



# OTOB Installation Guide

Release 10.1

Rother OSS GmbH

5 | 13, 2024



<b>1</b>	<b>Getting Started</b>	<b>3</b>
1.1	Introduction	3
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Prerequisites	7
2.2	Installation	7
<b>3</b>	<b>OTOBOnote</b>	<b>9</b>
3.1	Installing SELinux on Linux	9
3.2	Step 1: Installing OTOBOnote	10
3.3	Step 2: Installing Perl	10
3.4	Step 3: Installing OTOBOnote	11
3.5	Step 4: Installing MySQL	11
3.6	Step 5: Installing Apache	11
3.6.1	Installing SSL on Apache	12
3.6.2	Disabling SSL on Apache	12
3.7	Step 6: Installing Redis	13
3.8	Step 7: Installing Elasticsearch	13
3.9	Step 8: Installing Elasticsearch	14
3.9.1	Installing Elasticsearch on Ubuntu 18.04 LTS	14
3.9.2	Installing Elasticsearch on Linux	14
3.9.3	Elasticsearch	15
3.9.4	Elasticsearch	15
3.10	Step 9: Basic System Configuration	15
3.11	Step 10: First Login	15
3.12	Step 11: Start the OTOBOnote Daemon	15
3.13	Step 12: Cron jobs for the OTOBOnote user	15
3.14	Step 13: Setup Bash Auto-Completion (optional)	16
3.15	Step 14: Further Information	16
<b>4</b>	<b>Using Docker and Docker Compose</b>	<b>17</b>
4.1	Introduction	17
4.2	Installation	18
4.2.1	1. Clone the otopo-docker repo	18
4.2.2	2. Create an initial .env file	18
4.2.3	3. Configure the password for the database admin user	19
4.2.4	4. Set up a volume with SSL configuration for the nginx webproxy (optional)	19

4.2.5	安装 Docker/Docker Compose	19
4.2.6	6. Install and start OTOBO	20
4.3	安装	20
4.3.1	Docker	20
4.3.2	Docker	20
4.3.3	Docker	20
4.4	配置	22
4.4.1	Nginx	22
4.4.2	Single Sign On Using the Kerberos Support in Nginx	23
4.4.3	配置	24
4.4.4	Skip startup of specific services	24
4.4.5	Prepare offline installation	24
4.4.6	Customizing OTOBO Docker Compose	24
4.4.7	Customizing the OTOBO Docker image	25
4.4.8	配置	26
4.4.9	配置	27
4.4.10	配置	27
4.5	安装	28
<b>5</b>	<b>Migration from OTRS 6 or OTRS 7 / ((OTRS)) Community Edition to OTOBO version 10.1</b>	<b>29</b>
5.1	Overview over the Supported Migration Szenarios	29
5.2	配置	30
5.3	安装 OTOBO	31
5.4	Step 2: Deactivate <code>SecureMode</code> on OTOBO	32
5.5	Step 3: Stop the OTOBO Daemon	32
5.6	Optional Step: Mount <code>/opt/otrs</code> for Convenient Access	32
5.7	Optional Step: Install <code>sshpas</code> and <code>rsync</code> when <code>/opt/otrs</code> Should be Copied via <code>ssh</code>	32
5.8	Step 4: Preparing the OTRS / ((OTRS)) Community Edition system	33
5.8.1	配置 OTRS	33
5.8.2	Clear the Caches and the Operational Data	33
5.9	Optional Step for Docker: make required data available inside container	34
5.9.1	Docker <code>*/opt/otrs</code> <code>otobo_opt_otobo*</code>	34
5.10	Step 5: Perform the Migration!	34
5.11	Step 6: After Successful Migration!	36
5.12	Known Migration Problems	36
5.12.1	1. Login after migration not possible	36
5.12.2	2. Final page of the migration has a strange layout due to missing CSS files	36
5.12.3	3. Migration stops due to MySQL errors	36
5.12.4	4. Errors in Step 5 when migrating to PostgreSQL	36
5.12.5	5. Problems with the Deployment the Merged System Configuration	37
5.13	安装 OTOBO	37
5.13.1	1. Password policy rules	37
5.13.2	2. Under Docker: Manually migrate cron jobs	37
5.14	Special topics	38
5.14.1	Migration from Oracle to Oracle	38
5.14.2	Optional Step: Streamlined migration of the database (only for experts and special scenarios)	39
<b>6</b>	<b>安装</b>	<b>41</b>
6.1	安装 OTOBO	41
6.2	安装	42
6.2.1	安装 Ubuntu MySQL	42
6.3	安装	42
6.3.1	安装	42

