

---

# **pip2nix Documentation**

***Release 0.7.0***

**Tomasz Kontusz**

**Jan 02, 2018**



<b>1</b>	<b>Installation</b>	<b>1</b>
<b>2</b>	<b>Basic usage</b>	<b>3</b>
2.1	Ad-hoc python-packages.nix generation . . . . .	3
2.2	Using pip2nix in a project . . . . .	3
<b>3</b>	<b>Configuration file</b>	<b>5</b>
3.1	Location . . . . .	5
3.2	[pip2nix] . . . . .	5
3.3	[pip2nix:package:...] . . . . .	5
<b>4</b>	<b>Examples</b>	<b>7</b>
<b>5</b>	<b>Tips</b>	<b>9</b>
5.1	Missing dependencies . . . . .	9
<b>6</b>	<b>Hacking on pip2nix</b>	<b>11</b>
6.1	Development environment . . . . .	11
6.2	Running tests . . . . .	11
6.3	Changing the dependencies . . . . .	11
6.4	Releasing . . . . .	11
<b>7</b>	<b>Changelog</b>	<b>13</b>
7.1	0.7.0 . . . . .	13
7.2	0.6.0 . . . . .	13
7.3	0.5.0 . . . . .	14
<b>8</b>	<b>Indices and tables</b>	<b>15</b>



# CHAPTER 1

---

## Installation

---

Using *pip2nix* directly out of the git repository can be achieved in the following way:

```
$ git clone https://github.com/johbo/pip2nix
$ nix-env -f pip2nix/release.nix -iA pip2nix.python35
```

Instead of installing into the environment, another convenient way of using it is based on *nix-shell*:

```
$ nix-shell release.nix -A pip2nix.python36
```

Since *pip2nix* is not yet in a mature state, the usage of *nix-shell* is recommended. It does allow to investigate problems on the spot, since it is basically a development environment of *pip2nix*.



### 2.1 Ad-hoc python-packages.nix generation

To generate python-packages.nix for a set of requirements:

```
$ pip2nix generate -r requirements.txt
```

`pip2nix generate` takes the same set of package specifications `pip install` does. It understands `-r`, git links, package specifications, and `-e` (which is just ignored).

### 2.2 Using pip2nix in a project

When packaging a project with `pip2nix` you'll want to make sure it's called the same way every time you bump dependencies. To do that, you can create a `pip2nix.ini` file:

```
[pip2nix]
requirements = -r ./requirements.txt
```

This way you can just run `pip2nix generate` in the project's root. More about the configuration file in [Configuration file](#).

To actually use the generated packages file, you can create a `default.nix` with `pip2nix scaffold`. To work on a project *myProject* you'd use:

```
$ pip2nix scaffold --package myProject
$ cat > pip2nix.ini <<EOF
[pip2nix]
requirements = .
EOF
$ pip2nix generate
$ nix-shell # all the deps should be available
```





### 3.1 Location

pip2nix will search for a configuration file from current working directory up, until it finds either `pip2nix.ini` or `setup.cfg` that contains pip2nix-specific sections.

### 3.2 [pip2nix]

**requirements** comma-separated list of packages to process.

**output** default: `./python-packages.nix`

Where to write the generated packages set.

### 3.3 [pip2nix:package:...]



## CHAPTER 4

---

### Examples

---

The repository <https://github.com/johbo/pip2nix-generated> contains a few examples of using *pip2nix* together with *pip-compile*.



## 5.1 Missing dependencies

Some python packages depend on external libraries or applications to be available already when running `pip2nix` generate. The following example shows a typical error:

```
[nix-shell:~/wo/synapse]$ pip2nix generate -r requirements.txt -c constraints.txt

Collecting pynacl==0.3.0 (from -r requirements.txt (line 47))
  Using cached PyNaCl-0.3.0.tar.gz
  Saved /var/folders/v2/kx2sg5693tb1h84zc2hmjjgr0000gn/T/tmpNYy5RApip2nix/PyNaCl-0.3.
  ↳0.tar.gz
  Complete output from command python setup.py egg_info:
  Package libffi was not found in the pkg-config search path.
  Perhaps you should add the directory containing `libffi.pc'
  to the PKG_CONFIG_PATH environment variable

  [ ... ]

ld: library not found for -lffi
clang-4.0: error: linker command failed with exit code 1 (use -v to see
  ↳invocation)
  Traceback (most recent call last):
    File "<string>", line 1, in <module>
    File "/private/var/folders/v2/kx2sg5693tb1h84zc2hmjjgr0000gn/T/pip-build-KcVPbJ/
  ↳pynacl/setup.py", line 278, in <module>
      "Programming Language :: Python :: 3.4",
    File "/nix/store/hlcj0hzxamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
  ↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/__init__.py",
  ↳line 128, in setup
    File "/nix/store/hlcj0hzxamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
  ↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/__init__.py",
  ↳line 123, in _install_setup_requires
    File "/nix/store/hlcj0hzxamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
  ↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/dist.py", line
  ↳455, in fetch_build_eggs
```

```

File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/pkg_resources/__init__.py",
↳line 866, in resolve
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/pkg_resources/__init__.py",
↳line 1146, in best_match
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/pkg_resources/__init__.py",
↳line 1158, in obtain
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/dist.py", line
↳522, in fetch_build_egg
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/command/easy_
↳install.py", line 673, in easy_install
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/command/easy_
↳install.py", line 699, in install_item
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/command/easy_
↳install.py", line 882, in install_eggs
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/command/easy_
↳install.py", line 1150, in build_and_install
File "/nix/store/hlcj0hzzamapajgrbq3bkx1xlmfcx2f3-python2.7-setuptools-38.2.3/
↳lib/python2.7/site-packages/setuptools-38.2.3-py2.7.egg/setuptools/command/easy_
↳install.py", line 1138, in run_setup
distutils.errors.DistutilsError: Setup script exited with error: command 'clang'
↳failed with exit status 1

-----
Command "python setup.py egg_info" failed with error code 1 in /private/var/folders/
↳v2/kx2sg5693tb1h84zc2hmjjgr0000gn/T/pip-build-KcVPbJ/pynacl/

