OTOBO Installation Guide

Release 10.0

Rother OSS GmbH

2 15, 2024

□□

CHAPTER 1

[illegible]

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 [illegible]

[illegible]OTOBO[illegible]

[illegible]

[illegible]“username> command-to-execute“[illegible]“root>”[illegible]

> **Warning:** [illegible]“username>“ [illegible]

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 derivates, 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□□□□□□□□□□

□□□

```
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 □□□□

□□□□□□□□□□□□□OTOBO□□□□□□□□ OTOBO□□**□□**□□□□□

- □CPU
- 4 GB □□
- 10 GB □□□□

□□□□□□□□□□□□□□□□□□**□□**□□□□□□□

- 3 GHz Xeon □□□□ CPU
- 8 GB □□ □□□ 16 GB□
- 40 GB □□□□

Note: □□□□□□□OTOBO□□□□□□□□□□□□□□□□□□□OTOBO□□□

2.2 □□□□

Perl

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

Web Server

- Apache HTTP Server Version 2.4

□□□

- MySQL 5.6□□□□□
- MariaDB
- PostgreSQL 9.2 □□□□□
- Oracle 10g □□□□□

□□

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

□□□□□

- □□ Safari
- □□ Chrome
- □□ Internet Explorer 11
- □□ Edge

- Mozilla Firefox
- □□□□□□JavaScript□□□□□□

CHAPTER 3

OTOBO

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/stable/en/content/installation-docker.html .

3.1 SELinux

Note: SELinux OTOBO

SELinux "sestatus" `getenforce`

"sestatus" SELinux 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" □□□□□□□□"□□"□

```
root> getenforce
Disabled
```

3.2 □□1□□□OTOBO□□□□□□

□https://ftp.otobo.org/pub/otobo/□□□□□OTOBO□□□□□□□□□□□□□□□□□"tar"□□□□□□□□□"/root/otobo-update"□

```
root> mkdir /opt/otobo-install                                          # 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-10.0.tar.gz     # Download he␣
↪latest OTOBO 10 release
root> tar -xzf otobo-latest-10.0.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□□□□□□□□□□□□□□□□□□□

```
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"□□□□□□□□□□□□□□□□□□□□□□□□□□□□RHEL□□□□□□□□□□□□□□Perl□"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 whem 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: [illegible]

[illegible]

3.6.1 [illegible]

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 [illegible]

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
```

[illegible]

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

[illegible]

```
root> systemctl restart apache2
```

3.7 [illegible]

[illegible]

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

3.8 □□7□□□□□□

□□□□□□□□□□□□□□□□□□□□□□Linux□□□□□MySQL□□MariaDB□□□□□□□PostgreSQL□□Oracle□

□□□□□□□□□□□□□□□□□□□□□□□□□□Linux□□□□□MySQL□□□□□□□□□□

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

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

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

□□□MySQL□□□□□□□□□□□□□

□MySQL 5.7□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□OTOBO□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□MySQL□□□□□□□□□□□□□□□□□□□□“root“ □□□□

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

□MariaDBM 10.1□□□□□□□□□□□□□

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

□□□□□□□□□□□□□□□□□□□□

```
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';
```

□OTOBO□□□□□□□□□□□□□□□□□□□□□□□□

Note: □□□□□□□□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□□□□□□□□□□□□□□□□□□□□“[mysqld]“ □□□MySQL□□□□□□□□“/etc/my.cnf“□`/etc/mysql/my.cnf` □□ `/etc/mysql/mysql.conf.d/mysqld.cnf`□

```
max_allowed_packet    = 64M
```

□□□□□□□□□□□□□"mysqltuner" □□□□□□□□□□□□Github "https://github.com/major/MySQLTuner-perl"□□□□□□□□□□□□□□□Debian□Ubuntu□□□□□□

