The Rise of Configuration Management

Um in NixOS einzusteigen, müssen wir kurz etwas ausholen. Es war einmal ein armer Systemadmin der mit seinen Aufgaben wachsen musste.

Warning: There might be dragons.

Convincing Operating Systems

Die 90er. Hart. Rau. Dein OS wird *manuell* via einzelner Kommandos konfiguriert.

```
ssh user@server
vim /etc/postfix/main.cf
/etc/init.d/postfix restart
```

Du loggst dich ein, änderst Config. Restartest deinen Dienst.

Würg

The Rise of Configuration Management

2000.

Der Durchschnittsadmin peitscht immer mehr Server in Clustern vor sich her. Repitative Arbeit in einem Team von Admins.

- Login
- Change
- Repeat

Viele schreiben Shell Skripte in Not und Verzweiflung um nicht der Tippgicht zu verfallen.

The Rise of Configuration Management

Deine Daemons, das Internet (curl |bash), deine Kollegen* werden alles tun um den State deiner Systeme alternieren. Ohne das du es mitbekommst oder irgendwann nochmal weisst.

Die Bash History ist *keine* Dokumentation!

*Kollegen: Du selbst mit eingeschlossen!

The Rise of Configuration Management

Neue Tools! Als weitere Evolutionsstufe nach Shell Scripts haben sich schlaue Köpfe eigene DSLs ausgedacht um Systeme beschreiben zu können.

- CFEngine (1993) Oh gott, bitte nicht.
- Puppet (2005)
- Chef (2009)
- Salt (2011)
- Ansible (2012)

Jedes dieser Tools verfolgt ein anderes Ziel aber im Grunde alles Konfigurations Management.

The Rise of Configuration Management

Sie definieren wie das System aussehen soll und "Versprechen" die Umsetzung.

Use Cases:

- File erstellen
- User anlegen
- Dienst muss gestartet sein
- Exec eines bestimmten Kommands

CfgMgmt - Beispiel 1

Zum Beispiel in Puppet einen neuen User anlegen:

```
$ vim /etc/puppet/manifest/site.pp
```

```
user { 'bernd':
  ensure     => present,
  uid        => 1001,
  home       => /home/bernd,
}
```

CfgMgmt - Beispiel 2

oder in Ansible ein File erstellen:

```
vim ~/ansible/main.yaml
```

```
- file:
```

```
    path: /etc/foo.conf
    owner: foo
    group: foo
    mode: 0644
```

The Rise of Configuration Management

Step 1: Katalog aus Config-File(s) zusammen bauen Step 2: Auf Zielhost Katalog (State) aufbauen Step 3: Unterschiede identifizieren (Diff beider Kataloge) Step 4: Korrekturen durchführen auf Zielhost

Paper zu Promise Theory

NixOS

Was NixOS im Kern ausmacht ist, dass es Configmanagement und das OS verheiratet.

Man muss es sich so vorstellen als wäre das Config Management das einzige Interface wie man mit dem OS kommuniziert, anstatt ein Tool zu anzuweisen die Arbeit für einen zu erledigen.

NixOS

Warum ist das jetzt geil?

- Verlässliche atomare Upgrades
- Rollbacks
- Reproduzierbare System Konfiguration

NixOS Zitat aus dem IRC

> Setting up NixOS to me is like setting up Arch but you're guaranteed to only need to do it once
>
> jD91mZM2

NixOS: Configuration Management!

Gehen wir von einem Plain NixOS (frisch installiert) aus.

