Nix(OS) - The Power To Revolutionize!

The Purely Functional Linux Distribution

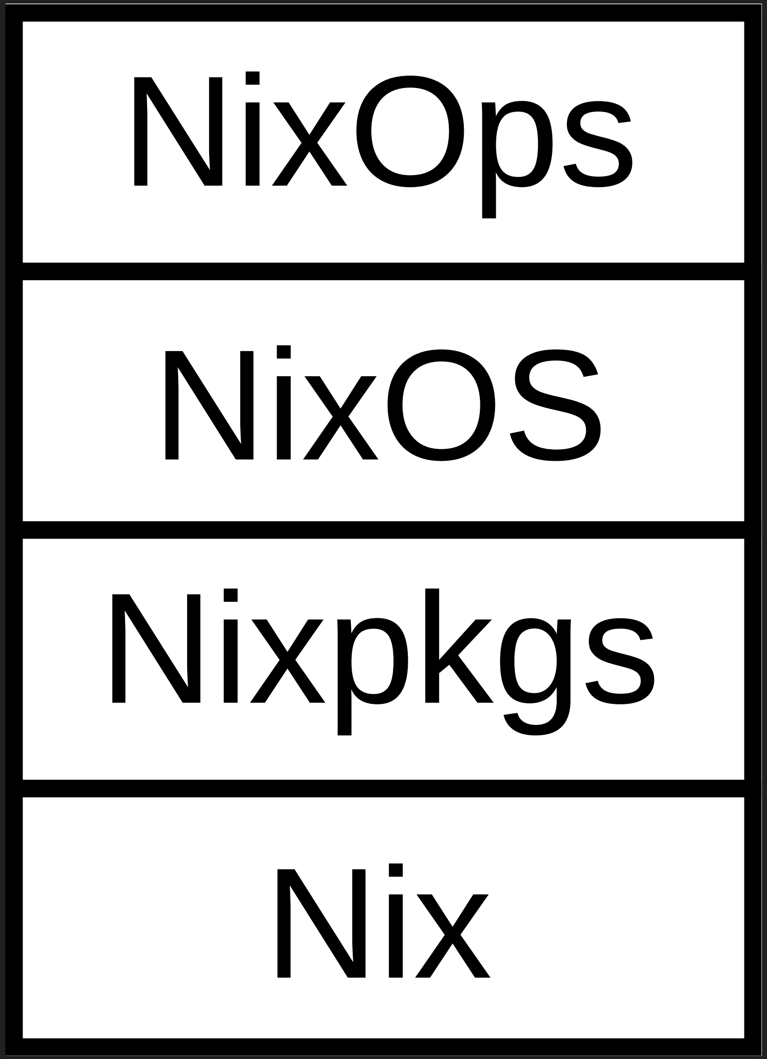
Before we begin

- Ask questions at any time
- Please ask lots of questions :)
- FYI: The slides contain some redundancy
- Give us feedback
  - Faster/slower
  - More/less details
  - Etc.

Main components

- Nix (package manager)
- Nixpkgs (Nix packages collection)
- NixOS (operating system)
- NixOps (DevOps / cloud deployment tool)

Nix* ISO/OSI model



Other tools

- Hydra (Nix based continuous build system)
- Disnix (distributed services deployment)
- PatchELF (change dynamic linker and RPATH)
- {cabal,go,node,pip,python,pypi,composer,hex,bower,vim,...}2

History

- Started as a **research project** (with funding)
- First paper in 2004 (many will follow)
- Nix package manager developed by **Eelco Dolstra** as part of his PhD research (~2003)
- First NixOS prototype developed by Armijn Hemel as his master's thesis project
- Hydra developed as part of the LaQuSo Buildfarm project

