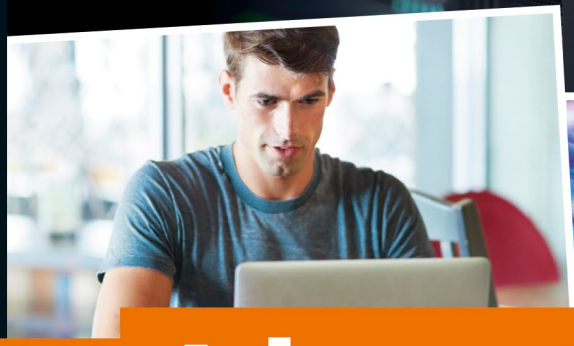
Dirk Deimeke
Stefan Kania
Daniel van Soest
Peer Heinlein
Axel Miesen

CentOS
Debian GNU/Linux
openSUSE Leap
Ubuntu Server LTS



```
$IPT -A int_to_dmz -m st
$IPT -A int_to_dmz -j ex
### Verkehr aus der DMZ
$IPT -A dmz_to_ext -m st
$IPT -A dmz_to_ext -m st
-s $MAIL -j ACCEPT
o_ext -m st
ACCEPT
```

Linux-Server

Das umfassende Handbuch

- Linux-Server einrichten und administrieren
- Backup, Sicherheit, Netzwerke und modernes DevOps
- Container mit Docker, Automatisierung mit Ansible, Datei-Server und Userverwaltung mit LDAP und Samba

7., aktualisierte Auflage

Mit allen Konfigurationsdateien zum Download

Rheinwerk Computing

Kapitel 4
Dateisysteme

In diesem Kapitel lernen Sie die Arbeitsweise der wichtigsten Linux-Dateisysteme sowie den Umgang mit den dazugehörigen Tools kennen.

Was genau macht eigentlich ein Dateisystem, und warum gibt es so viele? Als Linux-Admin haben Sie die Wahl zwischen den Dateisystemen Ext2, Ext3 und Ext4, ReiserFS, XFS, JFS und BtrFS – und das sind nur die gängigsten, universell einsetzbaren Dateisysteme. Dieses Kapitel informiert Sie im Grundlagenteil über die Struktur der Dateisysteme und über die Unterschiede zwischen ihnen. Der Praxisteil erläutert den Umgang mit den dazugehörigen Tools in der täglichen Administrationsarbeit.

4.1 Dateisysteme: von Bäumen, Journalen und einer Kuh

Dateisysteme speichern Daten. Aber sie speichern nicht nur Daten, sondern auch Daten über Daten. Dazu gehören die Größe, der Besitzer oder Informationen darüber, welche Benutzer Zugriffsrechte auf die Daten besitzen. Diese »Daten über Daten« heißen *Metadaten*.

Das Dateisystem speichert die Daten und Metadaten auch nicht nur, es *verwaltet* sie. Essenzielle Verwaltungsaufgaben, die jedes Dateisystem beherrschen muss, sind das

- Finden,
- Einfügen und
- Löschen.

Komplexere Verwaltungsvorgänge wie das *Ändern* setzen sich aus Abfolgen von *Einfügen* und *Löschen* zusammen. Wie schnell, sicher und effizient ein Dateisystem ist, hängt maßgeblich davon ab, wie gut diese grundlegenden Funktionen implementiert sind.

Frühere, heute kaum noch gebräuchliche Dateisysteme verwalteten die Informationen darüber, wo eine bestimmte Datei auf dem Datenträger zu finden war, in einer einfachen Liste. Die Elemente der Liste enthielten Zeiger auf den gesuchten Datenblock. Diese Liste (*File Allocation Table*, FAT) zu verwalten war zwar einfach, aber nicht effizient. Um ein bestimmtes Element zu finden, musste im Extremfall die ganze Liste linear durchsucht werden. Bei einer relativ geringen Anzahl von Dateien sind die Nachteile zu verschmerzen, aber der rasante Anstieg der Festplattenkapazitäten machte bessere Ordnungssysteme erforderlich. Auf sehr

kleinen und entfernbaren Datenträgern wie CDs und einigen USB-Sticks haben die Dateisysteme des alten Typs bis heute überlebt, da ihre geringe Effizienz hier kaum ins Gewicht fällt.

4.1.1 Bäume

Die Organisationsstruktur, mit der moderne Dateisysteme die gespeicherten Daten wiederfinden, ähnelt einem Baum (siehe Abbildung 4.1). Der Baum hat eine Wurzel, Gabelungen und Blätter. Die Gabelungen werden *Knoten* genannt.

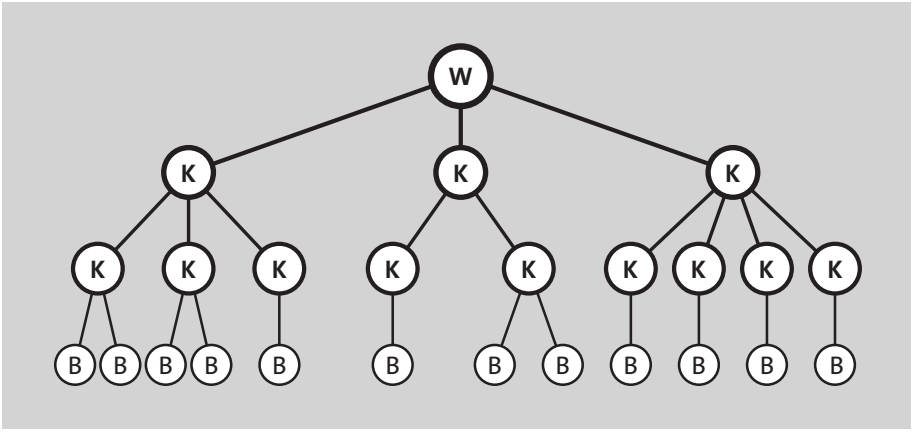


Abbildung 4.1 Schematische Darstellung einer Baumstruktur mit (W)urzel, (K)noten und (B)lättern

Von der Wurzel aus kann man über verhältnismäßig wenige Knoten jedes Blatt erreichen. Die maximale Anzahl der Schritte vom ersten Knoten (die Wurzel wird nicht mitgezählt) bis zu einem Blatt ist die *Höhe* des Baums. Ein Baum mit drei Knotenebenen hat also die Höhe 4 – drei Knotenebenen plus die Blattebene.

Baumarten und Suchstrategien

Um von der Wurzel aus möglichst schnell an jeden beliebigen Punkt des Baums zu gelangen, muss der Baum nicht hoch, sondern möglichst breit sein.[1] Daraus folgt automatisch, dass der Baum möglichst »in der Balance« sein muss, das heißt: Es soll keinen Zweig mit sieben Knoten geben, während ein anderer Zweig nur zwei Knoten hat. Außerdem soll die Höhe des Baums sich in allen Zweigen widerspiegeln. Abbildung 4.2 zeigt als Negativbeispiel einen unbalancierten Baum.

1 In einem Dateisystem kann ein Knoten durchaus 200 Unterknoten haben.

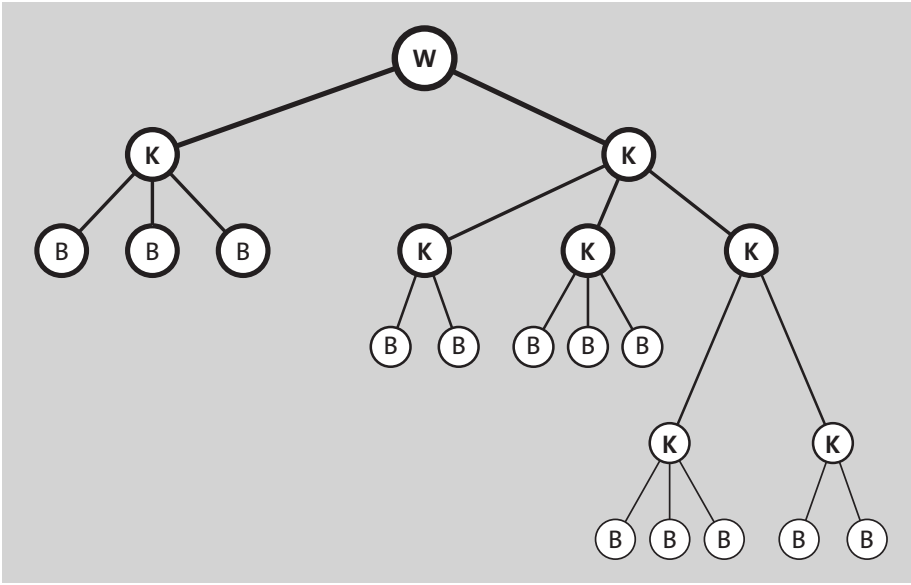


Abbildung 4.2 Eine unbalancierte Baumstruktur

Wurzel, Knoten und Blätter stehen hier natürlich nur metaphorisch für Datenstrukturen. Sie enthalten Schlüssel, anhand derer man sie innerhalb des Baums identifizieren kann. Die Datenstrukturen können neben den Schlüsseln Daten, Metadaten oder beides enthalten. Durch diese Varianz entstehen mehrere Baumarten:

- **B-Bäume**
  B-Bäume sind per Definition balanciert: Alle Blätter liegen auf der gleichen Ebene. Ist die Balance durch Dateioperationen wie Löschen oder Einfügen gefährdet, wird sie durch Neuorganisation von Blättern wiederhergestellt. Dieses Verfahren heißt *Rotation*. Ein weiteres Merkmal von B-Bäumen ist, dass Daten sowohl in den Knoten als auch in den Blättern gespeichert werden. Ext3 und BtrFS benutzen B-Bäume; Abbildung 4.1 ist eine schematische Darstellung eines B-Baums.
- **B+-Bäume**
  In einem B+-Baum werden Daten ausschließlich in den Blättern gespeichert, nicht in den Knoten. Davon profitieren insbesondere Löschoperationen, da Rotationen über Ebenengrenzen hinweg nicht auftreten können. Dateisysteme, die B+-Bäume benutzen, sind beispielsweise ReiserFS 3.x und XFS.
- **H-Bäume**
  Bei einem H-Baum steht die maximale Anzahl der Knoten und Blätter mit dem Anlegen des Dateisystems fest. Das bedeutet einen Verzicht auf Flexibilität, eliminiert aber auch die Notwendigkeit aufwendiger Reorganisationen. Abbildung 4.3 zeigt den schematischen Aufbau eines H-Baums.

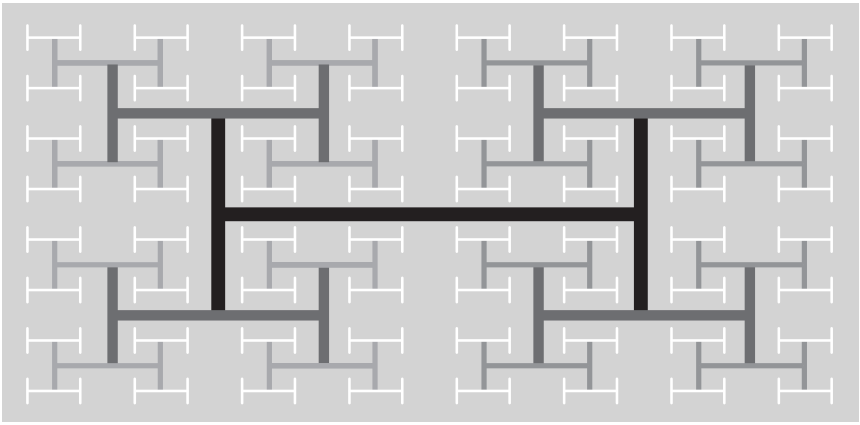


Abbildung 4.3 Schematische Darstellung eines H-Baums

4.1.2 Journale

Wenn Sie eine Datei anlegen, löschen, verschieben oder ändern, sind stets mehrere Datenbereiche auf einer Festplatte involviert: Schlüssel, Metadaten und die Daten selbst, die sich bei großen Dateien auch über mehrere, nicht zusammenhängende Blöcke erstrecken können.

Das Dateisystem hat also ein Problem. Nachdem es die erste Schreiboperation ausgeführt hat, ist der Datenbestand nicht mehr konsistent. Er ist erst dann wieder konsistent, wenn die letzte Schreiboperation erfolgreich abgeschlossen wurde. Fällt innerhalb dieser Zeitspanne der Strom aus, muss das Dateisystem beim Neustart durch eine aufwendige Prüfung und Reparatur wieder einen konsistenten Zustand herstellen. Dieser Vorgang ist auf großen Dateisystemen sehr zeitintensiv. Abhilfe schafft ein *Journaling File System*. Bevor das System die Schreibzugriffe ausführt, schreibt es in das Journal, was es zu tun gedenkt. In bestimmten Zeitabständen wird das Journal abgeschlossen.

Das bedeutet, die »To-do-Liste« im Journal wird mit den tatsächlich durchgeführten Aktionen abgeglichen und aktualisiert. Kommt es zu einem Ausfall, kann zwar immer noch Datenverlust auftreten – nämlich bei den Daten, die zwischen zwei Journal-Abschlusszyklen verändert wurden –, aber Inkonsistenzen werden vermieden, und das System kann ohne langwierige Prüfungen wieder starten.

4.1.3 Und die Kühe? COW-fähige Dateisysteme

Beim Speichern von Daten wird der ursprüngliche Plattenbereich mit den neuen Daten überschrieben. Reicht der Platz nicht mehr aus, werden die zusätzlichen Daten in einen freien Speicherbereich geschrieben. Das dazu notwendige Neupositionieren des Schreib-/Lesekopfs der Platte kostet wertvolle Zeit. Besser ist es, die Daten gleich zusam-

menhängend in einen freien Speicherbereich zu schreiben. So gewinnt man nicht nur Zeit, sondern erhält gratis noch eine Versionsverwaltung, denn die alten Daten sind ja noch auf der Platte. Diese Technik nennt sich »COW« und hat nichts mit wiederkäuenden Paarhufern zu tun, sondern steht für *Copy on Write*. COW-fähige Dateisysteme sind beispielsweise ZFS und BtrFS.

4.2 Praxis

Nach der Lektüre des Grundlagenteils haben Sie sicher schon ein Gespür dafür bekommen, wie die interne Organisationsstruktur eines Dateisystems sein Verhalten in der Praxis beeinflusst. Im Praxisteil lernen Sie den Umgang mit den Verwaltungswerkzeugen der Dateisysteme Ext2/3, ReiserFS und XFS.

4.2.1 Ext2/3-FS aufgebohrt: mke2fs, tune2fs, dumpe2fs, e2label

Die Dateisysteme der Ext-Familie werden von den meisten Linux-Distributionen als Standarddateisystem eingesetzt, weil sie als besonders robust gelten. Bereits beim Anlegen des Ext-Dateisystems können Sie es für seine zukünftigen Aufgaben optimieren.

Ein Ext-Dateisystem anlegen

Wenn Sie ein universell einsetzbares Filesystem benötigen, das gleichermaßen gut mit großen wie kleinen Dateien zurechtkommen soll, legen Sie es einfach mit dem folgenden Kommando an:

```
mke2fs /dev/<Gerätename>
```

Listing 4.1 Ext-Dateisystem ohne weitere Optionen anlegen

`mke2fs` wird abhängig von der Größe der Partition sinnvolle Default-Werte für die Anzahl der Inodes und für die Blockgröße wählen. Es geht dabei davon aus, dass das Dateisystem eine Mischung aus großen, mittleren und kleinen Dateien enthalten wird. Falls Sie allerdings bereits wissen, dass Ihr neues Dateisystem künftig mehrheitlich sehr kleine (Mail-Server) oder sehr große (Videoschnitt, ISO-Images) Dateien aufnehmen können soll, können Sie mit dem Parameter `-T` auf das Layout des Dateisystems Einfluss nehmen (siehe Tabelle 4.1).

Wert	Blockgröße	Inodes pro x Blöcke	Nutzung
small	1 KByte	x = 4	für kleine Partitionen (Default bis 512 MB)
news	4 KByte	x = 1	für viele kleine Dateien

Tabelle 4.1 Optimierung des Dateisystems auf bestimmte Aufgaben

Wert	Blockgröße	Inodes pro x Blöcke	Nutzung
largefile	4 KByte	x = 256	für große Dateien, 1 MB pro Inode
largefile4	4 KByte	x = 1.024	für große Dateien, 4 MB pro Inode

Tabelle 4.1 Optimierung des Dateisystems auf bestimmte Aufgaben (Forts.)

Die Routineüberprüfung

Nach einer bestimmten Anzahl von Mountvorgängen oder nach Ablauf einer definierten Zeitspanne – je nachdem, was zuerst eintritt – werden Ext-Dateisysteme einer Routineüberprüfung unterzogen. Auf Systemen, die oft neu gestartet werden, können Sie diese Werte mit dem Kommando `tune2fs` erhöhen. So erfolgt die Prüfung nach 50 Mounts oder 60 Tagen:

```
tune2fs -c 50 -i 60
```

Listing 4.2 Prüfintervall heraufsetzen

Auf- und Abwärtskompatibilität

Das Dateisystem Ext3 (*Third Extended Filesystem*) erweitert seinen Vorgänger Ext2 um die Journaling-Fähigkeit, die Inkonsistenzen zu vermeiden hilft. Beide haben eine hohe Stabilität und Robustheit, die über die Jahre durch schrittweise Verbesserungen und den massenhaften Einsatz auf unzähligen Linux-Systemen erreicht wurde. Beide Systeme sind miteinander kompatibel. Ein Ext3-Dateisystem kann als Ext2 gemountet werden und arbeitet mit Ausnahme des abgeschalteten Journals so, wie Sie es von einem Ext2-System gewohnt sind. Umgekehrt können Sie ein Ext2-Dateisystem jederzeit mit einem Ritterschlag zum Ext3-System aufwerten, so wie in Listing 4.3 dargestellt:

```
tune2fs -j /dev/sda3
```

Listing 4.3 Ext2-Filesystem in Ext3 konvertieren

Prinzipiell funktioniert das sogar, während das Dateisystem gemountet ist – das Journal würde in diesem Fall in einer sichtbaren Datei mit dem Namen *.journal* angelegt. Aber sicherer ist es, die Konvertierung am ausgehängten Dateisystem vorzunehmen.

Journaling

Ext3 kennt zwei Journaling-Methoden, um die Konsistenz der Metadaten zu schützen, und eine dritte, die auch die eigentlichen Nutzdaten mit einbezieht:

- **data=ordered**
  Dies ist die Standardeinstellung, ein Kompromiss zwischen Sicherheit und Geschwindigkeit. Erst nachdem die Nutzdaten einer Schreiboperation auf die Platte geschrieben wurden, werden die Journaleinträge aktualisiert.

- **data=writeback**
  Das Dateisystem wartet nicht mit dem Aktualisieren des Journals, bis die Nutzdaten tatsächlich geschrieben wurden. Diese Methode bringt einen Geschwindigkeitsvorteil, kann aber zu Datenverlust führen, wenn zum Zeitpunkt eines Ausfalls das Journal bereits aktualisiert war, aber die Daten noch nicht vollständig geschrieben wurden.
- **data=journal**
  Im Gegensatz zum *Metadata Journaling* der Methoden *ordered* und *writeback* etabliert diese Methode das *Full Journaling*. Daten und Metadaten werden zunächst in das Journal geschrieben. Erst nachdem die Daten von dort aus an ihre endgültige Position im Dateisystem kopiert wurden, wird das Journal aktualisiert. Weil auf diese Weise alle Schreibvorgänge doppelt ausgeführt werden, ist das *Full Journaling* bei Schreiboperationen recht langsam, vermeidet aber wirkungsvoll Dateninkonsistenzen.

Sie können insbesondere die Schreibperformance verbessern, indem Sie das Journal auf einen separaten Datenträger schreiben lassen, etwa einen kleinen RAID-Verbund. Dazu gehen Sie in zwei Schritten vor: Zuerst bereiten Sie das Block-Device vor, auf dem das Journal liegen soll, und im zweiten Schritt formatieren Sie die eigentliche Datenplatte und übergeben als Parameter das Journalgerät. Wenn Ihre Datenplatte */dev/sda1* und Ihre Journalplatte */dev/sdb1* heißt, lauten die beiden Kommandos wie folgt:

```
# Journaldatenträger vorbereiten:
mke2fs -O journal_dev /dev/sdb1

# Datenplatte formatieren, auf Journal-Device verweisen:
mke2fs /dev/sda1 -J device=/dev/sdb1
```

Listing 4.4 Dateisystem mit separatem Journal-Device

Zusätzlich können Sie mit dem Parameter `-L <Bezeichnung>` jedem Dateisystem einen Bezeichner (*Label*) zuweisen. Das ist auch nachträglich mit dem Befehl `e2label <Gerät> <Bezeichner>` möglich. Das *Label* darf nicht mehr als 16 Zeichen lang sein.

dumpe2fs: Informationen über das Filesystem

Mit `dumpe2fs /dev/<Gerätename>` lassen Sie sich Informationen über das Ext-Dateisystem ausgeben. Die Ausgabe ist recht lang, da auch alle Informationen über Superblocks und Blockgruppen ausgegeben werden. In den meisten Fällen reicht es aus, wenn Sie sich die verkürzte Ausgabe mit dem Kommando `dumpe2fs -h /dev/<Gerätename>` ausgeben lassen.

Hier sehen Sie unter anderem den aktuellen Status des Dateisystems und Informationen über den zyklischen *File System Check*:

```
[…]
Filesystem state:         clean
```

```
[…]
Filesystem created:       Wed Nov 11 10:25:43 2009
Last mount time:          Sun Sep 26 12:51:25 2010
Last write time:          Tue May 25 14:24:51 2010
Mount count:              14
Maximum mount count:      31
Last checked:             Tue May 25 14:24:51 2010
Check interval:           15552000 (6 months)
Next check after:         Sun Nov 21 13:24:51 2010
Lifetime writes:          3086 MB
[…]
```

Listing 4.5 Ausgabe von »dumpe2fs« (Auszug)

Sollten auf Ihrem Dateisystem einige Blöcke als *bad* (schlecht, beschädigt) markiert sein, können Sie sich mit dem Befehl `dumpe2fs -b` ausgeben lassen, welche Blöcke betroffen sind.

Dateisystemprüfung mit »e2fsck«

Wenn Sie den Verdacht haben, dass Ihr Dateisystem über noch nicht lokalisierte *bad blocks* stolpert, rufen Sie einmal `e2fsck -c /dev/<Gerätename>` auf. Das Dateisystem darf dabei nicht gemountet sein. `e2fsck` durchsucht das Dateisystem mithilfe des externen Programms `badblocks` nach Schadstellen. Die IDs der kaputten Blöcke werden einem speziellen Inode hinzugefügt und vom Dateisystem künftig gemieden.

Fehler, die nicht auf schlechten Blöcken beruhen, können Sie ebenfalls bei ausgehängtem Dateisystem mit dem Befehl `e2fsck -p /dev/<Gerätename>` orten. Das Programm wird versuchen, alle gefundenen Fehler ohne weitere Interaktion Ihrerseits zu beheben. Ist das nicht möglich, wird eine Beschreibung des Fehlers ausgegeben, und die Verarbeitung stoppt.

4.2.2 ReiserFS und seine Tools

ReiserFS war 2001 das erste Journaling-Dateisystem, das in den offiziellen Linux-Kernel einzog, damals allerdings noch in experimentellem Zustand. »Produktionsreif« wurde ReiserFS erst später. ReiserFS kann kleine Dateien etwas effizienter speichern als andere Dateisysteme und bietet dadurch auf einem ansonsten gleichen *Volume* etwa 5 % mehr Speicherkapazität. Die höhere Packungsdichte bedingt aber eine etwas geringere Schreibgeschwindigkeit und lässt sich deshalb abschalten, indem in der */etc/fstab* der Parameter `notail` gesetzt wird.

ReiserFS skaliert auf Systemen mit Mehrkernprozessoren nicht optimal, weil nicht alle Teile des Codes auf mehreren Kernen parallel ausgeführt werden können. Entwicklungen, die dieses Problem beheben, sind zwar in Arbeit, aber noch nicht in den offiziellen Linux-Kernel eingeflossen.

Ein ReiserFS-Dateisystem anlegen

Der folgende Befehl formatiert ein Block-Device mit dem Reiser-Dateisystem:

```
mkfs.reiserfs /dev/<Gerätename>
```

Listing 4.6 Ein ReiserFS-Dateisystem ohne weitere Optionen anlegen

Sie können den Befehl um `-l <Bezeichner>` ergänzen, um dem Dateisystem ein *Label* anzuheften. Wie bei den Ext-Dateisystemen darf das Label nicht mehr als 16 Zeichen umfassen. Auch ReiserFS erlaubt es, das Journal auf einen separaten Datenträger zu legen. Wenn Ihre Daten auf dem Gerät */dev/sda1* und das Journal auf */dev/sdb1* liegen sollen, lautet das Kommando:

```
mkfs.reiserfs /dev/sda1 -j /dev/sdb1
```

Listing 4.7 ReiserFS-Dateisystem mit ausgelagertem Journal anlegen

Sie können das Journal auch nachträglich auf einen anderen Datenträger auslagern. Dazu benötigen Sie den Befehl `reiserfstune`:

```
reiserfstune /dev/sda1 --journal-new-device /dev/sdb1
```

Listing 4.8 Journal nachträglich auf ein anderes Device legen

4.2.3 XFS

XFS wurde in den 90er-Jahren von *Silicon Graphics* entwickelt und steht seit 2001 für Linux zur Verfügung. Schreiboperationen auf das Journal laufen unabhängig von den eigentlichen Datentransaktionen ab, die XFS möglichst lange im RAM puffert (*Delayed Allocation)*. Auf diese Weise wird der Durchsatz erhöht und Fragmentierung beim Schreiben vermieden. Dadurch wird zwar auch die Gefahr eines Datenverlusts bei einem Totalausfall des Systems erhöht, die Konsistenz des Dateisystems ist jedoch sichergestellt.

