The Nix Project: Past, Present, Future

Eelco Dolstra

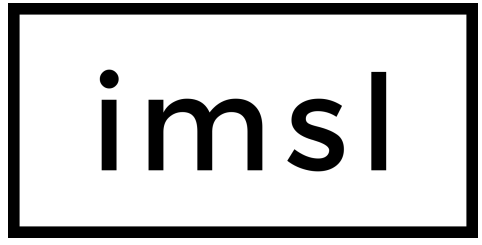
14 November 2015

Welcome to NixCon 2015!

A big thank you to the organisers!

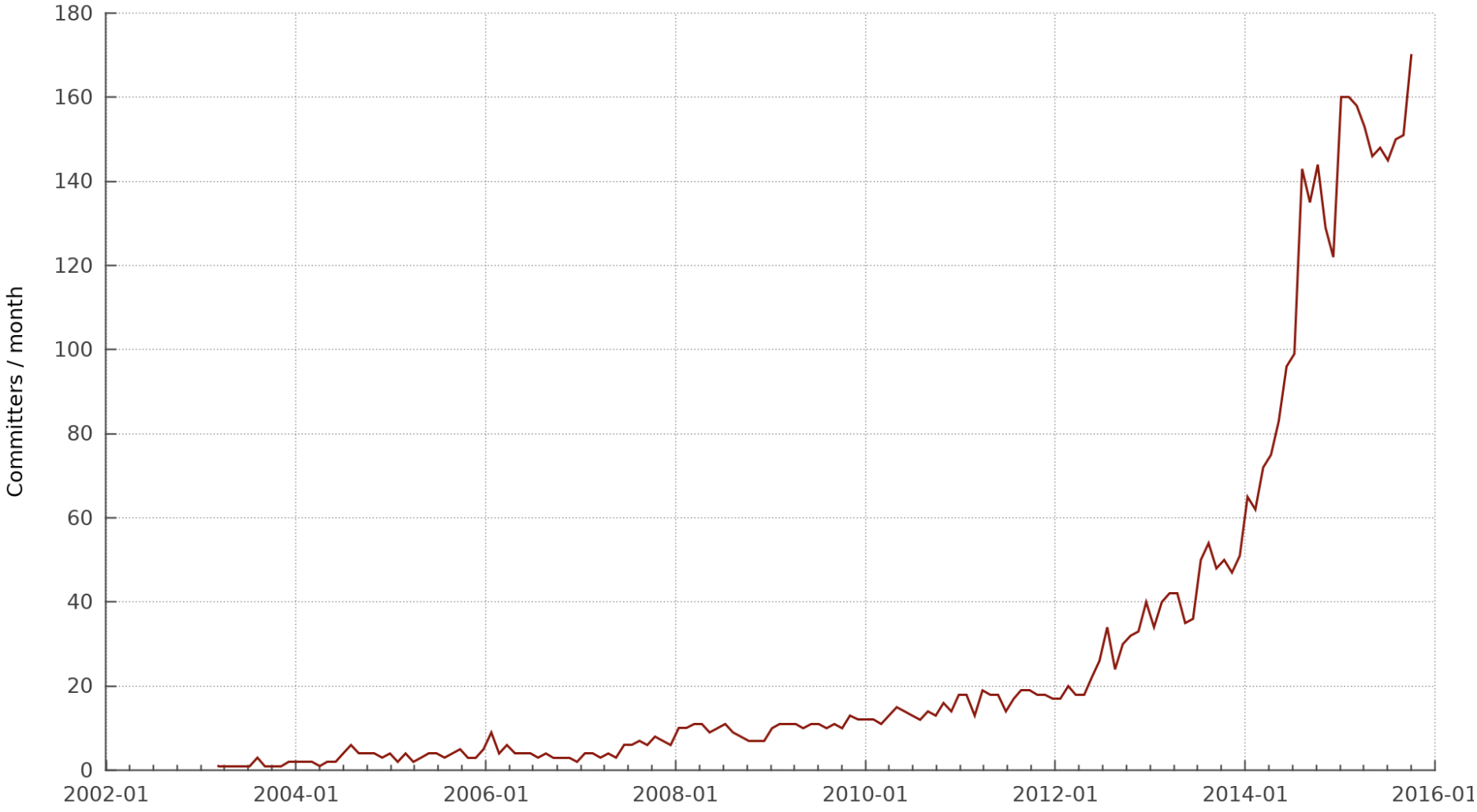
- Rok Garbas
- Paulus Esterhazy
- Peter Simons