Timeline

- 2003: init (research begins)
- 2007: NixOS becomes usable + x86_64 support
- 2008: Website moved to nixos.org
- 2009: Nix logo + Nix(OS) build on Hydra
- 2011: Migration from Subversion to Git(Hub)
- 2013: Switch from Upstart to systemd + NixOps 1.0 release
- 2015: NixOS Foundation + First NixCon (Berlin)
- 2017: Second NixCon (Munich)

Problems of classical package managers

- Upgrades/configuration changes destructively update the system state (overwriting files in sequence -> temporary inconsistency)
- State -> nondeterministic builds -> not reproducible
- Different versions of a binary
- Package conflicts
- No rollbacks
- No configuration management

Nix(OS)

- Atomic upgrades/rollbacks (software & configuration)
- Multiple versions of a package (side-by-side, e.g. testing a new Apache version)
- Deterministic & Reproducible builds
- Reliable upgrades (and rollbacks - configuration bound to correct software version + service reloads/restarts)
- Reliable channel upgrades/rollbacks (e.g. 17.03 -> 17.09)
- Unprivileged users can securely install software

Being functional

- Classically: Imperative configuration
  - Stateful changes (-> dependency hell, inconsistent states, etc.)
- NixOS: Declarative configuration
  - Packages/Configuration = immutable values
  - (Complete) rebuilds instead of destructive updates
  - Referential transparency (~an expression always evaluates to the same result)

Problems

- Lacking manpower/workforce (e.g. for better testing/security/documentation)
- Not all packages are reproducible (2016: 12.8%)
- Running pre-compiled binaries
- Scripts with hard-coded paths don't work
- No GUI for package/configuration management
- No LTS releases or super stable (i.e. old :P) branches
- Not all use-cases or configuration options supported
  - Some tricks available + PRs welcome ;)

Nix

- A purely functional package manager (transparent source/binary deployment)
- Secure multi-user support
- Stores packages in the Nix store (`/nix/store` by default)
- Each package has it's own unique identifier/directory
  - E.g. `qn96dbgqdryaw38zw6v08da34q5v4qz3-git-repo-1.12.37` (cryptographic hash, name, version)
  - Enables multiple versions & binary substitutes
  - "Forces" complete dependencies

Nix expressions / Nix expression language

- A DSL (not a GPL!)
  - Describes graphs of build actions ("derivations")
  - Packages, compositions of packages, configurations, ...
- Dynamically typed ("Nix won't be complete until it has static typing." @edolstra) - https://typing-nix.regnat.ovh/
- Lazy (a very important feature!)
- Purely functional (no side-effects, immutable store)
- Turing complete (e.g. Dhall is not -> dhall-nix)

nix-repl

- Demo

Nixpkgs (the Nix packages collection)

- Main GitHub repository (permissive MIT/X11 license)
- Contains definitions of packages (Nix) and modules (NixOS)
- Also contains tests, library functions, etc.
- Different branches (rolling: master, stable: release-YY.MM)
- Build and tested by Hydra (+ uploaded to binary cache)
- Distributed through (nixpkgs-)channels (nixpkgs-unstable, nixos-unstable(-small), nixos-YY.MM(-small))

An example Nix package (pgpdump)

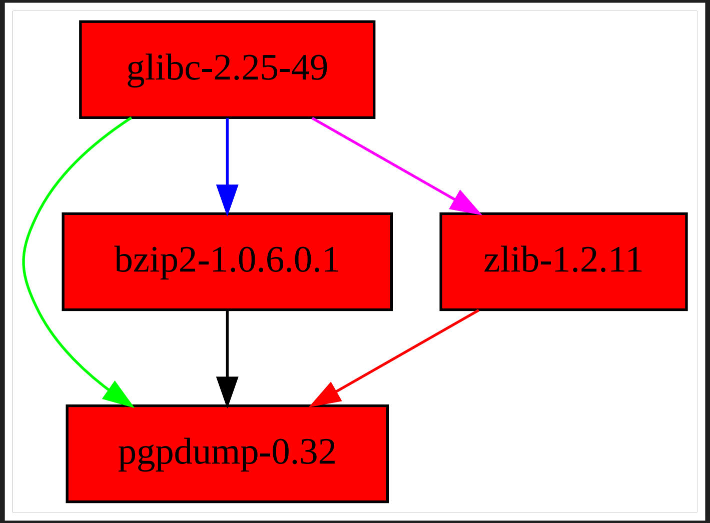
```
pgpdump = callPackage ../tools/security/pgpdump { };
```

```
{ stdenv, fetchFromGitHub
, supportCompressedPackets ? true, zlib, bzip2
}:

stdenv.mkDerivation rec {
  name = "pgpdump-${version}";
  version = "0.32";

  src = fetchFromGitHub {
    owner = "kazu-yamamoto";
    repo = "pgpdump";
    rev = "v${version}";
    sha256 = "1ip7q5sgh3nwdqbrzpp6sllkls5kma98kns53yspw1830xi1n8xc";
  };
```

