



# OTOB Installation Guide

Release 10.1

Rother OSS GmbH

May 12, 2024



---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	About This Manual . . . . .	3
<b>2</b>	<b>Hardware and Software Requirements</b>	<b>5</b>
2.1	Hardware Requirements . . . . .	7
2.2	Software requirements . . . . .	7
<b>3</b>	<b>OTOBO Installation</b>	<b>9</b>
3.1	Preparation: Disable SELinux when it is installed and enabled . . . . .	9
3.2	Step 1: Unpack and Install OTOBO . . . . .	10
3.3	Step 2: Install Additional Programs and Perl Modules . . . . .	10
3.4	Step 3: Create the OTOBO User . . . . .	11
3.5	Step 4: Activate the Default Configuration File . . . . .	11
3.6	Step 5: Configure the Apache Web Server . . . . .	11
	3.6.1 Configure Apache without SSL support . . . . .	12
	3.6.2 Configure Apache <b>with</b> SSL support . . . . .	13
3.7	Step 6: Set File Permissions . . . . .	13
3.8	Step 7: Setup the Database . . . . .	13
3.9	Step 8: Setup Elasticsearch . . . . .	14
	3.9.1 Elasticsearch installation example based on Ubuntu 18.04 LTS . . . . .	15
	3.9.2 Elasticsearch Installation on another Linux distribution . . . . .	15
	3.9.3 Elasticsearch Module Installation . . . . .	15
	3.9.4 Elasticsearch Configuration . . . . .	15
3.10	Step 9: Basic System Configuration . . . . .	16
3.11	Step 10: First Login . . . . .	16
3.12	Step 11: Start the OTOBO Daemon . . . . .	16
3.13	Step 12: Cron jobs for the OTOBO user . . . . .	16
3.14	Step 13: Setup Bash Auto-Completion (optional) . . . . .	16
3.15	Step 14: Further Information . . . . .	17
<b>4</b>	<b>Installing using Docker and Docker Compose</b>	<b>19</b>
4.1	Requirements . . . . .	19
4.2	Installation . . . . .	20
	4.2.1 1. Clone the otobo-docker repo . . . . .	20
	4.2.2 2. Create an initial .env file . . . . .	20
	4.2.3 3. Configure the password for the database admin user . . . . .	21
	4.2.4 4. Set up a volume with SSL configuration for the nginx webproxy (optional) . . . . .	21

4.2.5	5. Start the Docker containers with Docker Compose . . . . .	22
4.2.6	6. Install and start OTOBO . . . . .	22
4.3	Additional technical information . . . . .	22
4.3.1	List of Docker containers . . . . .	22
4.3.2	Overview over the Docker volumes . . . . .	22
4.3.3	Docker environment variables . . . . .	23
4.4	Advanced topics . . . . .	24
4.4.1	Custom configuration of the nginx webproxy . . . . .	24
4.4.2	Single Sign On Using the Kerberos Support in Nginx . . . . .	25
4.4.3	Choosing non-standard ports . . . . .	26
4.4.4	Skip startup of specific services . . . . .	26
4.4.5	Prepare offline installation . . . . .	26
4.4.6	Customizing OTOBO Docker Compose . . . . .	27
4.4.7	Customizing the OTOBO Docker image . . . . .	27
4.4.8	Building local images . . . . .	28
4.4.9	Automatic Installation . . . . .	29
4.4.10	List of useful commands . . . . .	29
4.5	Resources . . . . .	30
<b>5</b>	<b>Migration from OTRS 6 or OTRS 7 / ((OTRS)) Community Edition to OTOBO version 10.1</b>	<b>31</b>
5.1	Overview over the Supported Migration Szenarios . . . . .	31
5.2	Migration Requirements . . . . .	32
5.3	Step 1: Install the new OTOBO System . . . . .	33
5.4	Step 2: Deactivate <code>SecureMode</code> on OTOBO . . . . .	34
5.5	Step 3: Stop the OTOBO Daemon . . . . .	34
5.6	Optional Step: Mount <code>/opt/otrs</code> for Convenient Access . . . . .	34
5.7	Optional Step: Install <code>sshpas</code> and <code>rsync</code> when <code>/opt/otrs</code> Should be Copied via <code>ssh</code> . . . . .	35
5.8	Step 4: Preparing the OTRS / ((OTRS)) Community Edition system . . . . .	35
5.8.1	Stop All Relevant Services and the OTRS Daemon . . . . .	35
5.8.2	Clear the Caches and the Operational Data . . . . .	35
5.9	Optional Step for Docker: make required data available inside container . . . . .	36
5.9.1	Copy <code>/opt/otrs</code> into the volume <code>otobo_opt_otobo</code> . . . . .	36
5.10	Step 5: Perform the Migration! . . . . .	37
5.11	Step 6: After Successful Migration! . . . . .	38
5.12	Known Migration Problems . . . . .	38
5.12.1	1. Login after migration not possible . . . . .	38
5.12.2	2. Final page of the migration has a strange layout due to missing CSS files . . . . .	38
5.12.3	3. Migration stops due to MySQL errors . . . . .	38
5.12.4	4. Errors in Step 5 when migrating to PostgreSQL . . . . .	38
5.12.5	5. Problems with the Deployment the Merged System Configuration . . . . .	39
5.13	Step 7: Manual Migration Tasks and Changes . . . . .	39
5.13.1	1. Password policy rules . . . . .	39
5.13.2	2. Under Docker: Manually migrate cron jobs . . . . .	40
5.14	Special topics . . . . .	40
5.14.1	Migration from Oracle to Oracle . . . . .	40
5.14.2	Optional Step: Streamlined migration of the database (only for experts and spezial szenarios) . . . . .	41
<b>6</b>	<b>Updating</b>	<b>43</b>
6.1	Step 1: Stop All Relevant Services and the OTOBO Daemon . . . . .	43
6.2	Step 2: Backup Files and Database . . . . .	44
6.2.1	Example for a standard installation with Ubuntu and MySQL . . . . .	44
6.3	Step 3: Install the New Release . . . . .	44
6.3.1	Restore Old Configuration Files . . . . .	44

6.3.2	Restore Article Data	45
6.3.3	Restore Already Installed Default Statistics	45
6.3.4	Set File Permissions	45
6.3.5	Check Apache configuration files	45
6.4	Step 4: Check for new needed perl modules	45
6.5	Step 5: Update Installed Packages and reconfigure config	46
6.6	Step 6: Only for minor or major release upgrades (for example to upgrade from 10.0 to 10.1)	46
6.7	Step 7: Start your Services	46
<b>7</b>	<b>Updating a Docker-based Installation of OTOBO</b>	<b>47</b>
7.1	Updating the Docker Compose files	47
7.2	Checking the Docker Compose .env file	48
7.3	Fetch the new Docker images	48
7.4	Update OTOBO	48
<b>8</b>	<b>Backup and Restore</b>	<b>51</b>
8.1	Backup	51
8.2	Restore	52
8.3	Considerations for running OTOBO under Docker	52
<b>9</b>	<b>Backup and Restore using Docker</b>	<b>55</b>
9.1	Considerations for running OTOBO under Docker	55
<b>10</b>	<b>Kerberos Single Sign On in OTOBO Docker installation</b>	<b>57</b>
10.1	Generate Active Directory User	57
10.2	Generate Active Directory Keytab file	57
10.3	Create a new volume for your custom nginx configuration	59
10.4	Create new OTOBO .env file	59
10.5	Start OTOBO	60
10.6	Tell OTOBO to use the Kerberos-Authentication	60
10.7	Configure Browser to understand Kerberos SSO	60
10.8	Debugging and Problems	61
10.8.1	Kerberos debugging	61
<b>11</b>	<b>Adapt customer interface with corporate identity</b>	<b>63</b>
11.1	Change colors in Customer Area	63
11.2	Change Logos and Pictures	63
11.2.1	Change Customer Login Pictures and Text	64
11.2.2	Change Customer Dashboard tiles and options	65
<b>12</b>	<b>Installing Perl Modules from CPAN</b>	<b>67</b>
12.1	Docker-based installations	67
<b>13</b>	<b>Performance Tuning</b>	<b>69</b>
13.1	Ticket Index Module	69
13.2	Ticket Search Index	69
13.3	Document Search	71
13.3.1	Heap Size	71
13.3.2	Disk Allocation	72
13.4	Article Storage	72
13.5	Archiving Tickets	73
13.6	Caching	74
13.6.1	Install a Redis Cache Server	74
13.6.2	RamDisk Caching	74

13.7 Clustering . . . . .	75
<b>14 Documentation History</b>	<b>77</b>



This work is copyrighted by OTRS AG (<https://otrs.com>), Zimmersmühlenweg 11, 61440 Oberursel, Germany.

Copyright © for modifications and amendments 2019-2022 ROTHER OSS GmbH (<https://otobo.de>), Oberwaling 31, 94339 Leiblfing, Germany

Terms and Conditions OTRS: Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found on the GNU website.

Terms and Conditions Rother OSS: Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “COPYING”.

Published by: Rother OSS GmbH, (<https://otobo.de>), Oberwaling 31, 94339 Leiblfing, Germany.

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





OTOBO is an open source ticket request system with many features to manage customer telephone calls and emails. It is distributed under the GNU General Public License (GPL) and is tested on various Linux platforms.

### 1.1 About This Manual

This manual is intended for use by system administrators. The chapters describe the installation and updating of the OTOBO software.

There is no graphical user interface for installation and updating. System administrators have to follow the steps described in the following chapters.

All console commands look like `username> command-to-execute`. Username indicates the user account of the operating system, which need to use to execute the command. If a command starts with `root>`, you have to execute the command as a user with root permissions. If a command starts with `otobo>`, you have to execute the command as the user created for OTOBO.

**Warning:** Do not select `username>` when you copy the command and paste it to the shell. Otherwise you will get an error.

We assume that OTOBO will be installed to the directory `/opt/otobo`. If you want to install OTOBO to a different location, then you have to change the path in the commands or create a symbolic link to this directory.

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



---

## Hardware and Software Requirements

---

The OTOBO web application can be installed on Linux and other Unix derivatives, e.g. OpenBSD or FreeBSD. Running OTOBO on Microsoft Windows is not supported.

The web application uses a relational database as backend. So, to run OTOBO, you'll need to run at least a web server and a database server. The web server and the database server may be installed either on the same or on different hosts.

Alternatively, OTOBO can also run under Docker. When running under Docker, the web and the database server are already included in the setup. Support for deployment with Kubernetes is under development.

The OTOBO web application requires Perl along with additional Perl modules from CPAN. The modules can be installed either with a Perl package manager or via the package manager of your operating system (rpm, yast, apt-get). There is a console command for checking the module dependencies:

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

If some packages are missing, you can get an install command for your operating system by running the script with the `--list` option.

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

The listed commands should then be executed with root privileges.

