A Tale of Nix and Nickel

YOW! Lambda Jam

Yann Hamdaoui

May 5, 2021

TWEAG

Introduction

A cautionnary tale

Once upon a time...

Reproducibility

works on my machine ¯_(ツ)_/¯

Reproducibility

works on my machine ¯_(ツ)_/¯

Reproducibility

Reproducibility

works on my machine ¯_(ツ)_/¯

Reproducibility

1. Concrete and widespread

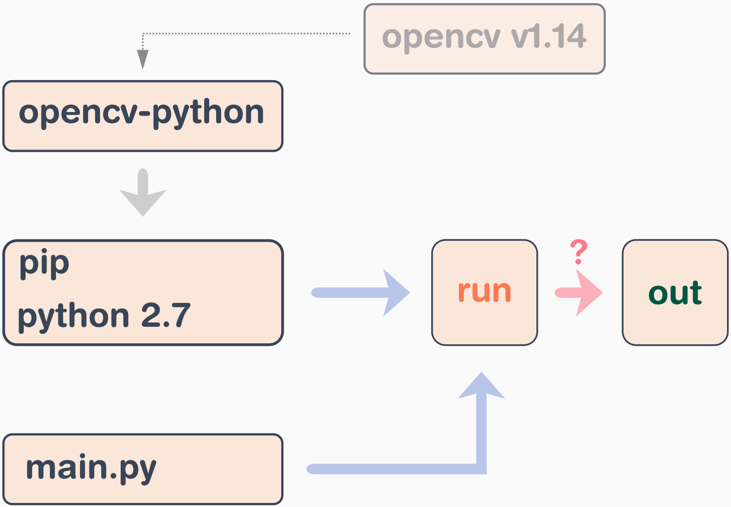
Reproducibility

works on my machine ¯_(ツ)_/¯

Reproducibility

1. Concrete and widespread
2. Mainstream tools do don’t this well

The problem


opencv v1.14

opencv-python

pip

python 2.7

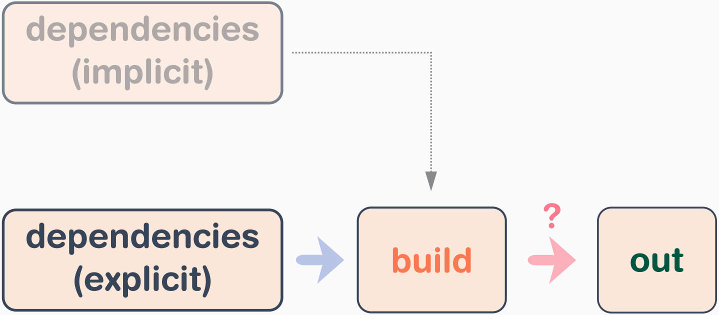
run

?

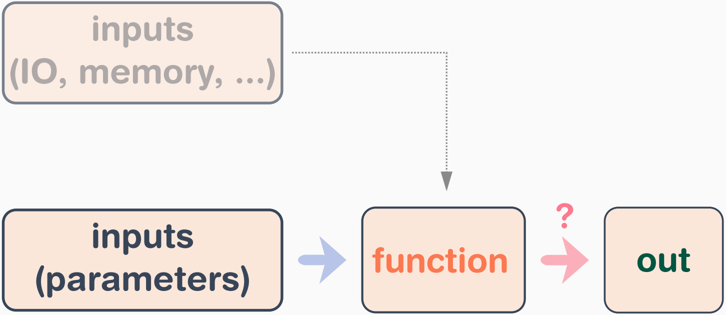
out

main.py

Looks familiar?



Looks familiar?



inputs (IO, memory, ...)

