OTOBO Installation Guide

Release 11.0

Rother OSS GmbH

20.09.2024

Contents

Published by: Rother OSS GmbH, (https://otobo.io), Oberwalting 31, 94339 Leiblfing, Germany.

Authors: OTRS AG (original version), Rother OSS GmbH (https://otobo.io).

CHAPTER 1

Einleitung

OTOBO ist ein Open-Source-Ticketsystem mit umfassenden Funktionalitäten zur Verwaltung von Kundenanrufen und -E-Mails. Es wird unter der GNU General Public License (GPL) vertrieben und wurde auf verschiedenen Linux-Plattformen getestet.

1.1 Über dieses Handbuch

Diese Anleitung wendet sich an Systemadministratoren und beschreibt das Vorgehen beim Installieren und Aktualisieren von OTOBO.

Für Installation und Updates steht keine grafische Benutzeroberfläche zur Verfügung. Die folgenden Kapitel führen Sie als Systemadministrator Schritt für Schritt durch die erforderlichen Abläufe.

Alle Befehle für die Kommandozeile entsprechen folgendem Aufbau: `benutzername > auszuführender-befehl`. Der Benutzername gibt an, mit welchem Benutzerkonto auf dem Betriebssystem der Befehl ausgeführt werden soll. Beginnt ein Befehl mit `root>`, muss dieser von einem Benutzer mit Root-Berechtigungen ausgeführt werden. Beginnt ein Befehl mit `otobo>`, muss dieser mit dem für OTOBO angelegten Nutzer durchgeführt werden.

Warnung: Achten Sie darauf, `benutzername>` nicht mitzumarkieren, wenn Sie den Befehl kopieren und in die Kommandozeile einfügen. Dies würde einen Fehler verursachen.

Wir setzen in dieser Anleitung voraus, dass OTOBO im Verzeichnis `/opt/otobo` installiert wird. Möchten Sie OTOBO in einem anderen Vereichnis installieren, ist es notwendig, den Pfad in den Befehlen entsprechend anzupassen oder einen symbolischen Link auf Ihr Zielverzeichnis zu erstellen.

```
root> ln -s /path/to/otobo /opt/otobo
```

CHAPTER 2

Hardware- und Software-Anforderungen

OTOBO kann als Webanwendung unter Linux und anderen Unix-Derivaten wie OpenBSD oder FreeBSD installiert werden. Das Ausführen von OTOBO unter Microsoft Windows wird nicht unterstützt.

Die Webanwendung nutzt eine relationale Datenbank als Backend. Um OTOBO ausführen zu können, benötigen Sie also mindestens einen Webserver und einen Datenbankserver. Diese können wahlweise beide auf einem Server oder auf unterschiedlichen Hosts installiert werden.

Alternativ kann OTOBO unter Docker ausgeführt werden. In diesem Fall sind Web- und Datenbankserver bereits im Setup enthalten. Die Bereitstellung mit Kubernetes wird derzeit gerade entwickelt.

OTOBO benötigt Perl mit einigen weiteren CPAN-Modulen. Sie können diese zusätzlichen Perl-Module über eine Perl-Paketverwaltung oder den Paketmanager Ihres Betriebssystems (rpm, yast, apt-get) installieren. Folgender Konsolenbefehl prüft die Modulabhängigkeiten:

```
otobo> /opt/otobo/bin/otobo.CheckModules.pl --inst
```

Falls Pakete fehlen, erhalten Sie einen Installationsbefehl für Ihr Betriebssystem, indem Sie das Skript mit der Option `--list` ausführen.

```
otobo> /opt/otobo/bin/otobo.CheckModules.pl --list | more
```

Die aufgeführten Befehle sollten dann mit Root-Berechtigungen ausgeführt werden.

In der Ausgabe des Modulprüfungsskripts werden die installierten Pakete und Versionsnummern angezeigt. Fehlende Module sind durch einen Kommentar gekennzeichnet.

```
Required packages:
  o Archive::Tar.....................ok (v2.32)
  o Archive::Zip.....................ok (v1.67)
  o Const::Fast......................ok (v0.014)
  o Date::Format.....................ok (v2.24)
  o DateTime.........................ok (v1.51)
    o DateTime::TimeZone.............ok (v2.38)
  o Convert::BinHex..................ok (v1.125)
  o DBI..............................ok (v1.643)
  o Digest::SHA......................ok (v6.02)
```

```
  o File::chmod.....................ok (v0.42)
  o List::AllUtils..................ok (v0.15)
  o LWP::UserAgent..................ok (v6.26)
  o Moo.............................ok (v2.003006)
  o namespace::autoclean............ok (v0.29)
  o Net::DNS........................ok (v1.22)
  o Net::SMTP::SSL..................ok (v1.04)
  o Path::Class.....................ok (v0.37)
  o Sub::Exporter...................ok (v0.987)
  o Template::Toolkit...............ok (undef)
  o Template::Stash::XS.............ok (undef)
  o Text::CSV.......................ok (v1.95)
  o Text::Trim......................ok (v1.04)
  o Time::HiRes.....................ok (v1.9760)
  o Try::Tiny.......................ok (v0.30)
  o URI.............................ok (v1.71)
  o XML::LibXML.....................ok (v2.0207)
  o YAML::XS........................ok (v0.81)
  o Unicode::Collate................ok (v1.27)
  o CGI::PSGI.......................ok (v0.15)
  o DBIx::Connector.................ok (v0.56)
  o Path::Class.....................ok (v0.37)
  o Plack...........................ok (v1.0047)
  o Plack::Middleware::ForceEnv.....ok (v0.02)
  o Plack::Middleware::Header.......ok (v0.04)
  o Plack::Middleware::Refresh......ok (undef)
  o Plack::Middleware::ReverseProxy..ok (v0.16)
  o Plack::Middleware::Rewrite......ok (v2.101)
  o SOAP::Transport::HTTP::Plack....ok (v0.03)

Recommended features for setups using apache:
  o ModPerl::Util...................ok (v2.000011)

Database support (installing one is required):
  o DBD::mysql......................ok (v4.050)

Various features for additional functionality:
  o Encode::HanExtra................ok (v0.23)
  o Net::LDAP.......................ok (v0.66)
  o Crypt::Eksblowfish::Bcrypt......ok (v0.009)
  o XML::LibXSLT....................ok (v1.99)
  o XML::Parser.....................ok (v2.46)

Features enabling communication with a mail-server:
  o Net::SMTP.......................ok (v3.11)
  o Mail::IMAPClient................ok (v3.42)
  o Authen::SASL....................ok (v2.16)
  o Authen::NTLM....................ok (v1.09)
  o IO::Socket::SSL.................ok (v2.067)

Optional features which can increase performance:
  o JSON::XS........................ok (v4.02)
  o Text::CSV_XS....................ok (v1.41)

Required packages if you want to use PSGI/Plack (experimental and advanced):
  o Gazelle.........................ok (v0.49)
  o Linux::Inotify2.................ok (v2.2)
  o Plack::App::File................ok (undef)
```

2.1 Hardware-Anforderungen

Die Hardware-Anforderungen sind weitgehend davon abhängig, wie OTOBO genutzt wird. Mit OTOBO können wenige Tickets pro Monat oder mehrere Tickets pro Tag verarbeitet werden. Die Anforderungen an den Speicherplatz hängen ebenfalls von der Anzahl der Tickets und der Größe der Anlagen ab.

Mindestanforderungen für den Testbetrieb:

- kleine CPU
- 4 GB RAM
- 10 GB Speicher

Mindestanforderungen für den Produktivbetrieb:

- 3 GHz Xeon oder vergleichbare CPU
- 8 GB RAM (16 GB empfohlen)
- 40 GB Speicher

Bemerkung: Hardware-Anforderungen variieren stark und ergeben sich aus den individuellen Einsatzszenarien. Bitte lassen Sie sich von Ihrem OTOBO Consultant beraten.

2.2 Software-Anforderungen

Perl

- Perl 5.24.0 oder höher
- Alle Perl-Pakete, die vom Konsolebefehl `/opt/otobo/bin/otobo.CheckModules.pl` ausgegeben werden

Webserver

- Apache HTTP Server Version 2.4

Datenbanken

- MySQL 5.6 oder höher
- MariaDB
- PostgreSQL 9.2 oder höher
- Oracle 10g oder höher

Optional

- Elasticsearch 7.x (Schnellsuche mit Live-Preview)
- Redis (schnelles Caching)
- nginx oder jeder andere Webserver, der als Reverseproxy eingesetzt werden kann (SSL-Unterstützung und Lastverteilung)

Webbrowser

- Apple Safari
- Google Chrome

- Microsoft Internet Explorer 11 (nutzbar, bietet aber nicht den vollen Funktionsumfang und alle GUI-Aspekte)
- Microsoft Edge
- Mozilla Firefox
- Jeder andere moderne Webbrowser mit JavaScript-Unterstützung

CHAPTER 3

OTOBO Installation

Dieses Kapitel beschreibt die Installation und grundlegende Konfiguration des zentralen OTOBO Frameworks.

Dieses Kapitel führt Sie Schritt für Schritt durch die Installation von OTOBO auf Ihrem Server. Anschließend können Sie sich über die Weboberfläche am System anmelden, um es zu konfigurieren und zu administrieren.

Bemerkung: Wir empfehlen Docker und Docker Compose für die OTOBO-Installation. Durch die Verwendung der mitgelieferten Docker-Images werden alle empfohlenen Abhängigkeiten (wie Elasticsearch, Redis Cache, etc.) automatisch installiert und konfiguriert. Updates werden dadurch stark vereinfacht und die Performance verbessert. Die Anleitung zur Docker-basierten Installation finden Sie unter https://doc.otobo.org/manual/installation/11.0/en/content/installation-docker.html.

3.1 Vorbereitung: Deaktivieren Sie SELinux (sofern aktiv)

Bemerkung: Ist SELinux auf Ihrem System installiert und aktiv, deaktivieren Sie es vor der Installation. Anderenfalls wird OTOBO nicht korrekt funktionieren.

Wenn Sie nicht sicher sind, ob SELinux installiert und aktiv ist, geben Sie die Befehle `sestatus` und `getenforce` ein.

Der Befehl `sestatus` gibt den SELinux-Status aus und informiert darüber, welche SELinux Policy angewendet wird. SELinux Status: Ist SELinux aktiv, wird als Status enabled angezeigt. Aktueller Betriebsmodus: wird hier enforcing ausgegeben, läuft SELinux im Enforcing-Modus. Policy aus der Config-Datei: wird hier targeted ausgegeben, kommt die Targeted Policy zum Einsatz.

So deaktivieren Sie SELinux unter RHEL/CentOS/Fedora.

1. Wählen Sie in der Datei `/etc/selinux/config` die Konfiguration `SELINUX=disabled`:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. Starten Sie Ihr System neu. Versichern Sie sich nach dem Neustart, dass der Befehl `getenforce` Disabled zurückgibt:

```
root> getenforce
Disabled
```

3.2 Schritt 1: OTOBO entpacken und installieren

Laden Sie das neueste OTOBO-Release von https://ftp.otobo.org/pub/otobo/ herunter. Entpacken Sie das Quell-Archiv (zum Beispiel mit `tar`) in das Verzeichnis `/opt/otobo-install`:

```
root> mkdir /opt/otobo-install && mkdir /opt/otobo                    # Create a␣
↪temporary install directory
root> cd /opt/otobo-install                                           # Change into␣
↪the update directory
root> wget https://ftp.otobo.org/pub/otobo/otobo-latest-11.0.tar.gz   # Download he␣
↪latest OTOBO 10 release
root> tar -xzf otobo-latest-11.0.tar.gz                               # Unzip OTOBO
root> cp -r otobo-10.x.x/* /opt/otobo                                   # Copy the␣
↪new otobo directory to /opt/otobo
```

3.3 Schritt 2: Ergänzende Programme und Perl-Module installieren

Verwenden Sie das folgende Skript, um eine Übersicht über alle installierten und erforderlichen CPAN-Module sowie andere externe Abhängigkeiten zu erhalten.

Bemerkung: Auf Debian-Systemen müssen Sie möglicherweise einige perl-Pakete manuell installieren:

```
apt-get install -y libarchive-zip-perl libtimedate-perl libdatetime-perl libconvert-
↪binhex-perl libcgi-psgi-perl libdbi-perl libdbix-connector-perl libfile-chmod-perl␣
↪liblist-allutils-perl libmoo-perl libnamespace-autoclean-perl libnet-dns-perl␣
↪libnet-smtp-ssl-perl libpath-class-perl libsub-exporter-perl libtemplate-perl␣
↪libtext-trim-perl libtry-tiny-perl libxml-libxml-perl libyaml-libyaml-perl libdbd-
↪mysql-perl libapache2-mod-perl2 libmail-imapclient-perl libauthen-sasl-perl␣
↪libauthen-ntlm-perl libjson-xs-perl libtext-csv-xs-perl libpath-class-perl libplack-
↪perl libplack-middleware-header-perl libplack-middleware-reverseproxy-perl␣
↪libencode-hanextra-perl libio-socket-ssl-perl libnet-ldap-perl libcrypt-eksblowfish-
↪perl libxml-libxslt-perl libxml-parser-perl libconst-fast-perl
```

```
root> perl /opt/otobo/bin/otobo.CheckModules.pl -list
Checking for Perl Modules:
  o Archive::Tar.....................ok (v1.90)
  o Archive::Zip.....................ok (v1.37)
  o Crypt::Eksblowfish::Bcrypt.......ok (v0.009)
...
```

Bemerkung: Bitte beachten Sie, dass OTOBO eine funktionierende Perl-Installation mit allen Core-Modulen wie dem Modul `version` voraussetzt. Diese Module werden nicht explizit vom Skript überprüft. Auf einigen Systemen, wie z. B. RHEL, sind nicht alle Core-Pakete standardmäßig installiert. Hier kann es erforderlich sein, ein `perl-core`-Paket von Hand zu installieren.

Zur Installation der erforderlichen und optionalen Pakete können Sie wahlweise CPAN oder die Paketverwaltung Ihrer Linux-Distribution verwenden.

Dieses Kommando gibt einen Installationsbefehl zum Installieren der fehlenden Abhängigkeiten aus:

```
root> /opt/otobo/bin/otobo.CheckModules.pl --inst
```

Bemerkung: Es gibt eine Reihe optionaler oder alternativer Module, die insbesondere in individualisierten OTOBO Systemen zum Einsatz kommen. Ein Aufruf von CheckModules.pl ohne jedes Argument gibt Aufschluss über den vollen Funktionsumfang des Systems.

3.4 Schritt 3: OTOBO-Benutzer anlegen

Legen Sie einen dedizierten Benutzer für OTOBO in einer eigenen Gruppe an:

```
root> useradd -r -U -d /opt/otobo -c 'OTOBO user' otobo -s /bin/bash
```

Fügen Sie den Benutzer zur Gruppe Webserver hinzu (wenn der Webserver nicht als OTOBO-Benutzer ausgeführt wird):

```
root> usermod -G www-data otobo
(SUSE=www, Red Hat/CentOS/Fedora=apache, Debian/Ubuntu=www-data)
```

3.5 Schritt 4: Standard-Konfigurationsdatei aktivieren

Die Datei `$OTOBO_HOME/Kernel/Config.pm.dist` wird mit OTOBO ausgeliefert. Sie enthält Konfigurationsdaten für OTOBO. Aktivieren Sie die Konfiguration, indem Sie die Datei ohne die Dateinamenerweiterung `.dist` kopieren.

```
root> cp /opt/otobo/Kernel/Config.pm.dist /opt/otobo/Kernel/Config.pm
```

3.6 Schritt 5: Apache-Webserver konfigurieren

Installieren Sie zunächst den Apache2-Webserver und mod_perl. In der Regel nutzen Sie hierzu die Paketverwaltung Ihres Systems. Nachstehend finden Sie alle Befehle, die Sie benötigen, um Apache in den gängigsten Linux-Distributionen aufzusetzen.

```
# RHEL / CentOS:
root> yum install httpd mod_perl

# SuSE:
root> zypper install apache2-mod_perl

# Debian/Ubuntu:
root> apt-get install apache2 libapache2-mod-perl2
```

Zentrale Bedeutung hat die Auswahl des Multi-Processing-Modules (MPM). Für den OTOBO-Betrieb empfehlen wir das Modul **mpm_prefork**. Wie andere Apache-Module kann auch das MPM-Modul mit den Tools a2dismod und a2enmod verwaltet werden.

```
root> # check which MPM is active
root> apache2ctl -M | grep mpm_
```

Möglicherweise ist mpm_prefork bereits vorausgewählt. Dann kann dieser Schritt übersprungen werden.

Deaktivieren Sie mpm_event, falls das Modul aktiv ist.

```
root> a2dismod mpm_event
```

Deaktivieren Sie mpm.worker, falls dieses MPM aktiv ist.

```
root> a2dismod mpm_worker
```

Und aktivieren Sie schließlich mpm_prefork.

```
root> a2enmod mpm_prefork
```

Damit OTOBO optimal funktioniert, müssen weitere Apache-Module aktiv sein. Auch deren Status können Sie auf den meisten Systemen mit dem Tool a2enmod prüfen.

```
root> a2enmod perl
root> a2enmod deflate
root> a2enmod filter
root> a2enmod headers
```

Bemerkung: Auf manchen Systemen sind nicht alle Apache-Module vorhanden, dann wird während der Installation eine Fehlermeldung ausgegeben. Fahren Sie einfach mit der Installation fort. In den meisten Fällen werden die fehlenden Module nicht benötigt.

In den meisten Apache-Installationen findet sich ein Verzeichnis `conf.d`. Auf Linux-Systemen finden Sie es in aller Regel unter `/etc/apache` oder `/etc/apache2`.

3.6.1 Apache ohne SSL-Unterstützung konfigurieren

Kopieren Sie die Vorlage `/opt/otobo/scripts/apache2-httpd.include.conf` in das Apache-Verzeichnis `sites-available`. In den meisten Fällen muss die Vorlage nicht weiter angepasst werden. Aktivieren Sie die neue Konfiguration.

```
# Debian/Ubuntu:
root> cp /opt/otobo/scripts/apache2-httpd.include.conf /etc/apache2/sites-available/
↪zzz_otobo.conf
root> a2ensite zzz_otobo.conf
root> systemctl restart apache2
```

3.6.2 Apache mit SSL-Unterstützung kofigurieren

Kopieren Sie die Vorlagen `/opt/otobo/scripts/apache2-httpd-vhost-80.include.conf` und `/opt/otobo/scripts/apache2-httpd-vhost-443.include.conf` in das Apache-Verzeichnis `sites-availible`.

```
# Debian/Ubuntu:
root> cp /opt/otobo/scripts/apache2-httpd-vhost-80.include.conf /etc/apache2/sites-
↪available/zzz_otobo-80.conf
root> cp /opt/otobo/scripts/apache2-httpd-vhost-443.include.conf /etc/apache2/sites-
↪available/zzz_otobo-443.conf
```

Bearbeiten Sie die Dateien und ergänzen Sie die benötigten Informationen wie den Pfad zum SSL-Zertifikat. Anschließend aktivieren Sie die OTOBO-Apache-Konfiguration:

```
root> a2ensite zzz_otobo-80.conf
root> a2ensite zzz_otobo-443.conf
```

Jetzt können Sie Ihren Webserver neu starten und die neuen Konfigurationseinstellungen laden. Auf den meisten Systemen gelingt dies mit folgendem Befehl:

```
root> systemctl restart apache2
```

3.7 Schritt 6: Dateiberechtigungen anpassen

Führen Sie folgenden Befehl aus, um die Datei- und Verzeichnis-Berechtigungen für OTOBO zu definieren. Es wird versucht, die passenden Benutzer- und Gruppeneinstellungen für Ihr Setup zu ermitteln.

```
root> /opt/otobo/bin/otobo.SetPermissions.pl
```

3.8 Schritt 7: Datenbank anlegen

Installieren Sie zuerst das Datenbank-Paket. Wir empfehlen, das mit Ihrem Linuxsystem zur Verfügung gestellte MySQL- oder MariaDB-Paket zu verwenden. Sie können aber auch PostgreSQL oder Oracle nutzen.

