Tox-Travis Documentation

Release 0.12

Ryan Hiebert

Dec 25, 2022

Contents

tox-travis is a plugin for tox that simplifies the setup between tox and Travis.

CHAPTER 1

Usage

Configure the Python versions to test with in `.travis.yml`, and install `tox-travis` with pip:

```
language: python
python:
  - "3.6"
  - "3.7"
install: pip install tox-travis
script: tox
```

tox will only run the `py36` or `py37` env (or envs that have a factor that matches) as appropriate for the version of Python that is being run by each Travis job.

CHAPTER 2

Topics

2.1 Env Detection

Env detection is the primary feature of Tox-Travis. Based on the matrix created in `.travis.yml`, it decides which Tox envs need to be run for each Travis job.

2.1.1 Usage

Configure the Python versions to test with in `.travis.yml`:

```
language: python
python:
  - "3.6"
  - "3.7"
install: pip install tox-travis
script: tox
```

And it will run the appropriate testenvs, which by default are any declared env with `py36` or `py37` as factors of the name. If no environments match a given factor, the `py36` or `py37` envs are used as a fallback.

2.1.2 Advanced Configuration

To customize what environments tox will run on Travis, add a section to `tox.ini` telling it what environments to run under which versions of Python:

```
[tox]
envlist = py{36,37}-django{21,22}, docs

[travis]
python =
  3.6: py36
  3.7: py37, docs
```

This would run the Python 3.6 variants under 3.6, and the Python 3.7 variants and the `docs` env under 3.7.

Note that Travis won't run all the envs simultaneously, because its build matrix is only aware of the Python versions. Only one Travis build will be run per Python version, unless other settings are specified in the Travis build matrix.

If you are using multiple Travis factors, then you can use those factors to decide what will run. For example, see the following `.travis.yml` and `tox.ini`:

```
language: python
python:
  - "3.6"
  - "3.7"
env:
  - DJANGO="2.1"
  - DJANGO="2.2"
matrix:
  include:
    - os: osx
      language: generic
install: pip install tox-travis
script: tox
```

```
[tox]
envlist = py{36,37}-django{21,22}, docs

[travis]
os =
  linux: py{36,37}-django{21,22}, docs
  osx: py{36,37}-django{21,22}
python =
  3.7: py37, docs

[travis:env]
DJANGO =
  2.1: django21
  2.2: django22, docs
```

Travis will run 5 different jobs, which will each run jobs as specified by the factors given.

- os: linux (default), language: python, python: 3.6, env: DJANGO=2.1

  This will run the env `py36-django21`, because `py36` is the default, and `django21` is specified.

- os: linux (default), language: python, python: 3.7, env: DJANGO=2.2

  This will run the env `py37-django22`, but not `docs`, because `docs` is not included in the DJANGO 2.2 configuration.

- os: linux (default), language: python, python: 3.6, env: DJANGO=2.1

  This will run the env `py36-django21`, because `py36` is the default. `docs` is not run, because Python 3.6 doesn't include `docs` in the defaults that are not overridden.

- os: linux (default), language: python, python: 3.7, env: DJANGO=2.2

  This will run the envs `py37-django22` and `docs`, because all specified factors match, and `docs` is present in all related factors.

- os: osx, language: generic

  This will run envs `py36-django21`, `py37-django21`, `py36-django22`, and `py37-django22`, because the `os` factor is present, and limits it to just those envs.

2.1.3 Unignore Outcomes

By default, when using `ignore_outcome` in your Tox configuration, any build errors will show as successful on Travis. This might not be desired, as you might want to control allowed failures inside your `.travis.yml`. To cater this need, you can set `unignore_outcomes` to `True`. This will override `ignore_outcome` by setting it to `False` for all environments.

Configure the allowed failures in the build matrix in your `.travis.yml`:

```
matrix:
  allow_failures:
  - python: 3.6
    env: DJANGO=master
```

And in your `tox.ini`:

```
[travis]
unignore_outcomes = True
```

2.2 After All

Deprecated since version 0.10.

