# Building conda packages from scratch



This tutorial describes how to build a conda package for Pyinstrument by writing the required files in the conda build recipe.

## Who is this for?

This tutorial is for Windows, macOS and Linux users who wish to generate a conda package by writing the necessary files. Prior knowledge of conda build and conda recipes is helpful.

## Before you start

- Check the prerequisites.
- You should have already completed Building conda packages with conda skeleton.

## Editing the meta.yaml file

1. Make a new directory for this tutorial named `pyinstrument`, and then change to the new directory:

```
mkdir pyinstrument
cd pyinstrument
```

2. To create a new `meta.yaml` file, open your favorite editor. Create a new text file and insert the information shown below. A blank sample `meta.yaml` follows the table to make it easier to match up the information.

   NOTE: To allow correct sorting and comparison, specify `version` as a string.

| | |
|---|---|
| name | pyinstrument |
| version | “0.13.1” (or latest from https://github.com/joerick/pyinstrument/releases) |
| git_rev | v0.13.1 (or latest from https://github.com/joerick/pyinstrument/releases) |
| git_url | https://github.com/joerick/pyinstrument.git |
| imports | pyinstrument |
| home | https://github.com/joerick/pyinstrument |
| license | BSD |
| license_file | LICENSE |

```
package:
  name:
  version:

source:
  git_rev:
  git_url:

requirements:
  host:
    - python
    - setuptools

  run:
    - python

test:
  imports:
    -

about:
  home:
  license:
  license_file:
```

3. Save the file in the same `pyinstrument` directory as `meta.yaml`. It should match this meta.yaml file.

## Writing the build script files build.sh and bld.bat

Besides `meta.yaml`, 2 files are required for a build:

- `build.sh`—Shell script for macOS and Linux.
- `bld.bat`—Batch file for Windows.

These 2 build files contain all the variables, such as for 32-bit or 64-bit architecture—the ARCH variable—and the build environment prefix—PREFIX. The 2 files `build.sh` and `bld.bat` must be in the same directory as your `meta.yaml` file.

This tutorial describes how to make both `build.sh` and `bld.bat` so that other users can build the appropriate package for their architecture.

1. Open a text editor and create a new file named `bld.bat`. Type the text exactly as shown:

```
"%PYTHON%" setup.py install --single-version-externally-managed --
record=record.txt
if errorlevel 1 exit 1
```

NOTE: In `bld.bat`, the best practice is to to add `if errorlevel 1 exit 1` after every command so that if the command fails, the build fails.

2. Save this new file `bld.bat` to the same directory where you put your `meta.yaml` file.
3. Open a text editor and create a new file named `build.sh`. Enter the text exactly as shown:

```
$PYTHON setup.py install --single-version-externally-managed --
record=record.txt  # Python command to install the script.
```

4. Save your new `build.sh` file to the same directory where you put the `meta.yaml` file.

You can run `build.sh` with `bash -x -e`. The `-x` makes it echo each command that is run, and the `-e` makes it exit whenever a command in the script returns nonzero exit status. If you need to revert this in the script, use the `set` command in `build.sh`.

## Building and installing

Now that you have your 3 new build files ready, you are ready to create your new package with conda build and install the package on your local computer.

1. Run conda build:

```
conda-build pyinstrument
```

When conda build is finished, it displays the package filename and location.

In this case the file is saved to:

```
~/anaconda/conda-bld/linux-64/pyinstrument-0.13.1-py27_0.tar.bz2
```

NOTE: Save this path and file information for the next task. The exact path and filename varies depending on your operating system and whether you are using Anaconda or Miniconda. The `conda-build` command tells you the exact path and filename.

2. Install your newly built program on your local computer by using the `use-local` flag:

```
conda install --use-local pyinstrument
```

If there are no error messages, Pyinstrument installed successfully.

## Converting a package for use on all platforms

Now that you have built a package for your current platform with conda build, you can convert it for use on other platforms by using the 2 build files, `build.sh` and `bld.bat`.

Use the `conda convert` command with a platform specifier from the list:

- `osx-64`.
- `linux-32`.
- `linux-64`.
- `win-32`.
- `win-64`.
- `all`.

EXAMPLE: Using the platform specifier `all`:

```
conda convert --platform all ~/anaconda/conda-bld/linux-64/pyinstrument-0.13.1-
py27_0.tar.bz2 -o outputdir/
```

NOTE: Change your path and filename to the path and filename you saved in Building and installing.

## Optional—Using PyPI as the source instead of GitHub

You can use PyPI or another repository instead of GitHub. There is little difference to conda build between building from Git versus building from a tarball on a repository like PyPI. Because the same source is hosted on PyPI and GitHub, you can easily find a script on PyPI instead of GitHub.

Replace this `source` section:

```
git_rev: v0.13.1
git_url: https://github.com/joerick/pyinstrument.git
```

With the following:

```
fn: pyinstrument-0.13.1.tar.gz
md5: e347036acc50720c0903dc2221b2605d
url: https://pypi.python.org/packages/source/p/pyinstrument/pyinstrument-
0.13.1.tar.gz
```

NOTE: The `md5` is found on the PyPI Pyinstrument page.

## Optional—Uploading new packages to Anaconda.org

After converting your files for use on other platforms, you may choose to upload your files to Anaconda.org, formerly known as binstar.org. It only takes a minute to do if you have a free Anaconda.org account.

1. If you have not done so already, open a free Anaconda.org account and record your new user name and password.
2. Run the command `conda install anaconda-client`, and then enter your Anaconda.org username and password.
3. Log into your Anaconda.org account with the command:

   ```
   anaconda login
   ```

4. Upload your package to Anaconda.org:

   ```
   anaconda upload ~/miniconda/conda-bld/linux-64/pyinstrument-0.12-py27_0.tar.bz
   ```

NOTE: Change your path and filename to the path and filename you saved in Building and installing.

TIP: To save time, you can set conda to always upload a successful build to Anaconda.org with the command: `conda config --set anaconda_upload yes`.

## More information

- For more information about all the possible values that can go into the `meta.yaml` file, see Defining metadata (meta.yaml).
- Command reference.