Beyond Python packaging with Nix

PyCon PL 2019

Asko Soukka

15.9.2019

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Hello PyCon PL 2019

Asko Soukka

- Python developer since 2002
- Full-time professional since 2008
- GSOC mentor since 2013
- User of Nix and Docker since 2014

- Plone- and Pyramid-projects
- Python microservices
- Robot Framework based RPA

The Problem

Python packaging

The Problem

Python packaging is not enough

Robot Framework RPA

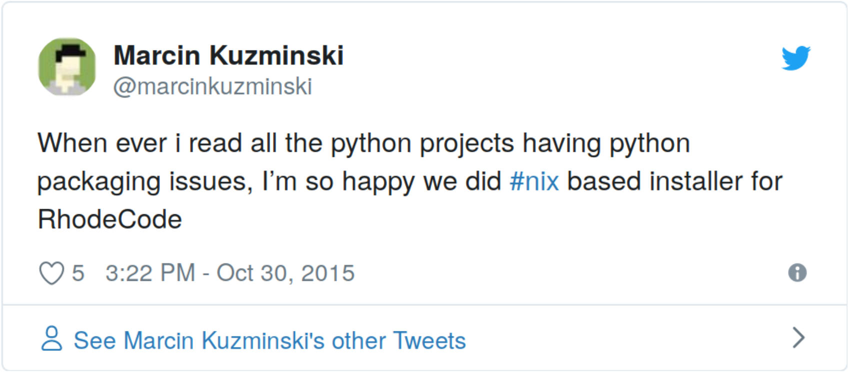
Story

- Fill a PDF form with provided values
- Automate web browser to submit the form

Requirements

- Robot Framework
- PDFtk
- Selenium
- Firefox

RhodeCode SCM



Marcin Kuzminski
@marcinkuzminski

When ever i read all the python projects having python packaging issues, I’m so happy we did #nix based installer for RhodeCode

5 3:22 PM - Oct 30, 2015

See Marcin Kuzminski's other Tweets

Standalone Scripts

Script

```
#! /usr/bin/env nix-shell
#! nix-shell -p "python3.withPackages(ps: with ps; [ exifread ])"
#! nix-shell -i python3
from exifread import process_file
print(process_file(open('image.jpg', 'rb')))
```

Executes

```
$ ./script.py
{'Image ImageDescription': (0x010E) ASCII=...
```

Beyond Python packaging with Nix

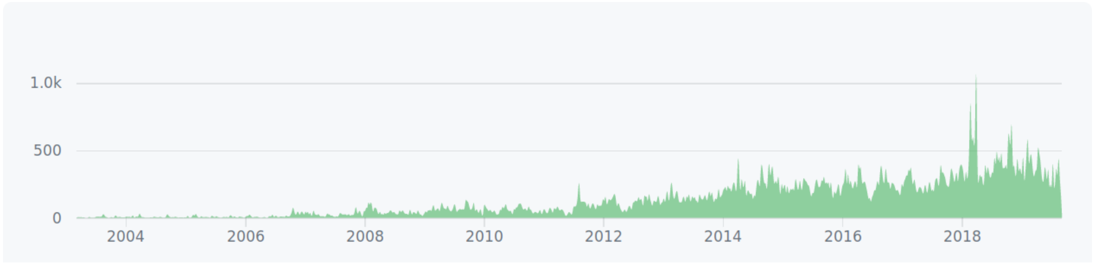
Hello Nix

Nix Ecosystem

Nix is a domain-specific, purely functional, lazily-evaluated language for software configuration and deployment.

- **Nix** – purely functional configuration language
- **Nix** – package manager for packages defined in Nix
- **Nixpkgs** – community Nix packages collection
- **NixOS** – Nixpkgs based GNU/Linux distribution

Nix History



- 2003 Started as a research project by Eelco Dolstra
- 2012 Nix 1.0
- 2013 NixOS 13.10

Nix Today

- Nix 2.3, NixOS 19.03, **2,257 all-time contributors**
- Nixpkgs has **over 40 000 packages**
- GNU/Linux (i686, **x86-64**, **aarch64**), **macOS**
- Supported by NixOS Foundation non-profit
- NixCon 2019 25.–27.10. @ Brno, Czech Republic

- **No official Windows-support yet**
  (Cygwin and WLS should work with small effort)

Nix Vocabulary

Nix **expressions** are instantiated into **derivations**, which are realised into build **outputs**. The collection of build outputs for complete deployment of a software is called **closure**.