Ein XFS-Dateisystem anlegen

Wie Ext und ReiserFS kann auch XFS sein Journal auf das gleiche Blockgerät schreiben wie die Nutzdaten oder ein »externes« Journal führen. So legen Sie das Dateisystem an:

```
# XFS mit internem Journal (Journal und Daten auf gleicher Partition)
mkfs.xfs /dev/sda1

# XFS mit externem Journal (Journal auf separater Partition)
mkfs.xfs -l logdev=/dev/sdb1 /dev/sda1
```

Listing 4.9 XFS-Filesystem mit internem und externem Journal

Das Dateisystem reorganisieren

Sollte das Dateisystem einmal merklich langsamer werden, können Sie es mit dem Befehl `xfs_fsr` reorganisieren. Geben Sie den Befehl einfach als `root` auf einer Kommandozeile ein – er funktioniert auch online, also während das Dateisystem gemountet ist. `xfs_fsr` ermittelt, welche Dateien besonders stark fragmentiert sind, und beginnt dort mit der Neuorganisation. Per Default bricht das Programm nach zwei Stunden automatisch ab, kann aber manuell neu gestartet werden und fährt an der Stelle fort, an der es aufgehört hat. Sie können auch mit dem Parameter `-t <Laufzeit_in_Sekunden>` eine andere Arbeitsdauer festlegen. Es empfiehlt sich, die Neuorganisation *cron*-gesteuert zu einer Zeit vornehmen zu lassen, in der das Dateisystem wenig belastet ist.

4.2.4 Das Dateisystem vergrößern oder verkleinern

Sie können das Dateisystem bei Bedarf vergrößern oder – im Fall von Ext2/3 und ReiserFS – auch verkleinern. Ein XFS-Dateisystem lässt sich hingegen nicht verkleinern. Beim Vergrößern eines Dateisystems muss die Partition natürlich eine gewisse Reserve aufweisen, die das Dateisystem nutzen kann. Liegt das Dateisystem auf einer logischen Partition, erweitern Sie diese zunächst mit dem `lvextend`-Kommando (siehe Abschnitt 3.2, »Rein logisch: Logical Volume Manager ›LVM‹«).

Eine Änderung der Größe ist immer ein recht tiefer Eingriff ins Dateisystem. Obwohl die Tools einen hohen Reifegrad haben, sollten Sie über ein aktuelles Backup verfügen. Vorher noch einen *File System Check* laufen zu lassen, ist ebenfalls eine gute Idee.

Die Größe eines Ext-Dateisystems ändern

Ext-Dateisysteme können Sie online und offline, also im eingehängten oder im nicht eingehängten Zustand, vergrößern, aber nur offline verkleinern. Wenn Sie Ihr Dateisystem auf die maximale Größe der (logischen) Partition ausdehnen möchten, geben Sie als `root` diesen Befehl ein:

```
resize2fs -p /dev/<Gerätename>
```

Listing 4.10 Ext-Dateisystem vergrößern

Der Parameter `-p` erzeugt einen Fortschrittsbalken, während die Vergrößerung läuft. Eine Verkleinerung nehmen Sie vor, indem Sie die Zielgröße des Dateisystems angeben. Denken Sie daran, dass Sie Ext-Dateisysteme nur im ausgehängten Zustand verkleinern können. Falls Ihr System diese Partition zum Starten benötigt, verwenden Sie eine Live-CD wie *Knoppix*.

Wenn Ihr Dateisystem 80 GB groß ist und auf 60 GB schrumpfen soll, lautet das Kommando:

```
resize2fs -p /dev/<Gerätename> 60G
```

Listing 4.11 Ext-Dateisystem verkleinern

Die Größe eines Reiser-Dateisystems ändern

Ein Reiser-Dateisystem lässt sich, sofern das Dateisystem nicht gemountet ist, vergrößern und verkleinern. Wie bei den Ext-Dateisystemen können Sie auch eine bestimmte Zielgröße angeben, auf die das Dateisystem vergrößert oder verkleinert werden soll:

```
resize_reiserfs /dev/<Gerätename> 60G
```

Listing 4.12 ReiserFS auf eine bestimmte Zielgröße bringen

Ebenso können Sie das vorhandene Dateisystem auf die maximale Größe aufblähen, die die (logische) Partition hergibt. Dazu lassen Sie einfach die Angabe der Zielgröße weg:

```
resize_reiserfs /dev/<Gerätename>
```

Listing 4.13 ReiserFS maximal vergrößern

Alternativ haben Sie die Möglichkeit, die Differenz zur aktuellen Größe des Filesystems anzugeben:

```
# Dateisystem um 20 GB verkleinern:
resize_reiserfs /dev/<Gerätename> -20G

# Dateisystem um 20 GB vergrößern:
resize_reiserfs /dev/<Gerätename> +20G
```

Listing 4.14 ReiserFS relativ verändern

Die Größe eines XFS-Systems ändern

Ein XFS-System können Sie lediglich vergrößern, nicht verkleinern. Die Änderung der Größe funktioniert nur, während das Dateisystem gemountet ist.

Eine Vergrößerung auf eine von Ihnen definierte Größe ist nicht möglich, XFS vergrößert sich immer auf die gesamte zur Verfügung stehende Partitionsgröße.

Im Unterschied zu den anderen Dateisystemen geben Sie bei XFS nicht das (logische) Gerät, sondern den Mountpunkt an:

```
xfs_growfs /<Mountpunkt> -o remount,resize
```

Listing 4.15 XFS vergrößern

4.2.5 BtrFS

Der Name *BtrFS* deutet auf die verwendeten B-Bäume hin (*B-Tree-FS*), wird aber meist wie »Butter-FS« oder »Better-FS« ausgesprochen. Obwohl die Entwicklung noch nicht vollständig abgeschlossen ist, insbesondere was die Filesystem-Tools angeht, erhält BtrFS bereits viel

Lob. Die Wahrscheinlichkeit ist hoch, dass BtrFS in nicht ferner Zukunft die Nachfolge der Ext-Dateisysteme als Quasi-Standard antreten wird.

[!] Sehr lange fand sich im offiziellen Getting-Started-Guide des BtrFS-Wiki[2] der Warnhinweis, dass man den aktuellsten Kernel verwenden solle, wenn man BtrFS einsetzt. Weiterhin war dort der Hinweis zu finden, dass man ein Backup besitzen und darauf vorbereitet sein sollte, es einzusetzen. Seit 2014 gilt das Dateisystem als stabil, regelmäßige Backups sollten Sie natürlich trotzdem erstellen.

In der Tat ist die Liste der unterstützten Funktionen beeindruckend. Als COW-System unterstützt BtrFS Snapshots. Der Zustand des Dateisystems wird dabei in einem Subvolume eingefroren und kann dort anderweitig genutzt werden, etwa für ein Backup. Solid-State-Speicher werden gesondert unterstützt, außerdem ist die Unterstützung der RAID-Level 0, 1 und 10 implementiert. Die Level 5 und 6 gelten immer noch als experimentell, und das schon seit Jahren. Wie XFS kann BtrFS im gemounteten Zustand defragmentiert werden. Außerdem erkennt BtrFS, wenn es auf einer SSD eingesetzt wird, und aktiviert dafür selbstständig sinnvolle Einstellungen.

Die große Stärke von BtrFS ist, dass es komplette Devices verwalten und durch Subvolumes gekapselte Mountpunkte exportieren kann. Im kommerziellen Bereich bietet das sehr ausgereifte, ZFS ähnliche Möglichkeiten. Da die Lizenz von ZFS (Common Development and Distribution License, CDDL) nicht kompatibel mit der Lizenz des Linux-Kernels (GNU Public License, GPL) ist, wird sich ZFS nie im Linux-Kernel befinden. Aus diesem Grund versuchen namhafte Firmen die Entwicklung von BtrFS nach vorn zu treiben, um die Features eines modernen Dateisystems mit integriertem Volumemanager auch unter Linux direkt nutzen zu können.

[+] Beachten Sie bitte, dass unter Debian und Ubuntu das Paket *btrfs-progs* und unter openSUSE das Paket *btrfsprogs* installiert sein sollte, wenn Sie mit BtrFS arbeiten wollen. In CentOS Stream gibt es derzeit keinen BtrFS-Support, was umso erstaunlicher ist, als dass Fedora mit Version 33 dieses Dateisystem als Standard für Desktops einsetzt.

Ein BtrFS-Dateisystem anlegen und die Komprimierung aktivieren

Da die Entwicklung nach wie vor andauert, sollten Sie BtrFS derzeit noch mit Vorsicht genießen. Das Kommando `btrfsfsck` existiert zwar, aber dahinter verbirgt sich derzeit nur ein Integritäts-Check, echte Reparaturen am Dateisystem können Sie damit noch nicht durchführen.

Wenn Sie BtrFS einmal ausprobieren möchten, können Sie es wie jedes andere Dateisystem auf einer freien Partition anlegen, im folgenden Beispiel verwenden wir */dev/sdb1*.

2 *https://archive.kernel.org/oldwiki/btrfs.wiki.kernel.org/index.php/Getting_started.html*

Dieses Beispiel dient Demonstrationszwecken. Auch wenn es möglich ist, direkt auf das Filesystem innerhalb der Partition zuzugreifen, geht die Konzeption von BtrFS in die Richtung, dass Sie das nicht müssen, sondern stattdessen mit Subvolumes arbeiten, die später in diesem Abschnitt beschrieben werden.


```
mkfs.btrfs /dev/sdb1
```

Listing 4.16 Ein BtrFS-Dateisystem anlegen bzw. eine Partition BtrFS zur Verfügung stellen

Auch das Einhängen des neuen Dateisystems in den Verzeichnisbaum funktioniert wie bei anderen Dateisystemen. Wenn Sie möchten, können Sie BtrFS beim Mountbefehl zusätzlich anweisen, die Datenkomprimierung einzuschalten:

```
mount -o compress /dev/sdb1 /mnt/meinbtrfs
```

Listing 4.17 Das neue Dateisystem mounten und dabei die optionale Komprimierung einschalten

Die Komprimierung verschlingt natürlich einige CPU-Zyklen, aber in Zeiten schneller Multicore-CPUs ist es unwahrscheinlich, dass Sie diese Verzögerung wahrnehmen. Im Gegenteil: Die Komprimierung wird sich eher positiv auf die Verarbeitungsgeschwindigkeit auswirken, weil Schreibzugriffe auf die Platten wesentlich flotter ablaufen.

Das gilt natürlich nicht für Daten, die bereits komprimiert sind, wie JPEG-Bilder, die meisten Videos und Musik im MP3-Format. BtrFS erkennt solche bereits komprimierten Dateien und macht gar keinen Versuch, sie noch weiter zu verdichten. Bei gut komprimierbaren Daten wie Text oder Quellcode ist der Gewinn an Geschwindigkeit und Plattenplatz allerdings beträchtlich.

Ein Ext2/3/4-Dateisystem in BtrFS konvertieren

Sie können ein mit Ext2, Ext3 oder Ext4 formatiertes Dateisystem nachträglich in BtrFS konvertieren. Dazu müssen Sie das Dateisystem (im folgenden Beispiel befindet es sich wieder auf der Partition */dev/sdb1*) zunächst aushängen:

```
# zuerst wird das Dateisystem ausgehängt
umount /dev/sdb1

# jetzt erfolgt die Konvertierung
btrfs-convert /dev/sdb1

# danach können Sie das Dateisystem wieder mounten (mit Komprimierung)
mount -o compress /dev/sdb1 /mnt/meinbtrfs
```

Listing 4.18 Eine Ext2/3/4-Partition in BtrFS konvertieren

Sie werden feststellen, dass die Konvertierung recht flott vonstattengeht, selbst auf großen Dateisystemen. Das liegt daran, dass der Konverter die Nutzdaten überhaupt nicht touchiert,

sondern lediglich die BtrFS-Metadaten schreibt und einen Snapshot des ursprünglichen Dateisystems erstellt. Wenn Sie entscheiden, dass die Konvertierung in BtrFS keine gute Idee war, können Sie den ursprünglichen Zustand jederzeit mithilfe des Snapshots wiederherstellen (siehe Listing 4.19). Dabei verlieren Sie allerdings alle Änderungen, die Sie seit der Konvertierung vorgenommen haben.

```
# wieder wird zuerst das Dateisystem ausgehängt
umount /dev/sdb1

# die Konvertierung rückgängig machen
btrfs-convert -r /dev/sdb1

# danach können Sie das Dateisystem wieder mounten
mount /dev/sdb1 /mnt/meinext3fs

# Kontrolle
df -h
```

Listing 4.19 Die Konvertierung rückgängig machen

Wenn Sie sich entschließen, BtrFS zu behalten, und auf die Möglichkeit der Rückkonvertierung keinen Wert legen, dann können Sie den Snapshot – er heißt `ext2_saved` – löschen:

```
btrfs subvolume delete /mnt/meinbtrfs/ext2_saved
```

Listing 4.20 Den Snapshot des alten Dateisystems entfernen

Subvolumes

Subvolumes werden oft als *Namespaces* innerhalb eines BtrFS-Dateisystems beschrieben. Es ist aber einfacher, sich Subvolumes als *virtuelle Dateisysteme* vorzustellen. Bei der Erstellung eines BtrFS-Dateisystems wird bereits ein *root-Subvolume* angelegt. Subvolumes können wieder weitere Subvolumes enthalten und so ineinander verschachtelt werden.

Wir haben bereits in der Partition *sdb1* ein Dateisystem erstellt, das wir nun nach */mnt/btrfs* mounten. Dort erstellen wir ein Verzeichnis *home* und darin ein Subvolume pro User; *opt* soll ein weiteres Subvolume werden. Diese Subvolumes lassen sich selbstverständlich anzeigen, wie Sie in Listing 4.21 sehen. Leider wird das standardmäßig erstellte Subvolume für das Filesystem nicht angezeigt, aber da es immer die ID 5 hat, ist es leicht zu identifizieren.

```
# btrfs subvolume create /mnt/btrfs/home/user1
Create subvolume '/mnt/btrfs/home/user1'
# btrfs subvolume create /mnt/btrfs/home/user2
Create subvolume '/mnt/btrfs/home/user2'
# btrfs subvolume create /mnt/btrfs/opt
Create subvolume '/mnt/btrfs/opt'
```

```
# btrfs subvolume list -apt /mnt/btrfs
ID      gen     parent  top level       path
--      ---     ------  ---------       ----
257     6       5       5               home/user1
258     7       5       5               home/user2
259     8       5       5               opt
```

Listing 4.21 Erstellung von Subvolumes

Subvolumes verhalten sich innerhalb des Parent-Volumes wie Verzeichnisse, können allerdings nicht mit `rm` gelöscht werden. Dafür muss `btrfs subvolume delete` verwendet werden. Der Parameter ist der gleiche wie beim Erstellen der Subvolumes.

Eine Besonderheit von Subvolumes ist, dass sie auch eigenständig mittels `mount` eingebunden werden können. Dazu kann entweder der Name oder die ID des Subvolumes verwendet werden:

```
# mount -o subvolume=home/user1 /dev/sdb1 /tmp/1
# mount -o subvolid=257 /dev/sdb1 /tmp/1
```

Listing 4.22 Mounten von Subvolumes

Snapshots

Sie können jederzeit von beliebigen Subvolumes Snapshots erstellen. Diese Abbilder enthalten allerdings nur Daten des Subvolumes, für das sie erstellt wurden, nicht für eventuell darunter liegende Subvolumes. Anders als bei LVM-Snapshots (siehe Abschnitt 3.2.8 »Backups mit Snapshots«) müssen Sie bei BtrFS keine besonderen Vorbereitungen treffen. Es hat sich als hilfreich erwiesen, den Snapshots die Endung »-snap« zu geben, um sie leichter identifizieren zu können. Snapshots sind beschreibbar!

BtrFS-intern werden Snapshots wie Subvolumes behandelt. Der einzige Unterschied ist, dass sie zu Beginn keine Daten enthalten. Erst wenn sich das Ursprungs-Subvolume oder der Snapshot verändert, wird – gemäß dem Copy-on-Write-Prinzip – Speicherplatz verwendet. Der Parameter `-r` erzeugt beim Erstellen einen nur lesbaren Snapshot:

```
# btrfs subvolume snapshot /mnt/btrfs/opt /mnt/btrfs/opt-snap
# btrfs subvolume snapshot -r /mnt/btrfs/opt /mnt/btrfs/opt-snap-ro
# btrfs mount -o subvolume=opt-snap /dev/sdb1 /tmp/snap
# btrfs mount -o subvolume=opt-snap-ro /dev/sdb1 /tmp/snap-ro
```

Listing 4.23 Erstellung von Snapshots

Defragmentierung

Wie bei allen COW-Dateisystemen ist Fragmentierung bei BtrFS ein Problem. Geänderte Daten dürfen nicht einfach überschrieben werden, sondern wandern in freie Speicherberei-

che. Ist der Speicherbereich kleiner als die zu schreibenden Daten, werden diese zerstückelt, und es kommt zu Fragmentierung. Deshalb müssen BtrFS-Dateisysteme regelmäßig defragmentiert werden. Die Defragmentierung können Sie manuell vornehmen, indem Sie ein Verzeichnis angeben (tatsächlich ein Verzeichnis, nicht ein Subvolume), dessen Daten neu geordnet werden sollen:

```
sudo btrfs filesystem defragment /opt
```

Listing 4.24 Defragmentierung der Daten eines Verzeichnisses

Sie können während der Defragmentierung weiterarbeiten, aber das System wird sich etwas zäher anfühlen. Verzeichnisse auf SSDs müssen nicht defragmentiert werden – die Fragmentierung findet dort zwar auch statt, ist aber unproblematisch, da sie im Gegensatz zu Festplatten nicht zu einer Verringerung des Plattendurchsatzes führt: Eine SSD muss ja keine Schreib-/Leseköpfe neu positionieren. Sie können auch eine Hintergrund-Defragmentierung aktivieren. Dazu fügen Sie einfach in der */etc/fstab* die Mountoption `autodefragment` hinzu.

Mehrere Devices in BtrFS

Wie eingangs beschrieben wurde, sollte man beim Erstellen des BtrFS-Dateisystems davon sprechen, dass Devices BtrFS zur Verfügung gestellt werden. BtrFS bietet Unterstützung für verschiedene RAID-Level, die sich getrennt nach Nutzdaten und Metadaten, die zur Verwaltung von BtrFS verwendet werden, einstellen lassen. Die beiden gebräuchlichsten Fälle sind *Striping* (RAID-0) und *Mirroring* (RAID-1). Wenn man nur Devices ohne Angabe eines RAID-Levels zur Verfügung stellt, sorgt BtrFS für ein Striping der Nutzdaten und ein Mirroring der Metadaten. Um ein Filesystem mit RAID-Level zu mounten, müssen Sie nur ein Device des Verbunds angeben:

```
# mkfs.btrfs -f -d raid1 -m raid1 /dev/sdb /dev/sdc
# mount -o compress /dev/sdb /mnt/btrfs
```

Listing 4.25 Ein BtrFS mit RAID-1 anlegen

Bitte unterschätzen Sie das Volumen der Metadaten nicht! BtrFS kann einige seiner Vorteile nur durch die intensive Nutzung von Metadaten zur Verfügung stellen. Selbst wenn Ihnen das `df`-Kommando freien Speicher anzeigt, heißt das nicht, dass dieser auch wirklich verfügbar ist, da der Speicher von Nutz- und Metadaten gleichermaßen genutzt wird. Mögliche Deduplizierung und Kompression sorgen ebenfalls für ungenaue Anzeigen durch das Betriebssystem. Als Beispiel kopieren wir einmal den Inhalt des */usr*-Verzeichnisses (1,1 GB) in das gerade angelegte Filesystem:

```
# du -hs /usr
1.1G    /usr
```

```
# cp -r /usr /mnt/btrfs
# df -h /mnt/btrfs
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        2.0G  552M  1.3G  30% /mnt/btrfs
```

Listing 4.26 Speicherauslastung von BtrFS

Um Verwirrung zu vermeiden, stellt BtrFS das `filesystem`-Subkommando zur Verfügung. Mit `show` können Sie sich den Aufbau anzeigen lassen:

```
# btrfs filesystem show /mnt/btrfs
Label: none  uuid: 71c1d2cc-479a-4090-ae7f-482d9ac21c29
        Total devices 2 FS bytes used 535.89MiB
        devid    1 size 2.00GiB used 853.50MiB path /dev/sdb
        devid    2 size 2.00GiB used 833.50MiB path /dev/sdc
```

Listing 4.27 Den Aufbau des Filesystems anzeigen lassen

Das Diskfree-Kommando in Listing 4.26 zeigt Ihnen 552 MB verbrauchten Speicherplatz. Das Kommando `btrfs filesystem show` aus Listing 4.27 gibt hingegen aus, dass 853 MiB belegt sind.

Wie der Speicher verwendet wird, zeigt Ihnen `btrfs filesystem df`. Wie Sie sehen können, werden 204 MiB für Metadaten verwendet:

```
# btrfs filesystem df /mnt/btrfs | column -ts,
Data            RAID1: total=620.75MiB   used=500.80MiB
Data            single: total=8.00MiB    used=0.00B
System          RAID1: total=8.00MiB     used=16.00KiB
System          single: total=4.00MiB    used=0.00B
Metadata        RAID1: total=204.75MiB   used=35.08MiB
Metadata        single: total=8.00MiB    used=0.00B
GlobalReserve   single: total=16.00MiB   used=0.00B
```

Listing 4.28 »df«-Subkommando von »btrfs filesystem«

Eine genauere und auch längere Ausgabe bekommen Sie mit `btrfs filesystem usage`:

```
# btrfs filesystem usage /mnt/btrfs
Overall:
    Device size:                   4.00GiB
    Device allocated:              1.65GiB
    Device unallocated:            2.35GiB
    Device missing:                  0.00B
    Used:                          1.05GiB
    Free (estimated):              1.30GiB      (min: 1.30GiB)
```

```
    Data ratio:                       1.99
    Metadata ratio:                   1.96
    Global reserve:               16.00MiB      (used: 0.00B)
Data,single: Size:8.00MiB, Used:0.00B
   /dev/sdb        8.00MiB

Data,RAID1: Size:620.75MiB, Used:500.80MiB
   /dev/sdb      620.75MiB
   /dev/sdc      620.75MiB

Metadata,single: Size:8.00MiB, Used:0.00B
   /dev/sdb        8.00MiB

Metadata,RAID1: Size:204.75MiB, Used:35.08MiB
   /dev/sdb      204.75MiB
   /dev/sdc      204.75MiB

System,single: Size:4.00MiB, Used:0.00B
   /dev/sdb        4.00MiB

System,RAID1: Size:8.00MiB, Used:16.00KiB
   /dev/sdb        8.00MiB
   /dev/sdc        8.00MiB

Unallocated:
   /dev/sdb        1.17GiB
   /dev/sdc        1.19GiB
```

Listing 4.29 Das »usage«-Subkommando von »btrfs filesystem«

4.3 Fazit

Sie haben nun eine Reihe von Dateisystemen kennengelernt und erfahren, dass jedes seine eigenen Stärken und Schwächen hat. Das versetzt Sie in die Lage, für jede Aufgabe das passende Dateisystem zu wählen. Obwohl die Standarddateisysteme wie Ext4 und XFS prinzipiell für alle Einsatzzwecke geeignet sind, können Sie durch den Einsatz eines spezielleren Dateisystems nicht selten ein Plus an Geschwindigkeit oder Sicherheit herausholen.

Der zunächst etwas unsortiert anmutende Zeichenblock zeigt an, womit jeder Apache-Prozess im Moment beschäftigt ist. Tabelle 8.1 listet die Bedeutung der Zeichen auf.

Zeichen	Bedeutung
_	wartet auf Verbindung
S	Prozess startet
R	Anfrage des Clients wird empfangen.
W	Antwort wird gesendet.
K	Keep-Alive – die Verbindungen bleiben für weitere Anfragen offen.
D	Nameserver-Lookup
C	Verbindung wird geschlossen.
L	Eintrag wird ins Logfile geschrieben.
G	Ein Prozess wird kontrolliert beendet.
I	*Idle Cleanup* – ein Prozess erhielt keine Verbindungen und wird aufgeräumt.
.	Verbindungs-Slot ohne laufenden Prozess

Tabelle 8.1 Statuskürzel in der Serverstatusanzeige

8.2 nginx

nginx ist ein schlanker und schneller Webserver, der dem Apache mächtig Konkurenz macht. Insbesondere das Ausliefern von statischem Inhalt erledigt nginx (man spricht den Namen als *Engine X* aus) extrem schnell bei gleichzeitig sehr geringem RAM-Bedarf. Ebenso zeichnet nginx sich dadurch aus, dass neue Technologien sehr schnell unterstützt werden.