In /etc/nixos/configuration.nix ist der Einstiegspunkt

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

{
  services.openssh.enable = true;
  services.openssh.permitRootLogin = "no";

  # Use the GRUB 2 boot loader.
  boot.loader.grub {
    enable = true;
    version = 2;
    device = "/dev/sda";
  }
}
```

Und das würde theoretisch ausreichen um ein NixOS zu betreiben.

NixOS: Configuration Management!

Um das System zu verändern

```
vim /etc/nixos/configuration.nix
nixos-rebuild switch
```

Demo Time!

NixOS: Internals!

Was passiert da jetzt unter der Haube

```
$ ls /nix/var/nix/profiles/
.
..
system #system-73-link
system-10-link #/nix/store/2ss198p8l60kxzyh20zm9wg1rh34n6g1-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
system-11-link #/nix/store/81fm4dxwk6nr46cdl78c24nk603r163m-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
system-12-link #/nix/store/7h3h5q35mvl4dm38253saz1bf7mg9yi2-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
system-13-link #/nix/store/81fm4dxwk6nr46cdl78c24nk603r163m-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
system-14-link #/nix/store/5ss77ahipw995ysrllxq84lsjx8gkzmw-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
system-1-link #/nix/store/jw12px138dybgir3dqik670gfx1xswg0-nixos-system-mqtt.k4cg.org-17.03.1825.df7e518568
```

NixOS: Rollback!

Das tolle daran, Angewandte Changes können (in bestimmten Maße) zurückgerollt werden.

```
nixos-rebuild switch --rollback
```

Ohne eben die natürlichen Ausnahmen (DB Version Upgrade mit Schema Changes... usw)

Was passiert da:

- Daemons werden wieder ausgeschaltet
- Alte config Versionen werden wieder verwendet
- usw usw...

Eben Alles was nixos ändert wird rückgängig gemacht, so ein Rollback der auch wirklich funktioniert!

NixOS: Wie funktioniert das technisch alles?!

Symlinks, Symlinks, Symlinks\

```
$ which git
/run/current-system/sw/bin/git
$ l /run/current-system/sw/bin/git
/run/current-system/sw/bin/git #/nix/store/gc80crp6ddm11h6zh3vnkzlbacxnswak-git-2.14.1/bin/git
```

NixOS: Wie funktioniert das technisch alles?!

Ein `ps` kann unter Umständen anfangs etwas verwirrend aussehen.

```
httpd -f /nix/store/yhdz9vwkm35r9f3fqj5dp9dzyg0154br-httpd.conf
 \_ httpd -f /nix/store/yhdz9vwkm35r9f3fqj5dp9dzyg0154br-httpd.conf
 \_ httpd -f /nix/store/yhdz9vwkm35r9f3fqj5dp9dzyg0154br-httpd.conf
 \_ httpd -f /nix/store/yhdz9vwkm35r9f3fqj5dp9dzyg0154br-httpd.conf
 \_ httpd -f /nix/store/yhdz9vwkm35r9f3fqj5dp9dzyg0154br-httpd.conf
dhcpcd -w --quiet --config /nix/store/65faqsgxbxgdpb9xj5q9r780lj0jcdwj-dhcpcd.conf
nix-daemon --daemon
/nix/store/ick7hkazslqz2b75nfgfr0fz5cgal3c8-cron-4.1/bin/cron -n
/nix/store/8dshy3jqzkw2c73czw6kbw0msx9ivfi6-systemd-232/lib/systemd/systemd --user
 \_ (sd-pam
 \_ /nix/store/sshcjcvscx2kxy3x11rbqsxd4g780ln4-openssh-7.4p1/bin/ssh-agent -a /run/user/1000/ssh-agent
/nix/store/pcl82rq2aw28015pqi27hwgrp5znrfq3-mosquitto-1.4.14/bin/mosquitto -c /nix/store/m699paqbxybhfdmlnkn53q2sc6r69ni5-mosquitto.conf -d
```

Nix: Der Paket Manager

NixOS basiert auf Nix, dem Paket Manager.

Nix: User-local Installs!

Ich muss unter NixOS allerdings kein `root` sein um neue Pakete zu installieren.

```
$ su bernd
$ nixenv -iA nixos.curl
$ curl http://russenschlampen.de
```

Nix: User-local Installs!

Was passiert ist, dass unter `/nix/store` die Binaries abgelegt werden.

