Nix loves Haskell

Peter Simons <simons@cryp.to>

2015-05-21

Road-map of this presentation

- Install Haskell packages from Hackage
- Install one (or many) Haskell compiler(s)
- Create a Haskell development environment
  - permanent environments with `ghcWithPackages`
  - ad-hoc environments for `nix-shell`
- Integrate your own (non-public) packages into Nix
- Take advantage of the central build farm
- Pitfalls to avoid

Install Haskell packages from Hackage

Lookups based on package names like

```
$ nix-env -i cabal-install
```

don't work! You have to use attribute paths:

```
$ nix-env -iA haskellPackages.cabal-install
```

To list all Haskell packages, run:

```
$ nix-env -qaP -A haskellPackages
```

Install Haskell packages from Hackage

There is one Haskell package set per compiler:

```
$ nix-env -qaP -A haskellPackages
$ nix-env -qaP -A haskell.packages.ghc6123
$ nix-env -qaP -A haskell.packages.ghc763
$ nix-env -qaP -A haskell.packages.ghc7101
```

The default `haskellPackages` currently refers to `haskell.packages.ghc7101`.

Install a Haskell compiler

```
$ nix-env -iA haskell.compiler.ghc784

$ nix-env -qaP -A haskell.compiler
haskell.compiler.ghc6104          ghc-6.10.4
haskell.compiler.ghc6123          ghc-6.12.3
haskell.compiler.ghc704           ghc-7.0.4
haskell.compiler.ghc722           ghc-7.2.2
haskell.compiler.ghc742           ghc-7.4.2
haskell.compiler.ghc763           ghc-7.6.3
haskell.compiler.ghc784           ghc-7.8.4
haskell.compiler.ghc7101          ghc-7.10.1
haskell.compiler.ghcHEAD          ghc-7.11.20150402
haskell.compiler.ghcjs            ghcjs-0.1.0
haskell.compiler.jhc              jhc-0.8.2
haskell.compiler.uhc              uhc-1.1.9.0
```

Install a Haskell compiler

For every compiler version XYZ, the attributes

```
haskell.compiler.ghcXYC
```

and

```
haskell.packages.ghcXYC.ghc
```

are synonymous.

Install more than one Haskell compiler (temporary)

```
$ nix-shell -p haskell.compiler.ghc7101
[nix-shell:~]$ ghc --numeric-version
7.10.1
[nix-shell:~]$ exit

$ nix-shell -p haskell.compiler.ghc784 \
            --command "cabal configure"
$ cabal build
```

Install several Haskell compilers (transient)

```
$ nix-env -p ~/ghc-7.6.3 -iA haskell.compiler.ghc763
$ nix-env -p ~/ghc-7.8.4 -iA haskell.compiler.ghc784
[...]

$ export PATH=$HOME/ghc-7.6.3/bin:$PATH
$ ghc --numeric-version
7.6.3

$ cabal configure --with-compiler=$HOME/ghc-7.6.3/bin/ghc
```

Install several Haskell compilers (permanent)

```
$ PROFILE_DIR=/nix/var/nix/profiles/per-user/$USER
$ nix-env -p $PROFILE_DIR/ghc-7.6.3 -iA ...
```

On NixOS, `/etc/profile` defines `$NIX_USER_PROFILE_DIR` automatically.

Create a Haskell development environment

Edit the file `~/.nixpkgs/config.nix`:

```
{
  packageOverrides = super: let self = super.pkgs; in
  {
    myHaskellEnv =
      self.haskell.packages.ghc7101.ghcWithPackages
        (haskellPackages: with haskellPackages; [
          arrows async cabal-install case-insensitive
          cgi criterion hspec HStringTemplate
        ]);
  };
}
```

Now installl with `nix-env -iA myHaskellEnv`.

Create a Haskell development environment

The generated ghc program is a wrapper script that re-directs the real ghc to use a "libdir" with all the specified packages installed:

```
#! /nix/store/xlxjcjbnbwnz...-bash-4.3-p33/bin/bash -e
realGhc=/nix/store/zzhasddj77xhwdban95...-ghc-7.10.1
ghcEnv=/nix/store/cgddwzz9hkdgprvbymph...-ghc-7.10.1
export NIX_GHC=$ghcEnv/bin/ghc
export NIX_GHCPKG=$ghcEnv/bin/ghc-pkg
export NIX_GHC_DOCDIR=$ghcEnv/share/doc/ghc/html
export NIX_GHC_LIBDIR=$ghcEnv/lib/ghc-7.10.1
exec $realGhc/bin/ghc "-B$NIX_GHC_LIBDIR" \
                      "${extraFlagsArray[@]}" "$@"
```

Create a Haskell development environment

Define the same environment variables in your `~/.bashrc`:

```
NIX_GHC="$HOME/.nix-profile/bin/ghc"
ghcVersion=$($NIX_GHC --numeric-version)
NIX_GHCPKG="$HOME/.nix-profile/bin/ghc-pkg"
NIX_GHC_DOCDIR="$HOME/.nix-profile/share/doc/ghc/html"
NIX_GHC_LIBDIR="$HOME/.nix-profile/lib/ghc-$ghcVersion"
export NIX_GHC NIX_GHCPKG NIX_GHC_DOCDIR NIX_GHC_LIBDIR
unset ghcVersion
```

