Scrapyd Documentation

Release 0.18

Scrapy group

October 07, 2015

Contents

Scrapyd is an application for deploying and running Scrapy spiders. It enables you to deploy (upload) your projects and control their spiders using a JSON API.

CHAPTER 1

Contents

1.1 Overview

1.1.1 Projects and versions

Scrapyd can manage multiple projects and each project can have multiple versions uploaded, but only the latest one will be used for launching new spiders.

A common (and useful) convention to use for the version name is the revision number of the version control tool you're using to track your Scrapy project code. For example: `r23`. The versions are not compared alphabetically but using a smarter algorithm (the same distutils uses) so `r10` compares greater to `r9`, for example.

1.1.2 How Scrapyd works

Scrapyd is an application (typically run as a daemon) that listens to requests for spiders to run and spawns a process for each one, which basically executes:

```
scrapy crawl myspider
```

Scrapyd also runs multiple processes in parallel, allocating them in a fixed number of slots given by the *max_proc* and *max_proc_per_cpu* options, starting as many processes as possible to handle the load.

In addition to dispatching and managing processes, Scrapyd provides a *JSON web service* to upload new project versions (as eggs) and schedule spiders. This feature is optional and can be disabled if you want to implement your own custom Scrapyd. The components are pluggable and can be changed, if you're familiar with the Twisted Application Framework which Scrapyd is implemented in.

Starting from 0.11, Scrapyd also provides a minimal *web interface*.

1.1.3 Starting Scrapyd

To start the service, use the `scrapyd` command provided in the Scrapy distribution:

```
scrapyd
```

That should get your Scrapyd started.

1.1.4 Scheduling a spider run

To schedule a spider run:

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=spider2
{"status": "ok", "jobid": "26d1b1a6d6f111e0be5c001e648c57f8"}
```

For more resources see: *API* for more available resources.

1.1.5 Web Interface

Scrapyd comes with a minimal web interface (for monitoring running processes and accessing logs) which can be accessed at http://localhost:6800/

1.2 Installation

This documents explains how to install and configure Scrapyd, to deploy and run your Scrapy spiders.

1.2.1 Requirements

Scrapyd depends on the following libraries, but the installation process takes care of installing the missing ones:

- Python 2.6 or above
- Twisted 8.0 or above
- Scrapy 0.17 or above

1.2.2 Installing Scrapyd (generic way)

How to install Scrapyd depends on the platform you're using. The generic way is to install it from PyPI:

```
pip install scrapyd
```

If you plan to deploy Scrapyd in Ubuntu, Scrapyd comes with official Ubuntu packages (see below) for installing it as a system service, which eases the administration work.

Other distributions and operating systems (Windows, Mac OS X) don't yet have specific packages and require to use the generic installation mechanism in addition to configuring paths and enabling it run as a system service. You are very welcome to contribute Scrapyd packages for your platform of choice, just send a pull request on Github.

1.2.3 Installing Scrapyd in Ubuntu

Scrapyd comes with official Ubuntu packages ready to use in your Ubuntu servers. They are shipped in the same APT repos of Scrapy, which can be added as described in Scrapy Ubuntu packages. Once you have added the Scrapy APT repos, you can install Scrapyd with `apt-get`:

```
apt-get install scrapyd
```

This will install Scrapyd in your Ubuntu server creating a `scrapy` user which Scrapyd will run as. It will also create the directories and files described below:

/etc/scrapyd

Scrapyd configuration files. See *Configuration file*.

/var/log/scrapyd/scrapyd.log

Scrapyd main log file.

/var/log/scrapyd/scrapyd.out

The standard output captured from Scrapyd process and any sub-process spawned from it.

/var/log/scrapyd/scrapyd.err

The standard error captured from Scrapyd and any sub-process spawned from it. Remember to check this file if you're having problems, as the errors may not get logged to the `scrapyd.log` file.

/var/log/scrapyd/project

Besides the main service log file, Scrapyd stores one log file per crawling process in:

```
/var/log/scrapyd/PROJECT/SPIDER/ID.log
```

Where `ID` is a unique id for the run.

/var/lib/scrapyd/

Directory used to store data files (uploaded eggs and spider queues).

1.3 Deploying your project

Deploying your project involves eggifying it and uploading the egg to Scrapyd via the addversion.json endpoint. You can do this manually, but the easiest way is to use the *scrapyd-deploy* tool provided by scrapyd-client which will do it all for you.

1.4 API

The following section describes the available resources in Scrapyd JSON API.

1.4.1 daemonstatus.json

To check the load status of a service.

- Supported Request Methods: `GET`

Example request:

```
curl http://localhost:6800/daemonstatus.json
```

Example response:

```
{ "status": "ok", "running": "0", "pending": "0", "finished": "0", "node_name": "node-name" }
```

1.4.2 addversion.json

Add a version to a project, creating the project if it doesn't exist.

- Supported Request Methods: `POST`
- Parameters:
  - `project` (string, required) - the project name
  - `version` (string, required) - the project version
  - `egg` (file, required) - a Python egg containing the project's code

Example request:

```
$ curl http://localhost:6800/addversion.json -F project=myproject -F version=r23 -F egg=@myproject.eg
```

Example response:

```
{"status": "ok", "spiders": 3}
```

1.4.3 schedule.json

Schedule a spider run (also known as a job), returning the job id.

- Supported Request Methods: `POST`
- Parameters:
  - `project` (string, required) - the project name
  - `spider` (string, required) - the spider name
  - `setting` (string, optional) - a scrapy setting to use when running the spider
  - `jobid` (string, optional) - a job id used to identify the job, overrides the default generated UUID
  - any other parameter is passed as spider argument

Example request:

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider
```

Example response:

```
{"status": "ok", "jobid": "6487ec79947edab326d6db28a2d86511e8247444"}
```

Example request passing a spider argument (`arg1`) and a setting (DOWNLOAD_DELAY):

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider -d setting=DOWNL
```

Note: Spiders scheduled with scrapyd should allow for an arbitrary number of keyword arguments as scrapyd sends internally generated spider arguments to the spider being scheduled

1.4.4 cancel.json

New in version 0.15.

Cancel a spider run (aka. job). If the job is pending, it will be removed. If the job is running, it will be terminated.

- Supported Request Methods: `POST`
- Parameters:
  - `project` (string, required) - the project name
  - `job` (string, required) - the job id

Example request:

```
$ curl http://localhost:6800/cancel.json -d project=myproject -d job=6487ec79947edab326d6db28a2d86511
```

Example response:

```
{"status": "ok", "prevstate": "running"}
```

1.4.5 listprojects.json

Get the list of projects uploaded to this Scrapy server.

- Supported Request Methods: `GET`
- Parameters: none

Example request:

```
$ curl http://localhost:6800/listprojects.json
```

Example response:

```
{"status": "ok", "projects": ["myproject", "otherproject"]}
```

1.4.6 listversions.json

Get the list of versions available for some project. The versions are returned in order, the last one is the currently used version.

- Supported Request Methods: `GET`
- Parameters:
  - `project` (string, required) - the project name

Example request:

```
$ curl http://localhost:6800/listversions.json?project=myproject
```

Example response:

```
{"status": "ok", "versions": ["r99", "r156"]}
```

1.4.7 listspiders.json

Get the list of spiders available in the last version of some project.

- Supported Request Methods: `GET`
- Parameters:
  - `project` (string, required) - the project name

Example request:

```
$ curl http://localhost:6800/listspiders.json?project=myproject
```

Example response:

```
{"status": "ok", "spiders": ["spider1", "spider2", "spider3"]}
```

1.4.8 listjobs.json

New in version 0.15.

Get the list of pending, running and finished jobs of some project.

- Supported Request Methods: `GET`
- Parameters:
  - `project` (string, required) - the project name

Example request:

```
$ curl http://localhost:6800/listjobs.json?project=myproject
```

Example response:

```
{"status": "ok",
 "pending": [{"id": "78391cc0fcaf11e1b0090800272a6d06", "spider": "spider1"}],
 "running": [{"id": "422e608f9f28cef127b3d5ef93fe9399", "spider": "spider2", "start_time": "2012-09-1
 "finished": [{"id": "2f16646cfcaf11e1b0090800272a6d06", "spider": "spider3", "start_time": "2012-09-
```

Note: All job data is kept in memory and will be reset when the Scrapyd service is restarted. See issue 12.

1.4.9 delversion.json

Delete a project version. If there are no more versions available for a given project, that project will be deleted too.

- Supported Request Methods: `POST`
- Parameters:
  - `project` (string, required) - the project name
  - `version` (string, required) - the project version

Example request:

```
$ curl http://localhost:6800/delversion.json -d project=myproject -d version=r99
```

Example response:

```
{"status": "ok"}
```

1.4.10 delproject.json

Delete a project and all its uploaded versions.

- Supported Request Methods: `POST`
- Parameters:
  - `project` (string, required) - the project name

Example request:

```
$ curl http://localhost:6800/delproject.json -d project=myproject
```

Example response:

```
{"status": "ok"}
```

1.5 Configuration file

Scrapyd searches for configuration files in the following locations, and parses them in order with the latest one taking more priority:

- `/etc/scrapyd/scrapyd.conf` (Unix)
- `c:\scrapyd\scrapyd.conf` (Windows)
- `/etc/scrapyd/conf.d/*` (in alphabetical order, Unix)
- `scrapyd.conf`
- `~/.scrapyd.conf` (users home directory)

The configuration file supports the following options (see default values in the *example*).

1.5.1 http_port

The TCP port where the HTTP JSON API will listen. Defaults to `6800`.

1.5.2 bind_address

The IP address where the HTTP JSON API will listen. Defaults to `0.0.0.0` (all)

1.5.3 max_proc

The maximum number of concurrent Scrapy process that will be started. If unset or `0` it will use the number of cpus available in the system multiplied by the value in `max_proc_per_cpu` option. Defaults to `0`.

1.5.4 max_proc_per_cpu

The maximum number of concurrent Scrapy process that will be started per cpu. Defaults to `4`.

1.5.5 debug

Whether debug mode is enabled. Defaults to `off`. When debug mode is enabled the full Python traceback will be returned (as plain text responses) when there is an error processing a JSON API call.

1.5.6 eggs_dir

The directory where the project eggs will be stored.

1.5.7 dbs_dir

The directory where the project databases will be stored (this includes the spider queues).

1.5.8 logs_dir

The directory where the Scrapy logs will be stored. If you want to disable storing logs set this option empty, like this:

```
logs_dir =
```

1.5.9 items_dir

New in version 0.15.

The directory where the Scrapy items will be stored. If you want to disable storing feeds of scraped items (perhaps, because you use a database or other storage) set this option empty, like this:

```
items_dir =
```

1.5.10 jobs_to_keep

New in version 0.15.

The number of finished jobs to keep per spider. Defaults to `5`. This refers to logs and items.

This setting was named `logs_to_keep` in previous versions.

1.5.11 finished_to_keep

New in version 0.14.

The number of finished processes to keep in the launcher. Defaults to `100`. This only reflects on the website /jobs endpoint and relevant json webservices.

1.5.12 poll_interval

The interval used to poll queues, in seconds. Defaults to `5`.

1.5.13 runner

The module that will be used for launching sub-processes. You can customize the Scrapy processes launched from Scrapyd by using your own module.

1.5.14 application

A function that returns the (Twisted) Application object to use. This can be used if you want to extend Scrapyd by adding and removing your own components and services.

For more info see Twisted Application Framework

1.5.15 node_name

New in version 1.1.

The node name for each node to something like the display hostname. Defaults to `${socket.gethostname()}`.

1.5.16 Example configuration file

Here is an example configuration file with all the defaults:

```
[scrapyd]
eggs_dir    = eggs
logs_dir    = logs
items_dir   = items
jobs_to_keep = 5
dbs_dir     = dbs
max_proc    = 0
max_proc_per_cpu = 4
finished_to_keep = 100
poll_interval = 5
http_port   = 6800
debug       = off
runner      = scrapyd.runner
application = scrapyd.app.application
launcher    = scrapyd.launcher.Launcher

[services]
schedule.json     = scrapyd.webservice.Schedule
cancel.json       = scrapyd.webservice.Cancel
addversion.json   = scrapyd.webservice.AddVersion
listprojects.json = scrapyd.webservice.ListProjects
listversions.json = scrapyd.webservice.ListVersions
listspiders.json  = scrapyd.webservice.ListSpiders
delproject.json   = scrapyd.webservice.DeleteProject
delversion.json   = scrapyd.webservice.DeleteVersion
listjobs.json     = scrapyd.webservice.ListJobs
daemonstatus.json = scrapyd.webservice.DaemonStatus
```

1.6 Contributing

Important: Read through the Scrapy Contribution Docs for tips relating to writing patches, reporting bugs, and

project coding style

These docs describe how to setup and contribute to Scrapyd.

1.6.1 Reporting Issues & Bugs

Issues should be reported to the Scrapyd project issue tracker on GitHub

1.6.2 Tests

Tests are implemented using the Twisted unit-testing framework. Scrapyd uses `trial` as the test running application.

1.6.3 Running tests

To run all tests go to the root directory of the Scrapyd source code and run:

```
trial scrapyd
```

To run a specific test (say `tests/test_poller.py`) use:

```
trial scrapyd.tests.test_poller
```

1.6.4 Writing tests

All functionality (including new features and bug fixes) should include a test case to check that it works as expected, so please include tests for your patches if you want them to get accepted sooner.

Scrapyd uses unit-tests, which are located in the scrapyd/tests directory. Their module name typically resembles the full path of the module they're testing. For example, the scheduler code is in:

```
scrapyd.scheduler
```

And their unit-tests are in:

```
scrapyd/tests/test_scheduler.py
```

1.6.5 Installing Locally

To install a locally edited version of Scrapyd onto the system to use and test, inside the project root run:

```
pip install -e .
```

1.7 Release notes

1.7.1 1.0

First standalone release (it was previously shipped with Scrapy until Scrapy 0.16).