PURELY FUNCTIONAL PACKAGE MANAGEMENT WITH NIX

Eric Rasmussen / @theerasmas

January 19, 2015

WHO AM I

- OSS contributor
- haskell/python/javascript developer
- works on a hybrid dev/ops team
- prior LUGOD speaker (not about Linux)

PACKAGING: THE GOOD PARTS

- installs in one click/command
- automatic dependency resolution

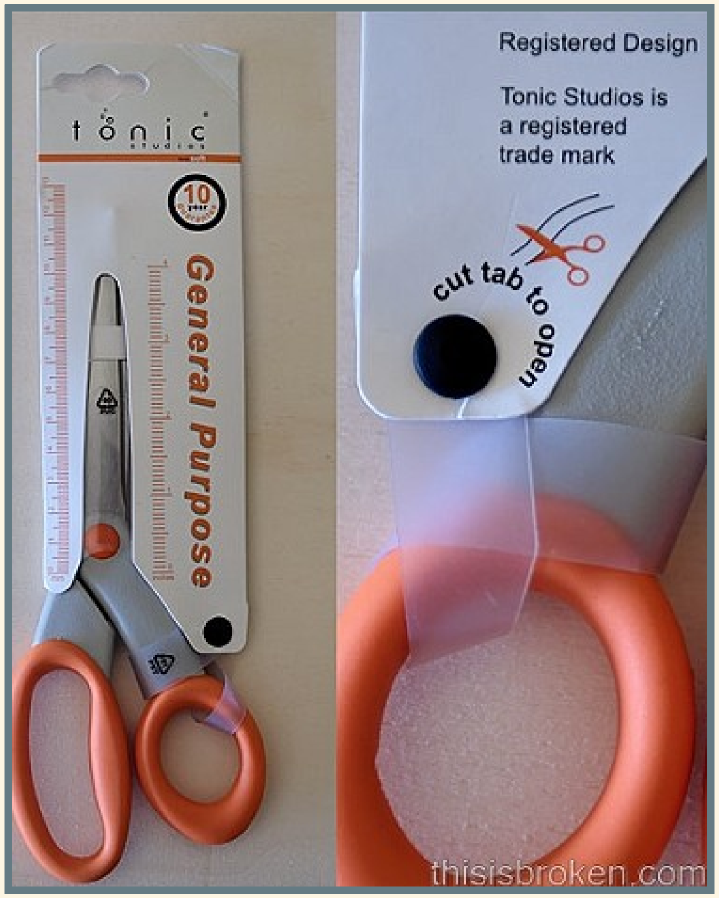
there's just one problem

GOOD NEWS EVERYONE
EVERYTHING IS BROKEN
memegenerator.net

PACKAGING: THE BAD PARTS

- dependency hell
- obscure errors
- high maintenance costs

TL;DR PACKAGING IS HARD



PACKAGE MANAGERS

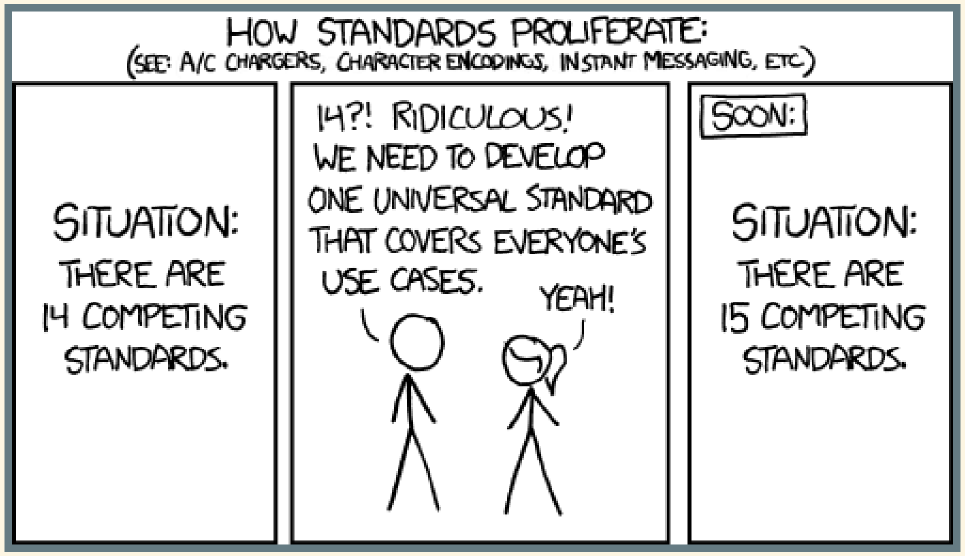
CHOICES (OS)

- dpkg/apt
- rpm/yum
- pacman
- homebrew
- mac ports
- various app stores

CHOICES (PL)

- easy_install
- pip
- go get
- maven
- npm
- rubygems
- sbt
- cabal
- package.el
- cpan
- pear
- pecl