The output of the module check script shows the installed packages and the version numbers. Missing modules are marked with a comment.

```
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 Requirements

Hardware requirements highly depend on the usage of OTOBO. OTOBO can be used to process a few tickets per month or to process hundreds of tickets per day. The storage requirement also depends on the number of tickets and size of attachments.

We recommend using a machine for testing purposes with **at least**:

- small CPU
- 4 GB RAM
- 10 GB storage

We recommend using a machine for production purpose with **at least**:

- 3 GHz Xeon or comparable CPU
- 8 GB RAM (16 GB recommend)
- 40 GB storage

---

**Note:** Hardware requirements depend on the usage of OTOBO. Please contact your OTOBO consultant before deploying any hardware.

---

## 2.2 Software requirements

### Perl

- Perl 5.24.0 or higher
- Perl packages listed by `/opt/otobo/bin/otobo.CheckModules.pl --list console` command

### Web Server

- Apache HTTP Server Version 2.4

### Databases

- MySQL 5.6 or higher
- MariaDB
- PostgreSQL 9.2 or higher
- Oracle 10g or higher

### Optional

- Elasticsearch 7.x (fast search function for live previews)
- Redis (fast caching)
- nginx or any other web server that can be used as a reverse proxy (SSL support and load distribution)

### Web browsers

- Apple Safari
- Google Chrome
- Microsoft Internet Explorer 11
- Microsoft Edge
- Mozilla Firefox
- Any other modern web browser with JavaScript support

---

## OTOBO Installation

---

This chapter describes the installation and basic configuration of the central OTOBO framework.

Follow the detailed steps in this chapter to install OTOBO on your server. You can then use its web interface to login and administer the system.

---

**Note:** As of OTOBO version 10.0.7, we recommend Docker and Docker Compose for the OTOBO installation. By using the provided Docker images, all recommended dependencies (such as Elasticsearch, Redis Cache, etc.) are installed and configured automatically. Updates are thus greatly simplified and the performance has been improved. You can find the instructions for Docker-based installation at <https://doc.otobo.org/manual/installation/10.1/en/content/installation-docker.html>.

---

### 3.1 Preparation: Disable SELinux when it is installed and enabled

---

**Note:** If your system uses SELinux, you should disable it, otherwise OTOBO will not work correctly.

---

Try the command `sestatus` and `getenforce` when you are not sure whether SELinux is installed and enabled on your system.

The `sestatus` command returns the SELinux status and the SELinux policy being used. SELinux status: enabled is returned when SELinux is enabled. Current mode: enforcing is returned when SELinux is running in enforcing mode. Policy from config file: targeted is returned when the SELinux targeted policy is used.

Here's how to disable SELinux for RHEL/CentOS/Fedora.

1. Configure `SELINUX=disabled` in the `/etc/selinux/config` file:

```
# 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. Reboot your system. After reboot, confirm that the `getenforce` command returns Disabled:

```
root> getenforce
Disabled
```

## 3.2 Step 1: Unpack and Install OTOBO

Download the latest otobo release from <https://ftp.otobo.org/pub/otobo/>. Unpack the source archive (for example, using `tar`) into the directory `/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-10.1.tar.gz # Download he
↳latest OTOBO 10 release
root> tar -xzf otobo-latest-10.1.tar.gz # Unzip OTOBO
root> cp -r otobo-10.x.x/* /opt/otobo # Copy the
↳new otobo directory to /opt/otobo
```

## 3.3 Step 2: Install Additional Programs and Perl Modules

Use the following script to get an overview of all installed and required CPAN modules and other external dependencies.

**Note:** On Debian systems you may need to manually install some perl packages:

```
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)
...
```

**Note:** Please note that OTOBO requires a working Perl installation with all core modules such as the module `version`. These modules are not explicitly checked by the script. You may need to install a `perl-core` package on some systems like RHEL that do not install the Perl core packages by default.

To install the required and optional packages, you can use either CPAN or the package manager of your Linux distribution.

Execute this command to get an install command to install the missing dependencies:

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

**Note:** There are a number of optional or alternative modules which can be installed, mostly for more customized versions of OTOBO. Calling `CheckModules.pl` without any argument will list its full functionality.

### 3.4 Step 3: Create the OTOBO User

Create a dedicated user for OTOBO within its own group:

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

Add the user to web server group (if the web server is not running as otobo user):

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

### 3.5 Step 4: Activate the Default Configuration File

There is an OTOBO configuration file bundled in `$OTOBO_HOME/Kernel/Config.pm.dist`. You must activate it by copying it without the `.dist` file name extension.

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

### 3.6 Step 5: Configure the Apache Web Server

First of all, you should install the Apache2 web server and `mod_perl`; you'd typically do this from your system's package manager. Below you'll find the commands needed to set up Apache on the most popular Linux distributions.

```
# 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
```

A critical setting of the Apache web server is the choice of the multi-processing module. For running OTOBO, the recommended choice is the module **mpm\_prefork**. Like other Apache modules the multi-processing module can be managed with the tools `a2dismod` and `a2enmod`.

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

All is fine when `mpm_prefork` already is enabled.

Disable `mpm_event` when it is currently active.

```
root> a2dismod mpm_event
```

Disable `mpm_worker` in case that MPM is enabled.

```
root> a2dismod mpm_worker
```

Finally activate `mpm_prefork`.

```
root> a2enmod mpm_prefork
```

OTOBO requires a few more Apache modules to be active for optimal operation. Again, on most platforms you can make sure they are active via the tool `a2enmod`.

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

---

**Note:** On some platforms not all Apache modules exist and an error is displayed when installing. Do not worry and finish the installation, in most cases the module will not be needed.

---

Most Apache installations have a `conf.d` directory included. On Linux systems you can usually find this directory under `/etc/apache` or `/etc/apache2`.

### 3.6.1 Configure Apache without SSL support

Copy the template file `/opt/otobo/scripts/apache2-httpd.include.conf` to the `apache sites-available` directory. In most cases no further editing of the template is required. Then enable the new configuration.

```
# 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 Configure Apache with SSL support

Copy the template files `/opt/otobo/scripts/apache2-httpd-vhost-80.include.conf` and `/opt/otobo/scripts/apache2-httpd-vhost-443.include.conf` to the `apache sites-available` directory.

```
# 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
```

Please edit the files and add the required information like SSL certificate storage path. After that, enable the OTOBO Apache configuration:

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

Now you can restart your web server to load the new configuration settings. On most systems you can use the following command to do so:

```
root> systemctl restart apache2
```

## 3.7 Step 6: Set File Permissions

Please execute the following command to set the file and directory permissions for OTOBO. It will try to detect the correct user and group settings needed for your setup.

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

## 3.8 Step 7: Setup the Database

First of all, you should install the database package. It is recommended to use the MySQL or MariaDB package, which will be delivered with your Linux system, but it is possible to use PostgreSQL or Oracle as well.

You'd typically do this from your systems package manager. Below you'll find the commands needed to set up MySQL on the most popular Linux distributions.

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

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

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

After installing the MySQL server you need configure it.

In MySQL higher or equal version 5.7 a new authentication module is active, and it is not possible to use the OTOBO web installer for database creation. Please login to the `mysql` console and set a different authentication module and password for the user `root` if this is the case:

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

For MariaDB > 10.1 use instead the following command:

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

If this command not work, please try the following commands:

```
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';
```

After OTOBO installation it is possible to change the authentication module again, if needed.

---

**Note:** The following configuration settings are minimum requirements for MySQL setups. Please add the following lines to the MySQL Server configuration file `/etc/my.cnf`, `/etc/mysql/my.cnf` or `/etc/mysql/mysql.conf.d/mysqld.cnf` under the `[mysqld]` section:

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

For MySQL prior to MySQL 8.0 the query cache size should also be set:

```
query_cache_size = 32M
```

Also add the following lines to the MySQL Server configuration file `/etc/my.cnf`, `/etc/mysql/my.cnf` or `/etc/mysql/mysql.conf.d/mysqldump.cnf` under the `[mysqldump]` section:

```
max_allowed_packet = 64M
```

---

For production purposes we recommend to use the tool `mysqltuner` to find the perfect setup. You can download the script from github <https://github.com/major/MySQLTuner-perl> or install it on Debian or Ubuntu systems via package manager:

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

After installing execute the script:

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

### 3.9 Step 8: Setup Elasticsearch

OTOBO recommends an active installation of Elasticsearch for quick search. The easiest way is to setup Elasticsearch on the same host as OTOBO and binding it to its default port.

### 3.9.1 Elasticsearch installation example based on 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 on another Linux distribution

Please follow the installation tutorial found at <https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html>.

### 3.9.3 Elasticsearch Module Installation

Additionally, OTOBO requires plugins to be installed into 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 Configuration

Elasticsearch has a multitude of configuration options and possibilities.

In order to ensure error-free operation, you should adjust the jvm heap space for larger OTOBO systems. Please adjust the settings in the file `/etc/elasticsearch/jvm.options`. You should always set the min and max JVM heap size to the same value. For example, to set the heap to 4 GB, set:

```
-Xms4g
-Xmx4g
```

In our tests, a value between 4 and 10 GB for medium-sized installations has proven to be the best.

---

**Note:** See <https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html> for more information.

---

Now you can restart your Elasticsearch server to load the new configuration settings. On most systems you can use the following command to do so:

```
root> systemctl restart elasticsearch
```

## 3.10 Step 9: Basic System Configuration

Please use the web installer at <http://localhost/otobo/installer.pl> (replace “localhost” with your OTOBO hostname) to set up your database and basic system settings such as email accounts.

## 3.11 Step 10: First Login

Now you are ready to login to your system at <http://localhost/otobo/index.pl> as user `root@localhost` with the password that was generated (see above).

## 3.12 Step 11: Start the OTOBO Daemon

OTOBO daemon is responsible for handling any asynchronous and recurring tasks in OTOBO. What has been in cron file definitions previously is now handled by the OTOBO daemon, which is required to operate OTOBO. The daemon also handles all GenericAgent jobs and must be started from the OTOBO user.

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

## 3.13 Step 12: Cron jobs for the OTOBO user

There are two default OTOBO cron files in `/opt/otobo/var/cron/*.dist`, and their purpose is to make sure that the OTOBO Daemon is running. They need to be activated by copying them without the “.dist” filename extension.

```
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
```

With this step, the basic system setup is finished.

## 3.14 Step 13: Setup Bash Auto-Completion (optional)

All regular OTOBO command line operations happen via the OTOBO console interface. This provides an auto-completion for the bash shell which makes finding the right command and options much easier.

You can activate the bash auto-completion by installing the package `bash-completion`. It will automatically detect and load the file `/opt/otobo/.bash_completion` for the `otobo` user.

After restarting your shell, you can just type this command followed by TAB, and it will list all available commands:

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

If you type a few characters of the command name, TAB will show all matching commands. After typing a complete command, all possible options and arguments will be shown by pressing TAB.

---

**Note:** If you have problems, you can execute the following line as user `otobo` and add it to your `~/ .bashrc` to execute the commands from the file.

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

---

### 3.15 Step 14: Further Information

We advise you to read the OTOBO [Performance Tuning](#) chapter.





---

## Installing using Docker and Docker Compose

---

With the dockerized OTOBO deployment you can get your personal OTOBO instance up and running within minutes. All of OTOBO's dependencies are already included in the provided collection of Docker images.

- Service db: MariaDB is set up as the default database.
- Service elastic: Elasticsearch is set up for the OTOBO power search.
- Service redis: Redis is enabled for fast caching.
- Service web: Gazelle is used as fast Perl webserver.
- Service nginx: Nginx is used as optional reverse proxy for HTTPS support.

We think that this setup is the perfect environment for an OTOBO installation.

### 4.1 Requirements

The minimal versions of required software, that have been tested, are listed here:

- Docker 19.03.13
- Docker Compose 1.25.0
- Git 2.17.1

---

**Note:** To get the required minimal versions on Ubuntu 18.04 follow the instructions in <https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-ubuntu-18-04> and <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04>.

---

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

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

Please check the Git and the Docker documentation for instructions on further setup.

## 4.2 Installation

The following instructions assume that all requirements are met, that you have a working Docker environment. We assume here that the user **docker\_admin** is used for interacting with Docker. The Docker admin may be either the **root** user of the Docker host or a dedicated user with the required permissions.

### 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 branch that corresponds to the current version of OTOBO. For example, when OTOBO 10.0.15 is the current version then please use the branch `rel-10_0`.

---

**Note:** The location of the cloned repository does not matter. For these instructions we chose `/opt/otobo-docker` as the working dir.

---

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo-docker.git --branch
↔<BRANCH> --single-branch
docker_admin> ls otobo-docker      # just a sanity check, README.md should exist
```

### 4.2.2 2. Create an initial .env file

The Docker Compose configuration file `.env` is your primary interface for managing your installation of OTOBO. This file must first be created and then be adapted by yourself. In order to simplify the task there are several example files that should be used as starting point. Which sample file it the best fit depends on your use case. In most cases the decision is between `.docker_compose_env_http` and `.docker_compose_env_https`, depending on whether TLS must be supported or not. The other files are for more specialised use cases.

**.docker\_compose\_env\_http** The OTOBO web app provides HTTP.

**.docker\_compose\_env\_https** The OTOBO web app provides HTTPS by running Nginx as a reverse proxy webserver.

**.docker\_compose\_env\_https\_custom\_nginx** Like `.docker_compose_env_https` but with support for a custom Nginx configuration.

**.docker\_compose\_env\_https\_kerberos** Like `.docker_compose_env_https` but with sample setup for single sign on. Note that Kerberos support is still **experimental**.

**.docker\_compose\_env\_http\_selenium** and **.docker\_compose\_env\_https\_selenium** These are used only for development when Selenium testing is activated.

---

**Note:** Use `ls -a` for listing the hidden sample files.

---

Per default OTOBO is served on the standard ports. Port 443 for HTTPS and port 80 for HTTP. When HTTPS is activated then the OTOBO web application actually still runs with HTTP. HTTPS support is achieved by an additional reverse proxy, which is implemented as a nginx service.

For the following commands we assume that HTTPS should be supported.

```
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

Change the following setting inside your .env file:

```
OTOBO_DB_ROOT_PASSWORD=<your_secret_password>
```

The password for the database admin user may be chosen freely. The database admin user is needed to create the database user **otobo** and the database schema **otobo**. OTOBO will actually use the dedicated database user **otobo**.

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

This step can be skipped when OTOBO should be available only via HTTP.

nginx needs for SSL encryption a certificate and a private key.

