Linux 2016: nftables, iproute2 & Co

Gnu's Not Unix
Gnu's Not Únix
Gnu's Not Ŭnix

2016061404 / Stephan Seitz <s.seitz@heinlein-support.de>

Wer ist die Heinlein Support GmbH?

- Wir bieten seit 20 Jahren Wissen und Erfahrung rund um Linux-Server und E-Mails
- IT-Consulting und 24/7 Linux-Support mit 20 Mitarbeitern
- Eigener Betrieb eines ISPs seit 1992
- Täglich tiefe Einblicke in die Herzen der IT aller Unternehmensgrößen

Inhalt

- Linux ./. *BSD ./. *nix
  - Veränderungen und zeitgemässe Änderungen
- Netzwerk
  - ipchains → ip*tables / arptables / ebtables → nftables
  - ifconfig / route / arp / brctl / vconfig → ip
  - Iproute2 et al.
  - NetworkManager
  - firewalld
- INIT-System
  - sysvinit → SMF (leider nicht...) → upstart → systemd
  - daemontools
- Dateisysteme
  - btrfs
  - ZFS
  - gluster, CephFS

GNU/Linux

- 1992
  - Basis für x86-Rechner
  - Fun Fact: IPv6 seit Version 2.2.0 :)
- 2016
  - Desktop-Rechner
    - x86. amd64, ARM
  - Server-Systeme
    - x86, amd64, x64, ARM, sparc, ...
  - Internet Of Things
    - ARM, etc...
  - Mobiles, Tablets, Wearables(?)
  - Infrastruktur
    - Steuer- / Regelsysteme
    - Switches / Router
- Diversifizierung → Unterstützung für spezialisierte Hardware

Stephan Seitz <s.seitz@heinlein-support.de>

Top 1

Netzwerk

Netzwerk

- Beispiel ifconfig ./. ip
  - Netzwerk Geräteverwaltung L2 + L3

```
ifconfig
```

```
ip addr show
ip link show
```

  - Geräte / Link aktivieren

```
ifconfig eth0 up
```

```
ip link set eth0 up
```

  - IPv4-Adressen verwalten

```
ifconfig eth0 192.168.123.1 netmask 255.255.255.0
```

```
ip address add 192.168.123.1/24 dev eth0
```

Netzwerk - IPv4 & ARP

- Beispiel ifconfig ./. ip
  - IP-Adresse entfernen

```
~~ifconfig~~
```

```
ip address del 192.168.123.1/24 dev eth0
```

  - Weitere IP auf einen Adapter binden (Bei ifconfig via „Alias“)

```
ifconfig eth0:alias 192.168.123.2 netmask 255.255.255.255
```

```
ip address add 192.168.123.2/32 dev eth0 label eth0:alias
```

- Beispiel arp ./. ip

```
arp -i eth0 -s 192.168.123.1 00:16:3E:11:22:33
```

```
ip neigh add 192.168.123.1 lladdr 00:16:3E:11:22:33 \
   nud permanent dev eth0
```

Netzwerk - IPv6 & ND

- Beispiel ifconfig ./. ip
  - IPv6-Adresse hinzufügen

```
ifconfig eth0 inet6 2003:abcd::1 netmask 64
```

```
ip -6 address add 2003:abcd::1/64 dev eth0
```

  - Generell lassen sich ohne Umweg über Alias-Adapter mehrere IPv4- als auch IPv6-Adressen binden.
  - Die Befehlsfolge für IPv6 und IPv4 ist im Wesentlichen identisch. Durch die Angabe des Parameters -4 oder -6 wird die Protokollfamilie inet bzw. inet6 gewählt.

