Building pnpm2nix: yet another npm to nix tool?

Adam Hose

October 26, 2018

Outline

Who am I?

- Joined nixos in 2017
- De-facto maintaining nodejs in nixpkgs for the last year or so
- Not a fan of javascript

Goals with the talk

- Introduce the nodejs packaging ecosystem
- Learn about the options to integrate with nix
- Learn about `pnpm`
- Inspire you to implement your own lang2nix in pure nix

Short intro to nodejs modules

- A module is a directory with a `package.json`
  - Entry point is determined by a `main` attribute in `package.json`
- Loading a module

```
const Path = require('path')
```

- What happens when you require() a module?
  - `nodejs` looks in `node_modules`
  - Walks recursively upwards until a `./node_modules` is found
  - Tracks circular imports and does "fake fixpoint"

Flattened/non-flattened

- Non-flattened (pure)

```
node_modules/foo/index.js
node_modules/foo/package.json
node_modules/foo/node_modules/bar/index.js
node_modules/foo/node_modules/bar/package.json
```

- Flattened (impure)

```
node_modules/foo/index.js
node_modules/foo/package.json
node_modules/bar/index.js
node_modules/bar/package.json
node_modules/baz/index.js
node_modules/baz/package.json
```

- Flattening made to solve two problems
  - Windows cant deal well with deep directory structures
  - Disk usage

In the beginning there was darkness

- nodejs released in 2009
  - At first released without a package manager
  - npm entered the scene in 2010
- It's pretty bad in most ways
  - Slow as a dog
  - Keeps redownloading things
  - Packages are a big flat mess
- Relatively recently added integrity checking
- Has `https://github.com/svanderburg/node2nix`
  - Only has code generation

Let there be light?

- Yarn - The hip one
- Tries to solve the performance story by aggresively caching
- Added lock-files from the start
  - Has integrity checking
  - Deterministic dependencies
- Flattens dependencies
- Has great Nix tooling
  `https://github.com/moretea/yarn2nix`
  - Has code generation & runtime `yarn.lock` support
  - Was a great source of inspiration for me how things could be
- *Still* flattens `node_modules`
  - Packages can `require()` unspecified dependencies
- Uses a pretty complex algorithm to flatten `node_modules`
  - Performance not optimal but much better than npm

Let there be light!

- pnpm - the "new" kid on the block
  - Development started in the same month as `yarn`
    - Though for some reason has not reached the same popularity
  - Takes lots of inspiration from `ied` which in turn takes inspiration from `nix` =)
- Main selling points are speed and space efficiency
  - Incredibly fast
- Has a few things in common with nix
  - Centralised store in `~/.pnpm-store`
  - Only `require()` what you directly depend on (with one exception)
- I made my own nix tooling `https://github.com/adisbladis`
  - No code generation, only runtime ingestion

Store structure

- Indirection for purity

```
~/.pnpm-store/2/registry.npmjs.org/yargs/9.0.1/ \
node_modules/yargs/package.json
~/.pnpm-store/2/registry.npmjs.org/yargs/9.0.1/ \
node_modules/yargs/index.js
```

Overrides!

- Some packages do not expect purity and breaks :(
- This happens more rarely than I thought it would
- Graph rewriting to the rescue!

```
module.exports = {
  hooks: {
    readPackage
  }
}

function readPackage(pkg) {
  // ms-rest-azure is missing dependency on request
  if (pkg.name === 'ms-rest-azure') {
    pkg.dependencies['request'] = '^2.83.0'
  }
  return pkg
}
```

Making pnpm2nix

- The hard
  - Specifications are wrong
    - For example bin/bins in `package.json`
  - Circular dependencies are more common than I thought
  - Large number of dependency types
    - Alternative registry
    - Link
    - Git
- The good
  - Pnpm has rich metadata
    - But not always consistent (peerDependencies)
  - Nix makes solving the hard and the ugly relatively easy
- The ugly
  - IFD
    - Amazing stopgap solution
    - Converting yaml to json
    - Matching semantic versions
  - Number of packages quickly becomes very large

Using pnpm2nix - Simplicity hiding complexity

```
with (import <nixpkgs> {});
with (import /path/to/pnpm2nix { inherit pkgs; });

mkPnpmPackage {
  packageJSON = ./package.json;
  shrinkwrapYML = ./shrinkwrap.yaml;
  src = lib.cleanSource ./.;
}
```