8.2.1 Installation

nginx gehört zum Standardumfang jeder größeren Distribution, daher kann die Installation einfach über die Paketquellen erfolgen:

- **Debian:** `apt install nginx`
- **Ubuntu:** `apt install nginx`
- **CentOS:** `dnf install nginx`
- **openSUSE Leap:** `zypper install nginx`

8.2.2 Grundlegende Konfiguration

Unter */etc/nginx* finden Sie die Konfigurationsdateien von nginx. In der Datei *nginx.conf* finden Sie grundsätzliche Einstellungen, HTTP-Optionen und Möglichkeiten, um auf die Performance Einfluss zu nehmen. Ähnlich wie beim Apache ist die eigentliche Konfiguration der (virtuellen) Hosts in separate Dateien ausgegliedert. Ihre Konfiguration wird im nächsten Abschnitt behandelt, hier schauen wir uns zunächst die Basiskonfiguration in der */etc/nginx/nginx.conf* an. Sie finden hier bereits eine funktionierende Konfiguration vor, die aber in der Regel noch Optimierungspotenzial hat. Folgende Einstellungen können Sie hinzufügen oder, wenn sie schon vorhanden sind, an Ihre Bedürfnisse anpassen:

- **worker_processes auto;**
  nginx fragt die Anzahl der (virtuellen) CPUs selbstständig ab und skaliert entsprechend. Das »harte« Einstellen der Worker-Prozesse auf einen festen Wert ist daher nicht mehr notwendig.
- **worker_rlimit_nofile 300000;**
  Beim Apache sind die systemweiten Limits für die Anzahl gleichzeitig offener Files unkritisch, weil sie pro (Sub-)Prozess gelten und Apache für jeden Request einen neuen Thread öffnet. Bei nginx ist das Limit aber unter Umständen ein Problem und sollte großzügig nach oben korrigiert werden. Dieser Wert ist standardmäßig nicht gesetzt.
- **multi_accept on**
  Da nginx schon per Default *non-blocking* arbeitet, sollte man das Abarbeiten von Requests nicht künstlich serialisieren. Dieser Wert ist standardmäßig nicht gesetzt und steht auf `off`.
- **worker_connections 2048;**
  Per Default verarbeitet nginX maximal 512 gleichzeitige Requests pro Worker-Prozess. Es spricht nichts dagegen, diesen Wert auf leistungsfähiger Hardware anzuheben.
- **sendfile on;**
  tcp_nopush on;
  Diese Einstellungen werden im `http`-Abschnitt der */etc/nginx/nginx.conf* vorgenommen. `Sendfile()` ist der *System Call*, der Dateien von der Platte in ein TCP-Paket bringt. Es arbeitet dabei deutlich effizienter als ältere Verfahren. `tcp_nopush` sorgt dafür, dass Header-Daten in einem einzigen Paket (statt mehreren) geschickt werden.

8.2.3 Virtuelle Hosts

Für die folgenden Beispiele wollen wir zunächst einen Server konfigurieren, der auf die Namen *www.example.com* und *example.com* hört. Die Daten, die er ausliefert, liegen im Verzeichnis */var/www/example.com*. Danach konfigurieren wir die Subdomain *sub.example.com*, die Daten aus */var/www/sub-example.com* ausliefert.

Verzeichnisse erstellen unter CentOS und openSUSE Leap

Die Verzeichnisse *sites-available* und *sites-enabled* sind standardmäßig unter CentOS und openSUSE Leap nicht vorhanden. Daher müssen Sie diese zunächst anlegen.

Legen Sie zunächst die Konfigurationsdatei */etc/nginx/sites-available/example.com* an. Sie enthält den sogenannten `server`-Konfigurationsblock. In diesem Konfigurationsblock definieren Sie mindestens folgende Einstellungen:

- **server_name** und **listen**
  die IP-Adresse und den Port, auf dem Ihr Server auf eingehende Verbindungen wartet
- **root**
  Diese Einstellung hat nichts mit dem Benutzer `root` zu tun, sondern entspricht dem `DocumentRoot` beim Apache. Hier wird definiert, wo die auszuliefernden Dateien liegen.
- **location**
  Hier legen Sie die Namen der Index-Dateien fest.

Eine gültige Konfigurationsdatei für eine Serverkonfiguration kann also etwa so aussehen wie in Lising 8.30 dargestellt.

```
server{
      listen       80;
      server_name  example.com www.example.com;
      root         /var/www/example.com;
      location / {
            index     index.html  index.htm;
      }
}
```

Listing 8.30 Eine einfache Server-Konfigurationsdatei

Wenn Sie nun zusätzlich die Subdomain *sub.example.com* einrichten wollen, haben Sie die Wahl: Sie können entweder eine weitere Konfigurationsdatei unter */etc/nginx/sites-available* anlegen, oder Sie schreiben einen zweiten `server`-Konfigurationsblock in die bereits bestehende Datei, die Sie gerade angelegt haben.

Die neue Konfiguration können Sie einfach unter dem bereits bestehenden Teil anhängen, wie in Listing 8.31 gezeigt:

```
server{
    listen       80;
    server_name  example.com www.example.com;
    root         /var/www/example.com;
```

```
    location / {
        index    index.html  index.htm;
    }
}

server{
    listen      80;
    server_name sub.example.com www.sub.example.com;
    root        /var/www/sub-example.com;

    location / {
        index   index.html  index.htm;
    }
}
```

Listing 8.31 Zwei Serverkonfigurationen (Domain und Subdomain) in einer Datei

Virtuellen Host aktivieren und deaktivieren

Damit die Konfiguration wirksam wird, müssen Sie im Verzeichnis */etc/nginx/sites-enabled* einen symbolischen Link auf die Konfigurationsdatei im Verzeichnis */etc/nginx/sites-available* anlegen. Das Kommando dazu sehen Sie in Listing 8.32:

```
$ ln -s /etc/nginx/sites-available/myserver /etc/nginx/sites-enabled/myserver
```

Listing 8.32 Symbolischen Link anlegen

Unter CentOS und openSUSE Leap müssen Sie zusätzlich noch das neu angelegte Verzeichnis *sites-enabled* in der Hauptkonfiguration (*nginx.conf*) hinzufügen. Ergänzen Sie dafür innerhalb des `http`-Blocks die Zeilen aus Listing 8.33:

```
http {
  […]
  include sites-enabled/*;
  […]
}
```

Listing 8.33 Inkludieren der ausgelagerten virtuellen Hostdateien

[!]

CentOS-SELinux-Anpassung!

Da wir die Standardpfade verändern, muss auf CentOS-Systemen SELinux an die neuen Gegebenheiten gewöhnt werden. Führen Sie dafür den nachstehenden Befehl aus: `chcon -Rt httpd_sys_content_t /var/www/`. Ansonsten erhalten Sie beim Aufruf stets den irreführenden Fehler *403*.

Danach muss nginx mit `systemctl restart nginx` einmal neu gestartet werden.

CentOS/openSUSE: Kein Zugriff? Firewallfreigabe einrichten!

Falls Sie trotz der obigen Konfiguration stets eine Fehlermeldung im Browser erhalten, müssen Sie unter Umständen zusätzliche Regeln in der lokalen Firewall einrichten:

```
### openSUSE Leap und CentOS:
server:~ # firewall-cmd --add-service=http
server:~ # firewall-cmd --add-service=https
server:~ # firewall-cmd --permanent --add-service=http
server:~ # firewall-cmd --permanent --add-service=https
```


8.2.4 HTTPS mit nginx

Einen nginx-Server HTTPS-fähig zu machen, erforderte in frühen Versionen noch einen erheblichen Aufwand, oft sogar das Neukompilieren des Servers. Diese Zeiten sind zum Glück vorbei. Inzwischen genügt es, einen privaten Schlüssel und ein Zertifikat zu generieren und dem Server den Pfad dorthin mitzuteilen. Am besten legen Sie unterhalb von */etc/nginx* ein Verzeichnis namens *ssl* an, in dem Sie Schlüssel und Zertifikate aufbewahren. Führen Sie die `openssl`-Kommandos aus Listing 8.34 aus, um die benötigten Dateien für unsere Beispiel-Domain *example.com* zu generieren:

```
# Verzeichnis erstellen und hineinwechseln
mkdir -p /etc/nginx/ssl && cd $_

# privaten Schlüssel und die Signaturanforderung (CSR) erzeugen:
openssl genrsa -out example.com.key 2048
openssl req -new -key example.com.key -out example.com.csr

# Letzter Schritt: signieren
openssl x509 -req -days 365 -in example.com.csr \
 -signkey example.com.key -out example.com.crt
```

Listing 8.34 Schlüssel und Zertifikat für die Domain »example.com« erzeugen

Zum Schluss teilen Sie nginx im `server`-Konfigurationsblock mit, dass er SSL benutzen soll und wo er den Schlüssel und das Zertifikat findet (siehe Listing 8.35). Nach einem Neustart ist der HTTPS-Server betriebsbereit.

```
server {
      listen              443 ssl;
      ssl_certificate     /etc/nginx/ssl/example.com.crt;
      ssl_certificate_key /etc/nginx/ssl/example.com.key;
```

- **statd**
  Das Stat-Protokoll registriert die Dateien, die vom Client auf dem Server geöffnet sind.

Bei NFSv4 werden alle Protokolle, die unter NFSv3 noch eigenständig sind, zusammengefasst. Dadurch laufen bei NFSv4 alle Daten über einen Standardport, den TCP-Port 2049. Im Gegensatz zu NFSv3 ist TCP das Standardtransportprotokoll, bei NFSv3 war das noch UDP.

Eine weitere große Änderung in der Funktion von NFSv4 ist das Verwenden eines Pseudodateisystems, in das dann die eigentlichen Freigaben eingebunden werden. Die direkte Freigabe wie bei NFSv3 ist zwar auch noch möglich, wird aber nicht mehr empfohlen.

16.3 Einrichten des NFSv4-Servers

Bevor der NFS-Server konfiguriert werden kann, müssen die entsprechenden Pakete auf dem Server installiert werden. Bei Debian und Ubuntu sind das die Pakete `nfs-kernel-server` und `nfs-common`. Unter openSUSE und CentOS installieren Sie das Paket `nfs-kernel-server`.

Unter openSUSE aktivieren Sie zusätzlich noch den Dienst `nfsserver` im *Runlevelediting*.

16.3.1 Konfiguration des Pseudodateisystems

Bei NFSv4 wird der eigentliche Zugriff auf die Daten über ein Pseudodateisystem bereitgestellt. Der Vorteil ist der, dass ein Client nicht auf den Rest des Dateisystems zugreifen kann. Erzeugen Sie ein Verzeichnis, in das später alle Freigaben eingebunden werden. In Listing 16.1 sehen Sie, wie das Verzeichnis gleich mit den richtigen Rechten angelegt wird.

```
root@adminbuch:~# mkdir -m 1777 /nfs4root
```

Listing 16.1 Erzeugung des Mountpoints für das Pseudodateisystem

In unserem Beispiel sollen das Verzeichnis */daten* und das Verzeichnis */abteilungen* freigegeben werden. Für diese beiden Freigaben erzeugen Sie als Erstes jeweils einen `bind`-Mountpoint im Pseudodateisystem und setzen die benötigten Rechte. In Listing 16.2 sehen Sie das Erstellen der Mountpoints:

```
root@adminbuch:~# cd /nfs4root/
root@adminbuch:/nfs4root# mkdir -m 1777 daten
root@adminbuch:/nfs4root# mkdir -m 1777 abteilungen
```

Listing 16.2 Erzeugung der bind-Mountpoints

Jetzt können Sie die eigentlichen Verzeichnisse, in denen sich die Daten befinden, in die gerade erzeugten *bind*-Mountpoints einbinden.

[+] Die Berechtigungen am bind-Mountpoint haben keine Auswirkungen auf die spätere Freigabe, sie dienen lediglich dazu, das eigentliche Verzeichnis an das Pseudodateisystem zu binden. Für die Zugriffe auf die Dateien gelten weiterhin die Berechtigungen in den Verzeichnissen der Freigabe (siehe Listing 16.3):

```
root@adminbuch:/nfs4root# mount --bind /daten daten/
root@adminbuch:/nfs4root# mount --bind /abteilungen abteilungen/
```

Listing 16.3 Einbinden der Datenverzeichnisse in die bind-Mountpoints

Wenn Sie das Einbinden ohne Fehler durchführen konnten, können Sie die bind-Mounts jetzt permanent in die Datei */etc/fstab* eintragen:

```
/daten    /nfs4root/daten    none    rw,bind    0  0
/abteilungen    /nfs4root/abteilungen    none    rw,bind    0  0
```

Listing 16.4 Einträge in der Datei »/etc/fstab«

16.3.2 Anpassen der Datei »/etc/exports«

In der Datei */etc/exports* werden die Freigaben für den NFS-Server eingetragen. Tragen Sie als Erstes das Pseudodateisystem in die Datei ein und anschließend die eigentlichen Freigaben:

```
# Eintrag für das Pseudodateisystem
/nfs4root  192.168.56.0/24(ro,sync,insecure,root_squash,no_subtree_check,fsid=0)

# Einträge für die Freigaben
/nfs4root/daten  192.168.56.0/24(rw,sync,insecure,root_squash,no_subtree_check)
/nfs4root/abteilungen   192.168.56.0/24(rw,sync,insecure,no_root_squash,\
                        no_subtree_check)
```

Listing 16.5 Einträge in der Datei »/etc/exports«

Die Parameter der Einträge haben die folgenden Bedeutungen:

- `/nfs4root`
  Dabei handelt es sich um die hier verwaltete Freigabe. Im ersten Eintrag ist es das Pseudodateisystem, und in den beiden anderen Einträgen sehen Sie die eigentlichen Freigaben.
- `192.168.56.0/24`
  Alle Hosts aus dem Netzwerk `192.168.56.0` haben Zugriff auf diese Freigaben. Über die Subnetzmaske können Sie den Zugriff bis auf einen Host genau beschränken.
- `ro`
  Das Dateisystem wird `read-only` eingehängt. Da es sich hier um das Pseudodateisystem handelt, ist das die richtige Option, denn unter keinen Umständen soll ein Client direkt in das Pseudodateisystem schreiben dürfen. Dieser Parameter ist die Standardeinstellung.

- `rw`
  Das Dateisystem wird `read-write` eingehängt. Setzen Sie diesen Parameter immer, wenn auf die Freigabe schreibend zugegriffen werden soll. Wenn Sie ihn nicht angeben, wird standardmäßig `read-only (ro)` gemountet.
- `sync/async`
  Die Parameter `sync` und `async` verändern das Verhalten des Servers beim Speichern von Daten, die der Client an den Server sendet.

  Hier sehen Sie das Verhalten bei der Einstellung `sync`:

  1. Ein Programm möchte Daten auf die NFS-Freigabe schreiben.
  2. Der Client leitet die Anfrage an den Server weiter.
  3. Der Server leitet den Schreibzugriff an das lokale Dateisystem weiter.
  4. Das lokale Dateisystem führt den Schreibzugriff aus.
  5. Das lokale Dateisystem meldet dem Server, ob der Schreibzugriff erfolgreich war oder ob er fehlgeschlagen ist.
  6. Der Server teilt dem NFS-Client mit, ob der Schreibzugriff erfolgreich war oder nicht.
  7. Der Client teilt dem Programm mit, ob der Schreibzugriff erfolgreich war oder nicht.

  Hier sehen Sie das Verhalten bei der Einstellung `async`:

  1. Ein Programm möchte Daten auf die NFS-Freigabe schreiben.
  2. Der Client leitet die Anfrage an den Server weiter.
  3. Der Server teilt dem Client mit, dass das Schreiben erfolgreich war.
  4. Der Client teilt dem Programm mit, dass das Schreiben erfolgreich war.
  5. Der Server leitet den Schreibzugriff an das lokale Dateisystem weiter.
  6. Das lokale Dateisystem führt den Schreibzugriff aus.

  Da bei der Verwendung von `async` der Schreibvorgang schon vorzeitig als abgeschlossen an die Anwendung gesendet wird, kann es hier zu Datenverlusten kommen, wenn zum Beispiel die Festplatte voll ist.
- `secure/insecure`
  Über diesen Parameter können Sie steuern, ob ein *NFS-Request* von einem TCP-Port kleiner 1024 stammen muss oder nicht. Die Voreinstellung ist `secure`. Um sie zu deaktivieren, wird `insecure` verwendet.
- `root_squash/no_root_squash`
  Über diesen Parameter steuern Sie, welche Rechte der lokale Benutzer `root` auf dem Client an der Freigabe hat. Setzen Sie hier den Wert `root_squash`, erhält der lokale `root` auf der Freigabe nur die Rechte, die der Benutzer `nobody` auf dem Server hat. Nur wenn der lokale `root` volle Dateisystemrechte erhalten soll, setzen Sie hier den Wert auf `no_root_squash`. Die Standardeinstellung ist `root_squash`.

- `no_subtree_check/no_subtree_check`
  Dieser Parameter schaltet das `subtree-checking` ein oder aus. Mit dem `subtree-checking` können Sie eine weitere Sicherheitsebene einführen. Dieser Parameter wird immer dann relevant, wenn Sie nur ein Verzeichnis freigegeben haben und nicht eine ganze Partition.

  Bei jedem Zugriff auf eine Freigabe muss der NFS-Server prüfen, ob sich die Datei, auf die der Client zugreifen will, in der Partition befindet. Zusätzlich muss er dann noch prüfen, ob sich die Datei auch in dem entsprechenden Verzeichniszweig befindet. Diese Prüfung ist etwas aufwendiger für den NFS-Server. Diese Prüfung wird als `subtree-checking` bezeichnet. Die Standardeinstellung ist `no_subtree_check`.
- `fsid=0`
  Dieser Parameter wird nur für das Pseudodateisystem benötigt. Durch diesen Parameter weiß das System, dass es sich hierbei um das Pseudodateisystem des NFSv4-Servers handelt.

Nachdem Sie alle Änderungen an der Datei */etc/exports* durchgeführt haben, laden Sie den NFS-Server neu. Bei allen Distributionen wird das Neuladen über das Kommando `systemctl` so wie in Listing 16.6 durchgeführt:

```
root@adminbuch:~# systemctl reload nfs-kernel-server
Re-exporting directories for NFS kernel daemon....
```

Listing 16.6 Neuladen der NFS-Serverkonfiguration bei Debian und Ubuntu

Es ist ratsam, an dieser Stelle wirklich nur ein `reload` durchzuführen und kein `restart`. Durch den Neustart des NFS-Servers würden die Clients kurzzeitig die Verbindung zum Server verlieren, und es könnte zu Datenverlusten kommen.

16.3.3 Tests für den NFS-Server

Nachdem Sie den NFS-Server gestartet haben, können Sie mit dem Kommando `rpcinfo -p localhost` den Server testen (siehe Listing 16.7). Dort sehen Sie dann alle NFS-Versionen, die der Server bereitstellen kann:

```
root@adminbuch:/nfs4root# rpcinfo -p localhost
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp  48615  mountd
    100005    1   tcp  42065  mountd
```

```
    100005    2   udp  51865  mountd
    100005    2   tcp  43881  mountd
    100005    3   udp  60791  mountd
    100005    3   tcp  53207  mountd
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049
    100003    3   udp   2049  nfs
    100227    3   udp   2049
    100021    1   udp  59451  nlockmgr
    100021    3   udp  59451  nlockmgr
    100021    4   udp  59451  nlockmgr
    100021    1   tcp  37079  nlockmgr
    100021    3   tcp  37079  nlockmgr
    100021    4   tcp  37079  nlockmgr
```

Listing 16.7 Ausgabe des Kommandos »rpcinfo«

Hier sehen Sie, dass das Protokoll `mountd` nicht für die NFS-Version 4 vorhanden ist. Denn bei NFSv4 ist die Funktion direkt im NFS integriert.

Um zu testen, ob alle Freigaben auch richtig bereitgestellt werden, können Sie den Server mit dem Kommando `exportfs -v` so wie in Listing 16.8 prüfen:

```
root@adminbuch:~# exportfs -v
/nfs4root 192.168.123.0/24(ro,wdelay,insecure,root_squash,\
                           no_subtree_check,fsid=0)

/nfs4root/daten  192.168.123.0/24(rw,wdelay,nohide,insecure,\
                                  root_squash,no_subtree_check)

/nfs4root/home  192.168.123.0/24(rw,wdelay,nohide,insecure,\
                                 no_root_squash,no_subtree_check)
```

Listing 16.8 Testen des NFS-Servers mit »exportfs -v«

Hier sehen Sie, dass ein weiterer Parameter beim Einrichten der Freigaben automatisch mit angegeben wird, nämlich der Parameter `wdelay`. Dieser Parameter sorgt dafür, dass die Schreibvorgänge etwas verzögert ausgeführt werden, wenn der Server mit weiteren Schreibzugriffen rechnet. Dadurch können mehrere Schreibvorgänge gemeinsam durchgeführt werden. Wenn Sie aber immer nur einzelne oder kurze Schreibzugriffe auf Ihren Server haben, können Sie die Standardeinstellung mit dem Parameter `no_wdelay` deaktivieren.

Der Parameter `no_wdelay` hat keinen Effekt, wenn Sie anstelle von `sync` den Parameter `async` verwenden.

16.4 Konfiguration des NFSv4-Clients

Damit der Client die Freigaben nutzen kann, benötigen Sie für jede der Freigaben ein Verzeichnis als Mountpoint auf dem Client. Um zu testen, ob sich die Freigaben ohne Fehler einbinden lassen, testen Sie das Mounten der Freigaben zunächst auf der Kommandozeile. Erst wenn der Test erfolgreich war, tragen Sie das Mounten dauerhaft in der Datei */etc/fstab* ein. Sollte es aufgrund fehlerhafter Einträge in der Datei */etc/fstab* beim Neustart des Systems zu Fehlern beim Mounten kommen, kann es Ihnen passieren, dass das System entweder gar nicht startet oder an der Stelle des Mountens sehr lange hängen bleibt. In Listing 16.9 sehen Sie das Mounten der Freigabe direkt über die Kommandozeile:

```
root@adminbuch-02:~# mount -t nfs4 adminbuch:/daten /daten
root@adminbuch-02:~# mount -t nfs4 adminbuch:/abteilungen /abteilungen
```

Listing 16.9 Mounten der Freigaben am Client

Wie Sie sehen, wird nicht der Pfad des Pseudodateisystems auf dem Server angegeben, sondern das eigentlich freigegebene Verzeichnis.

[+] Wichtig ist hier, dass Sie die Option `-t nfs4` verwenden, da sonst der Client das NFSv3-Protokoll verwenden würde, was auf jeden Fall zu einer Fehlermeldung führt.

Sie können das gesamte Pseudodateisystem in einen Mountpoint einbinden, indem Sie das Wurzelverzeichnis des Pseudodateisystems beim Mounten angeben. Dadurch lassen sich freigegebene Ordner eines Servers zu einer Freigabe zusammenfassen. In Listing 16.10 sehen Sie beispielhaft, wie Sie das Wurzelverzeichnis des Pseudodateisystems einhängen:

```
root@adminrepl:~# mount -t nfs4 adminbuch:/ /alles
```

Listing 16.10 Mounten des Wurzelverzeichnisses des Pseudodateisystems

Wie Sie sehen, wird hier als Quelle einfach das Wurzelverzeichnis angegeben. Dadurch werden alle im Pseudodateisystem eingebundenen Verzeichnisse im lokalen Verzeichnis */alles* zusammen angezeigt. Im Anschluss daran können Sie das erfolgreiche Mounten der Freigabe noch mit dem Kommando `mount` testen. Das Ergebnis sehen Sie in Listing 16.11:

```
root@adminrepl:~# mount
[…]
adminbuch:/daten on /daten type nfs4 (rw,relatime,vers=4.2,\
   rsize=262144,wsize=262144,namlen=255,\
   hard,proto=tcp,timeo=600,retrans=2,sec=sys,\
   clientaddr=192.168.56.85,local_lock=none,addr=192.168.56.81)
adminbuch:/ on /alles type nfs4 (rw,relatime,vers=4.2,\
   rsize=262144,wsize=262144,namlen=255,\
   hard,proto=tcp,timeo=600,retrans=2,sec=sys,\
   clientaddr=192.168.56.85,local_lock=none,addr=192.168.56.81)
```

```
adminbuch:/abteilungen on /abteilungen type nfs4 (rw,relatime,vers=4.2,\
   rsize=262144,wsize=262144,namlen=255,\
   hard,proto=tcp,timeo=600,retrans=2,sec=sys,\
   clientaddr=192.168.56.85,local_lock=none,addr=192.168.56.81)
```

Listing 16.11 Prüfen des Mountens mit »mount«

Jetzt können Sie die Einträge in der Datei */etc/fstab* so wie in Listing 16.12 vornehmen:

```
# NFSv4 Mounts
adminbuch:/daten          /daten          nfs4   rw   0  0
adminbuch:/abteilungen    /abteilungen    nfs4   rw   0  0
adminbuch:/               /alles          nfs4   rw   0  0
```

Listing 16.12 Einträge in der Datei »/etc/fstab«

16.5 Konfiguration des idmapd

Anders als bei NFSv3 werden die UIDs der Benutzer bei NFSv4 nicht automatisch auf die Benutzernamen gemappt. Für das Mapping der Benutzernamen wird sowohl auf dem Client als auch auf dem Server der *idmapd* benötigt.

Wenn Sie Ihren NFS-Server und NFS-Client wie in den vorhergehenden Abschnitten konfiguriert und gemountet haben, sehen Sie bei einem `ls -l` auf dem Client die Ausgabe wie in Listing 16.13:

```
root@adminbuch-02:/daten# ls -l
insgesamt 12
drwxrwx--- 2 root nogroup 4096  8. Aug 12:16 abteilung
drwxrwx--- 2 root nogroup 4096  8. Aug 12:16 buchhaltung
drwxrwx--- 2 root nogroup 4096  8. Aug 12:13 verwaltung
```

Listing 16.13 Auflistung der Rechte ohne »idmapd«

Die Zuordnung der Originalverzeichnisse auf dem NFS-Server sieht aber so aus, wie in Listing 16.14 dargestellt:

```
root@adminbuch:/daten# ls -l
insgesamt 12
drwxrwx--- 2 root benutzer    4096  8. Aug 12:16 abteilung
drwxrwx--- 2 root buchhaltung 4096  8. Aug 12:16 buchhaltung
drwxrwx--- 2 root verwaltung  4096  8. Aug 12:13 verwaltung
```

Listing 16.14 Auflistung der Rechte auf dem Server

Jetzt kommt der *idmapd* ins Spiel. Über RPC-Aufrufe werden nun die UIDs der Benutzer und Gruppen auf die entsprechenden Namen gemappt. Wichtig ist an dieser Stelle, dass die

Kapitel 23
DNS-Server

DNS wird stets unterschätzt und doch immer gebraucht! In diesem Kapitel zeigen wir Ihnen, wie Sie Ihren eigenen Nameserver aufsetzen und betreiben können (als Rekursive-DNS oder als eigenen autoritativen DNS-Server, der selbst Zonen verwaltet), wie Sie es vermeiden, sich am Schwergewicht »DNSSEC« zu verheben, und wie Sie die verschlüsselte Variante »DNS over HTTPS« (DoH) einrichten.