Netzwerk - VLAN & NIC-Bonding

- Beispiel vconfig ./. ip

```
vconfig add eth0 42
```

```
ip link add link eth0 name eth0.42 vlan id 42
```

- Beispiel ifenslave ./. ip

```
modprobe bonding mode=802.3ad miimon=100
ifconfig bond0 192.168.123.1 netmask 255.255.255.0 up
ifenslave bond0 eth0
ifenslave bond0 eth1
```

```
modprobe bonding mode=802.3ad miimon=100
ip link set bond0 up
ip address add 192.168.123.1/24 dev bond0
ip link set eth0 master bond0
ip link set eth1 master bond0
```

Stephan Seitz <s.seitz@heinlein-support.de>

Netzwerk - VLAN-Stacking

- QinQ / 802.1ad ?

```
~~vconfig~~
```

  - Ohne netlink / iproute2 nicht darstellbar.

```
ip link add name eth0.42 link eth0 \
   type vlan proto 802.1ad id 42

ip link add name eth0.42.23 link eth0.42 \
   type vlan proto 802.1q id 23
```

  - Seit Kernel 3.10 verfügbar.
  - Benötigt 4 Bytes, d.h. MTU für das gekapselte VLAN-Interface muss angepasst werden.

Stephan Seitz <s.seitz@heinlein-support.de>

Netzwerk - Bridge

- Beispiel brctl ./. ip
  - Bridge erzeugen und ein Interface binden

```
brctl addbr br0
brctl addif br0 eth0
```

```
ip link add name br0 type bridge
ip link set br0 up
ip link set eth0 up
ip link set eth0 master br0
```

  - Warum ist das einfacher? :-)
    - Nur ein Werkzeug
    - Konsequenter Befehlsaufbau

Netzwerk - Tunnel

- Fun with iproute2
  - Beispiel: Erzeugen eines L2TP over IP Tunnels

```
ip l2tp add tunnel \
   tunnel_id 1 \
   peer_tunnel_id 2 \
   encap ip \
   local 80.249.10.2 \
   remote 213.203.238.2

ip l2tp add session tunnel_id 1 \
   session_id 10 \
   Peer_session_id 10
```

  - Auch weitere Tunnel wie GRE oder TUN/TAP-Adapter

```
ip tuntap add dev tun0 mode tun
ip tuntap add dev tap0 mode tap
```

Netzwerk - ... geht da noch mehr?

- Verwendung mehrerer Routing-Tabellen
  - 1. Tabellendefinition

```
# cat /etc/iproute2/rt_tables
255     local       # Reserviert
254     main        # Reserviert
253     default     # Reserviert
0       unspec      # Reserviert
8       mgmt        # weitere Routing-Tabelle
```

  - 2. Routing-Tabelle aufbauen (wie üblich, mit Angabe der Tabelle)

```
ip route add 10.97.64.0/18 dev eth0 src 10.97.66.123 table mgmt
ip route add default via 10.97.64.1 table mgmt
```

  - 3. Regel erstellen, wann diese Tabelle verwendet werden soll

```
ip rule add from 10.97.66.123 lookup mgmt
```

Netzwerk - iptables ./. nftables

- Beispiel: TCP an Port 22 Loggen und verwerfen.
  - iptables (bekannt...)

```
iptables -A INPUT -p tcp —dport 22 -j LOG
iptables -A INPUT -p tcp —dport 22 -j LOG
```

  - nft - das nftables CLI-Werkzeug

```
nft add rule filter input tcp dport 22 log drop
```

  - Syntax ist sehr ähnlich dem ip-Befehl aufgebaut.
  - Was fällt bereits auf?
    - Zwei Targets in einer Regel.

Stephan Seitz <s.seitz@heinlein-support.de>

Netzwerk - iptables ./. nftables

- Beispiel: Angabe von Listen bei der Verwendung von nft
  - ip6tables mit Hilfe von multiport

```
ip6tables -A INPUT -p tcp \
    -m multiport –dports 23,80,443 -j ACCEPT
```

  - nft - generische Angabe

```
nft add rule ip6 filter input \
    tcp dport {telnet, http, https} accept
```

- Beispiel: Listen bei der Angabe von icmp-Typen

```
nft add rule ip6 filter input \
    icmp type {nd-neighbor-solicit, \
               nd-router-advert, \
               nd-neighbor-advert, \
               echo-request} accept
```

  - Bei ip6tables wären vier getrennte Statements notwendig

Netzwerk - iptables ./. nftables

- Beispiel: Verwendung von benannten Attributen
  - nft bietet eine interaktive Shell

```
# nft -i # Interaktiv
nft> add set global ipv4_ad { type ipv4_address;}
nft> add element global ipv4_ad {192.168.123.4, 192.168.123.5}
nft> add rule ip global filter ip saddr @ipv4_ad drop
```