6.3.2	XXXXXXXX	42
6.3.3	XXXXXXXXXXXXXXXX	43
6.3.4	XXXXXXXX	43
6.3.5	Check Apache configuration files	43
6.4	Step 4: Check for new needed perl modules	43
6.5	Step 5: Update Installed Packages and reconfigure config	44
6.6	Step 6: Only for minor or major release upgrades (for example to upgrade from 10.0 to 10.1)	44
6.7	Step 7: Start your Services	44
<b>7</b>	<b>Updating a Docker-based Installation of OTOBO</b>	<b>45</b>
7.1	Updating the Docker Compose files	45
7.2	Checking the Docker Compose .env file	46
7.3	!Docker!!!	46
7.4	Update OTOBO	46
<b>8</b>	<b>XXXXXX</b>	<b>49</b>
8.1	XX	49
8.2	XX	50
8.3	!Docker!!!OTOBO!!!	50
<b>9</b>	<b>XXXXXX</b>	<b>53</b>
9.1	!Docker!!!OTOBO!!!	53
<b>10</b>	<b>Kerberos Single Sign On in OTOBO Docker installation</b>	<b>55</b>
10.1	Generate Active Directory User	55
10.2	Generate Active Directory Keytab file	55
10.3	Create a new volume for your custom nginx configuration	57
10.4	Create new OTOBO .env file	57
10.5	Start OTOBO	58
10.6	Tell OTOBO to use the Kerberos-Authentication	58
10.7	Configure Browser to understand Kerberos SSO	58
10.8	Debugging and Problems	59
10.8.1	Kerberos debugging	59
<b>11</b>	<b>Adapt customer interface with corporate identity</b>	<b>61</b>
11.1	Change colors in Customer Area	61
11.2	Change Logos and Pictures	61
11.2.1	Change Customer Login Pictures and Text	62
11.2.2	Change Customer Dashboard tiles and options	63
<b>12</b>	<b>Installing Perl Modules from CPAN</b>	<b>65</b>
12.1	Docker-based installations	65
<b>13</b>	<b>XXXXX</b>	<b>67</b>
13.1	XXXXXXXX	67
13.2	XXXXXXXX	67
13.3	XXXXX	69
13.3.1	XXX	69
13.3.2	XXXX	69
13.4	XXXX	70
13.5	XXXX	70
13.6	XX	71
13.6.1	XXXX Redis CacheXXX	71
13.6.2	XXXXXXXX	71

13.7	□□	.....	72
14	□□□□		73



OTRS AG (<https://otrs.com>), Zimmersmühlenweg 11, 61440 Oberursel, Germany.

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

OTRS GNU 1.3 GNU

Rother OSS GNU 1.3 GNU

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

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

OTOBO

"username" > command-to-execute " "root"

**Warning:** "username"

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
```



## CHAPTER 2



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
```

```
root@otobo:~$
```

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

```
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 Requirements

Minimum system requirements for OTOBO installation. OTOBO requires the following system requirements:

- CPU
- 4 GB RAM
- 10 GB disk space

Recommended system requirements for OTOBO installation:

- 3 GHz Xeon or equivalent CPU
- 8 GB RAM or 16 GB
- 40 GB disk space

**Note:** OTOBO requires the following system requirements for OTOBO installation.

## 2.2 Prerequisites

### Perl

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

### Web Server

- Apache HTTP Server Version 2.4

### Database

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

### Cache

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

### Browser

- Safari
- Chrome
- Internet Explorer 11
- Edge

- Mozilla Firefox
- `enableJavaScript`

OTOBO

OTOBO

**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 SELinux

**Note:** SELinux

SELinux `sestatus` `getenforce`

`sestatus` SELinux SELinux “” SELinux

RHEL/CentOS/Fedora SELinux

1. `/etc/selinux/config` `SELINUX=disabled`

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

