishers:
  - xml-summary:
      files: '*_summary_report.xml'
```

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`, `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.
  - **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"
          - 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 projets.

Component: reporters

Macro reporter

Entry Point jenkins_jobs.reporters

Example:

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

  reporters:
    - email:
        recipients: breakage@example.com
```

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)

Example:

```
    reporters:
      - email:
          recipients: breakage@example.com
```

findbugs

FindBugs reporting for builds

Requires the Jenkins 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
        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: []
```

`bzr`

Specifies the bzr SCM repository for this job. Requires the Jenkins Bazaar Plugin.

Parameters

- **`url`** (*`str`*) – URL of the bzr branch
- **`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

  - **remote** – The name of the module in the repository at CVSROOT. (required)
  - **local-name** – 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"
            locations:
                - type: TAG
                  name: "tag name"
                  use-head: false
                  modules:
                    - remote: "remote1"
                      local-name: "localName"
                    - remote: "remote2"
            excluded-regions:
                - "pattern1"
                - "pattern2"
            compression-level: "1"
          - root: ":protocol:user@host2:path"
            locations:
                - modules:
                    - remote: "remote1"
      use-update: false
      prune-empty: false
      skip-changelog: true
      show-all-output: true
      clean-checkout: true
      clean-copy: true
```

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 '**')
- **excluded-users** (*list (str)*) – list of users to ignore revisions from when polling for changes. (if polling is enabled, optional)
- **included-regions** (*list (str)*) – list of file/folders to include (optional)
- **excluded-regions** (*list (str)*) – list of file/folders to exclude (optional)
- **local-branch** (*str*) – Checkout/merge to local branch (optional)
- **merge** (*dict*) –

  merge

  - **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')
- **basedir** (*str*) – location relative to the workspace root to clone to (default workspace)
- **skip-tag** (*bool*) – Skip tagging (default false)
- **shallow-clone** (*bool*) – Perform shallow clone (default false)
- **prune** (*bool*) – Prune remote branches (default false)
- **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.
- **use-author** (*bool*) – Use author rather than committer in Jenkin's build changeset (default false)
- **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)
- **scm-name** (*str*) – The unique scm name for this Git SCM (optional)
- **ignore-notify** (*bool*) – Ignore notifyCommit URL accesses (default false)
- **browser** (*str*) – what repository browser to use.

  browsers supported

  - **auto** - (default)
  - **assemblaweb** - https://www.assembla.com/
  - **bitbucketweb** - https://www.bitbucket.org/
  - **cgit** - http://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/p/gitiles/
  - **gitlab** - https://about.gitlab.com/
  - **gitlist** - http://gitlist.org/
  - **gitoriousweb** - https://gitorious.org/
  - **gitweb** - http://git-scm.com/docs/gitweb
  - **kiln** - https://www.fogcreek.com/kiln/
  - **microsoft-tfs-2013** - https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx

  - **phabricator** - http://phabricator.org/
  - **redmineweb** - http://www.redmine.org/
  - **rhodecode** - https://rhodecode.com/
  - **stash** - https://www.atlassian.com/software/stash
  - **viewgit** - http://viewgit.fealdia.org/
- **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)
- **choosing-strategy** (*str*) – Jenkins class for selecting what to build. Can be one of *default*, *inverse*, or *gerrit* (default 'default')
- **git-config-name** (*str*) – Configure name for Git clone (optional)
- **git-config-email** (*str*) – Configure email for Git clone (optional)

Extensions

- **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')
- **clean (*dict*)**
  - **after** (*bool*) - Clean the workspace after checkout
  - **before** (*bool*) - Clean the workspace before checkout
- **ignore-commits-with-messages** (*list(str)*) - Revisions committed with messages matching these patterns will be ignored. (optional)
- **force-polling-using-workspace** (*bool*) - Force polling using workspace (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)
  - **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)
- **wipe-workspace (*bool*) - Wipe out workspace before build** (default true)