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

□□□□□□□□□□□

```
root> mysqltuner --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 □□8□□□□□□□

□□ http://localhost/otobo/installer.pl □□□□□□□□□□“localhost”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.11 □□9□□□□□

□□□□□□□http://localhost/otobo/index.pl□□□□□□□□□□“root@localhost“ □□□□□□□□□□□□□□□□□□□

3.12 □□10□□□OTOBO Daemon

OTOBO□□□□□□□□□□OTOBO□□□□□□□□□□□□□□□□□□□□□cron□□□□□□□□□□□OTOBO□□□□□□□□□□□OTOBO□□□□□□□□□□□□□□□□

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

3.13 □□11□OTOBO□□□cron□□

□“/opt/otobo/var/cron/*.dist“□□□□□□□cron□□□□□□□OTOBO Daemon□□□□□□□□□□□□□□□□□□□□□□□□□“.dist”□□□□□□□□□

```
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
```

□□□□□□□□□□□□□□□□□

3.14 □□12□□□Bash Auto-Completion □□□□

□□□□□OTOBO□□□□□□□□□□OTOBO□□□□□□□□□□□□□Linux□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□“bash-completion“□□□□□□□□□□□□□□□“otobo“ □□□□□□□□□“/opt/otobo/.bash_completion“□□□

□□□□shell□□□□□□□□□□□□□□□TAB□□□□□□□□□□□□□□

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

□□□□□□□□□□□□□□□□□TAB□□□□□□□□□□□□□□□□□□□□□□□□□TAB□□□□□□□□□□□□□□□□□

Note: □□□□□□□□□□□□□"otobo" □□□□□□□□□□□□□□□□□"~/.bashrc" □□□□□□□□□□

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

3.15 □□13□□□□□

□□□□□□□□OTOBO□□□□□□performance-tuning □□□

CHAPTER 4

□□□□ Docker□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.

- MariaDB□□□□□□□□□□□
- □OTOBO□□Elasticsearch□□□□□□□
- □□Redis□□□□□□□
- Gazelle□□□□□□Perl□□□□□□
- nginx□□HTTPS□□□□□□□□□□

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

4.1 □□

□□□□□□□□□□□□□□□□□□□□□□□

- Docker 19.03.08
- DockerCompose 1.25.0
- Git 2.25.1

Note: □□□□□□Ubuntu18.04□□□□□□□□https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-ubuntu-18-04□https://wwwdigitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04□□□□□□□□□

□□□□□□□□□□□□□git□Docker□Docker Compose□□□□Ubuntu 20.04□□□□□□□

```
root> apt-get install git docker docker-compose
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 10.0.12 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 runnning 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.

.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□□□□□□□□□□ HTTPS□□□443□□□80□□□HTTPS□□OTOBO Web□□□□□□□□□□□□HTTP□□□ HTTPS□□□□□□□□□□□□□□□□□□□□□□□□□□nginx□□□□□

□□□□□□□□□□□HTTPS□

```
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

□□*.env*□□□□□□□□□□

`OTOBO_DB_ROOT_PASSWORD=<your_secret_password>`

“OTOBO_DB_ROOT_PASSWORD“□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□**otobo**□□□□□□□**otobo**□

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

□□OTOBO□□□□HTTP□□□□□□□□□□□□□□□

nginx□□SSL□□□□□□□□□□□

Note: □□

□□<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubu

Note: □nginx□□□□□□□□□□□□CA□□□□□□□□□□□□□□□□□□□CA□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□Nginx□□□

```
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
```

□□□□□□□□□□* .env *□□□□□□□□□□□□□□□□□□□□

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

□□□□□□□□□□□□□* / etc / nginx / ssl / *□Docker□□□□□□□□□□

4.2.5 □□□□ Docker□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
```

□□□HTTP□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

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

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/datal 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 MySQL□□□□□□□□□□□□□otobo db□

Elasticsearch□□

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

OTOBO_Elasticsearch_ES_JAVA_OPTS □□□□□* OTOBO_Elasticsearch_ES_JAVA_OPTS = -Xms512m -Xmx512m *□□□□□□□□□□□□□□□□4G□