```
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. `getenforce` `getenforce`

```
root> getenforce
Disabled
```

### 3.2 1 OTOBO

`https://ftp.otobo.org/pub/otobo/OTOBO.tar.gz` `/root/otobo-update`

```
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 2 Perl

CPAN

**Note:** On Debian systems you may need to manually install some perl 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
↳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:** OTOBO\*Perl“version“ RHELPerl



“perl-core“

CPAN Linux

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

**Note:** OTOBO CheckModules.pl

### 3.4 3 OTOBO

OTOBO

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

OTOBO

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

### 3.5 4

OTOBO "\$OTOBO\_HOME/Kernel/Config.pm.dist" ".dist"

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

### 3.6 5 Apache

Apache2 mod\_perl Linux Apache

```
# 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:** `sudo a2enmod ssl` is required to enable SSL support in Apache.

---

For more information, see the `httpd.conf` file in the `/etc/apache2/` directory.

### 3.6.1 Enabling SSL support in Apache

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 Enabling SSL support in Apache

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
```

For more information, see the `httpd.conf` file in the `/etc/apache2/` directory.



**Note:** MySQL MySQL “[mysqld]” MySQL “/etc/my.cnf” /etc/mysql/my.cnf /etc/mysql/mysql.conf.d/mysqld.cnf

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

MySQL 8.0

```
query_cache_size = 32M
```

MySQL MySQL “[mysqld]” MySQL “/etc/my.cnf” /etc/mysql/my.cnf /etc/mysql/mysql.conf.d/mysqld.cnf

```
max_allowed_packet = 64M
```

“mysqldtuner” GitHub “https://github.com/major/MySQLTuner-perl” Debian Ubuntu

```
root> apt-get install mysqldtuner
```

```
root> mysqldtuner --user root --pass NewRootPassword
```

### 3.9 8 Elasticsearch

OTOBO Elasticsearch Elasticsearch OTOBO

#### 3.9.1 Ubuntu 18.04 LTS Elasticsearch

JDK

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

Elasticsearch

```
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 Linux Elasticsearch

https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html

### 3.9.3 Elasticsearch

OTOBO 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

Elasticsearch

OTOBO jvm "/etc/elasticsearch/jvm.options" JVM

```
-Xms4g
-Xmx4g
```

4-10GB

**Note:** <https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html>

Elasticsearch

```
root> systemctl restart elasticsearch
```

## 3.10 Step 9: Basic System Configuration

<http://localhost/otobo/installer.pl> "localhost"

## 3.11 Step 10: First Login

<http://localhost/otobo/index.pl> "root@localhost"

## 3.12 Step 11: Start the OTOBO Daemon

OTOBO OTOBO cron OTOBO OTOBO

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

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

"/opt/otobo/var/cron/\*.dist" cron OTOBO Daemon ".dist"



---

## Getting 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 Prerequisites

Prerequisites for Ubuntu 18.04:

- Docker 19.03.08
- DockerCompose 1.25.0
- Git 2.25.1

**Note:** For Ubuntu 18.04 see <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>.

Prerequisites for Ubuntu 20.04:

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

Git Docker

## 4.2

Docker root Docker Doc

### 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:** `*/opt/otobo-docker*`

---

```
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.

---

OTOBO HTTPS44380HTTPSOTOBOWebHTTP  
 HTTPSnginx  
 HTTPS





## 4.2.6 6. Install and start OTOBO

`http://yourIPorFQDN/otobo/installer.pl` OTOBO

---

**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.

---

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

### 4.3.1 Docker

`otobo_web_1` 5000 OTOBO Web  
`otobo_daemon_1` OTOBO OTOBO  
`otobo_db_1` 3306 MariaDB  
`otobo_elastic_1` 9200 9300 Elasticsearch  
`otobo_redis_1` Redis  
`otobo_nginx_1` nginx HTTPS

### 4.3.2 Docker

Docker

`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` "redis"  
`otobo_nginx_ssl` TLS

### 4.3.3 Docker

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

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

**Elasticsearch**

Elasticsearch <https://www.elastic.co/guide/zh-CN/elasticsearch/reference/7.8/docker.html#docker-prod-prerequisites>

**OTOBO\_Elasticsearch\_ES\_JAVA\_OPTS** \* OTOBO\_Elasticsearch\_ES\_JAVA\_OPTS = -Xms512m -Xmx512m \*4G

**Webserver**

**OTOBO\_WEB\_HTTP\_PORT** HTTP80HTTPSHTTPHTTPS

**Nginx webproxy settings**

HTTPS

**OTOBO\_WEB\_HTTP\_PORT** HTTP80HTTPS

**OTOBO\_WEB\_HTTPS\_PORT** HTTPS443

**OTOBO\_NGINX\_SSL\_CERTIFICATE** Nginx WebproxySSL\* OTOBO\_NGINX\_SSL\_CERTIFICATE = /etc/nginx/ssl/acme.crt \*

**OTOBO\_NGINX\_SSL\_CERTIFICATE\_KEY** Nginx WebproxySSL\* 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**

docker-compose

**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 then one instance of OTOBO on the same server.

**COMPOSE\_PATH\_SEPARATOR** COMPOSE\_FILE

**COMPOSE\_FILE** \* docker-compose / otobo-base.yml docker-compose / otobo-override-http.yml 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

### 4.4.1 Nginx

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:**  OTOBO 10.0.4 

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 otopo_nginx_1
docker_admin> docker volume create otopo_nginx_custom_config

# find out where the new volume is located on the Docker host
docker_admin> otopo_nginx_custom_config_mp=$(docker volume inspect --format '{{ .
↳Mountpoint }}' otopo_nginx_custom_config)
docker_admin> echo $otopo_nginx_custom_config_mp # just a sanity check
docker_admin> ls $otopo_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:      listen 10.0.30.4:80
                listen 10.0.40.80:80
                listen 10.0.30.4:80
                listen 10.0.40.80:80
```

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
```

Restart Docker Compose

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

Restart Docker Compose

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

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.



```
$ 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

---

**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

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                               `

```
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

\*\* 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

Finally, here is a highly subjective collection of links.

### General info and tutorials

- Perl Maven: Getting Started with Perl on Docker
- Dockerfile
- 

### Tips and hints

- Ubuntu 18.04 LTS Docker Compose
- Ubuntu 18.04 LTS Docker
- Clean up unused images
- Docker IP
- [‘https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-18-04’](https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-18-04)

### Troubleshooting

- [‘https://stackoverflow.com/questions/34814669/when-does-docker-image-cache-invalidation-occur’](https://stackoverflow.com/questions/34814669/when-does-docker-image-cache-invalidation-occur)
- Using tcpdump
-



**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 Prerequisites

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. `OTOBODATA` directory must exist on the OTOBO server.
3. `OTOBODATA` directory must exist on the OTOBO server.
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:** `ssh root@localhost:~# yum install OTRS OTOBO`

### 5.3 1 OTOBO

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](#). For Docker-based installations we refer to the chapter [Docker/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.

`OPM OTRS "OTRS" OTOBO`

- 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
↪ 0
```

---

**Note:** `OTOBObackup-restore`

**See also:**

`OTOBObackup-restore`

---

## 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
```

```
rsync *
```

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

**Note:** OTRS / OTRS Community Edition

OTRS

Admin -> System Maintenance  
Admin-> Sessions

### 5.8.1 OTRS

cron

```
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:** `docker exec -it otopo_opt_otobo_1 root`

---

### 5.9.1 Docker `*/opt/otrs` `otobo_opt_otobo *`

`docker exec -it otopo_opt_otobo_1`

`root`

1. `docker exec -it otopo_opt_otobo_1`
2. `*/opt/otrs *`

`docker exec -it otopo_opt_otobo_1`

`docker exec -it otopo_opt_otobo_1`

```
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
```

`docker exec -it otopo_opt_otobo_1`

```
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
```

