HANSER


# Leseprobe

zu

# „OpenLDAP in der Praxis“

von Stefan Kania und Andreas Ollenburg

Print-ISBN: 978-3-446-46387-5
E-Book-ISBN: 978-3-446-46531-2
E-Pub-ISBN: 978-3-446-46584-8

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/978-3-446-46387-5
sowie im Buchhandel

# Inhalt

# Geleitwort

Es ist mittlerweile mehr als 20 Jahre her, seit das OpenLDAP-Projekt seine erste Version veröffentlicht hat. Solche langen Zeiträume sind im Open Source-Umfeld eher ungewöhnlich. Es zeigt, dass die Software ausgereift ist und die Entwickler die Bedürfnisse ihrer Nutzer erfüllen. Das Protokoll LDAP ist sogar noch ein paar Jahre älter, und trotzdem spielt es nach wie vor eine große Rolle. Es gibt unzählige Anwendungen, die LDAP als Benutzer- und Gruppenverzeichnis unterstützen. Durch die Erweiterbarkeit des Schemas kann OpenLDAP aber noch viel mehr: Das reicht von DNS-Einträgen über E-Mail-Verteiler bis hin zur Konfiguration von Telefonen mit Asterisk. Die Kombination einer breiten Basis standardisierter Datenobjekte mit der Möglichkeit der Erweiterbarkeit macht es sehr flexibel einsetzbar. OpenLDAP bildet damit einen Teil des Rückgrats der IT vieler Unternehmen.

Meine ersten Erfahrungen mit OpenLDAP reichen nun mehr als 15 Jahre zurück. Damals haben wir als eine Gruppe Studenten die Software LDAP Account Manager (LAM) entwickelt. Das Samba-Projekt hatte ganz frisch die Integration mit LDAP ermöglicht. An Samba in Version 2 werden sich wahrscheinlich nur wenige erinnern. Für Administratoren gab es zu dieser Zeit leider kein brauchbares grafisches Verwaltungswerkzeug, sodass wir selbst entwickeln mussten. Gestartet mit Fokus auf Samba 2+3 und Unix-Benutzern sowie -Gruppen besitzt LAM mittlerweile knapp 100 Module für diverse Anwendungen. Hieran sieht man auch, wie breit LDAP eingesetzt werden kann. Vielleicht setzen Sie in Ihrem Unternehmen bereits eine eigene LDAP-Schemaerweiterung ein oder planen dies. Das kann gerade für Eigenentwicklungen durchaus sinnvoll sein.

In den letzten Jahren hat sich das LDAP-Protokoll selbst zwar kaum verändert, aber OpenLDAP hat viele Änderungen erfahren. Besonders die vielen Overlays wie memberOf für Gruppenmitgliedschaften oder unique zur Durchsetzung von Eindeutigkeit unterstützen in vielen Anwendungsfällen. So vermeiden Sie z. B. die doppelte Vergabe von uid-Nummern. Mit ppolicy können nun auch Passwortrichtlinien durchgesetzt werden. Sogar Einträge, die nach einer definierten Zeit automatisch gelöscht werden, sind möglich. Schauen Sie sich hierzu das Overlay «dds» an. Sie finden alle genannten Overlays in diesem Buch beschrieben.

Es freut mich besonders, dass in diesem Buch auch die Anbindung von Kerberos an LDAP behandelt wird. Wer das schon einmal probiert hat, weiß ein Lied davon zu singen, wie viele Fallstricke es hier gibt. Das ist besonders frustrierend, wenn etwas nicht sofort funktioniert. Die Log-Ausgaben von Kerberos sind doch sehr spärlich und die Dokumentation nicht immer einfach verständlich. In diesem Buch finden Sie auch eine Beschreibung,

wie man Kerberos-Keytabs verwendet. Ich kann mich noch genau erinnern, wie viel Zeit ich zum Aufsetzen der Kerberos-Domäne verwendet habe, bevor ich überhaupt erst die Implementierung des Kerberos-Moduls für LAM starten konnte. Hier hilft die Schritt-für-Schritt-Anleitung im Kerberos-Kapitel ungemein.

Die Praxisbeispiele und die Kapitel zu Docker und Monitoring runden das Thema sehr gut ab. Damit sind alle typischen Aufgaben in einem Buch zusammengefasst. Es eignet sich für die erste Installation genauso gut wie als Nachschlagewerk für den laufenden Betrieb. Stefan Kania hat aus seiner langjährigen Arbeit als Trainer einen sehr guten Überblick, wo die typischen Herausforderungen liegen und wie man sie bewältigt.

Ich wünsche Ihnen viel Spaß beim Lesen dieses Buches!

*Roland Gruber*

Freising

# Vorwort

Herzlich willkommen bei der ersten Auflage unseres OpenLDAP-Buchs. Wir arbeiten jetzt schon seit mehreren Jahren in unterschiedlichen Projekten mit OpenLDAP und haben uns bis zu diesem Zeitpunkt immer alle benötigten Informationen aus dem Internet zusammengesucht. Viele der Artikel, Blogs und Einträge in Mailing-Listen haben uns geholfen, die meisten Aufgaben zu lösen, aber an einigen Stellen wurde auch immer mal ein Schritt übersprungen oder die Einträge waren so alt, dass uns die Informationen nicht geholfen haben. So sind bei uns in den letzten Jahren eine Menge Anleitungen und Dokumentationen entstanden (ja, es gibt Menschen, die schreiben Dokumentationen), die wir auf diesem Wege mit Ihnen teilen wollen.

Als wir uns den Aufbau des Buchs überlegt haben, waren wir uns schnell einig, dass das Buch Sie in die Lage versetzen soll, die hier aufgeführten Techniken möglichst einfach auf Ihre Umgebung umzusetzen. Wir wollten kein Buch schreiben, in dem jede Option der Manpage eines Kommandos beschrieben wird, sondern Sie sollten Beispiele erhalten, mit denen Sie alles sofort testen können. Natürlich haben wir an der einen oder anderen Stelle Optionen und Parameter erklärt, wenn die Verwendung nicht sofort eindeutig ist.

Am Anfang stand die Frage: «Welche Distribution verwenden wir für die Beispiele?» Da wir beide die meisten Erfahrung mit Debian-basierten Distributionen besitzen, war uns gleich klar, dass wir auf jeden Fall Debian nutzen wollen, und zwar die aktuelle Version Buster. Als zweite Distribution haben wir dann CentOS in der aktuellen Version 8 hinzugenommen. Da wir auch beide Erfahrungen mit anderen Distributionen haben, denken wir, dass eine Umsetzung auf andere Distributionen recht einfach ist. Der größte Unterschied liegt in den seltensten Fällen bei der Konfiguration des Dienstes, sondern meist nur bei der Installation der Pakete und der Grundkonfiguration des Dienstes. So können Sie alles, was wir hier im Buch erklären, auch einfach auf andere Distributionen umsetzen.

Für uns war es auch wichtig, dass wir bei allen Beschreibungen das Thema Sicherheit nicht außer Acht lassen. Sämtliche Kommunikation zwischen Servern und Diensten ist immer so konfiguriert, dass alle Daten verschlüsselt übertragen werden. Ein großer Abschnitt ist auch dem Thema ACLs gewidmet, da dort die meisten Sicherheitslücken auftreten.

Um mit dem Buch auf der Höhe der Zeit zu sein, beschreiben wir auch in einem Kapitel, wie Sie OpenLDAP in Docker-Containern betreiben können.

Jetzt bleibt uns nur noch, Ihnen viel Spaß mit dem Buch zu wünschen.

# Vorwort von Stefan Kania

Ich schreibe jetzt schon seit fast zehn Jahren Fachbücher. Angefangen hat alles mit dem Linux-Server-Buch. In dem Buch habe ich auch die beiden Kapitel zum Thema OpenLDAP und Kerberos verfasst. Fast genauso lange habe ich immer wieder überlegt, ein Buch nur zum Thema OpenLDAP zu schreiben, in dem ich alle meine Erfahrungen zusammenfassen kann und genügend Platz habe, alle meine Ideen umzusetzen. Denn das ist in einem Buch, das viele Dienste abdeckt, nicht möglich. Immer wieder habe ich den Ansatz gemacht und immer wieder habe ich das Projekt wieder verworfen, weil andere Bücher wie das Samba4-Buch oder andere Projekte dazwischen gekommen sind. Dann habe ich im Sommer 2019 auf einer Konferenz in Berlin abends mit Andreas darüber geredet und versucht, ihn für das Projekt zu gewinnen. Zur fortgeschrittenen Stunde hat er mir dann auch die Zusage gegeben. Nur so, zusammen mit Andreas, war ich dann in der Lage, dieses Buch umzusetzen.

## Danksagungen

An erster Stelle will ich auf jeden Fall Andreas danken. Ohne ihn wäre dieses Buch jetzt noch nicht fertig, viel schlimmer, es würde wahrscheinlich immer noch als angefangenes Projekt auf meinem Schreibtisch schlummern.

Auch Roland Gruber, dem Entwickler des LDAP Account Managers, möchte ich danken: zum einen dafür, dass er nach so vielen Jahren immer noch so viel an Entwicklung in den LAM steckt und das Programm stetig weiterentwickelt. Der LAM dient mir in meinen Schulungen und bei vielen Kunden immer als sehr hilfreiches Werkzeug. Zum anderen aber auch dafür, dass er sich unser Werk vorher schon angeschaut und das Geleitwort zum Buch geschrieben hat.

Ein großer Dank gilt auch dem Hanser Verlag, der vor zwei Jahren mein Samba4-Buch übernommen und es mir damit ermöglicht hat, auch das Buch weiterzuführen. Das ist jetzt mein zweites Buch beim Hanser Verlag, und die Zusammenarbeit macht einfach nur Spaß.

Ohne eine Familie kann so ein Buch auch nicht gelingen, denn für das Schreiben dieses oder anderer Fachbücher geht so manche Stunde am Schreibtisch ins Land. Das klappt nur dann, wenn man eine Frau hat, die einem den Rücken frei hält, einen motiviert, wenn man gerade keine Lust mehr hat, aber auch dafür sorgt, dass man Pausen macht und seine Zeit auch mal anders verbringt. Aus diesem Grund: Danke, mein Schatz.

Diese Auflage möchte ich einem sehr guten Freund widmen, ohne den ich heute beruflich nicht dort wäre, wo ich bin, der auch immer für mich da war, wenn es mal privat schwierig wurde. Michael, du warst für mich immer der beste Freund und Berater in den letzten Jahren, du fehlst mir.

# Vorwort von Andreas Ollenburg

Im Gegensatz zu Stefan ist dies mein Erstlingswerk. Ich war lange Jahre als Trainer und Consulter für SUSE-, Novell- und Microsoft-Produkte auf Achse. Dadurch war das Thema «Verzeichnisdienst» vor allem durch NDS/eDirectory, aber auch Active Directory, X.500 und eben auch OpenLDAP immer präsent und entwickelte sich im Lauf der Zeit zu einem meiner Schwerpunkte. Auf den «Certified Directory Engineer» von Novell bin ich auch heute noch ein bisschen stolz.

Wie Stefan schon erwähnte, war es insbesondere hinsichtlich OpenLDAP nicht immer einfach, einheitliche Dokumentationen zu finden. Als wir uns 2019 in Berlin wieder einmal trafen und er mir dieses Buchprojekt vorstellte, wurde ich daher schnell hellhörig. Bis er mich dann aber soweit hatte, mich mit ihm an ein Buch zu wagen, bedurfte es dann tatsächlich noch etwas Überzeugungsarbeit nebst dem gemeinsamen Genuss geistvoller Getränke. Aber jetzt bin ich doch froh, mich von Stefan habe überreden zu lassen.

## Danksagungen

Mein erster Dank geht natürlich an Stefan. Zum einen, weil er mich beharrlich bearbeitete, mit ihm dieses Buch zu realisieren. Zum anderen für seine wertvollen Tipps, seine Motivation und seine Geduld mir gegenüber, was das Schreiben eines Buches angeht. Ich hoffe, ich habe ihm nicht allzu viele graue Haare beschert. Aber bei unserer Frisur fällt das ja eh nicht so auf.

Bezogen auf den Hanser Verlag und Roland Gruber kann ich mich Stefans Worten nur anschließen.

Der größte Dank aber geht an die drei wertvollsten Menschen in meinem Leben: der besten Ehefrau von allen – auch wenn sie das nicht hören will – und unseren beiden großartigen – ebenfalls besten – Töchtern. Die drei mussten in den letzten Monaten oft auf mich verzichten und manchmal viel Geduld und Rücksicht aufbringen. Wenn es mal so gar nicht lief, haben sie mich nach Kräften unterstützt und manches Mal mehr oder weniger sanft angetrieben, weiterzumachen. Danke, ihr drei, was wäre ich ohne euch?

# 3 Installation des ersten OpenLDAP

In der Einleitung haben wir schon auf die zwei Distributionen hingewiesen, die wir im Buch ansprechen werden. Gerade in diesem Kapitel wird der Unterschied zwischen den beiden Distributionen am deutlichsten werden. Wir werden hier beiden Distributionen einen eigenen Abschnitt einräumen und jeweils die gesamte Installation und Grundkonfiguration des OpenLDAP vornehmen. In den weiteren Kapiteln sind die Unterschiede dann etwas kleiner, sodass meist nur einzelne Abschnitte für die Distributionen ergänzt werden.