And a big thank you to the sponsors

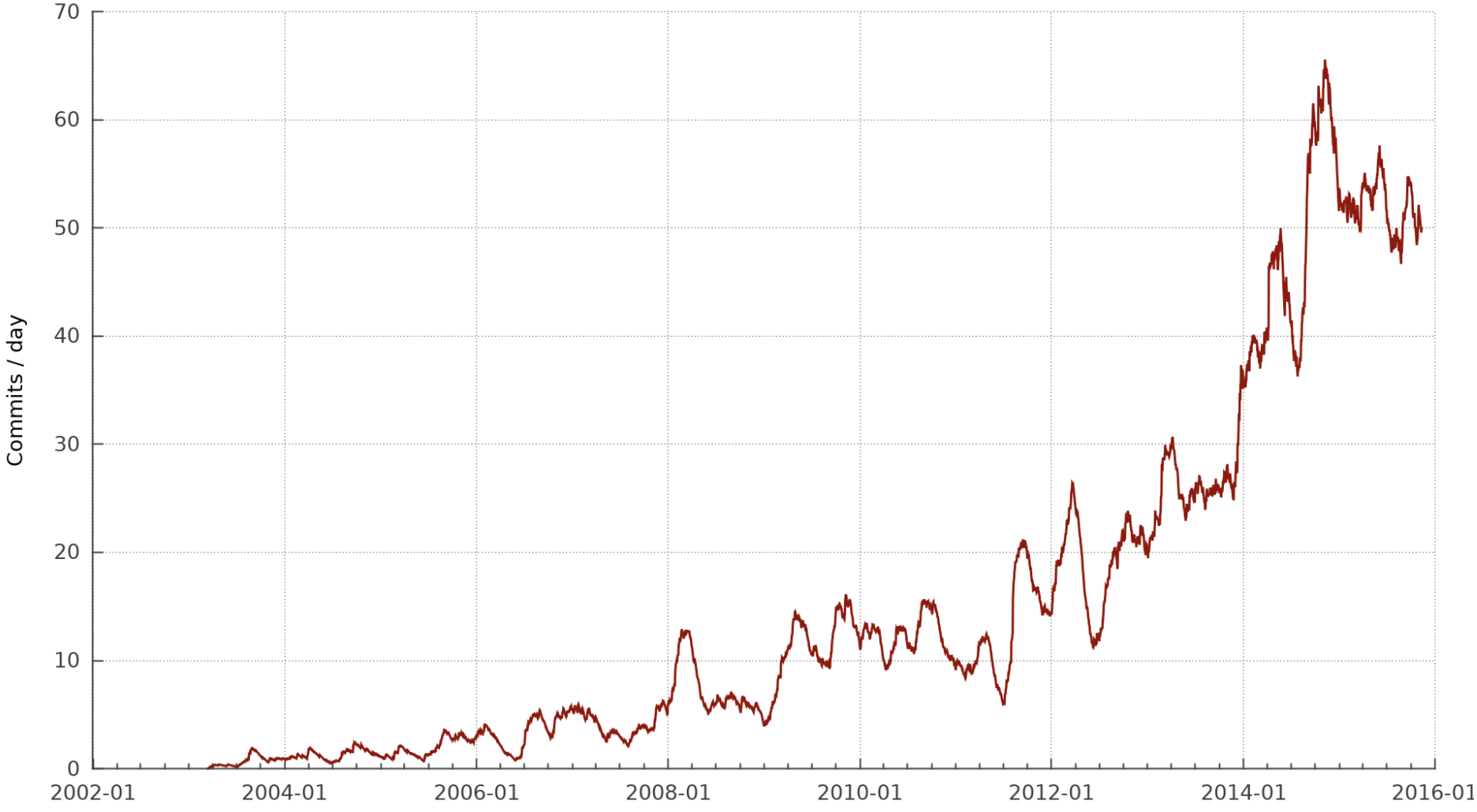


What has been going on?

