Conda-build recipes — conda-build 3.21.4+10.ge458431a.dirty documentation

https://docs.conda.io/projects/conda-build/en/latest/concepts/recipe.html

None Sun Jun, 20 23:52

To enable building conda packages, install and update conda and conda-build.

Building a conda package requires a recipe. A conda-build recipe is a flat directory that contains the following files:

- `meta.yaml` ---A file that contains all the metadata in the recipe. Only `package/name` and `package/version` are required.
- `build.sh` ---The script that installs the files for the package on macOS and Linux. It is executed using the `bash` command.
- `bld.bat` ---The build script that installs the files for the package on Windows. It is executed using `cmd`.
- `run_test.[py,pl,sh,bat]` ---An optional Python test file, a test script that runs automatically if it is part of the recipe.

- Optional patches that are applied to the source.
- Other resources that are not included in the source and cannot be generated by the build scripts. Examples are icon files, readme files and build notes.

Tip

When you use the conda skeleton command, the first 3 files---`meta.yaml`, `build.sh`, and `bld.bat` ---are automatically generated for you.

Conda-build process¶

Conda-build performs the following steps:

1. Reads the metadata.
2. Downloads the source into a cache.
3. Extracts the source into the source directory.
4. Applies any patches.
5. Re-evaluates the metadata, if source is necessary to fill any metadata values.
6. Creates a build environment and then installs the build dependencies there.
7. Runs the build script. The current working directory is the source directory with environment variables set. The build script installs into the build environment.
8. Performs some necessary post-processing steps, such as shebang and rpath.
9. Creates a conda package containing all the files in the build environment that are new from step 5, along with the necessary conda package metadata.
10. Tests the new conda package if the recipe includes tests:
    1. Deletes the build environment and source directory to ensure that the new conda package does not inadvertantly depend on artifacts not included in the package.
    2. Creates a test environment with the package and its dependencies.
    3. Runs the test scripts.

The conda-recipes repo contains example recipes for many conda packages.

Caution

All recipe files, including `meta.yaml` and build scripts, are included in the final package archive that is distributed to users. Be careful not to put sensitive information such as passwords into recipes where it could be made public.

The `conda skeleton` command can help to make skeleton recipes for common repositories, such as PyPI.

Deep dive¶

Let's take a closer look at how conda-build uses a recipe to create a package.

Templates¶

When you build a conda package, conda-build renders the package by reading a template in the meta.yaml. See Templating with Jinja.

Templates are filled in using your conda-build config, which shows the matrix of things to build against. The `conda build config` determines how many builds it has to do. For example, defining a conda_build_config.yaml of the form and filling it defines a matrix of 4 packages to build:

```
foo:
  - 1.0
  - 2.0
bar:
  - 1.2.0
  - 1.4.0
```

After this, conda-build determines what the outputs will be. For example, if your `conda build config` indicates that you want 2 different versions of Python, conda-build will show you the rendering for each Python version.

Environments¶

To build the package, conda-build will make an environment for you and install all of the build and run dependencies in that environment. Conda-build will indicate where you can successfully build the package. The prefix will take the form:

```
<path to conda>/conda-bld/<package name and string>/h_env_placeholder…
```

Conda-forge downloads your package source and then builds the conda package in the context of the build environment. For example, you may direct it to download from a Git repo or pull down a tarball from another source. See the Source section for more information.

What conda-build puts into a package depends on what you put into the build, host, or run sections. See the Requirements section for more information. Conda-build will use this information to identify dependencies to link to and identify the run requirements for the package. This allows conda-build to understand what is needed to install the package.

Building¶

Once the content is downloaded, conda-build runs the build step. See the Build section for more information. The build step runs a script. It can be one that you provided. See the Script section for more information.

If you do not define the script section, then you can create a build.sh or a bld.bat file to be run.

Prefix replacement¶

When the build environment is created, it is in a placeholder prefix. When the package is all bundled up, the prefix is set to a dummy prefix. When conda is ready to install the package, it rewrites the dummy prefix with the correct one.

Testing¶

Once a package is built, conda-build will test it. To do this, it creates another environment and installs the conda package. The form of this prefix is:

```
<path to conda>/conda-bld/<package name + string>/_test_env_placeholder…
```

At this point, conda-build has all of the info from the meta.yaml about what its runtime dependencies are, so those dependencies are installed as well. This generates a test runner script with a reference to the testing meta.yaml that is created. See the Test section for more information. That file is run for testing.

More information¶

Review Defining metadata (meta.yaml) to see a breakdown of the components of a recipe, including:

- Package name.
- Package version.
- Descriptive metadata.
- Where to obtain source code.
- How to test the package.