## 3.1 Grundlegende Überlegungen

Bevor wir mit der Installation beginnen, möchten wir hier auf die beiden verschiedenen Möglichkeiten der Konfiguration eingehen. Zum einen gibt es die Möglichkeit, den OpenLDAP über eine statische Konfiguration einzurichten und auch während der gesamten Nutzung des LDAP jede Änderung statisch vorzunehmen. Bei der statischen Konfiguration werden alle Änderungen immer aus einer Konfigurationsdatei gelesen.

Bei der dynamischen Konfiguration werden sowohl die Grundkonfiguration als auch alle weiteren Änderungen über ldif-Dateien in einer speziellen Datenbank im LDAP gespeichert, und der LDAP-Server verwaltet sich quasi selbst.

Wir werden in diesem Buch die Grundkonfiguration für beide Distributionen einmal mit der statischen und einmal mit der dynamischen Konfiguration durchführen. Wir werden im Verlauf des Buchs immer noch beide Möglichkeiten erwähnen und auch die entsprechenden Einträge aufzeigen. Sie sollten sich aber überlegen, gerade wenn Sie eine komplett neue LDAP-Infrastruktur aufbauen wollen, ob Sie nicht sofort die dynamische Konfiguration verwenden. Der Grund ist der, dass die statische Konfiguration auf der WebAusite *https://www.openldap.org* als *deprecated* angegeben wird. Das bedeutet, dass die Möglichkeit der Konfiguration über eine einfache ASCII-Textdatei leider auf Dauer nicht mehr möglich sein wird.

### 3.1.1 Die statische Konfiguration

Bei der statischen Konfiguration handelt es sich um die klassische Variante der Konfiguration. Alle Konfigurationsparameter werden in einer Konfigurationsdatei abgelegt. Immer,

wenn Sie eine Änderung durchgeführt haben, müssen Sie den LDAP-Server neu starten, damit die Änderung wirksam wird. Wenn Sie mehrere LDAP-Server einsetzen, müssen Sie die Konfiguration immer auf allen LDAP-Servern anpassen. Einzelne Änderungen können Sie innerhalb der Konfigurationsdatei mit Kommentaren versehen, um später sehen zu können, wann und warum Sie welche Änderung durchgeführt haben. Die Konfiguration lässt sich einfach sichern und auf einen anderen Server übertragen. Immer, wenn Sie nur wenige LDAP-Server verwenden und später selten bis gar keine Änderungen an der Konfiguration vornehmen müssen, ist die statische Konfiguration eine gute Wahl; die Frage ist nur, wie lange dieser Weg noch möglich ist. Wollen Sie heute einen OpenLDAP-Server mit der statischen Konfiguration einrichten, müssen Sie das in allen Distributionen erst umstellen, denn alle Distributionen verwenden heute standardmäßig die dynamische Konfiguration für OpenLDAP.

Nicht nur die grundlegende Konfiguration wird in der statischen Konfiguration in der Konfigurationsdatei abgelegt, sondern später auch alle ACLs und die Konfiguration der eventuell verwendeten Overlays (bei Overlays handelt es sich um eine Art Plug-in, das die Funktion des LDAP-Server erweitert; mehr zum Thema Overlays finden Sie in Kapitel 8, «Einsatz von Overlays»).

Sie können die Konfiguration Ihres LDAP-Servers auch erst statisch beginnen und später auf die dynamische Konfiguration umstellen. So haben Sie am Anfang den einfachen Einstieg und können die komplexen Änderungen dann dynamisch vornehmen. Alle Einstellungen werden Sie zuerst in der statischen Konfiguration vornehmen, und erst, wenn Sie den LDAP-Server komplett eingerichtet haben, stellen Sie auf die dynamische Konfiguration um.

Da die Datei slapd.conf bei allen Distributionen den nahezu selben Inhalt hat, möchten wir Ihnen an dieser Stelle die einzelnen Zeilen der Konfiguration erklären. Später werden wir auf die Unterschiede in den beiden Distributionen eingehen. Der größte Unterschied zwischen den Distributionen wird fast ausschließlich den globalen Bereich der Konfiguration betreffen, denn hier werden zum Beispiel die Pfade für die Module und die Schemendateien angegeben. In Listing 3.1 sehen Sie den Inhalt der Datei:

**Listing 3.1** Die slapd.conf

```
# This is the main slapd configuration file. See slapd.conf for more
# info on the configuration options.

########################################################################
# Global Directives:


# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include         /etc/ldap/schema/dyngroup.schema
```

```
# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile         /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile        /var/run/slapd/slapd.args

# Read slapd.conf(5) for possible values
loglevel        256

# Where the dynamically loaded modules are stored
modulepath    /usr/lib/ldap
moduleload    back_mdb

# The tool-threads parameter sets the actual amount of cpu's that is used
# for indexing.
tool-threads 1

#######################################################################
# Specific Directives for database #1, of type mdb:
# Database specific directives apply to this databasse until another
# 'database' directive occurs
database        mdb

# The base of your directory in database #1
suffix          "dc=example,dc=net"

# rootdn directive for specifying a superuser on the database.
rootdn          "cn=admin,dc=example,dc=net"
rootpw         {SSHA}Q9Qbs+SyRMldHOwicH6Q8xI62u0tgMmE

# The maximum number of entries that is returned for a search operation

sizelimit 500

# Where the database file are physically stored for database #1
directory       "/var/lib/ldap"

# Index-entries used in this database
index           objectClass eq

# Save the time that the entry gets modified, for database #1
lastmod         on

# Default ACLs for this database
access to attrs=userPassword,shadowLastChange
        by anonymous auth
        by self write
        by * none
```

```
access to *
        by * read

access to dn.base="" by * read
```

Die Konfigurationsdatei besteht aus zwei Teilen: Der erste Teil oberhalb der Linie aus Hashes ist der globale Teil, der für alle Datenbanken, die der OpenLDAP bereitstellt, relevant ist. Der Teil unterhalb betrifft immer eine bestimmte Datenbank. Der Parameter *database mdb* markiert dabei den Beginn einer neuen Datenbank.

- Als Erstes werden die Schemendateien eingebunden, die Reihenfolge ist dabei sehr wichtig. Beim Start des LDAP-Servers werden die Dateien eine nach der anderen abgearbeitet. Da durch die Möglichkeit der Vererbung von Attributen eine Abhängigkeit zwischen den einzelnen Schemata besteht, müssen Sie immer wissen, welche Objektklasse eventuell Attribute einer anderen Objektklasse nutzt. Die Standardobjektklassen, die hier eingebunden sind, befinden sich schon in der richtigen Reihenfolge. Wenn Sie ein eigenes Schema mit eigenen Objektklassen erstellen, ist es wichtig, dass Sie am Anfang Ihres Schemas etwaige Abhängigkeiten kommentieren.
- In den Zeilen *pidfile /var/run/slapd/slapd.pid* und *argsfile /var/run/slapd/slapd.args* werden die Prozess-ID und die Argumente, die beim Start des OpenLDAP-Servers verwendet werden, abgelegt. Gerade bei diesen beiden Einträgen müssen Sie später darauf achten, welche Distribution Sie einsetzen.
- Das *loglevel* legt fest, welche Ereignisse geloggt werden. Dabei werden die Zahlen nicht einfach hochgezählt, sondern die einzelnen Werte werden binär übergeben. Am Anfang ist ein *loglevel 256* eine gute Einstellung, denn da werden alle Zugriffe geloggt. Welche Loglevel Sie verwenden können, finden Sie in der Manpage zur slapd.conf.
- Die Parameter *modulepath* und *moduleload* werden immer dann benötigt, wenn Sie zusätzliche Module für zusätzliche Funktionen nutzen wollen. Bei Debian müssen Sie später für alle Funktionen hier Module laden, bei CentOS sind die meisten Module fest eingebunden.
- Der Parameter *sizelimit 500* legt fest, wie viele Antworten bei einer Suche zurückgegeben werden. Gerade wenn Sie einen großen LDAP-Baum mit mehreren Tausend Objekten pflegen, kann die Antwort den Server stark belasten. Deshalb diese Grenze. Über Filter können Sie die Suche einschränken. Die Übersteuerung der Einschränkung der zurückgegebenen Objekte bei der Suche muss immer direkt in der Datenbank definiert werden, für die sie zutreffen soll.
- Wenn Sie sehr viele Indexdatenbanken einsetzen, um die Suche nach Attributen oder Objekten zu beschleunigen, kann es sinnvoll sein, mehr als nur eine CPU für das Indizieren zu verwenden. Hier müssen Sie testen, ob eine weitere CPU für Sie Vorteile bringt.
- Mit *database mdb* beginnt nicht nur der Datenbankteil, sondern Sie legen auch den Datenbanktyp fest. Seit einiger Zeit wird nur noch der Datenbanktyp *mdb* empfohlen, alle anderen alten Datenbanktypen werden in Zukunft nicht mehr unterstützt. Eine Umstellung ist möglich, aber nicht einfach durch Ersetzung des Wertes in der Konfigurationsdatei. Dann müssen Sie die folgenden Schritte ausführen:
  - Die Datenbank als LDIF sichern.
  - Den Server-Dienst stoppen.

  - Die Datenbankdateien löschen.
  - Die Konfiguration anpassen.
  - Den Server-Dienst neu starten.
  - Die LDIF-Datei wieder einspielen.
- Der *suffix* legt die oberste Ebene Ihres LDAP-Baums fest.
- Beim *rootDN* handelt es sich um den Hauptadministrator, der immer alle Rechte hat und nie auf irgendeine Art und Weise beschränkt werden kann. Der Benutzer wird nicht in der Datenbank angelegt.
- Das *rootpw* ist das Passwort für den *rootDN*. Halten Sie dieses Passwort immer unter Verschluss. Mit diesem Passwort können sämtliche Änderungen am LDAP-Baum vorgenommen werden. In der Beispieldatei ist das Passwort im Klartext eingetragen. Für ein Produktivsystem ist das keine gute Idee; hier sollten Sie das Passwort eintragen, das Sie mit dem Kommando `slappasswd` generiert haben. Rufen Sie dazu das Kommando `slappasswd` ohne weitere Parameter auf. Geben Sie anschließend das gewünschte Passwort zweimal ein und kopieren Sie dann den Hashwert in die Datei slapd.conf, indem Sie das Klartextpasswort durch den Hashwert ersetzen. Denken Sie immer daran: Der *rootDN* hat immer alle Rechte im Verzeichnisbaum, Sie können ihn auch nicht über ACL in seinen Rechten beschneiden. Wählen Sie deshalb hier ein besonders sicheres Passwort. Ich werde Ihnen hier im Buch zeigen, wie Sie für jede Aufgabe ein eigenes Konto mit einem eigenen Passwort erstellen. Dann benötigen Sie den *rootDN* nur noch für Konfigurationsänderungen.
- Mit dem Parameter *directory* legen Sie fest, in welchem Verzeichnis die Datenbankdateien abgelegt werden. Für jede Datenbank, die Sie mit dem LDAP-Server verwalten wollen, müssen Sie ein eigenes Verzeichnis angeben.
- Über den Parameter *index* werden die Indexdatenbanken für diese LDAP-Datenbank festgelegt. Später werden wir noch näher auf die Indexeinträge eingehen.
- *Lastmod on* speichert die Zeit der letzten Änderung und das dazugehörige Objekt in der Datenbank.
- Am Schluss folgen die ACLs für die Zugriffe. Auf die ACLs gehen wir in Kapitel 9, «Berechtigungen mit ACLs», näher ein.

### 3.1.2 Die dynamische Konfiguration

Bei der dynamischen Konfiguration werden alle Einstellungen des LDAP-Servers in einer eigenen Datenbank im LDAP abgelegt: Sowohl die Grundkonfiguration, die Erweiterung des Funktionsumfang durch Overlays als auch die ACLs müssen Sie mittels LDIF-Files in die Datenbank einspielen. Kommentare können Sie in dieser Datenbank nicht ablegen, sodass Sie sämtliche Änderungen und Einstellungen extern dokumentieren müssen.

Der große Vorteil der dynamischen Konfiguration ist, dass Sie hier den LDAP-Server nach einer Änderung nicht neu starten müssen; die Änderung ist sofort wirksam, nachdem Sie die LDIF-Datei eingespielt haben. Auch Ihre ACLs müssen Sie dann immer über LDIF-Dateien anpassen. Wenn Sie mit grafischen Werkzeugen arbeiten wollen, dann achten Sie darauf, ob das Werkzeug die dynamische Konfiguration bearbeiten kann. Wir werden in Kapitel 6, «Grafische Werkzeuge», noch darauf zu sprechen kommen.

Wenn Sie mehrere LDAP-Server einsetzen und diese vielleicht auch noch an unterschiedlichen Standorten stehen und Sie häufig Änderungen an der Konfiguration oder den ACLs vornehmen müssen, dann ist die dynamische Konfiguration auf jeden Fall der bessere Weg.

## 3.2 Installation unter Debian

Wie bei allen Diensten müssen hier als Erstes die Pakete installiert werden. Sie benötigen für den LDAP-Server nur die Pakete slapd und ldap-utils.

