



Amsterdam Nix Meetup





ContainerSolutions

Nix

Maarten Hoogendoorn





ContainerSolutions

Nix{lang, pkgs, os, ops}

Maarten Hoogendoorn



Why you should Nix

Fully declarative specification

Reproducible environment

One language to rule them all!



The Nixiversum

- Nixlang special purpose language (source files)
- nix-build build system (bundler / npm / cargo / gradle)
- Nixpkgs Homebrew
- NixOS Linux distribution build on Nixpkgs
- NixOps Chef / Puppet / Ansible / Terraform



ContainerSolutions

**Building software,
the Nix way.**



Building software

- Source
- Run build instructions



Building software

- Source
- Run build instructions
- Dependencies



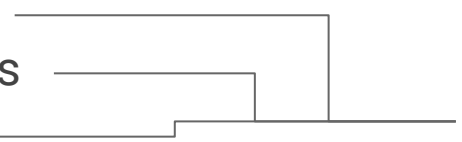
Building software

- Source
- Run build instructions
- Dependencies
- Store the build artifacts


Building software

- ~~Source~~ dependency
- Run build instructions
- Dependencies
- Store the build artifacts


Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- Inputs to build process
- 
- A diagram consisting of three horizontal lines of varying lengths, each starting from the right side of one of the first three list items. From the end of each horizontal line, a vertical line extends downwards, and then a horizontal line extends to the right, meeting a single vertical line that descends from the top of the third list item. This structure indicates that the first three items are inputs to the build process.

Building software


- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- Inputs to build process
- Build process has no side-effects *
- 
- A diagram consisting of three horizontal lines of varying lengths, each starting from the left and ending with a vertical line that drops down to a common horizontal line. The top line starts under 'Source dependency', the middle line under 'Run build instructions', and the bottom line under 'Dependencies'. These three vertical lines all meet a single horizontal line that extends to the right, ending under the text 'Inputs to build process'.

Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- 
- Inputs to build process
- The diagram consists of three horizontal lines of varying lengths, each starting from the left and ending with a vertical line that drops down to a common horizontal line. The top line starts under 'Source dependency', the middle line under 'Run build instructions', and the bottom line under 'Dependencies'. These three vertical lines all meet the common horizontal line, which then extends to the right towards the text 'Inputs to build process'.


- Build process has no side-effects *
- Pure function: $f(x) = x + 1$

Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- 
- Inputs to build process
- The diagram consists of three horizontal lines of varying lengths, each connected to a vertical line that drops down to a common horizontal line. The top line is the longest, the middle line is shorter, and the bottom line is the shortest. These lines represent the inputs to the build process.


- Build process has no side-effects *
- Pure function: $f(x) = x + 1$ (impure function: $f(x) = time_of_day$)

Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- 
- Inputs to build process
- The diagram consists of three horizontal lines of varying lengths, each connected to a dot on the left. The top line is the longest, the middle line is shorter, and the bottom line is the shortest. From the right end of each line, a vertical line goes down, and then a horizontal line goes to the right, all three of which eventually merge into a single horizontal line that points to the text 'Inputs to build process'.


- Build process has no side-effects *
- Pure function: $f(x) = x + 1$ (impure function: $f(x) = time_of_day$)
- Output only depends on x , the input.

Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- 
- Inputs to build process
- The diagram consists of three horizontal lines of varying lengths, each connected to a vertical line that drops down to a common horizontal line. The lines are connected to the first three items in the list above: 'Source dependency', 'Run build instructions', and 'Dependencies'. The text 'Inputs to build process' is positioned to the right of this common line.

- Build process has no side-effects *
- Pure function: $f(x) = x + 1$ (impure function: $f(x) = time_of_day$)
- Output only depends on x , the input.
- Nix does this, for building software!

Building software

- ~~Source~~ dependency
 - Run build instructions
 - Dependencies
 - Store the build artifacts
- 
- Inputs to build process
- The diagram consists of a horizontal line on the right side. From this line, three vertical lines extend upwards to the left, each ending in a horizontal line that points to one of the first four bullet points. The top vertical line connects to 'Source dependency', the middle one to 'Run build instructions', and the bottom one to 'Dependencies'. The horizontal line for 'Dependencies' is the longest, extending from the far right towards the left.

- Build process has no side-effects *
- Pure function: $f(x) = x + 1$ (impure function: $f(x) = time_of_day$)
- Output only depends on x , the input.
- Nix does this, for building software!
- Cache on the input, don't recompute/rebuild!

Hello World

```
{ stdenv, fetchurl }:  
stdenv.mkDerivation rec {  
  name = "hello-2.10";  
  
  src = fetchurl {  
    url = "mirror://gnu/hello/${name}.tar.gz";  
    sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";  
  };  
}
```

Hello World

```
{ stdenv, fetchurl }:
```

```
stdenv.mkDerivation rec {
```

```
  name = "hello-2.10";
```

```
  src = fetchurl {
```

```
    url = "mirror://gnu/hello/${name}.tar.gz";
```

```
    sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
```

```
  };
```

```
}
```



Hello World

```
{ stdenv, fetchurl }:
```

```
stdenv.mkDerivation rec {
```

```
  name = "hello-2.10";
```

```
  src = fetchurl {
```

```
    url = "mirror://gnu/hello/${name}.tar.gz";
```

```
    sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
```

```
  };
```

```
}
```



Hello World

```
{ stdenv, fetchurl }:
```

```
stdenv.mkDerivation rec {
```

```
  name = "hello-2.10";
```

```
  src = /nix/store/0ssi1wpaf7....;
```

```
}
```

Developer environment for free!

- Dependencies are know!
- Nix-shell

DEMO devshell of nixops

more?



Where to go next?

- **Download page**
<https://nixos.org/nixos/download.html>
- **A tour of nixlang**
<https://nixcloud.io/tour/>
- **Nix pills** <http://lethalman.blogspot.nl/2014/07/nix-pill-1-why-you-should-give-it-try.html>
- **IRC**
freenode, #nixos
- **E-mail list**
<http://lists.science.uu.nl/mailman/listinfo/nix-dev>

Why you should Nix

Fully declarative specification

Reproducible environment

One language to rule them all!



Thank you



ContainerSolutions

We Are Hiring!

Currently we are looking for:

Systems Developers
Application Developers
Support Engineers
UI/UX Developers



kubernetes

We're hiring in Amsterdam, London, Zürich and remote
container-solutions.com/careers



ContainerSolutions