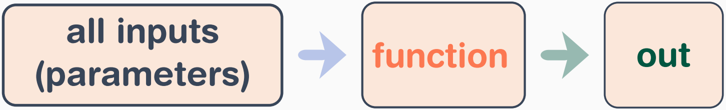
inputs (parameters)

function

?

out

Functional approach to reproducibility



all inputs (parameters)

function

out

Nix: the functional package manager

works on my machine ¯_(ツ)_/¯

Nix

works ~~on my machine ¯_(ツ)_/¯~~
everywhere

Principles

Principles

1. Describe a package and its dependencies in full

Describing

```
Derive(
  [("out","/nix/store/qya..-gh-from-shoe","","")],
  [
    ("/nix/store/ae4..-python-2-7-10.drv",
      ["out"]),
    ("/nix/store/78f..-opencv-1-14.drv",
      ["out"]),
    ...
  ["/nix/store/9kr..-default-builder.sh"],
  "x86_64-linux",
  ...
```

gh-from-shoe-1-0.drv

Principles

1. Describe a package and its dependencies in full
2. Build it in isolation

Building

```
gh-from-shoe-1.0$ nix build
```

1. Pull and build dependencies
   (`opencv-1-14`, `python-2-7-10`, . . .)
2. Create an isolated environment.
3. Run the builder.

Principles

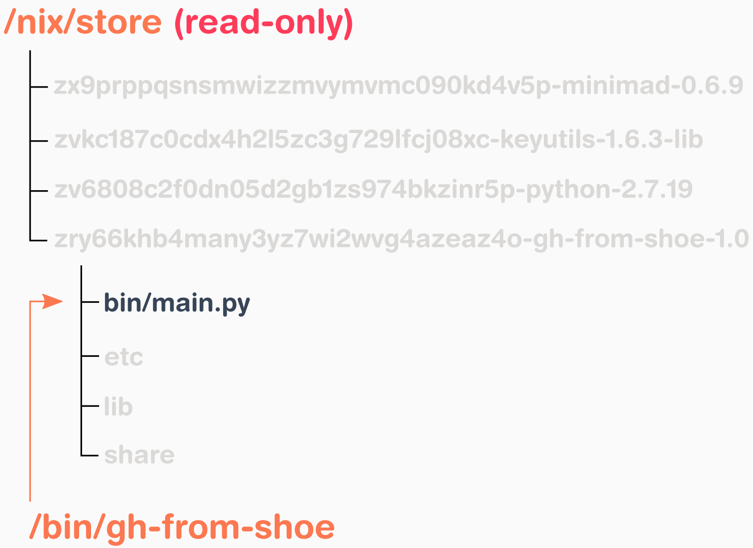
1. Describe a package and its dependencies in full
2. Build it in isolation
3. Put the result in the store

Storing

/nix/store (read-only)

- zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9
- zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib
- zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19
- zry66khb4many3yz7wi2wvg4azeaz4o-gh-from-shoe-1.0
  - bin
  - etc
  - lib
  - share

Storing



/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

zry66khb4many3yz7wi2wvg4azeaz4o-gh-from-shoe-1.0

bin/main.py

etc

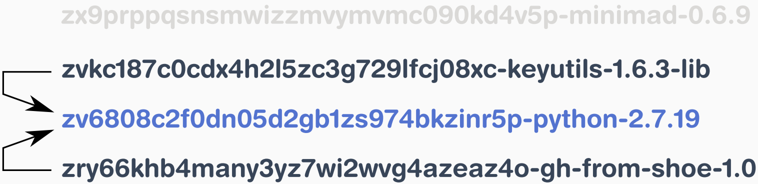
lib

share

/bin/gh-from-shoe

Sharing

/nix/store (read-only)



zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

zry66khb4many3yz7wi2wvg4azeaz4o-gh-from-shoe-1.0

Principles

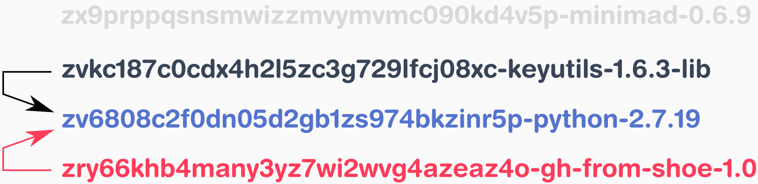
1. Describe a package and its dependencies in full
2. Build it in isolation
3. Put the result in the store
4. Profit!

