Nix Python

Fundamentals and Rationale

Rohit and Amrita Goswami
rog32@hi.is

Presented at CarpentryCon 2020

August 18, 2020

Python Modules

- A `.py` file is a module
- It is standalone if it only imports from the standard library

Pure Python Packages

- A directory with `__init__.py` in it is a package
- Use `pip`

More Distributions

- Distributions have zero or more packages
- Built by `setuptools` with `setup.py`
- Simple source only `.tar.gz`

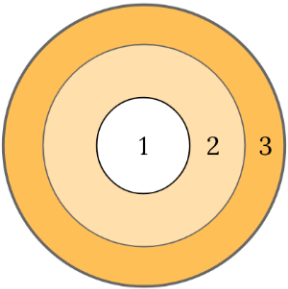
The Python Gradient

From here: `https://www.youtube.com/watch?v=iLVNWfPWAC8`

Packaging for Python **tools** and **libraries**

1. **`.py`** - standalone modules
2. **sdist** - Pure-Python packages
3. **wheel** - Python packages

(With room to spare for static vs. dynamic linking)

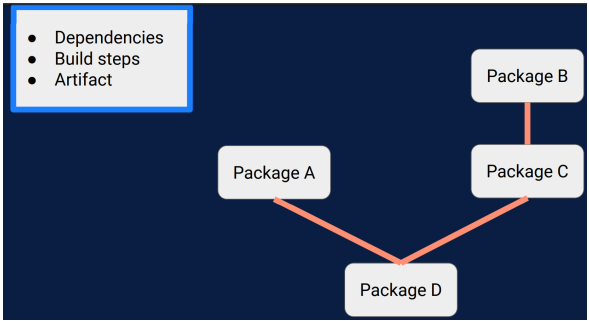


- Libraries and Dev tools are all we get (from PyPI)

Pip Requirements

- Python
- System libraries
- Build tools
  - Wheels don't work for arbitrary distributions

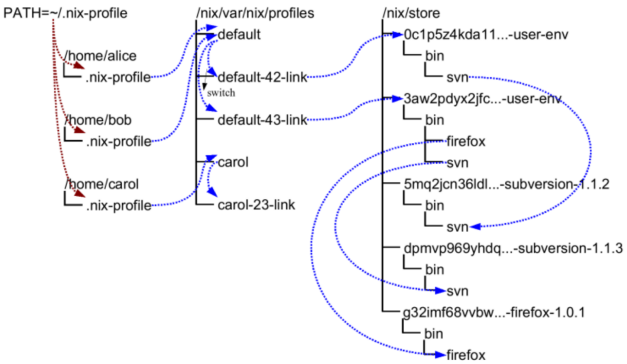
What?



- from `https://brianmckenna.org/files/presentations/rootconf19-nix.pdf`

Nix

Eelco Dolstra, Merijn de Jonge, and Eelco Visser. “Nix: A Safe and Policy-Free System for Software Deployment”. In: (2004), p. 15, Eelco Dolstra, Andres Löh, and Nicolas Pierron. “NixOS: A Purely Functional Linux Distribution”. In: *Journal of Functional Programming* 20.5-6 (Nov. 2010), pp. 577–615. ISSN: 1469-7653, 0956-7968. DOI: 10/dfrgtj



Why?

Protects against self harm

Exposes things taken for granted

Enforces consistency

Reliable Purely functional, no broken dependencies

Reproducible Each package is in isolation

How? store + hash + name + version

Installation (Multi-User)

```
sh <(curl https://nixos.org/nix/install) --daemon
```

- Needs `sudo` but should not be run as root
- Will make build users with IDs between 30001 and 30032 along with a group ID 30000

Nix Python - Trial I

```
nix-shell -p 'python38.withPackages(ps: with ps; [ numpy toolz ])'
```

- Check which `python` is loaded
- Check which modules are present

Shell in a File

```
with import <nixpkgs> {};

let
  pythonEnv = python35.withPackages (ps:
   ↪  [
    ps.numpy
    ps.toolz
  ]);
in mkShell {
  buildInputs = [
    pythonEnv
    which
  ];
}
```

- What tools are we adding?
- What environment are we using?