---

**Note:** For testing and development a self-signed certificate can be used. However for productive use you should work with regular registered certificates.

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.

---

**Note:** To specify a CA chain with a certificate in nginx, it is necessary to copy the CA chain file with the actual certificate into a file.

---

The certificate and the private key are stored in a volume, so that they can be used by nginx later on. In any case the volume needs to be generated manually, and we need to copy the certificate and key to the volume:

```
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
```

The names of the copied files need to be set in our newly created .env file. E.g.

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

Please adapt only the name of the files as the path /etc/nginx/ssl/ is hard coded in the Docker image.

## 4.2.5 5. Start the Docker containers with Docker Compose

Now we start the Docker containers using `docker-compose`. Per default the Docker images will be fetched from <https://hub.docker.com/u/rotheross>.

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

To verify that the six required services (five in the case of HTTP only) are actually running, type:

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

## 4.2.6 6. Install and start OTOBO

Run the OTOBO installer at <http://yourIPorFQDN/otobo/installer.pl>.

---

**Note:** Please configure OTOBO inside the installer with a new MySQL database. As MySQL database root password please use the password you configured in the variable `OTOBO_DB_ROOT_PASSWORD` of your `.env` file. Please leave the value `db` for the MySQL hostname untouched.

---

**Have fun with OTOBO!**

---

**Note:** To change to the OTOBO directory, inside the running container, to work on command line as usual, you can use the following Docker command: `docker-compose exec web bash`.

---

## 4.3 Additional technical information

This section gives some more technical insight into what is happening under the hood.

### 4.3.1 List of Docker containers

**Container `otobo_web_1`** OTOBO webserver on internal port 5000.

**Container `otobo_daemon_1`** OTOBO daemon. The OTOBO daemon is started and periodically checked.

**Container `otobo_db_1`** Run the database MariaDB on internal port 3306.

**Container `otobo_elastic_1`** Elasticsearch on the internal ports 9200 and 9300.

**Container `otobo_redis_1`** Run Redis as caching service.

**Optional container `otobo_nginx_1`** Run nginx as reverse proxy for providing HTTPS support.

### 4.3.2 Overview over the Docker volumes

The Docker volumes are created on the host for persistent data. These allow starting and stopping the services without losing data. Keep in mind that containers are temporary and only data in the volumes is permanent.

**otobo\_opt\_otobo** contains /opt/otobo in the container **web** and **daemon**.

**otobo\_mariadb\_data** contains /var/lib/mysql in the container **db**.

**otobo\_elasticsearch\_data** contains /usr/share/elasticsearch/data in the container **elastic**.

**otobo\_redis\_data** contains data for the container **redis**.

**otobo\_nginx\_ssl** contains the TLS files, certificate and private key, must be initialized manually.

### 4.3.3 Docker environment variables

In the instructions we did only minimal configuration. But the file `.env` allows to set more variables. Here is a short list of the most important environment variables. Note that more environment variables are supported by the base images.

#### MariaDB settings

**OTOBO\_DB\_ROOT\_PASSWORD** The root password for MariaDB. This setting is required for running the service **db**.

#### Elasticsearch settings

Elasticsearch needs some settings for productive environments. Please read <https://www.elastic.co/guide/en/elasticsearch/reference/7.8/docker.html#docker-prod-prerequisites> for detailed information.

**OTOBO\_Elasticsearch\_ES\_JAVA\_OPTS** Example setting: `OTOBO_Elasticsearch_ES_JAVA_OPTS=-Xms512m -Xmx512m` Please adjust this value for production env to a value up to 4g.

#### Webserver settings

**OTOBO\_WEB\_HTTP\_PORT** Set in case the HTTP port should deviate from the standard port 80. When HTTPS is enabled, the HTTP port will redirect to HTTPS.

#### Nginx webproxy settings

These setting are used when HTTPS is enabled.

**OTOBO\_WEB\_HTTP\_PORT** Set in case the HTTP port should deviate from the standard port 80. Will redirect to HTTPS.

**OTOBO\_WEB\_HTTPS\_PORT** Set in case the HTTPS port should deviate from the standard port 443.

**OTOBO\_NGINX\_SSL\_CERTIFICATE** SSL cert for the nginx webproxy. Example:  
`OTOBO_NGINX_SSL_CERTIFICATE=/etc/nginx/ssl/acme.crt`

**OTOBO\_NGINX\_SSL\_CERTIFICATE\_KEY** SSL key for the nginx webproxy. Example:  
`OTOBO_NGINX_SSL_CERTIFICATE_KEY=/etc/nginx/ssl/acme.key`

#### Nginx webproxy settings for Kerberos

This settings are used by Nginx when Kerberos is used for single sign on.

**OTOBO\_NGINX\_KERBEROS\_KEYTAB** Kerberos keytab file. The default is `/etc/krb5.keytab`.

**OTOBO\_NGINX\_KERBEROS\_CONFIG** Kerberos config file. The default is `/etc/krb5.conf`, usually generated from `krb5.conf.template`

**OTOBO\_NGINX\_KERBEROS\_SERVICE\_NAME** Kerberos Service Name. It is not clear where this setting is actually used anywhere.

**OTOBO\_NGINX\_KERBEROS\_REALM** Kerberos REALM. Used in `/etc/krb5.conf`.

**OTOBO\_NGINX\_KERBEROS\_KDC** Kerberos kdc / AD Controller. Used in `/etc/krb5.conf`.

**OTOBO\_NGINX\_KERBEROS\_ADMIN\_SERVER** Kerberos Admin Server. Used in /etc/krb5.conf.

**OTOBO\_NGINX\_KERBEROS\_DEFAULT\_DOMAIN** Kerberos Default Domain. Used in /etc/krb5.conf.

**NGINX\_ENVSUBST\_TEMPLATE\_DIR** Provide a custom Nginx config template dir. Gives extra flexibility.

### Docker Compose settings

These settings are used by Docker Compose directly.

**COMPOSE\_PROJECT\_NAME** The project name is used as the prefix for the volumes and containers. Per default this prefix is set to `otobo`, resulting in container names like `otobo_web_1` and `otobo_db_1`. Change this name when you want to run more than one instance of OTOBO on the same server.

**COMPOSE\_PATH\_SEPARATOR** Separator for the value of `COMPOSE_FILE`

**COMPOSE\_FILE** Use `docker-compose/otobo-base.yml` as the base and add the wanted extension files. E.g `docker-compose/otobo-override-http.yml` or `docker-compose/otobo-override-https.yml`.

**OTOBO\_IMAGE\_OTOBO, OTOBO\_IMAGE\_OTOBO\_ELASTICSEARCH, OTOBO\_IMAGE\_OTOBO\_NGINX, ...**  
Used for specifying alternative Docker images. Useful for testing local builds or for using updated versions of the images.

## 4.4 Advanced topics

### 4.4.1 Custom configuration of the nginx webproxy

The container `otobo_nginx_1` provides HTTPS support by running Nginx as a reverse proxy. The Docker image that runs in the container is composed of the official Nginx Docker image, [https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx), along with a OTOBO specific configuration of Nginx.

The default OTOBO specific configuration can be found within the Docker image at `/etc/nginx/template/otobo_nginx.conf.template`. Actually, this is only a template for the final configuration. There is a process, provided by the Nginx base image, that replaces the macros in the template with the corresponding environment variable. This process runs when the container starts up. In the default template file, the following macros are used:

**OTOBO\_NGINX\_SSL\_CERTIFICATE** For configuring SSL.

**OTOBO\_NGINX\_SSL\_CERTIFICATE\_KEY** For configuring SSL.

**OTOBO\_NGINX\_WEB\_HOST** The internally used HTTP host.

**OTOBO\_NGINX\_WEB\_PORT** The internally used HTTP port.

See step 4. for how this configuration possibility was used for setting up the SSL certificate.

**Warning:** The following approach is only supported in OTOBO 10.0.4 or later.

When the standard macros are not sufficient, then the customisation can go further. This can be achieved by replacing the default config template with a customized version. It is best practice to not simply change the configuration in the running container. Instead we first create a persistent volume that contains the custom config. Then we tell the `otobo_nginx_1` to mount the new volume and to use the customized configuration.

First comes generation of the new volume. In these sample commands, we use the existing template as a starting point.

```

# 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-10_0 # or latest-10_1, use the appropriate label
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

```

**Warning:** Your adapted nginx configuration usually contains the directive `listen`, which declares the ports of the webserver. The internally used ports have changed between OTOBO 10.0.3 and OTOBO 10.0.4. This change must be reflected in the adapted nginx configuration. So for version 10.0.3 or earlier listen to the ports 80 and 443. For OTOBO 10.0.4 listen to the ports 8080 and 8443.

After setting up the volume, the adapted configuration must be activated. The new volume is set up in `docker-compose/otobo-nginx-custom-config.yml`. Therefore this file must be added to `COMPOSE_FILE`. Then Nginx must be directed to use the new config. This is done by setting `NGINX_ENVSUBST_TEMPLATE_DIR` in the environment. In order to achieve this, uncomment or add the following lines in your `.env` file:

```

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

```

The changed Docker Compose configuration can be inspected with:

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

Finally, the containers can be started again:

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

See also the section “Using environment variables in nginx configuration (new in 1.19)” in [https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx).

## 4.4.2 Single Sign On Using the Kerberos Support in Nginx

### Short Description

For enabling authentication with Kerberos please base you .env file on the sample file `.docker_compose_env_https_kerberos`. This activates the special configuration in `docker-compose/otobo-override-https-kerberos.yml`. This Docker compose configuration file selects a Nginx image that supports Kerberos. It also passes some Kerberos specific settings as environment values to the running Nginx container. These settings are listed above.

As usual, the values for these setting can be specified in the .env file. Most of ghesse setting will be used as replacement values for the template [https://github.com/RotherOSS/otobo/blob/rel-10\\_1/scripts/nginx/kerberos/templates/krb5.conf.template](https://github.com/RotherOSS/otobo/blob/rel-10_1/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`.

Providing an user specific `/etc/krb5.conf` file is still possible. This can be done by mounting a volume that overrides `/etc/krb5.conf` in the container. This can be achieved by setting `OTOBO_NGINX_KERBEROS_CONFIG` in the .env file and by activating the mount directive in `docker-compose/otobo-override-https-kerberos.yml`.

`/etc/krb5.keytab` is always installation specific and must therefore always be mounted from the host system.

### Kerberos SSO Installation Tutorial

[Kerberos Single Sign On in OTOBO Docker installation](#)

## 4.4.3 Choosing non-standard ports

Per default the ports 443 and 80 serve HTTPS and HTTP respectively. There can be cases where one or both of these ports are already used by other services. In these cases the default ports can be overridden by specifying `OTOBO_WEB_HTTP_PORT` and `OTOBO_WEB_HTTPS_PORT` in the .env file.

## 4.4.4 Skip startup of specific services

The current Docker compose setup start five, six when HTTPS is activated, services. But there are valid use cases where one or more of these services are not needed. The prime example is when the database should not run as a Docker service, but as an external database. Unfortunately there is no dedicated Docker compose option for skipping specific services. But the option `-scale` can be abused for this purpose. So for an installation with an external database the following command can be used:

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

Of course the same goal can also be achieved by editing the file `docker-compose/otobo-base.yml` and removing the relevant service definitions.

## 4.4.5 Prepare offline installation

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
```

Now you can run the following command to download all Docker images from a specific file, in my example I use the `otobo-base.yml`.

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



After that, the images (.docker) are located in the docker-compose folder and can be uploaded to the target system via e.g. [SCP](#).

On the offline target system, go to the folder where the docker images are stored. And enter the following command to import them one by one.

In the following example I import the mariadb image:

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

#### 4.4.6 Customizing OTOBO Docker Compose

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.

A common example would be to make the database container accessible from the outside via port 3306. For this you could create an extra docker compose file that looks like:

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

Now we have to tell docker-compose to include our new file. For this you have to add your YAML file to the COMPOSE\_FILE variable in the .env file, for example:

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

Now we can use docker-compose to recreate our container

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

With this procedure you can customize any service or volumes.

#### 4.4.7 Customizing the OTOBO Docker image

Many customizations can be done in the external volume otobo\_opt\_otobo which corresponds to the directory /opt/otobo in the Docker image. This works e.g. for local Perl modules which can be installed into /opt/otobo/local. Here is an example that installs the not very useful CPAN module Acme::123.