- **Expression** – written Nix-function
- **Derivation** – instantiated expression
- **Output** – build result of a derivation
- **Closure** – the goal of complete deployment

Nix Expression

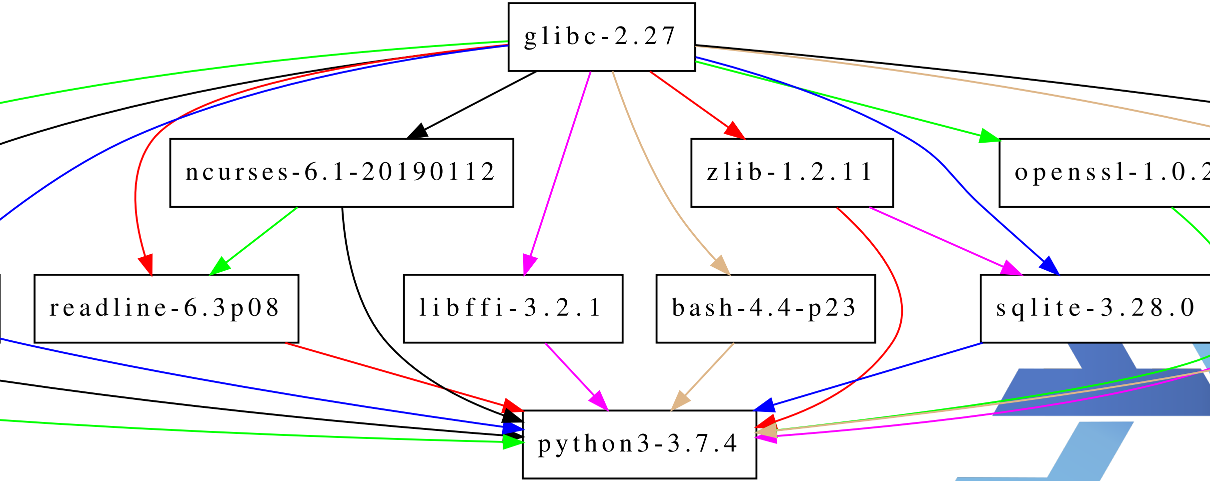
```
{ lib, buildPythonPackage, fetchPypi }:

buildPythonPackage rec {
  pname = "toolz";
  version = "0.10.0";
  src = fetchPypi {
    inherit pname version;
    sha256 = "08fdd5ef7c96480ad11c12d472de21acd...";
  };
  doCheck = false;
}
```

Nix Expression

```
with import <nixpkgs> {};
buildEnv {
  name = "env";
  paths = [
    (python37.withPackages (ps: with ps; [
      (callPackage ./toolz.nix {})
      numpy
    ]))
    graphviz
  ];
}
```

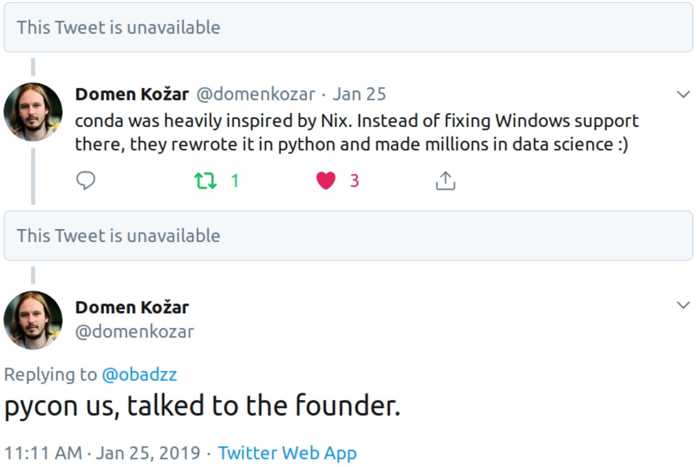
Python Closure



Python Closure

```
/nix/store/96p426c8n8k16j-python3-3.7.4
+---/nix/store/681354n3k44r8z90m35hm8945vsp95h1-glibc-2.27
|   +---/nix/store/681354n3k44r8z90m35hm8945vsp95h1-glibc-2.27 [...]
+---/nix/store/26ani5lvmf4yanr8m7jc1z3irdk16yqg-gdbm-1.18.1
|   +---/nix/store/681354n3k44r8z90m35hm8945vsp95h1-glibc-2.27 [...]
|   +---/nix/store/26ani5lvmf4yanr8m7jc1z3irdk16yqg-gdbm-1.18.1 [...]
+---/nix/store/...-ncurses-6.1-20190112
|   +---/nix/store/...-glibc-2.27 [...]
|   +---/nix/store/...-ncurses-6.1-20190112 [...]
+---/nix/store/...-readline-6.3p08
|   +---/nix/store/...-glibc-2.27 [...]
|   +---/nix/store/...-ncurses-6.1-20190112 [...]
```