Principles

1. Describe a package and its dependencies in full
2. Build it in isolation
3. Put the result in the store
4. Profit!
5. Clean

Cleaning

/nix/store (read-only)



zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib

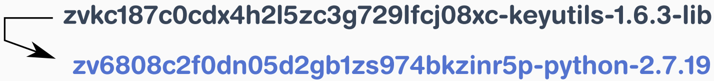
zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

zry66khb4many3yz7wi2wvg4azeaz4o-gh-from-shoe-1.0

Cleaning

/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

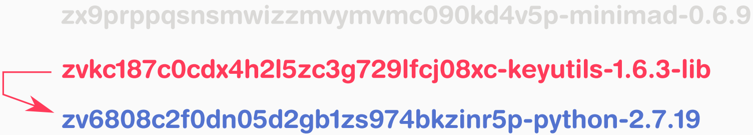


zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

Cleaning

/nix/store (read-only)



zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

zvkc187c0cdx4h2l5zc3g729lfcj08xc-keyutils-1.6.3-lib

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

Cleaning

/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

?

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

Cleaning

/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

?

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

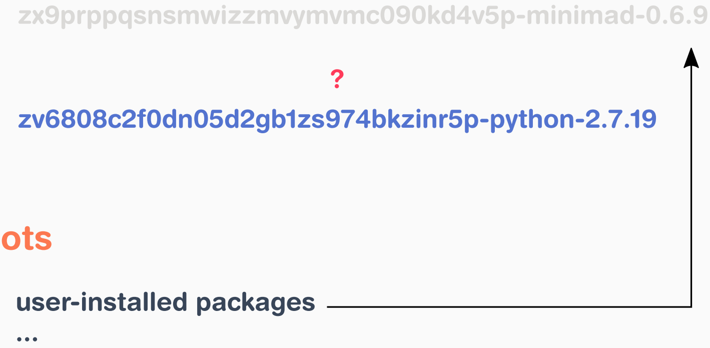
roots

user-installed packages

...

Cleaning

/nix/store (read-only)



zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

?

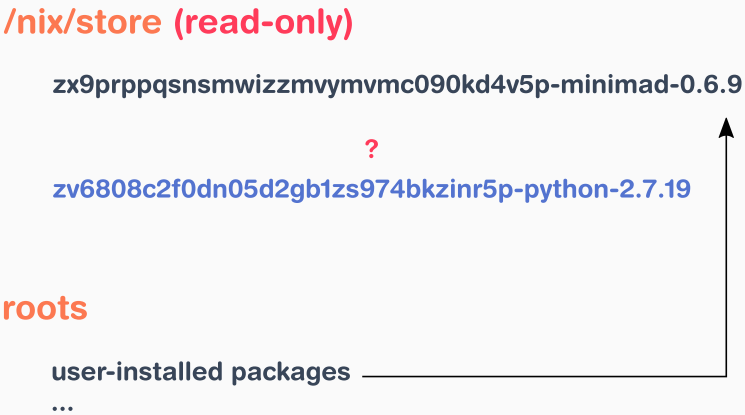
zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

roots

user-installed packages

...

Cleaning



/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

?

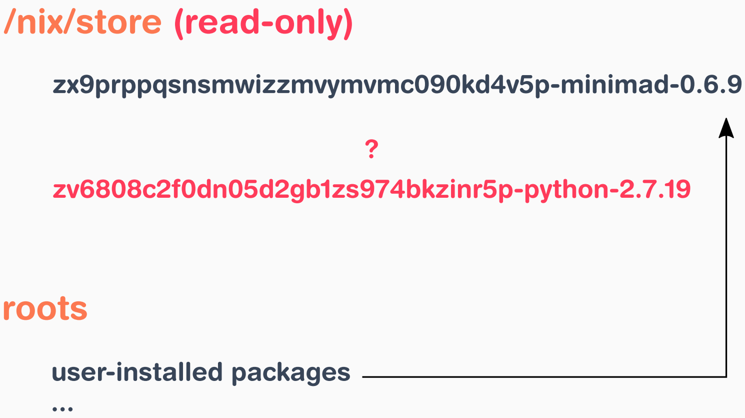
zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

roots

user-installed packages

...

Cleaning



/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

?

zv6808c2f0dn05d2gb1zs974bkzinr5p-python-2.7.19

roots

user-installed packages

...

Cleaning

/nix/store (read-only)

zx9prppqsnsmwizzmvymvmc090kd4v5p-minimad-0.6.9

roots

user-installed packages

...