- Beispiel: Mapping (z.B. Anwenden von Regeln auf neue Adapter)

```
nft> add map filter jump_map { type ifindex : verdict; }
nft> add element filter jump_map { eth0 : jump low_sec; }
nft> add element filter jump_map { eth1 : jump high_sec; }
nft> add rule filter input iif vmap @jump_map
nft> # Erweitern des Filters durch ein neues Element
nft> add element filter jump_map { tap0 : jump low_sec; }
```

Netzwerk - Distributionen...

- RedHat / CentOS / ScientificLinux / Oracle Linux 7
  - NetworkManager nm-cli
    - Von Desktop-Distributionen bereits bekannt.
  - firewalld
    - Adaptive Firewall, verwendet „am Ende“ iptables.
- SuSE Leap / SLED
  - wicked
    - Neuentwicklung eines dhcp-client (in Entwicklung)
- Ubuntu >= 14.04
  - ufw (ubiquitous firewall)
    - Adaptive Firewall, verwendet ebenfalls iptables.
- ...

Stephan Seitz <s.seitz@heinlein-support.de>

pid 1

das INIT-System

SysVinit

- Unix System V
  - Übliche Shell-Skripte
  - RunLevel-Konzept
  - Statisch, d.h. keine impliziten Abhängigkeiten
    - Explizit durch weitere Shell-Parser und komplexere Skripte
  - Nicht trivial für weitere Dienste erweiterbar.
    - /etc/init.d/mysql-server als Beispiel für komplexe Aktionen

systemd

- Sauberer „stateforward“ Aufbau
- Vereinfachter Boot-Vorgang
- Parallele und konkurrierende Aktionen
- Klar definierte API
- Einfache unit-Syntax
- Modularer Aufbau
- Geringer Speicherverbrauch
- Einfache Formulierung von Abhängigkeiten
- Konfigurationsdateien anstelle von Shell-Skripten
- Kommunikation über Unix-Domain-Sockets
- Job-Scheduling durch systemd-Kalender-Timer
- Event-Logging via journald. Syslog ist nach wie vor möglich.
- Logfiles sind binär.
- Prozessverfolgung über cgroups anstelle von PIDs
- Benutzeranmeldungen via systemd-logind
- Desktop-Integration via d-bus (ohne PolicyKit)

systemd - Konfigurationsbeispiel

→ Unit-File **/etc/systemd/system/helloworld.service**

```
[Unit]
Description=MyHelloWorldInADocker
After=docker.service
Requires=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill busybox1
ExecStartPre=-/usr/bin/docker rm busybox1
ExecStartPre=/usr/bin/docker pull busybox
ExecStart=/usr/bin/docker run --name busybox1 \
    busybox /bin/sh -c "while true; do \
        echo Hello World; sleep 1; \
    done"

[Install]
WantedBy=multi-user.target
```