Webserver□□

OTOBO_WEB_HTTP_PORT □□□□HTTP□□□□□□□□□80□□□HTTPS□□HTTP□□□□□□□□HTTPS□

nginx□□□□□□

□□□□□□□HTTPS□□□□

OTOBO_WEB_HTTP_PORT □□□□HTTP□□□□□□□□□80□□□□□□HTTPS□

OTOBO_WEB_HTTPS_PORT □□□□HTTPS□□□□□□□□□443□

OTOBO_NGINX_SSL_CERTIFICATE Nginx Webproxy□SSL□□□□□□* OTOBO_NGINX_SSL_CERTIFICATE = / etc / nginx / ssl / acme.crt *

OTOBO_NGINX_SSL_CERTIFICATE_KEY Nginx Webproxy□SSL□□□□□□* OTOBO_NGINX_SSL_CERTIFICATE_KEY = / etc / nginx / ssl / acme.key *

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, 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 simple 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  # 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: □□□□Nginx□□□□□□□□** listen **□□□□Web□□□□□□□□OTOBO 10.0.3□OTOBO 10.0.4□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□nginx□□□□□□□□□10.0.3□□□□□□□□□□□□□80□443□□□OTOBO 10.0.4□□□□□□8080□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
```

□□□□Docker Compose□□□□□□□□□□□□□□

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

□□□□□□□□□□□

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

□□□□https://hub.docker.com/_/nginx□□“□nginx□□□□□□□□□1.19□□□□□□”□□□

4.4.2 □□□□□□

□□□□□□□443□80□□□HTTPS□HTTP□□* .env *□□□□‘OTOBO_WEB_HTTP_PORT‘□‘OTOBO_WEB_HTTPS_PORT‘□□□□□□□

4.4.3 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. The advantage of this approach is that the 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.

4.4.4 □□□□□□

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.

4.4.5 □□□□

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` [illegible]

```
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
```

4.4.6 [illegible]

** 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 && tar -tvf otobo.tar` list files in an image
- `docker exec -it nginx-server nginx -s reload` reload nginx

docker-compose[illegible]

- `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 □□

- Perl Maven□
- Docker Compose□□□□□□□
- □Ubuntu 18.04 LTS□□□□Docker Compose□□□□□□□
- □Ubuntu 18.04 LTS□□□□Docker□□□□□□□
- docker-otrs□
- □□□
- Dockerfile □□□□□
- ‘Docker□□□□□<https://stackoverflow.com/questions/34814669/when-does-docker-image-cache-invalidation-occur>‘_
- Docker□□IP□□□
- □□□
- ‘□□□□□□□□<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-18-04>‘_
- □□□□□□□

CHAPTER 5

□ OTRS / ((OTRS)) Community Edition version 6 □□□ OTOBO version 10

□□□□□□□□□OTOBO□

OTRS□□□OTRS□□Community Edition□OTOBO□□□□□□□□□□□□□□□□□□□□□□OTOBO□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□

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 6 will be available in OTOBO 10. We do not modify any data of the OTRS 6 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 dedictated 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 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 datase to the OTOBO database server before the actual migration. This can speed up the general migration strategy.

5.2 □□□□□

1. Basic requirement for a migration is that you already have an ((OTRS)) Community Edition or OTRS 6.0.* 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 or ask your question in the OTOBO Community forum at https://forum.otobo.org/.

2. □□□□□□□□□OTOBO□□□□□□□□□□□□□□□□

3. □OTOBO□□□□□□□□□□OTOBO□□□□□OTRS□□□□□□OPM□□□□

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.* 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□□□□□□□□□□□□□□□□□□□□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 webservers. 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"□□□□□□"□□□□□□□□□□□□□□□□□□□□□□□□□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

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: □□□□□□□OTOBO□□□

See also:

□□□□□□OTOBO□backup-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
```

□□* rsysnc *□□□□□□□□□□□□□□□□

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

Note: □□□□□□□□□□OTRS /□□OTRS□□Community Edition□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□OTRS□□□□□□□□□□□□□□□□□□