Perks

Nix

- Declarative
- Reproducible
- Complete dependencies
- Fearless upgrades: atomic upgrades and rollbacks

Functional package management

Nix	Functional programming
Read-only store	Immutability
Hash addressing + sharing	Hash consing
Cleaning	Garbage collection
Reproducibility	Referential transparency

Nix expressions

Building a package should be a pure function: use a functional programming language!

Nix expressions

Nix expressions $=$ JSON $+$ λ

Nix expressions

```
{python2WithOpenCV, opencv, stdenv}:
stdenv.mkDerivation rec {
  pname = "gh-from-shoe-rust";
  version = "2021-04-30";

  buildInputs = [ python2WithOpenCV opencv ];

  installPhase = ''
    mkdir -p $out/bin
    cp ${./myscript.py} $out/bin/myscript
  '';
};
```

Derivation: Nix machine code

```
Derive(
  [("out","/nix/store/qya..-gh-from-shoe","","")],
  [
    ("/nix/store/ae4..-python-2-7-10.drv",
      ["out"]),
    ("/nix/store/78f..-opencv-1-14.drv",
      ["out"]),
    ...
  ["/nix/store/9kr..-default-builder.sh"],
  "x86_64-linux",
  ...
```

State of affairs

Nix expressions outgrew their initial scope.

In the wild

- Object systems (kind of): overriding
- A module system: NixOS
- Non-trivial algorithms (e.g. topological sort)
- No types
- and so on.

Nickel

Meet Nickel

A new take

- Gradual typing
- Run-time contracts
- Recursive records merge system
- Stand-alone language (Terraform, Kubernetes, etc.)

A teaser: contract

```
let Port = ...

let Service = {
  name | doc "Service name"
       | Str,

  openPorts | doc "Open ports (firewall)"
            | List #Port
            | default = [],
  ...
}
```

contracts.ncl

A teaser: configuration

```
let portToUrl : Str -> Num -> Str =
  fun host port => ...  in

{
  name = "nginx",
  openPorts = [80, 443],
  server = "localhost",
  urls = lists.map
    (portToUrl server)
    openPorts,
}
| Service
```

nginx.ncl

A teaser: result

```json
{
  "name": "nginx",
  "openPorts": [
    80,
    443
  ],
  "server": "localhost",
  "urls": [
    "http://localhost",
    "https://localhost"
  ]
}
```

nginx.json

Untyped code

By default, code is untyped:

- Terminating & fixed inputs
- JSON interop
- Contracts for validation

Example

```
services = [
  "init",
  {name = "firewall", bin = "/bin/firewall"},
  {name = "service", repo = "github.com/johndoe/dns-service"}
]
```

Typed code

Library code is statically typed:

- Triggered by *annotations*
- Scoped
- Type-inference

Example

```
map : forall a b. (a -> b) -> List a -> List b
    = fun f list =>
  if list == [] then []
  else
    let head = lists.head list in
    let tail = lists.tail list in
    [f head] @ map f tail
```

Interaction typed/untyped

Problem
Untyped code can sneak in ill-typed parameters

Example

```
let add : Num -> Num -> Num
        = fun x y => x + y in
add "a" 0
```

```
let add : Num -> Num -> Num = fun x y => x + y
                                         ^
This expression has type Str, expected Num
```