Using pnpm2nix for development

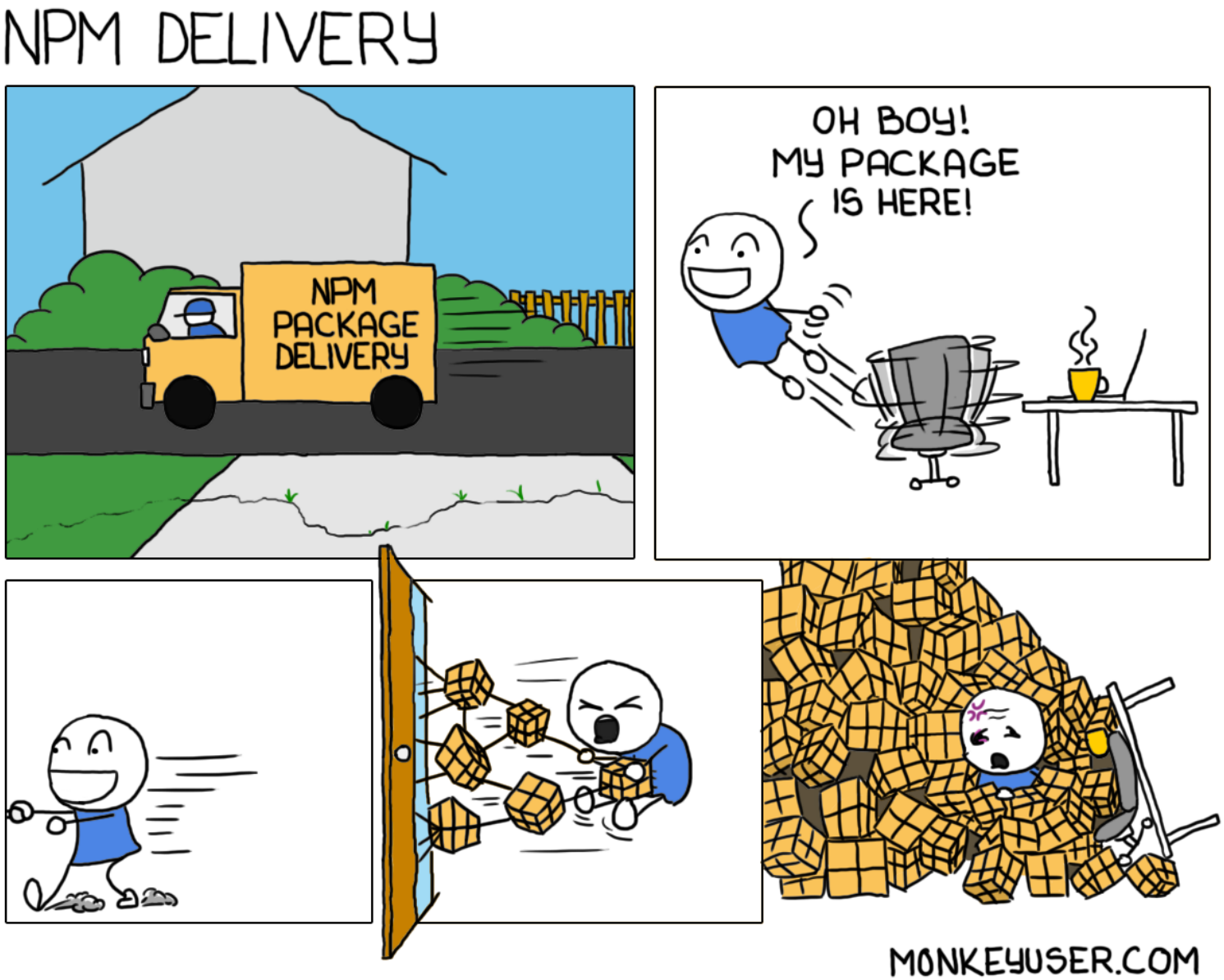
```
with (import <nixpkgs> {});
with (import /path/to/pnpm2nix { inherit pkgs; });

mkShell {
  buildInputs = [
    (mkPnpmEnv (import ./default.nix))
  ];
}
```

Overriding

```
let
  overrides = {
    sharp = (drv: drv.overrideAttrs(oldAttrs: {
      buildInputs = oldAttrs.buildInputs ++ (with pkgs; [
vips glib ]);
      NIX_CFLAGS_COMPILE = [
"-I${pkgs.glib.dev}/include/glib-2.0/"
"-I${pkgs.glib}/lib/glib-2.0/include/"
      ];
      # Force sharp to use the provided vips version by defa
      preBuild = ''
echo 'module.exports.download_vips = \
  (() => { return true })' >> binding.js
      '';
    }));
  };
in mkPnpmPackage { src = ./.; inherit overrides; }
```

Crazy dependencies



State of pnpm2nix

- Mostly correct
- Mostly feature complete
- Has a pretty comprehensive test-suite with real-world tests
- Interfaces stable-ish
- Future
  - Abstract out generic bits
  - Closure size reductions
    - All things native pulls in python - even at runtime
  - Reduce impure fetching
  - Getting rid of IFD

Acknowledgements

- Thanks to my previous employer Enuma Technologies
  - And thanks to the customer using it in production
- Thanks yarn2nix for being an inspiration

Closing words

- The nodejs module system is pretty solid
- The nodejs eco system is insane
- If you have to deal with it - Use pnpm!

Questions?