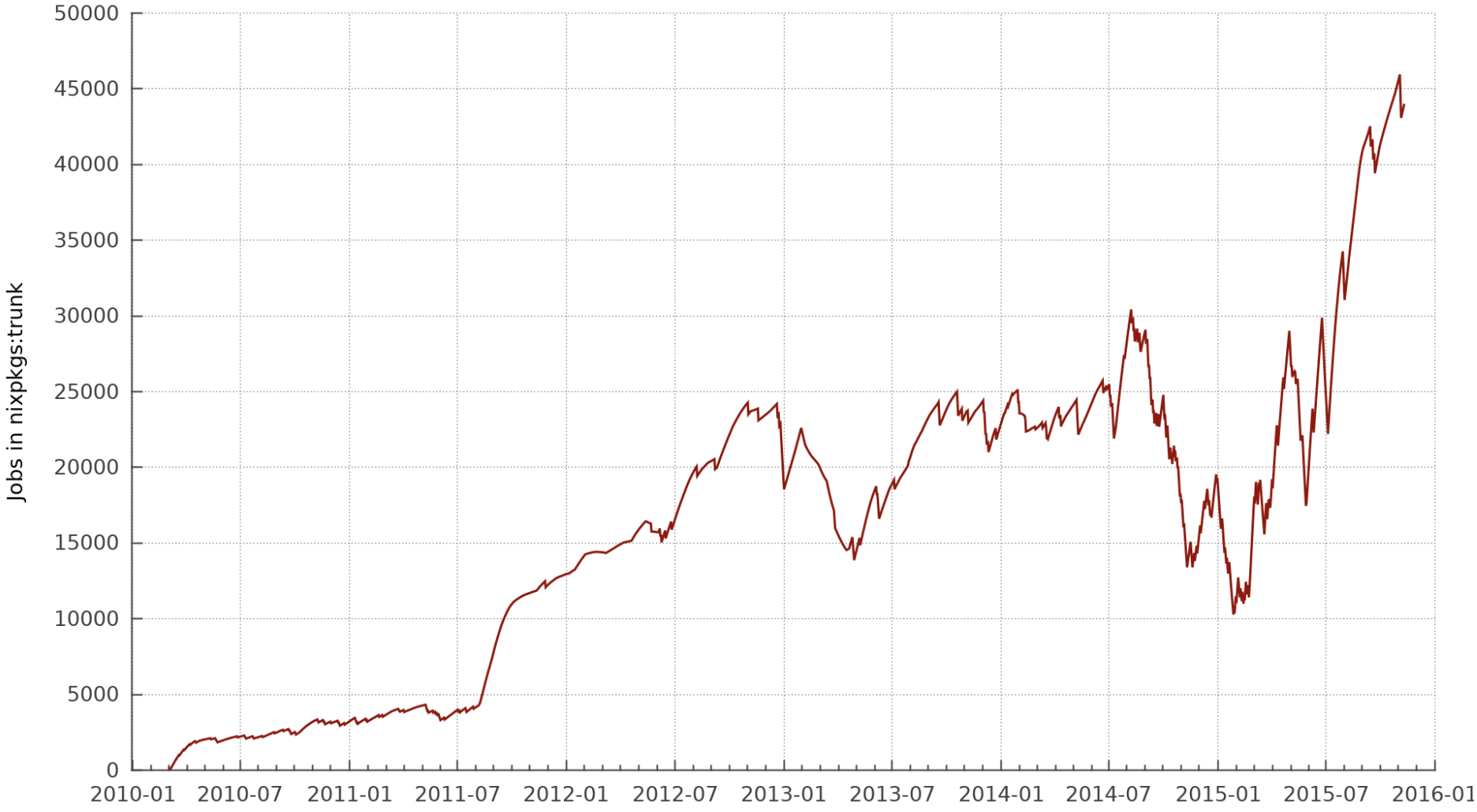
Nixpkgs committers per month



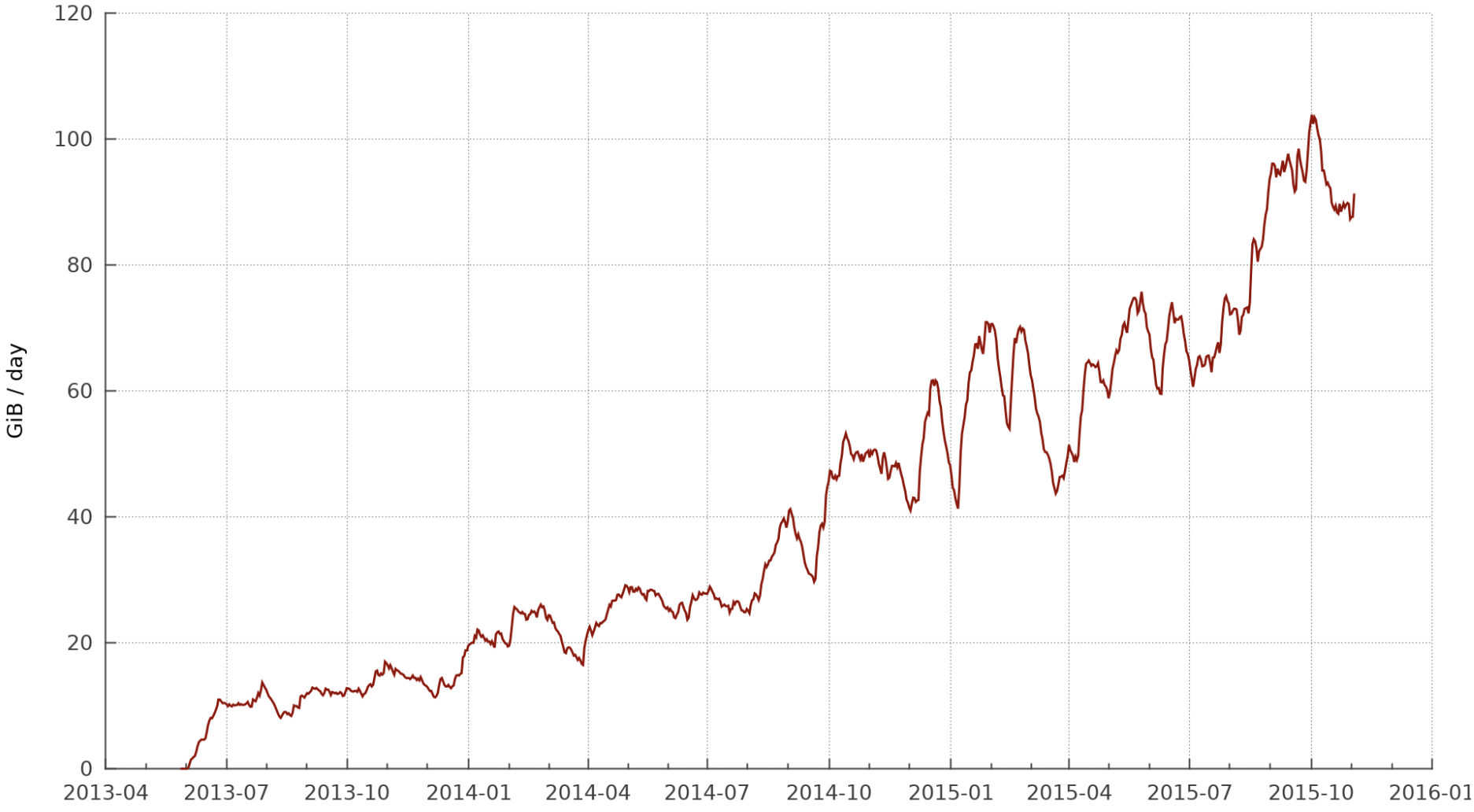
Nixpkgs commits per day



Nixpkgs size



Binary cache traffic



Binary cache users

GitHub

- 8246 closed PRs, 291 open
- 1673 closed issues,771 open

NixOS releases

- 14.12 “Caterpillar”
- 15.09 “Dingo”
- Zillions of new packages, services

Modular Tex Live packaging!

```
with import <nixpkgs> {};

texFunctions.runLaTeX {
  rootFile = ./doc.ltx;
  texPackages = {
    inherit (pkgs.texlive)
      scheme-basic pgf beamer ms soul ec
      textpos metafont helvetic times listings;
  };
}
```