Die Konfigurationsdatei für den OpenLDAP-Server unter Debian finden Sie im Verzeichnis /etc/ldap. In diesem Verzeichnis finden Sie die Einträge aus der Tabelle 3.1.

**Tabelle 3.1** Dateien und Verzeichnisse für die Konfiguration

| Name | Typ | Verwendung |
|---|---|---|
| sasl2 | Verzeichnis | sasl-Datenbanken |
| schema | Verzeichnis | alle Schemen-Files |
| slapd.d | Verzeichnis | für die dynamische Konfiguration |
| ldap.conf | Datei | Client-Konfiguration |

- sasl2
  Wenn Sie Passwörter mit SASL verschlüsseln wollen, wird eine zusätzliche Datenbank mit allen Benutzern und deren verschlüsseltem Passwort angelegt. Hier im Buch werden wir auf diese Art der Verschlüsselung von Passwörtern nicht eingehen, da Sie so immer zwei Datenbanken verwalten müssen. Hier im Buch werden wir Ihnen den Einsatz von Kerberos für die Verschlüsselung von Passwörtern erklären.
- schema
  In diesem Verzeichnis finden Sie alle Schemen-Dateien: einmal als ASCII-Text, um die Schemen in die statischen Konfiguration einbinden zu können, und einmal als LDIF-Datei für die dynamische Konfiguration.
- slapd.d
  In diesem Verzeichnis wird später die dynamische Konfiguration abgelegt. Nach der Installation der Pakete befindet sich in dem Verzeichnis die automatisch erstellte Konfiguration. Diese Konfiguration wird später gelöscht und durch die eigene überschrieben.
- ldap.conf
  Diese Datei wird für die Konfiguration der Client-Programme wie `ldapadd`, `ldapmodify` und für die Zertifikate für die TLS-Verschlüsselung benötigt.

### Die statische Konfiguration

Für die statische Konfiguration des Debian-Servers können Sie die slapd.conf aus Listing 3.1 verwenden. Erstellen Sie die Datei im Verzeichnis /etc/ldap. Bevor Sie die Umstellung auf die statische Konfiguration vornehmen, stoppen Sie den Dienst mit `systemctl`

## 3.4 Einspielen der ersten Objekte

Hier treffen sich die Wege beider Distributionen wieder, denn das Einspielen und Verwalten der Objekte unterscheidet sich zwischen den Distributionen nicht. Wenn Sie also Ihren LDAP-Server konfiguriert haben, ist es egal, welche Distribution Sie einsetzen. Nachdem Sie die LDIF-Datei erzeugt haben, spielen Sie die Änderungen mittels `ldapadd` in die Datenbank ein.

Für das Einspielen der ersten Objekte brauchen Sie zwei Dinge:

- das Kommando `ldapadd`, um die Objekte in die Datenbank einzuspielen.
- eine LDIF-Datei, in der die Objekte beschrieben sind, die Sie in die Datenbank einspielen wollen.

In Listing 3.22 sehen Sie die LDIF-Datei, in der sich die ersten Objekte befinden:

**Listing 3.22** Inhalt der ersten LDIF-Datei

```
dn: dc=example,dc=net
objectClass: domain
objectClass: dcObject
dc: example

dn: ou=users,dc=example,dc=net
ou: users
objectClass: top
objectClass: organizationalUnit

dn: ou=groups,dc=example,dc=net
ou: groups
objectClass: top
objectClass: organizationalUnit
```

Ohne eine Authentifizierung ist das Einspielen der Objekte in die Datenbank nicht möglich. Hier kennt OpenLDAP zwei verschiedene Möglichkeiten der Authentifizierung. Da wäre zum einen der *simple bind*, bei dem Sie sich mit einem im LDAP abgelegten Passwort authentifizieren (oder, wenn Sie den *rootDN* nutzen, dem Passwort aus der slapd.conf). Die andere Möglichkeit ist über ein SASL-Passwort, das entweder in der sasldb oder in Kerberos abgelegt ist. Am Anfang soll hier der *simple bind* verwendet werden, später im Buch werden wir noch auf die Verwendung von Kerberos für die sichere Authentifizierung eingehen.

In Listing 3.23 sehen Sie die Verwendung von `ldapadd`:

**Listing 3.23** Einspielen der Objekte

```
[root@ldapserver openldap]#  ldapadd -x -D "cn=admin,dc=example,dc=net" \
                             -W -f home.ldif
Enter LDAP Password:
adding new entry "dc=example,dc=net"
```

```
adding new entry "ou=users,dc=example,dc=net"

adding new entry "ou=groups,dc=example,dc=net"
```

Die im Kommando verwendeten Parameter haben dabei die folgende Bedeutung:

- *-x*
  Damit aktivieren Sie den *simple bind*.
- *-D "cn=admin,dc=exampe,dc=net"*
  Mit diesem Benutzer werden Sie die Objekte einspielen. Immer, wenn Sie Objekte in den LDAP-Baum einspielen wollen, müssen Sie darauf achten, dass der Benutzer auch Schreibrechte am LDAP hat. Im Moment ist das nur der *rootDN*, den Sie in der Datei slapd.conf eingetragen haben.
- *-W*
  Ein schreibender Zugriff auf den LDAP ist nur mit einem Benutzer möglich, der auch ein Passwort besitzt. Durch das große *-W* werden Sie nach dem Abschicken des Kommandos nach dem Passwort gefragt. Die Eingabe des Passworts wird nicht angezeigt. Sie können aber auch das kleine *-w*, gefolgt von dem Passwort im Klartext, verwenden. Denken Sie daran, dass das Passwort anschließend, zusammen mit dem Kommando, in Ihrer History steht. Sie sollten diese Art der Authentifizierung daher nicht auf der Kommandozeile verwenden, sondern höchstens zur Abarbeitung eines Kommandos in einem Skript. Aber hierfür werden Sie später im Buch auch noch eine Alternative mit Kerberos kennenlernen.
- *-f*
  Über diesen Parameter, gefolgt von einer LDIF-Datei, geben Sie den Pfad und den Dateinamen der LDIF-Datei mit den Objekten an.

**Tipp**

Sollten Sie sich beim Anlegen der LDIF-Datei vertippt haben, kommt es an der Stelle beim Anlegen des entsprechenden Objekts zu einer Fehlermeldung, und der Prozess wird abgebrochen. Nachdem Sie den Fehler behoben haben und das Kommando wieder genauso wie beim ersten Mal verwenden, wird das `ldapadd` wahrscheinlich feststellen, dass die ersten Objekte bereits existent sind. Sie müssen dann nicht alle bereits angelegten Objekte aus der LDIF-Datei entfernen oder auskommentieren, Sie können einfach den zusätzlichen Parameter *-c* einfügen. Der Parameter sorgt dafür, dass bereits existente Objekte übersprungen werden.

Um die gerade erstellten Objekte sehen zu können, verwenden Sie das Kommando `ldapsearch`. In Listing 3.24 sehen Sie die Abfrage:

**Listing 3.24** Auflisten aller Objekte

```
[root@ldapserver ~]# ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=net> (default) with scope subtree
# filter: (objectclass=*)
```

```
# requesting: ALL
#

# example.net
dn: dc=example,dc=net
objectClass: domain
objectClass: dcObject
dc: example

# users, example.net
dn: ou=users,dc=example,dc=net
ou: users
objectClass: top
objectClass: organizationalUnit

# groups, example.net
dn: ou=groups,dc=example,dc=net
ou: groups
objectClass: top
objectClass: organizationalUnit

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Die Anzeige wird hier im erweiterten LDIF-Format angezeigt. Wie Sie sehen, werden neben den Informationen zu den Objekten auch noch weitere Informationen und Kommentare angezeigt. Sie können diese Ausgabe mit bis zu dreimaliger Verwendung des Parameters *-L* anpassen. Ein *-L* sorgt für eine Ausgabe der Objekte im *LDIv1*-Format. Ein zweites *-LL* unterbindet zusätzlich alle Kommentare, und ein drittes *-LLL* unterbindet auch noch die Ausgabe des LDIF-Formats. In Listing 3.25 sehen Sie die verkürzte Ausgabe:

**Listing 3.25** Verkürzte Ausgabe

```
[root@ldapserver ~]# ldapsearch -x -LLL
dn: dc=example,dc=net
objectClass: domain
objectClass: dcObject
dc: example

dn: ou=users,dc=example,dc=net
ou: users
objectClass: top
objectClass: organizationalUnit

dn: ou=groups,dc=example,dc=net
ou: groups
objectClass: top
objectClass: organizationalUnit
```

Sie sehen hier, dass jetzt nur noch die Objekte mit ihren Attributen angezeigt werden. Außerdem fällt noch auf, dass bei beiden Auflistungen keine Benutzerauthentifizierung stattfindet. Solange Sie keine *Access Control Lists (ACL)* eingerichtet haben, können Informationen aus dem LDAP immer auch anonym abgefragt werden. Das soll im Verlauf des Buches noch verhindert werden. Dann bekommen Sie bei einer anonymen Anfrage keine Objekte mehr angezeigt, und ein Benutzer, der eine Anfrage mit dem eigenen Benutzernamen stellt, bekommt nur noch die Informationen angezeigt, an denen er das Leserecht besitzt. Hier noch mal der Hinweis: Der *rootDN* hat immer alle Rechte!

## 3.5 Erste Objekte

Um die Installation zu testen, sollen jetzt ein Testbenutzer und eine Gruppe im LDAP eingerichtet werden. Dazu erstellen Sie als Erstes eine LDIF-Datei mit den entsprechenden Parametern. In Listing 3.26 sehen Sie eine LDIF-Datei, mit der Sie eine Gruppe und zwei Benutzer in der vorher erstellten Struktur anlegen können:

**Listing 3.26** LDIF für die ersten Objekte

```
dn: cn=benutzer,ou=groups,dc=example,dc=net
objectClass: posixGroup
gidNumber: 10000
cn: benutzer

dn: cn=Stefan Kania,ou=users,dc=example,dc=net
objectClass: posixAccount
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
loginShell: /bin/bash
homeDirectory: /home/skania
uid: skania
cn: Stefan Kania
userPassword: {SSHA}qY8x5DiXfa6qlL800gE9dk7mT7vO5hqB
uidNumber: 10000
gidNumber: 10000
sn: Kania
givenName: Stefan