`docker exec -it otopo_opt_otobo_1`

## 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.





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

## 5.11 Step 6: After Successful Migration!

1. `sshpass`
2. Drop the databases and database users dedicated to the migration if you created any.
3. `OTOBO`

## 5.12 Known Migration Problems

### 5.12.1 1. Login after migration not possible

`Safari:OTRS`

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

`ScriptAliasotrsotoboOTBOCSSJavaScriptKernel  
/ Config.pm`

### 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 session\_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 otobo WITH SUPERUSER;`. Then retry running <http://localhost/otobo/migration.pl>. After the migration, return to the normal state by running `ALTER USER otobo 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 7

### 5.13.1 1. Password policy rules

OTOBO 10 " PreferencesGroups ### Password" CustomerPersonalPreference #### Password"

" PasswordMinSize "	8
" PasswordMin2Lower2UpperCharacters "	
" PasswordNeedDigit "	
" PasswordHistory "	10
" PasswordTTL "	30
" PasswordWarnBeforeExpiry "	5
" PasswordChangeAfterFirstLogin "	

### 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



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.



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

## 6.2 2

“/opt/otobo”

### 6.2.1 Ubuntu 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
```

**Warning:** “backup-restore”

## 6.3 3

https://ftp.otobo.org/pub/otobo/OTOBOTar“/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

OTOBOT10“Kernel/Config.pm”

```
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

OTOBOT“article” “/ opt / otobo / var /”



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

### 6.3.3 复制文件

复制文件到 /opt/otobo/var/stats 目录

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

### 6.3.4 设置权限

设置 OTOBO 目录的权限

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

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



---

## 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.
- `docker compose pull`
- The new Docker image has to be fetched.
- `docker volume inspect otopo_opt_otobo`
- 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:** `docker compose pull`

### 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 Docker

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
```



□□□□

OTOBOScript "otobo" help

## 8.1 □□

**Note:** OTOBO "otobo" help

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

□□□□□□□□

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
```









Please read to the chapter ■■■■■ for basic information about the backup and restore scripts.

## 9.1 ■ Docker■■■OTOB■■■

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.

■■■■■■■■■■

```
# 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/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 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:** `otobo` `otobo`

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'@'%';
```

`<TIMESTAMP>` `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 [Docker 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.