```
$ 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:~$
```

The nice thing of this approach is that the Docker image itself does not have to be modified.

Installing extra Debian packages is a little bit trickier. One approach is to create a custom Dockerfile and use the OTOBO image as the base image. Another approach is to create a modified image directly from a running container. This can be done with the command `docker commit`, <https://docs.docker.com/engine/reference/commandline/commit/>. A nice writeup of that process is available at <https://phoenixnap.com/kb/how-to-commit-changes-to-docker-image>.

But for the latter approach there are two hurdles to overcome. First, the image `otobo` runs per default as the user `otobo` with the UID 1000. The problem is that the user `otobo` is not allowed to install system packages. Thus, the first part of the solution is to pass the option `-user root` when running the image. However the second hurdle is that the default entrypoint script `/opt/otobo_install/entrypoint.sh` exits immediately when it is called as `root`. The reasoning behind that design decision is that running inadvertently as `root` should be discouraged. So, the second part of the solution is to specify a different entrypoint script that does not care who the caller is. This leaves us with following example commands, where we add fortune cookies to `otobo`:

Pull a tagged OTOBO image, if we don't have it yet, and check whether the image already provides fortune cookies:

```
$ docker run rotheross/otobo:rel-10_0_10 /usr/games/fortune
/opt/otobo_install/entrypoint.sh: line 57: /usr/games/fortune: No such file or
↳ directory
```

Add fortune cookies to a named container running the original OTOBO image. This is done in an interactive session as the user `root`:

```
$ docker run -it --user root --entrypoint /bin/bash --name otobo_orig rotheross/
↳ otobo:rel-10_0_10
root@50ac203409eb:/opt/otobo# apt update
root@50ac203409eb:/opt/otobo# apt install fortunes
root@50ac203409eb:/opt/otobo# exit
$ docker ps -a | head
```

Create an image from the stopped container and give it a name. Take into account that the default user and entrypoint script must be restored:

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

Finally we can doublecheck:

```
$ 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
```

The modified image can be specified in your `.env` file and then be used for fun and profit.

### 4.4.8 Building local images

---

**Note:** Building Docker images locally is usually only needed during development. Other use cases are when more current base images should be used for an installation or when extra functionality must be added to the images.

---

The Docker files needed for creating Docker images locally are part of the the git repository <https://github.com/RotherOSS/otobo>:

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

The script for the actual creation of the images is `bin/docker/build_docker_images.sh`.

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo.git
docker_admin> # checkout the wanted branch. e.g. git checkout rel-10_0_11
docker_admin> cd otobo
docker_admin> # modify the docker files if necessary
docker_admin> bin/docker/build_docker_images.sh
docker_admin> docker image ls
```

The locally built Docker images are tagged as `local-<OTOBO_VERSION>` using the version set up the file `RELEASE`.

After building the local images, one can return to the `docker-compose` directory. The local images are declared by setting `OTOBO_IMAGE_OTOBO`, `OTOBO_IMAGE_OTOBO_ELASTICSEARCH`, `OTOBO_IMAGE_OTOBO_NGINX` in `.env`.

## 4.4.9 Automatic Installation

Instead of going through <http://yourIPorFQDN/otobo/installer.pl>, one can take a short cut. This is useful for running the test suite on a fresh installation.

**Warning:** `docker-compose down -v` will remove all previous setup and data.

```
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 List of useful commands

#### Docker

- `docker system prune -a` system clean-up (removes all unused images, containers, volumes, networks)
- `docker version show version`
- `docker build --tag otobo --file=otobo.web.Dockerfile .` build an image
- `docker run --publish 80:5000 otobo` run the new image

- `docker run -it -v opt_otobo:/opt/otobo otobo bash` log into the new image
- `docker run -it -v opt_otobo:/opt/otobo --entrypoint bash otobo` try that in case `entrypoint.sh` is broken
- `docker ps` show running images
- `docker images` show available images
- `docker volume ls` list volumes
- `docker volume inspect otobo_opt_otobo` inspect a volume
- `docker volume inspect --format '{{ .Mountpoint }}' otobo_nginx_ssl` get volume mountpoint
- `docker volume rm tmp_volume` remove a volume
- `docker inspect <container>` inspect a container
- `docker save --output otobo.tar otobo:latest-10_0 && tar -tvf otobo.tar` list files in an image
- `docker exec -it nginx-server nginx -s reload` reload nginx

### Docker Compose

- `docker-compose config` check and show the configuration
- `docker-compose ps` show the running containers
- `docker-compose exec nginx nginx -s reload` reload nginx

## 4.5 Resources

Finally, here is a highly subjective collection of links.

### General info and tutorials

- [Perl Maven: Getting Started with Perl on Docker](#)
- [Dockerfile best practices](#)
- [Environment](#)

### Tips and hints

- [Newer version of Docker Compose on Ubuntu 18.04 LTS](#)
- [Newer version of Docker on Ubuntu 18.04 LTS](#)
- [Clean up unused images](#)
- [Docker Host IP](#)
- [Self signed certificate](#)

### Troubleshooting

- [Docker cache invalidation](#)
- [Using tcpdump](#)
- [Inspect failed builds](#)

---

## Migration from OTRS 6 or OTRS 7 / ((OTRS)) Community Edition to OTOBO version 10.1

---

Welcome and thank you for choosing OTOBO!

OTRS, ((OTRS)) Community Edition and OTOBO are very comprehensive and flexible in their application. Thus, every migration to OTOBO requires thorough preparation and possibly some rework, too.

Please take your time for the migration and follow these instructions step by step.

If you have any problem or question, please do not despair. Call our support line, write an email, or post your query in the OTOBO Community forum at <https://forum.otobo.org/>. We will find a way to help you!

---

**Note:** After the migration the data previously available in OTRS will be available in OTOBO 10. We do not modify any data of the OTRS installation during the migration.

---

### 5.1 Overview over the Supported Migration Szenarios

With the OTOBO Migration Interface it is possible to employ the following migration strategies:

1. The general migration strategy.

This is the regular way to perform a migration. Many different different combinations are supported:

**Change server:** Migrate and simultaneously move to a new application server.

**Separate application and web servers:** It's your choice whether you want to run application and database server on the same host or each on a dedicated host. This choice is regardless of the previous setup in OTRS / ((OTRS)) Community Edition.

**Different databases:** Migrate from any of the supported databases to any other supported database.

**Different operating system:** Switch from any supported operating system to any other supported operating system.

**Docker:** Migrate to a Docker-based installation of OTOBO 10.

2. A variant of the general strategy where the database migration is streamlined.

Use the ETL-like migration when the source database mustn't suffer from increased load or when access to the source database is a bottleneck. In the general strategy, the data is row by row first read from the otrs database and then inserted into the OTOBO database. In this variant, the complete otrs database tables are first exported, then transformed, and then imported into the otobo database.

3. Migration from an Oracle based OTRS 6 / OTRS 7 installation to an Oracle based OTOBO installation.

This is a special case that is not supported by the general migration strategy. This means that a variant of the streamlined strategy must be used.

**Warning:** All strategies work for both Docker-based and native installations. But for Docker-based installations some peculiarities have to be considered. These peculiarities are handled in the optional steps.

---

**Note:** It is also feasible to clone the OTRS data to the OTOBO database server before the actual migration. This can speed up the general migration strategy.

---

## 5.2 Migration Requirements

1. Basic requirement for a migration is that you already have an ((OTRS)) Community Edition or OTRS 6.0.\* / OTRS 7.0.\* running, and that you want to transfer both configuration and data to OTOBO.

**Warning:** Please consider carefully whether you really need the data and configuration. Experience shows that quite often a new start is the better option. This is because in many cases the previously used installation and configuration was rather suboptimal anyways. It might also make sense to only transfer the ticket data and to change the basic configuration to OTOBO Best Practice. We are happy to advise you, please get in touch at [hello@otobo.de](mailto:hello@otobo.de) or ask your question in the OTOBO Community forum at <https://forum.otobo.org/>.

2. You need a running OTOBO installation to start the migration from there!
3. This OTOBO installation must contain all OPM packages installed in your OTRS that you want to use in OTOBO, too.
4. If you are planning to migrate to another server, then the OTOBO webserver must be able to access the location where your ((OTRS)) Community Edition or OTRS 6.0.\* / OTRS 7.0.\* is installed. In most cases, this is the directory /opt/otrs on the server running OTRS. The read access can be effected via SSH or via file system mounts.
5. The otrs database must be accessible from the server running OTOBO. Readonly access must be granted for external hosts. If access is not possible, or when the speed of the migration should be optimised, then a dump of the database is sufficient.

---

**Note:** If SSH and database access between the servers is not possible, please migrate OTRS to OTOBO on the same server and only then move the new installation.

---

## 5.3 Step 1: Install the new OTOBO System

Please start with installing a new OTOBO system. Your old OTRS / ((OTRS)) Community Edition installation will be migrated to that new system. We strongly recommend to read the chapter [OTOBO Installation](#). For Docker-based installations we refer to the chapter [Installing using Docker and Docker Compose](#).

**Warning:** Under Apache, there are pitfalls with running two independent mod\_perl applications under on the same webserver. Therefore, it is advised to run OTRS and OTOBO on separate webserver. Alternatively remove the OTRS configuration from Apache before installing OTOBO. Use the command `a2query -s` and check the directories `/etc/apache2/sites-available` and `/etc/apache2/sites-enabled` for inspecting which configurations are currently available and which are enabled.

After finishing the installation please log in as `root@localhost`. Navigate to the OTOBO Admin Area `Admin -> Packages` and install all required OTOBO OPM packages.

**The following OPM packages and OTRS “Feature Addons” need NOT and should NOT be installed, as these feature**

- OTRSHideShowDynamicField
- RotherOSSHideShowDynamicField
- TicketForms
- RotherOSS-LongEscalationPerformanceBoost
- Znuny4OTRS-AdvancedDynamicFields
- Znuny4OTRS-AutoSelect
- Znuny4OTRS-EscalationSuspend
- OTRSEscalationSuspend
- OTRSDynamicFieldDatabase
- OTRSDynamicFieldWebService
- OTRSBruteForceAttackProtection
- Znuny4OTRS-ExternalURLJump
- Znuny4OTRS-QuickClose
- Znuny4OTRS-AutoCheckbox
- OTRSSystemConfigurationHistory
- Znuny4OTRS-PasswordPolicy

The following OTOBO packages have been integrated into OTOBO 11.0. This means that they should not be installed in the target system when the target system is OTOBO 11.

- ImportExport

## 5.4 Step 2: Deactivate SecureMode on OTOBO

After installing OTOBO, please log in again to the OTOBO Admin Area Admin -> System Configuration and deactivate the config option SecureMode.

---

**Note:** Do not forget to actually deploy the changed setting.

---

## 5.5 Step 3: Stop the OTOBO Daemon

This is necessary when the OTOBO Daemon is actually running. Stopping the Daemon is different between Docker-based and non-Docker-based installations.

In the non-Docker case execute the following commands as the user otobo:

```
# 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
```

When OTOBO is running in Docker, you just need to stop the service daemon:

```
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
```

---

**Note:** It is recommended to run a backup of the whole OTOBO system at this point. If something goes wrong during migration, you will then not have to repeat the entire installation process, but can instead import the backup for a new migration.

**See also:**

We advise you to read the OTOBO [Backup and Restore](#) chapter.

---

## 5.6 Optional Step: Mount /opt/otrs for Convenient Access

Often OTOBO should be running on a new server where /opt/otrs isn't available initially. In these cases the directory /opt/otrs on the OTRS server can be mounted into the file system of the OTOBO server. When a regular network mount is not possible, then using `sshfs` might be an option.



## 5.7 Optional Step: Install `sshpass` and `rsync` when `/opt/otrs` Should be Copied via `ssh`

This step is only necessary when you want to migrate OTRS from another server and when `/opt/otrs` from the remote server hasn't been mounted on the server running OTOBO.

The tools `sshpass` and `rsync` are needed so that `migration.pl` can copy files via `ssh`. For installing `sshpass`, please log in on the server as user `root` and execute one of the following commands:

```
$ # 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
```

The same thing must be done for `rsync` when it isn't available yet.

## 5.8 Step 4: Preparing the OTRS / ((OTRS)) Community Edition system

**Note:** Be sure to have a valid backup of your OTRS / ((OTRS)) Community Edition system, too. Yes, we do not touch any OTRS data during the migration, but at times a wrong entry is enough to cause trouble.

Now we are ready for the migration. First of all we need to make sure that no more tickets are processed and no users log on to OTRS:

Please log in to the OTRS Admin Area Admin -> System Maintenance and add a new system maintenance slot for a few hours. After that, delete all agent and user sessions (Admin -> Sessions) and log out.

### 5.8.1 Stop All Relevant Services and the OTRS Daemon

Please make sure there are no running services or cron jobs.

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

### 5.8.2 Clear the Caches and the Operational Data

The cached data and the operational data doesn't have to be migrated. The mail queue should at this point already be empty.