Contracts, the invisible glue

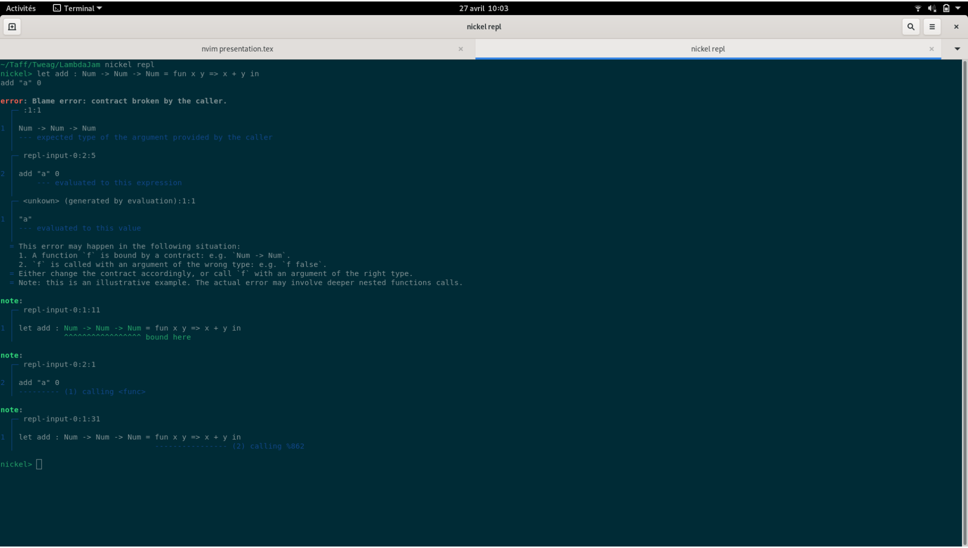
Typed code is protected by run-time casts, or *contracts*.

```
let safeNum = fun value =>
    if builtins.isNum value then value
    else panic! in

let addSafe = fun x y =>
  let safeX = safeNum x in
  let safeY = safeNum y in
  safeNum (safeX + safeY)
```

Generated code for `add`

Contracts, the invisible glue



```
~/Taff/Tweag/LambdaJam nickel repl
nickel> let add : Num -> Num -> Num = fun x y => x + y in
add "a" 0

error: Blame error: contract broken by the caller.
  ┌─ :1:1
  │
1 │ Num -> Num -> Num
  │ --- expected type of the argument provided by the caller
  │
  ┌─ repl-input-0:2:5
  │
2 │ add "a" 0
  │     --- evaluated to this expression
  │
  ┌─ <unkown> (generated by evaluation):1:1
  │
1 │ "a"
  │ --- evaluated to this value
  │
  = This error may happen in the following situation:
    1. A function `f` is bound by a contract: e.g. `Num -> Num`.
    2. `f` is called with an argument of the wrong type: e.g. `f false`.
  = Either change the contract accordingly, or call `f` with an argument of the right type.
  = Note: this is an illustrative example. The actual error may involve deeper nested functions calls.

note:
  ┌─ repl-input-0:1:11
  │
1 │ let add : Num -> Num -> Num = fun x y => x + y in
  │           ^^^^^^^^^^^^^^^^^ bound here

note:
  ┌─ repl-input-0:2:1
  │
2 │ add "a" 0
  │ --------- (1) calling <func>

note:
  ┌─ repl-input-0:1:31
  │
1 │ let add : Num -> Num -> Num = fun x y => x + y in
  │                               ---------------- (2) calling %862

nickel> 
```

First-class contracts

```
let Url =
  let pattern = "[-a-zA-Z0-9@:..." in
  fun label value =>
    if builtins.isStr value then
      if strings.isMatch value pattern then
        value
      else
        contracts.blame
          (contracts.tag "invalid URL" label)
    else
      contracts.blame
        (contracts.tag "not a string" label) in

let mkUrls
  | {url: #Url, pattern: Str} -> List #Url
  = ...
```

First-class contracts

```
Derivation | doc "A Nix package, in Nickel" = {
  name | Str,
  buildInputs | List #NixPackage,
},

NixPackage | doc "Interchange format" = {
  package | Str,
  input | Str
        | default = "nixpkgs",
  _type = "package",
},
```

First-class contracts

Perks

- Can check arbitrary properties
- Composable
- Allow safe typed/untyped interactions
- Built-in error reporting

Limits

- Run-time cost
- Untriggered code paths

Conclusion

Summary

- Reproducibility is a concrete and hard problem: Nix helps.
- Functional programming solves a similar problem: let's use the same solutions!
- What *broadly* interesting.

CONFLANG21

Configuration languages are a worthy area of research.

The 1st Workshop on Configuration Languages

Website	https://2021.splashcon.org/home/conflang-2021
Deadline	Friday 6 August 2021
Duration	1 day
Event	October 2021, at SPLASH 2021

Links

Nickel	https://github.com/tweag/nickel/
Nix	https://nixos.org/
Tweag's blog	https://www.tweag.io/blog

Contact

- yann.hamdaoui@tweag.io
- hello@tweag.io