Introduction to Nix{,OS,Ops}

Valentin Reis, Michael Mercier

June 7, 2017

Functional programming

Based on **lambda calculus** introduced by Alonzo Church (1930s)
Abstraction based on functions (lambda) and reduction

No side effects:

- No mutable variables
- No loops

Nix Primitive Types

- Integer
- Boolean
- Path
- String
- **No** Float

```
nix-repl> builtins.typeOf 42
"int"
nix-repl> builtins.typeOf true
"bool"
nix-repl> builtins.typeOf /tmp
"path"
nix-repl> builtins.typeOf "toto"
"string"
```

Nix Composite Types

- Set
- List
- Functions (lambda)

```
nix-repl> { "name" = "michael"; age = 31; }.name
"michael"
nix-repl> builtins.elemAt [ 12 13 "toto" ] 1
13
nix-repl> x: x+x
lambda
nix-repl> let add = x: x+x;
          in add 2
4
```

Lots of builtins functions

Lots of low and high level builtins functions to:

- Fetch tarballs and source code
- Manipulate strings, lists, sets and paths
- Do flow control (tests, assert, abort...)

```
builtins.fetchTarball
builtins.fetchurl
builtins.findFile
builtins.fromJSON
builtins.getAttr
builtins.getEnv
builtins.hasAttr
[...]
```

Why an other language?

Advantages over Descriptive Language

- Can express package complexity (Turing complete)
- Better reusability
- More concise and simple

Drawbacks

- A new language to learn
- A new language to maintain
- smaller community

Note that Guix uses the Guile language to avoid these issues. . .

Purely Functional package manager

- Package are defined in Nix expressions
- Atomic upgrades and rollbacks
- Several version of the same package on the same system
- Unprivileged package installation
- Provides isolated environments
- Reproducible build from source
- Cache available to get pre-compiled binaries
- Garbage collection

Getting Started

Simple to install (in single user mode)

```
curl https://nixos.org/nix/install | sh
```

- No privileges needed
- Any Unix is supported (even MacOS X)
- Configuration in *.nix-profile*
- Packages stored in */nix/store/*

To use it just make it available in your path with:

```
source ~/.nix-profile/etc/profile.d/nix.sh
```

Installing a Package

```
$ nix-env -i zsh
```

1. Check if the package is in your Store
2. If not check if the package is in the channel cache
3. If not build it locally
4. Then put it in your store
5. Make it available with symlinks