```

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 Optional Step for Docker: make required data available inside container

There are some specifics to be considered when your OTOBO installation is running under Docker. The most relevant: processes running in a Docker container generally cannot access directories outside the container. There is an exception though: directories mounted as volumes into the container can be accessed. Also, note that the MariaDB database running in `otobo_db_1` is not directly accessible from outside the container network.

**Note:** In the sample commands, we assume that the user `docker_admin` is used for interacting with Docker. The Docker admin may be either the `root` user of the Docker host or a dedicated user with the required permissions.

### 5.9.1 Copy `/opt/otrs` into the volume `otobo_opt_otobo`

In this section, we assume that the OTRS home directory `/opt/otrs` is available on the Docker host.

There are at least two viable possibilities:

1. copy `/opt/otrs` into the existing volume `otobo_opt_otobo`
2. mount `/opt/otrs` as an additional volume

Let's concentrate on option **a.** here.

First we need to find out where the volume `otobo_opt_otobo` is available on the Docker host.

```

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

For safe copying, we use `rsync`. Depending on your Docker setup, the command `rsync` might need to be run with `sudo`.

```

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
    
```

This copied directory will be available as `/opt/otobo/var/tmp/copied_otrs` within the container.

## 5.10 Step 5: Perform the Migration!

Please use the web migration tool at <http://localhost/otobo/migration.pl>. Be aware that you might have to replace “localhost” with your OTOBO hostname and you might have to add your non-standard port. The application then guides you through the migration process.

**Warning:** Sometimes, a warning is shown that the deactivation of **SecureMode** has not been detected. Please restart the webserver in this case. This forces the webserver to read in the current configuration.

```
# 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
```

**Note:** If OTOBO runs inside a Docker container, keep the default settings localhost for the OTRS server and /opt/otobo/var/tmp/copied\_otrs for the OTRS home directory. This is the path of the data that was copied in the optional step.

**Note:** The default values for OTRS database user and password are taken from Kernel/Config.pm in the OTRS home directory. Change the proposed settings if you are using a dedicated database user for the migration. Also change the settings when you work with a database that was copied into the otobo\_db\_1 Docker container.

**Note:** In the Docker case, a database running on the Docker host won't be reachable via 127.0.0.1 from within the Docker container. This means that the setting 127.0.0.1 won't be valid for the input field OTRS Server. In that case, enter one of the alternative IP-addresses reported by the command `hostname --all-ip-addresses` for OTRS Server.

**Note:** When migrating to a new application server, or to a Docker-based installation, quite often the database cannot be accessed from the target installation. This is usually due to the fact that the otobo database user can only connect from the host the database runs on. In order to allow access anyways it is recommended to create a dedicated database user for the migration. E.g. `CREATE USER 'otrs_migration'@'%' IDENTIFIED BY 'otrs_migration';` and `GRANT SELECT, SHOW VIEW ON otrs.* TO 'otrs_migration'@'%';`. This user can be dropped again after the migration: `DROP USER 'otrs_migration'@'%';`

Custom settings in Kernel/Config.pm are carried over from the old OTRS installation to the new OTOBO installation. When you have custom settings, then please take a look at the migrated file /opt/otobo/Kernel/Config.pm. You might want to adapt custom pathes or LDAP settings. In the best case you might find that some custom setting are longer needed.

When the migration is complete, please take your time and test the entire system. Once you have decided that the migration was successful and that you want to use OTOBO from now on, start the

OTOBO Daemon:

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

In the Docker case:

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

## 5.11 Step 6: After Successful Migration!

1. Uninstall `sshpas` if you do not need it anymore.
2. Drop the databases and database users dedicated to the migration if you created any.
3. Have fun with OTOBO!

## 5.12 Known Migration Problems

### 5.12.1 1. Login after migration not possible

During our migration tests, the browser used for the migration sometimes had problems. After restarting the browser, this problem usually was solved. With Safari it was sometimes necessary to manually delete the old OTRS session.

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

This can happen when the setting `ScriptAlias` has a non-standard value. The migration simply substitutes `otrs` for `otobo`. This might lead to the effect that the CSS and JavaScript can no longer be retrieved in OTOBO. When that happens, please check the settings in `Kernel/Config.pm` and revert them to sane values.

### 5.12.3 3. Migration stops due to MySQL errors

On systems that experienced problems with an upgrade in the past, the migration process may stop due to MySQL errors in the tables `ticket` and `ticket_history`. Usually these errors are NULL values in the source table that are no longer allowed in the target table. These conflicts have to be manually resolved before you can resume the migration.

As of OTOBO 10.0.12 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 these cases the not so helpful message “System was unable to complete data transfer.” is shown by `migration.pl`. The Apache logfile, and the OTOBO logfile, show a more meaningful message: “Message: ERROR: permission denied to set parameter “`session_replication_role`”, SQL: ‘set ses-

sion\_replication\_role to replica;”’. In order to give the database user **otobo** the needed superuser privileges, run the following statement as the PostgreSQL admin: `ALTER USER otopo WITH SUPERUSER;`. Then retry running <http://localhost/otobo/migration.pl>. After the migration, return to the normal state by running `ALTER USER otopo WITH NOSUPERUSER.`

It is not clear yet, whether the extended privileges have to be granted in every setup.

**See also:**

The discussion 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

The system configuration is migrated after the database tables were migrated. In this context, migration means merging the default settings of OTOBO with the system configuration of the source OTRS system. Inconsistencies can arise in this step. An real life example is the setting `Ticket::Frontend::AgentTicketQuickClose###State`. This setting is new in OTOBO 10 and the default value is the state `closed successful`. But this setting is invalid when the state `closed successful` has been dropped or renamed in the source system. This inconsistency is detected as an error in the migration step **Migrate configuration settings**. Actually, the merged system configuration is stored in the database, but additional validity checks are performed during deployment.

The problem must be alleviated manually by using OTOBO console commands.

- List the inconsistencies with the command `bin/otobo.Console.pl Admin::Config::ListInvalid`
- Interactively fix the invalid values with `bin/otobo.Console.pl Admin::Config::FixInvalid`
- Deploy the collected changes from migration.pl, including the deactivated **SecureMode** with `bin/otobo.Console.pl Maint::Config::Rebuild`

After these manual steps you should be able to run migration.pl again. The migration will continue with the step where the error occurred.

## 5.13 Step 7: Manual Migration Tasks and Changes

### 5.13.1 1. Password policy rules

With OTOBO 10 a new default password policy for agent and customer users is in effect, if local authentication is used. The password policy rules can be changed in the system configuration (`PreferencesGroups###Password` and `CustomerPersonalPreference###Password`).

Password Policy Rule	Default
<code>PasswordMinSize</code>	8
<code>PasswordMin2Lower2UpperCharacters</code>	Yes
<code>PasswordNeedDigit</code>	Yes
<code>PasswordHistory</code>	10
<code>PasswordTTL</code>	30 days
<code>PasswordWarnBeforeExpiry</code>	5 days
<code>PasswordChangeAfterFirstLogin</code>	Yes

## 5.13.2 2. Under Docker: Manually migrate cron jobs

In a non-Docker installation of OTOBO, there is at least one cron job which checks the health of the Daemon. Under Docker, this cron job no longer exists. Furthermore, there is no cron daemon running in any of the Docker containers. This means that you have to look for an individual solution for OTRS systems with customer-specific cron jobs (e. g. backing up the database).

## 5.14 Special topics

### 5.14.1 Migration from Oracle to Oracle

For migration to Oracle the ETL-like strategy must be employed. This is because Oracle provides no easy way to temporarily turn off foreign key checks.

On the OTOBO host a Oracle client and the Perl module `DBD::Oracle` must be installed.

---

**Note:** When using the Oracle instant client, then the optional SDK is also needed for installing `DBD::Oracle`.

---

There are many ways of cloning a schema. In the sample commands we use `expdp` and `impdp` which use Data Pump under the hood.

---

**Note:** The connect strings shown in this documentation refer to the case when both source and target database run in a Docker container. See also <https://github.com/bschmalhofer/otobo-ideas/blob/master/oracle.md>.

---

1. Clear out otobo

Stop the webserver for otobo, so that the DB connection for otobo is closed.

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

2. Export the complete OTRS schema.

```
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 the OTRS schema, renaming the schema to '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. Adapt the cloned schema otobo

```
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. Start the web server for otobo again

6. Proceed with step 5, that is with running migration.pl.

---

**Note:** If migrating to OTOBO version greater or equal 10.1 the script `/opt/otobo/scripts/DBUpdate-to-10.1.pl` has to be executed, to create the tables `stats_report` & `data_storage`, which were newly added in version 10.1.

---

## 5.14.2 Optional Step: Streamlined migration of the database (only for experts and special scenarios)

In the general migration strategy, all data in the database tables is copied row by row from the OTRS database into the OTOBO database. Exporting the data from the OTRS database and importing it into the OTOBO database might save time and is more stable in some circumstances.

---

**Note:** This variant works for both Docker-based and native installations.

---



---

**Note:** These instructions assume that OTRS is using MySQL as its backend.

---

First of all, we need a dump of the needed OTRS database tables. Then we need to perform a couple of transformations:

- convert the character set to utf8mb4
- rename a couple of tables
- shorten some table columns

After the transformation we can overwrite the tables in the OTOBO schema with the transformed data from OTRS. Effectively we need not a single dump file, but several SQL scripts.

When `mysqldump` is installed and a connection to the OTRS database is possible, you can create the database dump directly on the Docker host. This case is supported by the script `bin/backup.pl`.

**Warning:** This requires that an OTOBO installation is available on the Docker host.

```
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"
```

**Note:** Alternatively, the database can be dumped on another server and then be transferred to the Docker host afterwards. An easy way to do this is to copy /opt/otobo to the server running OTRS and perform the same command as above.

The script bin/backup.pl generates four SQL scripts in a dump directory, e.g. in 2021-04-13\_12-13-04 In order to execute the SQL scripts, we need to run the command `mysql`.

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-based installation:

Run the command `mysql` within the Docker container `db` for importing the database dump files. Note that the password for the database root is now the password that has been set up in the file `.env` on the Docker host.

```
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
```

For a quick check whether the import worked, you can run the following commands.

```
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'
```

or when running under Docker

```
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'
```

The database is now migrated. This means that during the next step we can skip the database migration. Watch out for the relevant checkbox.



---

**Note:** It is highly recommended to perform a test update on a separate testing machine first.

---

**Note:** On Debian systems you may need to manually install some perl packages before upgrading from 10.0 to 10.1

```
apt-get install -y libarchive-zip-perl libtimedate-perl libdatettime-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 Step 1: Stop All Relevant Services and the OTOBO Daemon

Please make sure there are no more running services or cron jobs that try to access OTOBO. This will depend on your service configuration.

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

Stop OTOBO cron jobs and the daemon (in this order):

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

## 6.2 Step 2: Backup Files and Database

Create a backup of the hole /opt/otobo directory and the database.

### 6.2.1 Example for a standard installation with Ubuntu and 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
```

Please check if all files are valid. Now we have a backup with all required data.

**Warning:** Don't proceed without a complete backup of your system. You can use also the backup-restore script for this.

## 6.3 Step 3: Install the New Release

Download the latest otobo release from <https://ftp.otobo.org/pub/otobo/>. and unpack the source archive (for example, using tar) into the directory /root/otobo-update:

```
root> cd /root/otobo-update # Change into
↳the update directory
root> wget https://ftp.otobo.org/pub/otobo/otobo-latest-10.1.tar.gz # Download he
↳latest OTOBO 10.1 release
root> tar -xzf otobo-latest-10.1.tar.gz # Unzip OTOBO
root> cp -r otobo-10.1.x/* /opt/otobo # Copy the
↳new otobo directory to /opt/otobo
```

### 6.3.1 Restore Old Configuration Files

We need only copy the file Kernel/Config.pm in OTOBO 10.

```
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 Restore Article Data

If you configured OTOBO to store article data in the file system you have to restore the `article` folder to `/opt/otobo/var/` or the folder specified in the system configuration.

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

### 6.3.3 Restore Already Installed Default Statistics

If you have additional packages with default statistics you have to restore the stats XML files with the suffix `*.installed` to `/opt/otobo/var/stats`.

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

### 6.3.4 Set File Permissions

Please execute the following command to set the file and directory permissions for OTOBO. It will try to detect the correct user and group settings needed for your setup.

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

### 6.3.5 Check Apache configuration files

Newer versions of OTOBO may need you to adjust the apache configuration. From version 10.1 and onwards we moved from CGI to PSGI. Take a look at `scripts/apache2-httpd-vhost-443.include.conf` to see what settings needs to be adjusted/added.