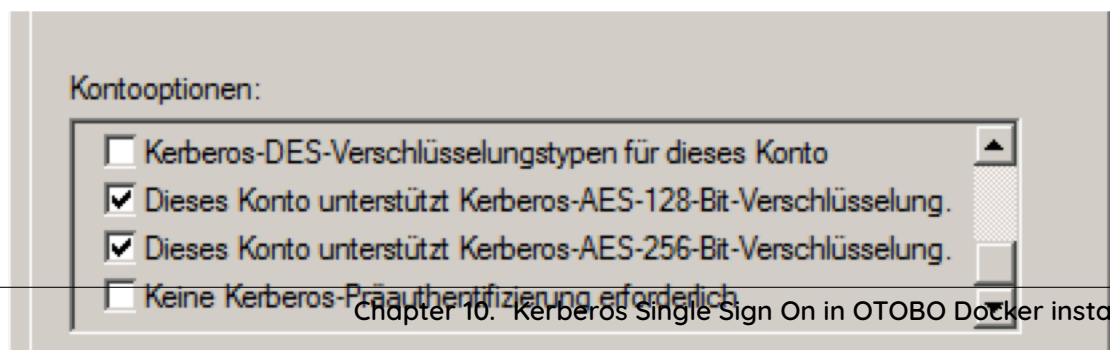
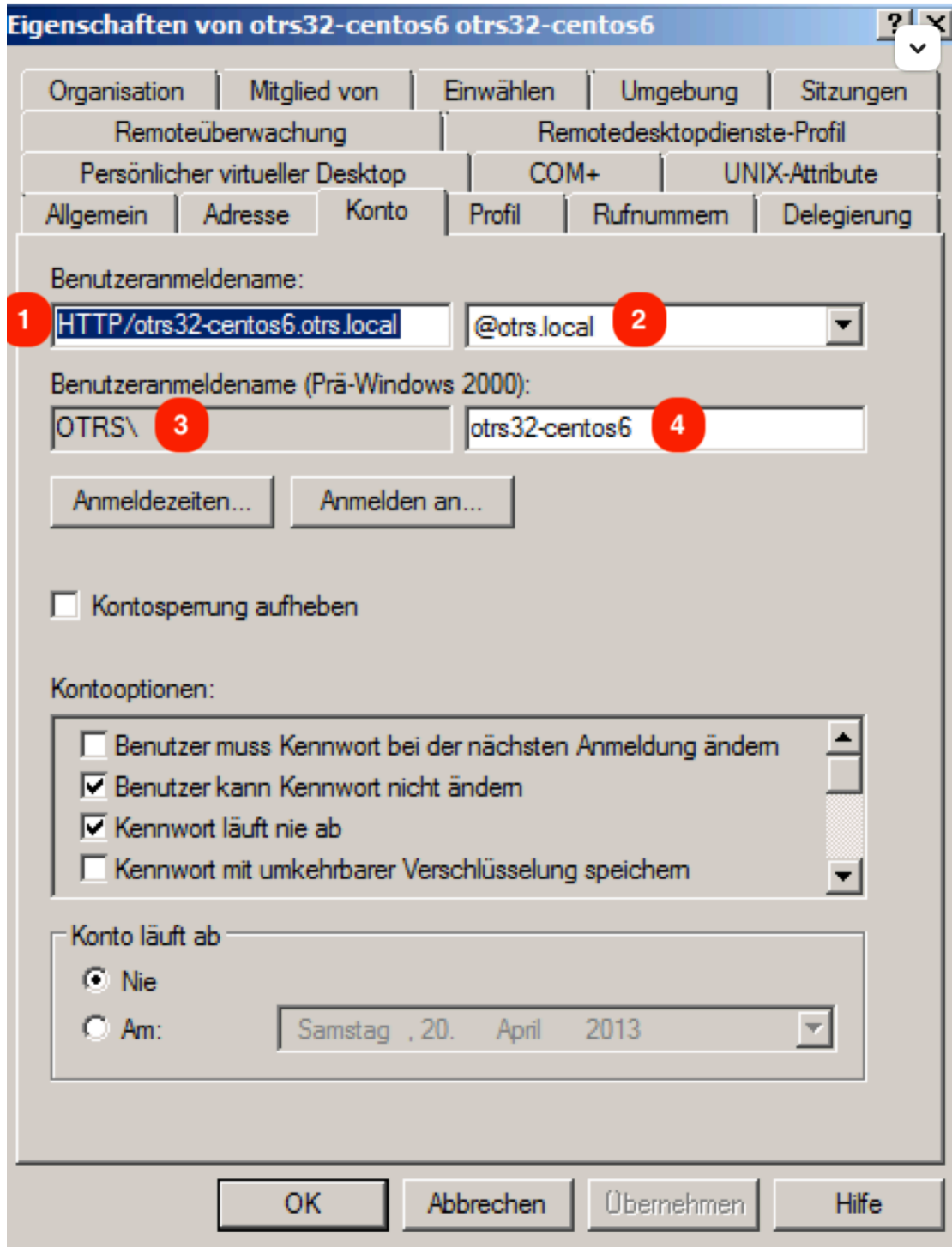
The password doesn't work properly with some special characters (e.g. '&').

You have to create a seperate 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/`.