dn: cn=Andreas Ollenburg,ou=users,dc=example,dc=net
objectClass: posixAccount
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
loginShell: /bin/bash
homeDirectory: /home/aollenburg
uid: aollenburg
cn: Andreas Ollenburg
userPassword: {SSHA}rDEb2oR0CCWC4EKWnubcWE4JUx1Hpq1i
```

# 12 Replikation des OpenLDAP-Baums

Bislang haben wir uns eine Umgebung mit einem einzelnen LDAP-Server angeschaut. Das mag in einigen Situation eventuell genügen. Aber ...

- ... was machen Sie, wenn der LDAP-Server ausfällt? Wie kritisch ist der jeweilige LDAP-Dienst? Können Sie sich die Zeit bis zu dessen Wiederherstellung leisten?
- ... wenn Sie verschiedene Standorte haben: Sind die Netzwerkverbindungen schnell genug für die Zugriffe eines Clients oder einer Anwendung an einem entfernten Standort auf den lokalen LDAP-Server?
- ... ist ein LDAP-Server genug für die eventuell hohe Anzahl an Anfragen und Zugriffen?

## 12.1 Grundlagen zur Replikation

Sie sehen: Sie werden also in der Praxis stets mehr als einen LDAP-Server benötigen. Und damit sind wir beim Thema *Replikation*. Die einzelnen Kopien einer Datenbank auf den beteiligten Servern müssen identisch gehalten werden. Wir wollen uns in diesem Kapitel die Mechanismen und die einzelnen Replikationsmodelle näher anschauen und umsetzen.

### 12.1.1 Change Sequence Number

Zunächst benötigen Sie ein Kriterium, um festzustellen, dass sich etwas geändert hat. An dieser Stelle sind *Zeitstempel* das Mittel der Wahl. Jeder Teilnehmer einer Replikation besitzt einen Zeitstempel der letzten Änderung an seiner lokalen Kopie. Nehmen die Teilnehmer Kontakt auf, dann vergleichen sie ihre lokalen Zeitstempel. Ist einer der Teilnehmer aktueller als die anderen, dann teilt er seine Neuigkeiten den anderen Teilnehmern mit. Nehmen wir also zwei Server *hostA* und *hostB*. Die beiden haben sich um 13:00 Uhr das letzte Mal untereinander aktualisiert. Um 13:03 Uhr wird eine Änderung an den Daten auf *hostB* vorgenommen. Um 13:05 Uhr nehmen die beiden Server wieder Kontakt auf, und *hostA* stellt fest, dass seine Daten veraltet sind. Er bekommt dann von *hostB* die aktuellen Informationen.

Gut, ganz so simpel ist das mit der Synchronisation zwar nicht. Hier soll erstmal nur das Prinzip der *Zeitstempel* an sich beleuchtet werden. An die Details arbeiten wir uns langsam

heran. Die Zeitstempel haben noch mehr Informationen. Der genaue Aufbau sieht wie folgt aus:

*YYYYmmddHHMMSSz.uuuuuu#<Zähler>#<ServerID>#000000*

Die einzelnen Parameter dieser als *Change Sequence Number* (*CSN*) bezeichneten Zeitstempel wurden in Tabelle 10.2 schon einmal detailliert aufgelistet.

Die *CSN* gibt also an, die wievielte Änderung zu einem bestimmten Zeitpunkt auf einem Server aktuell ist. Wir werden im Verlauf des Kapitels noch sehen, an welchen Stellen solche `CSNs` in der Datenbank auftreten und wie sie bei den einzelnen Replikationsszenarien zum Einsatz kommen. Am Ende des Kapitels finden Sie dann ein praktisches Beispiel, wie Ihnen die *CSN* beim Troubleshooting helfen kann.

### 12.1.2 Zeitsynchronisation

Wie Sie am *CSN* sehen können, ist es kritisch, dass die beteiligten Server einer Replikation eine einheitliche Zeit besitzen. Es ist eine der Achillesfersen der Replikation. Das muss nicht zwingend die Realzeit sein. Die Server müssen sich nur einig sein, wie spät es **jetzt** gerade ist. Für das Monitoring, eventuelles Auditing sowie das Troubleshooting ist es durchaus sehr sinnvoll, die Serverzeit möglichst identisch mit der tatsächlichen Zeit zu halten. Daher wollen wir uns an dieser Stelle die Zeit nehmen, uns damit zu beschäftigen, wie Ihre Server eine einheitliche und mit der Realzeit identische Zeit bekommen.

#### Problemsituation

In Situationen mit mehreren beschreibbaren Kopien einer LDAP-Datenbank (siehe dazu Abschnitt «Serverrollen») kommen Sie bei fehlender Zeitsynchronisation schnell in eine der beiden folgenden Situationen. In beiden Beispielen nehmen wir an, alle Server (bis auf einen) sind sich einig darüber, das es **jetzt** 12:00 Uhr ist. Der unsynchronisierte Server hat eine Zeitabweichung von 60 Minuten. An diesem Server (nennen wir ihn *Host A*) wurde der Nachname eines Benutzerobjekts auf *Mayer* geändert. An einem anderen Server (*Host B*) wurde der Nachname auf *Meier* geändert.

- Die Uhr auf *Host A* ist in der Zukunft.
  Für *Host A* ist es also schon 13:00 Uhr, und die Änderung findet in diesem Moment zunächst in seiner Kopie der Datenbank statt. Diese Änderung wird daher mit diesem Zeitstempel zu allen anderen Servern repliziert. Eine halbe Stunde später fällt auf, dass der Nachname falsch geschrieben wurde und auf *Meier* geändert werden soll. Diesmal landet die Änderung auf *Host B*. Dieser verwirft die Änderung allerdings, da er für dieses Attribut einen Zeitstempel von 13:00 Uhr in seiner Datenbank vorfindet, es seiner Zeit nach aber erst 12:30 Uhr ist. Die Änderung kann also frühestens nach weiteren 30 Minuten durchgeführt werden.
- Die Uhr auf *Host A* ist in der Vergangenheit.
  Die Uhr auf *Host A* steht also auf 11:00 Uhr. Diesmal «verliert» *Host A*. Findet die Änderung zunächst auf *Host B* statt (um 12:00 Uhr dessen Zeit), übernimmt *Host A* diese Änderung in seiner Datenbank mit dem entsprechenden Zeitstempel. Landet die nächste Änderung innerhalb der kommenden 59 Minuten auf *Host A*, so verwirft er diese, da es für ihn ja noch nicht 12:00 Uhr ist.

### Network Time Protocol (NTP)

Sie müssen also dafür sorgen, dass Sie die Zeit zwischen den Servern synchronisieren. Dafür hat sich in der Praxis das `Network Time Protocol` (*NTP*) durchgesetzt.

Dabei gibt es mehrere Abstufungen der Genauigkeit, *Stratum* genannt. Zeitanbieter vom Stratum 0 (**die** Zeit) sind dabei die Atom- oder Funkuhren. Diese liefern ihre Zeit an die genauesten NTP-Server vom Stratum 1. Diese haben eine maximale Abweichung von maximal $10^1$ Mikrosekunden zur realen Zeit der Atom- und Funkuhren. Stratum-1-Server geben ihre Zeit an Server vom Stratum 2 weiter. Diese dürfen wiederum eine maximale Zeitdifferenz von $10^2$ Mikrosekunden von ihren Anbietern aufweisen usw. In Bild 12.1 ist die Hierarchie hinsichtlich des Stratums nochmal dargestellt:

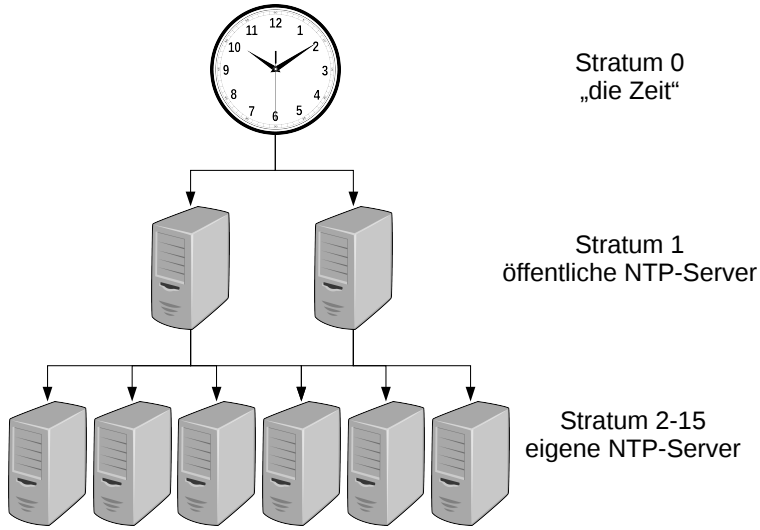


**Bild 12.1** NTP-Stratum

Das maximale *Stratum* ist 16. Dieses Stratum ist die ungenaueste Stufe der Zeitsynchronisation. Daher wird dieses Stratum auch als *unsynchronized* bezeichnet. In diesem Stratum beginnen alle NTP-Clients beim Start des NTP-Dienstes. Nach dem Start versucht der NTP-Dienst nun, sich an den NTP-Server mit dem geringsten Stratum anzunähern. Gelingt es ihm, sich lange genug in der benötigten Genauigkeit zu seinem NTP-Server zu bewegen, darf er das entsprechend nächste Stratum annehmen.

Überlegen Sie sich für Ihre Umgebung, welche NTP-Hierarchie Sie implementieren. Um die Zeitserver im Internet nicht über die Maßen zu belasten, richten Sie sich ein oder zwei Server ein, die sich die Zeit über *NTP* von außen holen. Diese nutzen Sie dann für die internen Server als NTP-Zeitquelle. Alternativ können Sie natürlich auch Funkuhren in Ihrem Netzwerk einrichten, die das *DCF77*-Funksignal der Atomuhr der *Physikalisch-Technischen Bundesanstalt* in Braunschweig über den Langwellensender in Mainflingen empfangen und per NTP angesprochen werden können. Je nach Größe oder Sicherheitsrichtlinien fügen Sie noch weitere Ebenen in Ihrer Synchronisationshierarchie ein. Wie Sie aus den Mechanismen von NTP ersehen, ist auch ein Stratum 3 oder 4 sehr nah an der realen Zeit und

genügt in den meisten Umgebungen, solange Sie keine extrem zeitkritischen Anwendungen besitzen.

Grau ist alle Theorie. Jetzt wollen wir uns die grundsätzliche NTP-Einrichtung in der Praxis anschauen. Unter Debian installieren Sie die Pakete ntp sowie ntpdate und konfigurieren den Dienst über die Datei /etc/ntp.conf. Dort können Sie mit dem Parameter *server* die von Ihnen gewählten NTP-Server oder mit dem Parameter *pool* einen *NTP-Server-Pool* wie *pool.ntp.org* eintragen. Falls die Zeit Ihres Servers zu sehr von der aktuellen Zeit abweicht, können Sie die Zeit vor Start des NTP-Dienstes mit `ntpdate <Server>` direkt mit der Zeit eines externen Servers abgleichen. Anschließend aktivieren und starten Sie den `ntp`-Dienst. Befehle wie `ntptrace` und `ntpq` geben Ihnen Hilfestellungen bei der Kontrolle des Dienstes. Listing 12.1 zeigt Ihnen die Einrichtung, den Start und die Kontrolle des NTP-Dienstes:

**Listing 12.1** Einrichtung von NTP unter Debian

```
ldap01:~# apt-get install ntp ntpdate
Reading package lists... Done
Building dependency tree
Reading state information... Done
[...]

ldap01:~# ntpdate pool.ntp.org
12 Jan 17:12:43 ntpdate[11245]: adjust time server 195.50.171.101 offset
     -0.000603 sec

ldap01:~# grep ^pool /etc/ntp.conf
pool 0.debian.pool.ntp.org iburst
pool 1.debian.pool.ntp.org iburst
pool 2.debian.pool.ntp.org iburst
pool 3.debian.pool.ntp.org iburst

ldap01:~# systemctl enable ntp

ldap01:~# systemctl start ntp

ldap01:~# ntptrace -m 1
localhost: stratum 2, offset -0.000346, synch distance 0.018630

ldap01:~# ntpq -p
remote           refid          st t when poll reach delay offset jitter
==============================================================================
 0.debian.pool.n .POOL.          16 p    --   64    0 0.000   0.000  0.001
 1.debian.pool.n .POOL.          16 p    --   64    0 0.000   0.000  0.001
 2.debian.pool.n .POOL.          16 p    --   64    0 0.000   0.000  0.001
 3.debian.pool.n .POOL.          16 p    --   64    0 0.000   0.000  0.001
 ns1.blazing.de  213.172.96.14    2 u   13   64    3 27.200  0.413  0.427
*ntp1.rz.uni-kon .GPS.            1 u    9   64    3 33.350  0.212  6.412
+sv1.ggsrv.de    205.46.178.169  2 u    6   64    3 30.484 -0.318  2.106
#ns2.customer-re 192.53.103.108  2 u    7   64    3 36.883  5.067  1.315
+217.79.179.106  131.188.3.222   2 u   10   64    3 28.159 -0.462  2.742
```

```
+golf.zq1.de     205.46.178.169  2 u    9   64    3 30.367  0.527  2.704
#162.159.200.123 10.50.8.4       3 u    4   64    3 70.275 -8.465  6.673
+ernie.gerger-ne 213.172.96.14   2 u    3   64    3 27.364 -0.266  8.351
#ns1.your-ns.de  129.70.132.32   3 u    2   64    3 33.933  1.321  8.727
+a.chl.la        131.188.3.222   2 u    7   64    3 30.121  0.310  9.827
+janetzki.eu     166.160.177.106 3 u    7   64    3 28.956  0.912  9.733
+aurum.valaphee. 131.188.3.220   2 u    5   64    3 31.154 -1.928  9.335
+ntp2.m-online.n 212.18.1.106    2 u    4   64    3 32.749 -0.133  9.568
#radio-sunshine. 205.46.178.169  2 u    2   64    3 39.361 -7.425 10.570
+25000-021.cloud 131.188.3.221   2 u    2   64    3 32.692 -1.201  2.731
+srv01.spectre-n 131.188.3.222   2 u    2   64    3 33.050 -1.276  8.666
```

(Nach dem Start kann es einige Minuten dauern, bis der Server sein Stratum erreicht hat, und wird zunächst im Stratum 16 bleiben. Geben Sie ihm ... Zeit.)

CentOS benutzt in der aktuellen Version das Paket *chrony*. Die Konfiguration erfolgt über die Datei /etc/chrony.conf ebenfalls über *server*- und/oder *pool*-Einträge. Für die Kontrolle des Dienstes können Sie das Tool `chronyc` nutzen. Listing 12.2 liefert Ihnen die Befehle, die Sie zum Einrichten, Starten und zur Kontrolle des Dienstes benötigen:

**Listing 12.2** Einrichtung von Chrony unter CentOS

```
ldap02:~# yum install chrony -y
Last metadata expiration check: 3:31:19 ago on Sun Jan 12 12:58:39 2020.
Dependencies resolved.
[...]

ldap02:~# grep ^pool chrony.conf
pool 2.centos.pool.ntp.org iburst

ldap02:~# systemctl enable chronyd

ldap02:~# systemctl start chronyd

ldap02:~# chronyc tracking
Reference ID    : D94FB36A (mail.unkn0wn.ch)
Stratum         : 3
Ref time (UTC)  : Sun Jan 12 16:38:58 2020
System time     : 0.000057110 seconds slow of NTP time
Last offset     : +0.000060181 seconds
RMS offset      : 0.029195314 seconds
Frequency       : 0.096 ppm fast
Residual freq   : -0.002 ppm
Skew            : 0.071 ppm
Root delay      : 0.041319627 seconds
Root dispersion : 0.010151371 seconds
Update interval : 1027.4 seconds
Leap status     : Normal

ldap02:~# chronyc sources
210 Number of sources = 4
```

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ ntp-gps.beuth-hochschule>     2  10   377   268  -1247us[-1186us] +/- 42ms
^* mail.unkn0wn.ch               2  10   377   107   +855us[ +916us] +/- 45ms
^+ s2.eideo.de                   3  10   377   26m  +3323us[+3391us] +/- 82ms
^+ twilight.domainfactory.de     3  10   377   625   -351us[ -288us] +/- 57ms
```

**Tipp**

Für NTP und Chrony existieren zusätzliche Sicherheitsmechanismen und Konfigurationsmöglichkeiten. Die würden an dieser Stelle den Rahmen sprengen. Konsultieren Sie dazu u. a. die Manuals der einzelnen Befehle und Dateien.