## 6.4 Step 4: Check for new needed perl modules

OTOBO needs new cpan packages for some version jumps. Please check if new packages are needed and install them if necessary.

**Note:** On Debian systems you may need to manually install some packages:

```
apt-get install -y libarchive-zip-perl libtimedate-perl libdatettime-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
```

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

## 6.5 Step 5: Update Installed Packages and reconfigure config

You can use the command below to update all installed packages. This works for all packages that are available from online repositories. You can update other packages later via the package manager (this requires a running 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 10.0 to 10.1)

```
root> su - otobo
otobo> /opt/otobo/scripts/DBUpdate-to-10.1.pl
```

## 6.7 Step 7: Start your Services

Start OTOBO cron jobs and the daemon (in this order):

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

Now the services can be started. This will depend on your service configuration, here is an example:

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

Now you can log into your system.

---

## Updating a Docker-based Installation of OTOBO

---

For running OTOBO under Docker we need the OTOBO software itself and an environment in which OTOBO can run. The OTOBO Docker image provides the environment and a copy of the OTOBO software. The software itself is installed in the volume `otobo_opt_otobo`. A named volume is used because run time data, e.g. configuration files and installed packages, is stored in the same directory tree.

When updating to a new version of OTOBO several things have to happen.

- The Docker Compose files have to be updated.
- The Docker Compose config file `.env` has to be checked.
- The new Docker image has to be fetched.
- The volume `otobo_opt_otobo` must be updated.
- Some maintenance tasks must be executed.

---

**Note:** In the sample commands below, the version **10.x.y**, corresponding to the tag **10\_x\_y**, is used as the example version. Please substitute it with the real version, e.g. **10.0.7**.

---

**Warning:** These instructions apply only to OTOBO 10.0.6 or later.

### 7.1 Updating the Docker Compose files

The OTOBO Docker Compose files can change between releases. Therefore it must be made sure that the correct setup is used.

---

**Note:** See <https://hub.docker.com/r/rotheross/otobo/tags> for the available releases.

---

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

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

# Update OTOBO docker-compose repository to version 10.x.y.
docker-admin> git checkout rel-10_x_y
```

## 7.2 Checking the Docker Compose .env file

The file `.env` controls the OTOBO Docker container. Within that file, the variables `OTOBO_IMAGE_OTOBO`, `OTOBO_IMAGE_OTOBO_ELASTICSEARCH`, and `OTOBO_IMAGE_OTOBO_NGINX` declare which images are used. The latest images are used when these variables are not set. If you want to use a specific version, then please set these variables accordingly.

## 7.3 Fetch the new Docker images

Docker compose can be used for fetching the wanted images from <https://hub.docker.com/r/rotheross/otobo/>.

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

# fetch the new images, either 'latest-10_0' or 'latest-10_1' or the specific version,
↳ declared in .env
docker_admin> docker-compose pull
```

## 7.4 Update OTOBO

In this step the volume `otobo_opt_otobo` is updated and the following OTOBO console commands are performed:

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

```
# 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

# 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

**# For minor or major release upgrades, you also have to run the upgrade script (for
↳example to upgrade from 10.0 to 10.1)**
root> docker exec -it otobo_web_1 perl scripts/DBUpdate-to-10.1.pl
```

**Note:** The above listed commands can be automated. For that purpose the script `scripts/update.sh` will be made available in OTOBO 10.0.8. This script runs the commands, starting with the **docker-compose pull** command.

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

**# For minor or major release upgrades, you also have to run the upgrade script (for
↳example to upgrade from 10.0 to 10.1)**
docker_admin> docker exec -it otobo_web_1 perl scripts/DBUpdate-to-10.1.pl
```





---

## Backup and Restore

---

OTOBO has built in scripts for backup and restore. Execute the scripts with the option `-h` for more information.

### 8.1 Backup

**Note:** To create a new backup, write permission for the destination directory is needed for the user `otobo`.

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

The output of the script:

```
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 Restore

**Note:** To restore the database make sure that the database `otobo` exists and contains no tables.

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

The output of the script:

```
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 Considerations for running OTOBO under Docker

The same scripts can be used with OTOBO running under Docker. However some Docker specific limitation must be considered.

First we need to make sure that the backup files are not created in the file system that is internal to the container. Because in that case all data would be lost when the container is stopped. Therefore

the backup directory must be in a volume. For now we only consider the most simple case, where the backup dir is a local dir on the Docker host. The location of the backup dir in the container can be arbitrarily chosen. In this example we choose the local dir `otobo_backup` as the location on the host, and `/otobo_backup` as the location in the container.

First we need to create the volume.

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

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

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

For creating the backup we need a running database and the volumes `otobo_opt_otobo` and `otobo_backup`. This means that the webserver and the Daemon may, but don't have to, be stopped.

```
# create a backup
docker_admin> docker run -it --rm --volume otopo_opt_otobo:/opt/otobo --volume otopo_
↪backup:/otobo_backup --network otopo_default rotheross/otobo:latest-10_0 scripts/
↪backup.pl -d /otobo_backup

# check the backup file
docker_admin> tree otopo_backup
```

For restoring the backup we also need to specify which backup should be restored. The placeholder `<TIMESTAMP>` is something like `2020-09-07_09-38`.

```
# create a backup
docker_admin> docker run -it --rm --volume otopo_opt_otobo:/opt/otobo --volume otopo_
↪backup:/otobo_backup --network otopo_default rotheross/otobo:latest-10_0 scripts/
↪restore.pl -d /opt/otobo -b /otobo_backup/<TIMESTAMP>
```



---

## Backup and Restore using Docker

---

Please read to the chapter [Backup and Restore](#) for basic information about the backup and restore scripts.

### 9.1 Considerations for running OTOBO under Docker

The standard scripts `backup.pl` and `restore.pl` can also be used with OTOBO running under Docker. However some Docker specific limitations have to be considered.

First, we need to make sure that the backup files are not created in the file system that is internal to a Docker container. Because in that case the created files would be lost when the container is stopped. This means that the backup directory must be located within a volume. For this manual we only consider the most simple case, where the backup directory is a local directory on the Docker host. The location of the backup dir in the container can be arbitrarily chosen. In this example we choose the local dir `otobo_backup` as the location on the host and `/otobo_backup` as the location in the container.

Secondly, commands in the Docker container usually run as the user `otobo` with the user id 1000 and the group id 1000. It must be made sure, that this user can write in the backup directory.

First we need to create the volume.

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

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

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

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

For creating the backup we need a running database and the volumes `otobo_opt_otobo` and `otobo_backup`. This means that the webserver and the OTOBO daemon may, but don't have to, be stopped.

```
# create a backup
docker_admin>docker run -it --rm --volume otopo_opt_otobo:/opt/otobo --volume otopo_
↳backup:/otobo_backup --network otopo_default rotheross/otobo:latest-10_0 scripts/
↳backup.pl --extra-dump-options="--single-transaction" -d /otobo_backup

# check the backup file
docker_admin>tree otopo_backup

.. note::

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

---

**Note:** To restore the database make sure that the database `otobo` exists and contains no tables.

---

To drop an existing `otobo` database and create a new one you can use the following commands. First, you have to connect to the MySQL CLI of the db container.

As soon as you are connected to the MySQL server, you can drop and recreate the `otobo` database.

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

For restoring the backup we also need to specify which backup should be restored. The placeholder `<TIMESTAMP>` is something like `2020-09-07_09-38`.

```
# restore a backup
docker_admin>docker run -it --rm --volume otopo_opt_otobo:/opt/otobo --volume otopo_
↳backup:/otobo_backup --network otopo_default rotheross/otobo:latest-10_0 scripts/
↳restore.pl -d /opt/otobo -b /otobo_backup/<TIMESTAMP>
```

---

## Kerberos Single Sign On in OTOBO Docker installation

---

Please read to the chapter [Installing using Docker and Docker Compose](#) for basic information about installing and configure OTOBO. This tutorial assumes that OTOBO has been installed and configured using Docker.

---

**Note:** In the following, we will refer from AD (Active Directory), of course the Kerberos configuration is also possible under LDAP.

---

### 10.1 Generate Active Directory User

Please create a new Active Directory User with the following settings and save the marked settings:

---

**Note:** Please use as Username only this syntax: HTTP/fqdn.from.your.otobo.de. fqdn.from.your.otobo.de needs to be a A-Record DNS entry, not a CNAME! In the next step, it is also possible to use other URLs for OTOBO, they must then point as CNAME to our A-record defined above.

The username part “HTTP/” should be written in capital letters, as Kerberos expects it that way.

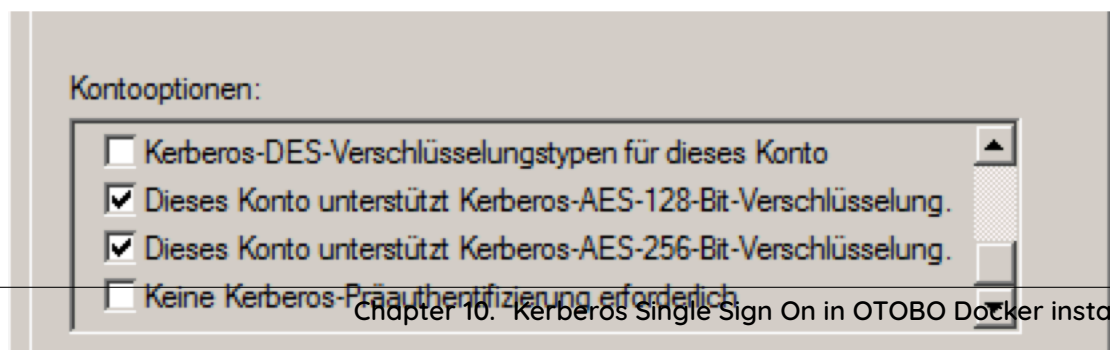
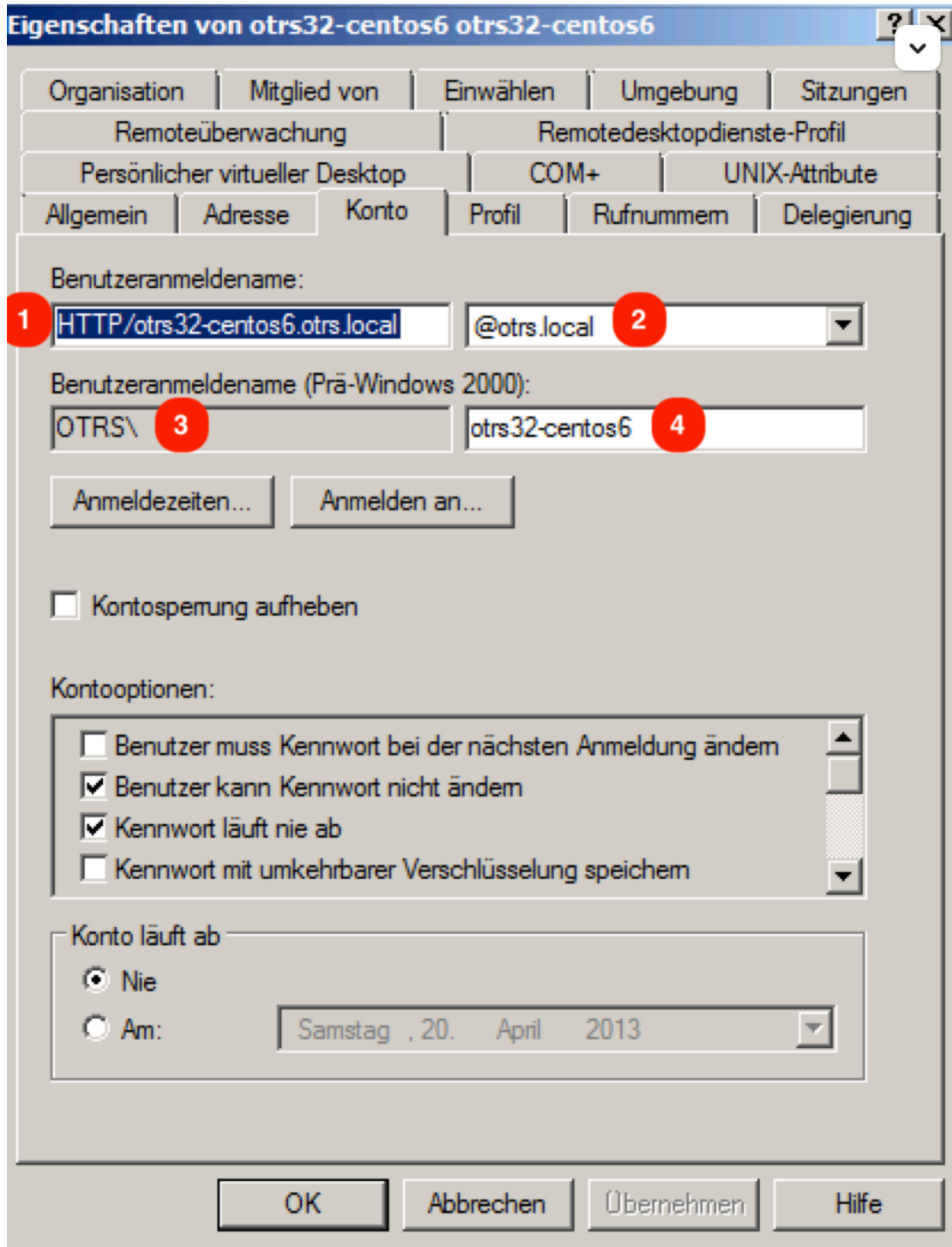
The password doesn't work properly with some special characters (e.g. '&').