In aller Regel erfolgt die Installation wieder über die Paketverwaltung Ihres Systems. Nachstehend finden Sie die zum Aufsetzen von MySQL in den gängigsten Linux-Distributionen benötigten Befehle.

```
# RHEL / CentOS:
root> yum install mysql-server

# SuSE:
root> zypper install mysql-community-server

# Debian/Ubuntu:
root> apt-get install mysql-server
```

Nach der Installation muss der MySQL-Server konfiguriert werden.

In MySQL ab Version 5.7 ist ein neues Authentifizierungsmodul aktiv, das ein Anlegen der Datenbank durch den OTOBO Web Installer verhindert. Bitte loggen Sie sich in diesem Fall in die MySQL-Konsole ein und definieren Sie ein anderes Authentifizierungsmodul und ein Passwort für den `root`-Benutzer:

```
root> mysql -u root
root> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
↪'NewRootPassword';
```

Für MariaDB > 10.1 verwenden Sie stattdessen folgenden Befehl:

```
root> mysql -u root
root> update mysql.user set authentication_string=password('NewRootPassword') plugin=
↪'mysql_native_password' where user='root';
```

Funktioniert dieser Befehl nicht, versuchen Sie es mit folgenden Kommandos:

```
root> mysql -u root
root> UPDATE mysql.user SET password = PASSWORD('NewRootPassword') WHERE user = 'root
↪';
root> UPDATE mysql.user SET authentication_string = '' WHERE user = 'root';
root> UPDATE mysql.user SET plugin = 'mysql_native_password' WHERE user = 'root';
```

Sofern nötig, können Sie das Authentifizierungsmodul nach der OTOBO-Installation wieder ändern.

Bemerkung: Folgende Konfigurationseinstellungen beschreiben die Minimalanforderungen für MySQL. Bitte ergänzen Sie die MySQL-Server-Konfigurationsdatei unter `/etc/my.cnf`, `/etc/mysql/my.cnf` oder `/etc/mysql/mysql.conf.d/mysqld.cnf` im Abschnitt `[mysqld]` um diese Zeilen:

```
max_allowed_packet   = 64M
innodb_log_file_size = 256M
```

Für MySQL-Versionen vor MySQL 8.0 sollten Sie außerdem den Query-Cache definieren:

```
query_cache_size     = 32M
```

Bitte ergänzen Sie die MySQL-Server-Konfigurationsdatei unter `/etc/my.cnf`, `/etc/mysql/my.cnf` oder `/etc/mysql/mysql.conf.d/mysqldump.cnf` im Abschnitt `[mysqldump]` um diese Zeilen:

```
max_allowed_packet   = 64M
```

Für den Produktivbetrieb empfehlen wir die Nutzung des Tools `mysqltuner`, mit dem Sie die bestmöglichen Einstellungen für Ihr System ermitteln können. Sie können das Skript unter `https://github.com/major/MySQLTuner-perl` aus GitHub herunterladen. In Debian- und Ubuntu-Systemen finden Sie es über die Paketverwaltung:

```
root> apt-get install mysqltuner
```

Ist die Installation abgeschlossen, führen Sie das Skript aus:

```
root> mysqltuner --user root --pass NewRootPassword
```

3.9 Schritt 8: Elasticsearch-Cluster aufsetzen

Für schnelle Suchen in OTOBO empfehlen wir ein aktives Elasticsearch-Cluster. Am einfachsten setzen Sie Elasticsearch auf dem gleichen Host wie OTOBO auf und lassen es den Standardport nutzen.

3.9.1 Beispiel für eine Elasticsearch-Installation unter Ubuntu 18.04 LTS

JDK-Installation

```
root> apt update
root> apt install openjdk-8-jdk
```

Elasticsearch-Installation

```
root> wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key␣
↪add -
root> echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee␣
↪/etc/apt/sources.list.d/elastic-7.x.list
root> apt update
root> apt -y install elasticsearch
```

3.9.2 Elasticsearch-Installation auf anderen Linux-Distributionen

Bitte nutzen Sie das Installations-Tutorial unter https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html.

3.9.3 Elasticsearch-Modul installieren

Außerdem erfordert OTOBO die Installation von Plugins in Elasticsearch:

```
root> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch ingest-
↪attachment
root> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch analysis-icu
```

3.9.4 Elasticsearch konfigurieren

Elasticsearch bietet viele unterschiedliche Konfigurationsoptionen und -möglichkeiten.

In größeren OTOBO-Systemen sollten Sie für einen fehlerfreien Betrieb den JVM Heap Space anpassen. Diese Einstellungen finden Sie in der Datei `/etc/elasticsearch/jvm.options`. Achten Sie darauf, dass Mindest- und Maximalwert für die JVM Heap Size übereinstimmen. Um den Heap auf 4 GB zu setzen, nehmen Sie folgende Einstellung vor:

```
-Xms4g
-Xmx4g
```

In unseren Tests hat sich ein Wert zwischen 4 und 10 GB für mittlere Systeme als optimal erwiesen.

Bemerkung: Bitte nutzen Sie das Installations-Tutorial unter `https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html`.

Jetzt können Sie Ihren Elasticsearch-Server neu starten, um die neuen Konfigurationseinstellungen zu laden. Auf den meisten Systemen gelingt dies mit folgendem Befehl:

```
root> systemctl restart elasticsearch
```

3.10 Schritt 9: Grundlegende Systemkonfiguration

Bitte verwenden Sie den Web Installer unter http://localhost/otobo/installer.pl (ersetzen Sie „localhost“ durch den Namen Ihres OTOBO-Hosts), um die Datenbank zu konfigurieren und grundlegende Systemeinstellungen wie die der E-Mail-Konten vorzunehmen.

3.11 Schritt 10: Erste Anmeldung

Geschafft! Jetzt können Sie sich mit dem zuvor generierten Passwort (s. o.) über http://localhost/otobo/index.pl als Benutzer `root@localhost` anmelden.

3.12 Schritt 11: OTOBO Daemon starten

Der OTOBO Daemon übernimmt asynchrone und wiederkehrende Aufgaben in OTOBO. Was früher in CronFile-Definitionen hinterlegt wurde, wird jetzt durch den OTOBO Daemon erledigt. Er ist deshalb für den OTOBO-Betrieb unerlässlich. Außerdem übernimmt der Daemon alle GenericAgent Jobs. Er muss vom OTOBO-Benutzer gestartet werden.

```
otobo> /opt/otobo/bin/otobo.Daemon.pl start
```

3.13 Schritt 12: CronJobs für den OTOBO-Benutzer

In `/opt/otobo/var/cron/\*.dist` liegen standardmäßig zwei OTOBO Cron-Dateien, die sicherstellen sollen, dass der OTOBO Daemon läuft. Aktivieren Sie diese, indem Sie die Dateien ohne die Dateinamenerweiterung „.dist“ kopieren.

```
root> cd /opt/otobo/var/cron/
root> for foo in *.dist; do cp $foo `basename $foo .dist`; done

root> cd /opt/otobo/
root> bin/Cron.sh start
```

Mit diesem Schritt ist die Grundeinrichtung des Systems abgeschlossen.

3.14 Schritt 13: Bash Auto-Completion einrichten (optional)

Alle regulären Befehlszeilenoptionen in OTOBO werden über die OTOBO-Konsolenschnittstelle ausgeführt. Damit wird eine Autovervollständigung für Eingaben in die Bash-Shell angeboten, die das Finden geeigneter Befehle und Optionen erheblich erleichtert.

Zum Aktivieren der Bash Auto-Completion installieren Sie das Paket `bash-completion`. Damit wird automatisch die Datei `/opt/otobo/.bash_completion` für den Benutzer `otobo` gesucht und geladen.

Sobald Sie Ihre Konsole neu gestartet haben, können Sie dann folgenden Befehl eingeben und durch TAB ergänzen, um alle verfügbaren Befehle anzuzeigen:

```
otobo> /opt/otobo/bin/otobo.Console.pl
```

Geben Sie einige Zeichen und lassen ein TAB folgen, werden alle auf der eingegebenen Zeichenfolge basierenden Befehle angezeigt. Geben Sie einen Befehl vollständig ein und drücken dann TAB, werden alle möglichen Optionen und Argumente angezeigt.

Bemerkung: Sollten Sie Probleme haben, können Sie folgende Zeile als `otobo`-Benutzer ausführen und zu Ihrem `~/.bashrc` hinzufügen, um die Befehle aus der Datei heraus auszuführen.

```
source /opt/otobo/.bash_completion
```

3.15 Schritt 14: Weiterführende Informationen

Wir empfehlen, das Kapitel OTOBO Performance Tuning zu lesen.

CHAPTER 4

Installation mit Docker und Docker Compose

Mit der Docker-basierten Installation können Sie Ihre komplette OTOBO-Instanz innerhalb weniger Minuten aufsetzen. Alle Abhängigkeiten sind in den Docker Images bereits enthalten.

- Service db: Als Standard-Datenbank ist MariaDB integriert.
- Service elastic: Für die OTOBO-Schnellsuche ist Elasticsearch eingerichtet.
- Service redis: Redis sorgt für schnelles Caching.
- Service web: Gazelle dient als schneller Perl-Webserver.
- Service nginx: Als optionaler Reverseproxy für die HTTPS-Unterstützung wird nginx eingesetzt.

Wir denken: Dieses Setup ist die perfekte Umgebung für eine OTOBO-Installation.

4.1 Technische Voraussetzungen

Hier sind die bisher getesteten Mindestanforderungen an die verwendete Software aufgeführt:

- Docker 19.03.13
- Docker Compose 1.25.0
- Git 2.17.1

Bemerkung: Für Ubuntu 24.04 LTS wird die Verwendung von Compose V2 empfohlen.

Compose V2 ist als Plugin für den Befehl `docker` implementiert. Ein Alias kann verwendet werden, um mit dem aus Compose V1 bekannten Befehl `docker-compose` kompatibel zu bleiben.

---

git, Docker, and Docker Compose can be installed with the standard system tools. Here is an example for installation on Ubuntu 20.04 Focal Fossa:

```
root> apt install git docker.io docker-compose curl
root> systemctl enable docker
```

Informationen zum weiteren Septup entnehmen Sie bitte der Git- und Docker-Dokumentation.

4.2 Installation

Wir gehen im Folgenden davon aus, dass alle erforderlichen Softwaremodule installiert sind und eine funktionierende Docker-Umgebung zur Verfügung steht. Außerdem, dass zur Interaktion mit Docker der Benutzer **docker_admin** verwendet wird. Der Docker-Admin kann entweder der **root**-Benutzer des Docker-Host oder ein spezieller Benutzer mit den erforderlichen Berechtigungen sein.

4.2.1 1. Clone the otobo-docker repo

The Docker images will eventually be fetched from the repository https://hub.docker.com. But there are some setup and command files that need to be cloned from the otobo-docker Github repository. Make sure that you specify the tag that corresponds to the current patch level version of OTOBO. For example, when OTOBO 11.0.2 is the current version then please use the tag rel-11_0_2.

Bemerkung: Wo das geklonte Repository liegt, spielt keine Rolle. In dieser Anleitung wählen wir /opt/otobo-docker als Arbeitsverzeichnis.

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo-docker.git --branch <TAG> -
↪-single-branch
docker_admin> cd otobo-docker    # change into the git sandbox
docker_admin> ls                 # just a sanity check, for example the file README.md␣
↪should exist
```

4.2.2 2. Create an initial .env file

In der Docker Compose Konfigurationsdatei .env werden die Einstellungen für die OTOBO-Umgebung vorgenommen. Diese Datei muss von Ihnen erstellt und individualisiert werden. Um Ihnen die Aufgabe zu erleichtern, werden zwei Beispieldateien mitausgeliefert, die Sie als Ausgangspunkt verwenden können. Soll per HTTPS auf die OTOBO Webapplikation via HTTPS zugegriffen werden, verwenden Sie bitte .docker_compose_env_https. Wir empfehlen, über HTTPS zuzugreifen. Wird kein HTTPS benötigt, verwenden Sie .docker_compose_env_http als Ausgangspunkt.

.docker_compose_env_http Die OTOBO Oberfläche ist über HTTP erreichbar.

.docker_compose_env_https Die OTOBO Oberfläche ist über Nginx als Reverse-Proxy per HTTPS erreichbar.

.docker_compose_env_https_custom_nginx Ähnlich zu .docker_compose_env_https, aber mit Unterstützung einer individuellen Nginx Konfiguration.

.docker_compose_env_https_kerberos Ähnlich zu .docker_compose_env_https, aber mit Beispiel Konfiguration für Single-Sign-On. Bitte beachten Sie, dass die Unterstützung für Kerberos noch **experimentell** ist.

.docker_compose_env_http_selenium und .docker_compose_env_https_selenium Diese werden nur für Entwicklungszwecke genutzt, wenn Selenium Tests aktiviert sind.

Bemerkung: `ls -a` zeigt die ausgeblendete Template-Dateien an.

In der Grundeinstellung erfolgt die Verbindung zu OTOBO über die Standardports: Port 443 für HTTPS und Port 80 für HTTP. Auch bei aktiviertem HTTPS, läuft die OTOBO Webapplikation via HTTP. Die HTTPS-Unterstützung erfolgt über einen zusätzlichen Reverseproxy, der als nginx-Service vorgeschaltet wird.

Für die folgenden Befehle gehen wir davon aus, dass HTTPS unterstützt werden soll.

```
docker_admin> cd /opt/otobo-docker
docker_admin> cp -p .docker_compose_env_https .env # or .docker_compose_env_http for␣
↪HTTP
```

4.2.3 3. Configure the password for the database admin user

Ändern Sie folgenden Wert in der .env-Datei:

`OTOBO_DB_ROOT_PASSWORD=<Ihr_geheimes_Passwort>`

Das Passwort für den Datenbankadministrator ist frei wählbar. Dieser wird benötigt, um den Datenbankbenutzer **otobo** und das Datenbankschema **otobo** anzulegen.

4.2.4 4. Set up a volume with SSL configuration for the nginx webproxy (optional)

Dieser Schritt kann entfallen, wenn nur per HTTP auf OTOBO zugegriffen werden soll.

nginx erfordert für die SSL-Verschlüsselung ein Zertifikat sowie einen privaten Schlüssel.

Bemerkung: Für Testzwecke und beim Entwickeln kann ein selbst signiertes Zertifikat verwendet werden. Grundsätzlich sind jedoch offizielle Zertifikate erforderlich.

See e.g. https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in on how to create self-signed certificates.

Bemerkung: Um in nginx eine CA-Chain zusammen mit einem Zertifikat anzugeben, muss das CA-Chain-File mit dem eigentlichen Zertifikat in eine Datei kopiert werden.

Das Zertifikat und der private Schlüssel werden in einem Volume hinterlegt, um später von nginx verwendet werden zu können. Legen Sie zunächst das Volume an und kopieren Sie anschließend Zertifikat und Schlüssel hinein:

```
docker_admin> docker volume create otobo_nginx_ssl
docker_admin> otobo_nginx_ssl_mp=$(docker volume inspect --format '{{ .Mountpoint }}'␣
↪otobo_nginx_ssl)
docker_admin> echo $otobo_nginx_ssl_mp  # just a sanity check
docker_admin> cp /PathToYourSSLCert/ssl-cert.crt /PathToYourSSLCert/ssl-key.key
↪$otobo_nginx_ssl_mp
```

Die Namen der kopierten Dateien müssen in die eben angelegte .env-Datei eingegeben werden. Z.B.

`OTOBO_NGINX_SSL_CERTIFICATE=/etc/nginx/ssl/ssl-cert.crt` und
`OTOBO_NGINX_SSL_CERTIFICATE_KEY=/etc/nginx/ssl/ssl-key.key`

Bitte passen Sie nur die Dateinamen an. Der Pfad /etc/nginx/ssl/ ist im Docker Image festgeschrieben.

4.2.5 5. Start the Docker containers with Docker Compose

Jetzt starten wir die Docker Container mit `docker-compose`. Standardmäßig werden die Docker Images von https://hub.docker.com/u/rotheross heruntergeladen.

```
docker_admin> docker-compose up --detach
```

Geben Sie folgende Befehle ein, um sicherzustellen, dass die sechs erforderlichen Dienste (fünf, wenn nur HTTP verwendet werden soll) wirklich laufen :