Sie sollten nach der Einrichtung auch auf den korrekten Betrieb Ihrer NTP-Synchronisation achten und diese möglichst überwachen (zum Beispiel mit *Munin*). Mehr zum Thema Überwachung finden Sie in Kapitel 14, «Monitoring mit Munin».

### 12.1.3 Serverrollen

Betrachten wir noch die möglichen Rollen, die ein LDAP-Server bei einer Replikation spielen kann: *Provider* und *Consumer* – der eine bietet Daten an, der andere nutzt diese. Bild 12.2 stellt die beiden Serverrollen noch einmal dar:

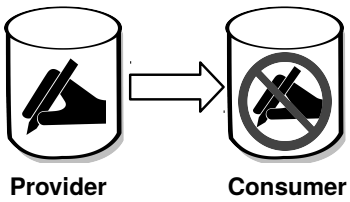


**Bild 12.2** Serverrollen bei der Replikation

- *Provider*
  Der *Provider* besitzt entweder eine beschreibbare oder schreibgeschützte Replik der Datenbank. Er repliziert seine Datenbank zu einem oder mehreren *Consumern*.
- *Consumer*
  Der *Consumer* besitzt eine schreibgeschützte Kopie der Datenbank seines *Providers*. Er kann allerdings wieder als *Provider* für weitere *Consumer* auftreten.

In Bild 12.3 wird eine Situation mit mehreren Servern dargestellt:

# 13 OpenLDAP mit Kerberos

Bis zu diesem Punkt haben Sie Ihren LDAP-Server schon soweit konfiguriert, dass Sie alle Zugriffe über TLS abgesichert haben. Wenn Sie Ihren Anwendern eine zentrale Anmeldung über LDAP bereitstellen wollen, damit sie sich in Ihrem Netzwerk an verschiedenen Diensten authentifizieren können, ist es sinnvoll, über ein Single Sign-on mithilfe von Kerberos nachzudenken. Denn alleine mit LDAP ist ein Single Sign-on nicht möglich. Ohne Kerberos müssen sich die Anwender bei jedem authentifizierten Zugriff auf eine Anwendung erneut mit ihrem Benutzernamen und Passwort authentifizieren. Durch den Einsatz von Kerberos melden sich die Anwender nur noch einmal im Netz an, und die Authentifizierung an Diensten wird über Kerberos geregelt. Das setzt voraus, dass Sie auch alle Anwendungen an den Kerberos anbinden. Die Dienste müssen «kerberisiert» werden.

Sind alle Voraussetzung erfüllt, meldet sich ein Anwender einmal an seinem System an und kann anschließend ohne weitere Authentifizierungen alle Anwendungen nutzen. Dazu benötigen die Dienste ebenfalls eine Kerberos-Konfiguration.

Ein weiterer Vorteil hinsichtlich der Sicherheit ist der, dass sich nicht nur die Anwender authentifizieren müssen, sondern auch alle Hosts und Services authentifizieren sich gegen den Kerberos.

*Kerberos* wurde 1978 am *Massachusetts Institute of Technology* (MIT) entwickelt und wird für die Authentifizierung von Netzwerkdiensten, Hosts und Services genutzt. Die aktuelle Version von Kerberos ist Version 5.

In den meisten Distributionen wird der Kerberos des Massachusetts Institute of Technology eingesetzt. Diese Version wurde aber aufgrund der verwendeten Verschlüsselungsalgorithmen unter die strengen Exportregeln der USA gestellt. Aus diesem Grund wurden verschiedene andere Implementierungen außerhalb der USA entwickelt. Die am meisten verwendete Kerberos-Implementierung ist die *Heimdal*-Implementierung aus Schweden. Als im Jahr 2000 die strenge Reglementierung für den MIT-Kerberos aufgehoben wurde, haben einige Distributionen den Heimdal-Kerberos entfernt und verwenden heute nur noch den MIT-Kerberos. Heute wird der Heimdal-Kerberos nur noch von Debian-basierten Distributionen bereitgestellt. Den MIT-Kerberos finden Sie aber auf allen Distributionen, und aus diesem Grund werden wir uns hier im Buch ausschließlich mit der MIT-Implementierung beschäftigen.

**Hinweis**

Wenn Sie einen Kerberos-Server mit einer der beiden Implementierungen einrichten, können Sie für den LDAP-Client sowohl die Kerberos-Client-Pakete des MIT-Kerberos einsetzen oder die Pakete des Heimdal-Clients. Nur die Syntax der Kommandos ist bei den Paketen unterschiedlich.

Die Verwendung von Kerberos bietet Ihnen eine einheitliche und sichere Authentifizierungsmethode. Bei einer Authentifizierung in einem Netzwerk, das mit Kerberos arbeitet, wird die Authentifizierung von einer *vertrauenswürdigen dritten «Partei» (Trusted Third Party)* übernommen. Hierbei handelt es sich dann um den *Kerberos-Dienst*.

In Bild 13.1 sehen Sie den Vorgang einer Authentifizierung:

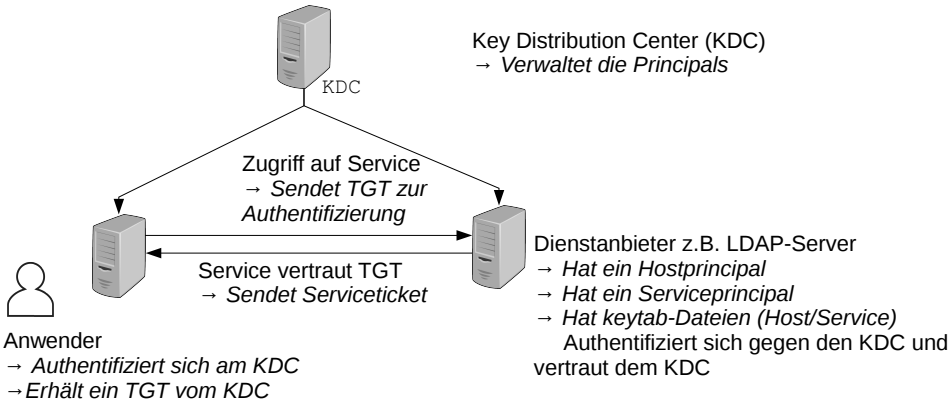


**Bild 13.1** Authentifizierungsmodell von Kerberos

Der Service und der Host, auf dem der Service läuft, authentifizieren sich mit ihrem Host- und Service-Principal. Für die Authentifizierung verwenden sie ihre keytab-Datei, in der das verschlüsselte Passwort liegt.

Der Anwender authentifiziert sich beim KDC durch die Eingabe seines Passworts bei der Anmeldung. War die Anmeldung erfolgreich, erhält er ein Ticket (Ticket Granting Ticket, TGT), mit dem er sich jetzt direkt bei einem Service anmelden kann.

Der Anwender greift auf einen Service zu und überträgt sein TGT an den Host, der den Service bereitstellt. Der Service prüft das TGT. Da der Service dem KDC traut und das TGT vom KDC ausgestellt wurde, ist der Anwender vertrauenswürdig. Jetzt bekommt der Anwender Zugriff auf den Dienst, und der Service sendet dem Anwender ein Service-Ticket. Bei jedem weiteren Zugriff auf den Service verwendet der Anwender jetzt automatisch das Service-Ticket. Der gesamte Vorgang ist für den Anwender transparent.

Der *Kerberos-Server* wird nur für die Authentifizierung sorgen, er kann jedoch nicht zur Autorisierung oder zum Verwalten der Benutzerkonten eingesetzt werden. Für die Aufgaben der Kontenverwaltung kann entweder eine lokale Anmeldung genutzt werden oder Sie richten eine zentrale Benutzerverwaltung mit LDAP ein.

Mithilfe von Kerberos können Sie in Ihrem Netzwerk ein *Single Sign-on (sso)* realisieren. Jeder Benutzer gibt sein Passwort nur einmal ein und erhält dann ein *Ticket*. Mit diesem

Ticket kann er sich dann automatisch bei allen Diensten im Netzwerk anmelden, die Kerberos unterstützen.

Damit Sie später nicht auf jedem Kerberos-Client die Namen der Kerberos-Server einrichten müssen, ist es sinnvoll, einen DNS-Server im Netzwerk zu betreiben. Der Vorteil ist der, dass Sie dann die Dienste, die der Kerberos-Server bereitstellt, über DNS *SRV-Records* anbieten können. Ohne die SRV-Records ist es notwendig, dass Sie die Kerberos-Server auf jedem Client konfigurieren. Das würde dazu führen, dass wenn Sie einen neuen Kerberos-Server in Ihr Netzwerk einbinden, eine Anpassung für jeden Client notwendig wäre.

Beim Einsatz von Kerberos werden verschiedene Begriffe verwendet. Deshalb erhalten Sie hier eine Übersicht über die Begriffe und deren Bedeutung:

- **Realm**
  Bei *Realm* handelt es sich um den administrativen Bereich, oft auch *Authentifizierungszone* genannt, für den Kerberos-Server. Hier wird häufig der Name der DNS-Domäne in Großbuchstaben verwendet.

**Hinweis**

Achten Sie bei der Verwendung des Realms immer darauf, dass Sie diesen auch groß schreiben. An vielen Stellen kommt es sonst zu Fehlern.

- **Principal**
  Der *Principal* wird von Kerberos für die Authentifizierung benötigt. Der Principal setzt sich aus einer oder mehreren Komponenten und dem Realm zusammen. Ein Principal für einen Benutzer hat nur eine Komponente, die mit dem @-Zeichen an den Realm gebunden wird (zum Beispiel *stefan@EXAMPLE.NET*). Für die Authentifizierung eines Service- oder Hosttickets werden dagegen zwei Komponenten verwendet, zum Beispiel *host/kerberos-01.example.net@EXAMPLE.NET* für einen Host in Ihrem Netzwerk. Für die Services gibt es die folgenden Komponenten:
  - *ldap* für den LDAP-Service
  - *HTTP* für Webservices
  - *nfs*, wenn Sie NFS-Server einbinden wollen
  - *imap* für Ihre IMAP-Server

  Einige Services wie zum Beispiel SSH nutzen den Host-Principal des Systems.
- **Ticket**
  Das *Ticket* wird für die Anmeldung an einem Host oder für einen Dienst benötigt. Das *Ticket* kann auch als virtueller Fahrschein betrachtet werden, ohne den der Dienst nicht benutzt werden kann.
- **Ticket Granting Ticket (TGT)**
  Das TGT erhält der Client vom *Ticket Granting Server* (TGS). Mit diesem TGT kann der Client, ohne weitere Authentifizierungen durch den KDC durchführen zu müssen, die Tickets für weitere Dienste erhalten. Wenn der Service ebenfalls Kerberos für die Authentifizierung nutzt, kann der Service das TGT des Benutzers bei der Anmeldung selbstständig prüfen, da er dem KDC vertraut und somit auch dem TGT des Benutzers.

- **Authentication Service (AS)**
  Der AS beantwortet die Anfragen der Clients und erzeugt die zufälligen Sitzungsschlüssel sowie die Kerberos-Tickets.
- **Ticket Granting Service (TGS)**
  Der TGS vergibt das *Ticket Granting Ticket (TGT)* an die Clients. Mithilfe dieses Tickets kann dann der Client Tickets für andere Dienste beziehen.
- **Key Distribution Center (KDC)**
  Hierbei handelt es sich um eine Datenbank, in der alle Principals mit ihrem Passwort abgelegt sind. Der TGS ist ein Teil des KDC.

## 13.1 Funktionsweise von Kerberos

Bevor ein Dienst oder ein Host Kerberos für die Authentifizierung von Benutzern über Tickets nutzen kann, muss der entsprechende Dienst *kerberized* werden. Denn nur dann kann der Dienst das Ticket des Anwenders gegen den Kerberos-Server überprüfen.

Wenn sich ein Anwender am System anmeldet, erhält er vom AS ein TGT. Will nun der Anwender einen Dienst nutzen oder auf einen anderen Host zugreifen, der *kerberized* ist, holt sich der Client mit dem TGT ein Ticket des entsprechenden Hosts oder Diensts. Der Client schickt dann das Ticket an den Host oder Dienst, dieser kann nun das Ticket gegen den Kerberos-Server prüfen und daraufhin den Zugriff zulassen. Dieser Vorgang findet für den Anwender absolut transparent statt. Greift ein Benutzer erneut auf den Dienst zu, nutzt er das Ticket, das er beim ersten Zugriff vom Service erhalten hat.

#### 13.1.0.1 Einstufiges Kerberos-Verfahren

Beim einstufigen Kerberos-Verfahren werden alle Authentifizierungen nur vom *AS* gegen die KDC-Datenbank durchgeführt. Jeder Zugriff von einem Client auf einen Dienst erfordert immer das Passwort des Benutzers. Ein *sso* ist mit diesem Verfahren nicht realisierbar, da kein TGT an den Client vergeben wird, mit dem er sich automatisch authentifizieren könnte. Dieses Verfahren wird heute kaum noch verwendet und soll deshalb hier nicht weiter besprochen werden.