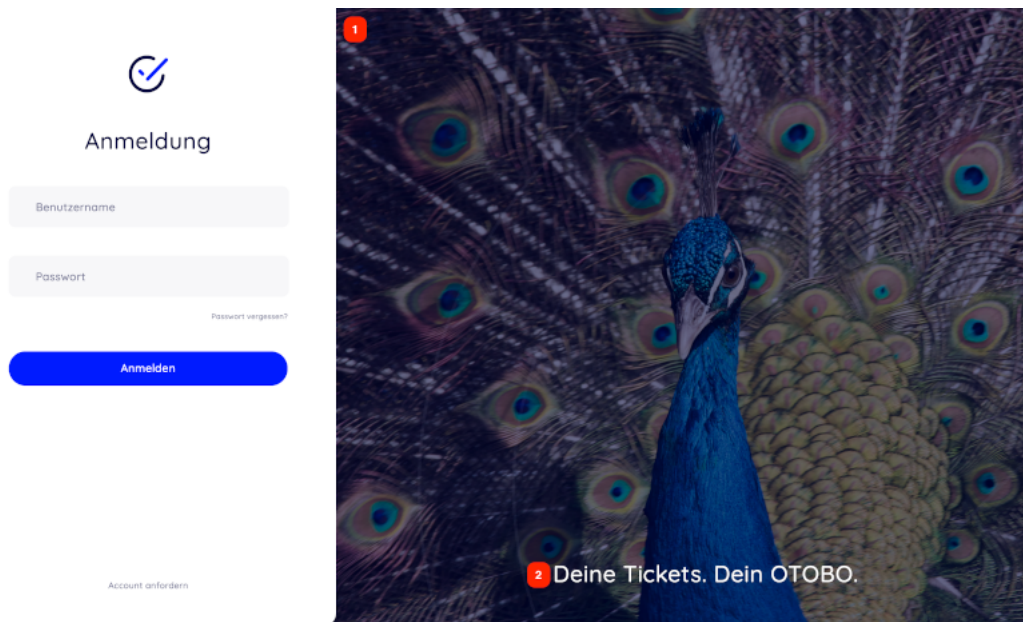
In the next step copy the Logo into the OTOBO Home directory:

```
**# 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**.

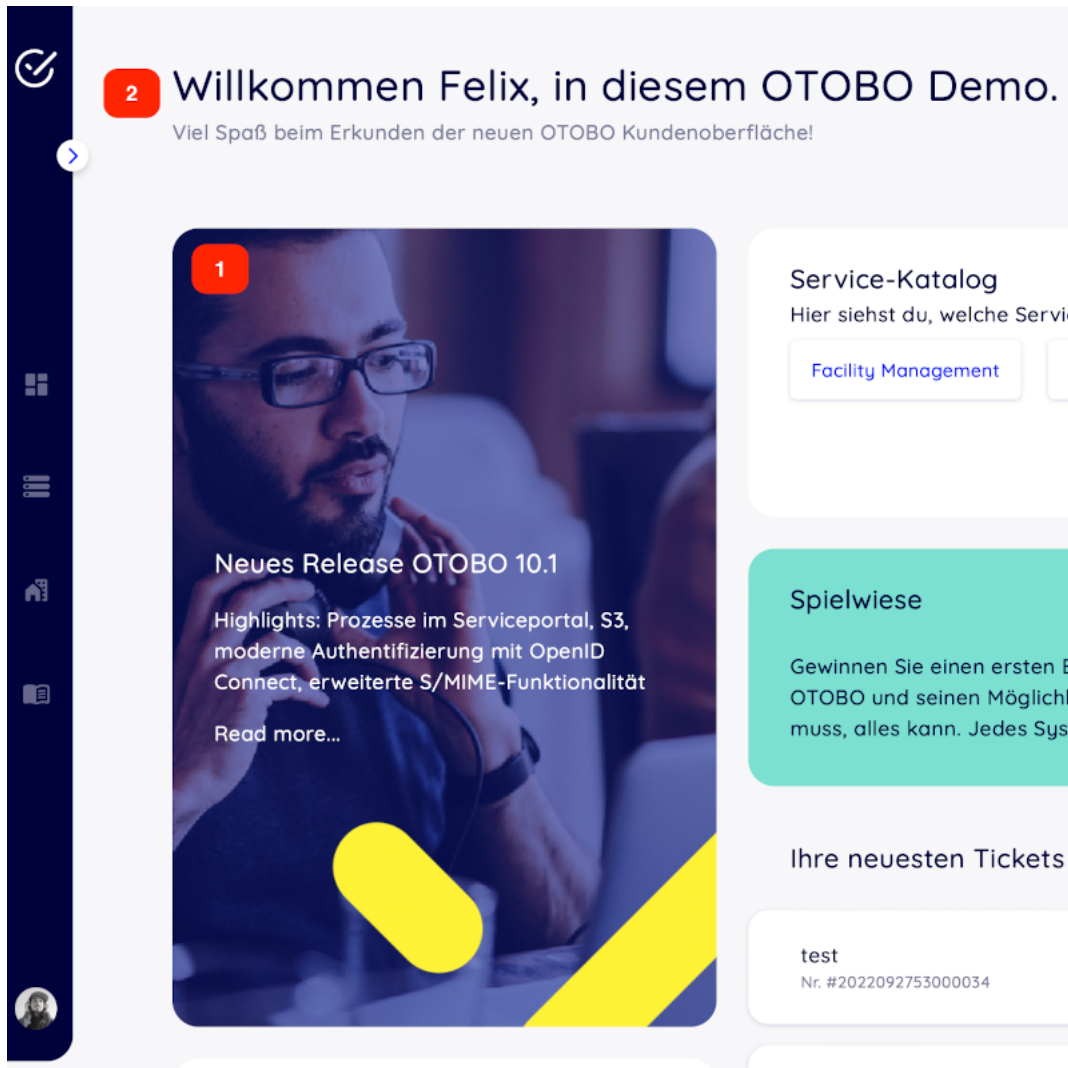
In the search result you will find all the options you need with colour definitions.

- 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
```