systemd - Konfigurationsbeispiel

→ Service aktivieren und starten

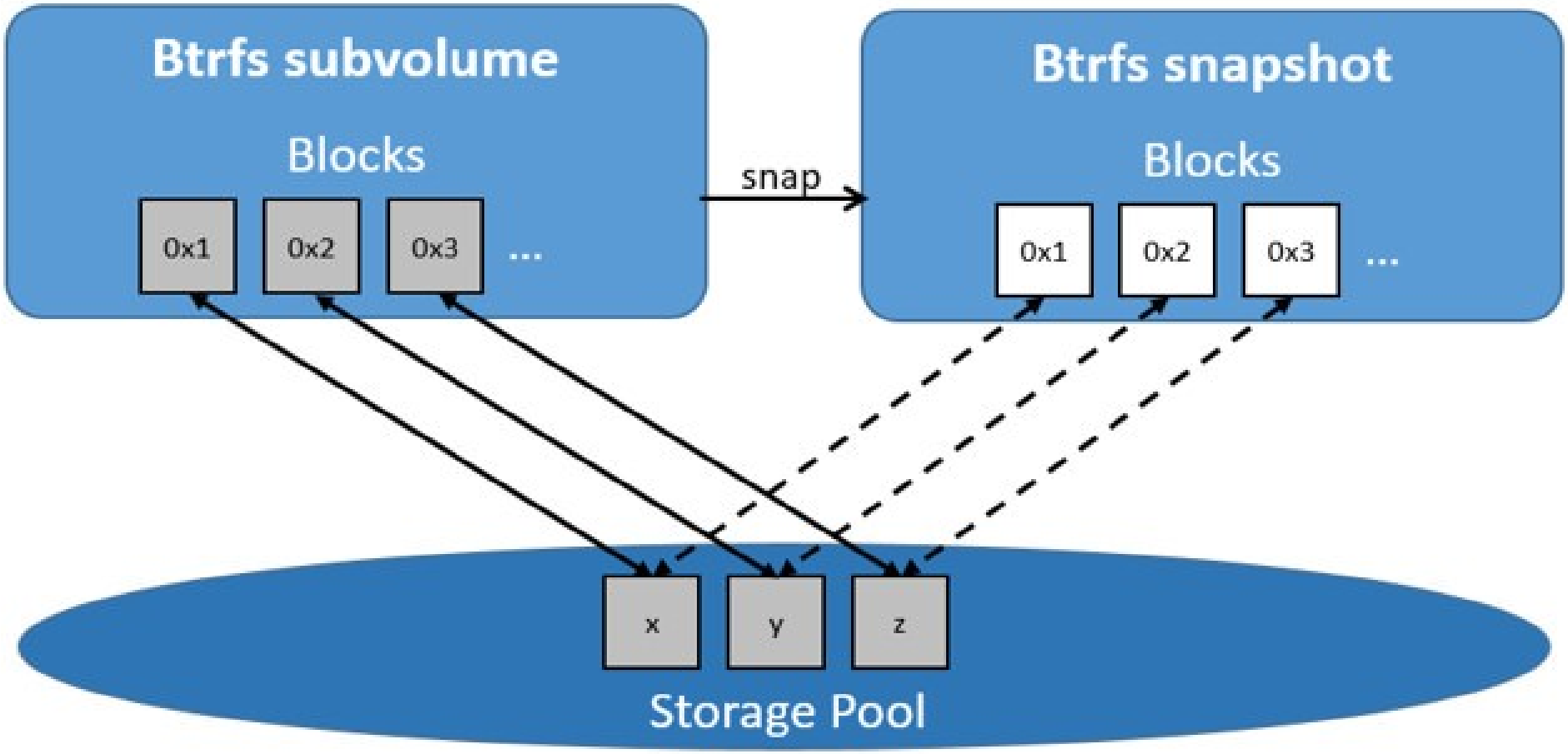
```
# systemctl enable /etc/systemd/system/hello.service
# systemctl start hello.service
```