```
docker_admin> docker-compose ps
docker_admin> docker volume ls
```

4.2.6 6. Install and start OTOBO

Führen Sie den OTOBO Installer von http://IhreIPoderFQDN/otobo/installer.pl aus.

Bemerkung: Konfigurieren Sie OTOBO im Installer mit einer neuen MySQL-Datenbank. Geben Sie als Root-Passwort für die MySQL-Datenbank das Passwort ein, das Sie für die Variable `OTOBO_DB_ROOT_PASSWORD` in die .env-Datei eingegeben haben. Bitte behalten Sie den Wert `db` für den MySQL-Hostnamen unverändert bei.

Viel Vergnügen mit OTOBO!

Bemerkung: Um im laufenden Container in das OTOBO-Verzeichnis zu wechseln und wie gewohnt in der Kommandozeile zu arbeiten, können Sie folgenden Docker-Befehl nutzen: `docker-compose exec web bash`.

4.3 Ergänzende technische Informationen

Dieser Abschnitt bietet einige Zusatzinformationen zu den technischen Hintergründen.

4.3.1 Liste der Docker Container

Container otobo_web_1 OTOBO-Webserver über den internen Port 5000.

Container otobo_daemon_1 OTOBO Daemon. Der OTOBO Daemon wird gestartet und regelmäßig geprüft.

Container otobo_db_1 Betreibt die Datenbank MariaDB über den internen Port 3306.

Container otobo_elastic_1 Elasticsearch über die internen Ports 9200 und 9300.

Container otobo_redis_1 Betreibt Redis als Caching-Dienst.

Optionaler Container otobo_nginx_1 Setzt nginx als Reverseproxy für HTTPS-Verbindungen ein.

4.3.2 Übersicht über die Docker Volumes

Die Docker Volumes werden auf dem Host erstellt, um persistente Daten darin zu speichern. So können die Dienste ohne Datenverlust gestartet und gestoppt werden. Bedenken Sie, dass Container nur begrenzte Zeit verfügbar sind, Volumes dagegen dauerhaft.

otobo_opt_otobo enthält /opt/otobo in den Containern **web** and **daemon**.

otobo_mariadb_data beinhaltet /var/lib/mysql im Container db.

otobo_elasticsearch_data enthält /usr/share/elasticsearch/datal im Container **elastic**.

otobo_redis_data enthält Daten für den Container redis.

otobo_nginx_ssl enthält die TLS-Dateien, das Zertifikat und den privaten Schlüssel. Muss von Hand initialisiert werden.

4.3.3 Docker-Umgebungsvariablen

Bisher haben wir nur das Nötigste konfiguriert. In der .env-Datei können weitere Konfigurationen vorgenommen werden. Im Folgenden finden Sie eine Auswahl der wichtigsten Umgebungsvariablen.

MariaDB Einstellungen

OTOBO_DB_ROOT_PASSWORD Das Root-Passwort für MariaDB. Diese Einstellung wird für den Betrieb des db Service benötigt.

Elasticsearch Einstellungen

Damit Elasticsearch in Produktivumgebungen optimal läuft, sind einige Einstellungen vorzunehmen. Detaillierte Informationen entnehmen Sie bitte der Elasticsearch-Dokumentation unter https://www.elastic.co/guide/en/elasticsearch/reference/7.8/docker.html#docker-prod-prerequisites.

OTOBO_Elasticsearch_ES_JAVA_OPTS Beispieleinstellung: OTOBO_Elasticsearch_ES_JAVA_OPTS=-Xms512m -Xmx512m Bitte setzen Sie diesen Wert in Produktivumgebungen auf einen Wert bis 4g.

Webserver Einstellungen

OTOBO_WEB_HTTP_PORT Hier sind Angaben nötig, falls der HTTP-Port vom Standardport 80 abweicht. Ist HTTPS aktiv, wird der HTTP-Port auf HTTPS umgeleitet.

Einstellungen für den nginx-Webproxy

Diese Einstellungen werden angewendet, wenn HTTPS aktiv ist.

OTOBO_WEB_HTTP_PORT Hier sind Angaben nötig, falls der HTTP-Port vom Standardport 80 abweicht.

OTOBO_WEB_HTTPS_PORT Bitte anpassen, falls der HTTPS-Port vom Standardport 443 abweicht.

OTOBO_NGINX_SSL_CERTIFICATE SSL-Zertifikat für den nginx-Webproxy. Beispiel: OTOBO_NGINX_SSL_CERTIFICATE=/etc/nginx/ssl/acme.crt

OTOBO_NGINX_SSL_CERTIFICATE_KEY SSL-Schlüssel für den nginx-Webproxy. Beispiel: OTOBO_NGINX_SSL_CERTIFICATE_KEY=/etc/nginx/ssl/acme.key

nginx-Webproxy-Einstellungen für Kerberos

Diese Einstellungen werden von Nginx verwendet, wenn Kerberos für das Single Sign-on eingesetzt wird.

OTOBO_NGINX_KERBEROS_KEYTAB Kerberos Keytab-Datei. Standard ist /etc/krb5.keytab.

OTOBO_NGINX_KERBEROS_CONFIG Kerberos Config-Datei. Standard ist /etc/krb5.conf, usually generated from krb5.conf.template

OTOBO_NGINX_KERBEROS_SERVICE_NAME Kerberos Service Name. Es ist unklar ob diese Einstellung tatsächlich irgendwo verwendet wird.

OTOBO_NGINX_KERBEROS_REALM Kerberos REALM. Verwendet in /etc/krb5.conf.

OTOBO_NGINX_KERBEROS_KDC Kerberos kdc / AD Controller. Verwendet in /etc/krb5.conf.

OTOBO_NGINX_KERBEROS_ADMIN_SERVER Kerberos Admin Server. Verwendet in /etc/krb5.conf.

OTOBO_NGINX_KERBEROS_DEFAULT_DOMAIN Kerberos Default Domain. Verwendet in /etc/krb5.conf.

NGINX_ENVSUBST_TEMPLATE_DIR Angeben eines nginx-Konfigurationsvorlangenverzeichnis. Gewährleistet zusätzliche Flexibilität.

Docker Compose-Einstellungen

Diese Einstellungen werden direkt von Docker Compose verwendet.

COMPOSE_PROJECT_NAME Der Projektname wird als Präfix für die Docker Volumes und Container verwendet. Als Standardwert wird otobo verwendet, sodass die Container z.B. otobo_web_1 und otobo_db_1 heißen. Wenn Sie mehrere OTOBO Instanzen auf dem selben Server betreiben, passen Sie den Projektnamen bitte jeweils individuell an.

COMPOSE_PATH_SEPARATOR Separator für die Werte im COMPOSE_FILE

COMPOSE_FILE Verwenden Sie docker-compose/otobo-base.yml als Grundlage und ergänzen Sie die erforderlichen Extension Files wie docker-compose/otobo-override-http.yml oder docker-compose/otobo-override-https.yml.

OTOBO_IMAGE_OTOBO, OTOBO_IMAGE_OTOBO_ELASTICSEARCH, OTOBO_IMAGE_OTOBO_NGINX, ... Werden verwendet, um alternative Docker Images zu deklarieren. Nützlich zum Testen lokaler Builds oder aktuellerer Versionen der Images.

4.4 Weiterführende Themen

4.4.1 Individuelle Konfiguration des nginx-Webproxy

Der Container `otobo_nginx_1` bietet HTTPS Unterstützung über einen Nginx als Reverse Proxy. Das Docker Image des Containers besteht aus dem offiziellen Nginx Image, https://hub.docker.com/_/nginx, zusammen mit einer OTOBO spezifischen Nginx Konfiguration.

Die standardmäßige OTOBO-spezifische Konfiguration ist im Docker-Image unter /etc/nginx/template/otobo_nginx.conf.template zu finden. Tatsächlich handelt es sich dabei nur um eine Vorlage für die endgültige Konfiguration. Es gibt einen Prozess, der vom Nginx-Basisimage bereitgestellt wird, der die Makros in der Vorlage durch die entsprechenden Umgebungsvariablen ersetzt. Dieser Prozess wird ausgeführt, wenn der Container startet. In der Standardvorlagendatei werden die folgenden Makros verwendet:

OTOBO_NGINX_SSL_CERTIFICATE Um SSL zu konfigurieren.

OTOBO_NGINX_SSL_CERTIFICATE_KEY Um SSL zu konfigurieren.

OTOBO_NGINX_WEB_HOST Der intern benutzte HTTP Hostname.

OTOBO_NGINX_WEB_PORT Der intern benutzte HTTP Port.

Wie diese Konfigurationsmöglichkeit genutzt werden kann, um ein SSL-Zertifikat zu installieren, lesen Sie in Schritt 4..

Wenn die Standard-Makros nicht ausreichen, kann die Anpassung weiter gehen. Dies kann erreicht werden, indem die Standard-Konfigurationsvorlage durch eine angepasste Version ersetzt wird. Es ist am besten, nicht einfach die Konfiguration im laufenden Container zu ändern. Stattdessen erstellen wir zuerst ein persistentes Volume, das die angepasste Konfiguration enthält. Dann weisen wir den otobo_nginx_1 an, das neue Volume einzubinden und die angepasste Konfiguration zu verwenden.

Zuerst muss ein neues Volume erstellt werden. In den Beispielbefehlen benutzen wir das bestehende Template als Ausgangspunkt.

```
# stop the possibly running containers
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose down

# create a volume that is initially not connected to otobo_nginx_1
docker_admin> docker volume create otobo_nginx_custom_config

# find out where the new volume is located on the Docker host
docker_admin> otobo_nginx_custom_config_mp=$(docker volume inspect --format '{{ .
↪Mountpoint }}' otobo_nginx_custom_config)
docker_admin> echo $otobo_nginx_custom_config_mp  # just a sanity check
docker_admin> ls $otobo_nginx_custom_config_mp    # another sanity check

# copy the default config into the new volume
docker_admin> docker create --name tmp-nginx-container rotheross/otobo-nginx-
↪webproxy:latest-11_0
docker_admin> docker cp tmp-nginx-container:/etc/nginx/templates/otobo_nginx.conf.
↪template $otobo_nginx_custom_config_mp # might need 'sudo'
docker_admin> ls -l $otobo_nginx_custom_config_mp/otobo_nginx.conf.template # just␣
↪checking, might need 'sudo'
docker_admin> docker rm tmp-nginx-container

# adapt the file $otobo_nginx_custom_config_mp/otobo_nginx.conf.template to your needs
docker_admin> vim $otobo_nginx_custom_config_mp/otobo_nginx.conf.template
```

Nachdem das Volume eingerichtet wurde, muss die angepasste Konfiguration aktiviert werden. Das neue Volume wird in docker-compose/otobo-nginx-custom-config.yml eingerichtet. Daher muss diese Datei zu **COMPOSE_FILE** hinzugefügt werden. Dann muss Nginx angewiesen werden, die neue Konfiguration zu verwenden. Dies geschieht durch Setzen von **NGINX_ENVSUBST_TEMPLATE_DIR** in der Umgebung. Um dies zu erreichen, kommentiere die folgenden Zeilen in deiner .env-Datei aus oder füge sie hinzu:

```
COMPOSE_FILE=docker-compose/otobo-base.yml:docker-compose/otobo-override-https.
↪yml:docker-compose/otobo-nginx-custom-config.yml
NGINX_ENVSUBST_TEMPLATE_DIR=/etc/nginx/config/template-custom
```

Verwenden Sie den folgenden Befehl, um Informationen zur angepassten Docker-Compose-Konfiguration anzuzeigen:

```
docker_admin> docker-compose config | more
```

Schließlich können Sie die Container wieder starten:

```
docker_admin> docker-compose up --detach
```

Weitere Informationen dazu finden Sie auf https://hub.docker.com/_/nginx im Abschnitt zur Verwen-

dung von Umgebungsvariablen in der nginx-Konfiguration („Using environment variables in nginx configuration (new in 1.19)“).

4.4.2 Single Sign-On mit Kerberos-Unterstützung in Nginx

Kurzbeschreibung

Um die Authentifizierung mit Kerberos zu aktivieren, muss deine .env-Datei auf der Beispieldatei .docker_compose_env_https_kerberos basieren. Dies aktiviert die spezielle Konfiguration in docker-compose/otobo-override-https-kerberos.yml. Diese Docker-Compose-Konfigurationsdatei wählt ein Nginx-Image aus, das Kerberos unterstützt. Sie übermittelt auch einige Kerberos-spezifische Einstellungen als Umgebungsvariablen an den laufenden Nginx-Container. Diese Einstellungen sind oben aufgeführt.

As usual, the values for these setting can be specified in the .env file. Most of ghese setting will be used as replacement values for the template https://github.com/RotherOSS/otobo/blob/rel-11_0/scripts/nginx/kerberos/templates/krb5.conf.template . The replacement takes place during the startup of the container. In the running container the adapted config will be available in /etc/krb5.conf.

Es ist immer noch möglich, eine benutzerspezifische /etc/krb5.conf-Datei bereitzustellen. Dies kann durch das Einbinden eines Volumes erreicht werden, das /etc/krb5.conf im Container überschreibt. Dies wird erreicht, indem OTOBO_NGINX_KERBEROS_CONFIG in der .env-Datei gesetzt und die Mount-Direktive in docker-compose/otobo-override-https-kerberos.yml aktiviert wird.

/etc/krb5.keytab ist immer installationsspezifisch und muss daher immer vom Host-System eingehängt werden.

Kerberos SSO Installationsanleitung

Kerberos Single Sign On in der OTOBO Docker-Installation

4.4.3 Individuelle Ports definieren

Standardmäßig nutzt OTOBO die Ports 443 und 80 für HTTPS und HTTP. In manchen Systemen werden dieses Ports jedoch bereits von anderen Diensten verwendet. In diesem Fall können Sie über die Variablen OTOBO_WEB_HTTP_PORT und OTOBO_WEB_HTTPS_PORT in der .env-Datei andere Ports definieren.

4.4.4 Startvorgang gewisser Dienste überspringen

Das aktuelle Docker-Compose-Setup startet fünf, sechs wenn HTTPS aktiviert ist, Dienste. Aber es gibt gültige Anwendungsfälle, in denen einer oder mehrere dieser Dienste nicht benötigt werden. Das Hauptbeispiel ist, wenn die Datenbank nicht als Docker-Dienst, sondern als externe Datenbank laufen soll. Leider gibt es keine spezielle Docker-Compose-Option, um bestimmte Dienste zu überspringen. Aber die Option –scale kann für diesen Zweck missbraucht werden. Für eine Installation mit einer externen Datenbank könnte der folgende Befehl verwendet werden:

```
docker_admin> docker-compose up --detach --scale db=0
```

Selbstverständlich kann das selbe Ziel auch erreicht werden, indem man die Datei docker-compose/otobo-base.yml anpasst und die relevanten Service-Definitionen entfernt.

4.4.5 Native Installation vorbereiten

Please download the latest version of otobo-docker on a system that has internet access and where docker is installed. Then navigate to the following folder otobo-docker/docker-compose.

```
cd otobo-docker/docker-compose
```

Jetzt kannst du den folgenden Befehl ausführen, um alle Docker-Images aus einer bestimmten Datei herunterzuladen, in meinem Beispiel verwende ich otobo-base.yml.

```
for i in $(cat otobo-base.yml| grep image:| cut -d":" -f3,4 | sed -e "s/-//1" -e"s/\}/
↪/g"); do docker pull $i; docker save $i -o $(echo $i|sed "s/\//-/g").docker; done
```

Danach befinden sich die Images (.docker) im Docker-Compose-Ordner und können über z.B. SCP auf das Zielsystem hochgeladen werden.

Gehe auf dem Offline-Zielsystem in den Ordner, in dem die Docker-Images gespeichert sind. Und gib den folgenden Befehl ein, um sie nacheinander zu importieren.

Im folgenden Beispiel importiere ich das mariadb-Image:

```
docker load --input mariadb:10.5.docker
```

4.4.6 Docker Image individualisieren

Instead of editing the files under docker-compose/ and risking to overwrite your own options with the next update of the otobo-docker folder, it is advisable to create an extra YAML file where the specific services are overwritten with additional options.

Ein gängiges Beispiel wäre, den Datenbankcontainer von außen über den Port 3306 zugänglich zu machen. Dafür könntest du eine extra Docker-Compose-Datei erstellen, die so aussieht:

```
$ cat custom_db.yml
services:
  db:
    ports:
      - "0.0.0.0:3306:3306"
```

Jetzt müssen wir docker-compose sagen, dass unsere neue Datei einbezogen werden soll. Dafür musst du deine YAML-Datei zur COMPOSE_FILE-Variable in der .env-Datei hinzufügen, zum Beispiel:

```
COMPOSE_FILE=docker-compose/otobo-base.yml:docker-compose/otobo-override-http.
↪yml:custom_db.yml
```

Jetzt können wir docker-compose verwenden, um unseren Container neu zu erstellen

```
$ docker-compose stop # if otobo is running
$ docker-compose up -d
```

Mit diesem Verfahren kannst du jeden Service oder jedes Volume anpassen.

4.4.7 Docker Image individualisieren

Viele Anpassungen können im externen Volume otobo_opt_otobo vorgenommen werden, das dem Verzeichnis /opt/otobo im Docker Image entspricht. So können Sie z. B. lokale Perl-Module in

/opt/otobo/local installieren. Der Vorteil dieser Vorgehensweise besteht darin, dass das Image selbst dabei nicht verändert wird.

```
$ docker exec -it ${COMPOSE_PROJECT_NAME:=otobo}_web_1 bash
otobo@ce36ff89e637:~$ pwd
/opt/otobo
otobo@ce36ff89e637:~$ cpanm -l local Acme::123
--> Working on Acme::123
Fetching http://www.cpan.org/authors/id/N/NA/NATHANM/Acme-123-0.04.zip ... OK
Configuring Acme-123-0.04 ... OK
Building and testing Acme-123-0.04 ... OK
Successfully installed Acme-123-0.04
1 distribution installed
otobo@ce36ff89e637:~$
```

Das Schöne an diesem Ansatz ist, dass das Docker-Image selbst nicht modifiziert werden muss.

Zusätzliche Debian-Pakete zu installieren ist etwas aufwändiger. Sie können dazu entweder ein individuelles Dockerfile auf Grundlage des OTOBO Image erzeugen. Oder Sie erstellen das angepasste Image direkt aus einem laufenden Container heraus. Nutzen Sie dazu den Befehl docker commit – mehr dazu unter https://docs.docker.com/engine/reference/commandline/commit/. Eine gute Anleitung mit Beispielen (auf Englisch) finden Sie auch unter https://phoenixnap.com/kb/how-to-commit-changes-to-docker-image.

Die zweite Vorgehensweise birgt zwei Probleme: Erstens wird das Image otobo standardmäßig mit dem Benutzer otobo und der UID 1000 ausgeführt. Der User otobo ist aber nicht berechtigt, Systempakete zu installieren. Der erste Schritt zur Lösung besteht deshalb darin, beim Ausführen des Images die Option‘–user root‘ zu übergeben. Zweitens bricht das Standard-Skript zur Definition des Einstiegspunktes /opt/otobo_install/entrypoint.sh sofort ab, wenn es als root aufgerufen wird. Das wurde so angelegt, um zu verhindern, dass der Vorgang unbeabsichtigt als root ausgeführt wird. Der zweite Schritt zur Lösung besteht deshalb darin, ein anderes Entrypoint-Skript anzugeben, dem egal ist, von wem es aufgerufen wird. Wir nutzen folgende Beispielbefehle, um Fortune Cookies für OTOBO zu setzen:

Pullen Sie ein getaggtes OTOBO Image (sofern nicht bereits vorhanden), und prüfen Sie, ob im Image schon Fortune Cookies bereitgestellt werden:

```
$ docker run rotheross/otobo:rel-11_0_2 /usr/games/fortune
/opt/otobo_install/entrypoint.sh: line 57: /usr/games/fortune: No such file or␣
↪directory
```

Ergänzen Sie die Fortune Cookies in einem benannten Container im originalen OTOBO Image. Dazu nutzen wir eine interaktive Session und den User root:

```
$ docker run -it --user root --entrypoint /bin/bash --name otobo_orig rotheross/
↪otobo:rel-11_0_2
root@50ac203409eb:/opt/otobo# apt update
root@50ac203409eb:/opt/otobo# apt install fortunes
root@50ac203409eb:/opt/otobo# exit
$ docker ps -a | head
```

Erzeugen Sie ein Image auf Basis des gestoppten Containers und vergeben Sie einen Namen. Beachten Sie, dass der Standardbenutzer und das Entrypoint-Skript wiederhergestellt werden müssen:

```
$ docker commit -c 'USER otobo'  -c 'ENTRYPOINT ["/opt/otobo_install/entrypoint.sh"]'␣
↪otobo_orig otobo_with_fortune_cookies
```

Schließlich prüfen wir, ob alles funktioniert hat:

```
$ docker run otobo_with_fortune_cookies /usr/games/fortune
A platitude is simply a truth repeated till people get tired of hearing it.
                -- Stanley Baldwin
```

Das modifizierte Image kann in Ihrer .env-Datei angegeben und frei verwendet werden.

4.4.8 Lokale Images erstellen

Bemerkung: Lokale Docker Images werden in der Regel nur zur Entwicklung erstellt. Weitere Anwendungsfälle sind z.B. wenn aktuellere Basis Images benötigt werden oder wenn Images um zusätzliche Funktionalität erweitert werden sollen.

Die zum Erstellen lokaler Docker Images benötigten Dateien finden Sie im Git Repository unter https://github.com/RotherOSS/otobo:

- otobo.web.dockerfile
- otobo.nginx.dockerfile
- otobo.elasticsearch.dockerfile

Das Script zum Erstellen der Images finden Sie unter bin/docker/build_docker_images.sh.

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo.git
docker_admin> cd otobo
docker_admin> # checkout the wanted branch. e.g. git checkout rel-11_0
docker_admin> # modify the docker files if necessary
docker_admin> bin/docker/build_docker_images.sh
docker_admin> docker image ls
```

Lokal erstellte Images werden auf Basis der in der Datei RELEASE angegebenen Versionsdaten als `local-<OTOBO_VERSION>` getaggt.

Nach dem Erstellen der lokalen Images können Sie in das docker-compose Verzeichnis zurückkehren. Neu erstellte Images können mit den entsprechenden Vorgaben in `OTOBO_IMAGE_OTOBO`, `OTOBO_IMAGE_OTOBO_ELASTICSEARCH`, `OTOBO_IMAGE_OTOBO_NGINX` in .env ausgewählt werden.

4.4.9 Automatische Installation

Anstatt die URL http://yourIPorFQDN/otobo/installer.pl zu verwenden, können Sie auch eine Abkürzung wählen. Diese Möglichkeit eignet sich besonders, um die Testsuite auf einem frisch installierten System laufen zu lassen.

Warnung: `docker-compose down -v` löscht alle vorhandenen Einstellungen und Daten.

```
docker_admin> docker-compose down -v
docker_admin> docker-compose up --detach
docker_admin> docker-compose stop daemon
docker_admin> docker-compose exec web bash\
-c "rm -f Kernel/Config/Files/ZZZAAuto.pm ; bin/docker/quick_setup.pl --db-password␣
↪otobo_root"
```

```
docker_admin> docker-compose exec web bash\
-c "bin/docker/run_test_suite.sh"
.......
docker_admin> docker-compose start daemon
```

4.4.10 Einige nützliche Befehle

Docker

- `docker system prune -a` Systembereinigung (entfernt alle unbenutzten Images, Container, Volumes, Netzwerke)
- `docker version` Version anzeigen
- `docker build --tag otobo --file=otobo.web.Dockerfile .` Image erstellen
- `docker run --publish 80:5000 otobo` Neues Image ausführen
- `docker run -it -v opt_otobo:/opt/otobo otobo bash` Login auf neuem Image
- `docker run -it -v opt_otobo:/opt/otobo --entrypoint bash otobo` falls das Skript entrypoint.sh nicht funktioniert
- `docker ps` Laufende Images anzeigen
- `docker images` Verfügbare Images anzeigen
- `docker volume ls` Volumes auflisten
- `docker volume inspect otobo_opt_otobo` Informationen über ein Volume ausgeben
- `docker volume inspect --format '{{ .Mountpoint }}' otobo_nginx_ssl` Volume Mountpoint ausgeben
- `docker volume rm tmp_volume` Volume entfernen
- `docker inspect <container>` Informationen zu einem Container anzeigen
- `docker save --output otobo.tar otobo:latest-11_0 && tar -tvf otobo.tar` list files in an image
- `docker exec -it nginx-server nginx -s reload` nginx-Reload

Docker Compose

- `docker-compose config` Informationen zur Konfiguration anzeigen
- `docker-compose ps` Informationen zu laufenden Containern anzeigen
- `docker-compose exec nginx nginx -s reload` nginx-Reload

4.5 Ressourcen

Und hier ist eine sehr subjektive Sammlung von Links.