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
- **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://www.bitbucket.org/
  - **fisheye** - https://www.atlassian.com/software/fisheye
  - **googlecode** - https://code.google.com/
  - **hgweb** - https://www.selenic.com/hg/help/hgweb
  - **kilnhg** - https://www.fogcreek.com/kiln/
  - **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
     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. (optional)
- **verbose** (*str*) – This flag is the toggle for turning on or off detailed logging in this plug-in. (optional)

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
```

repo

Specifies the repo SCM repository for this job. Requires the Jenkins Repo Plugin.

Parameters

- **manifest-url** (*str*) – URL of the repo manifest
- **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)
- **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
      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>
          <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)
- **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
              - 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.

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:
```

`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'
```

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 '')

Example:

```
    triggers:
      - build-result:
          combine: true
          cron: '* * * * *'
          groups:
            - jobs:
                - foo
                - example
              results:
                - unstable
            - jobs:
                - foo2
              results:
                - not-built
                - aborted
```

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 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.
    - **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-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
- **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 '')
- **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)
      File Path
        * **compare-type** (*str*) – ''PLAIN'', ''ANT'' or ''REG_EXP'' (optional) (default ''PLAIN'')
        * **pattern** (*str*) – Topic name pattern to match

- **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*)
- **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)
- **no-name-and-email** (*bool*) – Do not pass compound 'name and email' parameters (default false)
- **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)
- **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 con-

figuration 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
- **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:
      trigger-on:
        - patchset-created-event:
            exclude-drafts: true
            exclude-trivial-rebase: true
            exclude-no-code-change: true
        - 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
      silent: false
      silent-start: true
      escape-quotes: false
      no-name-and-email: false
      dependency-jobs: 'job1, job2'
      notification-level: ALL
      dynamic-trigger-enabled: true
      dynamic-trigger-url: http://myhost/mytrigger
      trigger-for-unreviewed-patches: true
```

```
        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** (*string*) – cron syntax of when to run (optional)
- **trigger-phrase** (*string*) – 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)
- **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)
- **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)
- **auth-id** (*string*) – the auth id to use (optional)
- **build-desc-template** (*string*) – the template for build descriptions in jenkins (optional)
- **status-context** (*string*) – the context to include on PR status comments (optional)
- **triggered-status** (*string*) – the status message to set when the build has been triggered (optional)
- **started-status** (*string*) – the status comment to set when the build has been started (optional)
- **status-url** (*string*) – the status URL to set (optional)
- **success-status** (*string*) – the status message to set if the job succeeds (optional)
- **failure-status** (*string*) – the status message to set if the job fails (optional)
- **error-status** (*string*) – the status message to set if the job errors (optional)
- **success-comment** (*string*) – comment to add to the PR on a successful job (optional)
- **failure-comment** (*string*) – comment to add to the PR on a failed job (optional)
- **error-comment** (*string*) – comment to add to the PR on an errored job (optional)

Example:

```
triggers:
  - github-pull-request:
      admin-list:
        - user1
        - user2
      white-list:
        - user3
        - user4
      org-list:
        - org1
        - org2
      cron: '* * * * *'
      build-desc-template: "build description"
      trigger-phrase: 'retest this please'
      only-trigger-phrase: true
      github-hooks: true
      permit-all: false
      auto-close-on-fail: false
      allow-whitelist-orgs-as-admins: true
      white-list-target-branches:
        - master
        - testing
      auth-id: '123-456-789'
```

gitlab

Makes Jenkins act like a GitlabCI 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-open-merge-request-push** (*bool*) – Rebuild open Merge Requests on Push Events (default: True)
- **ci-skip** (*bool*) – Enable [ci-skip] (default True)
- **set-build-description** (*bool*) – Set build description to build cause (eg. Merge request or Git Push) (default: True)
- **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 (default: True)
- **add-ci-message** (*bool*) – Add CI build status (default: False)
- **allow-all-branches** (*bool*) – Allow all branches (Ignoring Filtered Branches) (default: False)
- **include-branches** (*list*) – Defined list of branches to include (default: [])
- **exclude-branches** (*list*) – Defined list of branches to exclude (default: [])

Example:

```
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
      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'
```

gitlab-merge-request

Build merge requests in gitlab and report results. Requires the Jenkins Gitlab MergeRequest Builder Plugin..

Parameters

- **cron** (*string*) – cron syntax of when to run (required)
- **project-path** (*string*) – gitlab-relative path to project (required)

Example:

```
triggers:
  - gitlab-merge-request:
      cron: '* * * * *'
      project-path: 'test/project'
```

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. (optional)
- **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:
  - groovy-script:
        script-file-path: 'path/to/filename'
        cron: 'H/15 * * * *'
        enable-concurrent: true
        label: master
        system-script: true
```

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)
- **str 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
        label: master
```

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 dont 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 '')

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'
                    - '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. (optional)
- **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. (optional)
- **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. (optional)
- **check-modification-date** (*bool*) – Check last modification date. (default true)
- **check-content** (*bool*) – Check content. (default true)
- **check-fewer** (*bool*) – Check fewer or more files (default true)
- **cron** (*str*) – cron syntax of when to run (default '')

Example:

```
triggers:
  - monitor-folders:
      path: 'pathname'
      includes:
          - 'pattern1'
          - 'pattern2'
      excludes: 'pattern1'
      check-modification-date: false
      check-content: false
      check-fewer: false
      cron: 'H/15 * * * *'
```

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** (*string*) – 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** (*string*) – cron syntax of when to run (default '')
- **polling-node** (*string*) – 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** (*string*) – User name for basic authentication (optional)
  - **password** (*string*) – 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
          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+'
```

`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. (optional)
- **`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)

Example:

```
triggers:
  - script:
        script: 'exit 0'
        cron: 'H/15 * * * *'
        enable-concurrent: False
        label: master
        exit-code: 0
```

`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

`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** (*string*) – (optional) color mapping to use

Examples:

```
wrappers:
  - ansicolor

# Explicitly setting the colormap
wrappers:
  - ansicolor:
      colormap: vga
```

artifactory-generic

Wrapper for non-Maven projects. Requires the Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. http://www.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 (default '')
- **snapshot-repo-key** (*str*) – Snapshots repository name (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 Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. http://www.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 Artifactory plugin. Requires Artifactory Plugin

Parameters

- **url** (*str*) – URL of the Artifactory server. e.g. http://www.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 artifacts 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-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.

Example:

```
wrappers:
  - build-name:
      name: Build-${FOO}
```

`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 (optional)
- **variable** (*str*) – Define an environment variable to be used (optional)

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
```

`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* or *username-password-separated*. 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.

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)

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
 - credentials-binding:
     - username-password-separated:
        credential-id: b3e6f337-5d44-4f57-921c-1632d796caae
        username: myUsername
        password: myPassword
```

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:
      - 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)

Example:

```
wrappers:
    - delivery-pipeline:
        version-template: 1.0.0-${BUILD_NUMBER}
        set-display-name: true
```

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 (default '')

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
```

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 (default '')
- **properties-content** (*str*) – key value pair of properties (default '')
- **script-file** (*str*) – path to the script file (default '')
- **script-content** (*str*) – contents of a script (default '')

Example:

```
wrappers:
  - inject:
      properties-file: /usr/local/foo
      properties-content: PATH=/foo/bar
      script-file: /usr/local/foo.sh
      script-content: echo $PATH
```

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:
      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 passsword 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.

Example using defaults:

```
wrappers:
    - live-screenshot
```

or specifying the files to use:

```
wrappers:
    - live-screenshot:
        full-size: my_screenshot.png
        thumbnail: my_screenshot-thumb.png
```

`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.

Minimum config example:

```
wrappers:
  - logfilesize
```

Full config example:

```
wrappers:
  - logfilesize:
      set-own: true
      size: 1024
      fail: true
```

`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.

Example:

```
project-type: matrix
wrappers:
  - matrix-tie-parent:
      node: Unix
```

`mongo-db build wrapper`

Initalizes a MongoDB database while running the build. Requires the Jenkins MongoDB plugin.

Parameters

- **name** (*str*) – The name of the MongoDB install to use
- **data-directory** (*str*) – Data directory for the server (optional)
- **port** (*int*) – Port for the server (optional)
- **startup-params** (*str*) – Startup parameters for the server (optional)
- **start-timeout** (*int*) – How long to wait for the server to start in milliseconds. 0 means no timeout. (default '0')

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
```

nodejs-installator

Requires the Jenkins NodeJS Plugin.

Parameters **name** (*str*) – nodejs installation name

Example:

```
wrappers:
  - nodejs-installator:
      name: "latest node"
```

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 **buildsteps** (*list*) – List of build steps to execute

Buildstep Any acceptable builder, as seen in the example

Example:

```
    wrappers:
      - pre-scm-buildstep:
        - shell: |
            #!/bin/bash
            echo "Doing somethiung cool"
        - shell: |
            #!/bin/zsh
            echo "Doing somethin 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/sstephenson/rbenv.git)
- **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/sstephenson/ruby-build.git)
- **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** (*string*) – 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 conenct 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
```

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)

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 prefered, **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'
```

The **users** with one value in list equals to the **user**. In this case old style XML will be generated. Use this format if you use SSH-Agent plugin < 1.5.

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)
- **write-description** (*bool*) – Write a message in the description (default false)
- **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)

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
```

timestamps

Add timestamps to the console log. Requires the Jenkins Timestamper Plugin.

Example:

```
wrappers:
  - timestamps
```

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)

Example:

```
wrappers:
  - workspace-cleanup:
      include:
        - "*.zip"
```

xvfb

Enable xvfb during the build. Requires the Jenkins Xvfb Plugin.

Parameters

- **installation-name** (*str*) – The name of the Xvfb tool instalation (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 choosen based on current build executor number (optional)
- **assigned-labels** (*str*) – If you want to start Xvfb only on specific nodes specify its name or label (optional)
- **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** (*str*) – Offset for display names. (default: 1)
- **additional-options** (*str*) – Additional options to be added with the options above to the Xvfb command line (optional)
- **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)

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
```

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)

Example:

```
wrappers:
  - xvnc:
      screenshot: true
      xauthority: false
```

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.

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
```

2.7.4 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

2.8 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.

2.8.1 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.

2.8.2 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` (*parser*, *xml_parent*, *data*)

Parameters

- **`parser`** (*`YAMLParser`*) – the jenkins jobs YAML parser
- **`xml_parent`** (*`Element`*) – this attribute's parent XML element
- **`data`** (*`dict`*) – 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.

2.8.3 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.

`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` (*parser*, *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.

Parameters

- **`parser`** (*`YAMLParser`*) – the global YAML Parser
- **`xml_parent`** (*`Element`*) – the parent XML element
- **`data`** (*`dict`*) – the YAML data structure

`handle_data` (*parser*)

This method is called before any XML is generated. By overriding this method, the module may manipulate the YAML data structure on the parser however it likes before any XML is generated. If it has changed the data structure at all, it must return `True`, otherwise, it must return `False`.

Parameters **`parser`** (*`YAMLParser`*) – the global YAML Parser

Return type boolean

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.

2.8.4 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).

2.8.5 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`**(*config*, *plugins_list=None*)

`dispatch`(*component_type*, *parser*, *xml_parent*, *component*, *template_data={}*)
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`** (*`string`*) – the name of the component (e.g., *builder*)
- **`parser`** (*`YAMLParser`*) – the global YAML Parser
- **`xml_parent`** (*`Element`*) – the parent XML element
- **`template_data`** (*`dict`*) – values that should be interpolated into the component 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*)
This method is intended to provide information about plugins within a given module's implementation 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`** (*`string`*) – 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 3

Indices and tables

- genindex
- modindex
- search

Index