Das *Domain Name System* (DNS) dient primär dazu, Namen in IP-Adressen und IP-Adressen in Namen aufzulösen. In den Anfangstagen des Internets, als es noch *ARPANET* hieß und nur eine Handvoll Teilnehmer hatte, wurde die Namensauflösung von jedem Administrator eines Netzbereichs in einer Datei per Hand vorgenommen. Alle Änderungen mussten dabei stets an die übrigen Teilnehmer versandt werden, damit diese in deren lokalen *hosts*-Dateien eingepflegt werden konnten. Ein ziemlicher Aufwand, der dringend nach einer anderen Lösung verlangte. Diese wurde im Jahre 1983 mit DNS geschaffen: einem hierarchischen, verteilten System zum Auflösen von Namen in IP-Adressen und von IP-Adressen in Namen.

23.1 Funktionsweise

Zur Auflösung von Namen fragt ein Client zunächst immer den lokalen *resolver*. Dieser weiß, über welche Wege Namen aufgelöst werden können. Dabei wird, auch heutzutage noch, zuerst in die lokale Hostdatei geschaut. Wird dort kein Eintrag gefunden, fragt der *resolver* einen der konfigurierten Nameserver. Auf der Serverseite kann eine DNS-Anfrage mit drei verschiedenen Verfahren beantwortet werden:

- **autoritativ**
  Der gefragte Nameserver ist selbst für die Zone verantwortlich und holt die Daten aus einer lokalen Zonendatei.
- **nichtautoritativ**
  Der gefragte Nameserver ist nicht selbst für die Zone verantwortlich und muss die Daten ermitteln. Dabei wird zwischen den folgenden Abfragearten unterschieden:
  - *rekursiv*
    Der Server holt die Daten von einem anderen Nameserver, arbeitet also als Proxy (Stellvertreter): Client-Server-Verfahren.

– *iterativ*
  Der Server antwortet mit einem oder mehreren Verweisen oder einem *Resource Record* auf andere Nameserver, die den Namen auflösen können: Verfahren zwischen Nameservern.

Üblicherweise werden Clientanfragen rekursiv gestellt. Daher beginnt der Ablauf also damit, zu prüfen, wer für eine Zone zuständig ist. Dabei wird das Pferd von hinten aufgezäumt. Falls der gefragte DNS-Server nicht für die Zone selbst zuständig ist, fragt er die Root-Server (die die oberste Hierarchieebene im DNS darstellen), wer für die angefragte TLD (*Top Level Domain*) zuständig ist.

Ist diese Information bekannt, fragt er die Server der TLD-Ebene, wer für die Domäne verantwortlich ist. Anschließend wird der zuständige DNS-Server gefragt, wie der Name lautet, und die Antwort wird dem Client zurückgegeben.

Der Ablauf erfolgt also streng hierarchisch – von oben nach unten, vom Punkt zum Hostnamen. Dabei stellen Clients stets rekursive Anfragen. Im Hintergrund werden aber alle Verfahren genutzt, um eine Anfrage beantworten zu können (siehe Abbildung 23.1).

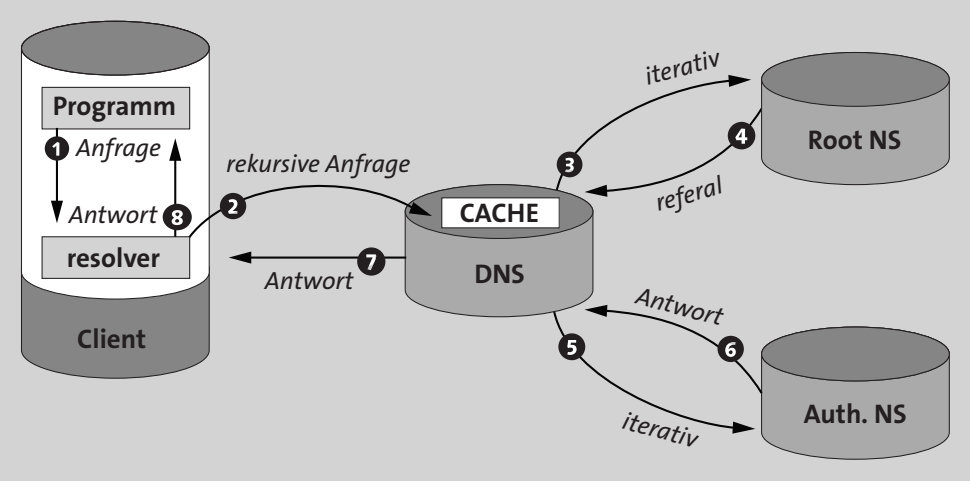


Abbildung 23.1 Ablauf einer Namensauflösung

[+]

Abschluss mit: ».«

DNS-Namen müssten eigentlich immer mit einem abschließenden Punkt angegeben werden, also zum Beispiel als "*www.rheinwerk-verlag.de.*", da dieser Punkt den Namen abschließt.

Der abschließende Punkt kann zwar weggelassen werden, ist aber streng genommen Bestandteil des Namens. Auf diese Besonderheit kommen wir im weiteren Verlauf des Kapitels noch zurück.

23.1.1 Unterschied: rekursiv und autoritativ

Bei DNS-Servern wird zwischen rekursiven und autoritativen Servern unterschieden. Ein rekursiver DNS ist »nur« ein Proxy-Server (man könnte auch »Stellvertreter« sagen). Er holt die angefragten Informationen ab und stellt das Ergebnis zur Verfügung. Von einem rekursiven Server erhalten Sie stets sogenannte *non authoritative*, also nicht autorisierte Antworten, da er selbst nicht für abgefragte Namen zuständig ist.

Ein autoritativer Server hingegen ist für eine (oder mehrere) Zonen zuständig und beantwortet nur Anfragen an Zonen, für die er selbst zuständig ist. Dafür ist die Rückmeldung aber autoritativ, also verbindlich.

23.1.2 Einträge im DNS: Resource Records

Namen sind nicht gleich Namen. Was auf den ersten Blick komisch anmutet, ist aber rein logisch zu verstehen. Damit die DNS-Server die Unterschiede in den jeweiligen Anfragen ausmachen können, wurden die sogenannten *Resource Records* (oder kurz *Records* oder *RR*) geschaffen. Anhand dieser Records kann ein DNS-Server unterscheiden, welcher Natur die Anfrage ist und wie diese beantwortet werden muss. In der nachstehenden Auflistung haben wir für Sie die gängigsten Record-Typen und deren Bedeutung aufgelistet:

- `A`
  liefert eine IPv4-Adresse zu einem Namen zurück. Typische Antwort: `192.168.0.20`
- `AAAA`
  liefert eine IPv6-Adresse zu einem Namen zurück. Typische Antwort:
  `fe80::a00:27ff:fee9:eb75`
- `PTR`
  liefert einen Namen zu einer IPv4/IPv6-Adresse zurück – auch als *Reverse Record* bezeichnet. Typische Antwort: `proxy.example.net.`
- `CNAME`
  stellt einen Verweis, eine Weiterleitung oder einen Alias dar.

Nicht auf Domänenebene

Beachten Sie, dass *CNAME*-Records nur auf Namen und nicht auf die Zone selbst verweisen dürfen. So darf der Name *www.example.net* als *CNAME* auf *example.net* verweisen, allerdings können Sie nicht *example.net* auf *www.example.net* verweisen lassen!

- `MX`
  liefert den zuständigen Mailserver (*mail exchange*) für eine Zone zurück. Typische Antwort: `10 mail.example.net.`

- `NS`
  liefert den zuständigen Nameserver für eine Zone zurück. Typische Antwort: `dns.example.net.`
- `SRV`
  liefert einen Server zurück, der einen Dienst anbietet, der meist in einem Windows-Active Directory zu finden ist. Mögliche Antwort: `ldap.example.net.`
- `TXT`
  liefert einen Text zurück. Mögliche Antwort: `"Hello World"`
- `SOA`
  liefert einen Ansprechpartner und Parameter zur abgefragten Zone zurück (SOA: engl. für *Start of Authority*). Typische Antwort:
  `dns.example.de. admin.example.net. 2021010401 28800 7200 604800 3600`

Dies sind bei Weitem nicht alle Record-Typen. Das *Domain Name System* hält noch viele Möglichkeiten bereit, die ein normalsterblicher Nutzer selten wahrnimmt, die für den Betrieb von Diensten (wie MX-Records für Mail oder SRV-Records für ein Windows-AD) oder des Internets jedoch unerlässlich sind. Jeder Record-Typ hat eine unterschiedliche Anzahl an Werten, die er ausliefern kann. Liefern zum Beispiel A- oder PTR-Records nur die IP-Adressen oder Namen zurück, so wird bei MX-Records gleichzeitig noch eine Gewichtung in Form eines Zahlenwerts mit ausgeliefert.

Bei SRV-Records wird sogar noch die Portnummer des angebotenen Dienstes mit ausgeliefert. Im Laufe der Zeit ist das ursprünglich als einfaches Adressbuch gedachte DNS-System immer weiter gewachsen.

23.1.3 Die Grundkonfiguration

Der wohl bekannteste und weltweit am meisten genutzte Nameserver ist der *Berkeley Internet Name Daemon* (BIND). Die Software wird ähnlich wie der DHCP-Server vom ISC gepflegt und weiterentwickelt. BIND ist für so gut wie jede Plattform verfügbar. Je nach Distribution heißt das passende Paket *bind* oder *bind9*.

Nach der Installation aus den Paketquellen müssen Sie zunächst einige grundlegende Einstellungen vornehmen und festlegen, für welche *Zonen* (Domains) Sie verantwortlich sind. Diese Einstellungen werden in der *named.conf* vorgenommen.

Je nach Distribution finden Sie diese entweder direkt unter */etc* oder in dem Unterverzeichnis */etc/bind*. Bei einigen Distributionen (wie zum Beispiel Debian und Ubuntu) wird die Konfiguration auch auf verschiedene Dateien aufgeteilt, sodass in der Hauptkonfigurationsdatei lediglich die einzelnen Konfigurationsdateien inkludiert werden. Listing 23.1 zeigt den Ausschnitt der Datei *named.conf* von einem Debian- oder Ubuntu-System:

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

Listing 23.1 »Include«-Einträge in der »named.conf«

Im Folgenden wird stets mit der Datei *named.conf* gearbeitet. Der erste entscheidende Block in der Konfigurationsdatei ist der `options`-Block. Jeder Block wird durch geschweifte Klammern eingeleitet und beendet, wobei jede Zeile (auch das Ende eines Blocks) mit einem Semikolon abgeschlossen werden muss. Kommentare können durch // oder # eingeleitet werden. Ganze Kommentarblöcke können durch /* und */ gekapselt werden. Listing 23.2 zeigt einen `options`-Block:

```
options {
        directory "/var/cache/bind";

        dnssec-validation auto
        listen-on-v6 { any; };
};
```

Listing 23.2 »options«-Block aus der »named.conf«

Die Direktive `directory` spezifiziert das Arbeitsverzeichnis des Nameservers. Dateien, die innerhalb der Konfigurationsdatei angegeben werden, beziehen sich immer relativ auf dieses Verzeichnis. Die Angabe von absoluten Pfaden ist aber auch möglich.

Unterschiedliche Pfade je Distribution

Bei SUSE-Systemen wird hier standardmäßig der Pfad */var/lib/named* verwendet, bei CentOS */var/named* und unter Debian und Ubuntu */var/cache/bind*. Um die Beispiele übersichtlicher zu gestalten, beziehen wir uns darin immer auf */var/cache/bind*.

Mit `listen-on-v6` kann bestimmt werden, auf welchen lokalen IPv6-Adressen der Nameserver lauschen soll. Im Beispiel aus Listing 23.2 wurde für `listen-on` die Option `any` verwendet – dadurch lauscht der Nameserver einfach auf allen IPv4- und IPv6-Interfaces des Servers. Durch die Angabe von `none` bei der Direktive `listen-on-v6` wird es abgeschaltet.

Um den Nameserver auf bestimmte Adressen zu beschränken, können Sie eine Liste von IPv6-Adressen angeben, auf denen der Nameserver lauschen soll – für IPv4-Adressen müssen Sie die Direktive `listen-on` verwenden. Selbstverständlich müssen die einzelnen Einträge jeweils mit einem Semikolon abgeschlossen werden. Auch die Angabe eines Ports ist möglich, was allerdings nur selten sinnvoll ist. Der Standardport ist 53 (TCP und UDP). In Listing 23.3 sehen Sie eine Mehrfachzuweisung von IP-Adressen und Ports:

```
listen-on port 10053 { 192.168.1.100; 192.168.2.100; };
listen-on { 192.168.100.100; };
```

Listing 23.3 Angabe der Ports und IP-Adressen über »listen-on«

Um sich den DNS-Cache Ihres Providers zunutze zu machen, können Sie mit der Direktive `forwarders` benachbarte Nameserver angeben. Bei Anfragen, die nicht aus Ihrem Cache beantwortet werden können, werden diese dann befragt.

Darüber hinaus kann mit der Direktive `forward` festgelegt werden, ob Anfragen ausschließlich an die anderen Nameserver weitergeleitet werden sollen (`only`) oder ob Ihr Server beim Scheitern der Abfrage selbst eine rekursive Anfrage starten soll (`first`). In Listing 23.4 sehen Sie ein Beispiel für die `forwarders`-Direktive:

```
options {
        […]
        forward first;
        forwarders {
                10.10.10.1;
                10.10.20.1;
         };
        […]
};
```

Listing 23.4 Beispiel für die »forwarders«-Direktive

Der nächste entscheidende Block beinhaltet die Zonendefinitionen, die mit `zone` eingeleitet werden. Um z. B. die Domain *example.net* zu betreiben, müssen Sie folgende Zone definieren:

```
zone "example.net" {
        type primary;
        file "primary/example.net";
};
```

Listing 23.5 Definition der Zone »example.net«

Die Direktive `type` gibt den Zonentyp an. In diesem Fall ist es `primary`, weil dieser Server der primäre DNS-Server der Domain *example.net* werden soll. Mit der Direktive `file` wird angegeben, in welcher Datei die Zonendaten für *example.net* liegen. Wie bereits erörtert wurde, ist der Pfad relativ zu dem mit `directory` gesetzten Pfad im `options`-Block (im Beispiel also *var/cache/bind/primary/example.net*).

23.1.4 Zonendefinitionen

Eine Zone enthält alle relevanten Informationen für eine Subdomain. Diese einzelnen Informationen werden als sogenannte *Resource Records* (RR) hinterlegt. Auch in Zonendateien

können Kommentare mit einem Semikolon eingeleitet werden. Vor dem ersten RR müssen Sie mit `$TTL` die Standard-*Time-To-Live* für alle RRs angeben. Dieser Wert gibt an, wie lange Einträge gültig sind, bevor diese neu abgefragt werden müssen. Die Angabe erfolgt dabei in Sekunden. Mit unterschiedlichen Suffixen können aber auch Wochen `w`, Tage `d`, Stunden `h` oder Minuten `m` angegeben werden.

Nach der TTL folgt der wichtigste Resource Record, der *SOA*. Das Kürzel steht für *Start Of Authority* und definiert die Zuständigkeit und die Eigenschaften der Zone. Abbildung 23.2 zeigt den Aufbau eines SOA-Records.

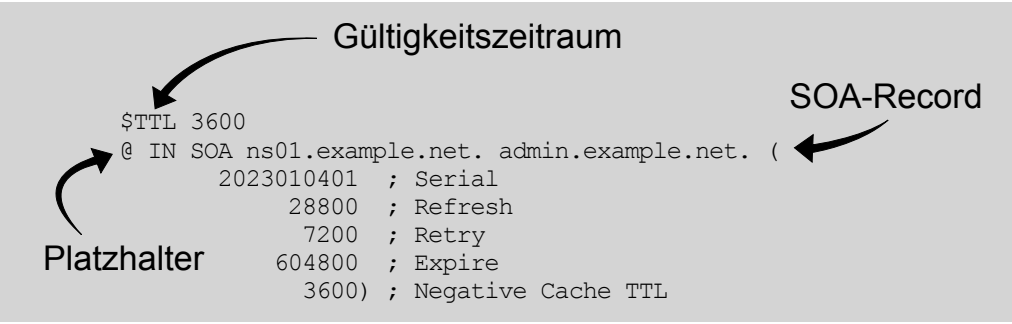


Abbildung 23.2 Aufbau des »SOA«-Records

Das @-Zeichen am Anfang der zweiten Zeile (zu Beginn des SOA-Records) gibt an, dass sich alle Einträge auf die in der */etc/named.conf* definierte Domain beziehen. Die Schreibweise hat den Vorteil, dass von mehreren Zonendefinitionen auf die gleiche Zonendatei verwiesen werden kann. So können Sie zum Beispiel die Records der Domänen *example.de*, *example.net* und *example.com* in nur einer Datei vorhalten. In diesem Beispiel verweisen wir von *example.net* auf die Zonendatei. Ausgeschrieben sähe die erste Zeile des SOA-Records so aus:

```
example.net.        IN      SOA     ns1.example.net. admin.example.net. (
```

Listing 23.6 »SOA-Record« ohne die Verwendung des Platzhalters »@«

Mit `IN` wird angegeben, dass es sich bei dem folgenden RR um die Klasse *Internet* handelt. Nach der Angabe des Record-Typs mit `SOA` folgt der FQDN des primären Nameservers der Zone, in diesem Fall ist es `ns1.example.net.` (die Angabe muss mit einem Punkt abgeschlossen werden!). Danach folgt die E-Mail-Adresse des zuständigen Administrators.

Das @-Zeichen muss hier allerdings durch einen Punkt ersetzt werden – somit steht `admin.example.net` also für `admin@example.net`. Falls die E-Mail-Adresse vor dem @-Zeichen einen Punkt enthält, muss dieser mit einem Backslash maskiert werden. Die E-Mail-Adresse `max.mustermann@example.net` muss daher als `max\.mustermann.example.net` angegeben werden. Anschließend folgen die SOA-spezifischen Konfigurationen in Klammern. Dabei müssen folgende Angaben gemacht werden:

- `Serial`
  `Serial` steht für die Versionsnummer der Zone. Bei jeder Änderung an der Zonendatei muss sie inkrementiert werden. Die Seriennummer wird von den Secondary Nameservern benutzt, um zu sehen, ob sie die Zone neu herunterladen müssen. Der Wertebereich für `Serial` beträgt 32 Bit. Im Prinzip könnten Sie bei 1 anfangen zu zählen. Es hat sich jedoch eingebürgert, `Serial` in folgendem Format darzustellen: YYYYMMDDXX. Dabei steht YYYY für die vierstellige Jahreszahl, MM für den aktuellen Monat, DD für den Tag des Monats und XX für einen fortlaufenden Zähler, der angibt, wie oft die Zone bereits an diesem Tag geändert wurde.
- `Refresh`
  Mit `Refresh` wird angegeben, nach welcher Zeit die Secondary Nameserver beim Primary nachschauen sollen, um zu prüfen, ob die `Serial` sich geändert hat.
- `Retry`
  `Retry` legt fest, nach welcher Zeit ein Secondary Nameserver erneut nachfragen soll, wenn der erste Versuch fehlgeschlagen ist.
- `Expire`
  Mit `Expire` wird quasi das Haltbarkeitsdatum für die Zone der Secondary Nameserver angegeben. Sollte der Secondary Nameserver nach Ablauf dieser Zeit noch immer keinen Kontakt zum Primary herstellen können, so wird die Zone ungültig.
- `Negative Cache TTL` (früher `minimum`)
  Gibt die Zeit in Sekunden an, die eine negative Rückmeldung des Servers im Cache vorgehalten werden darf.

[!]

Beachten Sie die Bedeutung von »Negative Cache TTL« (früher: »minimum«)

Die `Negative Cache TTL` (früher `minimum`) besaß im Laufe der Zeit unterschiedliche Bedeutungen. Sie werden daher noch in vielen Beispielen im Internet und in älteren Büchern eine andere Bedeutung finden. Ursprünglich gab der Wert an, wie lange die Standard-TTL der Resource Records dauert. Daher stammt auch der Name *minimum*. Durch RFC3208 wurde das aber abgelöst. Stattdessen wird heutzutage der $TTL-Eintrag aus der ersten Zeile der Zonendatei verwendet.

Richtig gesetzte Werte entscheiden über Erfolg und Misserfolg Ihres DNS-Servers. Die DENIC zum Beispiel schreibt für *.de*-Domains folgende Wertebereiche vor:

- `refresh`: 3.600 – 86.400 Sekunden
- `retry`: 900 – 28.800 Sekunden
- `expire`: 604.800 – 3.600.000 Sekunden
- `minimum (negTTL)`: 180 – 86.400

Auf *www.denic.de/hintergrund/nameservice/nameserver-und-nsentry-eintraege.html* finden Sie weitere Informationen zu den vorgeschriebenen Werten. Nach dem SOA-Record folgen weitere RR. Für eine gültige Zonendatei muss mindestens noch ein *NS*-Record definiert werden, der den verantwortlichen Nameserver für die Domain bestimmt. Der Resource Records sind dabei stets so aufgebaut:


```
<NAME> <TTL> <CLASS> <TYPE> <RDATA>
```

Listing 23.7 Aufbau eines »Resource Records«

Dabei haben die einzelnen Begriffe folgende Bedeutung:

- `<NAME>`
  gibt den Namen für den RR an. Für einen Namen, zum Beispiel *www.example.net*, kann sowohl die Kurzform `www` als auch die Langform `www.example.net.` verwendet werden. Bei der Langform ist der abschließende Punkt Pflicht – ohne diesen würde BIND den Zonennamen anfügen. Mit der Angabe ohne Punkt würde der Name also als `www.exmaple.net.example.net` erkannt werden. Wenn mehrere RR für den gleichen Namen definiert werden sollen, kann der Name weggelassen werden. Stattdessen muss die Zeile mit einem Leerzeichen beginnen – wir empfehlen Ihnen aber zur Wahrung der Übersicht, diese Funktion nicht zu verwenden.
- `<TTL>`
  Die TTL gibt an, wie lange der Eintrag im Cache gehalten werden darf. Die Angabe der TTL ist optional. Wird keine TTL angegeben, so wird die Standard-TTL verwendet, die mit `$TTL` in der ersten Zeile der Zonendatei angegeben wurde.
- `<CLASS>`
  Hier wird der Klassentyp angegeben. Der Klassentyp muss nur einmal während der Zonendefinition angegeben werden. In der Regel wird `IN` für *Internet* verwendet. Andere mögliche Werte sind `CH` für *Chaosnet* oder `HS` für *Hesiod*, diese werden aber kaum noch benutzt und spielen eigentlich keine Rolle mehr.
- `<TYPE>`
  Hier wird der Typ des RR angegeben. Die gebräuchlichsten Typen haben wir Ihnen bereits zu Beginn des Abschnitts vorgestellt. Diese werden wir gleich näher betrachten.
- `<RDATA>`
  Unter `<RDATA>` werden die Daten des Resource Records abgelegt. In der Regel sind das eine IP-Adresse, ein Name oder etwas komplexere Angaben wie beim SOA-Record.

Schauen wir uns die gebräuchlichsten Records einmal genauer an:

- **A- und AAAA-Records**
  Der am häufigsten verwendete Record ist der A-Record, da dieser für die Zuweisung von Hostnamen zu IPv4-Adressen verwendet wird. Ein A-Record wird wie folgt definiert:

  ```
  ns1.example.net.       12h     IN      A       192.168.2.10
  ```

Für IPv6-Adressen steht der AAAA-Record zur Verfügung:

```
ns1.example.net.        12h     IN      AAAA    2001:db8::1
```

- **NS-Records**
  Mit `NS-Records` werden die für eine Zone zuständigen Nameserver definiert. Aus Gründen der Redundanz sollten Sie mindestens zwei Nameserver in unterschiedlichen Netzen haben. Standardmäßig sieht ein NS-Record wie folgt aus:

```
example.net.  12h     IN      NS      ns1.example.net.
example.net.  12h     IN      NS      ns2.example.net.
```

[!]

Achtung: »Glue« nicht vergessen!

Befindet sich der Nameserver für eine Zone (wie in diesem Beispiel) selbst in der Zone, muss ein sogenannter *Glue-Record* erzeugt werden. Ein Glue-Record ist nichts weiter als ein *A-Record* für den Nameserver, der eine Ebene höher definiert wird. In diesem Fall wäre ein Glue-Record in der Zone `.net` notwendig. Ohne Glue-Record könnten Clients sich nicht an den für *example.net* verantwortlichen Nameserver wenden, da die IP-Adresse des Nameservers ja erst in der Zone definiert wird.