You have to create a separate AD-user. You can not use the one that you already use for your LDAP/AD sync.

---

### 10.2 Generate Active Directory Keytab file

In the next step, we connect to a domain controller of the Active Directory and open a console (cmd) there with administrator privileges. Now we use the tool 'ktpass.exe' to generate the needed keytab file:





```
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 (Username prä Win 2000) -> -> Picture Number 3++Picture Number
- -pass = Password from user otrs32-centos6 (Active Directory User)
- -out = c:/krb5.keytab

**Note:** Please write the domain (@OTRS.LOCAL) always in capital letters. The password must not contain some special characters.

In the next step please move the krb5.keytab file to the 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 Create a new volume for your custom nginx configuration

```
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-10_1
↪(achtung: Versionsnummer)
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 Create new OTOBO .env file

First of all we need to move the old file /opt/otobo-docker/.env to .env.tmp and create a new .env file including the kerberos settings.

```
# 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
```

Now copy your existing configuration options to the new .env file (at least OTOBO\_DB\_ROOT\_PASSWORD, OTOBO\_NGINX\_SSL\_CERTIFICATE, OTOBO\_NGINX\_SSL\_CERTIFICATE\_KEY) and insert the following Kerberos settings:

```
# Kerberos keytab OTOBO_NGINX_KERBEROS_KEYTAB=/opt/otobo-docker/nginx-conf/krb5.keytab
# Kerberos config (Important, please comment out this option like here!) # In default configura-
tion the krb5.conf file is generated automatically # OTOBO_NGINX_KERBEROS_CONFIG=/opt/otobo-
docker/nginx-conf/krb5.conf

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

# Kerberos REALM OTOBO_NGINX_KERBEROS_REALM=ROTHER-OSS.COM -> OTRS.LOCAL # -> Pic-
ture Number 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 Start OTOBO

After the initial Kerberos configuration we start OTOBO again:

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

## 10.6 Tell OTOBO to use the Kerberos-Authentication

In case you have configured AD-Authentication, de-activate it (e.g. by commenting out the respective lines from your Kernel/Config.pm). The authentication will not take place via LDAP anymore.

To use Kerberos-Authentication take the Kerberos-lines from Kernel/Config/Defaults.pm and put it into you Kernel/Config.pm E.g. these lines could work:

```
$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 Configure Browser to understand Kerberos SSO

For SSO to work, the browser must be configured accordingly.

**Chrome, Edge, Internet Explorer, etc.**

Add page under local or trusted pages and activate 'Integrated Windows Authentication' (Internet Options).

**Firefox**

Enter "about:config" in the firefox address line

and change the following settings:

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

## 10.8 Debugging and Problems

If the Kerberos SSO does not work, please check first if the NGINX container is started:

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

In the next step please check the NGINX logs for more information:

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

If NGINX is running, please login into the NGINX Container and check all needed files:

```
# 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 debugging

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

Now you are able to debug the Kerberos settings. Examples:

```
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
```

In case you stumble upon the issue that apparently the authentication works but the agent is not yet in the database, then your sync (if implemented) might not work. An error 52e (First bind failed) indicates that something is wrong with your Search User. This happens if you use the same user for the AD sync and as a SSO user. Please use separate AD users for that. In order to not have to create a new keytab and having to repeat the steps mentioned above, it could be easier to create a new user to use in your AD sync (probably in your Kernel/Config.pm).

In case SSO is not working properly, make sure: \* the user for which it is not working is in Active Directory \* the system has to be in the domain \* it is properly stated as a trusted page (see 'Configure Browser to understand Kerberos SSO')

---

## Adapt customer interface with corporate identity

---

In OTOBO it is very easy to adapt the customer area to your own corporate identity. Follow this tutorial step by step and OTOBO will shine in your own design in a short time.

---

**Note:** Currently, it is not so easy to adapt the agent area to one's own CI. Changes in the OTOBO .css files would be necessary here. One exception is the logo on the agent login page and the agent header. The logos can be easily exchanged by copying the logos to the server and then adjusting the options `AgentLoginLogo` and `AgentLogo` under `Admin -> System Configuration`.

---

### 11.1 Change colors in Customer Area

To change the colors for the OTOBO customer interface, please go to `Admin -> System Configuration` and change the following settings:

- `CustomerColorDefinitions`
- To change the colours on the Customer Dashboard, please go to `Admin -> System Configuration` and search for `CustomerDashboard`. In the search result you will find all the options you need with colour definitions.

### 11.2 Change Logos and Pictures

In the first step please copy your Logos and Pictures to the OTOBO Server. Please use an SCP client (WinSCP) for this purpose. Often you do not have the permissions to copy the logos to the right place. In this case, it is best to use the folder `/tmp/`.

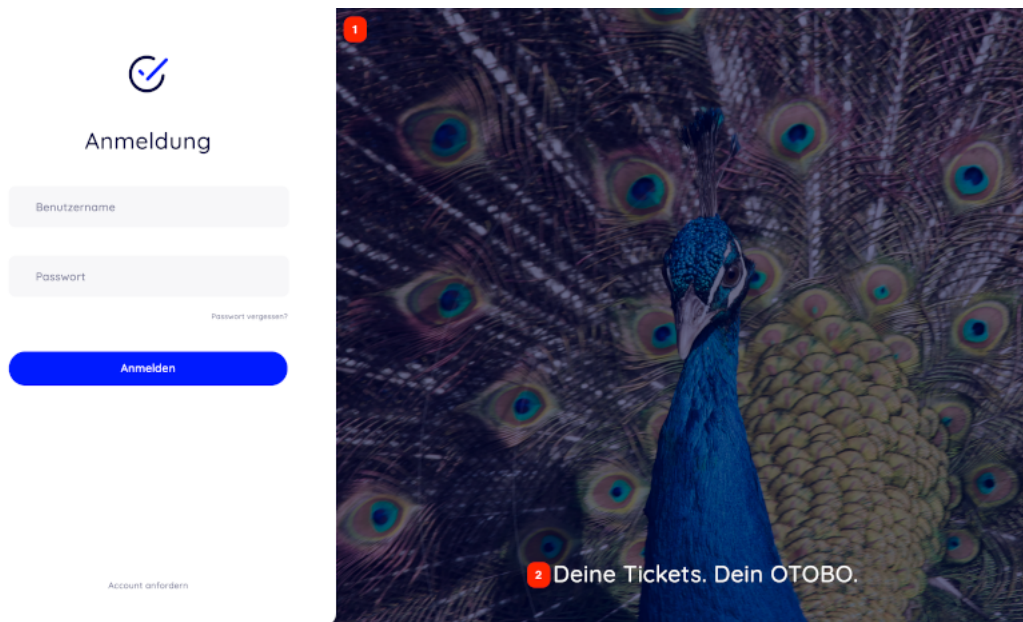
In the next step copy the Logo into the OTOBO Home directory:

```
**# Using OTOBO Docker Installation**
otobo_admin> docker cp /tmp/Logos.png otopo_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/
```

Now change inside the OTOBO Agent Interface to Admin -> System Configuration and change the following settings:

### 11.2.1 Change Customer Login Pictures and Text



- 1 and 2 - System Configuration Option **CustomerLogin::Settings**

#### Remove Opacity and Watermark

At the moment it is not possible to remove the overlay and watermark used in the image on the right by system configuration.

To remove the opacity, please adjust the option **#oooLoginBG > .oooBG** in the file `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;
}
```

To remove the watermark, please remove the following line inside the file:

`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>
```

---

**Note:** 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/10.1/en/content/how-to-publish-otobo-extensions.html>

---

## 11.2.2 Change Customer Dashboard tiles and options

To change the colours on the Customer Dashboard, please go to Admin -> System Configuration and search for **CustomerDashboard**.

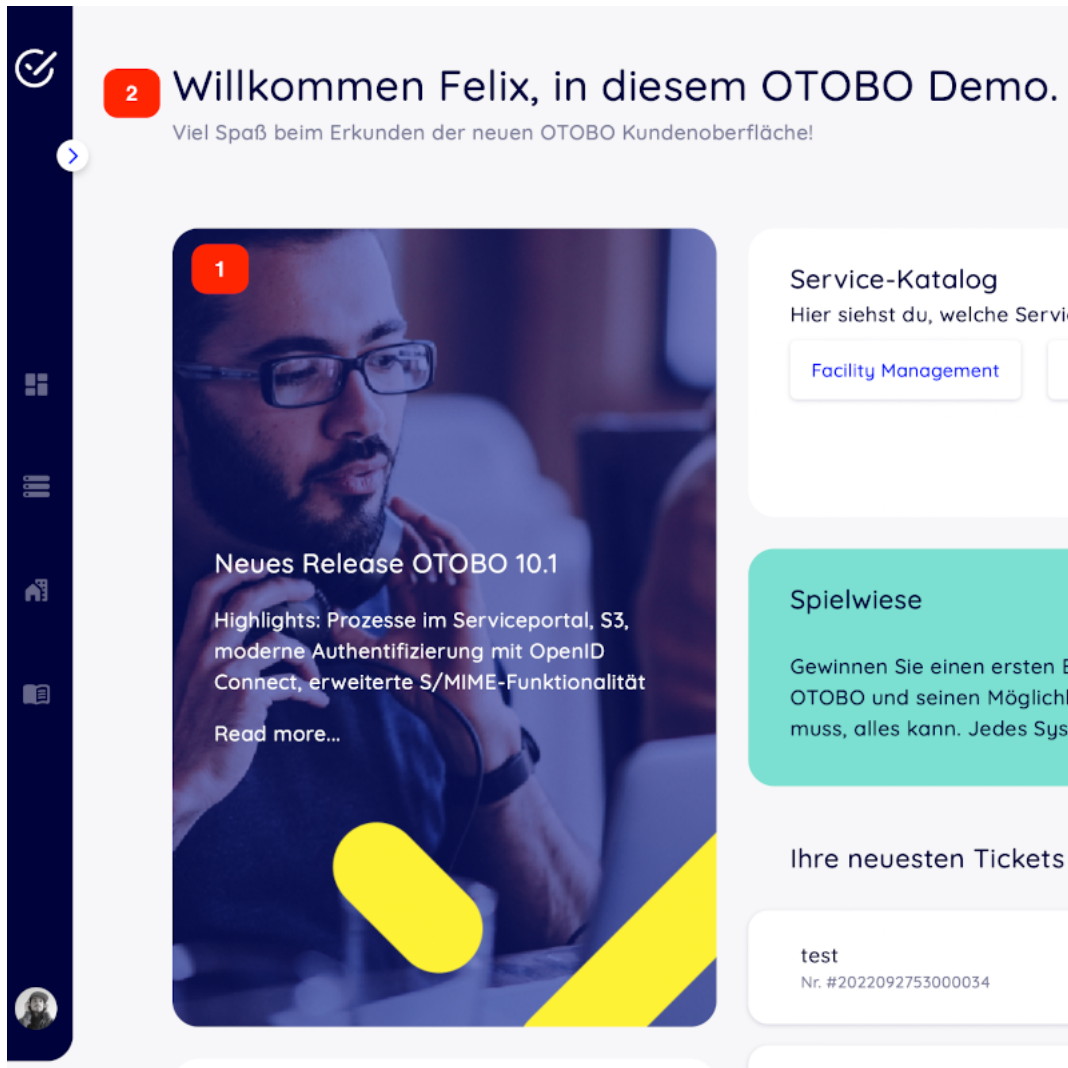
In the search result you will find all the options you need with colour definitions.

- 1 - To change the picture, link and text please use the System Configuration Option **CustomerDashboard::Tiles###FeaturedLink-01**
- 2 - To change the toplevel text please use the System Configuration Option **CustomerDashboard::Configuration::Text**

---

**Note:** Please disable the config options from not needed tiles.

---





---

## Installing Perl Modules from CPAN

---

When there are special requirements then the need for additional Perl modules may arise. Fortunately, Perl has an excellent package repository that can satisfy almost all needs. That repository is called CPAN and is available at <https://metacpan.org/>.

It is recommended to use the command line client `cpanm` for installing modules. `cpanm` is often already installed on your system. Please see <https://metacpan.org/pod/App::cpanminus> for what to do when it isn't already available.

Alternatively, many Perl modules are also available as packages for your operating system. These packages can be installed with your system's regular package manager.

Per default `cpanm` installs modules into a systemwide location. In this case modules must be installed as the root user. For example, the command

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

results in:

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

### 12.1 Docker-based installations

Special care must be taken when OTOBO runs under Docker. In this case an installation into a systemwide location would initially work as well. However, due to how Docker works, these installed modules would be lost when the container is restarted. Therefore the modules must be installed into a location that does survive a restart. The directory `/opt/otobo/local` within the volume **otobo\_opt\_otobo** can be used for that. Modules that are installed in `/opt/otobo/local` will be picked up by Perl because the environment variables `PERL5LIB` and `PATH` are preset accordingly.

The installed Perl modules will also be available after an upgrade of OTOBO. There is the general rule that files added to `/opt/otobo` won't be removed by an upgrade.

For installing Perl modules in a specific location we need to modify our install command. Specifically, we need to add the option `--local-lib`. Here is a sample session in the 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
```

This is a list of performance enhancing techniques for your OTOBO installation. The topics include configuration, coding, memory use, and more.