#### 13.1.0.2 Zweistufiges Kerberos-Verfahren

Beim zweistufigen Kerberos-Verfahren kommt der TGS ins Spiel. Jeder Client, der sich mit seinem in der KDC-Datenbank eingetragenen Principal und dessen Passwort angemeldet hat, erhält ein TGT, das nur eine begrenzte Gültigkeitsdauer aufweist (meist acht bis zehn Stunden). Dieses Ticket wird im Ticket-Cache des Benutzers abgelegt. Sie finden die Datei, in der das Ticket gespeichert wird, im Verzeichnis /tmp. Die Datei hat dort den Namen krb5cc_<user-uid>. Immer, wenn der Anwender jetzt auf einen neuen Dienst oder Host zugreifen will, wird der Kerberos-Client im Hintergrund mithilfe des TGT ein Ticket für den Zugriff vom TGS anfordern und damit die Authentifizierung für den Anwender transparent durchführen. Damit ist ein echtes *sso* möglich.

**Wichtig**

Bevor wir jetzt zur Installation und Konfiguration des Kerberos-Servers kommen, noch ein wichtiger Punkt. Kerberos ist sehr zeitkritisch. Bei der Überprüfung der Tickets wird auch immer die Zeit der Erstellung überprüft. Aus diesem Grund ist es sinnvoll, dass Sie einen Zeitserver in Ihrem Netzwerk haben und alle Hosts in Ihrem Netzwerk sich die Systemzeit von diesem Zeitserver holen. Auch ein funktionsfähiger DNS-Server, der alle Hosts, sowohl die Server als auch die Clients, auflösen kann, muss in Ihrem Netzwerk laufen.

## 13.2 Installation und Konfiguration des Kerberos-Servers

Der Kerberos-Server ist nicht abhängig vom LDAP-Server. Ein Kerberos-Server kann auch ohne zentrale Benutzerverwaltung eingerichtet und genutzt werden. Ohne eine zentrale Benutzerverwaltung würden die Passwörter im Kerberos verwaltet, und die Benutzerkonten würden lokal auf den einzelnen Systemen eingerichtet. Um die Funktionsweise von Kerberos besser erklären zu können, werden wir im ersten Schritt einen Kerberos-Server ohne OpenLDAP einrichten und anschließend den OpenLDAP-Server für die zentrale Benutzerverwaltung zum Kerberos hinzufügen. Wir werden dabei die bestehende Kerberos-Datenbank in den OpenLDAP importieren, sodass bestehende Principals in den LDAP übernommen werden können. Das betrifft besonders die Host- und Service-Principals. Bei den Benutzer-Principals gibt es da bestimmte Einschränkungen, auf die wir an der entsprechenden Stelle zu sprechen kommen.

Um den ersten Kerberos-Server einzurichten, installieren Sie als Erstes die benötigten Pakete. Für Debian installieren Sie die Pakete wie gewohnt mit `apt-get`, wie in Listing 13.1 zu sehen:

**Listing 13.1** Installation der benötigten Debian-Pakete

```
root@kerberos-01:~# apt-get install krb5-admin-server krb5-kdc \
                    krb5-user krb5-kdc-ldap
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Statusinformationen werden eingelesen... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  krb5-config
Vorgeschlagene Pakete:
  openbsd-inetd inet-superserver
Die folgenden NEUEN Pakete werden installiert:
  krb5-admin-server krb5-config krb5-kdc krb5-user krb5-kdc-ldap
```

Das Paket krb5-kdc-ldap wird hier schon mit installiert, um später die Kerberos-Datenbank in den LDAP verschieben zu können und den Kerberos-Server für die Verwendung von LDAP zu konfigurieren. Mehr dazu, wie Sie Kerberos in den LDAP integrieren können, fin-

den Sie in Abschnitt 13.8, «Kerberos im LDAP einbinden». Während der Installation werden die folgenden Parameter abgefragt, über die Sie sich im Vorfeld Gedanken machen sollten:

- **Der Realm**
  Mit dem Realm legen Sie den Administrationsbereich für den Kerberos fest. Hier tragen Sie den fqdn des Servers ein, der für den Realm verantwortlich ist. Meist wird hier der DNS-Name der Domäne verwendet. Hier im Buch arbeiten wir mit dem Realm EXAMPLE.NET. Wenn Sie für den Realm den DNS-Namen Ihrer Domäne verwenden, hat das den Vorteil, dass Sie alle Namen der Principals für Ihre Hosts den fqdn der Hosts zuordnen können. Besonders, wenn Sie für mehr als eine Domäne verantwortlich sind.
- **Der Administrationsserver**
  Auf dem Administrationsserver werden die Principals verwaltet. Diesen Server darf es für jeden Realm nur einmal geben. Auch hier geben Sie den fqdn des Servers an.
- **Der KDC-Server**
  Das ist der Server, auf dem sich die Datenbank mit den Principals befindet. Der Server mit der Datenbank kann ein anderer Server sein als der Server, von dem aus die Datenbank verwaltet wird. Aus diesem Grund wird während der Einrichtung nach dem Admin-Server und dem KDC-Server gefragt. Hier im Buch sollen beide Dienste auf derselben Maschine eingerichtet werden. Das ist in den meisten Fällen auch die einfachste und beste Vorgehensweise. Die Trennung findet deshalb statt, da Sie später noch zusätzliche KDCs einrichten können, die dann nur die Funktion des KDCs übernehmen.

Bei CentOS installieren Sie die Pakete aus Listing 13.2:

**Listing 13.2** Installation der benötigten CentOS-Pakete

```
[root@kerberos-01 ~]# yum install krb5-libs krb5-server \
                      krb5-workstation krb5-server-ldap
...
```

Hier werden Sie während der Installation der Pakete nicht nach Parametern für die Datei /etc/krb5.conf gefragt.

### 13.2.1 Konfiguration des ersten Kerberos-Servers

Nachdem Sie die Pakete installiert haben, können Sie jetzt mit der Konfiguration beginnen. Für die Konfiguration des Kerberos-Servers werden die Datei kdc.conf und die Datei /etc/krb5.conf verwendet. Die Datei kdc.conf finden Sie bei Debian im Verzeichnis /etc/krb5kdc, bei CentOS im Verzeichnis /var/lib/kerberos/krb5kdc.

#### Konfiguration der Datei /etc/krb5.conf

In dieser Datei finden Sie die Informationen für den Kerberos-Realm. An dieser Stelle legen Sie auch fest, auf welchen Systemen der KDC läuft und auf welchem System sich der Admin-Server für die Verwaltung der Datenbank befindet. Für einen Kerberos-Realm kann es mehrere KDCs geben, aber immer nur einen Admin-Server. Immer wenn Sie einen neuen KDC für einen Realm installieren, ist es (im Moment) noch notwendig, dass Sie auf allen KDCs und Client die krb5.conf anpassen.

Damit es nicht notwendig wird, dass Sie auf allen Clients bei jeder Änderung der KDCs die krb5.conf an alle Clients anpassen, werden wir Ihnen im Verlauf dieses Kapitel zeigen, wie Sie die benötigten Informationen über einen DNS-Server bei den Clients bekannt machen können.

Während der Installation der Pakete wird bereits eine Datei krb5.conf angelegt, in dieser befinden sich aber sehr viele nicht benötigte Einträge. Aus diesem Grund macht es Sinn, dass Sie die Datei immer auf Ihre Umgebung anpassen und alle nicht benötigten Einträge aus der Datei entfernen. In Listing 13.3 sehen Sie den Aufbau der Datei für die Beispielumgebung hier im Buch:

**Listing 13.3** Alle benötigten Einträge in der krb5.conf

```
[libdefaults]
        default_realm = EXAMPLE.NET
   [realms]
        EXAMPLE.NET = {
                kdc = kerberos-01.example.net
                admin_server = kerberos-01.example.net
        }

[domain_realm]
        .example.net = EXAMPLE.NET

[logging]
        kdc = FILE:/var/log/kdc.log
        admin_server = FILE:/var/log/kadmind.log
        default = SYSLOG:NOTICE:DAEMON
```

Die einzelnen Abschnitte und Parameter haben dabei folgende Bedeutungen:

- *[libdefaults]*
  In diesem Abschnitt befinden sich die Standardeinstellungen für die Kerberos-Libraries. An dieser Stelle können Sie einige Einstellungen vornehmen, die das Verhalten des Kerberos-Servers verändern. Eine vollständige Übersicht über die möglichen Parameter finden Sie unter *https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html#libdefaults*.
- *default_realm = EXAMPLE.NET*
  Dieser Parameter gibt den Standard-Realm für die Client-Server-Kommunikation an. Achten Sie darauf, dass Sie den Realm hier unbedingt in Großbuchstaben schreiben.
- *[realms]*
  In diesem Abschnitt werden alle *Realms* verwaltet, die ein Client kennt, und es wird definiert, auf welchen Hosts sich die entsprechenden Server befinden.
- *kdc = kerberos-01.example.net*
  Das ist der fqdn des KDC. An dieser Stelle können später auch noch weitere KDCs eingetragen werden.
- *admin_server = kerberos-01.example.net*
  Da der Admin-Server und der KDC auf verschiedenen Maschinen laufen können, ist es wichtig, dass hier der fqdn des Admin-Servers angegeben wird. In den meisten Fällen werden Sie den KDC und den Admin-Server auf derselben Maschine installieren. Wir

werden hier im Buch auch beide Dienste auf demselben Host installieren und konfigurieren.

- *[domain_realm]*
  Hier wird die Verbindung des Kerberos-Realms und des Domainnamens hergestellt. Dieser Eintrag wird von Programmen benötigt, die feststellen müssen, zu welchem Realm ein Host gehört. Hier wird der fqdn der DNS-Domain verwendet.
- *.example.com = EXAMPLE.NET*
  Da die Domain hier im Buch genauso heißt wie der Realm, steht auf beiden Seiten der Zuweisung derselbe Name. Wichtig ist, dass der Realm auch an dieser Stelle komplett großgeschrieben wird.
- *[logging]*
  In diesem Abschnitt wird das Logging für den KDC und den Admin-Server festgelegt.
- *kdc = FILE:/var/log/kdc.log*
  Dieses ist die Log-Datei für den KDC. Sie können auch im Verzeichnis /var/log ein Unterverzeichnis anlegen und die Log-Dateien in dieses Unterverzeichnis legen. Wenn Sie mit einem Unterverzeichnis arbeiten, denken Sie daran, die Rechte des Verzeichnisses so anzupassen, dass der Benutzer, unter dem der KDC und der Admin-Server läuft, schreibenden Zugriff auf das Verzeichnis hat.
- *admin_server = FILE:/var/log/kadmind.log*
  Dieses ist die Log-Datei für den Admin-Server.
- *default = SYSLOG:NOTICE:DAEMON*
  An dieser Stelle können Sie das Verhalten des *Syslogd* bestimmen wie beispielsweise hier das Log-Level *NOTICE*.

**Wichtig**

Achten Sie darauf, dass die Namen des KDC und des Admin-Servers über DNS auflösbar sind.

## Konfiguration der Datei kdc.conf

Die Datei kdc.conf dient zur Konfiguration des KDCs und liegt bei Debian im Verzeichnis /etc/krb5kdc/. Bei CentOS finden Sie die Datei im Verzeichnis /var/lib/kerberos. Bei Debian wurde der Realm bereits während der Installation der Pakete abgefragt. Haben Sie dort schon den korrekten Realm angegeben, brauchen Sie an der Datei kaum Änderungen vorzunehmen. In der Datei kdc.conf finden Sie bereits Einträge für den von Ihnen angegebenen Realm. Haben Sie bei der Installation der Pakete keinen oder nicht den endgültigen Realm angegeben, können Sie jetzt die Datei anpassen. Auf einem Debian-System sieht die Datei so aus wie in Listing 13.4:

**Listing 13.4** Die Datei kdc.conf auf einem Debian-System

```
[kdcdefaults]
    kdc_ports = 750,88

[realms]
    EXAMPLE.NET = {
```

```
        database_name = /var/lib/krb5kdc/principal
        admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
        acl_file = /etc/krb5kdc/kadm5.acl
        key_stash_file = /etc/krb5kdc/stash
        kdc_ports = 750,88
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        master_key_type = des3-hmac-sha1
        #supported_enctypes = aes256-cts:normal aes128-cts:normal
        default_principal_flags = +preauth
    }
```

Bei CentOS hat die Datei den Inhalt wie in Listing 13.5 und muss von Ihnen auf jeden Fall angepasst werden:

**Listing 13.5** Die kdc.conf auf einen CentOS-System

```
kdcdefaults]
    kdc_ports = 88
    kdc_tcp_ports = 88
    spake_preauth_kdc_challenge = edwards25519

[realms]
EXAMPLE.NET = {
     #master_key_type = aes256-cts
     acl_file = /var/kerberos/krb5kdc/kadm5.acl
     dict_file = /usr/share/dict/words
     admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
     supported_enctypes = aes256-cts:normal \
     aes128-cts:normal des3-hmac-sha1:normal \
     arcfour-hmac:normal camellia256-cts:normal \
     camellia128-cts:normal
}
```

**Wichtig**

Die Pfade bei CentOS und Debian sind unterschiedlich. Setzen Sie auf jeden Fall die richtigen Pfade für Ihre Distribution!