Grundsätzliche Informationen und Tutorials

- Perl Maven
- Dockerfile Best Practices
- Umgebung

Tips und Hinweise

- Cleanup
- Docker Host IP
- ‘Self signed certificate <https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu >‘_

Fehlersuche

- Docker Cache Entwertung
- tcpdump verwenden
- Informationen zu fehlgeschlagenen Builds

CHAPTER 5

Migration von OTRS / ((OTRS)) Community Edition Version 6 oder 7 auf OTOBO 10.1

Warnung: Bitte migrieren Sie zuerst Ihr OTRS auf OTOBO Version 10.1 und aktualisieren Sie dann Ihr OTOBO auf Version 11.

Hallo und danke, dass Sie sich für OTOBO entschieden haben!

OTRS, ((OTRS)) Community Edition und OTOBO sind Systeme mit umfassender Funktionalität, die große Flexibilität bieten. Deshalb erfordert jede Migration sorgfältige Vorbereitung und möglicherweise auch Nacharbeiten.

Bitte nehmen Sie sich deshalb Zeit für die Migration und befolgen Sie die Anleitung Schritt für Schritt.

Probleme oder Fragen? Kein Grund zu Verzweifeln :) Rufen Sie unsere Support-Hotline an, schreiben Sie uns eine E-Mail oder bitten Sie im OTOBO Community Forum unter https://forum.otobo.org/ um Unterstützung. Wir werden einen Weg finden, Ihnen zu helfen!

Bemerkung: Nach der Migration werden alle bisher in OTRS 6 verfügbaren Daten in OTOBO 10 verfügbar sein. Die Daten in Ihrem OTRS-Ursprungssystem werden während der Migration nicht verändert.

5.1 Übersicht über die unterstützen Migrationsmöglichkeiten

Mit dem OTOBO Migrationstool können folgende Migrationen durchgeführt werden:

1. Die grundsätzliche Migrationsstrategie.

   Dies ist der reguläre Weg, um eine Migration durchzuführen. Viele weitere Varianten sind möglich:

   Wechsel des Servers: Eine Migration mit gleichzeitigem Wechsel zu einem neuen Applikationsserver.

Trennung von Applikation und Webserver: Sie haben die Möglichkeit, den Anwendungs- und Datenbankserver auf einem gemeinsamen oder zwei verschiedenen Servern laufen lassen – unabhängig davon, wie das beim OTRS- / ((OTRS)) Community Edition-Vorgängersystem war.

Verschiedene Datenbanken: Sie können von jeder der unterstützten Datenbanken zu einer anderen migrieren.

Wechsel des Betriebssystems: Sie können während der Migration von jedem unterstützten Betriebssystem zu jedem anderen unterstützten Betriebssystem wechseln.

Docker: Auch die Migration zu einer Docker-basierten OTOBO 10-Installation ist möglich.

2. Eine Variante des allgemeinen Vorgehens, bei der die Datenbankmigration gestreamlint durchgeführt wird.

   Verwende die ETL-ähnliche Migration, wenn die Quelldatenbank nicht durch erhöhte Last beeinträchtigt werden darf oder wenn der Zugriff auf die Quelldatenbank ein Engpass ist. In der allgemeinen Strategie werden die Daten zunächst Zeile für Zeile aus der otrs-Datenbank gelesen und dann in die OTOBO-Datenbank eingefügt. In dieser Variante werden die kompletten otrs-Datenbanktabellen zuerst exportiert, dann transformiert und anschließend in die otobo-Datenbank importiert.

3. Migration von einer Oracle-basierten OTRS 6 Installation zu einer Oracle-basierten OTOBO Installation.

   Dies ist ein Sonderfall der nicht durch die grundsätzliche Migrationsstrategie beachtet wird. Dies bedeutet, dass eine Variante der gestreamlinten Strategie verwendet werden muss.

Warnung: Alle Strategien funktionieren sowohl für Docker-basierte als auch für native Installationen. Bei der Docker-basierten Installation gibt es einige Besonderheiten zu beachten. Sie werden in den optionalen Schritten behandelt.

Bemerkung: Es ist auch möglich, die OTRS Datenbank vor der Migration auf den OTOBO Datenbankserver zu klonen. Auf diese Weise kann die allgemeine Migrationsstrategie beschleunigt werden.

5.2 Migrationsvoraussetzungen

1. Grundvoraussetzung für eine Migration ist, dass Sie bereits eine laufende ((OTRS)) Community Edition oder ein OTRS 6.0.* haben und dass deren Konfiguration und Daten in OTOBO übernommen werden sollen.

Warnung: Bitte überlegen Sie genau, ob es wirklich sinnvoll ist, bestehende Daten und Konfiguration zu übernehmen. Unsere Erfahrung zeigt, dass die bisher genutzte Installation und Konfiguration in viele Fällen eher suboptimal und ein sauberer Neustart die bessere Option ist. Auch kann es sinnvoll sein, nur die Ticketdaten zu übertragen und die Grundkonfiguration gemäß OTOBO Best Practice vorzunehmen. Hierzu beraten wir Sie gerne, wenden Sie sich einfach per Mail an hallo@otobo.de oder stellen Sie Ihre Fragen im OTOBO Community Forum unter https://forum.otobo.org/.

2. Sie benötigen eine betriebsbereite OTOBO-Installation als Ausgangpunkt für die Migration!

3. Auf dieser OTOBO-Instanz müssen alle OPM-Pakete installiert sein, die Sie bisher in Ihrem OTRS nutzen und künftig in OTOBO nutzen möchten.
4. Wenn Sie auf einen anderen Server migrieren möchten, muss der OTOBO-Webserver in der Lage sein, auf das Verzeichnis zuzugreifen, in dem die ((OTRS)) Community Edition oder OTRS 6.0.* installiert sind. Meist ist dies das Verzeichnis /opt/otrs auf dem Server, auf dem das OTRS läuft. Der Lesezugriff erfolgt über SSH oder Dateisystem-Mounts.
5. Die otrs-Datenbank muss von dem Server aus erreichbar sein, auf dem OTOBO läuft. Externe Hosts müssen Lesezugriff haben. Wo kein Zugriff möglich ist oder die Migration beschleunigt werden soll, kann auch mit einem Datenbank-Dump gearbeitet werden.

Bemerkung: Sind SSH-Kommunikation und Datenbankzugriff von einem Server auf den anderen nicht möglich, migrieren Sie zunächst auf dem alten Server von OTRS zu OTOBO und ziehen die neue Instanz erst nachträglich um.

5.3 Schritt 1: Neues OTOBO-System installieren

Bitte beginnen Sie damit, das neue OTOBO-System zu installieren, auf das Sie Ihre OTRS / ((OTRS)) Community Edition anschließend migrieren möchten. Wir empfehlen dazu dringend, das Kapitel OTOBO Installation zu lesen. Installieren Sie Ihr System mit Docker, ist das Kapitel Installation mit Docker und Docker Compose relevant für Sie.

Warnung: Unter Apache kann es zu Problemen kommen, wenn zwei voneinander unabhängige Anwendungen zeitgleich unter mod_perl auf demselben Server laufen. Abhilfe schafft hier die mod_perl-Einstellung `PerlOptions +Parent`. Sie sorgt dafür, dass die Anwendungen jeweils eigene Perl-Interpreter verwenden. Bitte prüfen Sie Ihre Apache-Konfigurationsdateien im Verzeichnis /etc/apache2/sites-available und ergänzen Sie die Einstellung, falls sie noch nicht gesetzt ist.

Sobald die Installation abgeschlossen ist, melden Sie sich bitte in OTOBO als root@localhost an und öffnen im Admin-Bereich die Paketverwaltung (`Administration -> Paketverwaltung`). Dort können Sie alle benötigten OTOBO OPM-Pakete installieren.

Folgende OPM-Pakete und OTRS-„Feature Addons“ müssen und sollten NICHT installiert werden, da ihre Funktiona

- OTRSHideShowDynamicField
- RotherOSSHideShowDynamicField
- TicketForms
- RotherOSS-LongEscalationPerformanceBoost
- Znuny4OTRS-AdvancedDynamicFields
- Znuny4OTRS-AutoSelect
- Znuny4OTRS-EscalationSuspend
- OTRSEscalationSuspend
- OTRSDynamicFieldDatabase
- OTRSDynamicFieldWebService

- OTRSBruteForceAttackProtection
- Znuny4OTRS-ExternalURLJump
- Znuny4OTRS-QuickClose
- Znuny4OTRS-AutoCheckbox
- OTRSSystemConfigurationHistory
- Znuny4OTRS-PasswordPolicy

Die folgenden OTOBO-Pakete sind in OTOBO 11.0 integriert worden. Das bedeutet, dass sie nicht im Zielsystem installiert werden sollten, wenn das Zielsystem OTOBO 11 ist.

- ImportExport

5.4 Schritt 2: `SecureMode` in OTOBO deaktivieren

Bitte rufen Sie nach Abschluss der Installation erneut den OTOBO-Admin-Bereich auf, navigieren Sie zu `Administration -> Systemkonfiguration` und deaktivieren Sie die Konfig-Option `SecureMode`.

Bemerkung: Bitte vergessen Sie nicht, die geänderte Einstellung anschließend in Betrieb zu nehmen.

5.5 Schritt 3: OTOBO-Daemon stoppen

Dies ist nötig, wenn der OTOBO Daemon aktuell läuft. Das Vorgehen, um den OTOBO Daemon zu stoppen unterscheidet sich bei Docker-basierten und nicht-Docker-basierten (nativen) Installationen.

Bei nicht-Docker-basierten Installationen führen Sie die folgenden Befehle als User otobo aus:

```
# in case you are logged in as root
root> su - otobo

otobo> /opt/otobo/bin/Cron.sh stop
otobo> /opt/otobo/bin/otobo.Daemon.pl stop --force
```

Läuft OTOBO in Docker, genügt es, den Dienst `daemon` zu stoppen:

```
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose stop daemon
docker_admin> docker-compose ps     # otobo_daemon_1 should have exited with the code␣
↪0
```

Bemerkung: Wir empfehlen, an dieser Stelle ein Backup des gesamten OTOBO-Systems zu erstellen. Treten dann während der Migration Probleme auf, müssen Sie nicht den gesamten Installationsprozess erneut durchlaufen, sondern können einfach das Backup importieren und eine neue Migration anstoßen.

Siehe auch:

Hierzu empfehlen wir die Lektüre des Kapitels OTOBO Backup und Wiederherstellung.

5.6 Optionaler Schritt: Für leichteren Zugriff /opt/otrs einhängen

In vielen Fällen soll OTOBO auf einem neuen Server laufen, auf dem /opt/otrs noch nicht zur Verfügung steht. In diesen Fällen kann das /opt/otrs-Verzeichnis des OTRS-Servers in das Dateisystem auf dem OTOBO-Server eingehängt werden. Falls ein normales Einhängen über das Netzwerk nicht möglich ist, können Sie beispielsweise auf `sshfs` zurückgreifen.

5.7 Optionaler Schritt: Installieren von `sshpass` und `rsync`, wenn /opt/otrs über SSH kopiert werden soll

Dieser Schritt ist nur dann nötig, wenn Sie OTRS von einem anderen Server migrieren möchten und /opt/otrs des OTRS-Servers nicht auf dem OTOBO-Server eingehängt wurde.

Wir verwenden die Tools `sshpass` und `rsync`, um Dateien über eine SSH-Verbindung zu kopieren. Bitte melden Sie sich als `root`-Benutzer am Server an und führen Sie einen der folgenden Befehle aus:

```
$ # Install sshpass under Debian / Ubuntu Linux
$ sudo apt-get install sshpass
```

```
$ #Install sshpass under RHEL/CentOS Linux
$ sudo yum install sshpass
```

```
$ # Install sshpass under Fedora
$ sudo dnf install sshpass
```

```
$ # Install sshpass under OpenSUSE Linux
$ sudo zypper install sshpass
```

Gehen Sie analog vor, wenn rsync noch nicht verfügbar ist.

5.8 Schritt 4: OTRS / ((OTRS)) Community Edition Quellsystem vorbereiten

---

Bemerkung: Stellen Sie sicher, dass für Ihr OTRS / ((OTRS)) Community Edition ein aktuelles und vollständiges Backup vorliegt. Selbst wenn die Daten im Quellsystem während der Migration nicht angetastet werden, kann unter Umständen schon ein einziger falscher Eintrag Probleme bereiten.

Damit sind die grundlegenden Voraussetzungen für die Migration geschaffen. Stellen Sie nun sicher, dass keine Tickets mehr bearbeitet werden und sich Benutzer nicht mehr in OTRS anmelden können:

Rufen Sie im OTRS-Admin-Bereich `Administration -> Systemwartung` auf und fügen Sie ein neues Systemwartungs-Zeitfenster über einige Stunden hinzu. Löschen Sie anschließend alle Agenten- und User-Sitzungen (`Administration -> Sitzungsverwaltung`) und melden Sie sich ab.

5.8.1 Alle relevanten Services und den OTRS Daemon stoppen

Versichern Sie sich, dass keine Dienste oder Crons Jobs mehr ausgeführt werden.

```
root> su - otrs
otrs> /opt/otrs/bin/Cron.sh stop
otrs> /opt/otrs/bin/otrs.Daemon.pl stop --force
```

5.8.2 Löschen Sie die Caches und die Betriebsdaten

Die zwischengespeicherten Daten und die Betriebsdaten müssen nicht migriert werden. Die Mail-Warteschlange sollte zu diesem Zeitpunkt bereits leer sein.

```
root> su - otrs
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Cache::Delete
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Session::DeleteAll
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Loader::CacheCleanup
otrs> /opt/otrs/bin/otrs.Console.pl Maint::WebUploadCache::Cleanup
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Email::MailQueue --delete-all
```

5.9 Optionaler Schritt für Docker: Erforderliche Daten im Container verfügbar machen

Bei der Nutzung von Docker zur OTOBO-Installation sind einige Besonderheiten zu beachten. Die wichtigste: In einem Docker-Container ausgeführte Prozesse können grundsätzlich nicht auf Verzeichnisse außerhalb dieses Containers zugreifen. Einzige Ausnahme: Auf Verzeichnisse, die als Volumes in den Container gemounted werden, kann zugegriffen werden. Auch die in "otobo_db_1"laufende MariaDB-Datenbank ist von außerhalb des Container-Netzwerks nicht direkt erreichbar.

Bemerkung: Wir gehen im Folgenden davon aus, dass zur Interaktion mit Docker der Benutzer **docker_admin** verwendet wird. Der Docker-Admin kann entweder der **root**-Benutzer des Docker-Host oder ein spezieller Benutzer mit den erforderlichen Berechtigungen sein.

5.9.1 /opt/otrs in das Volume otobo_opt_otobo kopieren

Wir gehen in diesem Abschnitt davon aus, dass das OTRS Home-Verzeichnis /opt/otrs auf dem Docker-Host verfügbar ist.

Sie haben mindestens zwei Möglichkeiten:

1. /opt/otrs in das bestehende Volume otobo_opt_otobo kopieren
2. /opt/otrs als zusätzliches Volume mounten

Konzentrieren wir uns hier auf Option **a.**.

Als erstes müssen wir herausfinden, wo das Volume otobo_opt_otobo auf dem Docker-Host bereitsteht.

```
docker_admin> otobo_opt_otobo_mp=$(docker volume inspect --format '{{ .Mountpoint }}'␣
↪otobo_opt_otobo)
docker_admin> echo $otobo_opt_otobo_mp  # just a sanity check
```

Wir nutzen `rsync` für einen sicheren Kopiervorgang. Je nach Dockerumgebung muss `rsync` möglicherweise mit `sudo`-Rechten ausgeführt werden.

```
docker_admin> # when docker_admin is root
docker_admin> rsync --recursive --safe-links --owner --group --chown 1000:1000 --
↪perms --chmod "a-wx,Fu+r,Du+rx" /opt/otrs/ $otobo_opt_otobo_mp/var/tmp/copied_otrs
docker_admin> ls -la $otobo_opt_otobo_mp/var/tmp/copied_otrs  # just a sanity check

docker_admin> # when docker_admin is not root
docker_admin> sudo rsync --recursive --safe-links --owner --group --chown 1000:1000 --
↪perms --chmod "a-wx,Fu+r,Du+rx" /opt/otrs/ $otobo_opt_otobo_mp/var/tmp/copied_otrs
docker_admin> sudo ls -la $otobo_opt_otobo_mp/var/tmp/copied_otrs  # just a sanity␣
↪check
```

Das kopierte Verzeichnis wird im Container als /opt/otobo/var/tmp/copied_otrs verfügbar sein.

5.10 Schritt 5: Migration durchführen!

Bitte verwenden Sie das Web-Migrationstool, das Sie unter http://localhost/otobo/migration.pl finden (ersetzen Sie „localhost“ durch Ihren OTOBO Host und ergänzen Sie ggf. den Port). Der Assistent führt Sie Schritt für Schritt durch den Prozess.

Warnung: Gelegentlich wird eine Warnung angezeigt, dass die Deaktivierung des **SecureMode** nicht erkannt wurde. Bitte starten Sie in diesem Fall den Webserver neu. So wird die aktuelle Konfiguration eingelesen.

```
# native installation
root> service apache2 restart

# Docker-based installation
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose restart web
docker_admin> docker-compose ps     # otobo_web_1 should be running again
```

Bemerkung: Wird OTOBO in einem Docker Container ausgeführt, behalten Sie die Standardeinstellungen localhost für den OTRS-Server und /opt/otobo/var/tmp/copied_otrs für das OTRS-Home-Verzeichnis bei. Dieser Pfad führt zu den im optionalen Schritt kopierten Daten.

Bemerkung: Die Standardwerte für den OTRS-Datenbankbenutzer und dessen Passwort werden der Kernel/Config.pm im OTRS-Homeverzeichnis entnommen. Ändern Sie die vorgeschlagenen Einstellungen, wenn Sie einen speziellen Datenbankbenutzer für die Migration verwenden möchten. Passen Sie die Einstellungen auch dann an, wenn Sie mit einer Datenbank arbeiten, die in den otobo_db_1 Docker Container kopiert wurde.

Bemerkung: Wenn Sie Docker nutzen, kann die Datenbank auf dem Docker-Host nicht über `127.0.0.1` erreicht werden. Die Einstellung `127.0.0.1``ist deshalb kein gültiger Wert für das Eingabefeld ``OTRS Server`. Geben Sie stattdessen eine der alternativen IP-Adressen an, die Sie mit `hostname --all-ip-addresses` für `OTRS Server` ermittelt haben.

Bemerkung: Häufig kann das System nach der Migration auf einen neuen Anwendungsserver oder nach einer Docker-basierten Installation nicht auf die Datenbank zugreifen. Das liegt meist daran, dass der OTOBO-Datenbanknutzer sich nur von dem Host aus verbinden kann, auf dem auch die Datenbank läuft. Um dennoch Zugriff auf die Datenbank zu gewährleisten, empfehlen wir die Anlage eines speziellen Datenbankbenutzers für die Migration – z. B. `CREATE USER 'otrs_migration'@'%' IDENTIFIED BY 'otrs_migration';` und `GRANT SELECT, SHOW VIEW ON otrs.* TO 'otrs_migration'@'%';`. Dieser Benutzer kann nach der Migration wieder gelöscht werden: `DROP USER 'otrs_migration'@'%'`.

Benutzerdefinierte Einstellungen in Kernel/Config.pm werden von der alten OTRS-Installation auf die neue OTOBO-Installation übertragen. Wenn du benutzerdefinierte Einstellungen hast, schau dir bitte die migrierte Datei /opt/otobo/Kernel/Config.pm an. Du möchtest vielleicht benutzerdefinierte Pfade oder LDAP-Einstellungen anpassen. Im besten Fall stellst du fest, dass einige benutzerdefinierte Einstellungen nicht mehr benötigt werden.