## 13.1 Ticket Index Module

Ticket index module can be set via system configuration setting `Ticket::IndexModule`. There are two back end modules building the index for the ticket queue view:

**Kernel::System::Ticket::IndexAccelerator::RuntimeDB** This is the default option, which will generate each queue view on the fly from the ticket table. You will not have performance trouble until you have about 60,000 open tickets in your system.

**Kernel::System::Ticket::IndexAccelerator::StaticDB** The most powerful module, should be used when you have above 80,000 open tickets. It uses an extra `ticket_index` table, which will be populated with keywords based on ticket data. Use the following command for generating an initial index after switching back ends:

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

## 13.2 Ticket Search Index

OTOBO uses a special search index to perform full-text searches across fields in articles from different communication channels.

To create an initial index, use this command:

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

**Note:** Actual article indexing happens via an OTOBO daemon job in the background. While articles which were just added in the system are marked for indexing immediately, it could happen their index is available only after a few minutes.

There are some options available for fine-tuning the search index:

**Ticket::SearchIndex::IndexArchivedTickets** Defines if archived tickets will be included in the search index (disabled by default). This is advisable to keep the index small on large systems with archived tickets. If this is enabled, archived tickets will be found by full-text searches.

**Ticket::SearchIndex::Attribute** Basic full-text index settings.

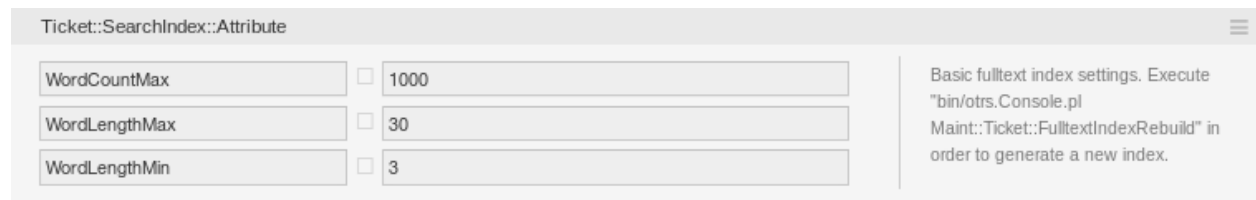


Fig. 13.1: Ticket::SearchIndex::Attribute Setting

**Note:** Run the following command in order to generate a new index:

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

**WordCountMax** Defines the maximum number of words which will be processed to build up the index. For example only the first 1000 words of an article body are stored in the article search index.

**WordLengthMin and WordLengthMax** Used as word length boundaries. Only words with a length between these two values are stored in the article search index.

**Ticket::SearchIndex::Filters** Filters based on regular expressions exclude parts of the original text from the full-text index.

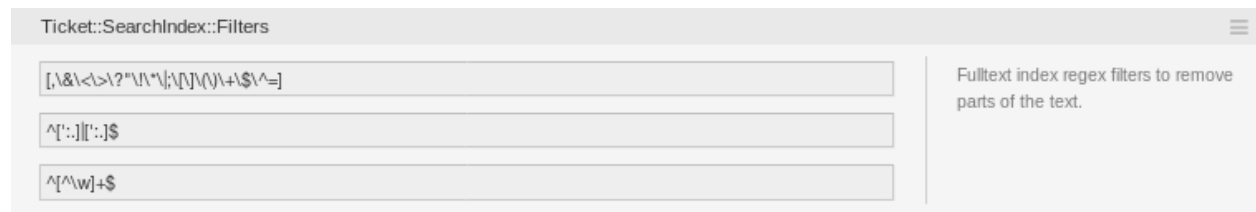


Fig. 13.2: Ticket::SearchIndex::Filters Setting

There are three default filters defined:

- The first filter strips out special chars like: , & < > ? " ! \* | ; [ ] ( ) + \$ ^ =
- The second filter strips out words which begin or end with one of following chars: ' : .
- The third filter strips out words which do not contain a word character: a-z, A-Z, 0-9, \_

**Ticket::SearchIndex::StopWords** English stop words for full-text index. These words will be removed from the search index.



Fig. 13.3: Ticket::SearchIndex::StopWords###en Setting

There are so-called stop-words defined for some languages. These stop-words will be skipped while creating the search index.

**See also:**

If your language is not in the system configuration settings or you want to add more words, you can add them to this setting:

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

## 13.3 Document Search

OTOBO uses Elasticsearch for its document search functionality. For a good introduction into the concepts, installation and usage of Elasticsearch, please follow the [Getting Started guide](#).

### 13.3.1 Heap Size

Elasticsearch is written in Java and therefore runs in a Java Virtual Machine (JVM) on any cluster node. Such a JVM uses a part of the memory, called heap, which size can be configured in configuration file `jvm.options`.

The heap minimum and maximum configurations are by default set to a value of 1 GB and can be modified with the following options:

- `Xms1g`: minimum heap size.
- `Xmx1g`: maximum heap size.

If the `Xms` has a lower value than `Xmx`, the JVM will resize the used heap anytime the current limit is exceeded, until the value of `Xmx` is reached. Such a resizing causes the service to pause until it is finished, which may decrease the speed and reactivity of the search or indexing actions. Therefore it is highly recommended to set those configurations to an equal value.

**Warning:** If the maximum heap size is exceeded, the related cluster node stops working and might even shutdown the service.

The higher the heap maximum value is set, the more memory can be used by Elasticsearch, which also increases the possible pauses for garbage collection, done by the JVM. Therefore it is recommended to set a value for `Xmx`, that is not higher than 50% of the physical memory.

For more information and good rules of thumb about the heap size, please follow [the official documentation](#).

### 13.3.2 Disk Allocation

While running the service, Elasticsearch inspects the available disk space. Based on the result, it decides whether to allocate new shards to a cluster node. In some cases it even relocates shards away from a node. This behavior is determined by the current disk capacity. It can be configured by settings in the configuration file `elasticsearch.yml`. Here are some relevant configuration settings. They come with good default values, but might be important in trouble shooting.

**`cluster.routing.allocation.disk.watermark.low`** Default value of 85%. When this limit is exceeded, Elasticsearch will no longer allocate more shards to the related cluster node. The operation of that node is not influenced and data can still be indexed and searched.

**`cluster.routing.allocation.disk.watermark.high`** Default value of 90%. When this limit is exceeded, Elasticsearch will try to relocate existing shards to other nodes that have enough space available.

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

Default value of 95%. When this limit is exceeded, Elasticsearch will update the configuration of all indices, that have at least one shard allocated to the related cluster node, to read-only index blocks. Specifically, they are flagged with `index.blocks.read_only_allow_delete`.

After that update, it is no longer possible to index new data to such indices. The indexes are restricted to searches and to delete actions only.

---

**Note:** If the flood stage was exceeded and certain indices are configured to read-only mode, such configuration will not automatically be changed by Elasticsearch. If the related disks contain enough free space again due to manual actions, it is needed to change the configuration back to normal mode manually.

---

For more information about disk watermarks and disk-based shard allocation, please follow [the official documentation](#).

## 13.4 Article Storage

There are two different back end modules for the article storage of phone, email and internal articles. The used article storage can be configured in the setting `Ticket::Article::Backend::MIMEBase::ArticleStorage`.

**`Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB`** This default module will store attachments in the database. It also works with multiple front end servers, but requires much storage space in the database.

---

**Note:** Don't use this with large setups.

---

---

**Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS** Use this module to store attachments on the local file system. It is fast, but if you have multiple front end servers, you must make sure the file system is shared between the servers. Place it on an NFS share or preferably a SAN or similar solution.

---

**Note:** Recommended for large setups.

---

You can switch from one back end to the other on the fly. You can switch the back end in the system configuration, and then run this command line utility to put the articles from the database onto the file system or the other way around:

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

You can use the `--target` option to specify the target back end.

---

**Note:** The entire process can take considerable time to run, depending on the number of articles you have and the available CPU power and/or network capacity.

---

If you want to keep old attachments in the database, you can activate the system configuration option `Ticket::Article::Backend::MIMEBase::CheckAllStorageBackends` to make sure OTOBO will still find them.

## 13.5 Archiving Tickets

As OTOBO can be used as an audit-proof system, deleting closed tickets may not be a good idea. Therefore there is a feature that allows you to archive tickets.

Tickets that match certain criteria can be marked as archived. These tickets are not accessed if you do a regular ticket search or run a generic agent job. The system itself does not have to deal with a huge amount of tickets any longer as only the latest tickets are taken into consideration when using OTOBO. This can result in a huge performance gain on large systems.

To use the archive feature:

1. Activate the `Ticket::ArchiveSystem` setting in the system configuration.
2. Define a generic agent job:
  - Click on the Add Job button in the Generic Agent screen.
  - Job Settings: provide a name for the archiving job.
  - Automatic Execution: select proper options to schedule this job.
  - Select Tickets: it might be a good idea to only archive those tickets in a closed state that have been closed a few months before.
  - Update/Add Ticket Attributes: set the field Archive selected tickets to archive tickets.
  - Save the job at the end of the page.
  - Click on the Run this task link in the overview table to see the affected tickets.
  - Click on the Run Job button.

---

**Note:** Up to 5000 tickets can be modified by running this job manually.

---

When you search for tickets, the system default is to search tickets which are not archived.

To search for archived tickets:

1. Open the ticket search screen.
2. Set Archive search to Unarchived tickets or All tickets.
3. Perform the search.

## 13.6 Caching

A fast cache module is a great help in terms of performance. We recommend to use a Redis Cache server or to create a ramdisk.

### 13.6.1 Install a Redis Cache Server

1. Install Redis Server

First of all you need to install the newest Redis Server. The easiest way is to [setup Redis](#) on the same host as OTOBO and binding it to its default port.

2. Install Perl module Redis or Redis::Fast

You can choose which Redis module to use: Redis or Redis::Fast (which is compatible with Redis but **~2x faster**). Please use `otobo.CheckModules.pl --list` to choose the right package for you:

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

3. Configure OTOBO for Redis

Please use the OTOBO SysConfig (Admin -> System Configuration) to configure OTOBO properly:

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 RamDisk Caching

OTOBO caches a lot of temporary data in `/opt/otobo/var/tmp`. Please make sure that this uses a high performance file system and storage. If you have enough RAM, you can also try to put this directory on a ramdisk like this:

```
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
```



---

**Note:** Add persistent mount point in `/etc/fstab`.

---

**Warning:** This will be a non-permanent storage that will be lost on server reboot. All your sessions (if you store them in the file system) and your cache data will be lost.

## 13.7 Clustering

For very high loads, it can be required to operate OTOBO on a cluster of multiple front end servers. This is a complex task with many pitfalls. We strongly advise to get in touch with our experts before trying to implement this on your own.



---

## Documentation History

---

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

Published by: Rother OSS GmbH, (<https://otobo.de>), Oberwaling 31, 94339 Leiblfing, Germany. Authors: OTRS AG (original version), Rother OSS GmbH (<https://otobo.de>)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found on the [GNU website](#).