

# Using NixOS for declarative deployment and testing

Sander van der Burg   Eelco Dolstra

Delft University of Technology, EEMCS,  
Department of Software Technology

February 5, 2010



# Linux distributions

There are a wide range of Linux distributions available, each having different properties and goals.



## Software deployment

All of the activities that make a software system available for use  
*Carzaninga et al.*

## Activities

- Install a Linux distribution with some desired packages
- Adapt/tweak configuration files
- Install custom pieces of software
- Upgrade a system

## Single installation



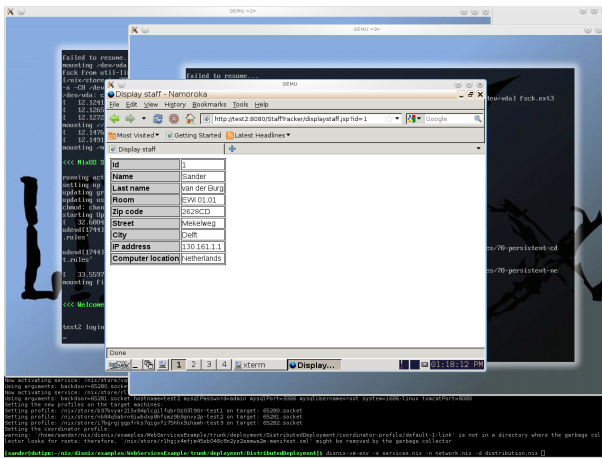
# Deployment scenario



- Multiple installations
- Machines are connected and dependent on each other

# Deployment scenario

## Virtual machines



The screenshot shows a terminal window with NixOS deployment logs and a web browser window displaying a table of staff information.

```
failed to resume...
mounting /dev/uda
fsck from ntfs-11
[...]
```

id	1
Name	Sander
Last name	van der Burg
Room	EWI 01.01
Zip code	2628CD
Street	Mekeelweg
City	Delft
IP address	130.161.1.1
Computer location	Netherlands

```
...
test2 login
...
new activating service: /nix/store/...
using arguments: /nix/store/...
new activating service: /nix/store/...
using arguments: /nix/store/...
setting the new profiles on the target machines
setting profile: /nix/store/... on target: 65280.socket
setting profile: /nix/store/... on target: 65281.socket
setting the coordinator profile
command: /home/sander/nix/dmns/examples/webServiceExample/trunk/deployment/DistributionDeployment/coordinator-profile/default-link is not in a directory where the garbage collector looks for roots, therefore, /nix/store/... might be removed by the garbage collector
sander@dmns:~$ nix -d /nix/store/... --services.nix -n network.nix -d distribution.nix
```

- Deploying a single machine is hard
  - Takes some effort
  - Upgrading may break the system
- Deploying a distributed environment is even harder
  - Machines may be dependent on each other, e.g. web application using a database
  - While upgrading, downtimes may occur
- Deploying (a network of) virtual machines is also hard
  - Takes quite some effort to perform system integration tests



A GNU/Linux distribution using the Nix package manager



# Nix store

Main idea: store all packages in isolation from each other:

```
/nix/store/rpdqxnilb0cg...  
-firefox-3.5.4
```

Paths contain a 160-bit **cryptographic hash** of all inputs used to build the package:

- Sources
- Libraries
- Compilers
- Build scripts
- ...

```
/nix/store  
├── 19w6773m1msy...-openssh-4.6p1  
│   ├── bin  
│   │   └── ssh  
│   └── sbin  
│       └── sshd  
├── smkabrbibqv7...-openssl-0.9.8e  
│   └── lib  
│       └── libssl.so.0.9.8  
├── c6jbqm2mc0a7...-zlib-1.2.3  
│   └── lib  
│       └── libz.so.1.2.3  
└── im276akmsrhv...-glibc-2.5  
    └── lib  
        └── libc.so.6
```

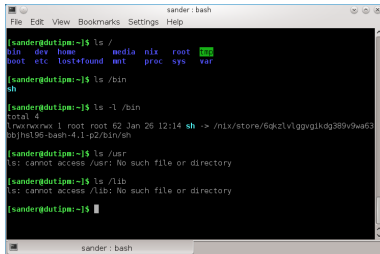
## openssh.nix

```
{ stdenv, fetchurl, openssl, zlib }:  
  
stdenv.mkDerivation {  
  name = "openssh-4.6p1";  
  src = fetchurl {  
    url = http://.../openssh-4.6p1.tar.gz;  
    sha256 = "0fpjlr3bfind0y94bk442x2p...";  
  };  
  buildCommand = ''  
    tar xjf $src  
    ./configure --prefix=$out --with-openssl=${openssl}  
    make; make install  
  '';  
}
```