- **MX-Records**
  Die Angabe der für eine Zone zuständigen Mailserver erfolgt mit dem *Mail Exchange* (MX)-Record. Ein MX-Record besteht immer aus einer Präferenz und dem Namen des Mailservers:

```
example.net.            12h     IN      MX      10      mx1.example.net.
example.net.            12h     IN      MX      10      mx2.example.net.
```

  Über die Präferenz wird festgelegt, welcher der Mailserver bevorzugt genutzt werden soll (niedrigerer Wert = höhere Priorität). Der Wertebereich für die Präferenz ist 16 Bit groß, dennoch hat es sich durchgesetzt, als Präferenz immer Vielfache von 10 zu verwenden. Die Präferenz 10 hat also die höchste Priorität. Es ist durchaus möglich (und in der Regel auch sinnvoll), mehrere Server mit der gleichen Präferenz anzugeben. Der einliefernde Mailserver nimmt dann zufällig einen der Mailserver (dies wird als *DNS-Round-Robin* bezeichnet).

  Ist dieser nicht erreichbar, wird ein anderer Mailserver der gleichen Priorität gewählt. Erst wenn alle MX-Server der gleichen Priorität durchprobiert wurden, darf der einliefernde Mailserver sich an Server einer niedrigeren Priorität wenden – dabei halten sich Spammer übrigens bewusst nicht an diese Regel. Viele Postmaster versäumen es nämlich leider, den sekundären Mailserver genauso gut zu sichern wie den primären. Daher versuchen Spammer oft gezielt, ihre Mails bei den MX-Servern mit der niedrigeren Priorität loszuwerden.

Bei merkwürdigen Verbindungsversuchen RFC beachten!

Beachten Sie, dass sich, wenn für eine Domain kein MX-Eintrag vorhanden ist, Mailserver laut RFC an den *A-Record* der Domäne wenden sollen! Falls Sie also ungewöhnliche Zugriffsversuche feststellen sollten, wird dies voraussichtlich am RFC-Standard liegen.

- **CNAME-Record**
  Mit dem `CNAME-Record` *(Canonical Name Record)* können Aliase (auch als *Verweise* bezeichnet) definiert werden:

  ```
  www.example.net.        3600    IN      CNAME   server01.example.net.
  ```

  Dabei gibt es ein paar Besonderheiten, die Sie beachten müssen. Weder NS-Records noch Zonen selbst dürfen auf einen *CNAME* zeigen. Darüber hinaus sollte bei MX-Records auf *CNAME*-Records verzichtet werden.

23.1.5 Die erste vollständige Zone

Mit den bisher vorgestellten Resource Records lässt sich bereits eine komplette Zone definieren. In Listing 23.8 sehen Sie ein Beispiel für die Zone *example.net*:

```
$TTL 12h
@       IN      SOA     ns1.example.net. admin.example.net. (
                        2023010701      ; Serial
                        12h             ; Refresh
                        1h              ; Retry
                        10d             ; Expire
                        1h              ; Negative Cache TTL
                        )
                IN      NS      ns1.example.net.
                IN      NS      ns2.example.net.

                IN      MX      10      mx1
                IN      MX      10      mx2

ns1             IN      A       192.168.2.10
ns2     24h     IN      A       192.168.100.10
mx1             IN      A       192.168.2.11
mx2             IN      A       192.168.2.12
www             IN      CNAME   ns1
```

Listing 23.8 Beispiel für eine »forward«-Zone

Nach einem Neustart des Servers können Sie die Konfiguration mit dem `dig`-Kommando überprüfen:

```
root@saturn:~# dig axfr example.net @localhost
; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> axfr example.net @localhost
;; global options: +cmd
example.net.   43200   IN   SOA  ns1.example.net. admin.example.net. \
                                        2023010201 43200 3600 864000 3600
example.net.          43200   IN      NS     ns1.example.net.
example.net.          43200   IN      NS     ns2.example.net.
example.net.          43200   IN      MX     10 mx1.example.net.
example.net.          43200   IN      MX     10 mx2.example.net.
mx1.example.net.      43200   IN      A      192.168.2.11
mx2.example.net.      43200   IN      A      192.168.2.12
ns1.example.net.      43200   IN      A      192.168.2.10
ns2.example.net.      86400   IN      A      192.168.100.10
www.example.net.      43200   IN      CNAME  ns1.example.net.
example.net.   43200   IN   SOA ns1.example.net. admin.example.net. \
                                       2023010201 43200 3600 864000 3600
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (TCP)
;; WHEN: Mon Jan 02 10:14:05 UTC 2023
;; XFR size: 11 records (messages 1, bytes 304)
```

Listing 23.9 Anzeige der »forward«-Zone mit »dig«

Die Option `axfr` des Befehls `dig` leitet einen Zonentransfer ein, daher wird die gesamte Zone geladen und dargestellt. Das ist für den Produktivbetrieb nicht unbedingt gewollt und muss noch eingeschränkt werden.

Mehr dazu finden Sie in Abschnitt 23.1.8, »Secondary-Server«. Zum Testen der ersten Konfiguration ist es aber ideal. Durch die Angabe von `@localhost` wird `dig` mitgeteilt, dass die Anfrage an *localhost* gesendet werden soll und nicht an den ersten Server aus der */etc/resolv.conf*.

Wie Sie in Listing 23.9 sehen können, werden die optionalen Werte wie `IN` und `TTL` automatisch bei allen Einträgen ergänzt, und die Zeitangaben werden in Sekunden umgerechnet, da dies das Standardformat ist.

23.1.6 Die hint-Zone

Um einen voll funktionsfähigen Server aufzusetzen, der auch Anfragen rekursiv auflösen kann, die außerhalb der eigenen Zonen (*example.net*) sind, müssen wir dem Server noch mitteilen, wo er die Root-Server findet.

Diese Konfigurationen werden in der sogenannten *hint*-Zone vorgenommen. Analog zur Zone *example.net* muss diese in der *named.conf* angegeben werden:

```
zone "." {
        type hint;
        file "/usr/share/dns/root.hints";
};
```

Listing 23.10 Eintrag für die »hint«-Zone in der »named.conf«

Die dazugehörige Zonendatei befindet sich entweder unter */usr/share/dns/root.hints*, unter */var/named/named.ca* oder unter */var/lib/named/root.hint* und wird mit dem BIND mitgeliefert. Aktuelle Versionen können von InterNIC unter *www.internic.com/domain/named.root* bezogen oder mittels `dig` abgefragt werden – setzen Sie dafür den Befehl `dig +bufsize=1200 +norec NS . @a.root-servers.net` ab (was zum Beispiel CentOS während der Installation durchführt). Listing 23.11 zeigt einen Ausschnitt aus der *hint*-Zone:

```
…
; FORMERLY NS.INTERNIC.NET
;
.                       3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.     3600000      AAAA  2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     199.9.14.201
B.ROOT-SERVERS.NET.     3600000      AAAA  2001:500:200::b
[…]
```

Listing 23.11 Ausschnitt aus der »hint«-Zone

Nach Abschluss der Konfiguration können Sie probieren, externe Domains aufzulösen. In Listing 23.12 sehen Sie die Auflösung einer externen Domäne:

```
root@debian:~# dig www.rheinwerk-verlag.de @localhost

; <<>> DiG 9.16.33-Debian <<>> www.rheinwerk-verlag.de @localhost
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46013
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 6e777cdbc29499180100000063b2b1ed64f8977f12d2f271 (good)
;; QUESTION SECTION:
;www.rheinwerk-verlag.de. IN A
```

```
;; ANSWER SECTION:
www.rheinwerk-verlag.de. 86400 IN A 46.235.24.168

;; Query time: 0 msec
;; SERVER: ::1#53(::1)
;; WHEN: Mon Jan 02 11:29:01 CET 2023
;; MSG SIZE  rcvd: 96
```

Listing 23.12 Auflösung einer externen Domäne

Herzlichen Glückwunsch, es funktioniert! Wie Sie in Listing 23.12 sehen können, hat *www.rheinwerk-verlag.de* die IP-Adresse 46.235.24.168. Die TTL des Eintrags beträgt 86.400 Sekunden, also 24 Stunden. Dass der Cache funktioniert, zeigt ein weiterer Aufruf:

```
root@saturn:~# dig www.rheinwerk-verlag.de @localhost
[…]
;; ANSWER SECTION:
www.rheinwerk-verlag.de. 86354  IN      A       46.235.24.168
[…]
;; Query time: 0 msec
[…]
```

Listing 23.13 Erneute Abfrage der externen Domäne

Die `Query time` der Anfrage war mit 0 Millisekunden recht schnell (üblicherweise kann die lokale Auflösung bis zu 5 Millisekunden in Anspruch nehmen), und der Timer der `TTL` steht bereits auf 86.354 Sekunden. Nach Ablauf dieser Zeit ist eine neue Anfrage bei einem der beiden für *www.rheinwerk-verlag.de* verantwortlichen Nameserver notwendig.

23.1.7 Reverse Lookup

Bisher haben wir lediglich Namen in IP-Adressen aufgelöst. Nun widmen wir uns der *Rückwärtsauflösung* (*Reverse Lookup*), also der Auflösung der IP-Adresse in einen Domainnamen. Für einen DNS-Server sind alles Namen, auch IP-Adressen. Um diese effektiv auflösen zu können, wurde das Konzept des Reverse Lookup geschaffen. Dabei werden die Zonendateien, in denen die Zuordnung von IP-Adressen zu Namen vorgenommen wird, in umgekehrter Reihenfolge aufgebaut. Diese Zonen haben immer das Suffix *in-addr.arpa* – davon können Sie ableiten, wann diese Funktion implementiert wurde (richtig: zu Zeiten des ARPANET).

Ein Reverse Lookup von 194.8.219.19 wird über diesen Weg auf *19.219.8.194.in-addr.arpa* umgebogen. Auf diese Weise wird die sonst auch übliche baumartige DNS-Struktur aufgebaut. Daher können Teilnetze an unterschiedliche Nameserver delegiert werden. In vielen Fällen sorgen Nameserver Ihres Providers für die Rückwärtsauflösung, insbesondere dann, wenn Ihnen ein kleineres Netz als /24 zugeteilt wird. Es gibt zwar Möglichkeiten, auch kleinere Netze für Reverse Lookups zu delegieren, aber nicht alle Provider bieten das an.

In Listing 23.14 sehen Sie ein Beispiel für eine Rückwärtsauflösung:

```
root@saturn:~# host 194.8.219.19
19.219.8.194.in-addr.arpa domain name pointer mail.rheinwerk-verlag.de.
```

Listing 23.14 Reverse Lookup für 194.8.219.19 mit dem »host«-Kommando

Für die Auflösung Ihrer IP-Adressen in Namen müssen Sie zunächst eine Zone definieren und diese dann mit Informationen befüllen. Der dafür notwendige Resource Record heißt *PTR* (*pointer*, engl. für *Zeiger*) – siehe Listing 23.15:

```
zone "2.168.192.in-addr.arpa" {
        type primary;
        file "reverse/db.2.168.192";
};
```

Listing 23.15 Ergänzungen in der »named.conf«

Wie schon für die Forward-Zone müssen Sie auch für die Reverse-Zone eine Zonendatei erstellen. In unserem Beispiel ist das die Datei */var/cache/bind/reverse/db.2.168.192*. Erstellen Sie diese Datei, und fügen Sie den Inhalt aus Listing 23.16 ein:

```
$TTL 12h
@       IN      SOA     ns1.example.net. admin.example.net. (
                        2               ; Serial
                        12h             ; Refresh
                        1h              ; Retry
                        10d             ; Expire
                        1h              ; Negative Cache TTL
                        )
                IN      NS      ns1.example.net.
                IN      NS      ns2.example.net.
10              IN      PTR     ns1.example.net.
11              IN      PTR     mx1.example.net.
12              IN      PTR     mx2.example.net.
```

Listing 23.16 Zonendatei für die »PTR-Records«

Mit dem `host`-Kommando können Sie nach einem Neustart des Nameservers die Konfiguration überprüfen. In Listing 23.17 sehen Sie eine Überprüfung der Rückwärtsauflösung:

```
root@saturn:~# host 192.168.2.10 localhost
Using domain server:
Name: localhost
Address: 127.0.0.1#53
Aliases:
10.2.168.192.in-addr.arpa domain name pointer ns1.example.net.
```

Listing 23.17 Beispiel für eine Rückwärtsauflösung

Damit auch das `host`-Kommando den lokalen Nameserver verwendet, müssen Sie diesen nach dem abzufragenden Namen angeben – also nicht wie bei `dig` mit einem führenden @-Zeichen versehen.

23.1.8 Secondary-Server

Hochverfügbarkeit muss bei Nameservern nicht über einen Cluster gelöst werden. Das Protokoll selbst liefert bereits alle Möglichkeiten, um die Verfügbarkeit gewährleisten zu können. Dabei ist Hochverfügbarkeit nicht nur in Ihrem eigenen Interesse zu gewährleisten, die *DENIC* schreibt für *de*-Domains zum Beispiel zwei bis fünf Nameserver vor. Dabei ist einer der Server stets der *Primary DNS* und damit der Taktgeber. Nur auf ihm werden die Zonendateien gepflegt. Die *Secondary DNS* oder Secondary-Server holen sich in regelmäßigen Abständen (siehe den SOA-Record) die Zonendateien vom Primary, antworten auf Anfragen für die Zone aber auch autoritativ.

Die Konfiguration eines Secondary-Servers ist denkbar einfach. Installieren Sie einen weiteren BIND-Server, und teilen Sie ihm in der *named.conf* mit, für welche Zonen er Secondary spielen soll. Bei der Zonenkonfiguration müssen Sie dann den Zonentyp auf `secondary` setzen. Zusätzlich muss mit `primaries` angegeben werden, wer der primäre Nameserver der Zone ist. In Listing 23.18 sehen Sie ein Beispiel für die Einträge in der *named.conf* eines Secondary-Servers:

```
zone "example.net" {
        type secondary;
        file "secondary/example.net";
        primaries { 192.168.2.10; };
};
```

Listing 23.18 Eintrag für eine Secondary-Zone in der »named.conf« auf dem Secondary-Server

Damit der Zonentransfer, also das Laden der Zone vom primären Nameserver, funktioniert, muss die bei `file` angegebene Datei für den BIND-Server beschreibbar sein. Auf dem Primary sind prinzipiell keine weiteren Konfigurationen notwendig. Um aber zu verhindern, dass jeder einfach Ihre Zonendateien abfragen kann, sollten Sie Zonentransfers unbedingt auf die Secondary-Server beschränken. Dafür müssen Sie im `options`-Block auf dem Primary mit der Direktive `allow-transfer` eine Liste von IP-Adressen setzen (siehe Listing 23.19).

```
options {
    directory "/var/cache/bind";
    listen-on-v6 { none; };
    allow-transfer { 127.0.0.1; 192.168.100.10; };
};
```

Listing 23.19 Einträge für den Zonentransfer auf dem Primary

Mit der Konfiguration aus Listing 23.19 erlauben Sie für alle Zonen den Transfer von `127.0.0.1` und `192.168.100.10`. Falls Ihre Zonen von unterschiedlichen Nameservern gehalten werden, können Sie die Direktive auch in den `zone`-Blöcken setzen.

Nach einem Neustart des Nameservers wird der Zonentransfer automatisch angestoßen. Nach wenigen Sekunden haben Sie auf dem Secondary-Server dann im angegebenen Verzeichnis die Zone Ihres Primary-Servers. Sehen wir uns nun die Zonendatei auf dem Secondary-Server etwas genauer an (siehe Listing 23.20):


```
example.net.        43200 IN SOA ns1.example.net. admin.example.net.\
                                                2023010203 43200 3600 864000 3600
example.net.        43200 IN NS ns1.example.net.
example.net.        43200 IN NS ns2.example.net.
example.net.        43200 IN MX 10 mx1.example.net.
example.net.        43200 IN MX 10 mx2.example.net.
mx1.example.net.        43200 IN A 192.168.2.11
mx2.example.net.        43200 IN A 192.168.2.12
ns1.example.net.        43200 IN A 192.168.178.137
ns2.example.net.        86400 IN A 192.168.178.146
www.example.net.        43200 IN CNAME ns1.example.net.
```

Listing 23.20 Ausgabe der »Secondary-Zone«

Wie Sie Listing 23.20 entnehmen können, sind die Zonendateien nicht zu 100 % identisch. Das ist darauf zurückzuführen, dass die Datei nicht einfach kopiert wird, sondern dass die Zone ausgelesen wird (wie mit dem `dig`-Kommando) und neu geschrieben wird. Trotz der anderen Darstellung ist das Ergebnis bei Abfragen aber identisch – wie so oft führen viele Wege nach Rom. Wie Ihnen sicherlich aufgefallen ist, wird die Zonendatei auf dem Secondary-Server nicht im Text-Format gespeichert, sondern im sogenannten »raw«-Format. Dieses ist leider nicht direkt lesbar und muss vorab konvertiert werden. Der Befehl aus Listing 23.21 wandelt die Zone *example.com* aus der Datei *example.com* um und speichert das Resultat unter dem Dateinamen *example.com.text*:

```
$ named-compilezone -f raw -F text -o example.net.text example.net example.net
```

Listing 23.21 Konvertierung der »Secondary-Zone« mittels »named-compilezone«

Bei Änderungen »Serial« anpassen!

Beachten Sie, dass der Secondary-Server nur Änderungen von Zonendateien lädt, bei denen sich die Angabe *Serial* verändert bzw. sich der Wert erhöht hat. Denn nur über diesen Wert stellt der Secondary-Server fest, ob es Änderungen in der Zone gibt. Wenn Sie `serial` nicht erhöhen, werden Ihre Änderungen auch nicht auf den Secondary übertragen.

23.1.9 DNS-Server und IPv6

Natürlich lässt sich der BIND auch als DNS-Server für IPv6 konfigurieren. In Listing 23.22 sehen Sie die entsprechenden Einträge der IPv6-Zonen in einer *named.conf*-Datei:

```
zone "example.net" in {
        type primary;
        file "primary/db.example";
};

zone "123.168.192.in-addr.arpa" in {
        type primary;
        file "primary/db.192.168.123";
};

zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa" {
        type primary;
        file "primary/db.fd00";
};
```

Listing 23.22 »named.conf« für IPv6

Wie Sie in Listing 23.22 sehen, ist der Nameserver für die Forward-Zone `example.net`, die IPv4-Reverse-Zone für das Netz *192.168.123.0/24* und die IPv6-Reverse-Zone für das Netz `f:d::` zuständig. In Listing 23.23 sehen Sie die geänderte Forward-Zonendatei:

```
$TTL 1800       ; 30 minutes
example.net.    IN SOA  adminbuch.example.net. root.adminbuch.example.net. (
                                2021010502 ; Serial
                                600        ; Refresh (10 minutes)
                                200        ; Retry (3 minutes 20 seconds)
                                604800     ; Expire (1 week)
                                1800       ; negTTL (30 minutes)
                                )
                        IN      NS      ns1.example.net.
                        IN      NS      ns16.example.net.

ns1                     IN      A       192.168.123.102
ns16                    IN      AAAA    fd00::102
admin-repl              IN      A       192.168.123.103
admin-repl6             IN      AAAA    fd00::103
winclient               IN      A       192.168.123.121
winclient6              IN      AAAA    fd00::121
```

Listing 23.23 Forward-Zonendatei für IPv6

Für jeden Host gibt es hier jeweils einen IPv4- und einen IPv6-Eintrag. In Listing 23.24 sehen Sie den Aufbau der Reverse-Zonendatei:

```
$TTL 259200     ; 3 days
$ORIGIN 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa. IN SOA ns.example.net. root.example.net. (
                                2023010201 ; Serial
                                86400      ; Refresh (1 day)
                                1800       ; Retry (30 minutes)
                                172800     ; Expire (2 days)
                                1800       ; negTTL (30 minutes)
                                )
                        IN      NS      ns16.example.net.

102                     IN      PTR     ns16.example.net.
103                     IN      PTR     admin-repl.example.net.
121                     IN      PTR     winclient6.example.net.
```

Listing 23.24 Reverse-Zonendatei für IPv6

Da in der Beispieldatei mit der Variablen `$ORIGIN` gearbeitet wird, ist es sehr einfach die Host-Einträge zu erstellen, da nur der eigentliche Hostteil der Adresse eingegeben werden muss. Wollen Sie aber die Adresse in der ausführlichen Form verwenden, hilft Ihnen das Programm `ipv6calc` mit dem Schalter `-r`, das für jede Distribution verfügbar ist:

```
root@ubuntu:~# ipv6calc -r fd00::103
no input type specified, try autodetection...found type: ipv6addr
3.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.int.
```

Listing 23.25 Berechnung der Reverse-Zonendatei mit »ipv6calc«

23.2 Vertrauen schaffen mit DNSSEC

Mit den *Domain Name System Security Extensions (DNSSEC)* wird das offene und nicht gerade vor Sicherheit strotzende DNS abgesichert. Dafür wird eine Vertrauenskette (beginnend bei der Root-Zone ».«) bis zum abgefragten Record erstellt. Darüber kann geprüft werden, ob eine Antwort auch wirklich vom zuständigen Nameserver beantwortet und auf dem Transportweg nicht verändert wurde. In diesem Abschnitt wollen wir Ihnen zeigen, wie DNSSEC arbeitet und wie Sie Ihre Zonen signieren, um sicher Auskunft geben zu können.

23.2.1 Die Theorie: »Wie arbeitet DNSSEC?«

Damit DNS-Abfragen überprüft werden können, wurden mehrere neue Record-Typen eingeführt. Über diese kann sichergestellt werden, dass eine Anfrage wirklich korrekt ist und vom

zuständigen autoritativen Nameserver stammt. Ähnlich wie beim HTTPS werden hierfür Signaturen verwendet (allerdings wird beim DNS die Kommunikation nicht verschlüsselt). Abbildung 23.3 zeigt, was im Hintergrund geschieht, um eine Abfrage zu überprüfen.

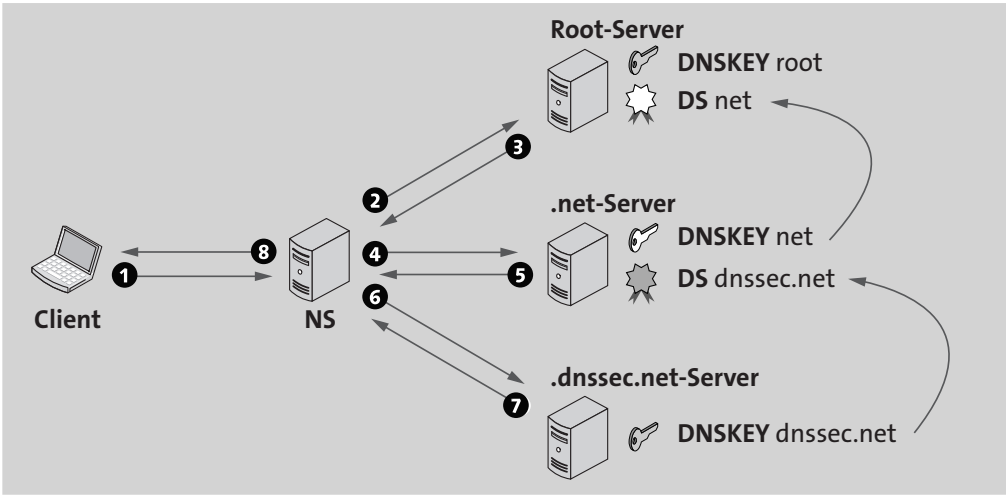


Abbildung 23.3 Ablauf der DNSSEC-Prüfung

Im ersten Schritt (1) fragt ein Client den lokalen Nameserver (NS) nach dem A-Record von *www.dnssec.net*. Der Nameserver beginnt nun mit der iterativen Namensauflösung. Dazu befragt er zunächst die Root-Zone (2). Da der Record mit DNSSEC abgesichert ist, liefert der Root-Server (3) bereits die dazugehörigen Records *DNSKEY* und *DS* mit. Nun fragt der Nameserver den für die Zone *.net* zuständigen Nameserver (4) und erhält erneut die Records *DNSKEY* und *DS* (5). Den Record *DNSKEY* kann der Nameserver nun mit dem vorherigen *DS*-Record abgleichen, um sicherzustellen, dass die Antwort valide ist. Dieser Prozess setzt sich nun fort (6 und 7), nämlich bei der Abfrage des autoritativen Nameservers von *dnssec.net*. Wenn alle Antworten valide waren, gibt der Nameserver die Antwort (8) an den Client weiter. Wie Sie sehen, ist das Vertrauensverhältnis hierarchisch; man spricht hier auch von einer *Chain of Trust*.

Neben den bereits gezeigten Records *DNSKEY* und *DS* gibt es noch weitere für DNSSEC notwendige Records. Diese schauen wir uns nun genauer an:

- **RRSIG (Resource Record Signature)**
  enthält die kryptografische Signatur.
- **DNSKEY (DNS Public Key)**
  enthält den öffentlichen Schlüssel.
- **DS (Delegation Signer)**
  enthält den Hash zu einem *DNSKEY*.

- **NSEC** und **NSEC3 (Next Secure)**
  wird zur eindeutigen Verneinung (engl. *denial-of-existence*) von einem Record verwendet – beim *NSEC3* kommen Hash-Werte statt Klartext zum Einsatz.
- **CDNSKEY** und **CDS**
  wird zur Aktualisierung des *DS*-Records einer Kindzone (engl. *child zone*) in der Elternzone verwendet.

Die Signierung von Records findet gebündelt statt. Diese Bündelung nennt man *RRsets (Record Sets)*. Dabei werden alle Records des gleichen Typs einer Zone zusammengefasst und signiert. Für die eigentliche Signierung kommen Schlüssel zum Einsatz (wie beim HTTPS). Beim DNS werden zwei Arten von Schlüsseln unterschieden:

- **ZSK (Zone-Signing Key)**
  wird zur Signierung von RRsets (Zonen) verwendet.
- **KSK (Key-Signing Key)**
  wird zur Signierung des ZSK verwendet.

