Nuenv: an experimental derivation builder for Nix

Luc Perkins, Determinate Systems

@lucperkins

Derivations ƛ

- Nix the language: strings, Booleans, integers, attribute sets, `builtins`, functions, and then…
- A magical leap 🦄🪄🧞🎠
  - Derivations: **build instructions that convert Nix data into filesystem state** (realisation 🇬🇧)
  - ```
    derivation {
      name = "hello-world";
      builder = "/bin/bash";
      system = "aarch64-darwin";
      args = [ "-c" "mkdir $out/share && echo 'Hello world' > $out/share/hello.txt" ];
    }
    ```
  - "Nested" derivations
  - The `builder` can be *lots* of things
    ```

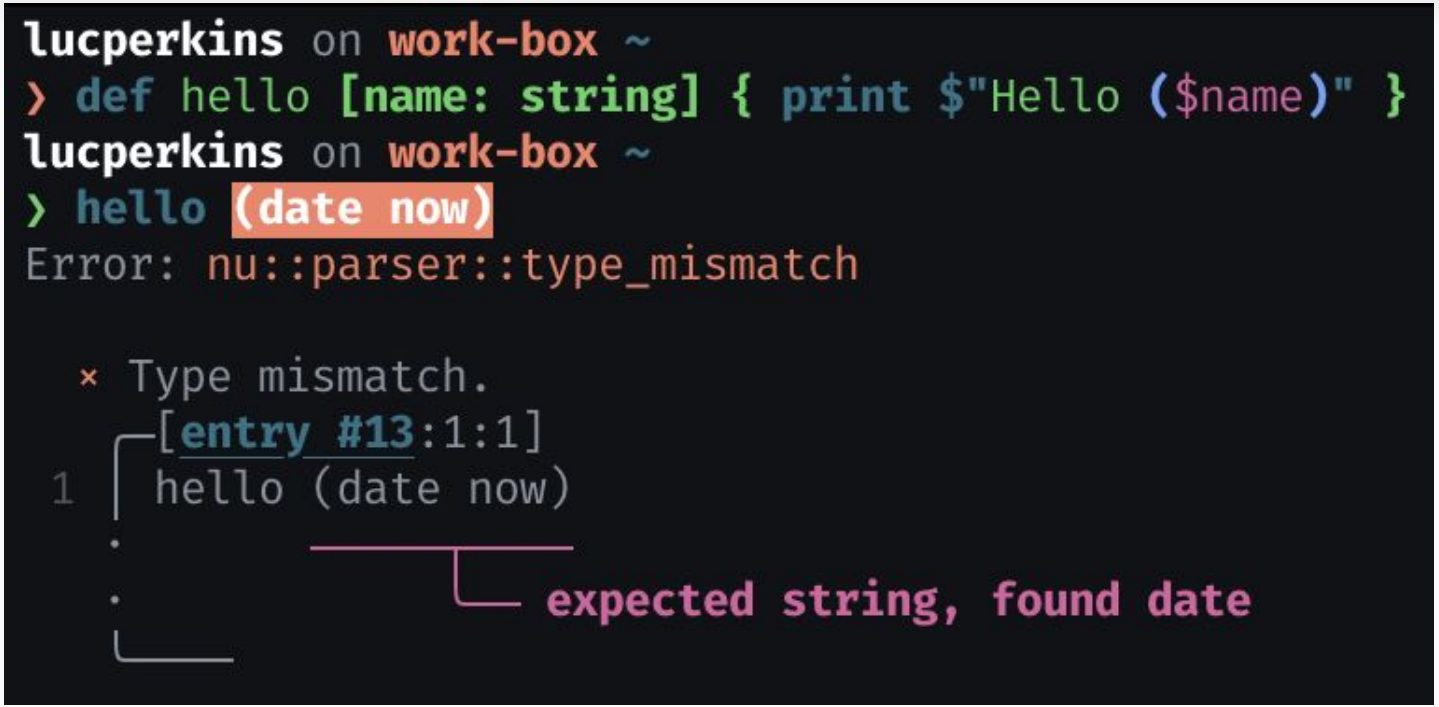
stdenv

- `stdenv.mkDerivation`
  - Wrapper around `derivation`
  - Also wrappable
    - `buildGoModule`
    - `buildRustPackage`
- Bash
  - The good
    - Known quantity
    - Broad-based knowledge
  - The less good
    - Not terribly expressive
    - Needs coreutils
    - Footguns

Enter Nushell ⬇️

- A (very exciting) alternative shell written in Rust
- https://nushell.sh
- Highly expressive; bordering on a programming language
- Features
  - Built-in JSON, YAML, TOML, CSV, SQLite, etc.
  - Functional programming constructs (map, sort, reduce, etc.)
    - `[1 3 5 7 10] | filter { |i| $i mod 2 == 0 }`
  - Type safety plus new types: durations, records, ranges, tables, etc.
  - No more coreutils

Type safety 


lucperkins on work-box ~
❯ def hello [name: string] { print $"Hello ($name)" }
lucperkins on work-box ~
❯ hello (date now)
Error: nu::parser::type_mismatch
× Type mismatch.
[entry #13:1:1]
1 hello (date now)
expected string, found date

Functions as CLI tools 🔨