□□□OTRS□

□□□□OTOBO□□□□□□ `Admin -> System Maintenance`□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“Admin-> Sessions”□□□□□□□□□

5.8.1 □□□□□□□□□OTRS□□□□

□□□□□□□□□□□□□□cron□□□

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

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□□□□□□□□□**root**□□□□Docker□□□□□□□□□□□□□□□□□□□□□□□□□□□□

5.9.1 Docker□□* / opt / otrs □□□□ otobo_opt_otobo *

□□□□□□□□□Docker□□□□* / opt / otrs *□

□□□□□□□□□

1. Docker□□* / opt / otrs □□□□ otobo_opt_otobo *
2. □□* / opt / otrs *□□□□□

□□□□□□□□□□□** a.**□

□□□□□□□□□Docker□□□□* / opt / otrs *□

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

□□□"rsync"□□□□□□□□□□□□Docker□□□□□□□□□□□□□□□"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
```

□□□□□□□□□□□□□* / opt / otobo / tmp / otrs *□

5.10 Optional Step: Streamlined migration of the database

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 transfomation 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.

□□□□mysqldump□□□□□□□□OTRS□□□□□□□□□□□□□Docker□□□□□□□□□□□□□□□* bin / backup.pl *□□□□□□□□

Warning: □□□□□□□□□Docker□□□□* / opt / otrs *□

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

```
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.

5.11 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.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□OTOBO□□□□OTOBO□□□□□

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

□Docker□□□□

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

5.12 Step 6: After Successful Migration!

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

5.13 Known Migration Problems

5.13.1 1. Login after migration not possible

□□Safari□□□□□□□□□□□□□OTRS□□□

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

□ScriptAlias□□□□□□□□□□□□□□□□□□□□□□□□□□□□□otrs□□otobo□□□□□□□□□OTOBO□□□□CSS□JavaScript□□□□□□□□□□□□□□Kernel / Config.pm□□□□□□□□□□□□□□□□□

5.13.3 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.

5.13.4 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/.

5.13.5 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.

5.14 □□ 7□□□□□□□□□□□

5.14.1 1. Password policy rules

□□□□□□□□□□□□□OTOBO 10□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“ PreferencesGroups ### Password”□“ CustomerPersonalPreference #### Password”□□□□□□□□□□□□

□□□□□□	□□
“ PasswordMinSize “	8
“ PasswordMin2Lower2UpperCharacters “	□
“ PasswordNeedDigit “	□
“ PasswordHistory “	10
“ PasswordTTL “	30□
“ PasswordWarnBeforeExpiry “	5□
“ PasswordChangeAfterFirstLogin “	□

5.14.2 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).

5.15 Special topics

5.15.1 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 `expdb` and `impdb` 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`.

CHAPTER 6

□□

Note: □□□□□□□□□□□□□□□□□□□□□□□□□□

6.1 □□ 1□□□□□□□□□□OTOBO□□□□

□□□□□□□□□□□□□□□cron□□□□□□□□OTOBO□□□□□□□□□□□□□□□□□□

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

□□OTOBO cron□□□□□□□□□□□□□□□□□

```
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
```

[illegible]

Warning: [illegible]'backup-restore'[illegible]

6.3 [illegible] 3[illegible]

[illegible]https://ftp.otobo.org/pub/otobo/[illegible]OTOBO[illegible]"tar"[illegible]"/root/otobo-update"[illegible]

```
root> cd /root/otobo-update                                               # Change into␣
↪the update directory
root> wget https://ftp.otobo.org/pub/otobo/otobo-latest-10.0.tar.gz       # Download he␣
↪latest OTOBO 10 release
root> tar -xzf otobo-latest-10.0.tar.gz                                   # Unzip OTOBO
root> cp -r otobo-10.x.x/* /opt/otobo                                     # Copy the␣
↪new otobo directory to /opt/otobo
```

6.3.1 [illegible]