Ist die Migration abgeschlossen, nehmen Sie sich bitte Zeit, um das System ausgiebig zu testen. Erst wenn Sie sicher sind, dass die Migration erfolgreich durchgeführt wurde und dass Sie von nun an mit OTOBO arbeiten möchten, starten Sie den OTOBO Daemon:

```
root> su - otobo
otobo>
otobo> /opt/otobo/bin/Cron.sh start
otobo> /opt/otobo/bin/otobo.Daemon.pl start
```

In der Docker-Umgebung:

```
docker_admin> cd ~/otobo-docker
docker_admin> docker-compose start daemon
```

5.11 Schritt 6: Nach der erfolgreichen Migration!

1. Deinstallieren Sie `sshpass`, wenn Sie es nicht mehr benötigen.
2. Löschen Sie ggf. speziell für die Migration angelegte Datenbanken und Datenbankbenutzer.
3. Viel Vergnügen mit OTOBO!

5.12 Bekannte Probleme bei der Migration

5.12.1 1. Login after migration not possible

Während unserer Migrationstests kam es gelegentlich zu Problemen mit dem für die Migration verwendeten Browser. Diese waren in der Regel nach einem Browser-Neustart gelöst. In Safari kann es erforderlich sein, die alte OTRS-Sitzung manuell zu löschen.

5.12.2 2. Final page of the migration has a strange layout due to missing CSS files

Kann vorkommen, wenn die Einstellung ScriptAlias keinen Standardwert enthält. Bei der Migration wird lediglich otrs durch otobo ersetzt. Das kann dazu führen, dass OTOBO nicht mehr korrekt auf CSS und

JavaScript zugreifen kann. Bitte überprüfen Sie in diesem Fall die Einstellungen in Kernel/Config.pm und korrigieren Sie diese auf geeignete Werte.

5.12.3 3. Migration stops due to MySQL errors

Auf Systemen, bei denen es in der Vergangenheit Probleme mit Upgrades gab, wird der Migrationsprozess unter Umständen wegen MySQL-Fehlern in den Tabellen ticket und ticket_history unterbrochen. Typischerweise stammen diese Fehler von NULL-Werten in der Quell-Tabelle, die in der Ziel-Tabelle nicht mehr erlaubt sind. Diese Konflikte müssen manuell behoben werden, bevor die Migration fortgesetzt werden kann.

There is a check in migration.pl that checks for NULL values before the data transfer is done. Note, that the resolution still needs to be performed manually.

5.12.4 4. Errors in Step 5 when migrating to PostgreSQL

In diesen Fällen wird von migration.pl die nicht so hilfreiche Nachricht „System konnte die Datenübertragung nicht abschließen." angezeigt. Das Apache-Logfile und das OTOBO-Logfile zeigen eine aussagekräftigere Nachricht: „Message: ERROR: permission denied to set parameter „session_replication_role", SQL: ‚set session_replication_role to replica;'". Um dem Datenbankbenutzer **otobo** die benötigten Superuser-Rechte zu geben, führe den folgenden Befehl als PostgreSQL-Admin aus: `ALTER USER otobo WITH SUPERUSER;`. Versuche dann erneut, http://localhost/otobo/migration.pl auszuführen. Nach der Migration kehre zum normalen Zustand zurück, indem du `ALTER USER otobo WITH NOSUPERUSER` ausführst.

Es ist bisher nicht geklärt, ob die erweiterten Berechtigungen in jedem Setup gewährt werden müssen.

Siehe auch:

Die Diskussion in https://otobo.de/de/forums/topic/otrs-6-mysql-migration-to-otobo-postgresql/.

5.12.5 5. Problems with the Deployment the Merged System Configuration

Die Systemkonfiguration wird migriert, nachdem die Datenbanktabellen migriert wurden. In diesem Kontext bedeutet Migration das Zusammenführen der Standardeinstellungen von OTOBO mit der Systemkonfiguration des Quell-OTRS-Systems. In diesem Schritt können Inkonsistenzen auftreten. Ein Beispiel aus der Praxis ist die Einstellung `Ticket::Frontend::AgentTicketQuickClose###State`. Diese Einstellung ist neu in OTOBO 10 und der Standardwert ist der Zustand `closed successful`. Aber diese Einstellung ist ungültig, wenn der Zustand `closed successful` im Quellsystem gelöscht oder umbenannt wurde. Diese Inkonsistenz wird im Migrationsschritt **Migrate configuration settings** als Fehler erkannt. Tatsächlich wird die zusammengeführte Systemkonfiguration in der Datenbank gespeichert, aber zusätzliche Gültigkeitsprüfungen werden während der Bereitstellung durchgeführt.

Das Problem muss manuell mit Hilfe der OTOBO Konsolenbefehle gelöst werden.

- Auflistung der Inkonsistenzen mit Hilfe des Befehls `bin/otobo.Console.pl Admin::Config::ListInvalid`
- Ungültige Werte können mit Hilfe des Befehls `bin/otobo.Console.pl Admin::Config::FixInvalid` interaktiv behoben werden
- Nehmen Sie die gesammelten Änderungen durch migration.pl inklusive des deaktivierten **Secure-Mode** mit `bin/otobo.Console.pl Maint::Config::Rebuild` in Betrieb

Nach diesen manuellen Schritten sollten Sie migration.pl erneut ausführen können. Die Migration setzt dann an dem Schritt fort, bei dem der Fehler aufgetreten ist.

5.13 Schritt 7: Manuelle Aufgaben und Anpassungen

5.13.1 1. Password policy rules

Mit OTOBO 10 wurde standardmäßig eine neue Passwortrichtlinie für Agenten und Kundenbenutzer eingeführt, wenn diese sich lokal authentifizieren. Die Regeln der Passwortrichtlinie können Sie in der Systemkonfiguration anpassen (`PreferencesGroups###Password` und `CustomerPersonalPreference###Password`).

Regel Passwortrichtlinie	Standard
`PasswordMinSize`	8
`PasswordMin2Lower2UpperCharacters`	Ja
`PasswordNeedDigit`	Ja
`PasswordHistory`	10
`PasswordTTL`	30 Tage
`PasswordWarnBeforeExpiry`	5 Tage
`PasswordChangeAfterFirstLogin`	Ja

5.13.2 2. Under Docker: Manually migrate cron jobs

Wird OTOBO nicht unter Docker installiert, gibt es mindestens einen Cron-Job, der den Zustand des Daemon überprüft. Unter Docker gibt es diesen Cron-Job nicht mehr. Darüber hinaus läuft in keinem der Docker-Container ein Cron-Daemon. Für OTRS-Systeme mit kundenspezifischen Cron-Jobs (z. B. Datenbank-Backups) müssen deshalb individuelle Lösungen gefunden werden.

5.14 Spezielle Themen

5.14.1 Migration von Oracle zu Oracle

Für die Migration zu Oracle muss die ETL-ähnliche Strategie angewendet werden. Dies liegt daran, dass Oracle keine einfache Möglichkeit bietet, Fremdschlüsselprüfungen vorübergehend zu deaktivieren.

Auf dem OTOBO Server müssen ein Oracle Client und das Perl Modul `DBD::Oracle` installiert sein.

Bemerkung: Falls der Oracle Instant Client genutzt wird, ist zusätzlich das optionale SDK nötig, um DBD::Oracle zu installieren.

Es gibt viele Wege um ein Schema zu klonen. In den Befehlbeispielen verwenden wir `expdb` und `impdb`, die unter der Haube Data Pump benutzen.

Bemerkung: Die Verbindungseinstellungen in dieser Dokumentation gehen davon aus, dass die Quell- und Ziel-Datenbank in einem Docker Container laufen. Mehr dazu unter https://github.com/bschmalhofer/otobo-ideas/blob/master/oracle.md.

---

1. Stoppen Sie otobo

Stoppen Sie den OTOBO Webserver, damit die Verbindung zur otobo Datenbank geschlossen wird.

```
-- in the OTOBO database
DROP USER otobo CASCADE
```

2. Export des kompletten OTRS Schemas.

```
mkdir /tmp/otrs_dump_dir
```

```
-- in the OTRS database
CREATE DIRECTORY OTRS_DUMP_DIR AS '/tmp/otrs_dump_dir';
GRANT READ, WRITE ON DIRECTORY OTRS_DUMP_DIR TO sys;
```

```
expdp \"sys/Oradoc_db1@//127.0.0.1/orclpdb1.localdomain as sysdba\" schemas=otrs␣
↪directory=OTRS_DUMP_DIR dumpfile=otrs.dmp logfile=expdpotrs.log
```

3. Import des OTRS Schemas und Umbenennung zu ‚otobo'.

```
impdp \"sys/Oradoc_db1@//127.0.0.1/orclpdb1.localdomain as sysdba\" directory=OTRS_
↪DUMP_DIR dumpfile=otrs.dmp logfile=impdpotobo.log remap_schema=otrs:otobo
```

```
-- in the OTOBO database
-- double check
select owner, table_name from all_tables where table_name like 'ARTICLE_DATA_OT%_CHAT
↪';

-- optionally, set the password for the user otobo
    ALTER USER otobo IDENTIFIED BY XXXXXX;
```

4. Anpassung des geklonten otobo Schemas

```
cd /opt/otobo
scripts/backup.pl --backup-type migratefromotrs # it's OK that the command knows only␣
↪about the otobo database, only last line is relevant
sqlplus otobo/otobo@//127.0.0.1/orclpdb1.localdomain < /home/bernhard/devel/OTOBO/
↪otobo/2021-03-31_13-36-55/orclpdb1.localdomain_post.sql >sqlplus.out 2>&1
double check with `select owner, table_name from all_tables where table_name like
↪'ARTICLE_DATA_OT%_CHAT';
```

5. Starten Sie den Webserver für otobo wieder
6. Fahren Sie mit Schritt 5 `migration.pl` fort.

Bemerkung: Wird auf eine OTOBO-Version migriert, die größer oder gleich 10.1 ist, muss das Skript `/opt/otobo/scripts/DBUpdate-to-10.1.pl` ausgeführt werden, um die Tabellen `stats_report` & `data_storage` zu erstellen, die neu in der Version 10.1 hinzugefügt wurden.

5.14.2 Optionaler Schritt: Gestreamlinte Migration der Datenbank (nur für Experten und Spezialszenarien)

Bei der allgemeinen Migrationsstrategie werden alle Daten der Datenbanktabellen Zeile für Zeile von der OTRS-Datenbank in die OTOBO-Datenbank kopiert. In manchen Fällen kann ein Export der OTRS-Datenbank und der anschließende Import in die OTOBO-Datenbank schneller und zuverlässiger sein.

Bemerkung: Diese Variante funktioniert sowohl bei Docker-basierten als auch bei nativen Installationen.

Bemerkung: Diese Anleitung setzt voraus, dass OTRS MySQL als Backend nutzt.

Zuerst muss ein Datenbank-Dump der OTRS-Tabellen erstellt werden. Anschließend führen wir ein paar Transformationen durch:

- Konvertierung des Zeichensatzes auf utf8mb4
- umbenennen einiger Tabellen
- Kürzung bestimmter Tabellenspalten

Nach der Umwandlung können wir die Tabellen im OTOBO-Schema mit den umgewandelten Daten aus OTRS überschreiben. Effektiv benötigen wir keine einzelne Dump-Datei, sondern mehrere SQL-Skripte.

Ist `mysqldump` installiert und kann eine Verbindung zur OTRS-Datenbank hergestellt werden, können Sie den Datenbankdump direkt auf dem Docker-Host erstellen. Hierzu können Sie das Skript bin/backup.pl verwenden.

Warnung: Dies setzt voraus, dass eine OTRS-Installation auf dem Docker-Host verfügbar ist.

```
otobo> cd /opt/otobo
otobo> scripts/backup.pl -t migratefromotrs --db-name otrs --db-host=127.0.0.1 --db-
↪user otrs --db-password "secret_otrs_password"
```

Bemerkung: Alternativ können Sie die Datenbank auch auf einen anderen Server dumpen und anschließend auf den Docker-Host verschieben. Dazu können Sie folgende Beispielbefehle verwenden.

Das Script bin/backup.pl generiert vier SQL-Dateien in einem Dump-Verzeichnis, z.B. in 2021-04-13_12-13-04. Um diese SQL-Dateien auszuführen, wird der Kommandozeilenbefehl `mysql` genutzt.

Native Installation:

```
otobo> cd <dump_dir>
otobo> mysql -u root -p<root_secret> otobo < otrs_pre.sql
otobo> mysql -u root -p<root_secret> otobo < otrs_schema_for_otobo.sql
otobo> mysql -u root -p<root_secret> otobo < otrs_data.sql
otobo> mysql -u root -p<root_secret> otobo < otrs_post.sql
```

Docker-basierte OTOBO-Installation:

Zum Importieren des Datenbankdumps nutzen wir `mysql` im Docker-Container otobo_db_1. Achtung: Das Passwort für den Datenbank-Root ist jetzt das in .env. definierte Passwort.

```
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo < /opt/
↪otobo/<dump_dir>/otrs_pre.sql
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo < /opt/
↪otobo/<dump_dir>/otrs_schema_for_otobo.sql
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo < /opt/
↪otobo/<dump_dir>/otrs_data.sql
```

```
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo < /opt/
↪otobo/<dump_dir>/otrs_post.sql
```

Nutzen Sie die folgenden Befehlen, um zu überprüfen, ob der Import erfolgreich war.

```
otobo> mysql -u root -p<root_secret> -e 'SHOW DATABASES'
otobo> mysql -u root -p<root_secret> otobo -e 'SHOW TABLES'
otobo> mysql -u root -p<root_secret> otobo -e 'SHOW CREATE TABLE ticket'
```

oder wenn es unter Docker läuft

```
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> -e 'SHOW␣
↪DATABASES'
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo -e 'SHOW␣
↪TABLES'
docker_admin> docker-compose exec -T db mysql -u root -p<root_secret> otobo -e 'SHOW␣
↪CREATE TABLE ticket'
```

Die Datenbank wurde nun migriert. Das bedeutet, dass Sie im nächsten Schritt die Datenbankmigration überspringen können. Beachten Sie hierzu die entsprechende Checkbox.

CHAPTER 6

Updates

Bemerkung: Es wird dringend empfohlen, zuerst ein Test-Update auf einem separaten Testsystem durchzuführen.

Bemerkung: Auf Debian-System kann es erforderlich sein, dass man einige perl-Pakete manuell installieren muss, bevor man auf 11.0 upgraden kann.

```
apt-get install -y libarchive-zip-perl libtimedate-perl libdatetime-perl␣
↪libconvert-binhex-perl libcgi-psgi-perl libdbi-perl libdbix-connector-perl␣
↪libfile-chmod-perl liblist-allutils-perl libmoo-perl libnamespace-autoclean-
↪perl libnet-dns-perl libnet-smtp-ssl-perl libpath-class-perl libsub-
↪exporter-perl libtemplate-perl libtemplate-perl libtext-trim-perl libtry-
↪tiny-perl libxml-libxml-perl libyaml-libyaml-perl libdbd-mysql-perl␣
↪libapache2-mod-perl2 libmail-imapclient-perl libauthen-sasl-perl libauthen-
↪ntlm-perl libjson-xs-perl libtext-csv-xs-perl libpath-class-perl libplack-
↪perl libplack-middleware-header-perl libplack-perl libplack-middleware-
↪reverseproxy-perl libencode-hanextra-perl libio-socket-ssl-perl libnet-
↪ldap-perl libcrypt-eksblowfish-perl libxml-libxslt-perl libxml-parser-perl␣
↪libconst-fast-perl
```

6.1 Schritt 1: Alle relevanten Dienste und den OTOBO Daemon stoppen

Stellen Sie sicher, dass keine Dienste oder Cronjobs mehr aktiv sind und auf OTOBO zugreifen möchten. Hierfür ist ausschlaggebend, wie Ihre Dienste konfiguriert sind.

```
root> systemctl stop postfix
root> systemctl stop apache2
root> systemctl stop cron
```

Stoppen Sie alle OTOBO Cron Jobs und den Daemon (in dieser Reihenfolge):

```
root> su - otobo
otobo> cd /opt/otobo/
otobo> bin/Cron.sh stop
otobo> bin/otobo.Daemon.pl stop
```

6.2 Schritt 2: Dateien und Datenbank sichern

Erstellen Sie ein Backup des gesamten `/opt/otobo`-Verzeichnis sowie der Datenbank.

6.2.1 Beispiel für eine Standardinstallation mit Ubuntu und MySQL

```
root> mkdir /root/otobo-update                          # Create a update directory
root> cd /root/otobo-update                             # Change into the update directory
root> cp -pr /opt/otobo otobo-prod-old                  # Backup the hole OTOBO directory␣
↪to the update directory
root> mysqldump -u otobo -p otobo -r otobo-prod-old.sql    # Backup the otobo database␣
↪to otobo-prod-old.sql
```

Stellen Sie sicher, dass alle Dateien valide sind. Sie haben nun ein Backup mit allen benötigten Daten.

Warnung: Don't proceed without a complete backup of your system. You can use also the Backup und Wiederherstellung script for this.

6.2.2 Schritt 2.1: Löschen Sie das CPAN-Verzeichnis, wenn Sie von 10.1 aktualisieren

Wenn Sie von 10.1 auf 11.0 aktualisieren, müssen Sie das cpan-lib-Verzeichnis bereinigen, da sich einige der cpan-Bibliotheken geändert haben.

```
root> rm -rf /opt/otobo/Kernel/cpan-lib/*
```

6.3 Schritt 3: Neues Release installieren

Laden Sie unter https://ftp.otobo.org/pub/otobo/ das neueste OTOBO-Release herunter und entpacken Sie das Quell-Archiv (zum Beispiel mit `tar`) in das Verzeichnis `/root/otobo-update`:

```
root> cd /root/otobo-update                                                  # Change into␣
↪the update directory
root> wget https://ftp.otobo.org/pub/otobo/otobo-latest-11.0.tar.gz      # Download he␣
↪latest OTOBO 11.0 release
root> tar -xzf otobo-latest-11.0.tar.gz                                      # Unzip OTOBO
root> cp -r otobo-11.0.x/* /opt/otobo                                        # Copy the␣
↪new otobo directory to /opt/otobo
```

6.3.1 Alte Konfigurationsdateien wiederherstellen

In OTOBO 10 reicht es aus, die Datei "Kernel/Config.pm"zu kopieren.

```
root> cd /root/otobo-update
root> cp -p otobo-prod-old/Kernel/Config.pm /opt/otobo/Kernel/
root> cp -p otobo-prod-old/var/cron/* /opt/otobo/var/cron/
```

6.3.2 Artikeldaten wiederherstellen

Wenn Sie OTOBO so konfiguriert haben, dass Artikeldaten im Dateisystem gespeichert werden, müssen Sie den Ordner `article` in `/opt/otobo/var/` (oder in dem in der Systemkonfiguration angegebenen Ordner) wiederherstellen.

```
root> cd /root/otobo-update
root> cp -pr otobo-prod-old/var/article/* /opt/otobo/var/article/
```

6.3.3 Bereits installierte Standardstatistiken wiederherstellen

Wenn Sie zusätzliche Pakete mit Standardstatistiken haben, müssen Sie die XML-Statistikdateien mit dem Suffix `*.installed` in `/opt/otobo/var/stats` wiederherstellen.

```
root> cd /root/otobo-update/otobo-prod-old/var/stats
root> cp *.installed /opt/otobo/var/stats
```

6.3.4 Dateiberechtigungen anpassen

Führen Sie folgenden Befehl aus, um die Datei- und Verzeichnis-Berechtigungen für OTOBO zu definieren. Es wird versucht, die passenden Benutzer- und Gruppeneinstellungen für Ihr Setup zu ermitteln.

```
root> /opt/otobo/bin/otobo.SetPermissions.pl
```

6.3.5 Apache-Konfigurationsdateien überprüfen

Neuere Versionen von OTOBO können dich dazu veranlassen, die Apache-Konfiguration anzupassen. Ab Version 10.1 sind wir von CGI zu PSGI gewechselt. Schau dir `scripts/apache2-httpd-vhost-443.include.conf` an, um zu sehen, welche Einstellungen angepasst/hinzugefügt werden müssen.

6.4 Step 4: Check for new needed Perl modules

OTOBO benötigt für einige Versionssprünge neue CPAN-Pakete. Bitte prüfen Sie, ob neue Pakete benötigt werden und installieren Sie diese gegebenenfalls.