> **Warning:** This feature is deprecated.
>
> Travis has added Build Stages, which are a better solution to this problem. You will also likely want to check out Conditions, which make it much easier to determine which jobs, stages, and builds will run.

Inspired by travis-after-all and travis_after_all, this feature allows a job to wait for other jobs to finish before it calls itself complete.

There are three environment variables that can be used to configure this feature.

- `GITHUB_TOKEN`. This is *required*, and should be encrypted in the `.travis.yml`, or set securely in the repository settings. This is used as the authentication method for the Travis CI API.
- `TRAVIS_POLLING_INTERVAL`. How often, in seconds, we should check the API to see if the rest of the jobs have completed. Defaults to 5.
- `TRAVIS_API_URL`. The base URL to the Travis API for this build. This defaults to `https://api.travis-ci.org`. A common override will be to the commercial version, at `https://api.travis-ci.com`.

Configure which job to wait on by adding the `[travis:after]` section to the `tox.ini` file. The `travis` key looks for values that would be keys in various items in the `[travis]` section, and the `env` key looks for values that would be keys in items in the `[travis:env]` section.

For example:

```
[travis:after]
travis = python: 3.5
env = DJANGO: 1.8
```

Then run `tox` in your test command like this:

```
tox --travis-after
```

For example, consider this mocked up `.travis.yml`, that corresponds to using the above `travis:after` section:

```
language: python
python:
  - "2.7"
  - "3.5"
env:
  global:
    - GITHUB_TOKEN='spamandeggs'  # Make sure this is encrypted!
  matrix:
    - DJANGO="1.7"
    - DJANGO="1.8"
install: pip install tox-travis
script: tox --travis-after
deploy:
  provider: pypi
  user: spam
  password: eggs  # Make sure to encrypt passwords!
  on:
    tags: true
    python: 3.5
    condition: $DJANGO = "1.8"
  distributions: sdist bdist_wheel
```

This example deploys when the build is from a tag and the build is on Python 3.5 and the build is using DJANGO="1.8". Together `tox --travis-after` and Travis' `on` conditions make sure that the deploy only happens after all tests pass.

If any configuration item does not match, or if no configuration is given, this will run exactly as it would normally. However, if the configuration matches the current job, then it will wait for all the other jobs to complete before it will be willing to return a success return code.

If the tests fail, then it will not bother waiting, but will rather return immediately. If it determines that another required job has failed, it will return an error indicating that jobs failed.

You can use this together with a deployment configuration to ensure that this job is the very last one to complete, and will only be successful if all others are successful, so that you can be more confident that you are shipping a working release.

The accepted configuration keys in the `[travis:after]` section are:

- `envlist`. Match with the running toxenvs. Expansion is allowed, and if set *all* environments listed must be run in the current Tox run.
- `travis`. Match with known Travis factors, as is done in the `[travis]` section. For instance, specifying that we should wait when python is version 2.7 would look like `travis = python: 2.7`.
- `env`. Match with environment variable factors, as might be specified in the `[travis:env]` section. For instance, if we want to match that `DJANGO` is `1.9`, then it would look like `env = DJANGO: 1.9`. The value must match exactly to succeed.

2.3 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:

2.3.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/tox-dev/tox-travis/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/tox-dev/tox-travis/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 :)

2.3.2 Get Started!

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

1. Fork the *tox-travis* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/tox-travis.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