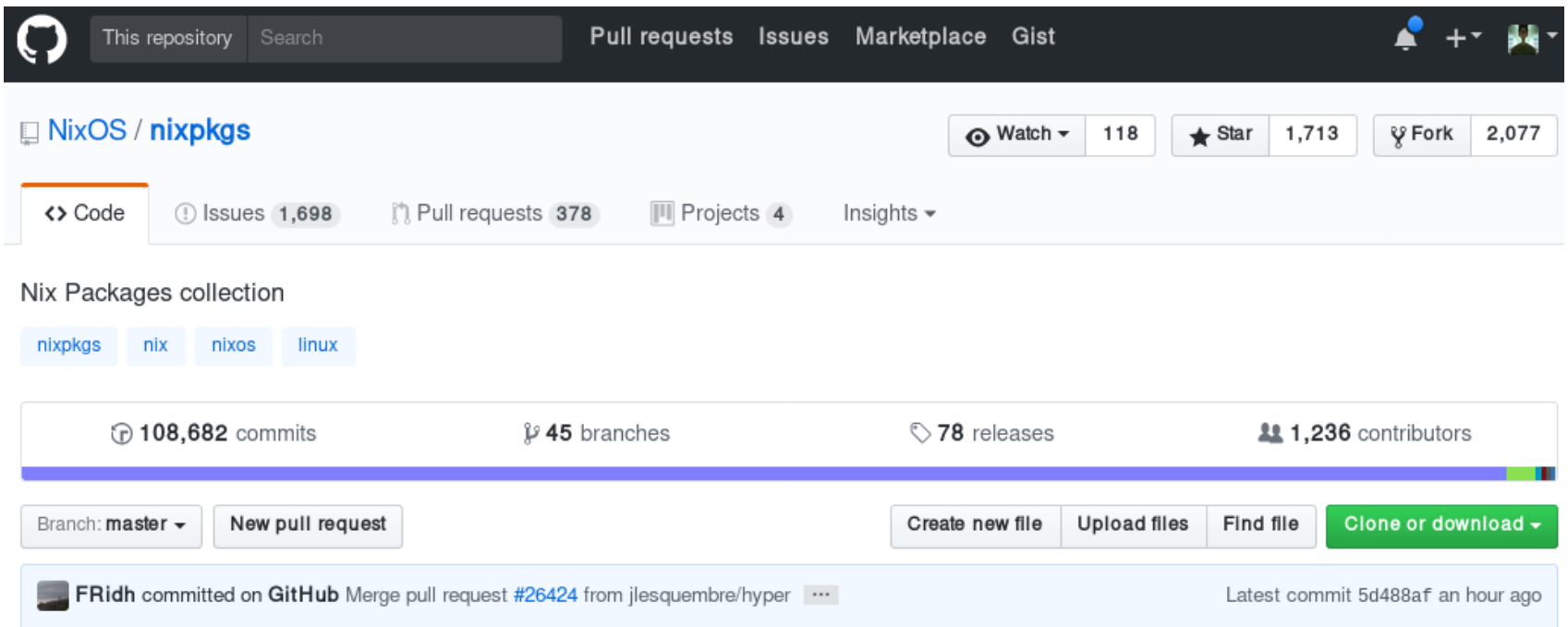
All packages are defined by a hash computed on the source and the Nix expression

```
/nix/store/1gqj6zr1x0n812qxijy066fzrypgh3im-hello-2.10
```

Nixpkgs

More than 6500 packages in one source repository!
Contains all the basic build functions and scripts

Very active community :)



What is a Nix package

```
{ stdenv, fetchurl }:

stdenv.mkDerivation rec {
  name = "hello-2.10";

  src = fetchurl {
    url = "mirror://gnu/hello/${name}.tar.gz";
    sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
  };

  doCheck = true;

  meta = {
    description = "A program that produces a familiar, friendly greeting";
    longDescription = ''
      GNU Hello is a program that prints "Hello, world!" when you run it.
      It is fully customizable.
    '';
    homepage = http://www.gnu.org/software/hello/manual/;
    license = stdenv.lib.licenses.gpl3Plus;
    maintainers = [ stdenv.lib.maintainers.eelco ];
    platforms = stdenv.lib.platforms.all;
  };
}
```

Environments and Shells

An environment with Redis and Python 3.5:

```
$ nix-shell -p redis -p python35
[...]
[nix-shell:~]$ which python
/nix/store/5kfn0xxh3ipjdyjly2d5wrmh4cidsm8k-python3-3.5.3/b
```

You can create one (or more) environment for each project:

1. Define an environment in *./default.nix*
2. Run *nix-shell*
3. Now you're in your development environment!

Nix takeaway

The Good

- Concise package description
- Easy to contribute (just a pull request)
- Easy to setup custom environments
- You can use it for (pretty much) everything :)

The Bad & the Ugly

- A new language to learn
- Small (but growing) community
- Not so well documented
- CLI interface not intuitive
- Error messages and tracebacks

NixOS

"NixOS, The Purely Functional Linux Distribution"
A GNU/Linux distribution written in the Nix language.
License: MIT
Platforms: 32bit, 64bit x86, (ARM).
Age: Young teenager.
Systemd-Based. GNU GPL version: Guix

Overview

We write some Nix expressions in a *.nix* file and get declarative (programmatic) configuration of:

Overview

We write some Nix expressions in a *.nix* file and get declarative (programmatic) configuration of:

- OS (hardware, boot, kernel, filesystem, X11..)