```
# Substitute all instances of <replace> in <file> with <with> and
# output the resulting string to <out>.
def substitute [
  file: path, # The target file
  out: path, # The output file
  --replace (-r): string, # The string to replace in <file>
  --with (-w): string, # The replacement for <replace>
] {
  ensureFileExists $file
  # Store the initial file contents in a variable
  let orig = (open $file)
  # Delete the original file
  rm $file
  # Build a new string with the substitution applied
  let s = ($orig | str replace -a $replace $with)
  # Write the new string to the target file
  $s | save $out
}
```

Built-in docs

```
~/dts/nuenv | substitute
substitute          Substitute all instances of <replace> in <file> with <with> and output the resulting string to <out>.
substituteInPlace   Substitute all instances of the string <replace> in <file> with the string <with>.
```

```
~/dts/nuenv > substitute --help
Substitute all instances of <replace> in <file> with <with> and output the resulting string to
<out>.

Usage:
  > substitute {flags} <file> <out>

Flags:
  -r, --replace <String> - The string to replace in <file>
  -w, --with <String> - The replacement for <replace>
  -h, --help - Display the help message for this command

Parameters:
  file <path>: The target file
  out <path>: The output file
```

Nuenv 🌴

- Nix builder
  - s/Bash/Nushell
  - Nushell's bona fides carried over into Nix
  - Flake with two function outputs:
    - **nuenv.mkDerivation**
    - **nuenv.writeScriptBin**
- Read more
  - https://github.com/DeterminateSystems/nuenv
  - https://flakehub.com/flake/DeterminateSystems/nuenv
- Status: very experimental
- Future: um, none?

Nuenv in action 🎽

```
hello = pkgs.nuenv.mkDerivation {
  name = "hello-nixcon";
  packages = [ pkgs.hello ];
  src = self;
  build = ''
    mkdir $"($env.out)/share"
    let msg = hello --greeting ($env.MESSAGE)
    $msg | save $"($env.out)/share/hello.txt"
  '';
  MESSAGE = ''
    NixCon! Servus, grüezi, und hallöchen
  '';
}
```

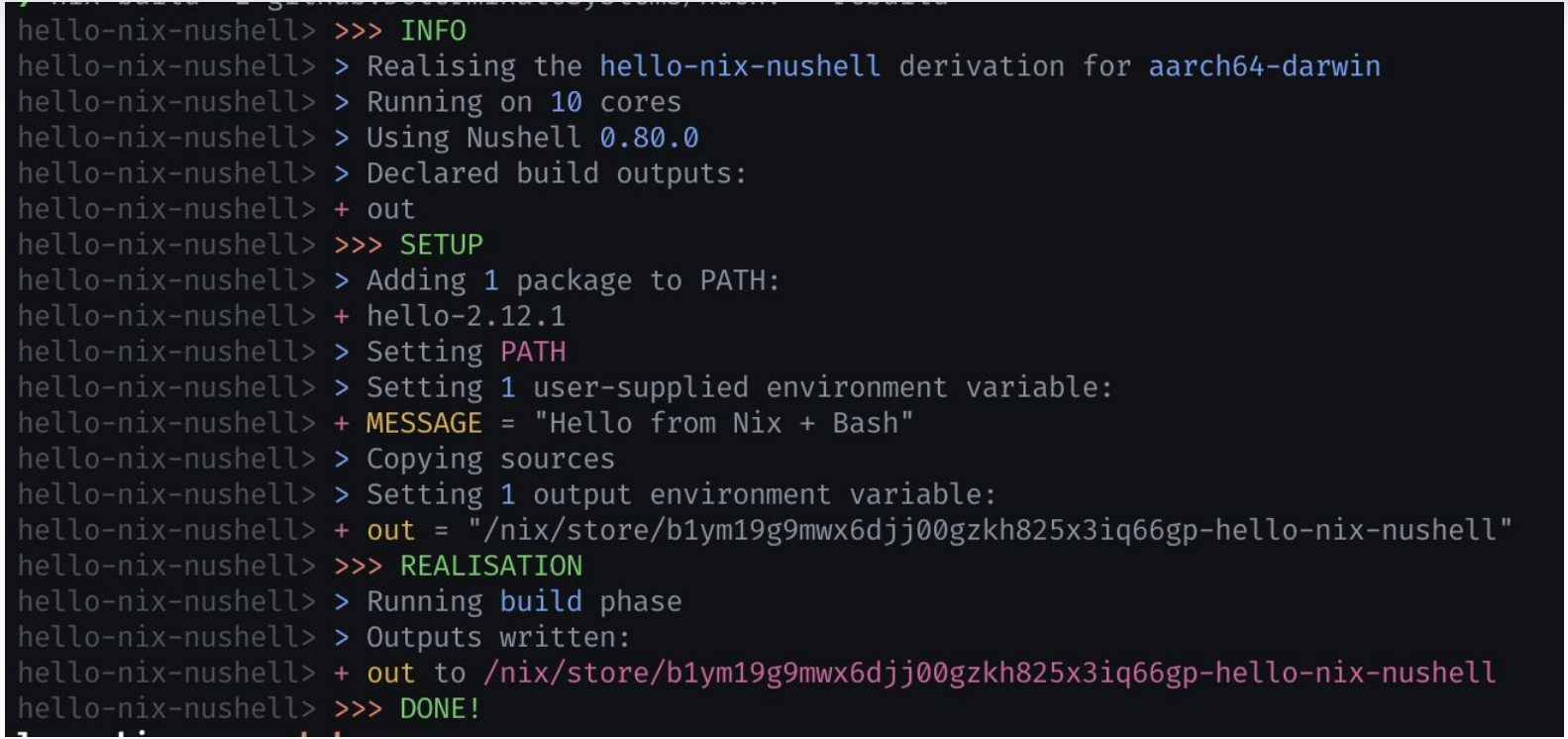
Try it out 👋