Bemerkung: Auf Debian-Systemen kann es notwendig sein, einige Pakete manuell zu installieren:

```
apt-get install -y libarchive-zip-perl libtimedate-perl libdatetime-perl libconvert-
↪binhex-perl libcgi-psgi-perl libdbi-perl libdbix-connector-perl libfile-chmod-perl␣
↪liblist-allutils-perl libmoo-perl libnamespace-autoclean-perl libnet-dns-perl␣
↪libnet-smtp-ssl-perl libpath-class-perl libsub-exporter-perl libtemplate-perl␣
↪libtemplate-perl libtext-trim-perl libtry-tiny-perl libxml-libxml-perl libyaml-
↪libyaml-perl libdbd-mysql-perl libapache2-mod-perl2 libmail-imapclient-perl␣
↪libauthen-sasl-perl libauthen-ntlm-perl libjson-xs-perl libtext-csv-xs-perl libpath-
↪class-perl libplack-perl libplack-middleware-header-perl libplack-perl libplack-
↪middleware-reverseproxy-perl libencode-hanextra-perl libio-socket-ssl-perl libnet-
```

```
root> su - otobo
otobo> perl /opt/otobo/bin/otobo.CheckModules.pl --list
```

6.5 Schritt 5: Aktualisieren bereits installierter Pakete und Konfiguration neukonfigurieren

Sie können den folgenden Befehl verwenden, um alle installierten Pakete zu aktualisieren. Dies funktioniert für alle Pakete, die in Online-Repositorys verfügbar sind. Alle anderen Pakete können Sie später über den Paketmanager aktualisieren (dies erfordert einen laufenden OTOBO-Daemon).

```
root> su - otobo
otobo> /opt/otobo/bin/otobo.Console.pl Admin::Package::ReinstallAll
otobo> /opt/otobo/bin/otobo.Console.pl Admin::Package::UpgradeAll
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Config::Rebuild
```

6.6 Step 6: Only for minor or major release upgrades (for example to upgrade from 10.1 to 11.0)

```
root> su - otobo
otobo> /opt/otobo/scripts/DBUpdate-to-11.0.pl
```

6.7 Schritt 7: Dienste starten

Starten Sie OTOBO Cron Jobs und den Daemon (in dieser Reihenfolge):

```
root> su - otobo
otobo> cd /opt/otobo/
otobo> bin/otobo.Daemon.pl start
otobo> bin/Cron.sh start
```

Jetzt können die Dienste wieder gestartet werden. Wie genau vorzugehen ist, hängt von deren Konfiguration ab. Ein Beispiel:

```
root> systemctl start postfix
root> systemctl start apache2
root> systemctl start cron
```

Jetzt können Sie sich an Ihrem System anmelden.

CHAPTER 7

Aktualisieren einer Docker-basierte OTOBO-Installation

Warnung: Don't update without a complete backup of your system. You can use the Backup und Wiederherstellung mit Docker script in your existing Docker installation for that.

Für eine Docker-basierte OTOBO-Umgebung wird OTOBO selbst sowie eine Umgebung benötigt, in der die Software ausgeführt werden kann. Das OTOBO Docker Image stellt die Umgebung sowie eine Kopie der OTOBO-Software bereit. Die Software selbst wird im Volume otobo_opt_otobo installiert. Das Volume hat deshalb einen Namen, weil alle Laufzeitdaten, z. B. Konfigurationsdateien und installierte Pakete, im selben Verzeichnisbaum gespeichert werden.

Beim Update auf eine neue OTOBO-Version müssen verschiedene Dinge geschehen.

- Die Docker Compose Dateien müssen aktualisiert werden.
- Die Docker Compose Konfig-Datei .env muss überprüft werden.
- Das neue Docker-Image muss abgerufen werden.
- Das Volume otobo_opt_otobo muss aktualisiert werden.
- Einige Wartungsaufgaben sind auszuführen.

Bemerkung: In the sample commands below, the version **11.x.y**, corresponding to the tag **11_x_y**, is used as the example version. Please substitute it with the real version, e.g. **11.0.2**.

7.1 Docker Compose-Dateien aktualisieren

Die OTOBO Docker Compose-Dateien wechseln von Release zu Release. Achten Sie deshalb darauf, jeweils das richtige Setup zu verwenden.

Bemerkung: Alle verfügbaren Releases finden Sie unter `https://hub.docker.com/repository/docker/rotheross/otobo/tags`.

```
# Change to the otobo docker directory
docker_admin> cd /opt/otobo-docker

# Get the latest tags
docker-admin> git fetch --tags

# Update OTOBO docker-compose repository to version 11.x.y.
docker-admin> git checkout rel-11_x_y
```

7.2 Docker Compose .env-Datei überprüfen

Die .env-Datei kontrolliert den OTOBO Docker Container. In dieser Datei wird über die Variablen OTOBO_IMAGE_OTOBO, OTOBO_IMAGE_OTOBO_ELASTICSEARCH und OTOBO_IMAGE_OTOBO_NGINX festgelegt, welche Images verwendet werden. Ist kein spezifischer Wert für die Variablen angegeben, wird jeweils das neueste Image verwendet. Möchten Sie eine spezifische Version verwenden, fügen Sie diese hier ein.

7.3 Docker Images abrufen

Die gewünschten Images können mit Docker Compose von https://hub.docker.com/repository/docker/rotheross/ abgerufen werden.

```
# Change to the otobo docker directory
docker_admin> cd /opt/otobo-docker

# fetch the new images, either the default tag 'latest-11_0' or the specific version␣
↪tag declared in .env
docker_admin> docker-compose pull
```

7.4 OTOBO aktualisieren

Warnung: Bitte beachten Sie, dass kleinere oder größere Upgrades immer nacheinander durchgeführt werden müssen. Wenn Sie von Version 10.0.* auf die neueste Version 11.0.* upgraden möchten, aktualisieren Sie bitte zuerst auf 10.1 und dann auf 11.0.

In diesem Schritt wird das Volume otobo_opt_otobo aktualisiert und folgende OTOBO Kommandozeilenbefehle ausgeführt:

- Admin::Package::ReinstallAll
- Admin::Package::UpgradeAll
- Maint::Config::Rebuild
- Maint::Cache::Delete

Bei kleineren und größeren Versions-Upgrades müssen zuvor auch Aktualisierungsaufgaben für das Kernsystem durchgeführt werden

```
# stop and remove the containers, but keep the named volumes
docker_admin> docker-compose down

# copy the OTOBO software, while containers are still stopped
docker_admin> docker-compose run --no-deps --rm web copy_otobo_next

# start containers again, using the new version and the updated /opt/otobo
docker_admin> docker-compose up --detach

# a quick sanity check
docker_admin> docker-compose ps

# ** Only for minor or major release upgrades! **
# run upgrade tasks for the OTOBO core (for example when upgrading from 10.1 to 11.0)
docker_admin> docker-compose exec web perl scripts/DBUpdate-to-11.0.pl

# complete the update, with running database
docker_admin> docker-compose exec web /opt/otobo_install/entrypoint.sh do_update_tasks

# inspect the update log
docker_admin> docker-compose exec web cat /opt/otobo/var/log/update.log
```

Bemerkung: Für einfache Patchlevel-Updates (z.B. von 11.0.2 auf 11.0.3) kann die Ausführung der oben genannten Befehle mit Hilfe des Skripts scripts/update.sh automatisiert werden. Dieses Skript führt die Befehle beginnend mit dem Befehl **docker-compose pull** aus. Beachten Sie, dass der Aufruf der Datenbank-Upgrade-Skripte nicht enthalten ist und daher nicht für Versions-Upgrades verwendet werden kann.

```
docker_admin> ./scripts/update.sh --help
docker_admin> ./scripts/update.sh
```

Bemerkung: Beim Upgrade von 10.1.x auf 11.0.x mit installiertem ITSM-Plugin muss zusätzlich der folgende Befehl ausgeführt werden:

```
docker exec -it otobo_web_1 perl bin/otobo.Console.pl␣
↪Admin::ITSM::Configitem::UpgradeTo11
```

CHAPTER 8

Backup und Wiederherstellung

OTOBO bietet fertige Skripte zum Erstellen und Wiedereinspielen von Backups. Für weitere Informationen führen Sie diese mit der Option `-h` aus.

8.1 Backup

Bemerkung: Der Benutzer `otobo` benötigt im Zielverzeichnis Schreibrechte, damit ein neues Backup erstellt werden kann.

```
otobo> /opt/otobo/scripts/backup.pl -h
```

Die Ausgabe des Skriptes:

```
Backup an OTOBO system.

Usage:
 backup.pl -d /data_backup_dir [-c gzip|bzip2] [-r DAYS] [-t␣
↪fullbackup|nofullbackup|dbonly]
 backup.pl --backup-dir /data_backup_dir [--compress gzip|bzip2] [--remove-old-
↪backups DAYS] [--backup-type fullbackup|nofullbackup|dbonly]

Short options:
 [-h]                    - Display help for this command.
 -d                      - Directory where the backup files should place to.
 [-c]                    - Select the compression method (gzip|bzip2). Default: gzip.
 [-r DAYS]               - Remove backups which are more than DAYS days old.
 [-t]                    - Specify which data will be saved␣
↪(fullbackup|nofullbackup|dbonly). Default: fullbackup.


Long options:
```

```
 [--help]                      - same as -h
 --backup-dir                  - same as -d
 [--compress]                  - same as -c
 [--remove-old-backups DAYS]   - same as -r
 [--backup-type]               - same as -t

Help:
Using -t fullbackup saves the database and the whole OTOBO home directory (except /
↪var/tmp and cache directories).
Using -t nofullbackup saves only the database, /Kernel/Config* and /var directories.
With -t dbonly only the database will be saved.

Override the max allowed packet size:
When backing up a MySQL one might run into very large database fields. In this case␣
↪the backup fails.
For making the backup succeed one can explicitly add the parameter --max-allowed-
↪packet=<SIZE IN BYTES>.
This setting will be passed on to the command mysqldump.

Output:
 Config.tar.gz          - Backup of /Kernel/Config* configuration files.
 Application.tar.gz      - Backup of application file system (in case of full backup).
 VarDir.tar.gz           - Backup of /var directory (in case of no full backup).
 DataDir.tar.gz          - Backup of article files.
 DatabaseBackup.sql.gz   - Database dump.
```

8.2 Wiederherstellen

Bemerkung: Bevor Sie die Datenbank wiederherstellen, stellen Sie sicher, dass die Datenbank “otobo“vorhanden ist und keine Tabellen enthält.

```
otobo> /opt/otobo/scripts/restore.pl -h
```

Die Ausgabe des Skriptes:

```
Restore an OTOBO system from backup.

Usage:
 restore.pl -b /data_backup/<TIME>/ -d /opt/otobo/

Options:
 -b                     - Directory of the backup files.
 -d                     - Target OTOBO home directory.
 [-h]                   - Display help for this command.
```

8.3 Sonderfalll: OTOBO und Docker

The same scripts can be used with OTOBO running under Docker. However some Docker specific limitation must be considered. Please read to the chapter Backup und Wiederherstellung mit Docker for information about that case.

CHAPTER 9

Backup und Wiederherstellung mit Docker

Bitte lesen Sie die Anleitung Backup und Wiederherstellung Backup und Wiederherstellung. Hier finden Sie die grundlegenden Informationen zu den Backup-Skripten.

9.1 Sonderfalll: OTOBO und Docker

Auch für Docker-basierte OTOBO-Umgebungen können die OTOBO-Standardskripte `backup.pl` und `restore.pl` verwendet werden. Beachten Sie jedoch einige Docker-spezifische Besonderheiten.

Erstens: Legen Sie die Backupdateien nicht im internen Dateisystem des Containers ab. Wird der Container gestoppt, sind die Daten dort verloren. Stattdessen empfehlen wir das Backup-Verzeichnis in einem Volume anzulegen. Wir betrachten zunächst nur den einfachsten Fall: Als Backup-Verzeichnis wird ein lokales Verzeichnis auf dem Docker-Host verwendet. Innerhalb des Containers kann das Backup-Verzeichnis an beliebiger Stelle angelegt werden. Für dieses Beispiel verwenden wir das lokale Verzeichnis `otobo_backup` als Speicherort auf dem Host und `/otobo_backup` im Container.

Zweitens: Innerhalb des Docker-Containers werden Befehle in der Regel vom User otobo mit der User-ID 1000 und der Gruppen-ID 1000 ausgeführt. Achten Sie darauf, dass dieser User im Backup-Verzeichnis Schreibrechte hat.

Als erstes legen wir das Volume an.

```
# create the backup directory on the host
docker_admin>mkdir otobo_backup

# give the backup dir to the user otobo, elevated privs might be needed for that
docker_admin>chown 1000:1000 otobo_backup

# create the Docker volume
docker_admin>docker volume create --name otobo_backup --opt type=none --opt device=
↪$PWD/otobo_backup --opt o=bind

# inspect the volume out of curiosity
docker_admin>docker volume inspect otobo_backup
```

Um das Backup zu erstellen, benötigen wir eine laufende Datenbank und die Volumes `otobo_opt_otobo` sowie `otobo_backup`. Das bedeutet, Webserver und OTOBO Daemon können, müssen aber nicht, gestoppt werden.

```
  # create a backup
  docker_admin>docker run -it --rm --volume otobo_opt_otobo:/opt/otobo --volume otobo_
↪backup:/otobo_backup --network otobo_default rotheross/otobo:latest-11_0 scripts/
↪backup.pl --extra-dump-options="--single-transaction" -d /otobo_backup

  # check the backup file
  docker_admin>tree otobo_backup

.. note::

  --extra-dump-options="--single-transaction" prevents the database tables from being␣
↪locked, so OTOBO can still be used during the backup.
```

Bemerkung: Bevor Sie die Datenbank wiederherstellen, stellen Sie sicher, dass die Datenbank "otobo"vorhanden ist und keine Tabellen enthält.

Um eine bestehende otobo Datenbank zu löschen und eine neue zu erstellen können folgende Befehle verwendet werden. Zuerst müssen Sie sich hierzu mit der MySQL CLI des db Containers verbinden.

Sobald man mit dem MySQL Server verbunden ist, kann die otobo Datenbank gelöscht und neu erstellt werden.

```
mysql@4f7783595190:/$>DROP DATABASE otobo;
mysql@4f7783595190:/$>CREATE DATABASE otobo CHARACTER SET utf8mb4 COLLATE utf8mb4_
↪unicode_ci;
mysql@4f7783595190:/$>GRANT ALL PRIVILEGES ON otobo.* TO 'otobo'@'%';
```

Um das System aus einem Backup wiederherstellen zu können, müssen wir angeben, welches Backup verwendet werden soll. Der Platzhalter `<TIMESTAMP>` steht für einen Wert wie `2020-09-07_09-38`.

```
# restore a backup
docker_admin>docker run -it --rm --volume otobo_opt_otobo:/opt/otobo --volume otobo_
↪backup:/otobo_backup --network otobo_default rotheross/otobo:latest-11_0 scripts/
↪restore.pl -d /opt/otobo -b /otobo_backup/<TIMESTAMP>
```

CHAPTER 10

Kerberos Single Sign On in der OTOBO Docker-Installation

Bitte lesen Sie das Kapitel Installation mit Docker und Docker Compose für grundlegende Informationen über die Installation und Konfiguration von OTOBO. Dieses Tutorial geht davon aus, dass OTOBO mit Docker installiert und konfiguriert wurde.

Bemerkung: Im Folgenden sprechen wir von AD (Active Directory). Natürlich funktioniert die Konfiguration von Kerberos auch mit LDAP.

10.1 Active Directory User erstellen

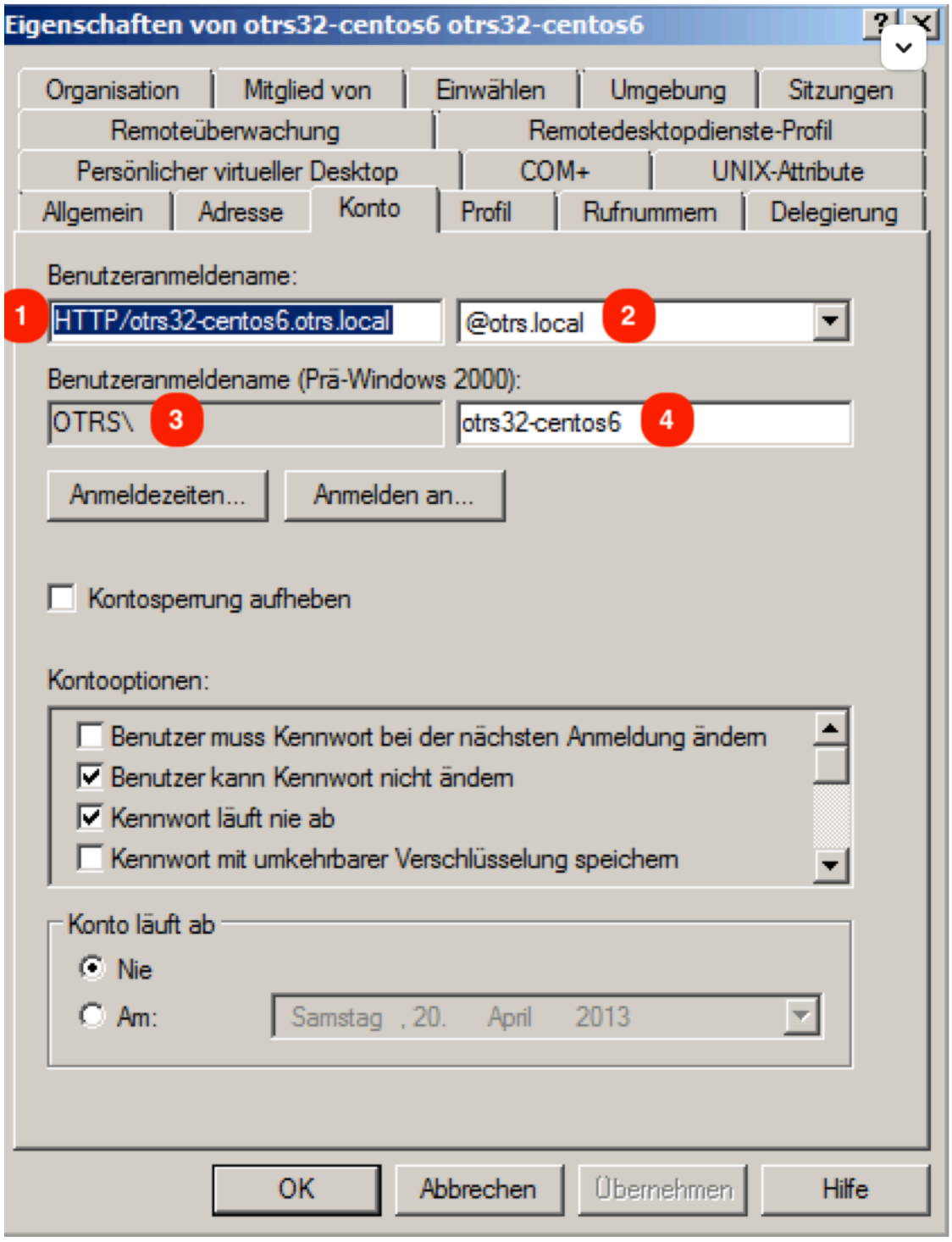
Bitte erstellen Sie einen neuen Active Directory Benutzer mit den folgenden Einstellungen und speichern Sie die markierten Einstellungen:

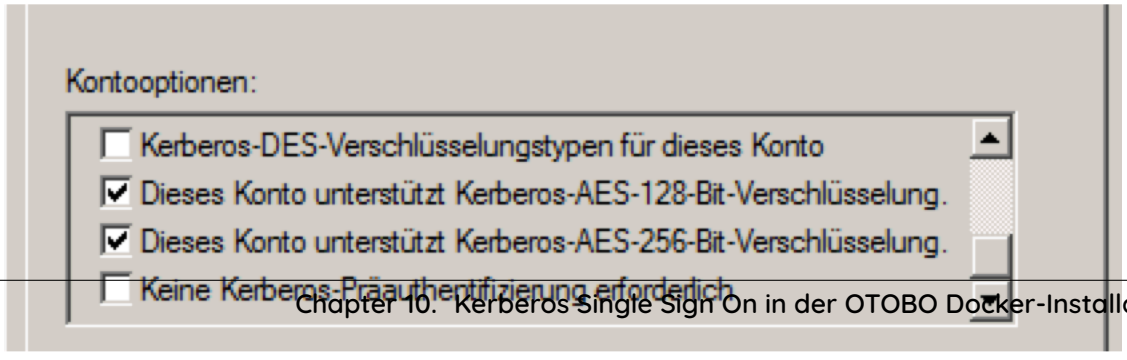
Bemerkung: Bitte verwende als Benutzernamen nur diese Syntax: HTTP/fqdn.from.your.otobo.de. fqdn.from.your.otobo.de muss ein A-Record DNS-Eintrag sein, kein CNAME! Im nächsten Schritt ist es auch möglich, andere URLs für OTOBO zu verwenden, sie müssen dann als CNAME auf unseren oben definierten A-Record zeigen.

Der Benutzername sollte in Großbuchstaben geschrieben sein, da Kerberos ihn auf diese Weise erwartet.

Das Passwort funktioniert nicht zuverlässig mit einigen Sonderzeichen (z.B. ‚&').

Sie müssen einen separaten AD-Benutzer anlegen. Sie können nicht jenen benutzen, den Sie bereits für Ihre LDAP/AD-Synchronisation verwenden.


Eigenschaften von otrs32-centos6 otrs32-centos6
Organisation
Mitglied von
Einwählen
Umgebung
Sitzungen
Remoteüberwachung
Remotedesktopdienste-Profil
Persönlicher virtueller Desktop
COM+
UNIX-Attribute
Allgemein
Adresse
Konto
Profil
Rufnummern
Delegierung
Benutzeranmeldename:
1
HTTP/otrs32-centos6.otrs.local
@otrs.local
2
Benutzeranmeldename (Prä-Windows 2000):
OTRS\
3
otrs32-centos6
4
Anmeldezeiten...
Anmelden an...
Kontosperrung aufheben
Kontooptionen:
Benutzer muss Kennwort bei der nächsten Anmeldung ändern
Benutzer kann Kennwort nicht ändern
Kennwort läuft nie ab
Kennwort mit umkehrbarer Verschlüsselung speichern
Konto läuft ab
Nie
Am:
Samstag , 20. April 2013
OK
Abbrechen
Übernehmen
Hilfe


Kontooptionen:
Kerberos-DES-Verschlüsselungstypen für dieses Konto
Dieses Konto unterstützt Kerberos-AES-128-Bit-Verschlüsselung.
Dieses Konto unterstützt Kerberos-AES-256-Bit-Verschlüsselung.
Keine Kerberos-Präauthentifizierung erforderlich

10.2 Active Directory Keytab-Datei erstellen

Im nächsten Schritt verbinden wir uns mit einem Domain-Controller des Active Directory und öffnen dort eine Konsole (cmd) mit Administratorrechten. Jetzt verwenden wir das Tool ktpass.exe, um die benötigte Keytab-Datei zu generieren:

```
ktpass.exe -princ HTTP/otrs32-centos6.otrs.local@OTRS.LOCAL -mapuser OTRS\otrs32-
↪centos6 -crypto All -pass Password -ptype KRB5_NT_PRINCIPAL -out c:\krb5.keytab
```

- -princ = HTTP/otrs32-centos6.otrs.local@OTRS.LOCAL -> Picture Number 1+@+Picture Number 2
- -mapuser = OTRSotrs32-centos6 (Benutzername prä Win 2000) -> -> Bildnummer 3++Bildnummer
- -pass = Passwort vom Benutzer otrs32-centos6 (Active Directory-Benutzer)
- -out = c:/krb5.keytab

Bemerkung: Bitte schreiben Sie die Benutzernamen (@OTRS.LOCAL) immer in Großbuchstaben. Das Passwort darf einige Sonderzeichen nicht enthalten.

Im nächsten Schritt verschieben Sie bitte die Datei krb5.keytab auf den OTOBO-Server:

```
# Create new directory
docker_admin> mkdir /opt/otobo-docker/nginx-conf

# Move the file krb5.keytab to the new directory (Attention, depending on where you␣
↪have placed the krb5.conf file, the command below will change.)
docker_admin> mv ?/krb5.keytab /opt/otobo-docker/nginx-conf/krb5.keytab
```

10.3 Erstelle ein neues Volume für deine individuelle nginx-Konfiguration

```
docker volume create otobo_nginx_custom_config
otobo_nginx_custom_config_mp=$(docker volume inspect --format '{{ .Mountpoint }}'␣
↪otobo_nginx_custom_config)
docker create --name tmp-nginx-container rotheross/otobo-nginx-webproxy:latest-11_0␣
↪(watch out: version number)
docker cp tmp-nginx-container:/etc/nginx/templates /tmp
docker cp tmp-nginx-container:/etc/nginx/templates/otobo_nginx-kerberos.conf.template.
↪hidden $otobo_nginx_custom_config_mp/otobo_nginx.conf.template
docker rm tmp-nginx-container
vim docker-compose/otobo-nginx-custom-config.yml
```

```
COMPOSE_FILE =>
docker-compose/otobo-nginx-custom-config.yml
NGINX_ENVSUBST_TEMPLATE_DIR=/etc/nginx/config/template-custom
```

10.4 Neue OTOBO .env*-Datei anlegen

Zunächst müssen wir die alten Datei `/opt/otobo-docker/.env` in `.env.tmp` umbenennen und erstellen dann eine neue Datei `.env` , die die Kerberos-Einstellungen beinhaltet.

```
# Stop OTOBO Container if running
docker_admin>cd /opt/otobo-docker
docker_admin>docker-compose down

# create a backup of the old .env file
docker_admin>mv /opt/otobo-docker/.env /opt/otobo-docker/.env.tmp

# create a new backupfile including kerberos settings
docker_admin>cp /opt/otobo-docker/.docker_compose_env_https_kerberos /opt/otobo-
↪docker/.env
```

Kopiere jetzt deine bestehenden Konfigurationsoptionen in die neue .env-Datei (mindestens OTOBO_DB_ROOT_PASSWORD, OTOBO_NGINX_SSL_CERTIFICATE, OTOBO_NGINX_SSL_CERTIFICATE_KEY) und füge die folgenden Kerberos-Einstellungen hinzu:

Kerberos keytab OTOBO_NGINX_KERBEROS_KEYTAB=/opt/otobo-docker/nginx-conf/krb5.keytab

Kerberos-Konfig (Wichtig, bitte kommentiere diese Option wie hier aus!) # In der Standardkonfiguration wird die krb5.conf-Datei automatisch generiert # OTOBO_NGINX_KERBEROS_CONFIG=/opt/otobo-docker/nginx-conf/krb5.conf

Kerberos Service Name OTOBO_NGINX_KERBEROS_SERVICE_NAME=HTTP/otrs32-centos6.otrs.local # -> Bild Nummer 1

Kerberos REALM OTOBO_NGINX_KERBEROS_REALM=ROTHER-OSS.COM -> OTRS.LOCAL # -> Bild Nummer 2

Active Directory Domain Controller / Kerberos kdc OTOBO_NGINX_KERBEROS_KDC=

Active Directory Domain Controller / Kerberos Admin Server OTOBO_NGINX_KERBEROS_ADMIN_SERVER=rother-oss.com

Kerberos Default Domain OTOBO_NGINX_KERBEROS_DEFAULT_DOMAIN=otrs.local

10.5 OTOBO starten

Nach der initialen Konfiguration von Kerberos starten wir OTOBO erneut:

```
# Start OTOBO using docker-compose
docker_admin> docker-compose up -d
```

10.6 In OTOBO angeben, dass die Kerberos-Authentifizierung verwendet werden soll

Falls du AD-Authentifizierung konfiguriert hast, deaktiviere sie (z.B. durch Auskommentieren der entsprechenden Zeilen in deiner Kernel/Config.pm). Die Authentifizierung wird nicht mehr über LDAP stattfinden.

Um die Kerberos-Authentifizierung zu nutzen, nimm die Kerberos-Zeilen aus Kernel/Config/Defaults.pm und füge sie in deine Kernel/Config.pm ein. Zum Beispiel könnten diese Zeilen funktionieren:

```
$Self->{AuthModule} = 'Kernel::System::Auth::HTTPBasicAuth';
```

```
# In case you need to replace some part of the REMOTE_USER, you can
# use the following RegExp ($1 will be new login).
$Self->{'AuthModule::HTTPBasicAuth::ReplaceRegExp'} = '^(.+?)@.+?$';
```

10.7 Den Browser so konfigurieren, dass er Kerberos SSO versteht

Damit SSO funktioniert, muss der Browser entsprechend konfiguriert werden.

Chrome, Edge, Internet Explorer, etc.

Fügen Sie eine Seite unter „Lokale oder vertrauenswürdige Seiten“ hinzu und aktivieren Sie „Integrierte Windows-Authentifizierung“ (Internetoptionen).

Firefox

„about:config“ in die Adresszeile von Firefox eingeben

und folgende Einstellungen anpassen:

- network.negotiate-auth.trusted-uris = https:// (or https://otobofqdn)
- network.negotiate-auth.delegation-uris = http:// (or https://otobofqdn)

10.8 Fehlererkennung und Problembehebung

Sollte das Kerberos SSO nicht funktionieren, überprüfen Sie bitte zunächst, ob der NGINX-Container gestartet ist:

```
# Check Container
docker_admin> docker ps
```

Im nächsten Schritt überprüfen Sie bitte die NGINX-Logs auf mehr Informationen:

```
# Check NGINX logs
docker_admin> docker logs otobo_nginx_1 -f
```

Sollte NGINX laufen, loggen Sie sich bitte in den NGINX-Container ein und überprüfen Sie alle benötigten Dateien:

```
# Login to the NGINX Container
docker_admin> docker exec -it otobo_nginx_1 bash

# Now please check if the krb5.conf file exists with your needed values
nginx_root> cat /etc/krb5.conf

# Now please check if the krb5.keytab file exists
nginx_root> cat /etc/krb5.keytab

# If not, please quit from the container and copy the file again using docker
docker_admin> docker cp /opt/otobo-docker/nginx-conf/krb5.keytab otobo_nginx_1:/etc/
↪krb5.keytab
```

10.8.1 Kerberos Fehlerbehebung

```
 # Login to the NGINX Container
docker_admin> docker exec -it otobo_nginx_1 bash
```

Nun sind Sie in der Lage, die Kerberos-Einstellungen auf Fehler zu überprüfen. Beispiele:

```
env KRB5_TRACE=/dev/stdout kvno HTTP/otrs32-centos6.otrs.local@OTRS.LOCAL
klist -e
```

```
kinit -VV -k -t /etc/krb5.keytab HTTP/otrs32-centos6.otrs.local@OTRS.LOCAL
```

Falls Sie auf das Problem stoßen, dass die Authentifizierung scheinbar funktioniert, der Agent aber noch nicht in der Datenbank ist, funktioniert Ihre Synchronisierung (falls implementiert) möglicherweise nicht. Ein Fehler 52e (Erste Bindung fehlgeschlagen) weist darauf hin, dass mit Ihrem Suchbenutzer etwas nicht stimmt. Dies passiert, wenn Sie denselben Benutzer für die AD-Synchronisierung und als SSO-Benutzer verwenden. Bitte verwenden Sie hierfür separate AD-Benutzer. Um nicht eine neue Schlüsseltabelle erstellen und die oben genannten Schritte wiederholen zu müssen, könnte es einfacher sein, einen neuen Benutzer zur Verwendung in Ihrer AD-Synchronisierung zu erstellen (wahrscheinlich in Ihrer Kernel/Config.pm).

Falls SSO nicht richtig funktioniert, stelle sicher: * dass der Benutzer, für den es nicht funktioniert, im Active Directory ist * dass das System in der Domäne ist * dass es ordnungsgemäß als vertrauenswürdige Seite angegeben ist (siehe ‚Browser konfigurieren, um Kerberos SSO zu verstehen')

CHAPTER 11

Kundenoberfläche an Corporate Identity anpassen

In OTOBO ist es sehr einfach, den Kundenbereich an Ihre eigene Corporate Identity anzupassen. Folgen Sie diesem Tutorial Schritt für Schritt und OTOBO wird in Kürze in Ihrem eigenen Design erstrahlen.

Bemerkung: Momentan ist es nicht besonders einfach, den Agentenbereich an seine eigene CI anzupassen. Änderungen der OTOBO .css-Dateien wären hierfür notwendig. Eine Ausnahme ist das Logo der Agenten-Anmeldemaske und im Hauptmenübereich der Agenten. Die Logos können einfach ausgetauscht werden, indem man die Logos auf den Server kopiert und dann die Optionen `AgentLoginLogo` und `AgentLogo` unter `Admin -> Systemkonfiguration` anpasst.

11.1 Farben im Kundenbereich anpassen

Um die Farben für den OTOBO Kundenbereich anzupassen, gehen Sie bitte zu `Admin -> Systemkonfiguration` und passen Sie die folgenden Einstellungen an:

- CustomerColorDefinitions
- Um die Farben im Kunden-Dashboard anzupassen, gehen Sie bitte in `Admin -> Systemkonfiguration` und suchen Sie nach `CustomerDashboard`. In den Suchergebnissen finden Sie alle Optionen, die Sie für die Farbdefinitionen benötigen.

11.2 Logos und Bilder bearbeiten

Im ersten Schritt kopieren Sie bitte Ihre Logos und Bilder auf den OTOBO-Server. Bitte verwenden Sie einen SCP-Dienst (bspw. WinSCP) für diesen Zweck. Häufig hat man nicht die Berechtigungen um die Logos an die richtigen Stellen zu kopieren. In diesem Fall ist es am besten, den Ordner /tmp/ zu verwenden.

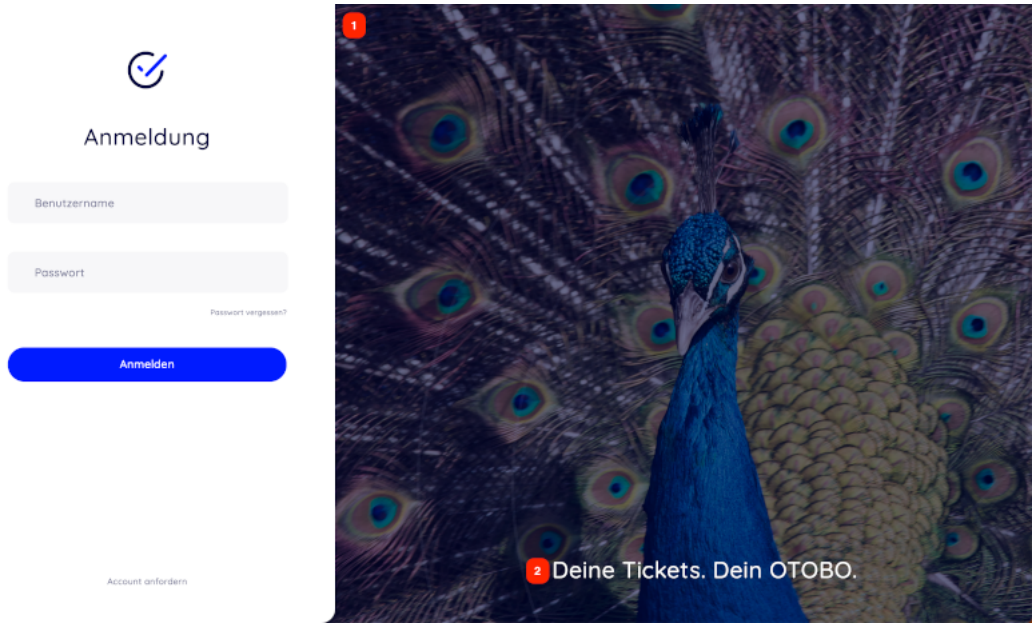
Im nächsten Schritt kopieren Sie das Logo in das OTOBO Home-Verzeichnis:

```
**# Using OTOBO Docker Installation**
otobo_admin> docker cp /tmp/Logos.png otobo_web_1:/opt/otobo/var/httpd/htdocs/skins/
↪Customer/default/img/

**# Nativ installation in /opt/otobo/**
otobo_admin> cp /tmp/Logos.png /opt/otobo/var/httpd/htdocs/skins/Customer/default/img/
```

Nun wechseln Sie im OTOBO Agentenbereich zu `Admin -> Systemkonfiguration` und passen Sie folgende Einstellungen an:

11.2.1 Bilder und Text für den Kundenlogin anpassen



- 1 und 2 - Systemkonfigurationseinstellung **CustomerLogin::Settings**

Deckkraft und Wasserzeichen entfernen

Im Moment ist es nicht möglich, in der Systemkonfiguration das Overlay und das Wasserzeichen zu entfernen, die im Bild rechts verwendet werden.

Um die Deckkraft zu entfernen, passen Sie die Option **#oooLoginBG > .oooBG** in der Datei an

`var/httpd/htdocs/skins/Customer/default/css/Core.Login.css`

```
 #oooLoginBG > .oooBG {
    position: relative;
    width: 100%;
    height: 100%;
 /*   opacity: 0.45; Disable opacity */
    background-size: cover;
    overflow: hidden;
}
```

Um das Wasserzeichen zu entfernen, löschen Sie bitte die folgenden Zeilen innerhalb der Datei:

`Kernel/Output/HTML/Templates/Standard/CustomerLogin.tt`

```
<!-- start login -->
<div id="oooLoginBG">
    <div class="oooBG" style="background-image: url([% Data.Background | html %]);">
# remove this line ->        <div id="oooBGSignet" style="background-image: url([%
↪Config("Frontend::WebPath") %]common/img/otobo-signet_border.svg);"></div>
    </div>
    <h1>[% Translate(Data.LoginText) | html %]</h1>
</div>
```

Bemerkung: Please add the files to a opm package in the next step, so that the changes remain persistent. You can find instructions on how to do this in our Admin Manual: https://doc.otobo.org/manual/developer/11.0/en/content/how-to-publish-otobo-extensions.html

11.2.2 Anpassen der Kunden-Dashboard-Kacheln und Optionen

Um die Farben des Kunden-Dashboard anzupassen, wechseln Sie bitte in `Admin -> Systemkonfiguration` und suchen Sie nach **CustomerDashboard**.

Im Suchergebnis finden Sie alle Optionen, die für die Farbdefinitionen benötigt werden.

- 1 - Um das Bild, die Verknüpfung und den Text anzupassen, verwenden Sie bitte die Systemkonfigurationseinstellung **CustomerDashboard::Tiles###FeaturedLink-01**
- 2 - Um den oben angezeigten Text anzupassen, verwenden Sie bitte die Systemkonfigurationseinstellung **CustomerDashboard::Configuration::Text**

Bemerkung: Bitte deaktivieren Sie die Konfigurationsoptionen für nicht benötigte Kacheln.


2
Willkommen Felix, in diesem OTOBO Demo.
Viel Spaß beim Erkunden der neuen OTOBO Kundenoberfläche!
1
Neues Release OTOBO 10.1
Highlights: Prozesse im Serviceportal, S3, moderne Authentifizierung mit OpenID Connect, erweiterte S/MIME-Funktionalität
Read more...
Service-Katalog
Hier siehst du, welche Servic
Facility Management
Spielwiese
Gewinnen Sie einen ersten E
OTOBO und seinen Möglichk
muss, alles kann. Jedes Syst
Ihre neuesten Tickets
test
Nr. #2022092753000034

CHAPTER 12

Perlmodule installieren

In Systemen mit besonderen Anforderungen kann es nötig sein, weitere Perlmodule zu installieren. Glücklicherweise verfügt Perl über ein exzellentes Paket-Repository, das nahezu alle Anforderungen befriedigt. Sie finden es unter der Bezeichnung CPAN unter https://metacpan.org/.

Es wird empfohlen, CPAN-Module mit dem Befehlszeilenclient `cpanm` zu installieren. `cpanm` ist auf vielen Systemen bereits installiert. Sollte dies bei Ihnen nicht der Fall sein, finden Sie weitere Informationen unter https://metacpan.org/pod/App::cpanminus.

Alternativ sind viele Perl-Module auch als Pakete für Ihr Betriebssystem verfügbar. Diese Pakete können über den regulären Paketmanager Ihres Systems installiert werden.

Standardmäßig wird das Modul durch `cpanm` in einem systemweiten Verzeichnis installiert. In diesem Fall müssen die Module als Root-Benutzer installiert werden. Der Befehl

```
root> cpanm Acme::Dice
```

ergibt z. B.:

```
otobo> perldoc -l Acme::Dice
/usr/local/share/perl/5.30.0/Acme/Dice.pm
```

12.1 Docker-basierte OTOBO Installationen

Besondere Vorsicht ist geboten, wenn OTOBO in Docker ausgeführt wird. In diesem Fall scheint es die einfachste Lösung, das gewünschte Modul einfach in das System-Perl zu installieren. Aufgrund der besonderen Arbeitsweise von Docker geht diese Anpassung aber verloren, sobald der Container neu gestartet wird. Die Module müssen also an einen Ort ausgelagert werden, der auch nach einem Neustart noch verfügbar ist. Das lokale Verzeichnis für die Installation kann mit der Option `--local-lib` angegeben werden. Die installierten Module werden von Perl auf Basis der Umgebungsvariablen `PERL5LIB` und `PATH` gefunden, die im Docker Image entsprechend gesetzt werden.

Die installierten Perl-Module sind auch nach einem OTOBO-Upgrade weiterhin verfügbar. Die Grundsatzregel lautet, dass unter `/opt/otobo` hinzugefügte Dateien bei einem Upgrade nicht gelöscht werden.

Um Perl Module in einem bestimmten Verzeichnis zu installieren, muss der Installationsbefehl angepasst werden. Konkret wird in diesem Fall die Option `--local-lib` angefügt. Im Folgenden ein Beispiel aus dem Container **web**.

```
# starting a bash session in the container web
docker_admin> cd /opt/otobo-docker/
docker_admin> docker-compose exec web bash
otobo@6ef90ed00cd0:~$ pwd
/opt/otobo

# installing the sample module Acme::Dice
otobo@6ef90ed00cd0:~$ cpanm --local-lib local Acme::Dice
--> Working on Acme::Dice
Fetching http://www.cpan.org/authors/id/B/BO/BOFTX/Acme-Dice-1.01.tar.gz ... OK
Configuring Acme-Dice-1.01 ... OK
Building and testing Acme-Dice-1.01 ... OK
Successfully installed Acme-Dice-1.01
1 distribution installed

# confirm the installation directory
otobo@6ef90ed00cd0:~$ perldoc -l Acme::Dice
/opt/otobo/local/lib/perl5/Acme/Dice.pm

# locally installed module is found because the environment is preset accordingly
otobo@6ef90ed00cd0:~$ echo $PERL5LIB
/opt/otobo_install/local/lib/perl5:/opt/otobo/local/lib/perl5
otobo@6ef90ed00cd0:~$ echo $PATH
/opt/otobo_install/local/bin:/opt/otobo/local/bin:/usr/local/sbin:/usr/local/bin:/usr/
↪sbin:/usr/bin:/sbin:/bin
```

CHAPTER 13

Performance Tuning

Hier finden Sie eine Reihe von Möglichkeiten, die Leistung Ihrer OTOBO-Installation zu verbessern – z. B. durch Konfiguration, Entwicklung, Speichernutzung, etc.

13.1 Ticket-Indexmodul

Das Ticket-Indexmodul kann in der Systemkonfiguration über die Einstellung `Ticket::IndexModule` konfiguriert werden. Zwei Backend-Module bilden den Index für die Ticket-Ansicht nach Queues:

`Kernel::System::Ticket::IndexAccelerator::RuntimeDB` Die Standardoption, in der jede Queue-Ansicht zur Laufzeit aus der Tickettabelle generiert wird. Keine Performance-Beeinträchtigungen zu erwarten bis etwa 60.000 offene Tickets im System sind.

`Kernel::System::Ticket::IndexAccelerator::StaticDB` Das leistungsstärkste Modul. Empfohlen bei über 80.000 offenen Tickets im System. Das Modul verwendet eine spezielle `ticket_index`-Tabelle, die auf Basis der Ticketdaten mit Keywords bestückt wird. Verwenden Sie folgenden Befehl, um nach der Umstellung auf ein anderes Backend einen ersten Index zu generieren:

```
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Ticket::QueueIndexRebuild
```

13.2 Ticket-Suchindex

OTOBO nutzt einen speziellen Suchindex, um felderübergreifende Volltextsuchen in Artikeln aus unterschiedlichen Kommunikationskanälen durchzuführen.

Verwenden Sie folgenden Befehl, um einen initialen Index zu erstellen:

```
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Ticket::FulltextIndex --rebuild
```

Bemerkung: Die eigentliche Artikelindexierung erfolgt über einen OTOBO Daemon-Job im Hintergrund. Auch wenn neu hinzugefügte Artikel sofort zur Indexierung markiert werden, kann es deshalb sein, dass ihr Index erst nach einigen Minuten zur Verfügung steht.

Es gibt einige Möglichkeiten, den Suchindex granularer abzustimmen:

`Ticket::SearchIndex::IndexArchivedTickets` Definiert, ob archivierte Tickets in den Suchindex aufgenommen werden (standardmäßig deaktiviert). Empfehlenswert, um den Suchindex in großen Systemen mit vielen archivierten Tickets klein zu halten. Ist die Option aktiviert, werden auch archivierte Tickets über die Volltextsuche gefunden.

`Ticket::SearchIndex::Attribute` Grundlegende Einstellungen für die Volltextsuche.

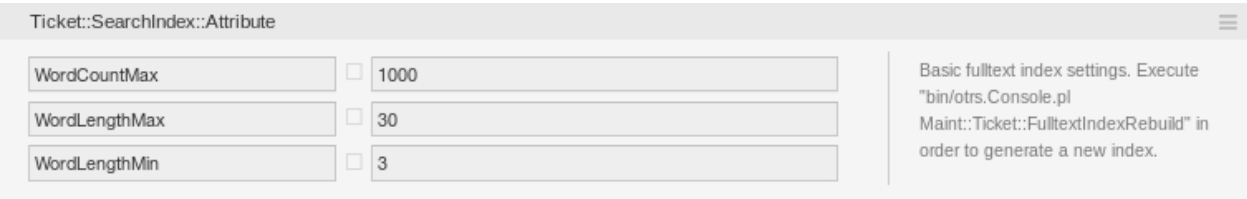


Figure 13.1: Einstellungen `Ticket::SearchIndex::Attribute`

Bemerkung: Führen Sie folgenden Befehl aus, um einen neuen Index zu generieren:

```
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Ticket::FulltextIndexRebuild
```

`WordCountMax` Definiert die maximale Anzahl beim Indexaufbau berücksichtigter Worte. Beispiel: Im Artikel-Suchindex werden nur die ersten 1.000 Wörter des Artikel-Body gespeichert.

`WordLengthMin` und `WordLengthMax` Werden zur Begrenzung der Wortlänge verwendet. Im Artikel-Sucheindex werden nur Wörter mit einer Länge zwischen diesen beiden Werten gespeichert.

`Ticket::SearchIndex::Filters` Filter auf Basis regulärer Ausdrücke schließen Teile des Originaltextes aus dem Volltextindex aus.

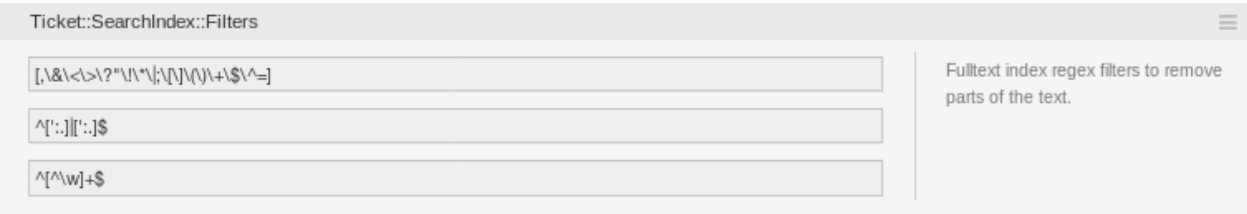


Figure 13.2: Einstellungen `Ticket::SearchIndex::Filters`

Standardmäßig sind drei Filter definiert:

- Der erste Filter sortiert Sonderzeichen aus. Zum Beispiel: , & < > ? “ ! * | ; [ ] ( ) + $ ^ =
- Der zweite Filter sortiert Wörter aus, die mit einem der folgenden Zeichen beginnen oder enden: , : .
- Der dritte Filter sortiert Wörter aus, die keines der folgenden Zeichen enthalten: a-z, A-Z, 0-9, _

`Ticket::SearchIndex::StopWords` Englische Stopwörter für den Volltext-Index. Diese Wörter werden aus dem Suchindex entfernt.

Figure 13.3: `Ticket::SearchIndex::StopWords###en` Einstellungen englische Stopwörter

Für einige Sprachen sind sogenannte Stopwort-Listen definiert. Diese Worte werden beim Anlegen des Suchindex ausgelassen.

Siehe auch:

Möchten Sie zusätzliche Wörter ergänzen, können Sie diese unter der folgenden Einstellung hinzufügen:

- `Ticket::SearchIndex::StopWords###Custom`

13.3 Dokumentensuche

OTOBO nutzt Elasticsearch für die Dokumentensuche. Eine gute Einführung in die Konzeption, Installation und Nutzung von Elasticsearch bietet der Elasticsearch Getting Started Guide.

13.3.1 Heap-Größe

Elasticsearch ist in Java geschrieben und läuft folglich in einer Java Virtual Machine (JVM) auf jedem Clusterknoten. Eine solche JVM nutzt einen Teil des Speichers, den sogenannten Heap. Seine Größe kann in der Konfigurationsdatei `jvm.options` definiert werden.

Mindest- und Maximalkonfiguration für den Heap sind standardmäßig auf einen Wert von 1 GB fesgelegt und können wie folgt angepasst werden:

- `Xms1g`: minimale Heapgröße.
- `Xmx1g`: maximale Heapgröße.

Liegt der `Xms`-Wert unter dem für `Xmx`, passt die JVM den verwendeten Heap bei jeder Überschreitung des aktuellen Limits automatisch an, bis der `Xmx`-Wert erreicht ist. Bei jeder Anpassung wird der Dienst bis zum Ende pausiert, so dass Schnelligkeit und Reaktivität der Suche und Indexierung beeinträchtig werden können. Wir empfehlen deshalb dringend, beide Parameter auf den gleichen Wert zu setzen.

Warnung: Wird die maximale Heapgröße überschritten, stoppt der zugehörige Clusterknoten die Arbeit und kann den Dienst sogar herunterfahren.

Je höher die maximale Heapgröße, desto mehr Speicher kann Elasticsearch nutzen. Damit steigt auch die Anzahl möglicher Pausen durch die Garbage Collection der JVM. Es wird deshalb empfohlen, den Wert für "Xmx"auf maximal 50 % des physischen Speichers zu setzen.

Weitere Informationen und gute Richtwerte für die Heapgröße entnehmen Sie bitte der offiziellen Elasticsearch-Dokumentation.

13.3.2 Festplattennutzung

Elasticsearch prüft während der Laufzeit den verfügbaren Festplattenplatz. Je nach Auslastung werden Clusterknoten weitere Shards zugewiesen oder sogar abgezogen. Dieses Verhalten wird durch die verfügbare Festplattenkapazität gesteuert und kann in der Konfigurationsdatei elasticsearch.yml konfiguriert werden. Hier finden Sie einige wichtige Konfigurationen, für die gute Standardwerte vorgegeben sind, die aber von Bedeutung sein können.

`cluster.routing.allocation.disk.watermark.low` Standardwert 85 %. Wird diese Grenze überschritten, weist Elasticsearch dem jeweiligen Clusterknoten keine weiteren Shards zu. Der Betrieb des jeweiligen Knotens wird nicht beeinträchtigt, Daten können weiterhin indexiert und durchsucht werden.

`cluster.routing.allocation.disk.watermark.high` Standardwert 90 % . Wird dieser Grenzwert überschritten, versucht Elasticsearch vorhandene Shards auf andere Knoten mit ausreichend Platz zu verschieben.

`cluster.routing.allocation.disk.watermark.flood_stage`

Standardwert 95 %. Wird dieser Grenzwert überschritten, aktualisiert Elasticsearch die Konfiguration aller Indizes, von denen mindestens ein Shard dem jeweiligen Clusterknoten zugeordnet ist, auf schreibgeschützte Indexblöcke. Diese werden mit dem Tag `index.blocks.read_only_allow_delete` gekennzeichnet.

Ab diesem Zeitpunkt ist es nicht mehr möglich, neue Daten zum jeweiligen Index hinzuzufügen. Es können nur noch Such- und Löschvorgänge durchgeführt werden.

Bemerkung: Wurde der Grenzwert überschritten und sind bestimmte Indizes auf Readonly gesetzt, wird Elasticsearch diese Konfiguration nicht automatisch wieder zurücksetzen. Haben Sie also von Hand dafür gesorgt, dass wieder genügend Festplattenspeicher verfügbar ist, müssen Sie die Konfiguration von Hand wieder auf Normalmodus zurücksetzen.

Weitere Informationen Hochwassermarken und der Zuweisung von Festplatten-Shards finden Sie in der offiziellen Elasticsearch-Dokumentation.

13.4 Artikelspeicher

Es gibt zwei verschiedene Backendmodule für die Speicherung von Telefon-, E-Mail- und internen Artikeln. Der verwendete Artikelspeicher kann in der Systemkonfiguration unter `Ticket::Article::Backend::MIMEBase::ArticleStorage` definiert werden.

Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB Dieses Standardmodul speichert Anhänge in der Datenbank. Es unterstützt auch mehrere Frontend-Server, benötigt jedoch erheblichen Speicherplatz in der Datenbank.

Bemerkung: Verwenden Sie dieses Modul nicht für große Systeme.

Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS Verwenden Sie dieses Modul, um Anhänge im lokalen Dateisystem zu speichern. Das Modul ist schnell. Achten Sie aber in Umgebungen mit mehreren Frontendservern darauf, dass die Server das Dateisystem gemeinsam nutzen. Verwenden Sie eine NFS-Freigabe oder eine ähnliche Lösung.

Bemerkung: Empfohlen für große Systeme.

Sie können direkt von einem Backed zum anderen wechseln. Sie können das Backend in der Systemkonfiguration ändern und dann dieses Kommandozeilentool ausführen, um Artikel aus der Datenbank in ein Dateisystem zu verlagern - und andersherum:

```
otobo> /opt/otobo/bin/otobo.Console.pl Admin::Article::StorageSwitch --target␣
↪ArticleStorageFS
```

Verwenden Sie die Option `--target`, um das Ziel-Backend festzulegen.

Bemerkung: Der gesamte Prozess kann je nach Anzahl der vorhandenen Artikel sowie verfügbarer CPU-Leistung und Netzwerkkapazität erhebliche Zeit in Anspruch nehmen.

Wenn Sie alte Anhänge in der Datenbank behalten möchten, können Sie die Option `Ticket::Article::Backend::MIMEBase::CheckAllStorageBackends` in der Systemkonfiguration aktivieren, um sicherzustellen, dass OTOBO diese auch weiterhin findet.

13.5 Tickets archivieren

Da OTOBO als revisionssicheres System betrieben werden kann, ist das Löschen geschlossener Tickets möglicherweise nicht empfehlenswert. Es gibt deshalb eine Funktionalität zum Archivieren von Tickets.

Tickets, die bestimmten Kriterien entsprechen, können als archiviert markiert werden. Archivierte Tickets bleiben bei der regulären Ticketsuche und beim Ausführen von Generic Agent-Jobs außen vor. So muss das System nicht bei jeder Anfrage enorme Ticketmengen verarbeiten, stattdessen werden im Regelfall nur die neuesten - relevanten - Tickets berücksichtigt. Insbesondere bei großen Systemen kann dies eine erhebliche Leistungssteigerung verursachen.

So verwenden Sie die Archivierungsfunktion:

1. Aktivieren Sie die Einstellung `Ticket::ArchiveSystem` in der Systemkonfiguration.
2. Definieren Sie einen GenericAgent-Job:
   - Klicken Sie in der Maske GenericAgent auf die Schaltfläche Job hinzufügen.
   - Job-Einstellungen: Geben Sie einen Namen für den Archivierungsauftrag an.
   - Automatische Ausführung: Wählen Sie geeignete Parameter zur Planung des Jobs aus.
   - Tickets selektieren: Es kann empfehlenswert sein, Tickets erst einige Monate nach dem Schließen zu archivieren.

- Ticketattribute aktualisieren / hinzufügen: Setzen Sie das Feld Ausgewählte Tickets archivieren auf Tickets archivieren.
- Speichern Sie Ihre Einstellungen am Ende der Seite.
- Klicken Sie in der Übersichtstabelle auf Diese Aufgabe ausführen, um alle betroffenen Tickets anzuzeigen.
- Klicken Sie auf die Schaltfläche Diesen Job ausführen.

Bemerkung: Durch manuelles Ausführen dieses Jobs können bis zu 5.000 Tickets geändert werden.

In der Ticketsuche werden archivierte Tickets standardmäßig nicht berücksichtigt.

So suchen Sie nach archivierten Tickets:

1. Öffnen Sie die Ticketsuche.
2. Setzen Sie Archivsuche auf Nicht archivierte Tickets oder Alle Tickets.
3. Führen Sie die Suche aus.

13.6 Caching

Ein schnelles Cache-Modul verbessert die Systemleistung erheblich. Wir empfehlen die Verwendung eines Redis Cache Servers oder die Erstellung einer RAM-Disk.

13.6.1 Redis Cache Server installieren

1. Redis Server installieren

Installieren Sie zunächst die neueste Version von Redis Server. Am einfachsten ist es, Redis auf dem gleichen Host wie OTOBO aufzusetzen und an den Standardport zu binden.

2. Redis Perl Modul oder Redis::Fast installieren

Es liegt bei Ihnen, welches Redis-Modul Sie nutzen: Redis oder Redis::Fast (das mit Redis vergleichbar aber **~2x schneller** ist). Die Ausgabe des Befehls `otobo.CheckModules.pl --list` unterstützt Sie bei der Wahl des passenden Paketes für Ihre Installation:

```
otobo> /opt/otobo/bin/otobo.CheckModules.pl --all
```

3. OTOBO für Redis konfigurieren

Bitte verwenden Sie die OTOBO SysConfig (Administration -> Systemkonfiguration), um OTOBO entsprechend zu konfigurieren:

```
| Setting                       | Description                | Default value  |
| ----------------------------- | -------------------------- | -------------- |
| Cache::Redis###Server         | Redis server URL           | 127.0.0.1:6379 |
| Cache::Redis###DatabaseNumber | Number of logical database | 0              |
| Cache::Redis###RedisFast      | Use or not Redis::Fast     | 0              |
| Cache::Module                 | Activate Redis Cache Module| DB (use Redis) |
```

13.6.2 RAM-Disk-Caching

OTOBO speichert viele temporäre Daten in `/opt/otobo/var/tmp`. Stellen Sie sicher, dass für dieses Verzeichnis ein Hochleistungsdateisystem und -speicher eingesetzt wird. Wenn Sie über genügend RAM verfügen, können Sie auch versuchen, dieses Verzeichnis wie folgt auf einer RAM-Disk abzulegen:

```
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Session::DeleteAll
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Cache::Delete
root> mount -o size=16G -t tmpfs none /opt/otobo/var/tmp
```

Bemerkung: Fügen Sie einen dauerhaften Mountpunkt in “/etc/fstab/“hinzu.

Warnung: Dies ist ein nicht permanenter Speicher, der bei einem Server-Neustart verloren geht. All Ihre Sitzungen (sofern im Dateisystem gespeichert) und Ihr Cache sind dann verloren.

13.7 Cluster

Bei sehr hohen Lasten kann es erforderlich sein, OTOBO auf einem Cluster aus mehreren Frontend-servern zu betreiben. Dieses Szenario ist komplex und birgt viele Fallstricke. Wir empfehlen dringend, sich mit unseren Experten in Verbindung zu setzen, bevor Sie ein solches Projekt alleine umsetzen.

CHAPTER 14

Dokumentationshistorie

1. 2019 - OTRS Installation Guide - OTRS AG (https://otrs.com)
2. 2020 - OTOBO Installationsanleitung - Rother OSS GmbH (https://otobo.de)

Veröffentlicht durch: Rother OSS GmbH (https://rother-oss.com), Oberwalting 31, 94339 Leiblfing, Deutschland. Autoren: OTRS AG (Originalversion), Rother OSS GmbH (Anpassung für OTOBO, https://otobo.de)