Overview

We write some Nix expressions in a *.nix* file and get declarative (programmatic) configuration of:

- OS (hardware, boot, kernel, filesystem, X11..)
- packages (*nixpkgs* is in scope.)

Overview

We write some Nix expressions in a *.nix* file and get declarative (programmatic) configuration of:

- OS (hardware, boot, kernel, filesystem, X11..)
- packages (*nixpkgs* is in scope.)
- services (systemd, containers..).

Overview cont.

Features:

Overview cont.

Features:

- Atomic updates

Overview cont.

Features:

- Atomic updates
- Rollbacks (incl. from bootloader)

Overview cont.

Features:

- Atomic updates
- Rollbacks (incl. from bootloader)
- Reproducible OS build

Overview cont.

Features:

- Atomic updates
- Rollbacks (incl. from bootloader)
- Reproducible OS build
- Source-based (now merged into *nixpkgs*).

Getting Started

Boot on any NixOS machine (e.g. liveCD), then

```
mkfs.ext4 -L nixos /dev/sda1
mount /dev/disk/by-label/nixos /mnt
```

the NixOS configuration file in */mnt/etc/nixos/configuration.nix*

```
nixos-install --root=/mnt
```

Voila

Getting Started

Boot on any NixOS machine (e.g. liveCD), then

```
mkfs.ext4 -L nixos /dev/sda1
mount /dev/disk/by-label/nixos /mnt
```

the NixOS configuration file in */mnt/etc/nixos/configuration.nix*

```
nixos-install --root=/mnt
```

Voila

Other options: install manually from existing OS, *nixos-in-place*, *nixos-assimilate*, *mkISO*.

Example configuration.nix

```
{ config, pkgs, ... }:
{ imports = [./hardware-configuration.nix];
  time.timeZone = "Europe/Paris";
  boot.loader.grub = {
    enable = true;
    version = 2;
    device = "/dev/sda";
  };
}
```

Low level OS features

```
boot.initrd.availableKernelModules =
  [ "xhci_pci" "ehci_pci" ];
fileSystems."/" = {
  device = "/dev/disk/by-uuid/c1...";
  fsType = "ext4";
};
swapDevices = [
  { device = "/dev/disk/by-uuid/0a..."; }
];
```

Networking

```
networking = {
  hostName = "myhostname";
  networkmanager.enable = true;
  firewall = {
    enable = true;
    allowedUDPPortRanges =
      [ {from = 60000; to = 61000;} ];
    allowedTCPPorts = [ 8384 ];
  };
};
```

User management

```
users.extraUsers.fre = {
  initialHashedPassword = "hash" ;
  shell = "${pkgs.zsh}/bin/zsh";
  description = "toto";
  extraGroups =
  ["wheel" "networkmanager" "atd" "lp"];
  openssh.authorizedKeys.keys =
  [ "ssh-rsa AA.." ];
};
```

High-level features

- X11
- fonts
- timezone
- user systemd services
- containers

You will quickly learn to

$ sudo nixos-rebuild switch :
compiles and builds the new configuration and changes your environment's symlinks.

You will quickly learn to

$ sudo nixos-rebuild switch :
compiles and builds the new configuration and changes your environment's symlinks.

#!/usr/bin/env :
there is no /bin, /sbin, /lib, /usr, and so on.
All packages are kept in /nix/store and your scripts will break.
(blobs need to be run through patchELF; for ephemeral use some standard environments like *steam-run* exist.)

You will quickly learn to; cont.

Use *https://nixos.org/nixos/options.html*:

```
$ ls -lah /etc/ssh/ssh_config
lrwxrwxrwx 1 root root   26 Sep 16 12:51
ssh_config -> /etc/static/ssh/ssh_config
$ ls -lah /etc/static/ssh/ssh_config
lrwxrwxrwx 1 root root 58 Jan  1  1970
/etc/static/ssh/ssh_config ->
    /nix/store/g5j....k4a-etc-ssh_config
```

What doesn't it do?