## all-packages.nix

```
openssh = import ../tools/networking/openssh {  
  inherit fetchurl stdenv openssl zlib;  
};  
  
openssl = import ../development/libraries/openssl {  
  inherit fetchurl stdenv perl;  
};  
  
stdenv = ...;  
openssl = ...;  
zlib = ...;  
perl = ...;
```

- `nix-env -f all-packages.nix -iA openssh`
- Produces a `/nix/store/l9w6773m1msy...-openssh-4.6p1` package in the Nix store.



```
sander: bash
File Edit View Bookmarks Settings Help

[sander@dutipa:~]$ ls /
bin  dev  home  media  nix  root  [red]
boot  etc  lost+found  mit  proc  sys  var

[sander@dutipa:~]$ ls /bin
sh

[sander@dutipa:~]$ ls -l /bin
total 4
lrwxrwxrwx 1 root root 62 Jan 26 12:14 sh -> /nix/store/6qkzlvlgvgikdg389v9wa63
bbjhs196-bash-4.1-p2/bin/sh

[sander@dutipa:~]$ ls /usr
ls: cannot access /usr: No such file or directory

[sander@dutipa:~]$ ls /lib
ls: cannot access /lib: No such file or directory

[sander@dutipa:~]$
```

- In NixOS, all packages including the Linux kernel and configuration files are managed by Nix.
- NixOS does not have directories such as: `/lib` and `/usr`
- NixOS has a minimal `/bin` and `/etc`
- But NixOS is more than just a distribution managed by Nix

# NixOS configuration

```
/etc/nixos/configuration.nix
```

```
{pkgs, ...}:  
  
{  
  boot.loader.grub.device = "/dev/sda";  
  
  fileSystems = [ { mountPoint = "/"; device = "/dev/sda2"; } ];  
  swapDevices = [ { device = "/dev/sda1"; } ];  
  
  services = {  
    openssh.enable = true;  
  
    xserver = {  
      enable = true;  
      desktopManager.kde4.enable = true;  
    };  
  };  
  
  environment.systemPackages = [ pkgs.mc pkgs.firefox ];  
}
```

## nixos-rebuild switch

- Nix package manager builds a complete system configuration
  - Includes all packages and generates all configuration files, e.g. OpenSSH configuration
- Upgrades are (almost) atomic
  - Components are stored safely next to each other, due to hashes
  - No files are automatically removed or overwritten
- Users can switch to older generations of system configurations not garbage collected yet

# NixOS bootloader

GNU GRUB version 0.97 (636K lower / 129984K upper memory)

## NixOS - Default

### Windows

```
NixOS - Configuration 269 (2009-08-11 23:21:10 - 2.6.27.29-default)
NixOS - Configuration 268 (2009-08-11 18:24:09 - 2.6.27.29-default)
NixOS - Configuration 267 (2009-08-05 10:47:20 - 2.6.27.29-default)
NixOS - Configuration 266 (2009-08-05 10:35:27 - 2.6.27.29-default)
NixOS - Configuration 265 (2009-08-05 10:35:06 - 2.6.27.29-default)
NixOS - Configuration 264 (2009-08-04 15:27:25 - 2.6.27.29-default)
NixOS - Configuration 263 (2009-08-04 15:07:21 - 2.6.27.29-default)
NixOS - Configuration 262 (2009-08-04 14:11:27 - 2.6.27.29-default)
NixOS - Configuration 261 (2009-08-04 10:42:23 - 2.6.27.29-default)
NixOS - Configuration 260 (2009-08-04 10:29:25 - 2.6.27.29-default)
```

Use the ↑ and ↓ keys to select which entry is highlighted.  
Press enter to boot the selected OS, 'e' to edit the  
commands before booting, or 'c' for a command-line.

GNU/Linux

# Distributed deployment

- NixOS has good properties for deployment of a single system
- Can we extend these properties to distributed systems?



# Motivating example: Trac

(please configure the [header\_logo] section in trac.ini)

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

---

Wiki | Timeline | Roadmap | View Tickets | Search

[Start Page](#) | [Index](#) | [History](#) | [Last Change](#)

## Welcome to Trac 0.11.7

Trac is a **minimalistic** approach to **web-based** management of **software projects**. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.

