TWEAG

Reproducible software environments for data analysis with Nix

Simeon Carstens

www.tweag.io

Tweag I/O

Software engineering lab based in Paris with employees all around the world.

We specialize in

- software engineering, with a focus on functional programming
- DevOps, with a focus on **reproducible** software systems and builds
- data science – from the first model to production deployment

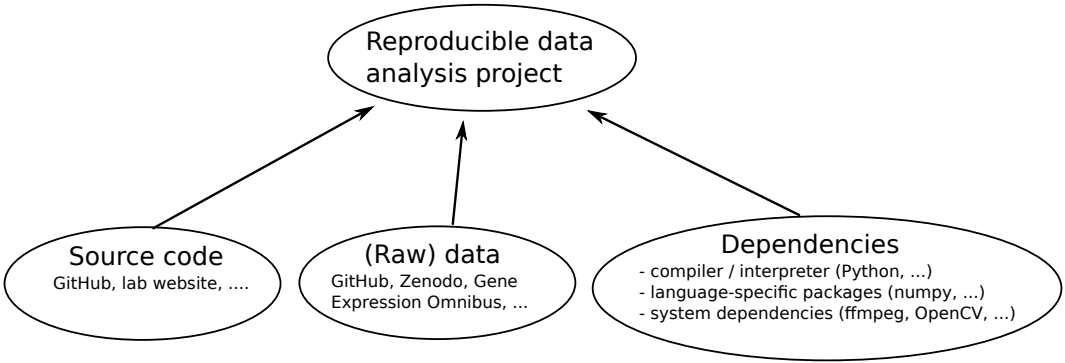
Industries: among others finance, biotech, automotive

Need help with your project? Want to work with us?

www.tweag.io

hello@tweag.io

Reproducing a data analysis



– demo time –

Nix: a declarative package manager

Interpret building a software component as a deterministic function:

$$f(\text{source code}, \text{dep1}, \text{dep2}, ...) = \text{new component}$$

www.nixos.org

Component arguments are again functions:

$$\begin{aligned} \texttt{my\_package} &= \text{build\_my\_package}(\text{source}, \texttt{numpy}) \\ \texttt{numpy} &= \text{build\_numpy}(\texttt{numpy}\ \text{source}, \texttt{BLAS}, \ldots))) \\ &\vdots \end{aligned}$$

Each component is stored under a path with a unique, recursively computed hash:

- → different versions can coexist
- → each component can be uniquely identified

Recursive specification of dependencies

By specifiying a dependency, all its dependencies are automatically specified, too.

– example: two different `pandas` versions –

Nix shells for reproducible research: two for one

Nix shell: a complete software environment with all dependencies

Enhancing developer experience:
For a given project, Alice...

- declares mixed software environment in text file
- develops software in that environment
- drops in and out of environment as needed when changing projects

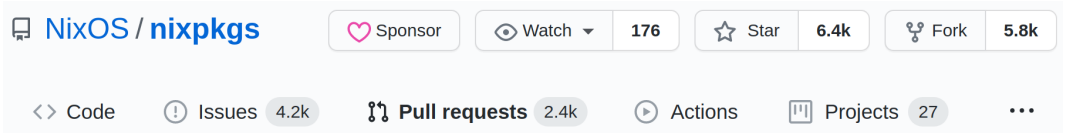
Reproducibility:
Bob...

1. parses environment declaration file with Nix
2. is dropped into the *exactly*[*] same development environment Alice used
3. and runs her program.

[*] down to system libraries and compilers!

– demo time –

Nix: open-source and powered by the community



Nix package collection (`nixpkgs`):

- build instructions for 60,000+ libraries and packages
- maintained by the community
- organized in “channels”, e.g., `nixpkgs-unstable`, `nixpkgs-20.09-darwin`

Contribution is just one pull request away!

Developments based on Nix:

NixOS: Linux distribution

jupyterWith: reproducible Jupyter environments

Alternatives and drawbacks

Language-specific virtual environments (Python: `virtualenv`, R: `renv`, ...):

- project-local
- language depdendencies only
- limited to a single installation of the language package

Docker containers*:

- building images is not reproducible
- not very development-friendly: mounting volumes, graphical output is difficult, IDE not installed...
- doesn't solve problem of intra-project dependency clashes

*Nix can be used to create super-lightweight Docker images!

Drawbacks of Nix

Some weak points:

- steep conceptual learning curve
- configuration language difficult for non-functional programmers
- fully supports only Linux (MacOS mostly supported, Windows not supported)

Ongoing work:

- improving the command line interface
- development of a potential replacement for the Nix language

Language-specific package helpers increase user-friendliness:

- e.g., Python: `poetry2nix` automatically turns a Poetry project into a Nix package

Conclusion

Nix:
package manager which takes into account the **full** dependency tree

Nix shell:
reproducible software environment that can be easily shared with others

Nix community:
maintains and expands Nix package collection; active and helpful:
- forum: https://discourse.nixos.org/
- IRC: #nixos on the freenode network

www.nixos.org