[illegible]OTOBO 10[illegible]"Kernel/Config.pm" [illegible]

```
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 [illegible]

[illegible]OTOBO[illegible]"article" [illegible]"/ opt / otobo / var /"[illegible]

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

6.3.3 [illegible]

[illegible]"*.installed" [illegible]XML[illegible]/ opt / otobo / var / stats[illegible]

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

6.3.4 [illegible]

[illegible]OTOBO[illegible]

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

6.4 [illegible] 4[illegible]

[illegible]OTOBO[illegible]

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

6.5 [illegible] 5[illegible]

[illegible]

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

[illegible]

CHAPTER 7

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□□□□□□□□□□□□□□□
- The new Docker image has to be fetched.
- `docker volume inspect otobo_opt_otobo` □□□□□□
- Some maintainance 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: □□OTOBO 10.0.4□□□□□□□□□□□□

7.1 Updating the Docker Compose files

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

Note: See https://hub.docker.com/repository/docker/rotheross/otobo/tags for the available releases.

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

# Get the latest tags
docker-admin> git pull --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/repository/docker/rotheross/.

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

# fetch the new images, either 'latest' 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
```

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.

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

CHAPTER 8

□□□□□

OTOBO□□□□□□□□□□□□□□□□□□□□□□□□□□□"-h"□□□□□□□□□□□□□

8.1 □□

Note: □□□□□□□□□□□□□□ "otobo"□□□□□□□□□□□□□□□

```
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
```

```
 [--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 [illegible]

Note: [illegible]"otobo" [illegible]

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

[illegible]

```
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 [illegible]Docker[illegible]OTOBO[illegible]

[illegible]OTOBO[illegible]Docker[illegible]Docker[illegible]

[illegible]"/otobo_backup" [illegible]

[illegible]

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

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

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

□□□□□□□□□□□□□□□□□□□□□□□□“otobo_opt_otobo“ □□ otobo_backup□□□□□□□□□□□□Daemon□□□□□□□□□□□□□

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

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

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“<TIMESTAMP>“ □□□“2020-09-07_09-38“□□□□□□

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

CHAPTER 9

□□□□□

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

9.1 □Docker□□□OTOBO□□□

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 located within in 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 otobo_backup

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

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

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

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 otobo_opt_otobo:/opt/otobo --volume otobo_
↪backup:/otobo_backup --network otobo_default rotheross/otobo:latest scripts/backup.
↪pl -d /otobo_backup

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

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

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□"<TIMESTAMP>" □□□"2020-09-07_09-38"□□□□□□

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

CHAPTER 10

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 is isn't already available.

Alternatively, many Perl modules are also available as packages for your operating system. These packages can be installed with your systems 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
```

10.1 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
```

CHAPTER 11

[illegible]

[illegible]OTOBO[illegible]

11.1 [illegible]

[illegible]"Ticket :: IndexModule"[illegible]

" **Kernel::System::Ticket::IndexAccelerator::RuntimeDB** " [illegible]60,000[illegible]

" **Kernel::System::Ticket::IndexAccelerator::StaticDB** " [illegible]80,000[illegible]"ticket_index"[illegible]

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

11.2 [illegible]

OTOBO[illegible]

[illegible]

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

Note: [illegible]OTOBO[illegible]

[illegible]

" **Ticket::SearchIndex::IndexArchivedTickets** " [illegible]

" **Ticket::SearchIndex::Attribute** " [illegible]

Note: [illegible]

Fig. 11.1: `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.