An Aside into Purity

```
nix-shell --pure --run
↪  'bash'
```

- Why?
- What do we solve with this?

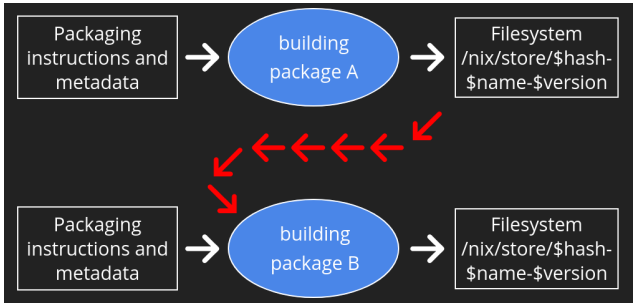


Figure: Stateless builds from `https://slides.com/garbas/mozilla-all-hands-london-2016#/7/0/3`

Nix with Scripts

```
#! /usr/bin/env nix-shell
#! nix-shell -i python3 -p "python3.withPackages(ps: [ps.numpy])"

import numpy

print(numpy.__version__)
```

Friendly Nix

```
nix-env -i nox
nox niv
```

Niv	For pinning packages
Nox	Interactive package management
Lorri	For automatically reloading environments

Existing Projects?

Pip pip2nix
Poetry poetry2nix
Anaconda/Miniconda Why? -_-
Virtualenv ...

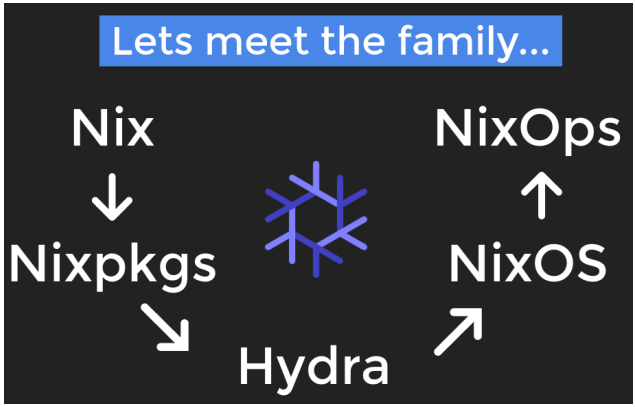
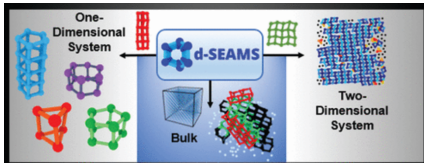


Figure: From `https://slides.com/garbas/mozilla-all-hands-london-2016#/8/0/8`

Future Directions!

- Read up on the Python Guide
- Try Nix Pills
- Roll your own environment
- Make a docker image
- Try a more complex system (d-SEAMS [3])



References I

- Eelco Dolstra, Merijn de Jonge, and Eelco Visser. “Nix: A Safe and Policy-Free System for Software Deployment”. In: (2004), p. 15.
- Eelco Dolstra, Andres Löh, and Nicolas Pierron. “NixOS: A Purely Functional Linux Distribution”. In: *Journal of Functional Programming* 20.5-6 (Nov. 2010), pp. 577–615. ISSN: 1469-7653, 0956-7968. DOI: 10/dfrgtj.
- Rohit Goswami, Amrita Goswami, and Jayant K. Singh. “D-SEAMS: Deferred Structural Elucidation Analysis for Molecular Simulations”. In: *Journal of Chemical Information and Modeling* 60.4 (Apr. 27, 2020), pp. 2169–2177. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.0c00031. arXiv: 1909.09830.

End

Thank you