```
# GitHub
nix build -L "github:DeterminateSystems/nuenv#nixcon"

# FlakeHub
nix build -L \
  "https://flakehub.com/f/DeterminateSystems/nuenv/0.1.*.tar.gz#nixcon"

# And then
cat ./result/share/hello.txt
```

Output ➡️


hello-nix-nushell> >>> INFO
hello-nix-nushell> > Realising the hello-nix-nushell derivation for aarch64-darwin
hello-nix-nushell> > Running on 10 cores
hello-nix-nushell> > Using Nushell 0.80.0
hello-nix-nushell> > Declared build outputs:
hello-nix-nushell> + out
hello-nix-nushell> >>> SETUP
hello-nix-nushell> > Adding 1 package to PATH:
hello-nix-nushell> + hello-2.12.1
hello-nix-nushell> > Setting PATH
hello-nix-nushell> > Setting 1 user-supplied environment variable:
hello-nix-nushell> + MESSAGE = "Hello from Nix + Bash"
hello-nix-nushell> > Copying sources
hello-nix-nushell> > Setting 1 output environment variable:
hello-nix-nushell> + out = "/nix/store/b1ym19g9mwx6djj00gzkh825x3iq66gp-hello-nix-nushell"
hello-nix-nushell> >>> REALISATION
hello-nix-nushell> > Running build phase
hello-nix-nushell> > Outputs written:
hello-nix-nushell> + out to /nix/store/b1ym19g9mwx6djj00gzkh825x3iq66gp-hello-nix-nushell
hello-nix-nushell> >>> DONE!

Scripts 

```
# GitHub
nix run "github:DeterminateSystems/nuenv#run-me"

# FlakeHub
nix run "https://flakehub.com/f/DeterminateSystems/nuenv/0.1.*.tar.gz#run-me"
```

The sandbox sandbox 🏝️

```
# GitHub
nix develop "github:DeterminateSystems/nuenv#nuenv"

# FlakeHub
nix develop "https://flakehub.com/f/DeterminateSystems/nuenv/0.1.*.tar.gz#nuenv"

# Run
nuenvCommands
# substitute and substituteInPlace

# Example
relativePath --help
```

How I built it 🏛️

```
derivation {
  inherit name packages src system; # From function args
  builder = "${pkgs.nushell}/bin/nu";
  args = [ ./bootstrap.nu ];
  __structuredAttrs = true; # JSON!
  __nu_builder = ../nuenv/builder.nu; # builder script
  __nu_env = [ ../nuenv/env.nu ]; # env file with helpers for builder.nu
}
```

Bootstrap script 🥾

```
let attrs = (open $env.NIX_ATTRS_JSON_FILE)
let-env PATH = ($attrs.__nu_builder | parse "{storePath}/nu" | get storePath.0)
nu --commands (open $attrs.__nu_builder)
```

The builder 🏗️

```
let here = $env.NIX_BUILD_TOP
let attrs = (open $attrsJsonFile)

let packagesPath = (
  $attrs.packages | each { |pkg| $"($pkg)/bin" } | str join (char esep)
)

let-env PATH = $packagesPath

for src in $attrs.src { cp -r $src $here }

for attr in $attrs.extraAttrs {
  let-env $attr.key = $attr.value
}
```

Run! 🐎


```
do --capture-errors {
  nu --commands $attrs.build
  let exitCode = $env.LAST_EXIT_CODE
  if $exitCode != 0 {
    exit $exitCode
  }
}
```

Criticisms ❌

- Size 😬
  - 25+ MB
  - **But**: cost is up front
- Too good 😩
  - Super expressive ➡ new dangers
  - **But**: super expressive
- Brand new thing to learn 😖
  - Oh fun, a new set of mountains to climb
  - **But**: was Nix worth it?

General lessons 📖

- 💯 learning experience
- Wide open spaces 🌅
  - Real alternatives are possible
  - Replace stdenv vs. localized change
- Nix’s open heart ❤️

Happy NixCon! ❄️❄️❄️❄️