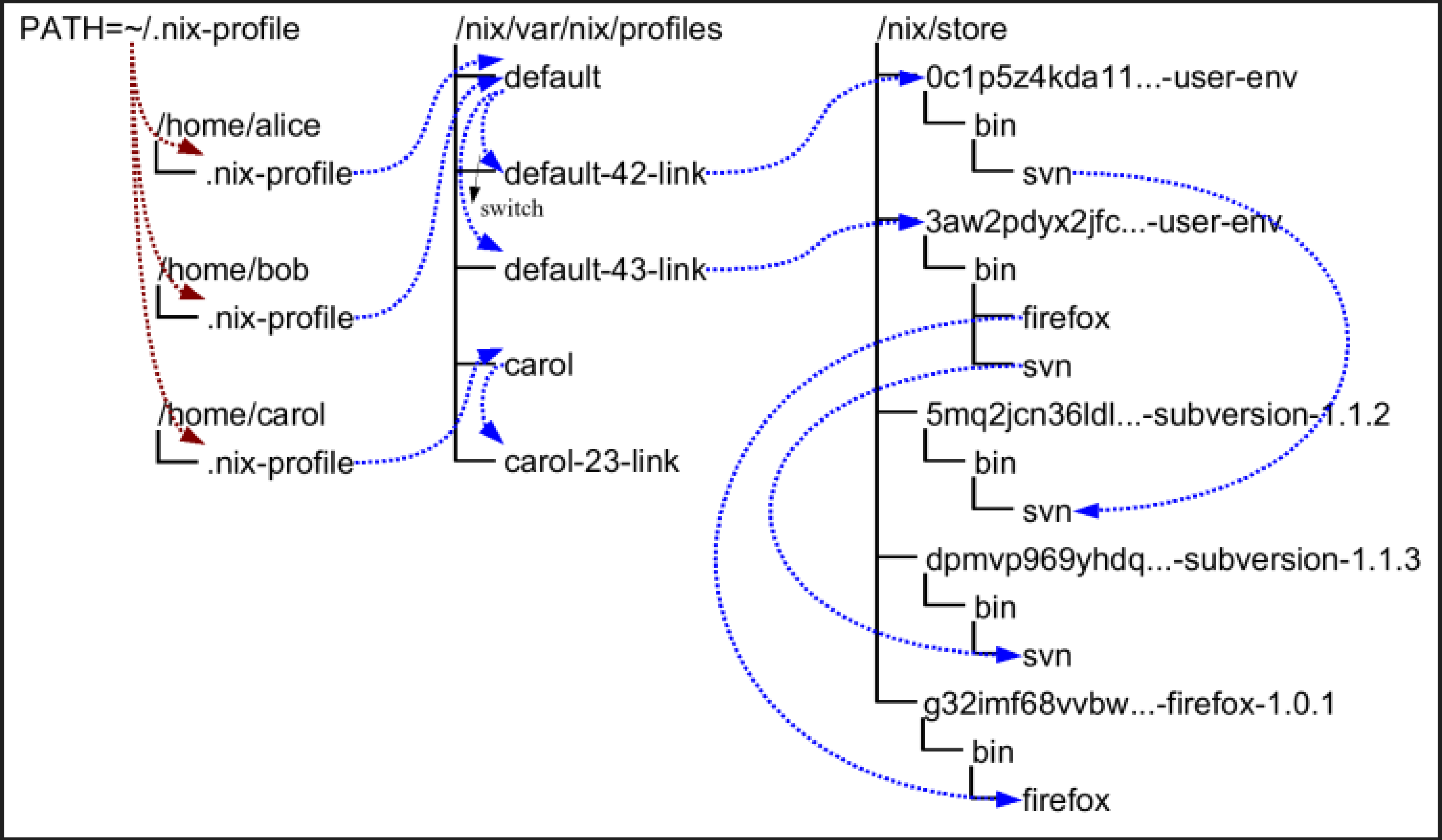
Dependency graphs

pgpdump's runtime dependencies:

```
nix-store -q --graph $(nix-store --realise $(nix-instantiate -A pgpdum
```



nix-env (manipulate or query Nix user environments)



https://nixos.org/nix/manual/figures/user-environments.png

nix-shell

- Demo

NixOS

- Implements a declarative and purely functional system configuration model
- Based on Nix (package + configuration management)
- NixOS modules (separation of concerns)
- Form the full "system configuration"

```
{ config, pkgs, ... }: {
  options = { nested attribute set of option declarations using mkOp
  config = { nested attribute set of option definitions };
}
```

An example NixOS module

```
{ config, lib, pkgs, ... }:

with lib;

let
  cfg = config.programs.vim;
in {
  options.programs.vim = {
    defaultEditor = mkOption {
      type = types.bool;
      default = false;
      description = ''
        When enabled, installs vim and configures vim to be the defaul
        using the EDITOR environment variable.
      '';
```

An example NixOS configuration

```
{ config, pkgs, ... }:

{
  system.stateVersion = "17.09";

  nix.useSandbox = true;

  boot.kernelPackages = pkgs.linuxPackages_latest;

  i18n = {
    consoleFont = "Lat2-Terminus16";
    consoleKeyMap = "de";
    defaultLocale = "en_US.UTF-8";
  };
```

nixos-container

- Demo

Nixpkgs overlays

```
self: super:
# self: Final package set / fixed-point result (use as dependencies)
# super: Previous evaluation result
{
  nix = super.nix.override {
    storeDir = "${<nix-dir>}/store";
    stateDir = "${<nix-dir>}/var";
  };
  boost = super.boost.override {
    python = self.python3;
  };
  rr = super.callPackage ./pkgs/rr {
    stdenv = self.stdenv_32bit;
  };
}
```

NixUP & co.

- Nix User Profile
- Manage $HOME

Community

- A great & kind community
- nix-devel mailing list
- Bugs and PRs via GitHub (Nixpkgs)
- #nixos on irc.freenode.net
- Blogs (NixOS planet)
- Local meetups (check meetup.com)
- NixCon
- Commercial support via consulting companies

NixCon

- Conference for Nix and NixOS
- Last in Munich in Oktober
- Next unknown

Learning

- Learn X in Y minutes, where X=nix
- A tour of Nix
  - By Joachim Schiele & Paul Seitz from Tübingen ;)
- Unofficial user's wiki
- Manuals (Nix, Nixpkgs, NixOS NixOps)

Nix Pills

- A ported version of the Nix Pills (a series of blog posts)
- A tutorial introduction into the Nix package manager and Nixpkgs package collection
- In the form of short chapters called 'pills'
- https://nixos.org/nixos/nix-pills/

Trying out Nix*

- Use Nix side-by-side with your regular package manager:

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

- Experiment with nix-env, nix-shell, nix-repl, etc.
- Try out NixOS (e.g. VirtualBox demo appliance)
- Install NixOS

Thank you :)

- Questions?
- Discussion