All aspects of Trac have been designed with the single goal to **help developers write great software** while **staying out of the way** and imposing as little as possible on a team's established process and culture.

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. [WikiFormatting](#) will give you a detailed description of available Wiki formatting commands.

"[trac-admin yourenvdir initenv](#)" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains [documentation](#) to help you get started with your project.

You can use [trac-admin](#) to configure [Trac](#) to better fit your project, especially in regard to *components*, *versions* and *milestones*.

[TracGuide](#) is a good place to start.

Enjoy!

*The Trac Team*

Trac can be deployed in a *distributed* environment:

- Subversion server
- Database server
- Web server

# Distributed NixOS configuration

```
network.nix
```

```
{ storage = {pkgs, ...}:
  {
    services.nfsKernel.server.enable = true; ...
  };

postgresql = {pkgs, ...}:
  {
    services.postgresql.enable = true; ...
  };

webserver = {pkgs, ...}:
  {
    fileSystems = [
      { mountPoint = "/repos"; device = "storage:/repos"; } ];
    services.httpd.enable = true;
    services.httpd.extraSubservices = [ { serviceType = "trac"; } ]; ...
  };

  ...
}
```

nixos-deploy-network network.nix

- Build system configurations by the Nix package manager
- Transfer complete system and all dependencies to target machines in the network
  - Efficient: only missing store paths must be transferred
  - Safe: Existing configuration is not affected, because no files are overwritten or removed
- Activate new system configuration
  - In case of a failure, roll back all configurations
  - Relatively cheap operation, because old configuration is stored next to new configuration

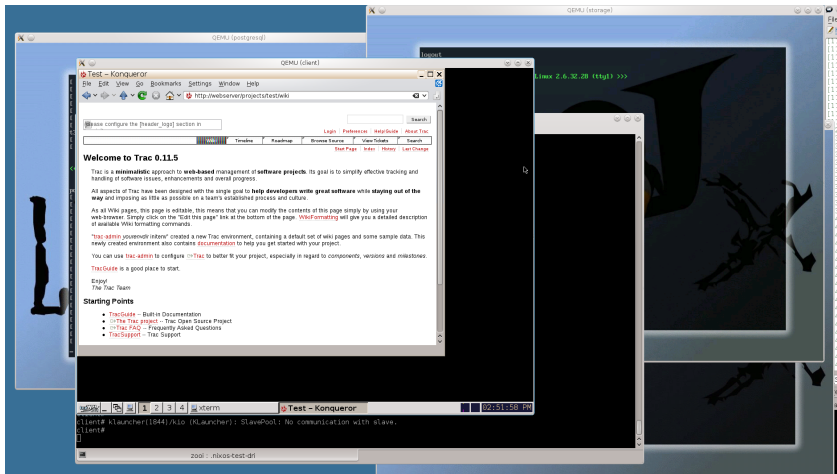
```
nixos-build-vms network.nix; ./result/bin/nixos-run-vms
```

- Builds a network of QEMU-KVM virtual machines closely resembling the network of NixOS configurations
- We don't create disk images
- The VM mounts the Nix store of the host system using SMB/CIFS

```
nixos-build-vms network.nix; ./result/bin/nixos-run-vms
```

- Possible because complete configuration is in the Nix store
- This is efficient and safe due to the nature of the Nix store
  - Components with same hash codes are shared between VMs
  - The hash part of the name isolates components from each other
- Difficult to do for imperative Linux distributions, which have `/etc`, `/usr`, `/lib` directories.