□□□□

```
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, this 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
```





Ticket::SearchIndex::Attribute

WordCountMax	<input type="checkbox"/>	1000
WordLengthMax	<input type="checkbox"/>	30
WordLengthMin	<input type="checkbox"/>	3

Basic fulltext index settings. Execute "bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild" in order to generate a new index.

Fig. 13.1: Ticket::SearchIndex::Attribute

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

“ WordCountMax “ 1000

WordLengthMin and WordLengthMax

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

Ticket::SearchIndex::Filters

[\\&\\<>\\?\"\\^\\ ;\\[]\\(\\)+\\\$\\^=]	Fulltext index regex filters to remove parts of the text.
^[:]:[[:.]]\$	
^[^w]+\$	

Fig. 13.2: Ticket::SearchIndex::Filters

- [\\&\\<>\\?\"\\^\\|;\\[]\\(\\)+\\\$\\^=]
- ^[:]:[[:.]]\$
- [a-zA-Z0-9\_]

“ Ticket::SearchIndex::StopWords “

Ticket::SearchIndex::StopWords###en

a
about
above
after
again
against
all
am

English stop words for fulltext index. These words will be removed from the search index.

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



### 13.4 配置

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorage 属性。

“Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB”

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageDB 属性。

**Note:** 配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageDB 属性。

“Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS”

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageFS 属性。NFS 存储。

**Note:** 配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageFS 属性。

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageFS 属性。

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

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageFS 属性。

**Note:** 配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: ArticleStorageFS 属性。

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: CheckAllStorageBackends 属性。

### 13.5 配置

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: CheckAllStorageBackends 属性。

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: CheckAllStorageBackends 属性。

配置 OTOBO 的 Ticket :: Article :: Backend :: MIMEBase :: CheckAllStorageBackends 属性。

1. 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。

2. 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。

- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。
- 配置 OTOBO 的 Ticket :: ArchiveSystem 属性。

**Note:** 5000

- 1.
2. \*
- 3.

### 13.6

Redis Cache

#### 13.6.1 Redis Cache

1. Redis

Redis OTOBO Redis <https://redis.io/topics/quickstart>

2. Redis Perl Redis::Fast

Redis Redis 'Redis :: Fast' Redis '2' 'otobo.CheckModules.pl -list'

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

3. Redis OTOBO

OTOBO SysConfig -> OTOBO

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

OTOBO /opt/otobo/var/tmp RAM

```
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:** /etc/fstab



# CHAPTER 14

---

□□□□

---

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

©Rother OSS GmbH<sup>®</sup> Oberwaling 31, 94339<sup>®</sup>OTRS AG (original version)©Rother OSS GmbH (<https://rother-oss.com>)

©GNU<sup>®</sup>1.3<sup>®</sup>‘GNU’<<https://www.gnu.org/licenses/fdl-1.3.txt>>‘<sup>®</sup>