→ Logging

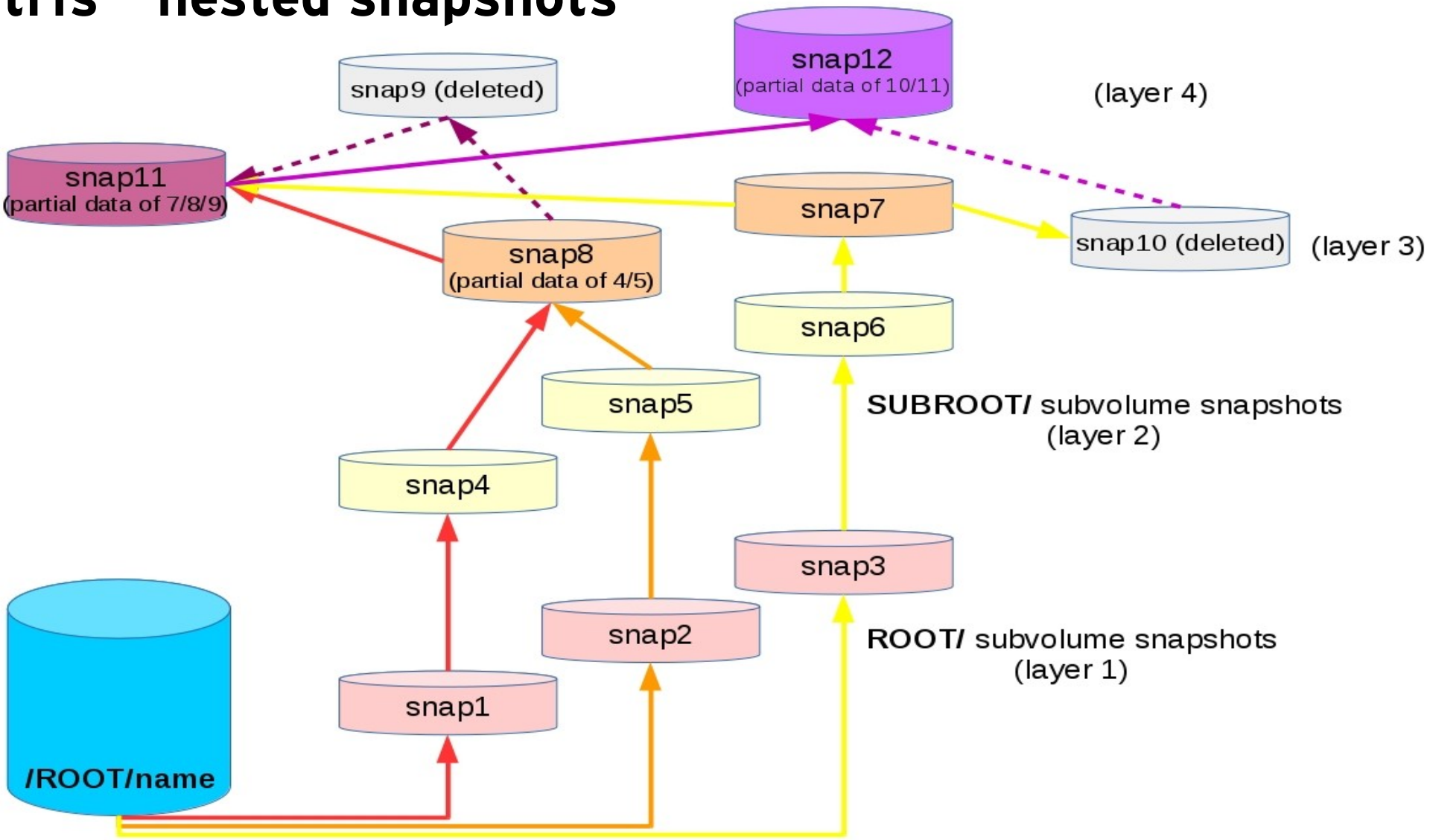
```
# journalctl -f -u hello.service
-- Logs begin at Fri 2016-06-14 00:05:55 UTC. --
Jun 14 17:46:26 localhost docker[23470]: Hello World
Jun 14 17:46:27 localhost docker[23470]: Hello World
Jun 14 17:46:28 localhost docker[23470]: Hello World
```

Filesysteme

btrfs butterfs betterfs btreefs :)



btrfs - nested snapshots



Wir suchen neue Kollegen für:

Helpdesk, Administration, Consulting!

Wir bieten:
Spannende Projekte, Kundenlob, eigenständige Arbeit,
keine Überstunden, Teamarbeit

...und natürlich: Linux, Linux, Linux...

http://www.heinlein-support.de/jobs

Stephan Seitz <s.seitz@heinlein-support.de>

heinlein

Heinlein Support hilft bei allen Fragen rund um Linux-Server

HEINLEIN AKADEMIE

Von Profis für Profis: Wir vermitteln die oberen 10% Wissen: geballtes Wissen und umfangreiche Praxiserfahrung.

HEINLEIN HOSTING

Individuelles Business-Hosting mit perfekter Maintenance durch unsere Profis. Sicherheit und Verfügbarkeit stehen an erster Stelle.

HEINLEIN CONSULTING

Das Backup für Ihre Linux-Administration: LPIC-2-Profis lösen im CompetenceCall Notfälle, auch in SLAs mit 24/7-Verfügbarkeit.

HEINLEIN ELEMENTS

Hard- und Software-Appliances und speziell für den Serverbetrieb konzipierte Software rund ums Thema eMail.