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

```
$ tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

2.3.3 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 and 3.4+ and for PyPy, PyPy3. Check https://travis-ci.org/tox-dev/tox-travis/pull_requests and make sure that the tests pass for all supported Python versions.

2.4 Changelog

2.4.1 0.13

- Add Python 3.7 support in trove classifiers.

2.4.2 0.12 (2019-03-14)

- Fix reading envlist from `setup.cfg` (#110). - thanks to @voronind for the pull request.
- Add docs and tests to sdist (#121). - thanks to @jayvdb for the pull request.
- Release an sdist in addition to the wheel.

2.4.3 0.11 (2018-09-21)

- Drop support for Python 3.2 and 3.3 (#113).
- Fix autogen_configs for tox 3.4.0 (#115).
- Various documentation fixes.

2.4.4 0.10 (2017-11-13)

- Deprecate the After All feature (#93). Travis now has Build Stages, which are a better solution.

2.4.5 0.9 (2017-11-12)

- Allow PyPy3 support to work with PyPy3 5.5 (#66). - thanks to @kirbyfan64 for the pull request.
- Move toxenv to tox_configure hook (#78). - thanks to @rpkilby for the pull request demonstrating the idea.
- Respect Tox config file CLI option (#59). - thanks to @giginet for the bug report.
- Move the project into the `tox-dev` GitHub organization. - thanks to @obestwalter for bringing it up, and @rpkilby for helping fix references to the old location.
- Various refactors and test improvements. - thanks to @jdufresne for several pull requests and @rpkilby for many reviews.
- Only deploy the universal wheel to PyPI (#87). Due to a deployment bug, a version-specific egg was released, along with the intended sdist and wheel. The sdist has also been abandoned for release.

2.4.6 0.8 (2017-01-11)

- Add Python 3.6 support in trove classifiers.
- Skip after waiting for pull requests (#46). - thanks to @rpkilby for fixing this bug.
- Add `unignore_outcomes` setting to allow reversing Tox's `ignore_outcomes` setting on Travis (#48). - thanks to @Bouke for the implementation.

2.4.7 0.7.2 (2016-12-20)

- Undo the README changes, and fix HISTORY markup for PyPI.

2.4.8 0.7.1 (2016-12-20)

- Fix the README markup to display properly on PyPI.

2.4.9 0.7 (2016-12-20)

- Deprecate the `[tox:travis]` section in favor of the `python` key to the new `[travis]` section.
- Allow specifying envs by other Travis factors. Includes `os`, `language`, and `python`.
- Allow specifying envs for environment variables, in a new `[travis:env]` section.
- Special thanks to @rpkibly for driving this work (#34)
- Backward incompatible changes:
  - If *any* declared tox envs match the envs matched from factors, no additional envs will be included automatically. For example, if `envlist` is `docs`, and the configuration for python 3.4 is `py34, docs`, it previously would have run both the declared `docs` env, as well as the undeclared `py34` env, while now it will only run the declared `docs` env. This may result in *fewer* envs running than expected, but in edge cases that were believed to be unlikely.
  - Previously, if no Python version was given in the environment, it would automatically choose an appropriate env based on the Python version running. Now if no Python version is given in the environment no env is determined by default, which may result in *more* envs running in a job than expected.

- Add the `--travis-after` command to enable a job to wait until all others have completed. (#13) - thanks to @ssbarnea for the feature suggestion.

2.4.10 0.6 (2016-10-13)

- Require pytest<3 for Python 3.2 (#33)

2.4.11 0.5 (2016-07-28)

- Prefer `TRAVIS_PYTHON_VERSION` to sys.version_info (#14) - thanks to @jayvdb for the code review
- Add Python 3.2 support (#17) - thanks to @jayvdb for the bug report, discussion, and code review
- Support PyPy3 v5.2 with setuptools hackery (#24) - thanks to @jayvdb for the pull request

2.4.12 0.4 (2016-02-10)

- Generate default env from sys.version_info (#9) - thanks to @jayvdb for the bug report

2.4.13 0.3 (2016-01-26)

- Match against testenvs that are only declared as sections (#7) - thanks to @epsy
- Include unmatched envs verbatim to run (also #7) - thanks to @epsy again

2.4.14 0.2 (2015-12-10)

- Choose testenvs from `tox.ini` by matching factors.
  - This is a slightly *backward incompatible* change
  - If a Python version isn't declared in the `tox.ini`, it may not be run.
  - Additional envs may be run if they also match the factors, for example, `py34-django17` and `py34-django18` will both match the default for Python 3.4 (`py34`).
  - Factor matching extends to overrides set in `tox.ini`.

2.4.15 0.1 (2015-05-21)

- Initial Release

2.5 License