Weshalb werden zwei Schlüssel verwendet? Dies hat den charmanten Vorteil, nicht unnötig viele Daten austauschen zu müssen. Der KSK ist eher statisch, da er nicht nur lokal, sondern auch beim übergeordneten Nameserver hinterlegt wird – ein Austausch ist relativ komplex. Der ZSK hingegen wird bei jeder Veränderung neu erstellt, der Austausch muss also zeitnah und einfach erfolgen. Daher wird der ZSK lokal verwendet und durch den KSK signiert. Durch die Vertrauensverkettung kann den ZSKs vertraut werden, wenn diese vom unveränderten KSK erzeugt wurden (was wiederum vom übergeordneten Nameserver abgefragt werden kann).

Dem aufmerksamen Leser wird nicht entgangen sein, dass die Vertrauenskette an einem Punkt endet: bei der Root-Zone. Weshalb sollten wir also der Root-Zone vertrauen? Hier kommt die *Root Signing Ceremony* ins Spiel. Bei dieser öffentlichen, überwachten und streng kontrollierten Zeremonie wird der Schlüssel erzeugt und in den Root-DNS-Servern eingespielt. Dieser Prozess wird alle paar Monate wiederholt und im Voraus geplant – hierzu finden Sie alle Daten unter *https://www.iana.org/dnssec/ceremonies*.

23.2.2 Anpassungen am Server

Damit Ihre Zone ebenfalls geprüft werden kann, müssen Sie ein paar Vorbereitungen treffen. In älteren Versionen des *bind* mussten Sie DNSSEC explizit aktivieren. Bei der Abfrage über den Server ist dies nicht mehr notwendig, da die Konfiguration bereits gesetzt ist (Konfiguration: `dnssec-validation auto;`).

Ob die Prüfung funktioniert, können Sie leicht mit der Testdomäne *www.dnssec-failed.org* überprüfen. In Listing 23.26 sehen Sie, wie sich das Ergebnis verändert, wenn die Option `dnssec-validation` auf dem Wert `auto` oder `off` steht.

```
### dnssec-validation auto
daniel@ubuntu:~$ dig @localhost www.dnssec-failed.org. A
[…]
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 26007
[…]
;www.dnssec-failed.org. IN A
[…]

### dnssec-validation off
daniel@ubuntu:~$ dig @localhost www.dnssec-failed.org. A
[…]
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26024
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
[…]
;; QUESTION SECTION:
;www.dnssec-failed.org. IN A

;; ANSWER SECTION:
www.dnssec-failed.org. 7200 IN A 68.87.109.242
www.dnssec-failed.org. 7200 IN A 69.252.193.191
[…]
```

Listing 23.26 Unterschiedliche Ergebnisse mit und ohne DNSSEC-Prüfung

Wie Sie sehen, erhalten Sie totz fehlerhafter DNSSEC-Konfiguration eine »gültige« Antwort, wenn `dnssec-validation` auf den Wert `off` eingestellt ist.

23.2.3 Schlüssel erzeugen

Anschließend müssen Sie die benötigten Schlüssel erzeugen. Dafür verwenden wir das Programm `dnssec-keygen`. Zur Erzeugung eines ZSK führen Sie den Befehl aus Listing 23.27 aus:

```
daniel@dns:/var/cache/bind/primary$ sudo dnssec-keygen -a ECDSAP256SHA256 example.net
Generating key pair.
Kexample.net.+008+33733
```

Listing 23.27 Einen ZSK erzeugen mit »dnssec-keygen«

Mit den Parametern `-a ECDSAP256SHA256` wurde der Verschlüsselungsalgorithmus definiert. Standardmäßig wird ein 1024 Bit langer Schlüssel erstellt. Der Befehl erzeugt zwei Dateien, das sogenannte *key pair* (dt. *Schlüsselpaar*). Da das Programm im Verzeichnis */var/cache/bind/primary* ausgeführt wurde, finden Sie dort nun die Dateien

Kexample.net.+008+33733.key (öffentlicher Schlüssel) und *Kexample.net.+008+33733.private* (privater Schlüssel). Die Benennung der Dateien folgt dieser Syntax:

```
K <ZONENAME> + <ALGORITHMUS> + <IDENTIFIER> . <TYP>
```

Listing 23.28 Syntax zur Benennung von Schlüsselpaaren bei DNSSEC

Alle Schlüssel fangen mit einem `K` an, gefolgt vom Zonennamen. Anschließend wird, durch ein Pluszeichen getrennt, der Algorithmus als vierstelliger Dezimalwert angegeben. Des Weiteren folgt, erneut durch ein Pluszeichen getrennt, eine Ziffer, die den Schlüssel identifiziert. Als Dateiendung wird stets der Typ verwendet, also `.key` für den öffentlichen Schlüssel und `.private` für den privaten Schlüssel. Anschließend können Sie mit dem gleichen Tool den KSK erzeugen, so wie in Listing 23.29 dargestellt:

```
daniel@dns:/var/cache/bind/primary$ sudo dnssec-keygen -a ECDSAP256SHA256 \
-f KSK example.net

Generating key pair.
Kexample.net.+008+57472
```

Listing 23.29 Einen KSK erzeugen mit »dnssec-keygen«

Die Befehle sind fast identisch: Für den KSK wurde lediglich der Parameter `-f KSK` hinzugefügt. Und da der KSK statisch ist, wurde er mit einer Länge von 2048 Bit erzeugt. Auch er erzeugt wieder den öffentlichen (`.key`) und den privaten (`.private`) Schlüssel.

23.2.4 Schlüssel der Zone hinzufügen und die Zone signieren

Die soeben erzeugten öffentlichen Schlüssel (`.key`) müssen Sie nun noch der bestehenden Zonendatei (*example.net*) hinzufügen. Dafür können Sie den Inhalt der Dateien einfach an die Zonendatei anhängen oder diese mit der Direktive `$INCLUDE` inkludieren. In Listing 23.30 haben wir die zweite Methode eingesetzt:

```
$ echo "\$INCLUDE Kexample.net.+008+33733.key" >> example.net
$ echo "\$INCLUDE Kexample.net.+008+57472.key" >> example.net
```

Listing 23.30 Schlüssel an die Zone anhängen

Damit ist die Zone *example.net* zum Signieren vorbereitet. Das Signieren wird mit dem Programm `dnssec-signzone` vorgenommen. Dieses erwartet einige Parameter und Werte. Sehen wir uns daher zunächst die Syntax des Programms an:

```
$ dnssec-signzone -A -3 <SALT> -N INCREMENT -o <ORIGIN> -t <ZONE FILE>
```

Listing 23.31 Syntax von »dnssec-signzone«

Sehen wir uns nun die einzelnen Parameter und deren Werte etwas genauer an:

- `-A`
  verhindert, dass bei der Erzeugung mit dem Verfahren *NSEC3* unsichere Delegationen ebenfalls signiert werden.
- `-3 <SALT>`
  Mit diesem Parameter wird der beim Verfahren *NSEC3* benötigte *SALT* (engl. für *Salz*) als Hash-Wert angegeben. Dieser dient dazu, kryptografisch bessere Ergebnisse zu erzielen, da er bei der Erzeugung »eingestreut« wird.
- `-N INCREMENT`
  Mit dieser Direktive weisen Sie `dnssec-signzone` an, die Angabe `Serial` der Zone beim Signieren direkt zu erhöhen, sodass Sie dies nicht nachträglich per Hand anpassen müssen.
- `-o <ORIGIN>`
  gibt den Origin der Zone an – wenn keiner angegeben wird, wird der Dateiname als Ursprung verwendet (wie sonst beim DNS auch).
- `-t`
  gibt am Ende des Verarbeitungsvorgangs eine Zusammenfassung aus.
- `<ZONE FILE>`
  gibt die zu signierende Zonendatei an.

Tipp: »SALT«

Um einen guten *Salt* zu erzeugen, können Sie einfach für den Platzhalter `<SALT>` aus Listing 23.32 das nachstehende Kommando verwenden:

```
$(head -c 1000 /dev/random | sha1sum | cut -b 1-16)
```

Dieses liest 1000 Zeilen aus dem Zufallsgenerator `random`, berechnet den Hash-Wert und gibt eine 16-stellige Hash-Zahl aus.

In Listing 23.32 sehen Sie den zum bisherigen Beispiel passenden Befehl:

```
$ sudo dnssec-signzone -A -3 <SALT> -N INCREMENT -o example.net -t example.net

Verifying the zone using the following algorithms:
- ECDSAP256SHA256
Zone fully signed:
Algorithm: ECDSAP256SHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                            ZSKs: 1 active, 0 stand-by, 0 revoked
example.net.signed
Signatures generated:                       17
Signatures retained:                         0
Signatures dropped:                          0
```

```
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Signing time in seconds:                 0.015
Signatures per second:                1062.566
Runtime in seconds:                      0.043
```

Listing 23.32 Zone signieren mit »dnssec-signzone«

Der Befehl aus Listing 23.32 erzeugt die Datei *dsset-example.net.* (die im weiteren Verlauf relevant wird) und die Datei *example.net.signed*. Öffnen Sie diese im Editor Ihrer Wahl, um sich das bisherige Arbeitsergebnis ansehen zu können. Die signierte Zonendatei ist immens gewachsen. Hatte die Ausgangsdatei noch 24 Zeilen, so verfügt die signierte Zone über 132 Zeilen. Ohne die Hilfsprogramme ist ein Umgang mit DNSSEC kaum zu bewältigen.

23.2.5 Signierte Zone aktivieren

Damit Ihr DNS-Server die signierte Zone auch verwendet, müssen Sie ihm dies in der Datei *named.conf.zones* auch mitteilen. Passen Sie die Zonendefinition so an (siehe Listing 23.33):

```
zone "example.net" {
        type primary;
        file "/var/cache/bind/primary/example.net.signed";
};
```

Listing 23.33 Aktivieren der signierten Zone in »named.conf.zones«

Zu guter Letzt müssen Sie die Änderungen noch aktivieren. Starten Sie dafür den *bind* einmal neu. Ab jetzt ist Ihr DNS-Server in der Lage, für die Zone *example.net* mit DNSSEC zu antworten.

23.2.6 Signierung prüfen

Prüfen können Sie dies mit dem Alleskönner `dig`:

```
daniel@dns:~$ dig DNSKEY example.net. @localhost +multiline

; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> DNSKEY example.net. @localhost +multiline
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59930
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 29f1c32edab29fbb0100000063b2dc4fec8c94037864959f (good)
```

```
;; QUESTION SECTION:
;example.net. IN DNSKEY

;; ANSWER SECTION:
example.net. 43200 IN DNSKEY 256 3 13 (
/OBID1pR/ZSFpFuT5uniC8mQ7Q3pjIO/izEan8PH6fUG
jD9fss8iL/i3+nqPekhrD6oqgjWT3dJOq5miG76MQg==
) ; ZSK; alg = ECDSAP256SHA256 ; key id = 46574
example.net. 43200 IN DNSKEY 257 3 13 (
uxNfN7iDBzqcBWOu1y+mA/s3gqEimlFmNekQuEheZ9/3
G/4ezNwW3lwvwSK62+TBOa57GDfuiccoBjJOZ9BgHA==
) ; KSK; alg = ECDSAP256SHA256 ; key id = 29274

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Mon Jan 02 13:19:51 UTC 2023
;; MSG SIZE  rcvd: 228
```

Listing 23.34 Prüfung von DNSSEC mit »dig«

Wie Sie in Listing 23.34 sehen, gibt der lokale DNS-Server den angefragten DNSSEC-Record korrekt aus. Dies ist aber nur die halbe Miete. Um zu prüfen, ob der DNS-Server wirklich DNSSEC spricht, müssen Sie das dig-Kommando so ausführen, wie in Listing 23.35 dargestellt:

```
daniel@dns:~$ dig A example.net. @localhost +noadditional +dnssec +multiline
; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> A example.net. @localhost +noadditional \
  +dnssec +multiline
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56649
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 1ffe34d5118c115901000000063b2dc1924b0b38460c4da61 (good)
;; QUESTION SECTION:
;example.net. IN A

;; AUTHORITY SECTION:
example.net. 3600 IN SOA ns1.example.net. admin.example.net. (
2023010208 ; serial
43200      ; refresh (12 hours)
3600       ; retry (1 hour)
864000     ; expire (1 week 3 days)
```

```
3600       ; minimum (1 hour)
)
example.net. 3600 IN RRSIG SOA 13 2 43200 (
20230201122731 20230102122731 46574 example.net.
Y5owL[…]i45fQ== )
VO39KG3CR3EQPOF3K37RLHEV3O8BIEMH.example.net. 3600 IN NSEC3 1 1 10 3160OD06D564CC5D (
6I3RN[…]2E9234KB NS SOA MX RRSIG DNSKEY NSEC3PARAM )
VO39KG3CR3EQPOF3K37RLHEV3O8BIEMH.example.net. 3600 IN RRSIG NSEC3 13 3 3600 (
20230201122731 20230102122731 46574 example.net.
PgNTO[…]plNrw== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Mon Jan 02 13:28:57 UTC 2023
;; MSG SIZE  rcvd: 416
```

Listing 23.35 Prüfung der Signierung mit »dig«

23.2.7 Die Signierung veröffentlichen

Fertig? Noch nicht ganz. Damit auch die große weite Welt Ihren signierten Zonen vertraut, müssen Sie die *DS*-Records Ihrem Domänen-Registrar melden. Diese Records stehen in der ebenfalls von `dnssec-signzone` erzeugten Datei *dsset-example.net.* (siehe Listing 23.36):

```
daniel@saturn:/var/cache/bind/primary$ cat dsset-example.net.
example.net. IN DS 29274 13 2 F415[…]47DB E9OCEEEC
```

Listing 23.36 Ausgabe der Datei »dsset-example.net.«

Wie Sie sehen, war `dnssec-signzone` so freundlich und hat den benötigten Record direkt in DNS-Schreibweise aufbereitet und abgelegt. Der Record-Typ DS verfügt über mehrere Werte, nämlich über den Identifier, den verwendeten Algorithmus, den *Digest*-Typ (jeweils als Dezimalwert) und den *Digest* selbst (ähnlich wie die Dateibenennung der Schlüsselpaare).

Nachdem Sie den DS-Record in Ihrem Domänen-Registrar eingespielt haben und dieser Ihre DS-Records aktiviert hat, haben Sie es geschafft. Mithilfe von `dig` können Sie dies selbstverständlich kontrollieren. Führen Sie dafür einfach das Kommando aus Listing 23.37 aus:

```
daniel@saturn:~$ dig +trace +noadditional DS example.net. @8.8.8.8 | grep DS
```

Listing 23.37 Prüfung der DS-Records mit »dig«

Der Befehl fragt den Google-DNS (`8.8.8.8`) nach dem Record-Typ DS und limitiert die Ausgabe auf den Inhalt der DS-Records. Beachten Sie dabei, dass Sie die Zone entsprechend anpassen.

23.2.8 Weniger anstrengend: Mehr Automatismus!

Natürlich geht's auch etwas bequemer und Sie müssen nicht bei jeder Konfigurationsanpassung die Zonen neu signieren. Dafür wurden im *bind* zwei Optionen geschaffen:

- `auto-dnssec maintain`
  Mit dieser Option übernimmt der *bind* die DNSSEC-Signierung selbstständig. Wenn zum Beispiel eine Signatur ausläuft oder ein neuer Zonenschlüssel vorhanden ist, wird das Signieren ausgelöst.
- `inline-signing yes`
  Diese Option weist *bind* zusätzlich an, bei einem Zonentransfer die geänderten Zonendaten zu signieren. Allerdings werden damit nur neue unsignierte Einträge verarbeitet. Ältere Einträge werden, wenn sie ablaufen, durch das `auto-dnssec maintain` bei Bedarf neu signiert.

Diese Optionen müssen pro Zone definiert werden. Für unsere bisherige Beispielzone könnte die Konfiguration wie folgt aussehen:

```
zone "example.net" {
        type primary;
        file "primary/example.net";
        auto-dnssec maintain;
        inline-signing yes;
};
```

Listing 23.38 Zonen-Konfiguration mit Automatismen

Rechte beachten!
Bitte denken Sie daran, dass die Dateien für den Dienst lesbar sein müssen. Ansonsten kann der Automatismus leider nichts für Sie tun. Achten Sie daher stets darauf, dass die Schlüssel auch dem Benutzer gehören, unter dem der *bind* läuft.

Nach einem Neustart des Dienstes werden alle Signaturen für Sie angelegt. Die signierte Zonendatei können Sie leider nicht direkt öffnen, um das Ergebnis zu kontrollieren, aber selbstverständlich gibt es dafür ein Tool. Mithilfe von `named-checkzone` können Sie das Format in eine für Menschen lesbare Form umwandeln lassen, so wie in Listing 23.39 dargestellt:

```
$ named-checkzone -D -f raw example.net example.net.signed
```

Listing 23.39 Zonen-Konfiguration mit Automatismen

Dass der Automatismus wirklich funktioniert, können Sie an den Logmeldungen sehen. Nach einem Neustart des Dienstes sollten Sie im Log Meldungen finden wie die in Listing 23.40:

```
named[703]: zone example.net/IN (signed): sending notifies (serial 2023010213)
named[703]: zone example.net/IN (signed): reconfiguring zone keys
named[703]: zone example.net/IN (signed): next key event: 03-Jan-2023 13:03:06.544
```

Listing 23.40 Systemmeldung des »bind« mit aktiviertem Automatismus

23.2.9 Fazit

Das DNSSEC bläht die Zonendateien immens auf. Wenn Sie kein konkretes Ziel verfolgen, zum Beispiel die Absicherung Ihrer Mailserver mit DNSSEC, bringt Ihnen DNSSEC kaum Vorteile. Aufgrund seiner Komplexität empfehlen wir Ihnen dringend, über eine Managementlösung nachzudenken. Rein über die Konsole ist DNSSEC relativ zäh zu bewältigen, zumindest wenn mehr als eine Zone verwaltet werden soll.

23.3 Client-Anfragen absichern mit »DNS over HTTPS (DoH)«

Bisher haben wir lediglich mittels DNSSEC dafür gesorgt, dass Dritte nachvollziehen können, dass die Antwort auf eine Anfrage wirklich vom Eigentümer einer Domäne stammt. Allerdings können auch rekursive DNS-Abfragen mittlerweile verschlüsselt werden. Beim *DNS over HTTPS (DoH)* werden, wie der Name bereits verrät, die DNS-Anfragen von Clients durch HTTPS geleitet. Das übergeordnete Ziel dabei ist, die Privatsphäre und Sicherheit zu erhöhen. Wie Sie Ihrem *bind* das DoH beibringen, zeigen wir Ihnen in diesem Abschnitt.

23.3.1 Installation

Das DoH wird von *bind* ab der Version 9.17 unterstützt. Lediglich Ubuntu 22.04 stellt Ihnen bereits einen aktuelleren *bind* in den Standardpaketquellen zur Verfügung. Bei den übrigen Distributionen müssen Sie Hand anlegen, was wir Ihnen nun genau zeigen.

Debian

Aktivieren Sie zunächst die *backports*-Repositories in der Datei */etc/apt/sources.list*:

```
deb http://deb.debian.org/debian bullseye-backports main contrib non-free
deb-src http://deb.debian.org/debian bullseye-backports main contrib non-free
```

Listing 23.41 »Backports«-Repositories aktivieren in »/etc/apt/sources.list«

Anschließend müssen Sie die Paketquellen neu laden und können dann die Version 9.18 installieren (siehe Listing 23.42):

```
$ apt -t bullseye-backports upgrade
$ apt install -f bind9 bind9utils bind9-dnsutils -t bullseye-backports
```

Listing 23.42 Repositories aktualisieren und den »bind« installieren

Wichtig dabei ist die Angabe des Repositorys mit dem Schalter `-t`, da ansonsten die regulären Repositories verwendet werden.

openSUSE Leap

Auch bei openSUSE müssen Sie ein alternatives Repository hinzufügen, die Paketquellen neu laden und können dann die neuere Version von *bind* installieren. Setzen Sie dafür die Kommandos aus Listing 23.43 ab:

```
$ zypper addrepo https://download.opensuse.org/repositories/network/15.4/network.repo
$ zypper refresh
$ zypper install bind-9.18.1-lp154.394.2.x86_64 bind-utils-9.18.1-lp154.394.2.x86_64
```

Listing 23.43 Repository ergänzen, aktualisieren und den »bind« installieren

Wenn Sie die Installation ohne die Versionsnummer ausführen, dann erhalten Sie einen Hinweis, dass eine neuere Version vorhanden ist – sogar inklusive des notwendigen Kommandos, um diese zu installieren.

CentOS

Für CentOS existiert leider kein Paket und auch auf den üblichen Plattformen (wie *ELRepo* oder *rpmfind.net*) wurde zur Drucklegung keine aktuellere Version als 9.16 angeboten. Hier müssen Sie den *bind* leider per Hand kompilieren, um das DoH nutzen zu können.

23.3.2 Vorbereitungen

Bevor wir uns um die eigentliche Konfiguration kümmern, müssen wir zunächst ein paar Vorbereitungen treffen. In diesem Fall wortwörtliche ein Paar, also zwei:

- Verzeichnisse erstellen: */etc/<bind|named>/keys*
- Zertifkat erstellen

Den ersten Punkt können Sie schnell erledigen, den zweiten theoretisch auch. Da die gängigen Tools die Zertifikatskette nicht prüfen, sondern das HTTPS lediglich zur Verschlüsselung einsetzen, sind offizielle Zertifikate nicht zwingend erforderlich. Um aber potenzielle Fehler zu vermeiden und generell einen guten Stil zu wahren, sollten Sie dennoch welche einsetzen. Wir empfehlen Ihnen an dieser stelle *Let's Encrypt*-Zertifikate – wie Sie solche beziehen können zeigen wir Ihnen ausführlich in Abschnitt 31.4, »Einmal mit allem und kostenlos bitte: Let's Encrypt«.

Natürlich können Sie für DoH auch Zertifikate Ihrer eigenen CA verwenden oder (wie wir im folgenden Beispiel) einfach selbst signierte verwenden – wir setzen dafür auf zwei »OpenSSL-One-Liner«, die sowohl die CA erzeugen als auch die benötigten Server-Zertifikate und -Schlüssel:

```
# CA erstellen
$ openssl req -x509 -sha512 -nodes -extensions v3_ca -newkey rsa:4096 \
-keyout MyRootCA.key.pem -days 7320 -out MyRootCA.cert.pem \
-subj "/C=DE/ST=NRW/L=Moers/O=Adminbuch-Ltd/CN=MyRootCA"

# Zertifikat erstellen
$ openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \
-subj "/C=DE/ST=NRW/L=Moers/O=Adminbuch-Ltd/CN=ns1.example.net" \
-reqexts SAN -extensions SAN -config <(cat /etc/ssl/openssl.cnf \
<(printf '[SAN]\nsubjectAltName=DNS:ns1.example.net')) \
-CA MyRootCA.cert.pem -CAkey MyRootCA.key.pem \
-keyout ns1.example.net.key.pem -out ns1.example.net.cert.pem
```

Listing 23.44 Selbst signierte CA und ein Zertifikat für DoH erstellen

Die beiden Kommandos erzeugen in dem Verzeichnis, in dem sie ausgeführt werden, die Dateien *MyRootCA.key.pem* (CA-Schlüssel), *MyRootCA.cert.pem* (CA-Zertifikat), *ns1.example.net.key.pem* (Serverschlüssel) und *ns1.example.net.cert.pem* (Serverzertifikat).

Bitte passen Sie bei dem Kommando aus Listing 23.44 unbedingt die Parameter `subj` an Ihre Gegebenheiten an, und passen Sie im zweiten Kommando auch das via `printf` inkludierte `[SAN]` an Ihren Servernamen an – selbstverständlich sollten Sie dann auch die Dateinamen der Parameter `keyout` und `keyout` anpassen. Kopieren Sie anschließend das Serverzertifikat und den Schlüssel nach */etc/bind/keys*.

Mehr zum Thema Zertifikate?

Alles zum Thema Zertifikate finden Sie in Kapitel 31, »Verschlüsselung und Zertifikate«.

23.3.3 Konfiguration

Die eigentliche Konfiguration besteht nur aus ein paar wenigen Zeilen, die Sie in der *named.conf* ergänzen müssen. Listing 23.45 zeigt die entsprechenden Zeilen:

```
// # Zertifikate & Schlüssel
tls local-tls {
    key-file "/etc/bind/keys/ns1.example.net.key.pem";
    cert-file "/etc/bind/keys/ns1.example.net.cert.pem";
};

// # HTTP-Endpunkt
http local-http-server {
        endpoints { "/dns-query";  };
};
```

```
options {
    […]
    // # HTTPS-Ports
    https-port 443;

    // # IPv4 und IPv6
    listen-on port 443 tls local-tls http local-http-server { any; };
    listen-on-v6 port 443 tls local-tls http local-http-server { any; };
};
```

Listing 23.45 DoH in »named.conf« aktivieren

Bitte achten Sie darauf, dass der `options`-Block ergänzt werden muss. Nach dem obligatorischen Neustart des Dienstes können Sie Ihren DNS-Server mittels DoH abfragen.