Not Unlike Conda



This Tweet is unavailable

Domen Kožar @domenkozar · Jan 25
conda was heavily inspired by Nix. Instead of fixing Windows support there, they rewrote it in python and made millions in data science :)

1 3

This Tweet is unavailable

Domen Kožar
@domenkozar

Replying to @obadzz

pycon us, talked to the founder.

11:11 AM · Jan 25, 2019 · Twitter Web App

Much Batteries

Nix expressions **compose** together unlike anything to allow goodies like:

- TeX Live environment tools
- Container image tools
- Cross-compilation tools
- NixOS image building tools
- KVM based system test tools
- ...

Installing Nix

Single-User Installation

Install

```
$ sudo mkdir /nix
$ sudo chown username /nix
$ sh <(curl https://nixos.org/nix/install) --no-daemon
```

Uninstall

```
$ rm -rf /nix
```

Installation Errors

When this happens

```
error: cloning builder process: Invalid argument
error: unable to start build process
babblebabblebabble...
```

Disable sandboxed builds

```
$ mkdir -p ~/.config/nix
$ echo "sandbox = false" > ~/.config/nix/nix.conf
```

Or ask help

- https://discourse.nixos.org/

nix-env

Nix User Environment

Search available packages

```
$ nix search python37
* nixpkgs.python37 (python3)
  A high-level dynamically-typed programming language
```

Install found packages

```
$ nix-env -iA nixpkgs.python37
installing 'python3-3.7.4'
```

Nix User Environment

List installed packages

```
$ nix-env -q
python3-3.7.4
```

Uninstall packages

```
$ nix-env -e python3
uninstalling 'python3-3.7.4'
```

Nix User Environment

Update configured channels

```
$ nix-channel --update
unpacking channels...
```

Upgrade installed packages

```
$ nix-env -u
upgrading 'python3' to 'python3-3.7.4'
```

Nix User Environment

Install Python with packages

```
$ nix-env -f "<nixpkgs>" -i -E \
  "f: (f {}).python37.withPackages(ps: with ps; [ numpy ])"
```

Rollback to previous state

```
$ nix-env --rollback
switching from generation 3 to 2
```

nix-shell

Ad-hoc Nix-shell

Activate nix-shell

```
$ nix-shell -p python37
[nix-shell:~]$ which python
/nix/store/...--python3-3.7.4/bin/python
```

Use without activation

```
$ nix-shell -p python37 --run "python --version"
Python 3.7.4
```

Ad-hoc Nix-shell

Activate pure nix-shell

```
$ nix-shell --pure -p python37
[nix-shell:~]$ which python
/nix/store/...--python3-3.7.4/bin/python
```

Use purely without activation

```
$ nix-shell --pure -p python37 --run "python --version"
Python 3.7.4
```

Ad-hoc Nix-shell

Activate Nix-shell with Python environment

```
$ nix-shell -p "python37.withPackages(ps: [ ps.numpy ])"
[nix-shell:~]$ python -c "import numpy; print(numpy)"
<module 'numpy' from '/nix/store/...-python3-3.7.4-env/...'>
```

Cleanup downloads and builds

```
$ nix-collect-garbage
```

Nix-shell as Script Interpreter

Nix-shell can be used as a script interpreter to execute a script with the requirements (`-p`) and interpreter (`-i`) defined in the script itself.

```
#! /usr/bin/env nix-shell
#! nix-shell -p "python3.withPackages(ps: with ps; [ scikitlearn ])"
#! nix-shell -i python3
from sklearn import datasets
iris = datasets.load_iris()
digits = datasets.load_digits()
```

Managed Nix-shell

Nix-shell defaults to build the shell from `./shell.nix`.

```
with import <nixpkgs> {};
mkShell {
  buildInputs = [
    (python37.withPackages (ps: with ps; [
      scikitlearn
    ]))
    gnumake
  ];
}
```

