Speedrack Documentation

Release 0.6.1

Clint Howarth

January 02, 2014

Contents

Speedrack is a single-node cron system with an attractive web interface, job-tracking and configurable notification via email. It's not the task runner you need, but it might be the one you deserve.

It is intended to drive regular executions of low-resource shell scripts, routing output to defined locations. The web front end makes it easy to see a high-level overview. It's convenient and easy.

Contents:

CHAPTER 1

Installation and Demo Mode

Speedrack requires *Python 2.6+*. Python comes with easy_install.

(If you don't have `pip`, prepend the following instructions with: `easy_install pip`)

```
pip install speedrack
speedrack run
```

Open a browser at `http://localhost:8118`, see the demo tasks churning (can take a minute). Note that some of the test tasks are designed to fail (and often), to illustrate what failure would look like. Likewise with misconfigured tasks, and so on.

Don't expose this to the internet; **it's not even trying for security**.

(If you have have libevent/gevent installed, Speedrack will run using that. Flask's dev server is fine, too.)

CHAPTER 2

Configuration

```
speedrack init <target directory>
```

This will write two files to the target directory: `speedrack_settings.py` and `speedrack_tasks.yaml`. They're the settings from the demo mode and are useful to build your tasks from.

2.1 Application settings

`speedrack_settings.py` contains *application behavior*: your application name, mail server settings, and so on. You won't need them all. If you're not familiar with python, you don't need to learn it to administer this application — just stick with the simple declarative syntax.

CHAPTER 3

Sub-path support

If Speedrack is running at a non-root location on a domain (e.g., http://www.example.com/speedrack/ resolves to the application root view), then you'll need to let Speedrack know to modify the URLs it makes appropriately. In your configuration file, set the variable `SCRIPT_NAME` to whatever path prefix points to the Speedrack index (in this case, `SCRIPT_NAME = '/speedrack'`).

Note: This setting alone does not actually change how Speedrack **parses** requests. Rather, this tells Speedrack that when it **generates** a URL, it should prefix it with the provided string. This means that when the client clicks a link or requests a stylesheet in the original response, the new request is pointed at the right location.

Speedrack also needs to be told where the URLs it generates in the emails it sends should point. The variable `EMAIL_URL_PREFIX` should be set to the external root URL of the application (no trailing slash), and then links will be relative to that root URL.

3.1 Task definitions and behavior

`speedrack_tasks.yaml` defines each *task*: a shell command and schedule of execution. The sample file contains the tasks used for the demo you started with.

Warning: **Restrict access to this file.** Speedrack reads this file and executes the commands therein, *as the user who launched speedrack*. `speedrack init` gets the basic settings correct, but if you build your own, you should remove global write (and group, if necessary).

CHAPTER 4

Running

Once you've got your settings and tasks in place:

```
speedrack run --settings speedrack_settings.py --tasks speedrack_tasks.yaml
```

Browse `http://localhost:8118/config` to check your active configuration at any time.

CHAPTER 5

Task Execution

5.1 Scheduling

Speedrack uses simple strings for *interval* (`1h10m`) and APScheduler's add_cron_job for *cron* scheduling (e.g. `{ "day_of_week": 2, "hour": 23 }`. The most current documentation is in the app itself, under the Help tab.

Speedrack can launch jobs *immediately* via the Run Now button.

5.2 Success / Failure

`FAIL_BY_NONZERO_STATUS_CODE` and `FAIL_BY_STDERR` determine different indicators of task failure, both defaulted to true. Setting both of these to false will work, but you'll get less information in the task browser.

This is also configurable per task. The administrator is responsible for getting the defaults / exceptions right.

5.3 Sudo control

Sudo execution requires the hosting user to have password-free sudo access to the given user. Commands will end up being roughly equivalent to:

```
$ sudo -u task_sudo_user task_command
```

5.4 Config Reloading

When you reload the application via the web interface, Speedrack waits for all current tasks to complete. This means that if you have a task that is going to take two hours to complete, it might be a while before Speedrack shuts down. Kill and restart Speedrack if you need it to stop and reload right this second.

CHAPTER 6

About

What's a *speedrack*? At a bar or food service event, the most common mixins are kept at arm-level, for efficiency — this is the speedrack. Every reach is lost time, more fatigue, and the line gets longer. Likewise, every reach for data or regular task execution, the line's getting longer.

Like a lot of geeks, I'll work three days to save myself three minutes during crunch times. It's nice when things just work.

(note: this took longer than three days)

6.1 Ingredients

Speedrack is made from Flask, Twitter Bootstrap, and APScheduler.

6.2 Contributing

Some notes to hopefully save some effort on everyone's part.

- *Bugs*: love to hear about them!
- *Simple contributions* (known bugs, docs, typos, tweaks): just fork and request pull (tests <3)
- *Complex contributions* (new features): *email first*
- *Suggestions to do it right*: Save yourself time. :)

6.3 Gratitude and Inspiration

The good parts of this Flask application drew on knowledge and practices that Andrew Roberts and I accumulated in our work together. He's excellent; the bad parts are all mine.

Speedrack was inspired by Pydog, by Philip Montgomery. My team used Pydog happily for years, but as our reliance on it grew, we needed something a little more stable. We already loved Flask, so it was an easy choice.

CHAPTER 7

Alternatives

Please, play the field. This solution works well for my team, but you should look at what else is out there.

For example, if you have a group of machines you'd like to run tasks on, please investigate Celery or RQ — they're both quality work. There are countless alternatives, and most of them are probably better than this.

Speedrack is trivial to set up and administer. It runs well on one node. It's a good bad solution.

CHAPTER 8

Changes

0.6.0 added SCRIPT_NAME configuration variable for supporting sub-path server roots

0.5.2 bugfix for APScheduler stale dependency and overambitious setup

0.5.1 bugfix for sudo controls

0.5 added in-app manual suspension of tasks, more convenient than commenting out task in yaml and reloading

0.4.1 bugfix for spam_fail

0.4 added spam_fail setting, for hammering failure messages

0.3.1 added simple sudo controls

0.3.0 fixed email and spam settings

0.2.11 add params tab to show single-task execution criteria

0.2.10 add single-file view

0.2.9 reduce number of copied version strings add placeholder for tasks with no description

0.2.8 standard numbering scheme

0.2.h use gevent if available, otherwise use Flask's development server gevent/libevent is no longer a hard requirement, and is not installed with speedrack

0.2.f convert docs to sphinx documentation and wire to readthedocs

0.2.e merge debug and config to one page

0.2 beta

0.1 prototype/alpha

CHAPTER 9

Contributors

Authored by Clint Howarth.

With contributions from:

- Jonathan Eunice

CHAPTER 10

Todo

10.1 1.0

- more tests
- refactor models.Executor / Task / aps.executor_func

10.2 Future

- detect sanity of config variables to ensure no weird write failures
- “retain X previous failures”, to prevent issues from being flushed out by successes in display
- add ability to list all jobs (or what, 100?) if asked instead of pithy message
- add users - no auth, just cookie for attribution
- run now improvements - store “manual execution” in job history / display? / store “executed by”?
- failure notification - raven / sentry
- live update of page? via pjax - 5s interval