```

This happens because *pip2nix* depends on the following call to find out about some meta information of the package:

```
python setup.py egg_info
```

Running the command inside of another invocation of *nix-shell* can usually mitigate the trouble. As a one-shot command it looks as follows:

```
nix-shell -p python27Packages.cffi \
  --command 'pip2nix generate -r requirements.txt -c constraints.txt'
```

Entering the sub shell needs a tweak to the environment variable *PATH* at the moment. The next example shows how to run this in two steps:

```
[nix-shell:~/wo/synapse]$ PATH=/bin:$PATH nix-shell -p python27Packages.cffi
[nix-shell:~/wo/synapse]$ pip2nix generate -r requirements.txt -c constraints.txt
```

### 6.1 Development environment

Just running `nix-shell` when in the repository should drop you into a shell with `python2.7` and `pip2nix` wrapper in `$PATH`. To use a different python, pass `--argstr pythonPackages python35Packages` to `nix-shell`.

### 6.2 Running tests

To run tests while in the development environment run `py.test`. It will search for all tests under current directory.

To test all supported platforms, run `nix-build ./release.nix` - this is actually what CI does.

### 6.3 Changing the dependencies

When changing `setup.py` you should also run `pip2nix` to regenerate `python-packages.nix`. If you don't have a working copy around, run `./bootstrap.sh` from top level directory. The script will install `pip2nix` with `pip` into a `virtualenv`, and use that to generate `python-packages.nix`.

### 6.4 Releasing

```
nix-shell ./release-shell.nix
bumpversion dev
rm -rf pip2nix.egg-info/ dist/
nix-shell --pure --run 'python ./setup.py sdist'
twine upload dist/*
bumpversion --no-tag minor
```





### 7.1 0.7.0

- Update template for the file *default.nix* to also ignore the *.hg* folder. This is useful for Mercurial based projects.  
Thanks to Marcin Kuzminzki.
- Fix to quote package and dependency names and improve the readability of the generated output.  
Thanks to Asko Soukka.
- Adjust *release.nix* for better Hydra integration.  
Thanks to Martin Bornhold.
- Mark tests as xfail to avoid trouble when building on NixOS itself. Details can be found here <https://github.com/johbo/pip2nix/issues/35>.
- Use *python36Packages* by default inside of *default.nix*. I noticed that I was specifying it nearly always when working on *pip2nix*. Via *release.nix* we still have all Python versions easily available.
- Fix the attribute name of ZPL licenses, so that it matches the attribute names from *Nixpkgs*.
- Add an example about *setuptools* into the generated layer with manual overrides. This is a useful entry when running into issues around an infinite recursion.
- Update docs with a hint how to run inside of *nix-shell*.
- Update docs with a pointer to examples in *pip2nix-generated*.
- Add section “Tips” to the documentation.

### 7.2 0.6.0

- Change the file *python-packages.nix* into a function.  
To adjust import it like the following:

```
pythonPackagesGenerated = import ./python-packages.nix {  
  inherit pkgs;  
  inherit (pkgs) fetchurl fetchgit;  
};
```

- Add new attribute *pip2nix.python36* into the file *release.nix*.
- Adjust the template for the file *default.nix* to be compatible with the new python packages which are based on the fix point combinator. See <https://github.com/NixOS/nixpkgs/pull/20893> for more details.

## 7.3 0.5.0

- Fixes for git URL support, parsing the output of *nix-prefetch-git* as JSON.
- Use *nix-prefetch-url* to fetch dependencies and get their *sha256* hash.
- Allow version 9 of pip itself for better compatibility with recent nixpkgs versions.
- Update *python-packages.nix* and *release-python-packages.nix*. This should also avoid the warnings due to using *md5* as a hash type.

## CHAPTER 8

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`