Managed Nix-shell

Activate nix-shell

```
$ nix-shell
```

Use without activation

```
$ nix-shell --run "make check"
```

Use at Travis-CI

```
language: nix
script: nix-shell --run "make check"
```

Managed Nix-shell

Ensure **reproducibility by locking** to exact Nixpkgs revision.

```
with import (builtins.fetchTarball {
  url = "https://github.com/NixOS/nixpkgs-channels/archive/....tar.gz";
  sha256 = "0h3s9sn0fzq31hgig5yhcw1pnr7kc7cchixn5b85rgvm70nrwhi6";
}) {};
mkShell {
  ...
}
```

Hash can be precalculated with `nix-prefetch-url --unpack`.

nix-build

Nix-build

Assuming a project environment defined in `./env.nix`

```
with import <nixpkgs> {};
buildEnv {
  name = "env";
  paths = [(python37.withPackages (ps: with ps; [ scikitlearn ]))];
}
```

nix-build can realise the environment into an **output link**.

```
  $ nix-build env.nix -o env
```

Nix-build

Nix-build output link (e.g. `./env`) doubles as a **garbage collector lock**.

```
$ stat -c '%N' ./env
'env' -> '/nix/store/f1yia0prn3a8n16da0lwa4hfdgaw055z-env'

$ stat -c '%N' /nix/var/nix/gcroots/auto/bkd0...
'/nix/var/nix/gcroots/auto/...' -> '/home/.../env'
```

Remove the output link to allow `nix-collect-garbage` to collect the build output (and free some disk space).

Nix dockerTools

Nix dockerTools

Nixpkgs' **dockerTools** provide expressions for creating Docker images.

```
with import <nixpkgs> {};
dockerTools.buildLayeredImage {
  name = "acme";
  tag = "latest";
  contents = [
    (import ./env.nix)
    busybox
  ];
}
```

Nix dockerTools

Image is built with **nix-build** and loaded to use with **Docker**.

```
$ docker load < $(nix-build docker.nix)
Loaded image: acme:latest
```

Calling nix-build results in a new build only if any of the dependencies have changed. Everything gets cached in `/nix/store` as usual with Nix.

Nix dockerTools

```
with import <nixpkgs> {};
dockerTools.buildImage {
  ...
  runAsRoot = ''
    #!${pkgs.stdenv.shell}
    ${pkgs.dockerTools.shadowSetup}
    groupadd --system --gid 65543 nobody
    useradd --system --uid 65543 --gid 65543 -d / -s /sbin/nologin nobody
  '';
  config = { User = "nobody"; };
  keepContentsDirlinks = true;
}
```

Python Packaging in Nix

Packaging Python Application

```
with import <nixpkgs> {};

python3Packages.buildPythonApplication {
  pname = "hello-world";
  version = "1.0.0";
  src = lib.cleanSource ./.;
# nativeBuildInputs = [];
# checkInputs = [];
# buildInputs = [];
# propagatedBuildInputs = [];
}
```

Packaging with Dependencies

```
with import <nixpkgs> {};

let self = rec {
  my_lib = python3Packges.buildPythonPackage { ... };
};
in python3Packages.buildPythonApplication {
  ...
  propagatedBuildInputs = with self; [
    my_lib
  ];
}
```

Generating Python Packages

pypi2nix

- github.com/nix-community/pypi2nix
- No re-use of Python packages in Nixpkgs

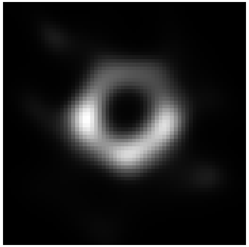
setup.nix

- github.com/nix-community/setup.nix
- Overlays on top of Python packages in Nixpkgs

First image of a black hole: Python DIY

Nix-packaging and Travis-CI example

github.com/datakurre/EHTM87



by following the instructions given by Maciej Wielgus.

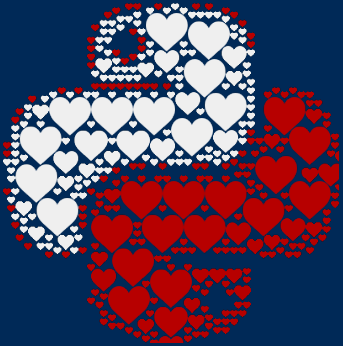
Nix Resources

Official resources

- https://nixos.org/
- https://nixos.org/nix/manual/
- https://nixos.org/nixpkgs/manual/

Community resources

- https://github.com/nix-community/awesome-nix/
- https://discourse.nixos.org/
- #nixos @ Freenode



datakurre.github.io/pyconpl19