NixOS does not manage:

- mutable parts of the system. (e.g. */var/www/*)
- user configuration (*~/.**) *yet*
- secrets (/nix/store is world readable)

NixOps

"NixOps, The NixOS Cloud Deployment Tool"
An infrastructure deployment tool for networks of NixOS machines.
License: LGPL3.0
Age: 4y
Similar projects: Puppet/Chef/Ansible.

Overview

We write some Nix expressions in a .nix file and get declarative configuration of:

- OS
- packages
- services
- network, provisioning.. a.k.a. deployment.

Overview

We write some Nix expressions in a .nix file and get declarative configuration of:

- OS
- packages
- services
- network, provisioning.. a.k.a. deployment.

Plus the NixOps utilities.

Example

```
{
  webserver = {
    deployment.targetEnv = "virtualbox";
    services.httpd.enable = true;
    services.httpd.documentRoot = "/data";
    fileSystems."/data" = {
      fsType = "nfs4";
      device = "fileserver:/";
    };
  };

  fileserver = {
    deployment.targetEnv = "virtualbox";
    services.nfs.server.enable = true;
    services.nfs.server.exports = "...";
  };
}
```

Network

```
{
 main =
 { nodes, ... }:
 { networking = {
     firewall = {
       allowedTCPPorts = [80];
       allowedUDPPorts = [80];
     };
     nat.enable = true;
     nat.forwardPorts = [
         {destination = "${nodes.web.config.networking.privateIPv4}:80";
         sourcePort =  80;}
       ];
         ...
 };

 web = { ...  };
}
```

Provisioning

```
{
 main = { ...  };

 web =
 {
   deployment= {
     targetEnv = "container";
     container.host = resources.machines.main;
   };
    ...
 };
}
```

Provisioning, cont.

NixOps supports the following resources:

- NixOS hosts and containers.
- VirtualBox virtual machines.
- Amazon EC2 instances and other resources (such as S3 buckets, EC2 key pairs, elastic IPs, etc.).
- Google Cloud Engine instances and other resources (such as networks, firewalls, IPs, disks, etc.).
- Hetzner machines.

Tooling

```
$ nixops info
+-------------+-----------------+-----------+-----------------------+-------------+
| Name        |      Status     | Type      | Resource Id           | IP address  |
+-------------+-----------------+-----------+-----------------------+-------------+
| backup      | Up / Up-to-date | container | backup                | 10.233.3.2  |
| cgit        | Up / Up-to-date | container | cgit                  | 10.233.5.2  |
| fileserver  | Up / Up-to-date | container | fileser               | 10.233.1.2  |
| prosody     | Up / Up-to-date | container | prosody               | 10.233.8.2  |
| radicale    | Up / Up-to-date | container | radical               | 10.233.7.2  |
| webfre      | Up / Up-to-date | container | webfre                | 10.233.4.2  |
| websplit    | Up / Up-to-date | container | webspli               | 10.233.2.2  |
| webstray    | Up / Up-to-date | container | webstra               | 10.233.6.2  |
| main        | Up / Up-to-date | none      | nixops - [...] - main |             |
+-------------+-----------------+-----------+-----------------------+-------------+
```

Other approaches

μservice	Disnix allows to deploy and monitor services running on any Nix-enabled machine.
Docker	Build (non-nixOS) Docker images with dockerTools.buildImage.
Cloud	Nix-Kubernetes allows to configure kubernetes using Nix (declarative imperative configuration ++)

References

On the language itself:

https://medium.com/@MrJamesFisher/nix-by-example-a0063a1a4c55

On Guix vs Nix:

http://sandervanderburg.blogspot.fr/2012/11/on-nix-and-gnu-guix.html

Use nix-shell for python project:

http://datakurre.pandala.org/2015/10/nix-for-python-developers.html

Series of blogs on Nix:

http://lethalman.blogspot.fr/2014/07/nix-pill-1-why-you-should-give-it-try.html