DO WE REALLY NEED ANOTHER?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

(via http://xkcd.com/927/)

INTRODUCING NIX

- new model for package management
- introduced in Eelco Dolstra's PhD Thesis (2006)
- based on functional programming principles

WHAT NIX OFFERS

- minimal and portable
- declarative
- reproducible builds
- deterministic

FUNCTIONAL PURITY

- Function takes inputs and produces output
- Ex: Addition takes two numbers and makes a new one
- 40 + 2 = 42

FUNCTIONAL PURITY

- Most programming languages don't enforce this!

```
40 + 2 =
       = new log file with debug output
       = database calls
       = HTTP service calls...
       = 42, maybe?
```

NIX PACKAGES ARE PURE

- Input: other packages, configuration options
- Output: a package

EXAMPLE: GCC

```
/nix/store/r8vvq9kq18pz08v249h8my6r9vs7s0n3-gcc-4.
```

- inside the prefix: bin, lib, share, ... directories
- r8vvq9kq18pz08v249h8my6r9vs7s0n3 is a hash of function inputs

PURITY IN NIX

- no global install directories (/usr, /bin)
- /nix/store is immutable (mounted read-only)
- nix expressions cannot write to files

IMPURITY IN NIX

- packages can make network calls (curl/git clone/etc)
- race conditions in parallel builds
- stdenv on OS X depends on globals

EXAMPLE: NGINX

```
{ stdenv, fetchurl, fetchgit, openssl, zlib, pcre, libxml2, libxslt, expat }:

stdenv.mkDerivation rec {
  name = "nginx-${version}";
  version = "1.4.4";

  src = fetchurl {
    url = "http://nginx.org/download/nginx-${version}.tar.gz";
    sha256 = "1f82845mpgmhvm151fhn2cnqjggw9w7cvsqbva9rb320wmc9m63w";
  };
  buildInputs = [ openssl zlib pcre libxml2 libxslt ];
  configureFlags = [ "--with-http_spdy_module" ];
  postInstall = "mv $out/sbin $out/bin";

  meta = with stdenv.lib; {
    description = "A reverse proxy and lightweight webserver";
    maintainers = [ maintainers.iElectric ];
    platforms = platforms.all;
    license = licenses.bsd2;
  };
```

BINARY PACKAGES

- binary packages are built in hydra build farms
- purity lets us substitute pre-built packages based on the hash
- major speedups when installing on common platforms

ADD NIX TO YOUR WORKFLOW IN 2 DAYS

- Day 1: install a package
- Day 2: myEnvFun

DAY 1: INSTALLATION

```
$ curl -L http://git.io/nix-install.sh | bash
$ source ~/.nix-profile/etc/profile.d/nix.sh
$ nix-env -i nginx
```

DAY 2: CONFIG.NIX / MYENVFUN

- Note: fun is for "functional" (having fun is optional)

```
# ~/.nixpkgs/config.nix
{
   packageOverrides = pkgs : with pkgs; {
     pyred2 = pkgs.myEnvFun {
         name = "pyred2";
         buildInputs = [ python27Full redis ];
     };

     pyred3 = pkgs.myEnvFun {
         name = "pyred3";
         buildInputs = [ python3 redis ];
     };

   };
}
```

Using myEnvFun

```
$ nix-env -i env-pyred2
$ load-env-pyred2
env-pyred2 loaded

pyred2:[eric@nixos:~]$ python
python             python2.7          python2-config
python2            python2.7-config   python-config
```

EASY TO UNINSTALL IF NEEDED

```
$ rm -rf /nix
$ rm -rf ~/nix-profile/
```

INTERMISSION

NIXOS

- Declarative config at the system level
- Nix as package manager
- Nix expressions to configure the OS

NIXOS

- stateless config management
- NixOS modules for services

CONFIGURATION.NIX

```nix
{ config, pkgs, ... }: with pkgs;
  {
    networking.firewall.allowedTCPPorts = [ 8000 ];

    services.postgresql = {
      enable = true;
      package = pkgs.postgresql93;
      authentication = pkgs.lib.mkOverride 10 ''
          local postgres root ident
          host myuser myuser 127.0.0.1/32 password
          local all all ident
      '';
      initialScript = "bootstrap_or_something.sql";
    };

    environment.systemPackages = [ emacs24-nox git tmux ghc.ghc783 ];

  }
}
```

ENFORCING GOOD HABITS

- Harder to make one-off hacks
- Config and build changes must be codified
- Example: add hosts to /etc/hosts

```
 # configuration.nix
# will extend /etc/hosts
networking.extraHosts = ''
  some_ip some_host
  some_ip2 some_host2
'';
```

IS NIXOS FOR ME?

- maybe!
- requires learning nix/writing packages
- great IRC support but few docs/tutorials

IS NIX FOR ME?

- try it out!
- won't interfere with existing packages

REFERENCES

- NixOS.org
- Nix Package Manager Manual
- NixOS Manual
- Domen Kožar's 2014 Fosdem talk