Um Ihnen eine eventuelle Anpassung der Parameter einfach zu machen, finden Sie in folgender Aufzählung eine Erklärung der einzelnen Parameter:

- *[kdcdefaults]*
  In diesem Abschnitt werden alle Parameter verwaltet, die unabhängig von allen verwalteten *Realms* auf dem Server vom KDC nötig sind.
- *kdc_ports = 750,88*
  Dieser Parameter legt die Ports fest, auf denen der KDC Anfragen entgegennimmt. Bei den hier angegebenen Ports handelt es sich um die Standard-Ports. Sie sollten nicht unnötig verändert werden, da eine Änderung dazu führen kann, dass einige Anwendungen sich mit dem KDC verbinden können, wenn in der Anwendung der Port für die Kom-

munikation mit dem KDC fest eingebunden ist. Hierbei handelt es sich um UDP-Ports. Wenn der KDC läuft, können Sie mit dem Kommando `ss -uln` testen, ob die Ports erreichbar sind. In der daraus resultierenden Auflistung sehen Sie dann die beiden UDP-Ports 750 und 88, die vom KDC verwendet werden.

- *[realms]*
  In diesem Bereich werden die Parameter für alle *Realms* verwaltet, für die der KDC verantwortlich ist. Zu jedem Realm, den Sie auf diesem KDC einrichten wollen, gibt es anschließend einen eigenen Konfigurationsbereich.
- *EXAMPLE.NET =*
  Das ist der Realm, für den dieser KDC verantwortlich ist. In den geschweiften Klammern werden die für diesen Realm relevanten Parameter eingetragen.
- *database_name = /var/lib/krb5kdc/principal*
  Das sind die Namen und der Pfad zur Kerberos-Datenbank. In dem Beispiel sehen Sie die Pfade auf einem Debian-System. Bei CentOS lautet der Pfad /var/lib/kerberos/principal.
- *admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab*
  Hierbei handelt es sich um die Datei, die der Admin der Datenbank benutzt, um sich am Kerberos zu authentifizieren. Auch hier sehen Sie im Listing wieder den Pfad für ein Debian-System.
- *acl_file = /etc/krb5kdc/kadm5.acl*
  Das ist die Datei, in der die Zugriffsregeln auf die Datenbank festgelegt werden. Diese Datei ist im Moment noch nicht vorhanden und muss von Ihnen später noch erstellt werden, um festzulegen, wer welche Rechte in der Datenbank hat.
- *key _stash_file = /etc/krb5kdc/stash*
  In dieser Datei befindet sich das Master-Passwort, das Sie während der Initialisierung der Datenbank festlegen werden.
- *kdc_ports = 750,88*
  Das sind die Ports, über die der KDC dieses Realms erreicht werden kann.
- *max_life = 10h 0m 0s*
  Hier geben Sie an, wie lange ein Ticket gültig ist. Zehn Stunden ist der Standardwert.
- *max_renewable_life = 7d 0h 0m 0s*
  Bei diesem Wert geben Sie an, wie lange ein Ticket maximal verlängert werden kann, bevor es neu angefordert werden muss. Sieben Tage ist der Standardwert.
- *master_key_type = des3-hmac-sha1*
  Dieser Parameter legt den Typ der Verschlüsselung für das Master-Passwort fest. Dieser Parameter ist bei CentOS nicht gesetzt. Wollen Sie eine spezielle Verschlüsselung für den Master-Key, müssen Sie den Parameter nachtragen.
- *supported_enctypes = aes256-cts:normal aes128-cts:normal*
  Hier können Sie festlegen, welche Verschlüsselungen der Kerberos-Server unterstützen soll. Die Zeile ist hier auskommentiert, da es sich bei den angegebenen Werten um die Standardverschlüsselungen handelt.
- *default_principal_flags = +preauth*
  Dieser zusätzliche Parameter, der beim Starten des Kerberos-Servers übergeben wird, sorgt dafür, dass eine Pre-Authentication stattfinden muss. Das erhöht die Sicherheit

# 15 OpenLDAP im Container

Nach der Virtualisierung von Servern haben sich *Container-Technologien* als weitere Form der ressourcensparenden Alternative zur Installation einzelner Dienste auf separater Hardware durchgesetzt. Die Idee hinter den Containern ist, nicht mehr ganze Betriebssysteme zu virtualisieren, in welchen dann ein einzelner Dienst wie ein Webservice, ein Webauftritt oder Ähnliches eingerichtet wird. In das *Image*, welcher als Vorlage für einen Container dient, werden lediglich die Ressourcen gepackt, welche für einen speziellen Dienst notwendig sind, und im Container, der aus diesem Image gestartet wird, wird nur der Prozess gestartet, welcher diesen Dienst zur Verfügung stellt. Das alles spart Platz, Speicher und CPU-Zeit. Zudem vereinfacht dies Aktualisierungen, da Sie einfach das Image austauschen und den Container aus diesem neuen Image neu starten müssen. Geht etwas schief, greifen Sie einfach auf das alte Image zurück.

Am Ende dieses Kapitels werden Sie in der Lage sein, mit Images und Containern zu arbeiten: von der Erstellung eigener Images über die Einrichtung und Start bis hin zur Einbindung der OpenLDAP-Container in Ihr Netzwerk. Exemplarisch für Container-Technologien werden Sie dabei *Docker* kennenlernen.

## 15.1 Docker

Zu einem der Platzhirsche in der Welt der Container hat sich *Docker* gemausert. Sicherlich gibt es noch einige Alternativen wie Rocket, Cri-O oder LXC. Die Mechanismen der Konkurrenten sind allerdings überall ähnlich, sodass Sie sich im Anschluss auch in diese Alternativen gut werden einfinden können.

Zunächst gilt es, zwei Begriffe zu klären: *Image* und *Container*. Das *Image* ist quasi das Ausgangsmaterial, die eigentlichen Dateien eines laufenden *Containers*. Zieht man einen Vergleich zu VMware, so ähnelt ein Image am ehesten einer *VMDK*-Datei, also der virtuellen Festplatte. Diese *Images* können Sie selbst erstellen oder aus öffentlichen *Repositorys* beziehen. Beide Möglichkeiten werden Sie in diesem Kapitel kennenlernen. Basierend auf diesem *Image* starten Sie dann im Anschluss einen Container mit von Ihnen gewählten Konfigurationen.

### 15.1.1 Einrichtung des Docker-Servers

Zuerst einmal müssen Sie *Docker* installieren und einrichten. *Docker* ist mittlerweile Bestandteil der meisten Distributionen und kann daher beispielsweise unter Debian mit

```
apt-get install docker
```

installiert werden. Unter CentOS installieren Sie den Dienst entsprechend mit

```
yum install docker
```

Der zugehörige Dienst lässt sich im Anschluss mit

```
systemctl start docker && systemctl enable docker
```

starten und dauerhaft einrichten.

### 15.1.2 Der erste Container

Ein zentrales *Repository* für Images ist *DockerHub* unter *docker.io*. Geben Sie später nur den Namen eines Images an, so greift der Docker-Dienst auf dieses Repository zu. Für einen ersten Test laden und starten Sie das Image *Busybox* und führen dort den Befehl `date` aus. Listing 15.1 zeigt Ihnen die entsprechenden Befehle:

**Listing 15.1** Der erste Container

```
mydocker:~ # docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
322973677ef5: Pull complete
Digest: sha256:1828
     edd60c5efd34b2bf5dd3282ec0cc04d47b2ff9caa0b6d4f07a21d1c08084
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

mydocker:~ # docker run busybox date
Mon Dec 23 13:54:07 UTC 2019
```

Mit dem *Tag* `latest` wird die aktuelle Version (*latest*) angezeigt. Diese Tags können aber auch eine konkrete Versionsnummer beinhalten. Geladen wird in unserem Fall dann das *Busybox*-Image aus einem sogenannten *Namespace* (hier: *library*). Mit diesen *Namespaces* lässt sich ein Repository unterteilen und Berechtigungen für den Zugriff gewähren. Der vollständige URI für das Image wäre daher *docker.io/library/busybox:latest*.

Der Start des Images ist dann recht unspektakulär und besteht einfach in dem Aufruf des Kommandos `date`. (Die Ausgabe zeigt Ihnen, dass es in diesem Moment vielleicht Zeit gewesen wäre, sich eher den Weihnachtsvorbereitungen als dem Verfassen eines Fachbuchs zu widmen.)

Ohne Angaben eines Befehls würde die *Busybox* kurz starten und sich wieder beenden, da dieses Image keine Anweisungen enthält, welcher Prozess oder welches Skript standardmäßig zu starten ist. Das *Busybox*-Image dient oft als Basis für die Erstellung weiterer Images. Es wird auf der Projektseite auch als das *«Schweizer Taschenmesser für embedded-Linux»* bezeichnet und kommt unter anderem auf vielen *Embedded-Systemen* wie der

FritzBox von AVM, den TomTom-Navigationsgeräten oder dem Thermomix zum Einsatz. In einem weiteren Beispiel nehmen wir uns ein Image, welches automatisch einen Dienst startet sobald der darauf basierende Container startet. Auch in der Container-Welt gibt es ein *Hello, World*-Beispiel, wie Sie in Listing 15.2 nachvollziehen können:

**Listing 15.2** Hello World mit Docker

```
mydocker:~ # docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:4
     fe721ccc2e8dc7362278a29dc660d833570ec2682f4e4194f4ee23e415e1064
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

mydocker:~ # docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 (amd64)
 3. The Docker daemon created a new container from that image which runs the
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent
 it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Dieses Image besitzt einen Befehl (einen sogenannten *Entrypoint*) welcher automatisch beim Start ausgeführt wird und Ihnen mitteilt, dass mit Ihrer Docker-Installation offenbar alles in Ordnung ist.

Jetzt wollen wir uns einige nützliche Befehle für den Umgang mit Containern anschauen. Zwei haben Sie ja bereits kennengelernt: `docker pull` zum Laden eines Images sowie `docker run` zum Starten eines Containers. Um weitere Optionen und Befehle kennenzulernen, nehmen wir uns als Beispiel einen Webserver. Starten Sie einmal einen Webserver-Container, wie in Listing 15.3 gezeigt:

**Listing 15.3** Ein Webserver unter Apache

```
mydocker:~ # docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
000eee12ec04: Pull complete
32b8712d1f38: Pull complete
f1ca037d6393: Pull complete
c4bd3401259f: Pull complete
51c60bde4d46: Pull complete
Digest: sha256:
    ac6594daaa934c4c6ba66c562e96f2fb12f871415a9b7117724c52687080d35d
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest

mydocker:~ # docker run httpd
AH00558: httpd: Could not reliably determine the server's fully \
        qualified domain name, using 172.17.0.2. Set the 'ServerName' \
        directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully \
        qualified domain name, using 172.17.0.2. Set the 'ServerName' \
        directive globally to suppress this message
[Mon Dec 23 23:59:30.780974 2019] [mpm_event:notice] \
        [pid 1:tid 140205162103936] AH00489: Apache/2.4.41 (Unix) \
        configured -- resuming normal operations
[Mon Dec 23 23:59:30.781075 2019] [core:notice] \
        [pid 1:tid 140205162103936] AH00094: Command line: \
        'httpd -D FOREGROUND'
```

Leider können Sie jetzt erst einmal nichts auf dieser Shell machen, da der Container im Vordergrund gestartet wurde und Sie sich quasi mittendrin im `httpd`-Prozess befinden. Brechen Sie also mit STRG+C den Vorgang ab und beenden damit den Container wieder. Dann starten Sie den Container erneut, diesmal allerdings mit einer zusätzlichen Option:

```
docker run -d httpd
```

Der Container startet erneut, nun aber ***d****etached*. Daher sehen Sie auch kaum Ausgaben, außer einer 64-stelligen Hexadezimalzahl, der ID, unter welcher der Container nun gestartet und vom Docker-Dienst angesprochen wird. Listing 15.4 zeigt Ihnen eine entsprechende Ausgabe:

**Listing 15.4** Starten des Web-Containers im Hintergrund

```
mydocker:~ # docker run -d httpd
01205c72fd871804048fa56d4095bd92d65f6371b09f113614c7a92e17f8bcde
```

Sie sind also «losgelöst» vom `httpd`-Container, der nun im Hintergrund weiterläuft. Kontrollieren Sie mit `docker ps`, ob der Container läuft, wie in Listing 15.5 gezeigt:

**Listing 15.5** docker ps

```
mydocker:~ # docker ps
CONTAINER ID IMAGE COMMAND            CREATED  STATUS   PORTS  NAMES
01205c72fd87 httpd "httpd-foreground" 2min.ago Up 2min. 80/tcp festive_minsky
```

Der angezeigte Name (in unserem Beispiel *festive_minsky*) wird von Docker zufällig generiert. Sie können diesen Namen oder die *Container ID* nutzen, um nun auf diesen Container zuzugreifen. So können Sie zum Beispiel die Logs dieses Containers ausgeben lassen oder einen Befehl in dem Container absetzen, zum Beispiel eine Shell starten und sich das Dateisystem des Containers anschauen, wie Sie in Listing 15.6 nachvollziehen können:

**Listing 15.6** Protokolle eines Containers