Nix releases

- 1.8, 1.9, 1.10
- Signed binary caches
- Automatic downloading of Nix expressions
  - `nix-env -f https://github.com/NixOS/nixpkgs-channels/archive/8a3e....tar.gz -i firefox`
- Mac OS X sandbox support

Hydra improvements

- Nixpkgs/NixOS continuous builds are a major bottleneck
- New Hydra queue runner schedules builds more efficiently
- Hydra provisioner fires up EC2 spot instances to do builds
- More improvements on the way

NixOS Foundation

- Non-profit org to support NixOS and related projects
- Primarily intended to handle donations for
  - Hydra hardware
  - AWS costs
  - Conferences
  - ...

Roadmap

Disclaimer: Roadmaps are fiction

11.1. Future work

There are a number of possibilities for further research, as well as a several remaining practical issues.

A fully Nixified system Since Nix works best in a "pure" environment where *all* components are built and deployed by Nix, it is natural to consider an operating system distribution on that basis; that is, a Unix system (probably based on Linux or FreeBSD) that has no /bin, /lib, /usr, etc.—all components are in /nix/store. In addition, the configuration of the system can be managed along the lines of the services in Chapter 9. For instance, the configuration of which services (such as daemons) should be enabled or disabled can be expressed in a Nix expression that builds a component that starts and stops other components passed as build-time inputs. To change the configuration, the Nix expression is changed and rebuilt.

A pure[1] Nix-based Linux system (tentatively called NixOS) is currently in active development and in a bootable state. Building components in this pure environment has already revealed a few impurities in Nixpkgs. On the other hand, the number of such impurities has turned out to be surprisingly low: once NixOS was up and running, it took only two one-line modifications (to stdenv, no less!) to get Firefox and all its dependencies to build.

Nix Future Work (2006)

- A fully Nixified system (yes)
- The intensional model (no)
- A language for builders (no)
- A type system (no)
- Efficient storage (no)
- Blacklisting (no)
- Upgrading in the intensional model (sort of)
- Feature selection and automatic instantiation (eh...)

Nix Roadmap

- Nix 2.0
  - New command line interface
  - Distributed binary cache
- Nix 3.0
  - Content-addressed Nix store

New command line

- Nix command line interface is pretty crufty
  - Lots of commands, inconsistent behaviour
  - Old suboptimal design choices, like `nix-env` using package names (slow) rather than attribute names
  - Nixpkgs options not exposed
  - `nix-env` only does imperative package management
- Solution: Replace by a single git-style command

New command line (cont'd)

- `nix install`, `nix gc`, `nix shell`, ...
- Based on attribute names: `nix install xorg.xmessage`
- Maybe do caching of `nix search`
- Maybe support declarative style: `nix rebuild`
- Maybe deprecate channels — `NIX_PATH=nixpkgs=http://` is enough

Nixpkgs discoverability

- `nix query` should show available package options
- Requires new package formalism

```
firefox = mkPackage
  { enableBranding ? false
  , enablePulseaudio ? true
  , ... }:
  ...
```

Distributed binary cache

- Download binary only when it has at least N signatures
- Requires greater build determinism

Content-addressed Nix store

- Packages stored in /nix/store/HASH where HASH = hash(contents)
- A.k.a intensional model
- Allows any user to install packages (using any binary cache)
- Also requires greater build determinism

NixOS Roadmap

- Closure size reduction
- Improved container support
- Systemd update, GCC 5, ...

Closure size reduction

- NixOS system closures are too big

```
$ du -sch $(nix-store -qR /run/current-system)
3.3G    total
```

- Bad for containers, cloud deployments
- Solution: multiple outputs — move stuff like headers, docs into separate store paths that can be downloaded/GCed separately
- Has been on-going for quite a while

Hydra improvements

- Upload build results directly to binary cache
- Removes dependency on disk of central Hydra machine (which has only 3 TB)
- Will need more Mac OS X builders