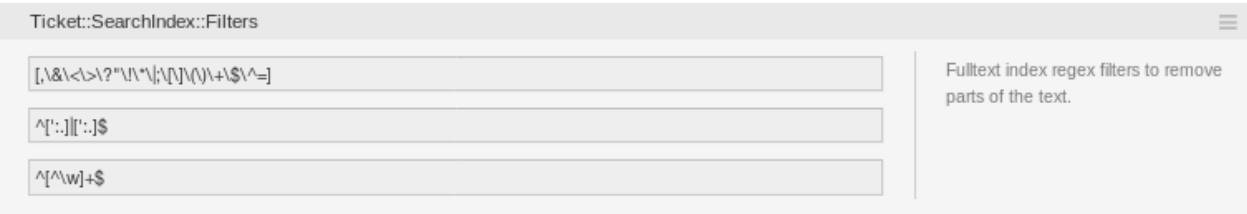

Fig. 11.2: `Ticket::SearchIndex::Filters`

- <>“* |; [] + $ ^ =
- ’
- a-z A-Z 0-9 _

“ Ticket::SearchIndex::StopWords “

Fig. 11.3: `Ticket::SearchIndex::StopWords###en`

[illegible]

See also:

[illegible]

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

11.3 [illegible]

OTOBO[illegible]Elasticsearch[illegible]Elasticsearch[illegible]“[illegible]<https://www.elastic.co/guide/zh-cn/elasticsearch/reference/current/getting-started.html>” __ [illegible]

11.3.1 [illegible]

Elasticsearch[illegible]Java[illegible]Java[illegible]JVM[illegible]JVM[illegible]*[illegible]*[illegible]“jvm.options”[illegible]

[illegible]1 GB[illegible]

- `Xms1g` [illegible]
- `Xmx1g`: maximum heap size.

[illegible]“Xms”[illegible]“Xmx”[illegible]JVM[illegible]“Xmx”[illegible]

> **Warning:** [illegible]

[illegible]Elasticsearch[illegible]JVM[illegible]“Xmx”[illegible]50[illegible]

[illegible]‘[illegible] <https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html>‘__[illegible]

11.3.2 [illegible]

[illegible]Elasticsearch[illegible]“ elasticsearch.yml”[illegible]

“ cluster.routing.allocation.disk.watermark.low “ [illegible]85[illegible]Elasticsearch[illegible]

`cluster.routing.allocation.disk.watermark.high` [illegible]90[illegible]Elasticsearch[illegible]

“ cluster.routing.allocation.disk.watermark.flood_stage “

[illegible]95[illegible]Elasticsearch[illegible]“ index.blocks.read_only_allow_delete”[illegible]

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:** [illegible]*[illegible]*[illegible]Elasticsearch[illegible]

[illegible] [illegible][illegible]

11.4 [illegible]

[illegible]"Ticket :: Article :: Backend :: MIMEBase :: ArticleStorage"[illegible]

" Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB "
[illegible]

> **Note:** [illegible]

" Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS "
[illegible]NFS[illegible]SAN[illegible]

> **Note:** [illegible]

[illegible]

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

[illegible] `--target` [illegible]

> **Note:** [illegible]CPU[illegible]/[illegible]

[illegible]"Ticket :: Article :: Backend :: MIMEBase :: CheckAllStorageBackends"[illegible]OTOBO[illegible]

11.5 [illegible]

[illegible]OTOBO[illegible]

[illegible]

[illegible]

1. [illegible] "Ticket::ArchiveSystem"[illegible]
2. [illegible]
   - [illegible]*[illegible]*[illegible]*[illegible]*[illegible]
   - [illegible]
   - [illegible]
   - [illegible]
   - [illegible]/[illegible]*[illegible]*[illegible]*[illegible]*[illegible]
   - [illegible]
   - [illegible]*[illegible]*[illegible]
   - [illegible]*[illegible]*[illegible]

Note: □□□□□□□□□□□□□□5000□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□

1. □□□□□□□□□
2. □*□□□□*□□□*□□□□□*□□*□□□□*□
3. □□□□□

11.6 □□

□□□□□□□□□□□□□□□□□□□□□□□□□Redis Cache□□□□□□□□□□□

11.6.1 □□□□ 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
```

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) |
```

11.6.2 □□□□□□

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“□□□□□□□□□

> **Warning:** [illegible]

11.7 [illegible]

[illegible]OTOBO[illegible]Rother OSS[illegible] [illegible]OTOBO [illegible]

CHAPTER 12

□□□□

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