# Virtualization



## trac.nix

```
testScript = ''
  $postgresql→waitForJob("postgresql");
  $postgresql→mustSucceed("createdb trac");

  $webserver→mustSucceed("mkdir -p /repos/trac");
  $webserver→mustSucceed("svnadmin create /repos/trac");

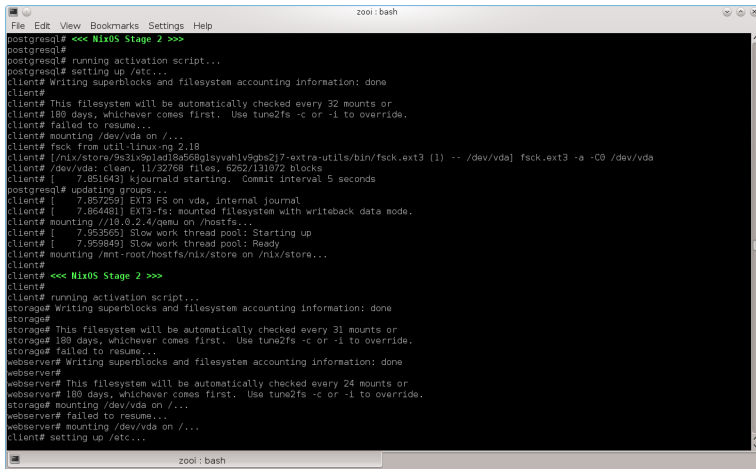
  $webserver→waitForFile("/var/trac");
  $webserver→mustSucceed("mkdir -p /var/trac/projects/test");
  $webserver→mustSucceed("trac-admin /var/trac/projects/test initenv ".
    "Test postgres://root\@postgresql/trac svn /repos/trac");

  $client→waitForX;
  $client→execute("konqueror http://webserver/projects/test &");
  $client→waitForWindow(qr/Test.*Konqueror/);

  $client→screenshot("screen");
'';
```



## nix-build tests.nix -A trac



```
zooi: bash
File Edit View Bookmarks Settings Help
postgreSQL# <<< NixOS Stage 2 >>>
postgreSQL#
postgreSQL# running activation script...
postgreSQL# setting up /etc...
client# Writing superblocks and filesystem accounting information: done
client#
client# This filesystem will be automatically checked every 32 mounts or
client# 180 days, whichever comes first. Use tune2fs -c or -i to override,
client# failed to resume...
client# mounting /dev/vda on /...
client# fsck from util-linux-ng 2.18
client# [/nix/store/9s3lx9pld18a568glsyvahlv9gbs2j7-extra-utils/bin/fsck.ext3 (1) -- /dev/vda) fsck.ext3 -a -C0 /dev/vda
client# /dev/vda: clean, 11/32768 files, 6262/131072 blocks
client# [ 7.851649] kjournald starting. Commit interval 5 seconds
postgreSQL# updating groups...
client# [ 7.857259] EXT3 FS on vda, internal journal
client# [ 7.864481] EXT3-fs: mounted filesystem with writeback data mode.
client# mounting //18.0.2.4/qemu on /hostfs...
client# [ 7.953565] Slow work thread pool: Starting up
client# [ 7.959849] Slow work thread pool: Ready
client# mounting /mnt-root/hostfs/nix/store on /nix/store...
client#
client# <<< NixOS Stage 2 >>>
client#
client# running activation script...
storage# Writing superblocks and filesystem accounting information: done
storage#
storage# This filesystem will be automatically checked every 31 mounts or
storage# 180 days, whichever comes first. Use tune2fs -c or -i to override,
storage# failed to resume...
webservers# Writing superblocks and filesystem accounting information: done
webservers#
webservers# This filesystem will be automatically checked every 24 mounts or
webservers# 180 days, whichever comes first. Use tune2fs -c or -i to override,
storage# mounting /dev/vda on /...
webservers# failed to resume...
webservers# mounting /dev/vda on /...
client# setting up /etc...
```



- Distributed deployment of a Hydra build environment
- Continuous integration and testing of NixOS
  - NixOS installer
  - OpenSSH
  - Trac
  - NFS server
- Continuous integration and testing of various GNU projects
  - Install NixOS system with bleeding edge glibc
- Other free software projects

- Examples:
  - Cfengine
  - Stork
- Related work uses *convergent* models
- NixOS models are *congruent*

- *NixOS*. A GNU/Linux distribution used to reliably deploy a complete system from a declarative specification
- `nixos-deploy-network`. Efficiently/Reliably deploy a network of NixOS machines
- `nixos-build-vms`. Efficiently generate a network of cheap NixOS virtual machines instances
- NixOS test driver. Perform distributed test cases in a network of NixOS virtual machines

- NixOS website: <http://nixos.org>
  - *Nix*. A purely functional package manager
  - *Nixpkgs*. Nix packages collection
  - *NixOS*. Nix based GNU/Linux distribution
  - *Hydra*. Nix based continuous build and integration server
  - *Disnix*. Nix based distributed service deployment
- Software available under free and open-source licenses (LGPL/X11)

- Nix package manager can be used on any Linux system, FreeBSD, OpenSolaris, Darwin and Cygwin
- Virtualization features can be used on *any* Linux system running the Nix package manager and KVM.

# Questions