Warum ich jetzt NixOS benutze

Matthias Beyer

Furtwangen University

Unix Friends and User Group

mail@beyermatthias.de

11. April 2015

Overview

Nix - The purely functional package manager

- Atomare Updates

Nix - The purely functional package manager

- Atomare Updates
- Rollbacks

Nix - The purely functional package manager

- Atomare Updates
- Rollbacks
- Side-by-Side installation of multiple package versions

Nix - The purely functional package manager

- Atomare Updates
- Rollbacks
- Side-by-Side installation of multiple package versions
- Multi-User package management

Nix - The purely functional package manager

- Atomare Updates
- Rollbacks
- Side-by-Side installation of multiple package versions
- Multi-User package management
- Build Environments

Nix - The purely functional package manager

- Atomare Updates
- Rollbacks
- Side-by-Side installation of multiple package versions
- Multi-User package management
- Build Environments
- Reproduzierbares package management

Der /nix/store

/nix/store/b6q64986n0smwwfwrhg655lcg3pxhk56-git-2.1.4/

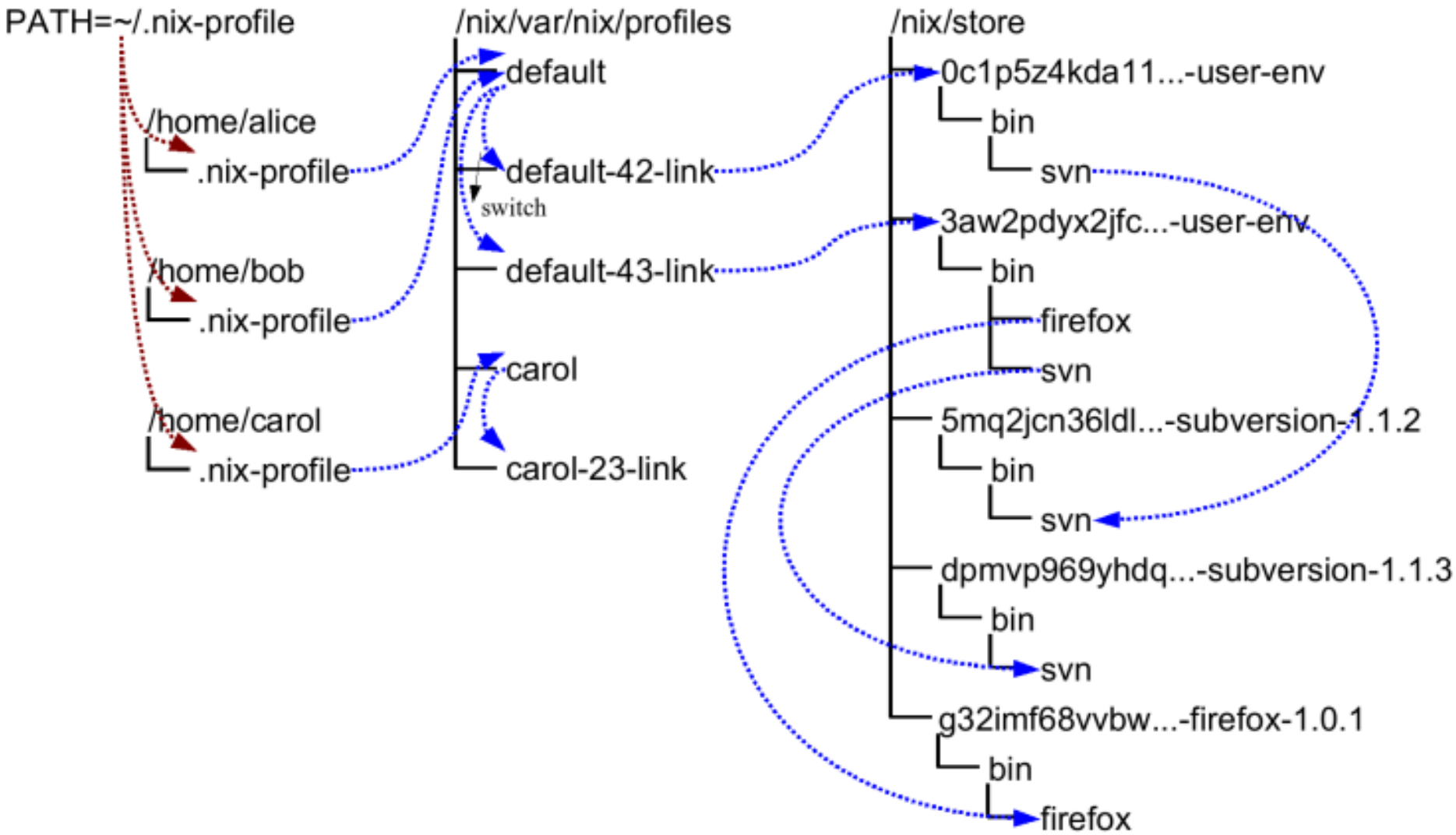
Der /nix/store

```
/nix/store/b6q64986n0smwwfwrhg655lcg3pxhk56-git-2.1.4/
    bin/
        git
        git-shell
        ...
    etc/
    lib/
    libexec/
    share/
```

Der /nix/store

```
$ cd /nix/store
$ ls | grep git
4mhm1cl2mx8x30xi7wazzyla7g86crb8-git-2.3.0
b6q64986n0smwwfwrhg655lcg3pxhk56-git-2.1.4
```

Und was benutz ich jetzt?