```
$ ls /nix/store/ | grep curl | grep bin
dk2668744spfp0vy14c39nhnbmmc85ws-curl-7.56.0-bin
qs3lyiysdpbbknxy3q9vc585adkvam74-curl-7.56.1-bin
r8inh6r6a5nz3paskiz9j9pnn7dyx30j-curl-7.55.0-bin
```

Jeder User hat einen eigenen PATH mit Binard

```
/run/current-system/sw/bin
/home/bernd/.nix-profile/bin
```

Nix: User-local Installs!

In diesem Folder ist dann, vereinfacht gesagt nichts anderes als Symlink

```
$ ls /home/bernd/.nix-profile/bin
curl #/nix/store/qs3lyiysdpbbknxy3q9vc585adkvam74-curl-7.56.1-bin/bin/curl
git #/nix/store/9iyw9xxh6bpvp0g8zamsc5264mykfbps-git-2.14.1/bin/git
influx #/nix/store/ny1gm0r14z1am0vfi8rn9z7l1xpyi555-influxdb-1.0.2-bin/bin/influx
influxd #/nix/store/ny1gm0r14z1am0vfi8rn9z7l1xpyi555-influxdb-1.0.2-bin/bin/influxd
iotop #/nix/store/d4qqyhigib69s5fni6srq0bf9higb608-iotop-0.6/bin/iotop
lynx #/nix/store/anz7v2ijmfi4mb82jilb17kjr22jggps-lynx-2.8.9dev.11/bin/lynx
swaks #/nix/store/hczryibja30py3s07073d2jwm8vrlh7k-swaks-20170101.0/bin/swaks
telnet #/nix/store/sh7lppfxk3w1pigrclsqw3dq9g81bavh-telnet-1.2/bin/telnet
```

Nix: User local Rollbacks!

Diese ganze Rollback Geschichte funktioniert natürlich auch bei User Paketen.

```
$ which telnet
which: no telnet in PATH

$ nix-env -iA nixos.telnet
installing â€˜telnet-1.2â€™
building path(s) â€˜/nix/store/pkl7rpx907dxlv28byblb9lrj6khymgp-user-environmentâ€™
created 268 symlinks in user environment

$ which telnet
/home/noqqe/.nix-profile/bin/telnet
```

```
$ nix-env --rollback
switching from generation 11 to 10

$ which telnet
which: no telnet in PATH
```

Nix: User local Rollbacks!

```
$ nix-env --list-generations
   1   2018-02-22 10:03:38
   2   2018-02-22 10:04:04
   3   2018-03-19 21:00:20
   4   2018-03-19 21:04:03
   5   2018-04-02 22:19:55
   6   2018-04-12 21:04:17
   7   2018-04-12 21:04:37
   8   2018-04-12 21:05:38
   9   2018-11-12 20:48:22
  10   2018-11-12 20:50:24   (current)
  11   2018-11-12 20:52:00
```

Nix: Package Manager

Lässt sich auch auf Linux/macOS installieren.

Obacht, `curl | bash` ahead!

Das heisst man hat dann all diese Vorteile von local Install und Versionen.

Nix Eco System: Überblick

nix - Package Manager nixos - Operating System nixops - Service Orchestration Tool

---

NixOps: Abstraktion ist die Lösung aller Probleme

NixOps ist sozusagen noch ein Layer mehr. Also mehrere Maschinen maintainen. Sozusagen eine Mischung aus Ansible und Terraform.

```
{
  webserver =
    { deployment.targetEnv = "virtualbox";
      services.httpd.enable = true;
    };

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

Generell habe ich damit aber keine Erfahrung.

Was machen wir in der K4CG mit NixOS?

Demo Time!

Meta Slide

Danke!

Für diesen Talk habe ich mpd benutzt und würde gerne erfahren wie ihr diese Art der Darstellung fandet.

Außerdem lenkt das wunderbar von inhaltlichen Fragen zum Vortrag ab. noqqe@nuel-102806 ~/C/nixos-talk (master %)>