```
mydocker:~ # docker logs festive_minsky
AH00558: httpd: Could not reliably determine the server's fully \
         qualified domain name, using 172.17.0.2. Set the 'ServerName' \
         directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully \
         qualified domain name, using 172.17.0.2. Set the 'ServerName' \
         directive globally to suppress this message
[Tue Dec 24 00:31:43.295532 2019] [mpm_event:notice] \
         [pid 1:tid 140367638717568] AH00489: Apache/2.4.41 (Unix) \
         configured -- resuming normal operations
[Tue Dec 24 00:31:43.319263 2019] [core:notice] \
         [pid 1:tid 140367638717568] AH00094: Command line: \
         'httpd -D FOREGROUND'

mydocker:~ # docker exec -it festive_minsky bash
root@01205c72fd87:/usr/local/apache2# ls -l /
total 68
drwxr-xr-x 1 root root 4096 Nov 23 00:36 bin
drwxr-xr-x 2 root root 4096 Nov 10 12:17 boot
drwxr-xr-x 5 root root 340 Dec 23 00:31 dev
drwxr-xr-x 1 root root 4096 Dec 24 00:31 etc
drwxr-xr-x 2 root root 4096 Nov 10 12:17 home
drwxr-xr-x 1 root root 4096 Nov 23 00:36 lib
drwxr-xr-x 2 root root 4096 Nov 18 00:00 lib64
drwxr-xr-x 2 root root 4096 Nov 18 00:00 media
drwxr-xr-x 2 root root 4096 Nov 18 00:00 mnt
drwxr-xr-x 2 root root 4096 Nov 18 00:00 opt
dr-xr-xr-x 228 root root 0 Dec 24 00:31 proc
drwx------ 1 root root 4096 Dec 24 00:41 root
drwxr-xr-x 3 root root 4096 Nov 18 00:00 run
drwxr-xr-x 2 root root 4096 Nov 18 00:00 sbin
drwxr-xr-x 2 root root 4096 Nov 18 00:00 srv
dr-xr-xr-x 13 root root 0 Dec 24 00:31 sys
drwxrwxrwt 1 root root 4096 Nov 23 00:35 tmp
drwxr-xr-x 1 root root 4096 Nov 18 00:00 usr
drwxr-xr-x 1 root root 4096 Nov 18 00:00 var
```

Verlassen Sie den Container mit `exit` wieder.

Wenn Ihnen der Container schon seine IP-Adresse verrät, dann testen Sie doch mal, ob der Webserver auch sauber gestartet ist. Starten Sie einen Browser auf Ihrem Host und rufen Sie die URL `http://172.17.0.2` (beziehungsweise die IP-Adresse, die in Ihrer Ausgabe aus Listing 15.6 angegeben ist). Bild 15.1 zeigt Ihnen das Ergebnis beim Zugriff mit dem lokalen Browser auf dem Docker-Host:

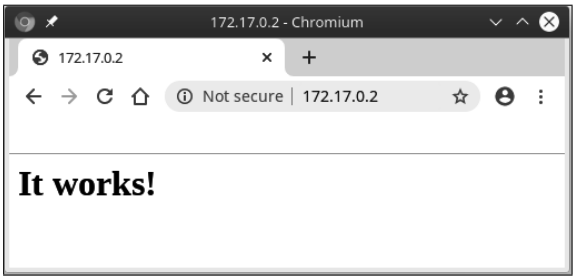


**Bild 15.1** httpd-Container

**Hinweis**

Das funktioniert zu diesem Zeitpunkt nur von Ihrem Docker-Host aus, da der Container noch nicht von außen erreichbar ist. Dazu kommen wir später noch. Hat Ihr Host keine grafische Oberfläche, so können Sie auch Tools wie `w3m`, `wget` oder `curl` nutzen, um den Container zu testen:

```
mydocker # curl http://172.17.0.2
<html><body><h1>It works!</h1></body></html>
```

Wenn Sie genug vom Container haben, beenden Sie ihn wieder mit `docker stop festive_minsky` (beziehungsweise dem Namen oder der ID Ihres Containers). Mit `docker start festive_minsky` lässt sich der Container direkt wieder starten.

Wollen Sie den Container löschen, lässt sich das mit einem `docker rm festive_minsky` durchführen. Um zu kontrollieren, ob Sie noch weitere beendete und nicht mehr benötigte Container auf Ihrem Host besitzen, setzen Sie den Befehl `docker ps -a` ab und räumen Sie auf. Listing 15.7 zeit Ihnen, wie Sie aufräumen können:

**Listing 15.7** Anzeigen beendeter Container

```
mydocker:~ # docker ps -a
CONTAINER ID IMAGE   COMMAND CREATED  STATUS     PORTS NAMES
932a46fe2da6 busybox "date"  2hrs ago Exited (0)       fervent_mahavira

mydocker:~ # docker rm fervent_mahavira
```

## 15.2 OpenLDAP

Aber kehren wir nun für weitere Beispiele zu OpenLDAP zurück und nehmen uns ein Image mit einem LDAP-Service vor. Davon werden Sie sicher reichlich im Internet finden. Später bauen Sie sich auch noch ein eigenes Image. Wir verwenden für unsere Beispiele das Image `osixia/openldap`. Beim Start können Sie gleich einen sprechenden Namen für den späteren Zugriff angeben, wie Sie in Listing 15.8 sehen:

**Listing 15.8** OpenLDAP-Container

```
mydsocker:~ # docker run -d --name ldapcontainer osixia/openldap
36766a24d082e119c390b19493a0b6be795f7f628e6f28e4bf647d86a087768b

mydocker:~ # docker ps
CONTAINER ID IMAGE           COMMAND               CREATED       STATUS
  PORTS            NAMES
36766a24d082 osixia/openldap "/container/tool/run" 6 seconds ago Up 5 seconds
  389/tcp, 636/tcp ldapcontainer

mydocker:~ # docker logs ldapcontainer
*** CONTAINER_LOG_LEVEL = 3 (info)
*** Search service in CONTAINER_SERVICE_DIR = /container/service :
*** link /container/service/:ssl-tools/startup.sh to /container/run/startup/:
    ssl-tools
*** link /container/service/slapd/startup.sh to /container/run/startup/slapd
*** link /container/service/slapd/process.sh to /container/run/process/slapd/
    run
*** Set environment for startup files
*** Environment files will be proccessed in this order :
Caution: previously defined variables will not be overriden.
/container/environment/99-default/default.startup.yaml
/container/environment/99-default/default.yaml

To see how this files are processed and environment variables values,
run this container with '--loglevel debug'
*** Running /container/run/startup/:ssl-tools...
*** Running /container/run/startup/slapd...
Database and config directory are empty...
Init new ldap server...
 Backing up /etc/ldap/slapd.d in /var/backups/slapd-2.4.48+dfsg-1~bpo10+1\
         ... done.
 Creating initial configuration... done.
 Creating LDAP directory... done.
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of restart.
Start OpenLDAP...
Waiting for OpenLDAP to start...
Add bootstrap schemas...
config file testing succeeded
Add image bootstrap ldif...
Add custom bootstrap ldif...
Add TLS config...
No certificate file and certificate key provided, generate:
/container/service/slapd/assets/certs/ldap.crt and /container/service/\
           slapd/assets/certs/ldap.key
2019/12/24 00:48:21 [INFO] generate received request
2019/12/24 00:48:21 [INFO] received CSR
2019/12/24 00:48:21 [INFO] generating key: ecdsa-384
2019/12/24 00:48:21 [INFO] encoded CSR
2019/12/24 00:48:21 [INFO] signed certificate with serial number \
```

```
          391996448698248834201445880913856130277495567280
Link /container/service/:ssl-tools/assets/default-ca/default-ca.pem \
          to /container/service/slapd/assets/certs/ca.crt
Disable replication config...
Stop OpenLDAP...
Configure ldap client TLS configuration...
Remove config files...
First start is done...
*** Set environment for container process
*** Remove file /container/environment/99-default/default.startup.yaml
*** Environment files will be proccessed in this order :
Caution: previously defined variables will not be overriden.
/container/environment/99-default/default.yaml

To see how this files are processed and environment variables values,
run this container with '--loglevel debug'
*** Running /container/run/process/slapd/run...
5e01d0d5 @(#) $OpenLDAP: slapd (Jul 30 2019 16:24:19) $
  Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>
5e01d0d5 slapd starting
```

Die Standardwerte dieses LDAP-Services finden Sie in Tabelle 15.1:

**Tabelle 15.1** Standardwerte des Images osixia/openldap

| | |
|---|---|
| Kontext | dc=example,dc=org |
| Administrator | cn=admin,dc=example,dc=org |
| Kennwort | admin |

Um auf den Dienst zuzugreifen, müssen Sie noch die IP-Adresse des Servers herausfinden. Dazu geben Sie den Befehl

```
docker inspect ldapcontainer|grep IPAddress
```

ein. Der Befehl `docker inspect` liefert Ihnen insgesamt alle möglichen Informationen zum laufenden Container. Auf einige werden wir später noch zurückkommen. Danach können Sie sich mit dem folgenden Befehl den Inhalt des LDAP-Verzeichnisses anschauen. Listing 15.9 zeigt Ihnen den ersten Kontakt mit Ihrem OpenLDAP-Container:

**Listing 15.9** Inhalt des LDAP-Verzeichnisses

```
mydocker:~ # docker inspect ldapcontainer|grep IPAddress
   "SecondaryIPAddresses": null,
   "IPAddress": "172.17.0.2",
     "IPAddress": "172.17.0.2",

mydocker:~ # ldapsearch -H ldap://172.17.0.2 -D "cn=admin,dc=example,dc=org"
     \
  -w admin -b "dc=example,dc=org"
# extended LDIF
#
# LDAPv3
```

```
# base <dc=example,dc=org> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.org
dn: dc=example,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
o: Example Inc.
dc: example

# admin, example.org
dn: cn=admin,dc=example,dc=org
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9RDBVNXdVNk1laXdEZzNpb21hbi9LNDl0alVoSldqVWU=

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Der erste Schritt wäre also geschafft. Momentan ist der Server allerdings nur vom Docker-Host aus erreichbar. Zudem ist die OpenLDAP-Datenbank noch im Container enthalten, und alle Änderungen wären nach dem Löschen des Containers verloren. Als Nächstes müssen Sie also den Container bzw. den Dienst in Ihr Netzwerk einbinden und dafür sorgen, dass die Nutzdaten Ihres Dienstes dauerhaft gespeichert werden.

### 15.2.1 Netzwerken

Wie Sie gesehen haben, sind die Dienste von Docker-Containern ohne zusätzliche Konfiguration nicht aus dem externen Netzwerk des Docker-Hosts erreichbar. Docker bietet Ihnen mehrere Möglichkeiten an, einen Container in Ihr Netzwerk zu integrieren. Tabelle 15.2 zeigt Ihnen die möglichen Netzwerkkonfigurationen:

**Tabelle 15.2** Docker-Netzwerke

| | |
|---|---|
| bridge | Container, welche miteinander kommunizieren sollen, werden einer internen *Bridge* zugeordnet. |
| overlay | Dieses Netzwerk überspannt mehrere Docker-Hosts und verbindet Container somit miteinander unabhängig davon, auf welchem Host diese gerade gestartet wurden. |
| host | Container nutzen das Netzwerk des Docker-Hosts inklusive der IP-Adresse direkt. |
| macvlan | Hierbei ordnen Sie dem Container eine eigene MAC-Adresse zu. Diese kann das Netzwerk des Hosts unabhängig von dessen Netzwerkeinstellungen nutzen. |
| none | Container, welche kein Netzwerk benötigen. |

Die aktuelle Netzwerkkonfiguration Ihres Docker-Hosts können Sie sich wie in Listing 15.10 anzeigen lassen:

**Listing 15.10** Standardnetze von Docker

```
mydocker:~# docker network list
NETWORK ID   NAME   DRIVER SCOPE
224ca3f4c152 bridge bridge local
0f27284d963a host   host   local
2791fa9ea7ac none   null   local
```

Geben Sie wie im Fall aus Listing 15.8 keine Netzwerkkonfiguration an, so wird der Container der Bridge mit dem Namen `bridge` zugeordnet. Das können Sie entweder mit dem Befehl `docker inspect ldapcontainer` oder einer Untersuchung des *bridge*-Netzwerks herausfinden. Letztere finden Sie in Listing 15.11:

**Listing 15.11** Untersuchen eines Docker-Netzwerks

```
mydocker:~# docker inspect bridge
[
 {
  "Name": "bridge",
  "Id": "224ca3f4c1522f4724b34b4151580ef5fd95944dbcf4bb0c98fae640336ef3ad",
  "Created": "2019-12-24T01:47:55.308165415+01:00",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
   "Driver": "default",
   "Options": null,
   "Config": [
    {
     "Subnet": "172.17.0.0/16",
     "Gateway": "172.17.0.1"
    }
   ]
  },
  "Internal": false,
  "Attachable": false,
  "Ingress": false,
```

# Stichwortverzeichnis