Archive Rotator Documentation

Release 0.2.1

Max Harper

December 03, 2015

Contents

Flexible utility for rotating backup files.

Contents:

CHAPTER 1

Overview

Flexible utility for rotating backup files.

- Free software: MIT license
- Documentation: https://archive-rotator.readthedocs.org.
- Code: https://github.com/maxharp3r/archive-rotator

This utility rotates files - typically backup archives. It offers three rotation algorithms - FIFO, Tower of Hanoi, and tiered (a generalization of grandfather-father-son). It requires no configuration file, just command-line parameters. The script is stand-alone (python required) and tracks its state by applying a naming convention to rotated files.

Learn about the concept of archive rotation: http://en.wikipedia.org/wiki/Backup_rotation_scheme

1.1 Example Use

We assume that you have an archive, say */path/to/foo/mydump.tgz* that is the result of an external process (e.g., *tar -czf*, or *mysqldump*) that recurs (e.g., using *cron*). You will use this script to add this file into a rotation set. If the size of the set is greater than the maximum number of files to keep (configurable), then the rotation set will be trimmed according to a rotation algorithm, which is configurable.

Example of running:

```
archive-rotator -v -n 5 /path/to/foo/mydump.tgz
```

This will rename mydump.tgz to something like this:

```
/path/to/foo/mydump.tgz.2012-12-20-133640.backup-0
```

Given this configuration, the rotation script automatically keep at most 5 files in the rotation. When the script is run six times, the set of archives would be too big, so the oldest will be deleted. This is an example of the simple (FIFO) rotation algorithm.

CHAPTER 2

Installation

```
pip install archive-rotator
```

CHAPTER 3

Usage

3.1 Command Line

Say we wish to rotate */files/website-backup.zip* daily. Let's use the tower of hanoi algorithm, to balance recency and longevity. We do this:

```
archive-rotator -v --hanoi -n 8 --ext ".zip" /files/website-backup.zip
```

For documentation of command-line parameters, run *archive-rotator -h*:

```
usage: archive-rotator [-h] [-n NUM_ROTATION_SLOTS] [-v] [--ext EXT]
                       [-d DESTINATION_DIR] [--ignore-missing] [--simple]
                       [--hanoi] [--tiered]
                       path

Move a file into a rotation of backup archives.

positional arguments:
  path                  Path of input file to rotate

optional arguments:
  -h, --help            show this help message and exit
  -n NUM_ROTATION_SLOTS, --num NUM_ROTATION_SLOTS
                        Max number of files in the rotation
  -v, --verbose         Print info messages to stdout
  --ext EXT             Look for and preserve the named file extension
  -d DESTINATION_DIR, --destination-dir DESTINATION_DIR
                        Put the rotated archive in this directory.Use if the
                        rotated archives live in a different directory from
                        the source file.
  --ignore-missing      If the input file is missing, log and exit normally
                        rather than exiting with an error
  --simple              Use the first-in-first-out rotation pattern (default)
  --hanoi               Use the Tower of Hanoi rotation pattern
  --tiered              Use the tiered rotation pattern
```

3.2 Programmatic

You can rotate files from python. Example:

```
from archive_rotator import rotator
from archive_rotator.algorithms import SimpleRotator

rotator.rotate(SimpleRotator(5), "/my/path/foo.tar.gz", ".tar.gz", verbose=True)
```

CHAPTER 4

Algorithms & Examples

4.1 FIFO (simple)

This rotation scheme keeps the last n archives. It emphasizes recency at the cost of history and/or disk space.

```
archive-rotator -v -n 5 /path/to/foo/mydump.tgz
```

Given this configuration, the rotation script keeps the most recent 5 files.

4.2 Tower of Hanoi

This rotation scheme is described here: http://en.wikipedia.org/wiki/Backup_rotation_scheme#Tower_of_Hanoi It emphasizes long history and saving disk space, but is not very tunable.

Example of running:

```
archive-rotator --hanoi -v -n 6 /path/to/foo/mydump.tgz
```

Given this configuration, the rotation script automatically keep at most 6 files in the rotation, rotated every 1, 2, 4, 8, 16, and 32 runs, respectively. So, after 32 rotations, the directory will look something like this:

```
/path/to/foo/mydump.tgz.2013-01-03-094732.backup-16
/path/to/foo/mydump.tgz.2013-01-03-094734.backup-24
/path/to/foo/mydump.tgz.2013-01-03-094735.backup-28
/path/to/foo/mydump.tgz.2013-01-03-094736.backup-30
/path/to/foo/mydump.tgz.2013-01-03-094737.backup-31
/path/to/foo/mydump.tgz.2013-01-03-094738.backup-32
```

4.3 Tiered

This is a generalization of the grandfather-father-son rotation algorithm (described here - http://en.wikipedia.org/wiki/Backup_rotation_scheme#Grandfather-father-son). This algorithm is capable of handling a variety of rotation schedules.

This algorithm, unlike the others, accepts a list of one or more *-n* configurations. Each one is a "tier". For example:

```
# three tiers: the first will hold 6 files, the second will hold 3, the third will hold 12
archive-rotator --tiered -v -n 6 -n 3 -n 12 /path/to/foo/mydump.tgz
```

If the example above were run daily, we'd approximate 6 daily, 3 weekly, and 12 monthly backups in the rotation set.

You may configure any number of slots to each tier. If we have a single tier with 8 slots, the algorithm will behave identically to the FIFO algorithm configured with eight slots. If we add a second tier with two slots, then the algorithm will fill one of those two second-tier slots every ninth run. If we add a third tier with one slot, then the algorithm will put the archive into the third tier slot for every two it puts into the second tier.

```
Two tier example: -n 3 -n 2.
id          : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
maps to slot: 0 1 2 3 0 1 2 7 0 1  2  3  0  1  2 ...
tier:       : 0 0 0 1 0 0 0 1 0 0  0  1  0  0  0 ...
```

```
Three tier example: -n 2 -n 2 -n 2.
id          : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
maps to slot: 0 1 2 0 1 5 0 1 8 0  1  2  0  1  5 ...
tier:       : 0 0 1 0 0 1 0 0 2 0  0  1  0  0  1 ...

If we have tiers of size j, k, l, and m, then m is rotated every (j+1)(k+1)(l+1) runs.
```

This algorithm can replicate the behavior of both FIFO and Tower of Hanoi.

```
# FIFO with 6 slots:
archive-rotator --tiered -v -n 6 /path/to/foo/mydump.tgz
# hanoi with 4 slots:
archive-rotator --tiered -v -n 1 -n 1 -n 1 -n 1 /path/to/foo/mydump.tgz
```

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/maxharp3r/archive-rotator/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/maxharp3r/archive-rotator/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 :)

5.2 Get Started!

Ready to contribute? Here's how to set up *archive-rotator* for local development.

1. Fork the *archive-rotator* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/archive-rotator.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 archive-rotator
$ cd archive-rotator/
$ 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:

```
$ make lint
$ make test
$ make test-all
```

   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.

5.3 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 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/maxharp3r/archive-rotator/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_archive_rotator
```

CHAPTER 6

History

6.1 0.2.1 (2015-12-03)

- Improve and document programmatic access

6.2 0.2.0 (2015-12-03)

- Much improved test coverage and refactored codebase
- Added –destination-path parameter

6.3 0.1.0 (2015-11-23)

- First release on PyPI

CHAPTER 7

Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-pypackage

CHAPTER 8

Indices and tables

- genindex
- modindex
- search