cmdfor Documentation

Release 0+untagged.6.gac0aa7b.dirty

Jesse Galley

Apr 10, 2018

Contents

Contents:

CHAPTER 1

cmdfor

nd for every line of input

- Free software: MIT license
- Documentation: (COMING SOON!) https://cmdfor.readthedocs.org.

1.1 Features

A shell utility (and package) which runs a command for every line of input.

It allows for spawning an arbitrary number of concurrent threads, and control over where to keep each commands output.

In my daily work, I have to run all manner of commands on huge batches of items. These things are usually not CPU bound, so it makes sense to multithread these tasks.

Thus, I find myself doing bash commands such as the following, which takes an input file of items, splits it into equal(ish) parts, and then spawns a worker for each part, all the while keeping granular logs and return codes:

```
lines=`wc -l domains.txt | awk '{print $1}'`; threads=10; split=$(((lines/
↪threads)+1)); mkdir -p in out; split -d -l ${split} domains.txt in/part. ; ls in/ |␣
↪while read -r f; do cat in/${f} | while read -r d; do host -t a "${d}" > out/${d} 2>
↪&1; echo -e "${d}\t$?"; done > log.${f} & echo ${!}; done > pids
```

That gets pretty tiring to type all the time. Why not use `xargs -P` you say? Well that works perfectly fine for cases where I don't need to make very complicated commands, and don't need to log all return codes. Maybe I can do all of that with `xargs`, but I wanted to make this anyway as a learning experience.

1.2 How-To

The program can take input from STDIN or from a file passed with the -i option.

All arguments that aren't options are considered the subcommand to run. All wildcards `{}` are replaced with the corresponding positional field from the input data.

To delete a list of files, basically the same behaviour as xargs:

```
cat files.txt | cmdfor rm {}
```

To run the fictional command `imaplogin` for every line of a csv that contains <email>,<password> fields, logging each individual command's output to an file in the directory ./out:

```
cat email_users.csv | cmdfor -d, -o ./out -- imaplogin -u {} -p {}
```

To look up the IP addresses of a huge amount of hostnames, using 10 concurrent threads, and storing each individual commands stdout and stderr in seperate files in the directory ./results, with each file being named after the hostname on which the query was performed:

```
cat hostnames.txt | cmdfor -t 10 -Eo ./results -l 1 -- host -t a {}
```

1.3 To-Do

1. Come up with a real test case. Since this is a shell utility and really only deals with shell subcommands, I don't know what will work and what won't on travis.ci (can I run a shell command there?) 2. By default, it suppresses all output from subprocesses, and writes a message to STDOUT for each process spawn and reap. This output is too verbose for the default behaviour, and so it should be toggled with -v. The default should be quitier and simpler. Perhaps just the returncodes of each task. 3. Refactoring some stuff to be a little less messy. The function signatures are huge, and there are messages generated in odd places. I think it would be better to pass a context object.

CHAPTER 2

Installation

At the command line:

```
$ pip install cmdfor
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv cmdfor
$ pip install cmdfor
```

CHAPTER 3

Usage

To use cmdfor in a project:

```
import cmdfor
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/jwgalley/cmdfor/issues.

If you are reporting a bug, please include:

- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

cmdfor could always use more documentation, whether as part of the official cmdfor docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jwgalley/cmdfor/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *cmdfor* for local development.

1. Fork the *cmdfor* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/cmdfor.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv cmdfor
$ cd cmdfor/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 cmdfor tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.3, 3.4, 3.5 and for PyPy. Check https://travis-ci.org/jwgalley/cmdfor/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Maintainer

- Jesse Galley <jesse@jessegalley.net>

5.2 Contributors

None yet. Why not be the first? See: CONTRIBUTING.rst

CHAPTER 6

History

2018-04-09 v0.1.0 initial release, still need to do tests and docs

CHAPTER 7

Indices and tables

- genindex
- modindex
- search