23.3.4 Funktionstest

Um die Behauptung zu überprüfen, können Sie einfach das übliche Kommando `dig` verwenden. Mit dem Parameter `+https` stellt `dig` die DNS-Anfragen via HTTPS, so wie in Listing 23.46 dargestellt, an Ihren DNS-Server:

```
daniel@client:~# dig +https @ns1.example.net rheinwerk-verlag.de

; <<>> DiG 9.18.8-1~bpo11+1-Debian <<>> +https @ns1.example.net rheinwerk-verlag.de
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18767
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 1a68e4ba0b45a04b0100000063b4271caad43559d88e5354 (good)
;; QUESTION SECTION:
;rheinwerk-verlag.de. IN A

;; ANSWER SECTION:
rheinwerk-verlag.de. 85428 IN A 46.235.24.168

;; Query time: 79 msec
;; SERVER: 192.168.178.137#443(ns1.example.net) (HTTPS)
;; WHEN: Mon Jan 02 14:30:04 UTC 2023
;; MSG SIZE  rcvd: 96
```

Listing 23.46 DNS-Abfrage via HTTPS mit »dig«

Mindestens Version 9.17 oder häher benötigt!

Bitte beachten Sie, dass die Sie mindestens die Version *9.17* oder höher von `dig` benötigen. Zur Drucklegung war diese lediglich in Ubuntu 22.04 nativ enthalten.

23.3.5 Client-Konfiguration

Um Linux-Clients in Ihrem Netzwerk beizubringen, Ihren DNS-Server via DoH abzufragen, genügt es, das Tool *dnscrypt-proxy* aus den Paketquellen zu installieren. Anschließend müssen Sie nur noch das zuvor erstellte CA-Zertifikat unter */etc/ssl/certs/* ablegen und die Konfigurationsdatei */etc/dnscrypt-proxy/dnscrypt-proxy.toml* anpassen. Diese muss mindestens die Zeilen aus Listing 23.47 enthalten, um Anfragen an den DNS-Server *ns1.example.net* zu stellen zu können:

```
# Empty listen_addresses to use systemd socket activation
listen_addresses = []
server_names = ['ns1.example.net']

#[sources]
#  [sources.'public-resolvers']
#  url = 'https://download.dnscrypt.info/resolvers-list/v2/public-resolvers.md'
#  cache_file = '/var/cache/dnscrypt-proxy/public-resolvers.md'
#  minisign_key = 'RWQf6LRCGA9i53mlYecO4IzT51TGPpvWucNSCh1CBMOQTaLn73Y7GFO3'
#  refresh_delay = 72
#  prefix = ''

[static]
  [static.'ns1.example.net']
  stamp = 'sdns://AgcAAAAAAAAADz[…]XVlcnk'
```

Listing 23.47 Konfiguration in »dnscrypt-proxy.toml«

Mit den Parameter `server_names` können der (oder die) lokalen Nameserver angegeben werden. Diese müssen wiederum über einen eigenen Eintrag unterhalb von `static`-Block verfügen. Wichtig ist, dass Sie den Block `sources` auskommentieren, da ansonsten eine Liste öffentlich verfügbarer Resolver aus dem Internet geladen und eingerichtet wird!

Den Parameter aus der letzten Zeile müssen Sie nun noch generieren. Dazu können Sie einfach das Online-Tool *https://dnscrypt.info/stamps/* verwenden – das Tool stammt im Übrigen aus der Feder der *dnscrypt-proxy*-Autoren. Dort müssen Sie lediglich unter PROTOCOL das DNS-OVER-HTTPS (DOH) auswählen und die Felder mit Ihren Daten ausfüllen – für unser Beispiel bis hierher sieht das so aus wie in Abbildung 23.4 dargestellt.

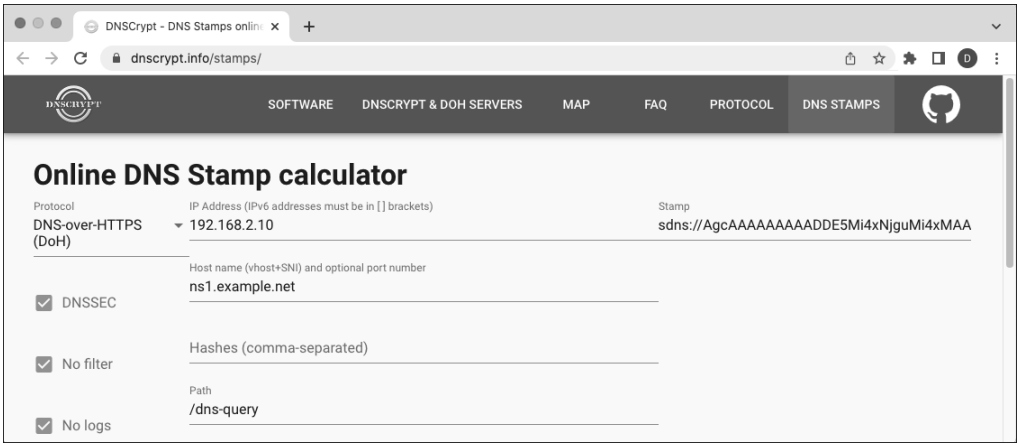


Abbildung 23.4 Online-Generierung des »Stamp«

Nach einem Neustart des Diensts mittels `systemctl reload dnscrypt-proxy` stellt dieser nun unter der Localhost-Adresse `127.0.2.1` den DNS-Proxy bereit. Stellen Sie DNS-Anfragen an diesen werden diese via DoH an Ihren DNS-Server weitergeleitet. Dieses Tool können Sie auf jedem Client separat betreiben oder zentral bereitstellen – dafür genügt es, beim Parameter `listen_addresses` eine IP-Adresse anzugeben.

Kapitel 24
OpenSSH

In diesem Kapitel lernen Sie den fortgeschrittenen Umgang mit dem OpenSSH-Client sowie dem OpenSSH-Server.

Im Jahr 1969 wurde *telnet* (TELeNETwork) entwickelt. Es eröffnete die Möglichkeit, auf der Kommandozeile eines entfernten Rechners zu arbeiten, als säße man direkt davor. Sie können sich leicht vorstellen, dass *telnet* sich in Windeseile verbreitete und äußerst beliebt war, weil es das Ende der »Turnschuhadministration« einläutete. Sicherheitsbedenken spielten noch keine Rolle, da die Anzahl der vernetzten Rechner zu Anfang der 70er-Jahre des vergangenen Jahrhunderts noch ausgesprochen übersichtlich war. So störte es zunächst niemanden, dass die Anmeldung ebenso wie der Inhalt der *telnet*-Session unverschlüsselt übertragen wurde. Das änderte sich, als *telnet* in den 90ern immer öfter Opfer von Sniffer-Angriffen wurde. 1995 entwickelte der finnische Student Tatu Ylönen das SSH-Protokoll, und 1999 debütierte ein SSH-Fork (Code-Ableger) unter dem Namen *OpenSSH* als Bestandteil der OpenBSD-Distribution 2.6.

OpenSSH wurde auf eine Vielzahl anderer Betriebssysteme portiert. Wenn Sie sich auf einem Linux-System die Versionsnummer von OpenSSH anzeigen lassen, finden Sie darin ein »p«:

```
$ ssh -V
OpenSSH_8.4p1 Debian-5+deb11u1, OpenSSL 1.1.1n  15 Mar 2022
```

Listing 24.1 OpenSSH-Versionsnummer anzeigen lassen

An diesem Buchstaben erkennen Sie, dass es sich um eine portierte Version handelt. Auch die Versionsnummer von OpenSSL, dessen Bibliotheken die SSH-Tools benötigen, wird angezeigt.

24.1 Die SSH-Familie

Zur SSH-Familie gehören folgende Programme:

- `ssh`: ein Secure-Shell-Client als Ersatz für *telnet*, *rlogin* und *rsh*
- `scp`: Secure Copy, ersetzt *rcp*.
- `sftp`: Secure FTP

Vorwort

Willkommen zur siebten Auflage von »Linux-Server. Das umfassende Handbuch«! Auch mehr als 10 Jahre nach der Erstauflage finden sich noch neue Themen oder große Änderungen, die eine neue Auflage füllen können. Das gilt nicht nur für die Cloud! Auch der eigene Server, den Sie in Ihrem Rechenzentrum oder Serverraum pflegen, ist wichtiger denn je. Innovationen und Änderungen in diesem Bereich werden uns noch lange begleiten.

Wieder standen wir vor der Wahl, welche Distributionen wir für diese Ausgabe bearbeiten sollten. Debian und Ubuntu waren von Anfang an fest eingeplant. Auch Suse war fest gesetzt. Bei Redhat haben wir uns dafür entschieden, wieder auf die Stream-Variante zu setzen, die die Zukunft der neuen Ableger AlmaLinux oder Rocky Linux nicht ganz klar ist. Wir werden diese Situation aber weiterhin beobachten, denn es ist davon auszugehen, dass sich in diesem Bereich in den nächsten Jahren viel tun wird.

In dieser Aktualisierung gab es wieder viele kleine und große Änderungen. Am wichtigsten ist die neue Version von OpenLDAP. 14 Jahre nachdem Version 2.4 erschienen ist, wurde mit Version 2.6 eine vollständige Überarbeitung veröffentlicht, auf die wir ausführlich in Kapitel 17 eingehen. Beim Thema Datenbanken haben wir uns schon in der letzten Auflage dafür entschieden, MySQL durch MariaDB zu ersetzen, da MySQL teilweise gar nicht mehr in den Repositories der Distributionen vorhanden ist. Wie wir heute sagen können: Eine gute Entscheidung. Bei den Distributionen hat sich – wie bereits angeführt – auch etwas geändert. Bei Debian ist die Version Debian Bullseye neu hinzugekommen, bei Suse sind wir auf Suse Leap 15.4 umgestiegen. Von Ubuntu ist eine neue LTS-Version auf dem Markt, die Version 22.04. Bei CentOS werden wir weiterhin die Version Stream nutzen.

Fast alle Kapitel wurden von uns komplett überarbeitet, teilweise neu geschrieben, um möglichst aktuell zu bleiben. Hier war es uns wieder besonders wichtig, die neuen Funktionen von Programmen aufzunehmen und eventuell ältere Optionen und Vorgehensweisen herauszunehmen. Zudem haben wir auch wieder viele Anregungen und Kommentare erhalten, die uns dazu inspiriert haben, eine neue Auflage zu schreiben. Außerdem schreiben ein paar der Autoren noch Bücher zu speziellen Themen und haben daraus die eine oder andere Idee übernommen. Sicherlich hätten wir an manchen Stellen noch tiefer einsteigen können, aber der Umfang des Buches kommt so langsam an die Grenzen des technisch Machbaren. Zudem soll das Buch einen Überblick über möglichst viele verschiedene Dienste geben; es kann und will kein Wälzer sein, der alle Dienste bis zum letzten Bit beschreibt.

Wie schon bei der sechsten Auflage wollen wir Ihnen mit diesem Buch eine Anleitung bieten, wie Sie die verschiedensten Dienste, die Ihnen ein Linux-System bereitstellen kann, schnell und einfach konfigurieren. Ohne große Umwege geht es über die Konfiguration hin zu einem funktionsfähigen Dienst, den Sie dann an Ihre eigenen Bedürfnisse anpassen können.

Zudem haben wir alle großen Neuerungen für Sie so zusammengefasst, dass Sie auch neue Techniken nachlesen und umsetzen können.

Wir wollen Ihnen ein Nachschlagewerk an die Hand geben, das Sie mit vielen verschiedenen Techniken und Diensten vertraut macht. In den einzelnen Kapiteln gehen wir auch immer wieder auf Besonderheiten der verschiedenen Distributionen ein. Gerade durch die Vielfalt der Dienste und Techniken können Sie dieses Buch wie ein Schweizer Taschenmesser nutzen: immer griffbereit und immer das richtige Werkzeug dabei. Mit jeder Auflage bekommt dieses Schweizer Taschenmesser ein paar Werkzeuge mehr, und die bestehenden Werkzeuge wurden an vielen Stellen noch schärfer und präziser gemacht. Auch in dieser Auflage haben wir viele Beispiele aus unserer täglichen Arbeit aufgenommen, denn das, was wir in den verschiedenen Unternehmen erleben und einrichten, ist immer eine gute Grundlage, um Ihnen zu helfen, möglichst schnell ans Ziel zu gelangen.

Für wen haben wir das Buch geschrieben?

Dieses Buch richtet sich an alle Linux-Systemadministratoren, die immer wieder vor der Aufgabe stehen, neue Dienste in ihrem Netzwerk zu etablieren, und die am Anfang einen möglichst schnellen und kompakten Einstieg in das Thema wünschen. Grundlegende Linux-Kenntnisse, wie sie zum Beispiel in LPIC-1 verlangt werden, sollten auf jeden Fall schon vorhanden sein, damit Sie die einzelnen Dienste erfolgreich in das eigene Netz integrieren können.

Wie können Sie mit diesem Buch arbeiten?

Wir haben das Buch so geschrieben, dass Sie gezielt mit den Beispielen aus den einzelnen Kapiteln einen neuen Dienst konfigurieren und testen können. An vielen Stellen verweisen wir aber auch auf andere Dienste, die hier im Buch beschrieben sind, um Ihnen die Möglichkeit zu geben, auch komplexere Aufgaben zu realisieren.

Was dieses Buch nicht ist

Dieses Buch ist kein Lehrbuch, um den Umgang mit Linux von Grund auf zu verstehen, dafür gibt es viele andere Bücher auf dem Markt. Auch war das Buch von Anfang an nicht dazu gedacht, einen oder mehrere einzelne Dienste bis ins Letzte zu konfigurieren. Denken Sie an Ihr Schweizer Taschenmesser: Es kann Ihnen bei vielen Aufgaben helfen, aber für spezielle Aufgaben gibt es spezielle Werkzeuge. Das Gleiche gilt für unser Buch.

Viele Aufgaben können Sie mithilfe unseres Buches erledigen, aber wenn es dann sehr speziell wird, brauchen Sie ein Buch, das genau dieses eine Thema bis in kleinste Detail beschreibt.

Über 10 Jahre Linux-Server Buch

Am 28. Januar 2011 wurde die erste Auflage unseres Buches Linux-Server veröffentlicht.

Die Idee zum Buch erblickte im April 2008 das Licht der Welt, bis zum Erscheinen der ersten Auflage vergingen also fast drei Jahre. Nach dem Ausstieg des Ideengebers Marcus Fischer fand sich ein erstes Autorenteam, das sich im Januar 2009 mit allen Beteiligten in einem Café in Stuttgart traf, um die weitere Vorgehensweise zu besprechen. Leider hat sich dieses erste Team bis auf Stefan Kania und Dirk Deimeke wieder zerschlagen, neue Mitautoren wurden gesucht und mit Charly Kühnast, Daniel van Soest und Stefan Semmelroggen auch gefunden. Im November fand ein Treffen des neuen Teams auf der allerersten OpenRheinRuhr-Konferenz im Bottroper Saalbau statt, ein Mitglied des Teams konnte damals leider nur telefonisch teilnehmen. Nachdem die Eckbedingungen für das Buch klar waren, fand die weitere Kommunikation überwiegend auf der eigens dafür eingerichteten Mailingliste statt. In unregelmäßigen Abständen gab es weitere Audio- und Videokonferenzen, um strittige Punkte und Inhalte miteinander abzustimmen.

Wir durften gemeinsam erleben, wie schwierig es ist, in einem verteilten Team zu arbeiten, bei dem sich die Teilnehmer kaum persönlich kennen. Alle von uns haben verschiedene Hintergründe und Lebensläufe, aber wir sind alle Experten für die Bereiche, die wir im Buch übernommen haben. Viel wurde darüber diskutiert, was in das Buch kommt und wie umfangreich die einzelnen Kapitel werden sollen. Jeder hatte seine Ideen, und wie das so ist: Wenn viele Köche an einem Menü arbeiten, ist für jeden sein Teil der wichtigste. Nachdem wir uns auf die Inhalte geeinigt hatten, konnten wir endlich loslegen. Für die meisten von uns war das Schreiben eine komplett neue Erfahrung. Schulungsunterlagen hatte der eine oder andere schon erstellt, aber ein Buch! Das ist noch mal eine ganz anderer Herausforderung.

Für einige von uns war der Umgang mit LaTeX neu, aber selbst der größte Skeptiker hat es zum Schluss lieben gelernt. Technisch haben wir anfänglich alles mit Subversion als Versionskontrollsystem versioniert und sind im Laufe der Zeit auf Git umgestiegen.

Es hat geklappt, wie Sie sehen. Über die Jahre wurden wir immer sicherer, und der Inhalt wurde von Auflage zu Auflage besser. Aber wenn man dann so dabei ist, bekommt man immer neue Ideen, was noch alles wichtig sein kann. Jetzt sind wir an die Grenzen des technisch Machbaren gelangt, was den Druck angeht, nd mehr als das, was Sie jetzt in der Hand halten, geht nicht. Obwohl wir noch einige Ideen hätten ...

Über die verschiedenen Auflagen hinweg haben wir den einen oder anderen Autor verloren, aber auch immer wieder neue gute Autoren dazu gewonnen. Bei über 10. Jahren sollte man, denken wir, alle Namen nennen, die geholfen haben das Buch zu gestalten. Hier die Liste aller Autoren:

- Dirk Deimeke
- Stefan Kania
- Charly Kühnast
- Stefan Semmelroggen

- Daniel van Soest
- Peer Heinlein
- Axel Miesen

Auch der verantwortliche Lektor aus dem Verlag hat während der Zeit einige Male gewechselt. Ohne einen guten Lektor kann so ein Projekt wie dieses Buch nicht über so lange Zeit erfolgreich sein. Aus diesem Grund möchten wir auch die Lektoren auflisten, die uns in dieser Zeit mit Rat und Tat zur Seite gestanden haben:

- Jan Watermann
- Sebastian Kestel
- Anne Scheibe
- Christoph Meister

Wir hatten im Vorfeld nicht gedacht, wie wichtig sie sind, aber wir möchten auch ganz besonders den beiden Korrektorinnen danken, die uns zwischenzeitlich in den Wahnsinn getrieben haben, weil sie mit großer Sorgfalt unsere Fehler identifiziert und angemahnt haben. Sie tragen mit ihrer großartigen Arbeit zum Erfolg dieses Buchs bei:

- Friederike Daenecke
- Angelika Glock

Eine weitere Person aus dem Verlag soll hier auch nicht unerwähnt bleiben. Wir möchten unserem Hersteller danken, der bei allen Auflagen für uns als Ansprechpartner für technische Probleme bereitstand und uns wichtige Tipps zum Satz und zu der Bearbeitung der Bilder gegeben hat:

- Norbert Englert

Nur alle zusammen konnten wir das Buch über die Jahre immer wieder aktualisieren.

Ein ganz großer Dank geht natürlich auch an Sie, denn ohne die Leser kann das beste Buch nicht mehr als 10 Jahre bestehen. Für uns war auch das Feedback der Leserinnen und Leser immer wieder wichtig, denn manchmal bekommt man durch ein Feedback eine ganz andere Sichtweise auf bestimmte Dinge und kann diese Erkenntnis in einer neuen Auflage einfließen lassen.

Haben Sie viel Spaß mit dem Buch und möge es Ihnen bei dem einen oder anderen Projekt gute Hilfe leiste.

Vorwort von Dirk Deimeke

Im April 2008 kam Marcus Fischer, der Autor zahlreicher Ubuntu-Bücher beim Rheinwerk Verlag, mit der Idee auf mich zu, ein Linux-Adminbuch zu schreiben. Da es kein deutsches Werk gibt, das die Lücke zwischen Einsteigerbüchern und Fachbüchern schließt, die sich ei-

nem einzelnen Thema widmen, war und bin ich immer noch Feuer und Flamme. In den folgenden fünf Monaten arbeiteten wir zusammen mit Jan Watermann, dem damaligen Lektor, an dem Konzept des Buches. Uns war zu jedem Zeitpunkt klar, dass es ein Buch werden sollte, das viel Bezug zur Praxis hat und einige Probleme behandelt, denen Linux-Systemadministratoren täglich begegnen. Das schreibt sich so leicht in ein oder zwei Sätzen, aber es war ein längerer Dialog, da jeder eine etwas andere Vorstellung davon hatte, wie das Buch aussehen sollte. Der Begriff »Kochbuch« durfte aufgrund von Markenrechten nicht verwendet werden, traf aber das, was wir machen wollten, am besten.

Nachdem Marcus aufgrund seiner Dissertation keine Zeit mehr hatte, an dem Buch zu arbeiten, ging die Suche nach Autoren los, und Mitstreiter wurden gefunden. Aufgrund interner Schwierigkeiten trennte sich die initiale Gruppe jedoch wieder, und es drohte das Aus. In einem zweiten Anlauf fanden sich dann die Autoren zusammen, die die erste Auflage des Buchs geschrieben haben. Stefan Kania ist außer mir aus der ersten Gruppe dabeigeblieben. Zu uns gestoßen sind für die zweite Auflage Stefan Semmelroggen, Daniel van Soest und Charly Kühnast. Mit der dritten Auflage hat sich das Team leider wieder geändert: Stefan Semmelroggen verließ das Team, und Peer Heinlein stieß dazu. In der vierten Auflage verließ uns Charly Kühnast. In der fünften Auflage dürfen wir jetzt Axel Miesen als zusätzlichen Autor begrüßen, der die Kapitel zu Ansible und Docker beigesteuert hat. Gleichzeitig durften wir für diese Auflage mit einem neuen Lektor – Christoph Meister – zusammenarbeiten.

Anfang 2011 erschien die Ursprungsversion des Buches. Aufgrund von Änderungen in den Distributionen und von Anregungen unserer Leser gingen wir in die zweite Runde für Mitte 2012. Nach den größeren Änderungen der dritten Auflage legten wir mit der vierten Auflage noch eins drauf und haben den Verlag gewechselt – nein, im Ernst, in diese Zeit fiel auch die Umbenennung des Verlags von Galileo Press in Rheinwerk Verlag. Da wir immer noch sehr viele Ideen haben, müssen wir uns neuen Herausforderungen stellen, und zwar, das Format zu halten und nicht zu ausschweifend zu werden. Wir kommen leider sehr nah an die technischen Limits, die ein Buch mit rund 1300 Seiten erreicht.

Und – WOW! – jetzt halten Sie die siebte Auflage in den Händen.

Seit der vierten Auflage bieten wir Unterstützung für die am weitesten verbreiteten Distributionen mit längerer Laufzeit und sind mit openSUSE Leap kompatibel mit dem SUSE Linux Enterprise Server, und mit CentOS sind wir kompatibel mit Red Hat Enterprise Linux. Sie, liebe Leser und Leserinnen, haben diese Änderungen angenommen und uns bewiesen, dass wir auf dem richtigen Weg sind.

Neben einem intensiven Review und einem Test, ob die angegebenen URLs noch funktionieren, habe ich noch kleinere Änderungen in die siebte Auflage aufgenommen.

Natürlich wurden alle Beispiele mit den neuen Versionen der Distributionen getestet und entsprechend angepasst. Wir hoffen, dass wir Ihnen mit diesem Buch weiterhin Unterstützung bei Ihrer täglichen Arbeit geben können!

Danksagung
Allen voran möchte ich meiner Frau danken, ohne die mein Teil an diesem Buch nie möglich gewesen wäre. Christoph Meister vom Rheinwerk Verlag danke ich für seine wertvollen Hinweise und für seine Geduld. Die Zusammenarbeit mit Dir macht Spaß, gerade auch weil Du sehr viel Verständnis für uns »Autoren im Nebenberuf« aufbringst.

Aber auch hinter den Kulissen haben wir bei »den Rheinwerkern« helfende Hände gefunden, allen voran möchte ich unserer Korrektorin Friederike Daenecke danken, die jeden noch so kleinen sprachlichen Fehler gefunden und behoben hat. Danke! Unserem Hersteller Norbert Englert möchte ich danken, weil wir mit seiner Hilfe aus dem Manuskript ein ansehnliches Buch machen konnten.

Mein besonderer Dank gilt aber meinen Mitautoren Daniel, Peer, Axel und Stefan für die tolle Zusammenarbeit.

Dass die Idee, die diesem Buch zugrunde liegt, so erfolgreich ist, damit haben wir nicht gerechnet. Noch viel weniger haben wir damit gerechnet, dass dieses Buch begonnen hat, sich als Standardwerk zu etablieren.

Jetzt halten Sie, liebe Leserin, lieber Leser, bereits die siebte Auflage in Ihren Händen. Sie ist möglich geworden, weil Sie uns mit Ihren Anregungen und Ihrer konstruktiven Kritik motiviert haben. Danke!

Vorwort von Stefan Kania

Bei dieser Auflage lagen meine Schwerpunkte beim Thema OpenLDAP, denn dort hat es die meisten Änderungen gegeben. Nicht nur Kleinigkeiten, sondern nach 14 Jahren wurde wieder eine neue Version veröffentlicht. Im ersten Moment sah es so aus, als würde alles beim Alten bleiben, denn die Konfiguration eines alten OpenLDAP 2.4 konnte direkt übernommen werden. Aber jeder weitere Blick zeigte, wie viel Potential in der neuen Version liegt. Die komplette Replikation wurde erneuert, verbessert und noch performanter gestaltet. Zusätzliche Overlays sorgen für noch mehr Möglichkeiten. Alle Neuerungen kann man in einem Kapitel mit 100 Seiten gar nicht abdecken, aber ich denke, dass Sie einen guten Überblick über die Neuerungen erhalten.

Im Samba-Kapitel habe ich das Thema Dateisystemrechte überarbeitet und einige Dinge so genau beschrieben, wie es auf wenigen Seiten möglich war. Auch die Gruppenrichtlinien für Linux-Clients hätte ich gerne aufgenommen, aber leider reicht der Platz hierfür nicht aus.

Das Kerberos-Kapitel habe ich auf die neue OpenLDAP-Version angepasst und auch hier alles noch mal überarbeitet.