System/User/Development - Environments

- System-weites Env

$\rightarrow$ Reproduzierbar!

System/User/Development - Environments

- System-weites Env
- User-eigenes Env

$\rightarrow$ Reproduzierbar!

System/User/Development - Environments

- System-weites Env
- User-eigenes Env
- Projekt-eigenes Env

$\rightarrow$ Reproduzierbar!

Wie sieht so ein Package aus?

- Source-File = Funktion

Wie sieht so ein Package aus?

- Source-File = Funktion
- Returns "Derivation"

Wie sieht so ein Package aus?

- Source-File = Funktion
- Returns “Derivation”
- Abhängikeiten = Funktionsparameter

Wie sieht so ein Package aus?

"Top-level" File der Packete:

```
dwm = callPackage ../applications/window-managers/dwm {
  patches = config.dwm.patches or [];
};
```

Wie sieht so ein Package aus?

```
{stdenv, fetchurl, libX11, libXinerama, patches ? []}:

let
  name = "dwm-6.0";
in
stdenv.mkDerivation {
  inherit name;

  src = fetchurl {
    url = "http://dl.suckless.org/dwm/${name}.tar.gz";
    sha256 = "0mpbivy9j80l1jqq4bd4g4z8s5c54fxrjj44avmfwncjwqyli
  };

  buildInputs = [ libX11 libXinerama ];
  // ...
```

Wie sieht so ein Package aus?

```
  prePatch = ''sed -i "s@/usr/local@$out@" config.mk'';

  # Allow users set their own list of patches
  inherit patches;

  buildPhase = " make ";

  meta = {
    homepage = "www.suckless.org";
    description = "Dynamic window manager for X";
    license = stdenv.lib.licenses.mit;
    maintainers = with stdenv.lib.maintainers; [viric];
    platforms = with stdenv.lib.platforms; all;
  };
}
```

Channels

- “Versionen” von Nix packages (vergl. Ubuntu 14.04, 12.04)

Channels

- “Versionen” von Nix packages (vergl. Ubuntu 14.04, 12.04)
  - nixos-13.10
  - nixos-14.02
  - nixos-14.12
  - nixos-unstable
  - nixpkgs-unstable

Channels

- “Versionen” von Nix packages (vergl. Ubuntu 14.04, 12.04)
  - nixos-13.10
  - nixos-14.02
  - nixos-14.12
  - nixos-unstable
  - nixpkgs-unstable
- Systemweit

Channels

- “Versionen” von Nix packages (vergl. Ubuntu 14.04, 12.04)
  - nixos-13.10
  - nixos-14.02
  - nixos-14.12
  - nixos-unstable
  - nixpkgs-unstable
- Systemweit
- Per User

Channels

- “Versionen” von Nix packages (vergl. Ubuntu 14.04, 12.04)
  - nixos-13.10
  - nixos-14.02
  - nixos-14.12
  - nixos-unstable
  - nixpkgs-unstable
- Systemweit
- Per User
- Eigene Channels erstellen

NixOS!

Bis jetzt: Linux-Distro Unabhängig.
Nix läuft...

- Linux

NixOS!

Bis jetzt: Linux-Distro Unabhängig.
Nix läuft...

- Linux
- Mac OSX (und alle anderen Apfelvergewaltigungen)

NixOS!

Bis jetzt: Linux-Distro Unabhängig.
Nix läuft...

- Linux
- Mac OSX (und alle anderen Apfelvergewaltigungen)
- (theoretisch) BSD

NixOS!

Jetzt:

- System-Config
  - Bootloader

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS
  - Services

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS
  - Services
    - systemd
    - theoretisch erweiterbar auf andere init-systeme
    - (ursprünglich Upstart)

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS
  - Services
    - systemd
    - theoretisch erweiterbar auf andere init-systeme
    - (ursprünglich Upstart)
  - Treiber

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS
  - Services
    - systemd
    - theoretisch erweiterbar auf andere init-systeme
    - (ursprünglich Upstart)
  - Treiber
  - Kernel
    - . . . -Versionen
    - . . . -Patches

NixOS!

Jetzt:

- System-Config
  - Bootloader
  - FS
  - Services
    - systemd
    - theoretisch erweiterbar auf andere init-systeme
    - (ursprünglich Upstart)
  - Treiber
  - Kernel
    - . . . -Versionen
    - . . . -Patches
  - Packages

NixOS!

- User-Config

NixOS!

- User-Config
  - bashrc

NixOS!

- User-Config
  - bashrc
  - vimrc

NixOS!

- User-Config
  - bashrc
  - vimrc
  - ...

NixOS!

(Fast) Alles in einer Sprache konfigurierbar.

NixOps

- Deployment-Tool for NixOS

NixOps - Definition

```
{
  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 = "...";
    };
}
```

NixOps - Deploy

```
$ nixops create -d simple network.nix
$ nixops deploy -d simple
```

NixOps

- Amazon EC2 instances
- VirtualBox virtual machines
- Hetzner machines
- NixOS containers
- Many Amazon resources, such as S3 buckets, EC2 key pairs elastic IPs, ...

Mein Setup

(Siehe Terminal)

Und? Nix gelernt?

```
exit(EXIT_SUCCESS);
```