Automate your publishing to PyPI with PBR and Travis Documentation

73VW

Oct 14, 2020

Contents

Automate your publishing to PyPI with PBR and Travis

1.1 ✓ Step 1: write the setup for your python project

If you want to publish your project to PyPi, you first have to make a setup file.

In this tutorial, I will cover the use of PBR which simplifies the process.

1.1.1 *Setup.py*

As you can see in the python setup file included in this repository, the syntax is quite basic.

```python
"""Setup example."""

from setuptools import setup


setup(
    setup_requires=['pbr'],
    pbr=True
)
```

You simplify define that you want to use PBR and that's it!

1.1.2 *Setup.cfg*

As you can see in the config setup file included in this repository, the syntax not more complicated than the last file.

Let's go through every section together.

First of all, the metadata section:

```
# Type of python distribution
[bdist_wheel]
universal=0

[metadata]
# App name
name = Publishing to PyPI with pbr and Travis
# Who made it?
author = Maël Pedretti
# Do I really need to explain the following?
author_email = mael.pedretti@he-arc.ch
# The short description of your app
summary = Publishing to Pypi with PBR and Travis.
# License type
license = MIT
# Which file contains the long description?
description-file =
    README.rst
# Where can I access the project?
home-page = https://github.com/73VW/Publishing-to-PyPI-with-pbr-and-Travis
# Which version of Python does it need to run?
python_requires = >=3.6
# How do you classify your app? https://pypi.python.org/pypi?%3Aaction=list_
↪classifiers
classifier =
    Development Status :: 4 - Beta
    Environment :: Other Environment
    Intended Audience :: Education
    Operating System :: MacOS :: MacOS X
    Operating System :: Microsoft :: Windows
    Operating System :: POSIX
    Programming Language :: Python :: 3 :: Only
    Programming Language :: Python :: 3.6
    Programming Language :: Python :: Implementation :: CPython
    Topic :: Education

# Automatically find root package
[options]
packages = find:

# Which files that are not source code do you want to deploy?
[files]
data_files =
    some/example = some/example/*

# Where does your app start?
[entry_points]
console_scripts =
    automabot = your_package.__main__:main
```

After a few tweaking you are now ready to go!

1.2 ✓ Step 2: Enable Travis!

Two ways of enabling Travis are presented here. One using Travis CLI and one without.

1.2.1 *Using travis CLI*

Run `travis login` and login to travis.

Now you can run `travis init`.

If you are in a git repository, Travis will detect it and ask if this is correct.

Otherwise, it will tell you it can detect the repo.

Once you hit `Enter`, Travis asks the main language. In this case, type `Python`.

Now a new file called `.travis.yml` has been created and is available in your repo. Moreover, Travis is now enabled for this repo.

We will go through this file later.

1.2.2 *Manually*

- Go to Travis home page.
- Sign up or Sign in.
- Go to your profile page and sync your account.
- Your public Github repositories are now listed above.
- Toggle the project you want.

1.2.3 *.travis.yml*

Now let's write our setting file.

As the doc is really well made, I suggest you go check it first as I won't explain everything. You can find it here https://docs.travis-ci.com/user/getting-started/.

However, I will explain the settings I usually use.

```
# Do I really need to explain this line?
language: python

# You can use a cache to build faster
cache: pip

# python version. You can define more than one if you want to run multiple tests
python:
  - '3.6'

# your install script or your install list
install: pip install rstcheck

# your test script or your install list
script: rstcheck --recursive .

# settings for notifications, I personnaly don't like to be spammed on my email
notifications:
  email:
    on_failure: never
    on_pull_requests: never
```

```
# the interresting part!
deploy:
  # If you need to deploy files Travis has built, use the next line
  skip_cleanup: true
  # In this case we want to deploy to pypi
  provider: pypi
  # What distribution we want to deploy
  distributions: sdist bdist_wheel
  # When do we want to deploy?
  on:
    # In this case I want to deploy only when a tag is present...
    tags: true
    # ... and when tag is on master and respects the form "v0.0.0"
    branch:
      - master
      - /v?(\d+\.)?(\d+\.)?(\*|\d+)$/
  # Your pypi username
  user: 73VW
  # Your Pypi password secured by Travis if you have Travis CLI installed
  password:
    secure: cGJz+vETnxwWAZQvzveJKOyn3rWy3/
↪tcVmJvTVuflrgKgwMRm+sfQZB3vo39LzDcDbMzlzxLO4SUsqDpCxlPPM1pCjqHeUkke76pXA3HGTqfSS5VBic979pBDBqzFe8SE
↪yiLDv/qZpi6cooxJBtlK184AZvCIfjiu8ua5JqJ/SBghzrwLf7b5VbWg/
↪WOtS8NEB+TYhZhpmkYLPXnOoJLYbbrOYA/sz/QfwXke2NCTp7apZFAtU1lFN2gVWsmff7ysRWwwHW/
↪iidCAcu9BXlwMt2x2dv5PqSSqN1QdwCQ+cGcewlIPInHwCpXwI4sJXPEHeax0J5c206Yf4PMkzgrUj1+UmpB2AKJkMF0+kGd+MC
  # Use the following if you don't have Travis CLI
  password : ${PYPI_PASSWORD}
```
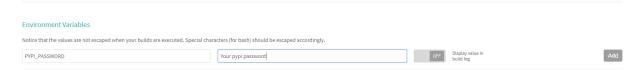
1.2.4 *Password*

If you don't have Travis CLI installed, use the second option I mentioned above and do the following:

- In your profile page, find your project and click on the little gear . This will bring you to the settings.

- Go to the `Environment Variables` section and add a new variable.

- If you take my example, its name will be PYPI_PASSWORD and the value your password.

If you have Travis CLI, this one is for you.

- Leave blank the password section, like the following.

```
user: 73VW
# Your Pypi password
password:
```

- Now let's encrypt it! Simply run `travis encrypt --add deploy.password` and Travis will ask for your password, encrypt it and paste it to the file.

Now your are ready to go!

1.3 ✓ *So what now?!*

Well, let's try to push everyhing to the repository to check if everything is alright and if the tests pass!

Go to Travis home page and check if everything went well!

As you remember, we haven't set up any tag in github so this commit shouldn't get deployed.

Travis will also tell it:

```
Skipping a deployment with the pypi provider because this is not a tagged
commit
```

1.4 ✓ *Let's tag it!*

Now, create a tag. This is easy with git. Git tag doc can be found here.

Note that with `git tag` the option `-a` allows you to specify the version and `-m` the message.

So your command will be the following:

```
git tag -a 0.0.1 -m "First pypi deployment"
```

Now you can check if it was created by running `git tag`. The result should look like the following.

```
$ git tag
v0.0.1
```

And now push and check again Travis and pypi and your package should be deployed!

PSA: Don't forget to add `--tags` to your push command otherwise they will stay in your local repo.

✓**Deployed!**

1.5 *Global notes*

✓ Your project must be public in order to use Travis. Otherwise you have to upgrade to Travis pro.

✓ Your email address must be verified on pypi in order to upload a new project. Otherwise upload will be rejected.

✓Your tag version **MUST** be in the form [DIGIT.DIGIT.DIGIT]. Check https://docs.openstack.org/pbr/3.1.0/semver.html for more infos.

Note: This project is translated in a few languages by awesome contributors. You will soon be able to select the language in the bottom left corner.

If you see a mistake, feel free to open an issue here

Our great contributors

- @DeKan
- @krishnaprasanthg
- @dafinoer
- @leoassuncao
- @ehx
- @cassianofnts
- @qwerty432
- @IonTarna
- @Ectalite
- @spitzlei