Danksagung
Mein Dank geht dieses Mal in erster Linie an den Verlag, der unser Projekt schon so lange unterstützt und uns die Möglichkeit gibt, soviel Wissen zusammen zu stellen. Bei den Autoren muss ich doch dieses Mal einen meiner Mitstreiter besonders hervorheben, der der schon seit ein paar Auflagen den gesamten Satz übernimmt und dafür sorgt, dass das Buch am ende nicht nur viel Inhalt hat, sondern auch noch gut aussieht. Danke, Daniel, für die viele Arbeit, die du immer wieder in das Buch steckst.

Vorwort von Peer Heinlein

Als ich 1992 als Jugendlicher eine Computermailbox zu meinem Hobby machte, kam mir nie in den Sinn, dass dies auch fast 30 Jahre später noch mein täglicher Lebensinhalt sein würde.

Was anfangs noch ein MS-DOS-System war, wurde schon wenig später auf das damals noch revolutionäre und brandneue Linux-System umgerüstet. Den Um- und Irrweg über ein Windows-System habe ich darum nie gehen müssen – weder auf Servern noch auf meinen Privatcomputern –, und vermutlich liegt es daran, dass ich bis heute ein halbwegs entspanntes Verhältnis zu meinen Kisten habe. Sie machen schließlich das, was sie machen sollen. Meistens.

Die Vernetzung von Menschen und der freie Fluss der Kommunikation sind seitdem mein Lebensinhalt geworden. Seit rund 30 Jahren bin ich darum als Trainer dabei, anderen Administratoren die Linux-Systemadministration zu vermitteln: technische Fakten, vor allem aber auch »Softskills«, also Kompetenzen rund um das technische Verständnis der Abläufe, die Fähigkeit, sich selbst neue Themen zu erarbeiten, sicher und zielgerichtet Fehler einzukreisen und Fehlverhalten zu debuggen. Die Arbeit mit Menschen ist es, die Spaß macht. Computer selbst sind kein Selbstzweck, sie sind nur Mittel zum Zweck: Arbeitstiere. Aber praktische Arbeitstiere, wenn man sie effizient und sicher einsetzt.

In diesem Werk habe ich vor allem die Verantwortung für die Mailserver-Kapitel übernommen, schließlich habe ich bereits einige Fachbücher rund um Postfix und Dovecot veröffentlicht. In diesem Administrationshandbuch haben wir die Gelegenheit genutzt, statt eines umfassenden vollständigen Nachschlagewerks eine klare, nachvollziehbare Anleitung zum Aufbau eines eigenen Mailsystems auszuarbeiten, ohne allzu viel Grundlagenwissen vorauszusetzen oder den Anspruch zu haben, den perfekten Postmaster auszubilden.

Dazu gehört natürlich auch im Bereich Anti-Spam/Anti-Virus ein Kapitel zu »rspamd«, denn das hat sich in den letzten Jahren vom Geheimtipp zum Standard in diesem Bereich entwickelt. Wir haben ihm viel zu verdanken – es hält nicht nur die Postfächer sauber von nervigem Spam, es beschützt auch uns und unsere Daten: Denn es übernimmt auch den Kampf gegen Viren und per Mail verschickte erpresserische Ransomware und hat hier sicherlich schon viel Ärger und (bei verschlüsselten Dateien?) auch Leid verhindert.

Eine ebensolche Allzweckwaffe ist die bewährte Monitoring-Lösung »Checkmk«: Damit überwachen wir nicht nur plumpe Verfügbarkeiten von Diensten, sondern erfassen auch fortlaufend viele qualitative Messwerte vieler, vieler kleiner Details unserer Server. Das ist wichtig für die Analyse komplexer und verwirrender Fehlerbilder, notwendig für eine spätere Informationsgewinnung, wie es zu Beeinträchtigungen kommen konnte. Über Wochen hinweg erfasste Daten, automatisch grafisch aufbereitet. Eine wirklich mächtige Software, mit der zu arbeiten einfach Spaß macht.

Aber Spaß kann auch schnell vorbei sein. Getreu dem Motto »Zuerst hatten wir kein Glück, und dann kam auch noch Pech hinzu« habe ich auch das Kapitel »Backup und Recovery« zu verantworten, denn mit »Relax & Recover (ReaR)« kann die von vielen Administratoren so vernachlässigte Disaster Recovery bequem Einzug finden. Also: Vergessen Sie vor lauter Euphorie über Aufbau und Überarbeitung Ihrer Linux-Systeme nicht, rechtzeitig an den Plan B zu denken, falls es mal schiefgeht! Vielen Dank an Schlomo Schapiro für ReaR – und auch für die Fachkontrolle meines Backup-Kapitels.

Danksagung
Am »rspamd«-Kapitel haben meine Kollegen Carsten Rosenberg und Manu Zurmühl mit viel Aufwand mitgearbeitet und Anleitungen sowie Schulungsmaterial beigesteuert. Am »Checkmk«-Teil hat mein Kollege Robert Sander geduldig mit vielen Praxistipps und seiner ganzen Erfahrung aus- und nachgeholfen. Mit unseren weiteren Consulting-Kollegen Mirko Ludeke, Leah Herbach und Torsten Lange bilden alle zusammen ein starkes Team, auf das man sich stets verlassen kann und das, so kann man schon sagen, unseren Kunden in jedem Notfall schnell und kompetent hilft und eigentlich stets als Sieger aus einem Troubleshooting hervorgeht. Vielen Dank für Eure Geduld, Unterstützung und Hilfe - Ihr seid ein ganz wesentlicher Teil von »Heinlein« und bietet Rückendeckung und Verlässlichkeit für so viele Linux-Admins da draußen. Und bei dieser Gelegenheit einen ganz persönlichen Dank an Robert dafür, dass Du nun schon so lange für mich der Fels in der Brandung bist.

Vielen Dank an die aktuellen und früheren Autoren Charly, Daniel, Dirk und vor allem auch an Stefan Kania. Auch mit ihm arbeite ich seit rund 20 Jahren zusammen und habe ihn nicht nur als fachlich jederzeit hochkompetenten Spezialisten, sondern auch privat sehr zu schätzen gelernt. Vielen Dank für diesen doch schon immens langen und tollen Weg, den wir zusammen gehen.

Der Dank an mein Team hier bei Heinlein Support kann gar nicht groß genug sein, denn Ihr haltet mir in so vielen Bereichen den Rücken frei, damit ausreichend Zeit bleibt, auch Projekte wie dieses Buch voranzutreiben. Vielen Dank für alles, was wir gemeinsam im Team leisten und was auch jeder Einzelne bewegt, vorantreibt, korrigiert und gestaltet.

Zu guter Letzt danke ich meiner Frau Ivonne und meinen Kindern Antonia und Carolin für alles und uns als Familie dafür, dass alles so ist, wie es ist!

Vorwort von Daniel van Soest

Wie die Jungfrau Maria zum Kind, so bin ich zu diesem Buch gekommen – oder eher dazu, ein Koautor dieses Buches zu werden. Nun halten Sie bereits die siebte Auflage in den Händen; davon hätte ich vor fast zehn Jahren nicht einmal zu träumen gewagt.

Der Praxisbezug ist mir sehr wichtig, ebenso wie das Aufbauen von Hintergrundwissen. Während meiner nun mehr als 20-jährigen Berufserfahrung im Kommunalen Rechenzentrum Niederrhein (KRZN) durfte ich viele Hürden überwinden, aber noch mehr Erfolge feiern. Ich habe in diesem Buch stets versucht, nicht nur die Technik zu erläutern, sondern auch einen Großteil meiner Erfahrung mit einfließen zu lassen. Für dieses Buch war einer meiner Leitsätze: »Man kann nur die Technik beherrschen, die man versteht.«.

Ich hoffe, diesem Motto gerecht geworden zu sein und mit diesem Buch nicht nur eine Anleitung geschaffen zu haben, sondern Sie darin unterstützen zu können, die Technik zu verstehen, selbst kreative Ideen zu entwickeln und nicht nur stumpf nach Plan zu arbeiten.

Abschließend bleibt mir nur noch eins: Ihnen viel Spaß mit diesem Buch zu wünschen.

Danksagung
Vorab möchte ich mich bei meinen Koautoren Dirk, Stefan, Peer und Axel bedanken. Die Zusammenarbeit war sowohl kreativ als auch produktiv, auch wenn wir die eine oder andere Hürde meistern mussten: Das Ergebnis kann sich sehen lassen. Ebenso möchte ich mich bei Christoph Meister bedanken – ohne Dein Lektorat, die Geduld und die guten Lösungsansätze wären wir jetzt nicht da, wo wir sind. Nicht vergessen werden darf auch Norbert Englert: Vielen Dank für die schönen Bilder und noch viel mehr für die LaTeX- und Satz-Unterstützung. Natürlich darf die Korrektorin nicht vergessen werden – vielen Dank, Frau Daenecke. Ebenso geht mein Dank an meine Band (4Dirty5): Danke, dass Ihr mir die Möglichkeit gebt, den Ausgleich zu bekommen, den ich zum Alltag brauche, und dass ihr meine gelegentliche Abwesenheit verkraftet habt.

Zum Abschluss möchte ich mich bei den wichtigsten Personen in meinem Leben bedanken, ohne viele Worte zu bemühen: Ich bin dankbar, Euch meine Familie nennen zu dürfen. Danke, Nicole, danke Tom, danke Linda!

Vorwort von Axel Miesen

Zur fünften Auflage des Linux-Server-Handbuchs im Jahr 2019 durfte ich zum ersten Mal zwei Kapitel zu den Themen Ansible und Docker beisteuern; nun haben wir bereits die siebte Auflage erreicht, und ich freue mich immer noch sehr, bei diesem Projekt und diesem Team dabei zu sein.

Auch dieses Mal gibt es in meinen Kapiteln im Vergleich zur vorherigen Auflage wieder zahlreiche Änderungen. Beim Thema Ansible habe ich mich entschieden, viele Grundlagen noch

ausführlicher darzustellen und dafür (hauptsächlich aus Platzgründen) auf die Unterthemen »Rollen« und »Ansible Vault« zu verzichten bzw. nur einen kleinen Ausblick darauf zu geben.

Das Kapitel über Docker trägt nun den neuen Titel »Containervirtualisierung mit Docker und Podman«, womit eine weitere Neuerung bereits deutlich wird: Die Docker-Alternative Podman, die sich immer größerer Beliebtheit erfreut, hat nun auch in diesem Buch einen angemessenen Raum bekommen. Zudem habe ich den Unterabschnitt über private Registries sehr vereinfacht; ich setze dort nun auf die freie Software Harbor, die im Vergleich zur klassischen Docker Registry kaum Wünsche offen lässt und zudem sehr schnell an den Start zu bringen ist.

Ich hoffe, dass Ihnen mit meinen Kapiteln die ersten Schritte im Konfigurationsmanagement und in der schönen (nicht mehr ganz so neuen) Containerwelt leichter fallen werden, und wünsche Ihnen viel Spaß und Erfolg!

Danksagung
Mein großer Dank gilt zunächst meinen Autorenkollegen, die dieses Buch in vielen Jahren zu einem großen Erfolg geführt haben und mir auch immer mit nützlichen Tipps weitergeholfen haben. Auch dem Dank ans Rheinwerk-Team, namentlich Christoph Meister, Friederike Daenecke und Norbert Englert, möchte ich mich unbedingt anschließen.

Nicht zuletzt danke ich den beiden wichtigsten Personen in meinem Leben: meiner Lebensgefährtin Ana und meiner Tochter Lena. Danke, dass ihr immer für mich da seid, und dass ihr stets Verständnis dafür hattet, wenn ich mitunter auch am Wochenende mal an diesem Buch gearbeitet habe.

Über dieses Buch

An dieser Stelle möchten wir Ihnen erklären, was wir uns bei der Verwendung der verschiedenen Formatierungsmöglichkeiten gedacht haben. Hier finden Sie auch die Beschreibung zu den im Buch verwendeten Icons und die Begründung, warum wir uns gerade für diejenigen Distributionen entschieden haben, die im Buch verwendet werden.

Formales

Damit Sie den größtmöglichen Nutzen aus diesem Buch ziehen können, verwenden wir einige formale Konventionen, die im Folgenden erläutert werden.

Kommandozeile

Gleich zu Beginn ein Hinweis an den mausverwöhnten Windows-Nutzer: Wir werden im Rahmen dieses Buches hauptsächlich Gebrauch von der Kommandozeile machen, da sich viele Aufgaben unter Linux einfacher und ökonomischer durch einige Tastaturkommandos erledigen lassen. Nur in einem Kapitel stehen die grafischen Werkzeuge mehr im Vordergrund, und zwar im Samba-4-Kapitel. Auch als Linux-Admin werden Sie dort die grafischen Werkzeuge benötigen, denn nicht alle Aufgaben können über die Kommandozeile realisiert werden: Wenn Sie eine Active Directory-Domäne verwalten wollen, kommen Sie an den grafischen Werkzeugen nicht vorbei.

Das soll allerdings nicht heißen, dass wir gänzlich auf den Komfort einer grafischen Umgebung verzichten, denn wie bei vielen Dingen im Leben gilt auch hier: Die Mischung macht's.

Für viele Bereiche gibt es heute grafische Werkzeuge, gerade webbasierte, die Ihnen als Administrator das Leben leichter machen können. Auch wir nutzen diese Werkzeuge und werden an den entsprechenden Stellen auf sie eingehen.

Befehle eingeben

Für Kommandozeilenbefehle soll folgende Schreibweise verwendet werden: Im fließenden Text werden Konsolenbefehle durch `Nicht-Proportionalschrift` gekennzeichnet. Viele Beispiele zu den Kommandos werden aber auch in Listings dargestellt und in `Nicht-Proportionalschrift` wiedergegeben. In den Listings können Sie von der Befehlszeile bis zum Ergebnis alles nachvollziehen:

```
stefan@adminbuch~$ ps
 PID TTY          TIME CMD
 4008 pts/2    00:00:00 bash
 4025 pts/2    00:00:00 ps
```

Listing 1 Beispiel für ein Listing

Privilegierte Rechte

Für die Administration von Linux-Systemen werden Sie immer root-Rechte benötigen, um die entsprechenden Konfigurationsdateien bearbeiten oder um Dienste starten oder stoppen zu können.

Ubuntu vertritt im Unterschied zu anderen Linux-Distributionen eine eigene Philosophie: Der Standardbenutzer der ersten Installation kann jeden Administratorbefehl durch Voranstellen des Befehls `sudo` ausführen. Anschließend muss dann das Passwort des Standardbenutzers eingegeben werden:

```
stefan@adminbuch~$ sudo systemctl restart systemd-networkd
[sudo] password for <user>: <Hier eigenes Passwort eingeben>
```

Listing 2 Arbeiten als root

Sind mehrere Befehle als Administrator einzugeben, so kann das Voranstellen von `sudo` auch lästig werden. In diesem Fall verschaffen Sie sich mit dem folgenden Befehl vorübergehend eine root-Shell:

```
stefan@adminbuch~$ sudo -s
[sudo] password for <user>: <Hier eigenes Passwort eingeben>
root@adminbuch~#
```

Listing 3 Eine root-Shell öffnen unter Ubuntu

Eingabe langer Befehle

Und noch eine weitere wichtige, eher technische Konvention: Einige der vorgestellten Kommandozeilenbefehle oder Ausgaben von Ergebnissen erstrecken sich über mehrere Buchzeilen. Im Buch kennzeichnet am Ende der entsprechenden Zeilen ein »\«, dass der Befehl oder die Ausgabe in der nächsten Zeile weitergeht. Geben Sie das Kommando auf der Konsole ohne den Backslash und ohne Zeilenumbruch ein.

Screenshots

Wie heißt es doch so schön: Ein Bild sagt mehr als tausend Worte. Wann immer es sinnvoll erscheint, soll daher ein Screenshot zur Erhellung des Sachverhalts beitragen.

Internetverweise

Da wir in diesem Buch sehr viele verschiedene Dienste ansprechen, ist es nicht möglich, alle Funktionen und Fähigkeiten eines Dienstes bis ins kleinste Detail zu beschreiben. Aus diesem Grund haben wir an geeigneten Stellen auf Internetadressen verwiesen. Verweise auf Internetadressen werden besonders ausgezeichnet, zum Beispiel so: *www.debian.org*

Icons

Sie werden in den einzelnen Kapiteln am Rand häufig Icons finden, die Sie auf bestimmte Zusammenhänge oder Besonderheiten hinweisen sollen. Die Icons haben die folgenden Bedeutungen:

Hier wird es immer sehr wichtig

Wann immer Sie das nebenstehende Symbol sehen, ist Vorsicht angeraten: Hier weisen wir auf besonders kritische Einstellungen hin oder auf Fehler, die dazu führen können, dass das System nicht mehr stabil läuft. Damit sich die Warnungen deutlich vom restlichen Text abheben, haben wir diese Textbereiche zusätzlich mit einem grauen Kasten hinterlegt.

Beispiele – etwa für Konfigurationsdateien – haben wir mit diesem Symbol gekennzeichnet. Wir haben an vielen Stellen Beispiele eingefügt, die es Ihnen leichter machen, eine entsprechende Aufgabe umzusetzen.

Alle Textstellen, die wir mit diesem Icon versehen haben, sollten Sie unbedingt lesen! Hier handelt es sich um wichtige Hinweise zu den unterschiedlichen Distributionen, die wir verwenden, oder um wichtige Eigenschaften oder Konfigurationsmöglichkeiten eines Dienstes.

Es gibt keine fehlerfreie Software! Große und kleine Fehler, die bei den einzelnen Diensten bekannt sind, werden durch diesen kleinen »Bug« gekennzeichnet. Die nachweislich erste Erwähnung des Wortes »Bug« stammt übrigens von Grace Hopper, einer Computerpionierin aus den USA: *http://de.wikipedia.org/wiki/Grace_Hopper*

Bei diesem Symbol finden Sie nützliche Tipps und Tricks zu bestimmten Aufgaben.

Linux-Distributionen

Als damals der Gedanke zur ersten Auflage für dieses Buch aufkam, mussten wir uns erst einmal einig werden, welche Distributionen wir denn für das Buch verwenden wollten. Aufgrund der folgenden Kriterien haben wir dann unsere Entscheidung getroffen:

- Wir wollten auf jeden Fall mindestens eine Distribution, die *rpm*-Pakete, und eine, die *deb*-Pakete für die Softwareverwaltung nutzt.
- Da es in diesem Buch um Serverdienste geht, musste die Distribution nicht unbedingt die aktuellsten Pakete haben, wie man es gerne auf einem Desktop hat, sondern uns kam es in erster Linie auf die Stabilität an. Dennoch haben wir bei manchen Diensten durchaus auf eine bestimmte minimale Versionsnummer geachtet.
- Die Distributionen sollten sehr verbreitet sein und oft in Firmen zum Einsatz kommen.
- Der Supportzeitraum sollte mindestens vier bis fünf Jahre betragen, also ungefähr die Laufzeit, die IT-Systeme in Unternehmen haben.

Aufgrund dieser Kriterien haben wir uns im Laufe der Zeit immer wieder Gedanken gemacht, welche Distribution wir einsetzen, so auch dieses Mal. Dabei ist die Auswahl auf die folgenden Distributionen gefallen:

- **Debian Bullseye**
  Debian ist seit Jahren für stabile Versionen und hohe Zuverlässigkeit bekannt. Auch ist die Bereitstellung der Sicherheitsupdates für einen langen Zeitraum gesichert.
- **openSUSE Leap**
  Viele Leser haben uns gefragt, warum wir nicht mehr mit openSUSE arbeiten, und wir sehen auch, dass die openSUSE-Distributionen, auch in Unternehmen, immer öfter eingesetzt werden. Gerade wenn es um Desktop-Systeme in Domänen geht, wird openSUSE oft verwendet. Deshalb haben wir uns auch im Samba-4-Kapitel dafür entschieden, openSUSE Leap als grafischen Client einzusetzen.
- **Ubuntu-Server 22.04 LTS**
  Der Ubuntu-Server basiert auf Debian und stellt mit der *LTS-(Long Term Support-)*Version eine gute Alternative zum Debian-Server dar. Der Ubuntu-Server setzt dabei auf neuere Pakete und Kernel als Debian, da bei Ubuntu die Releasezyklen kürzer sind.
- **CentOS Stream**
  CentOS wird auch in dieser Auflage genutzt, und zwar die Version *Stream*. Bei CentOS Stream handelt es sich um ein *rolling release*, das bedeutet, dass es keine neuen Versionen mehr geben wird, sondern die bestehende Version immer aktualisiert wird. Bei dieser neuen Edition hat es die verschiedensten Änderungen gegeben, auch hinsichtlich der unterstützten Software. So ist zum Beispiel OpenLDAP nicht mehr Bestandteil der Distribution. Wir haben lange überlegt, ob und welche Version von CentOS wir ins Buch aufnehmen, und dann für CentOS Stream entschieden. Wir hoffen, dass auch in Zukunft alles das, was wir im Buch beschrieben haben, weiterhin möglich sein wird.

Wenn Sie sich jetzt fragen: »Aber meine Lieblingsdistribution erfüllt die Punkte auch, warum ist die nicht dabei?«, können wir an dieser Stelle nur sagen, dass wir natürlich alle Dienste unter allen von uns verwendeten Distributionen konfiguriert und ausgetestet haben. Allein für das Testen mit vier verschiedenen Distributionen benötigt man schon eine geraume Zeit. Deshalb haben wir uns für diese vier Distributionen entschieden.

Jetzt bleibt uns nur noch, Ihnen viel Spaß mit dem Buch zu wünschen und zu hoffen, dass Ihnen unser Buch bei Ihrer täglichen Arbeit eine Hilfe sein wird.

Inhalt

11 Datenbank 371

12 Syslog 397

TEIL VI Automatisierung

27 Scripting 1043

28 Konfigurationsmanagement mit Ansible 1065

TEIL VII Sicherheit, Verschlüsselung und Zertifikate

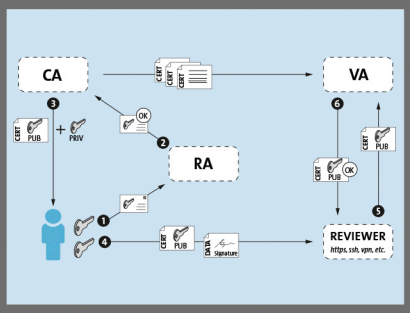
30 Sicherheit 1183

Linux-Server

»Von Praktikern für Praktiker.«

Das Schweizer Messer für Linux-Admins

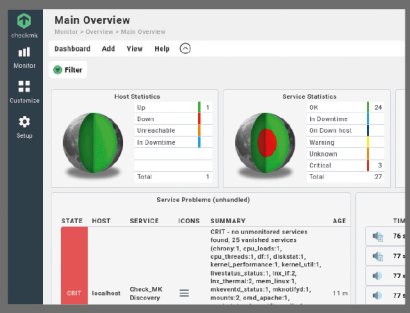
Linux-Serversysteme sind das Fundament einer funktionierenden IT-Infrastruktur. Von den Netzwerkgrundlagen bis zur Virtualisierung, vom automatisierten Deployment bis zum sicheren Einrichten der Dienste begleitet Sie dieses Standardwerk bei der Administration und zeigt Ihnen, wie Sie Ihre Umgebung sicher und effizient im Griff haben.



Sicher planen und einrichten

```
FROM httpd:2.4
WORKDIR /usr/local/apache2
ENV COLOR=white
RUN sed -ri \
-e 's/^#(LoadModule .*mod_include.so)/\1/'
-e 's/(Options Indexes FollowSymLinks)/\1
conf/httpd.conf
RUN /bin/echo -e '\n\
AddOutputFilter INCLUDES .html\n\
SetEnv COLOR ${COLOR}\n\
' >> conf/httpd.conf
COPY index.html htdocs/
CMD ["httpd-foreground", "-e", "debug"]
```

Werkzeuge richtig nutzen



Mit zahlreichen Praxistipps

So verwalten Sie Linux sicher und effizient

Von Apache bis Zertifikat finden Sie in diesem Buch alles, was Sie für die fortgeschrittene Server-Administration brauchen. Dieses Hintergrundwissen und die vorgestellten Werkzeuge und Automatisierungsskripte helfen Ihnen, einen störungsfreien Betrieb sicherzustellen.

Schritt für Schritt zur optimalen Infrastruktur

Detaillierte Anleitungen und kommentierte Konfigurationen zeigen Ihnen, wie Sie die Server-Dienste sicher in Ihre IT-Infrastruktur integrieren. Dank Containervirtualisierung und automatisiertem Konfigurationsmanagement erhöhen Sie die Effizienz und Agilität Ihrer Systeme.

Netzwerke umfassend einrichten und absichern

Ob Routing, Bonding, IPv6 oder Firewalls: Richten Sie Ihre Netze optimal ein und behalten Sie mit Monitoring den Überblick. Sorgen Sie für den Schutz Ihrer sensiblen Daten, indem Sie OpenVPN einrichten, Backups einsetzen und Ihre Kommunikation und Systeme verschlüsseln.

Alle Konfigurationsbeispiele als Vorlage zum Download

Das Autorenteam

Dirk Deimeke, Stefan Kania, Daniel van Soest, Peer Heinlein und Axel Miesen: Dieses Expertenteam steht für langjährige Erfahrung im Einsatz von Linux-Servern. Die Autoren sind auf die Administration mittlerer bis sehr großer Linux-Infrastrukturen spezialisiert und geben ihr Wissen auch in praxisnahen Schulungen weiter.

Aus dem Inhalt

Rheinwerk Computing

€ 69,90 [D] € 71,90 [A]

9 783836 296151

ISBN 978-3-8362-9615-1

Linux