Publishing Your Python Package on conda and conda-forge

Will Norris Follow

Feb 10, 2021 · 5 min read ★

It is worth publishing your python packages on conda and conda-forge

Photo by David Clode on Unsplash

Why Use Conda?

I recently wrote a post guiding users through the process of publishing their python package to PyPI to be installed via pip. However, many users prefer to use conda. I like to have my packages available on conda since it plays nicely with packages that have external dependencies. Sometimes a package will also only be available on a conda channel, and it is best to avoid mixing pip and conda installations in a single environment in order to steer clear of dependency issues.

> ***Note**: If you have not already published your package on PyPI, make sure to see the initial steps I layout in my other post. You need to have the correct file hierarchy, `__init__.py` files, and a License. See my example Github repo for an example of proper file hierarchy.*

Conda Channels

Channels are simply the locations where packages are stored. Some very popular packages are stored in the default conda channel, many smaller packages are stored in individual user's own channel, or there are community channels like conda-forge.

You can specify the channel you would like to use when installing a package in the command line. Or you can create a conda configuration file (`.condarc`), which allow you to list channels in order of priority.

```
conda install swepy ––channel conda–forge
```

Here, I am specifying that I would like to download the python package `swepy` from the conda-forge channel. Conda-forge is a channel made of thousands of contributors; it operates like PyPI as a central repository for many people's packages. But what makes it exceptional is that every package on conda-forge must have all of its dependencies on conda-forge as well. This means that you can use conda-forge for your entire environment and avoid pesky dependency issues!

In this post I will show you how to publish your package on your personal conda channel, and then I will go through the steps needed to publish on the conda-forge channel.

Publishing a Conda Package

1. Create a `meta.yaml` file

2. Create your `build.sh` and/or `bld.bat` files

Creating a meta.yaml file

Your `meta.yaml` file is the recipe that tells conda how to build your library. It is similar to the `setup.py` used to create a library for pip.

Here is an example `meta.yaml` from one of my early libraries, swepy:

```yaml
package:
  name: "swepy"
  version: "1.3.0"

source:
  git_rev: v1.3.1
  git_url: https://github.com/wino6687/SWEpy

requirements:
  host:
    - python
    - setuptools
  build:
    - python
  run:
    - python
    - affine
    - gdal
    - pynco
    - pandas
    - jupyter
    - ipykernel
    - requests
    - tqdm
    - scikit-image
    - scikit-learn
    - netCDF4
    - cython

about:
  home: https://github.com/wino6687/SWEpy
  license: MIT
  license_familY: MIT
  license_file: LICENSE
  summary: "Easy, quick access to CETB files for SWE analysis"
```

```
36
37  extra:
38    recipe-maintainers:
39      - wino6687
```

meta.yaml hosted with ❤️ by GitHub view raw

We can see that I list the package information first, then where conda can find the source code. In the `requirements` section, the list under `run:` holds all of the dependencies your library requires. And finally, the `about` section holds the basic information like the License and a brief summary.

Create build.sh and/or bld.bat Files

- `build.sh` is the shell script for macOS and linux
- `bld.bat` is the batch file for windows

These files are quite simple, and should be exactly as show below:

```
1  "%PYTHON%" setup.py install
2  if errorlevel 1 exit 1
```

bld.bat hosted with ❤️ by GitHub view raw

```
1  $PYTHON setup.py install     # Python command to install the script.
```

build.sh hosted with ❤️ by GitHub view raw

Build Your Package

Now, in the root directory of your package, run the following command:

```
conda build .
```

Once that is complete it will display the path to the build file. It should looks something like this:

```
~/anaconda/conda-bld/linux-64/packagename-py37_0.tar.bz2
```

Upload Your Package to Anaconda

1. Create an account with Anaconda
2. Install the Anaconda Client `conda install anaconda-client`
3. Log into your Anaconda account with `anaconda login`
4. Upload your package using the file path from earlier

```
anaconda upload ~/anaconda/conda-bld/linux-64/packagename-
py37_0.tar.bz2
```

Congrats! Your package will now be available on your personal channel and can be installed via conda instead of pip. You will need to specify your personal channel when installing:

```
conda install mypackage --channel <username>
```

Or add it to your `.condarc` file:

```
conda config --add channels <username>
conda install mypackage
```

conda-forge

Conda-forge is by far my favorite place to publish python packages. Before a package can be published on conda-forge it must go through a brief review process, and all of its

dependencies must be already installable via conda-forge. Using conda-forge for all of your packages can save you some serious headaches by avoiding dependency issues that can arise by mixing channels or installers.

Steps to Take Before Submitting to conda-forge

1. Write unit tests (I like to use pytest)
2. Ensure all dependencies are published on conda-forge already
3. Write at least basic documentation for your package

It is ideal that you have written some unit tests for your package, so that they can be run when updating your conda-forge feedstock. You also should have your code documented so users have some idea of how to use it.

Adding Your conda-forge Recipe

1. Fork conda-forge/staged-recipes
2. Create a new branch from the staged-recipes master branch
3. Add a new recipe (`meta.yaml`) in the "recipes/<mypackagename>/ directory
4. Submit these changes as a pull request (This will trigger testing on all operating systems)
5. Once merged, your will get a new feedstock repository where your package will be built and uploaded to conda-forge!

Out of all of these steps, the only part that can be a bit confusing is creating your recipe. The file hierarchy you should create in the forked "staged-recipes" repo looks like this:

```
/recipes
    /<packagename>
        meta.yaml
```

Below is the recipe I have used for one of my libraries, SWEpy. The recipe is a `meta.yaml` file much like the one used for your basic conda package with a little more detail required.

There are a couple things to note:

- I am using my PyPI package as the source. This way, when I update my PyPI package with my CI service, my conda-forge package is automatically updated as well.
- I have a more complex testing suite setup for conda-forge that involves mocking a server and serving two small files in order to test file scraping without needing access to the web.

- Recipes can be much more complex than this, and involve installing dependencies outside of Python. Refer to the conda-forge docs to see how to make more complex recipes.

Once you submit your pull request, someone from conda-forge will comment if there are any changes that need to be made before it is merged into it's own feedstock. After it has been merged into its own feedstock, it will be built and then deployed so that it can be downloaded via the conda-forge channel!

Wrapping Up

You now know how to publish your Python library on conda and conda-forge! But there is nothing more annoying than manually maintaining your packages in multiple places.

I solved this issue by setting up continuous integration with Travis-CI that automatically deploys any new versions of my code to PyPI, and then deploys new versions from PyPI to my conda-forge feedstock.

You can read my other post about how I setup my workflow to make Python libraries more robust. Continuous integration is your best friend when it comes to updating your libraries; Travis will run your unit tests, and if they pass, deploy them to keep your packages up to date. This way you never have to bother with manually uploading new versions again.

Links and Documentation

- Example Github Repo
- My Guide for Building Robust Python Libraries
- Conda Package Documentation
- conda-forge Add Recipe Guide

Get this newsletter

Python Anaconda Conda Python Libraries Open Source

About Write Help Legal

Get the Medium app