picky Documentation

Release 0.9

Simplistix Ltd

June 22, 2015

Contents

Picky is a tool for making sure that the packages you have installed with pip or conda match those you have specified.

Both pip and conda have a notion of a file containing the package specifications for an environment. For pip, the name `requirements.txt` is used; conda doesn't appear to have a standard name yet, so picky defaults to using `conda_versions.txt`.

Regardless of the name, problems can arise when the specification in those files is either incorrect or incomplete, resulting in unexpected packages or versions of packages being used across development, testing and production environments.

Picky provides a quick check that can be used as part of a continuous integration pipeline to ensure that all packages in an environment are as specified:

```
$ picky
testfixtures 4.1.2 in pip freeze but 4.1.0 in requirements.txt
Babel 1.3 missing from pip freeze
python 2.7.9 in conda list -e but 3.4.0 in conda-versions.txt
readline 6.2 missing from conda list -e
```

If the specifications don't match the environment, the return code is set, which will hopefully cause a continuous integration job to fail:

```
$ echo $?
1
```

Picky can also be used to update an existing set of specifications:

```
$ picky --update
testfixtures 4.1.2 in pip freeze but 4.1.0 in requirements.txt
Babel 1.3 missing from pip freeze
python 2.7.9 in conda list -e but 3.4.0 in conda-versions.txt
readline 6.2 missing from conda list -e
Updating 'requirements.txt'
Updating 'conda_versions.txt'
```

CHAPTER 1

Installation Instructions

The best way to install picky is with pip:

```
pip install picky
```

Of course, once it's installed, make sure it's in your `requirements.txt`!

Python version requirements

This package has been tested with Python 2.6, 2.7, 3.3+ on Linux, and is also expected to work on Mac OS X and Windows.

CHAPTER 2

Usage

Picky has three main uses cases:

- creating requirements files from an existing environment:

```
$ picky --update
Babel 1.3 missing from requirements.txt
python 2.7.9 missing from conda_versions.txt
Updating 'requirements.txt'
Updating 'conda_versions.txt'
```

- ensuring the requirements completely match the packages installed:

```
$ picky
testfixtures 4.1.2 in pip freeze but 4.1.0 in requirements.txt
Babel 1.3 missing from pip freeze
python 2.7.9 in conda list -e but 3.4.0 in conda-versions.txt
readline 6.2 missing from conda list -e
```

- updating the specifications based on the current environment:

```
$ picky --update
testfixtures 4.1.2 in pip freeze but 4.1.0 in requirements.txt
Babel 1.3 missing from pip freeze
python 2.7.9 in conda list -e but 3.4.0 in conda-versions.txt
readline 6.2 missing from conda list -e
Updating 'requirements.txt'
Updating 'conda_versions.txt'
```

2.1 Return codes

The return code set by `picky` will be non-zero if the requirements files do not exactly match the packages found in the environment:

```
$ picky
Babel 1.3 missing from requirements.txt
python 2.7.9 missing from conda_versions.txt
$ echo $?
1
```

This can be useful in continuous integration environments to check that all packages used in your environment are specified and pinned to specific versions by your requirements files.

2.2 Log levels

If you want more information about what `picky` is doing, run it with a lower log level, such as `debug`:

```
$ picky -l debug
2015-05-01 09:08:10 INFO    Using '/path/to/pip' for pip
2015-05-01 09:08:10 INFO    Using 'requirements.txt' for pip
2015-05-01 09:08:10 INFO    Using '/path/to/conda' for conda
2015-05-01 09:08:10 INFO    Using 'conda_versions.txt' for conda
```

2.3 Using with Pip

By default, `picky` will look for the `pip` binary on the current `$PATH` and will look for the requirements in a file called `requirements.txt` in the current working directory.

Both the location of the pip binary and the requirements file can be specified explicitly by using the `--pip` and `--pip-requirements` options, respectively.

2.4 Using with Conda

By default, `picky` will look for the `conda` binary on the current `$PATH` and will look for the requirements in a file called `conda_versions.txt` in the current working directory.

Both the location of the pip binary and the requirements file can be specified explicitly by using the `--conda` and `--conda-versions` options, respectively.

A new conda environment can be created from a `conda_versions.txt` file as follows:

```
conda create -n <environment name> --file conda_versions.txt
```

Note: Build numbers, the third section of a conda version specifier, are ignored by `picky` as these may differ across platforms even though the package version is otherwise identical.

2.5 Using with Conda and Pip

When used in a conda environment that also has some packages installed with pip, picky will ensure that both the conda and pip requirements files do not conflict with each other, and that a package will only appear in one or other of the requirements files.

CHAPTER 3

Development

If you wish to contribute to this project, then you should fork the repository found here:

https://github.com/Simplistix/picky/

Once that has been done, you can follow these instructions to perform various development tasks:

3.1 Setting up the environment

All development requires that you have a conda environment set up, this can be created by doing the following from within a checkout of the above repository, assuming you have installed conda by following its instructions:

```
$ conda create -n picky --file=conda_versions.txt
$ source activate picky
(picky) $ pip install -e .[build,test]
```

3.2 Running the tests

Once you have set up and activated your conda environment, the tests can be run from the root of your checkout as follows:

```
$ nosetests
```

3.3 Building the documentation

The Sphinx documentation is built by doing the following from the directory containing setup.py:

```
$ cd docs
$ make html
```

To check that the description that will be used on PyPI renders properly, do the following:

```
$ python setup.py --long-description | rst2html.py > desc.html
```

The resulting `desc.html` should be checked by opening in a browser.

3.4 Making a release

The following should be done with your conda environment activated and will build the distribution, upload it to PyPI and register the metadata with PyPI:

```
$ pip install -e .[build,test]
$ python setup.py sdist bdist_wheel
$ twine upload dist/picky-<version>*
```

Running pip again will make sure the correct package information is used.

This should all be done on a unix box so that a *.tgz* source distribution is produced.

Once the above is done, make sure to go to https://readthedocs.org/projects/picky/versions/ and make sure the new release is marked as an Active Version.

CHAPTER 4

Changes

4.1 0.9 (22 June 2015)

- Python 3 support
- Fixed handling of package ‘extras’ in pip output and specifications.
- Fixed handling of arbitrary equality clauses in pip output and specifications.

4.2 0.8 (22 June 2015)

- Initial Release

CHAPTER 5

Still to Come

- Automated code coverage metrics

CHAPTER 6

License

CHAPTER 7

Indices and tables

- genindex
- modindex
- search