Or:

```
if [ -e ~/.nix-profile/bin/ghc ]; then
  eval $(grep export ~/.nix-profile/bin/ghc)
fi
```

Create a temporary Haskell development environment

Save as `shell.nix`:

```
with (import <nixpkgs> {}).pkgs;
let
  ghc = haskell.packages.ghc7101.ghcWithPackages
          (pkgs: with pkgs; [ aeson lens monad-par ]);
in
stdenv.mkDerivation {
  name = "my-haskell-env-0";
  buildInputs = [ ghc ];
  shellHook = "eval $(grep export ${ghc}/bin/ghc)";
}
```

Now run `nix-shell`, or even `nix-shell --pure`.

Create a temporary Haskell development environment

To parameterize the compiler version, edit the file as follows:

```
{ compiler ? "ghc7101" }:

with (import <nixpkgs> {}).pkgs;
let
  ghc = haskell.packages.${compiler}.ghcWithPackages
          (pkgs: with pkgs; [ aeson lens monad-par ]);
in
[...]
```

Now run "`nix-shell --argstr compiler ghc784`" to select a different compiler.

Create a temporary Haskell development environment

Every Haskell package has an `env` attribute that provides a temporary shell environment in which that package can be built:

```
$ cabal get lens && cd lens-4.10
Downloading lens-4.10...
Unpacking to lens-4.10/

$ nix-shell "<nixpkgs>" -A haskellPackages.lens.env
[nix-shell:/tmp/lens-4.10]$ cabal configure
Resolving dependencies...
Configuring lens-4.10...
[...]
```

Create a development environment for your own packages

```
$ nix-env -iA cabal2nix

$ cabal2nix --shell . >shell.nix
$ nix-shell --command "cabal configure"

$ cabal build
[...]
```

Create a development environment for your own packages

```
$ cabal2nix --shell .
with (import <nixpkgs> {}).pkgs;
let pkg = haskellPackages.callPackage
            ({ mkDerivation, base, stdenv, transformers }:
             mkDerivation {
               pname = "mtl";
               version = "2.2.1";
               src = ./.;
               buildDepends = [ base transformers ];
               homepage = "http://github.com/ekmett/mtl";
               license = stdenv.lib.licenses.bsd3;
             }) {};
in
  pkg.env
```

Create builds for your own packages

```
$ cd ~/src/foo
$ cabal2nix . > default.nix
```

Now edit `~/.nixpkgs/config.nix`:

```
{
  packageOverrides = super: let self = super.pkgs; in
  {
    foo = self.haskellPackages.callPackage
            ../src/foo {};
  };
}
```

Build it by running "`nix-env -iA foo`".

Create builds for your own packages

Use this `~/.nixpkgs/config.nix` file to register the build inside of `haskellPackages`, so that other libraries may depend on it:

```
{
  packageOverrides = super: let self = super.pkgs; in
  {
    haskellPackages = super.haskellPackages.override {
      overrides = self: super: {
        foo = self.callPackage ../src/foo {};
        bar = self.callPackage ../src/bar {};
      };
    };
  };
}
```

How to download binary packages

NixOS users add this setting to `/etc/nixos/configuration.nix`:

```
nix.trustedBinaryCaches = [
  http://hydra.cryp.to
  http://hydra.nixos.org
];
```

Everyone else adds this setting to `/etc/nix/nix.conf`:

```
trusted-binary-caches = http://hydra.nixos.org \
  http://hydra.cryp.to
```

Now run `nix-env`, `nix-build`, and `nix-shell` with this option:

```
--option extra-binary-caches http://hydra.nixos.org
```

GHC's infamous non-deterministic library ID bug

GHC and distributed build farms don't get along well:

```
https://ghc.haskell.org/trac/ghc/ticket/4012
https://github.com/NixOS/nixpkgs/issues/7792
```

When you see an error like this one

```
package foo-0.7.1.0 is broken due to missing package
text-1.2.0.4-98506efb1b9ada233bb5c2b2db516d91
```

then you have to garbage collect `foo` and all its dependents, and re-install from scratch.

Where to get help . . .

Haskell-specific resources:

- https://github.com/NixOS/nixpkgs/issues/4941
- A Journey into the Haskell NG infrastructure
  - Part 1: http://lists.science.uu.nl/pipermail/nix-dev/2015-January/015591.html
  - Part 2: http://lists.science.uu.nl/pipermail/nix-dev/2015-January/015608.html
  - Part 3: http://lists.science.uu.nl/pipermail/nix-dev/2015-April/016912.html
- Oliver Charles Wiki: http://wiki.ocharles.org.uk/Nix

General Nix resources:

- `https://nixos.org/`
- `nix-dev@lists.science.uu.nl`
- `#nixos` auf `irc.freenode.org`