AutoJenkins Documentation

Release 0.7.0

Carles Barrobés

May 11, 2012

CONTENTS

AutoJenkins is a tool to automate or remote-control Jenkins. You can e.g. create and delete build jobs, trigger builds, read latest build results, etc.

Contents:

CHAPTER

ONE

INTRODUCTION TO AUTOJENKINS

AutoJenkins was written to handle automation (remote control) of Jenkins tasks. Includes a class `autojenkins.Jenkins` that you can use to drive Jenkins.

Things you can do with it:

- Copy a job (e.g. from a template job)
- Delete a job
- Obtain the `config.xml` file for that job
- Trigger building of a job
- Obtain latest execution results
- ...

AutoJenkins may be used as an API or as a command-line tool.

1.1 AutoJenkins as API

Sample use:

```
from autojenkins import Jenkins

j = Jenkins('http://jenkins.pe.local')

# trigger a manual build and check results
j.build('warehouse-screens-us544_login')
j.last_result('warehouse-screens-us544_login')

# get only the result string (one of 'SUCCESS', 'UNSTABLE', 'FAILURE'):
j.last_result('warehouse-screens-us544_login')['result']

# get the configuration file for a job:
j.get_config_xml('template')

# Create a new job from a job named 'template', replacing variables
j.create_copy('my-new-job', 'template',
              repo='mbf-warehouse-screens',
              branch='us544_login',
              package='warehouse_screens')

# build
```

```
j.build('my-new-job')

# check result and delete if successful:
result = j.last_result('my-new-job')['result']
if result == 'SUCCESS':
    j.delete('my-new-job')
```

1.2 AutoJenkins from the Command Line

Available commands:

- `ajk-list` - list all jobs in a server
- `ajk-create` - create a job
- `ajk-build` - start building a job
- `ajk-delete` - delete a job

1.2.1 `ajk-list`

List all jobs in a Jenkins server. Each line in the output represents a job, and is colored according to the job's last build state:

- Blue: success
- Yellow: unstable
- Red: failure
- Gray: not built

A `*` symbol next to a job name indicates that the job is being built right now.

If instead of colored output, you prefer a string stating the status of the build, use the `--no-color` option. This is useful if you e.g. want to pipe the output into a `grep` command that filters jobs depending on status.

```
$ ajk-list -h

Usage: ajk-list host

Run autojenkins to list all jobs.

Options:
  -h, --help      show this help message and exit
  -n, --no-color  do not use colored output
```

1.2.2 `ajk-create`

Create a job from a template job, replacing variables that use the django/jinja2 syntax `{{ variable }}`.

Usage help:

```
$ ajk-create -h

Usage: ajk-create host jobname [options]
```

```
Run autojenkins to create a job.

Options:
  -h, --help            show this help message and exit
  -t TEMPLATE, --template=TEMPLATE
                        the template job to copy from
  -D PROP=VALUE         substitution variables for the template
  -b, --build           start a build right after creation
```

Sample command:

```
$ ajk-create http://my.server my-job -t template -Dbranch=my-branch
```

1.2.3 `ajk-delete`

Delete a job from a Jenkins server.

Usage help:

```
Usage: ajk-delete host [jobname] [options]

Run autojenkins to delete a job.

Options:
  -h, --help  show this help message and exit
```

1.3 More Info

Sources can be found in Github at https://github.com/txels/autojenkins

CHAPTER

TWO

AUTOJENKINS.JOBS

class `autojenkins.jobs.`**`Jenkins`**(*base_url*, *auth=None*)

Main class to interact with a Jenkins server.

`all_jobs`()

Get a list of tuples with (name, color) of all jobs in the server.

Color is `blue`, `yellow` or `red` depending on build results (SUCCESS, UNSTABLE or FAILED).

`build`(*jobname*, *wait=False*, *grace=10*)

Trigger Jenkins to build a job.

Parameters **wait** – If `True`, wait until job completes building before returning

`copy`(*jobname*, *copy_from='template'*)

Copy a job from another one (by default from one called `template`).

`create`(*jobname*, *config_file*, ***context*)

Create a job from a configuration file.

`create_copy`(*jobname*, *template_job*, *enable=True*, ***context*)

Create a job from a template job.

`delete`(*jobname*)

Delete a job.

`disable`(*jobname*)

Trigger Jenkins to disable a job.

`enable`(*jobname*)

Trigger Jenkins to enable a job.

`get_config_xml`(*jobname*)

Get the `config.xml` file that contains the job definition.

`is_building`(*jobname*)

Check if a job is building

`job_info`(*jobname*)

Get all information for a job as a Python object (dicts & lists).

`job_url`(*jobname*)

Get the human-browseable URL for a job.

`last_build_info`(*jobname*)

Get information for last build of a job.

`last_build_report`(*jobname*)

Get full report of last build.

`last_result`(*jobname*)
Obtain results from last execution.

`last_success`(*jobname*)
Return information about the last successful build.

`set_config_xml`(*jobname*, *config*)
Replace the `config.xml` of an existing job.

`transfer`(*jobname*, *to_server*)
Copy a job to another server.

`wait_for_build`(*jobname*, *poll_interval=3*)
Wait until job has finished building

Sources can be found at https://github.com/txels/autojenkins

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX