OTOBO Installation Guide

10.0

Rother OSS GmbH

2020 10 20

Contents

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

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

CHAPTER 1

Introduction

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 tested on various Linux platforms.

1.1 About This Manual

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

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

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

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

We supposed that OTOBO will be installed to `/opt/otobo`. If you want to install OTOBO to a different directory, 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

Hardware and Software Requirements

OTOBO can be installed on Linux and other Unix derivates (e.g. OpenBSD or FreeBSD). Running OTOBO on Microsoft Windows is not supported.

To run OTOBO, you'll also need to use a web server as reverse proxy and a database server. Apart from that, you should install Perl and/or install some additional Perl modules on the OTOBO machine.

Perl must be installed on the same machine as OTOBO. The database back end and the web server may be installed locally or on another host.

For Perl, you will need some additional modules which can be installed either with Perl from CPAN, or via the package manager of your operating system (rpm, yast, apt-get).

OTOBO has also a console command for missing modules.

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

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

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

The listed commands should then be executed with root privileges.

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

```
Checking for Perl Modules:
  o Archive::Tar.....................ok (v2.24)
  o Archive::Zip.....................ok (v1.63)
  o Crypt::Eksblowfish::Bcrypt.......ok (v0.009)
  o Crypt::SSLeay....................ok (v0.73_06)
  o CryptX...........................ok (v0.061)
  o Date::Format.....................ok (v2.24)
  o DateTime.........................ok (v1.50)
  o DBI..............................ok (v1.641)
  o DBD::mysql.......................ok (v4.046)
```

```
  o DBD::ODBC.......................Not installed!  Use: 'apt-get install -y libdbd-
↪odbc-perl' (optional - Required to connect to a MS-SQL database.)
  o DBD::Oracle.....................Not installed!  Use: 'cpan DBD::Oracle'␣
↪(optional - Required to connect to an Oracle database.)
  o DBD::Pg.........................Not installed!  Use: 'apt-get install -y libdbd-
↪pg-perl' (optional - Required to connect to a PostgreSQL database.)
  o Digest::SHA.....................ok (v5.96)
  o Encode::HanExtra................ok (v0.23)
  o EV..............................ok (v4.22)
  o IO::Socket::SSL.................ok (v2.060)
  o JSON::XS........................ok (v3.04)
  o List::Util::XS..................ok (v1.46_02)
  o LWP::UserAgent..................ok (v6.35)
  o Mail::IMAPClient................ok (v3.39)
    o Authen::SASL..................ok (v2.16)
    o Authen::NTLM..................ok (v1.09)
  o Moose...........................ok (v2.2011)
  o Net::DNS........................ok (v1.17)
  o Net::LDAP.......................ok (v0.65)
  o Search::Elasticsearch...........ok (v6.00)
  o Specio..........................ok (v0.42)
  o Specio::Subs....................ok (v0.42)
  o Template........................ok (v2.27)
  o Template::Stash::XS.............ok (undef)
  o Text::CSV_XS....................ok (v1.36)
  o Time::HiRes.....................ok (v1.9741)
  o XML::LibXML.....................ok (v2.0132)
  o XML::LibXSLT....................ok (v1.96)
  o XML::Parser.....................ok (v2.44)
  o YAML::XS........................ok (v0.74)

Checking for External Programs:
  o GnuPG...........................ok (v2.2.8)
  o npm.............................ok (v5.8.0)
    o Node.js.......................ok (v8.11.4)
  o OpenSSL.........................ok (v1.1.1/OpenSSL)
```

2.1 Hardware Requirements

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

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

- small CPU
- 4 GB RAM
- 10 GB storage

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

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

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

2.2 Software requirements

Perl

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

Web Servers

- Apache2
- nginx
- Any other web server that can be used as a reverse proxy

Databases

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

Optional

- Elasticsearch 7.x (fast search function for live previews)
- Node.js 8.9 or higher (only for development)

Web browsers

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

CHAPTER 3

OTOBO Installation

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

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

3.1 Preparation: Disable SELinux when it is installed and enabled

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

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

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

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

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

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

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

```
root> getenforce
Disabled
```

3.2 Step 1: Unpack and Install OTOBO

Download the latest otobo release from https://ftp.otobo.org/pub/otobo/. Unpack the source archive (for example, using `tar`) into the directory `/opt/otobo-install`:

```
root> mkdir /opt/otobo-install                                          # 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 Step 2: Install Additional Programs and Perl Modules

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

```
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)
...
```

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

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

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

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

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

3.4 Step 3: Create the OTOBO User

Create a dedicated user for OTOBO within its own group:

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

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

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

3.5 Step 4: Activate the Default Configuration File

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

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

3.6 Step 5: Configure the Apache Web Server

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

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

# SuSE:
root> zypper install apache2-mod_perl

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

OTOBO requires a few Apache modules to be active for optimal operation. 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
```

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

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

3.6.1 Configure Apache without SSL support

Copy the appropriate template in `/opt/otobo/scripts/apache2-httpd.include.conf` to a file called `zzz_otobo.conf` in the Apache configuration directory (to make sure it is loaded after the other configurations).

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

3.6.2 Configure Apache with SSL support

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

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

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

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

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

```
root> systemctl restart apache2
```

3.7 Step 6: Set File Permissions

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

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

3.8 Step 7: Setup the Database

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

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

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

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

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

After installing the MySQL server you need configure it.

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

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

For MariaDB > 10.1 use instead the following command:

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

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

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

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

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

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

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

```
query_cache_size     = 32M
```

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

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

After installing execute the script:

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

3.9 Step 8: Setup Elasticsearch

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

3.9.1 Elasticsearch installation example based on Ubuntu 18.04 LTS

JDK Installation

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

Elasticsearch Installation

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

3.9.2 Elasticsearch Installation on another Linux distribution

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

3.9.3 Elasticsearch Module Installation

Additionally, OTOBO requires plugins to be installed into Elasticsearch:

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

3.9.4 Elasticsearch Configuration

Elasticsearch has a multitude of configuration options and possibilities.

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

```
-Xms4g
-Xmx4g
```

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

```
See ``https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.
↪html`` for more information.
```

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

```
root> systemctl restart elasticsearch
```

3.10 Step 8: Basic System Configuration

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

3.11 Step 9: First Login

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

3.12 Step 10: Start the OTOBO Daemon

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

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

3.13 Step 11: Cron jobs for the OTOBO user

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

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

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

With this step, the basic system setup is finished.

3.14 Step 12: Setup Bash Auto-Completion (optional)

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

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

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

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

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

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

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

3.15 Step 13: Further Information

We advise you to read the OTOBO *Performance Tuning* chapter.

CHAPTER 4

Installing using Docker and Docker Compose

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

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

: At the moment the Docker Compose environment is not tested in depth for production use. Please use the standard installation process for production use. Brave early adaptors are very welcome to share their experiences.

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

4.1 Requirements

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

- Docker 19.03.06
- Docker Compose 1.25.0
- Git 2.17.1

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

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

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

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

4.2 Installation

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

4.2.1 1. Clone the otobo-docker repo

The Docker images will eventually be fetched from https://hub.docker.com. But some setup and command files need to be cloned from the otobo-docker Github repository. Make sure that you use the fitting versions of these files. The git tag must correspond to the version of OTOBO. Here we use OTOBO 10.0.4 as an example.

: This manual had been published before OTOBO 10.0.4 was released. This means that the git label rel-10.0.4 might not exist yet.

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

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo-docker.git --branch rel-10_
↪0_4 --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* allows you to manage your installation of OTOBO. This file must first be created and then be adapted by yourself. In order to simplify the task there are two example files that should be used as starting point. If the OTOBO web application is planned to be accessed via HTTPS, then please use *.docker_compose_env_https*. Access via HTTPS is the recommended mode of operation. If HTTPS is not required then use *.docker_compose_env_http* as the starting point.

: Use `ls -a` for listing the hidden template files.

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

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

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

4.2.3 3. Configure the password for the database admin user

Change the following setting inside your *.env* file:

```
OTOBO_DB_ROOT_PASSWORD=<your_secret_password>
```

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

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

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

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

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

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

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

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

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

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

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

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

4.2.5 5. Start the Docker containers with Docker Compose

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

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

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

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

4.2.6 6. Install and start OTOBO

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

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

Have fun with OTOBO!

: 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 exec -it otobo_web_1 bash`.

4.3 Additional technical information

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

4.3.1 List of Docker containers

Container otobo_web_1 OTOBO webserver on internal port 5000.

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

Container otobo_db_1 Run the database MariaDB on internal port 3306.

Container otobo_elastic_1 Elasticsearch on the internal ports 9200 and 9300.

Container otobo_redis_1 Run Redis as caching service.

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

4.3.2 Overview over the Docker volumes

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

otobo_opt_otobo contains */opt/otobo* in the container *web* and *daemon*.

otobo_mariadb_data contains */var/lib/mysql* in the container *db*.

otobo_elasticsearch_data contains */usr/share/elasticsearch/data1* in the container *elastic*.

otobo_redis_data contains data for the container *redis*.

otobo_nginx_ssl contains the TLS files, certificate and private key, must be initialized manually.

4.3.3 Docker environment variables

In the instructions we did only minimal configuration. But the file *.env* allows to set more variables. Here is a list of all supported environment variables:

MariaDB settings

OTOBO_DB_ROOT_PASSWORD The root password for MySQL. Must be set for running otobo db.

Elasticsearch settings

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

OTOBO_Elasticsearch_ES_JAVA_OPTS Example setting: *OTOBO_Elasticsearch_ES_JAVA_OPTS=-Xms512m -Xmx512m* Please adjust this value for production env to a value up to 4g.

Webserver settings

OTOBO_WEB_HTTP_PORT Set in case the HTTP port should deviate from the standard port 80. When HTTPS is enabled, the HTTP port will redirect to HTTPS.

nginx webproxy settings

These setting are used when HTTPS is enabled.

OTOBO_WEB_HTTP_PORT Set in case the HTTP port should deviate from the standard port 80. Will redirect to HTTPS.

OTOBO_WEB_HTTPS_PORT Set in case the HTTPS port should deviate from the standard port 443.

OTOBO_NGINX_SSL_CERTIFICATE SSL cert for the nginx webproxy. Example: *OTOBO_NGINX_SSL_CERTIFICATE=/etc/nginx/ssl/acme.crt*

OTOBO_NGINX_SSL_CERTIFICATE_KEY SSL key for the nginx webproxy. Example: *OTOBO_NGINX_SSL_CERTIFICATE_KEY=/etc/nginx/ssl/acme.key*

Docker Compose settings

These settings are used by Docker Compose directly.

COMPOSE_PROJECT_NAME The project name is used as a prefix for the generated volumes and containers. Must be set because the compose file is located in *scripts/docker-compose* and thus **docker-compose** would be used per default as the project name.

COMPOSE_PATH_SEPARATOR Separator for the value of COMPOSE_FILE

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

OTOBO_IMAGE_OTOBO, OTOBO_IMAGE_OTOBO_ELASTICSEARCH, OTOBO_IMAGE_OTOBO_NGINX Used for specifying alternative Docker images. Useful for testing local builds.

4.4 Advanced topics

4.4.1 Custom configuration of the nginx webproxy

The default Docker-based OTOBO installation provides the container *otobo_nginx_1*. This container provides HTTPS support for the HTTP-based OTOBO web application. The de-

fault config template for nginx can be found within the Docker image, specifically in the file */etc/nginx/template/otobo_nginx.conf.template*. When the container is started, the actually used configuration file is generated from the template. This is done by replacing each macro in the template with the corresponding environment variable. In the default template file, only the following macros are used: * *${OTOBO_NGINX_SSL_CERTIFICATE}* * *${OTOBO_NGINX_SSL_CERTIFICATE_KEY}* * *'${OTOBO_NGINX_WEB_HOST}* * *${OTOBO_NGINX_WEB_PORT}*

There are various possibilities for customizing the nginx configuration. One way is to use a locally built image that is derived from the image *otobo-nginx-webproxy*. In such a local image, nginx can be configured in a very flexible way.

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

Another supported approach is to only override the default config template with a customized version. In this case, we have to create a volume that contains the adapted nginx config template, first.

```
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose down
docker_admin> docker volume create otobo_nginx_custom_config
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> 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
docker_admin> # adapt the file $otobo_nginx_custom_config_mp/otobo_nginx.conf.template␣
↪to your needs
docker_admin> docker-compose up --detach
```

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

After setting up the volume, the adapted configuration must be activated. In order to achieve this, uncomment or add the following lines in your *.env* file:

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

The changed Docker Compose configuration can be inspected with:

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

Finally, the containers can be started again:

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

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

4.4.2 Choosing non-standard ports

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

4.4.3 Building local images

: Building Docker images locally is usually only needed during development.

The 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*
- *bin/docker/build_docker_images.sh*

```
docker_admin> cd /opt
docker_admin> git clone https://github.com/RotherOSS/otobo.git
docker_admin> cd otobo
docker_admin> bin/docker/build_docker_images.sh
docker_admin> docker image ls
```

The locally built images are tagged as `local-<OTOBO_VERSION>` using the version set up the file *RELEASE*. After building the local images, one can specify the images to be used by setting `OTOBO_IMAGE_OTOBO`, `OTOBO_IMAGE_OTOBO_ELASTICSEARCH`, `OTOBO_IMAGE_OTOBO_NGINX` in *.env*.

4.4.4 Automatic Installation

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

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

```
docker_admin> docker-compose down -v
docker_admin> docker-compose up --detach
docker_admin> docker stop otobo_daemon_1
docker_admin> docker exec -t --user otobo otobo_web_1 bash\
-c "rm -f Kernel/Config/Files/ZZZAAuto.pm ; bin/docker/quick_setup.pl --db-password␣
↪otobo_root"
docker_admin> docker exec -t --user otobo otobo_web_1 bash\
-c "bin/docker/run_test_suite.sh"
.......
docker_admin> docker start otobo_daemon_1
```

4.4.5 List of useful commands

Docker

- `docker system prune -a` system clean-up (removes all unused images, containers, volumes, networks)
- `docker version` show version
- `docker build --tag otobo --file=otobo.web.Dockerfile .` build an image
- `docker run --publish 80:5000 otobo` run the new image
- `docker run -it -v opt_otobo:/opt/otobo otobo bash` log into the new image
- `docker run -it -v opt_otobo:/opt/otobo --entrypoint bash otobo` try that in case entrypoint.sh is broken
- `docker ps` show running images
- `docker images` show available images
- `docker volume ls` list volumes
- `docker volume inspect otobo_opt_otobo` inspect a volume
- `docker volume inspect --format '{{ .Mountpoint }}' otobo_nginx_ssl` get volume mountpoint
- `docker volume rm tmp_volume` remove a volume
- `docker inspect <container>` inspect a container
- `docker save --output otobo.tar otobo:latest && 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

4.4.6 Resources

- Perl Maven
- Docker Compose quick start
- Newer version of Docker Compose on Ubuntu 18.04 LTS
- Newer version of Docker on Ubuntu 18.04 LTS
- docker-otrs
- cleanup
- Dockerfile best practices
- Docker cache invalidation
- Docker Host IP
- Environment
- Self signed certificate

- Inspect failed builds

CHAPTER 5

Migration from OTRS / ((OTRS)) Community Edition version 6 to OTOBO version 10

Welcome and thank you for choosing OTOBO!

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

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

If you have any problem or question, please do not give up :) 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.

: After the migration all data previously available in OTRS 6 will be available in OTOBO. We do not modify any OTRS data during the migration.

5.1 Migration Possibilities

With the OTOBO Migration Interface it is possible to perform the following migrations:

1. A 1:1 migration on the same application server with the same database server.
2. A migration and simultaneous move to a new application server and operating system.
3. It is irrelevant whether your OTRS/ ((OTRS)) Community Edition was previously installed on two separate servers (application and database servers), or whether you want to change OTOBO to such a configuration.
4. It is possible to migrate from any of the supported databases to any other one.
5. It is possible to switch from any supported operating system to any other supported operating system during the migration.

6. It is possible to migrate to a Docker based installation of OTOBO 10.

5.2 Migration Requirements

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 configuration and data to OTOBO.

: Please consider carefully whether you really need the data and configuration. Experience shows that quite often a new start is the better option, as the previously used installation and configuration was rather suboptimal anyway. 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. You need a running OTOBO installation to start the migration from there!
3. This OTOBO installation must contain all OPM packages installed in your OTRS that you want to use in OTOBO, too.

4. If you are planning to migrate to another server, then the OTOBO webserver must be able to access the location where your ((OTRS)) Community Edition or OTRS 6.0.* is installed. In most cases, this is the directory /opt/otrs on the server running OTRS. The access can be effected via SSH or via file system mounts. Furthermore, the *otrs* database must be accessible from the server running OTOBO. Readonly access must be granted for external hosts.

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

5.3 Step 1: Install the new OTOBO System

Please start with installing the new OTOBO system to which your OTRS / ((OTRS)) Community Edition instance will then be migrated. We strongly recommend to read the *OTOBO Installation* chapter.

: Under Apache, there are pitfalls with running two independent applications under mod_perl on the same server. These pitfalls can be alleviated with the mod_perl setting `PerlOptions +Parent`. This setting makes sure that the relevant application uses it's own dedicated Perl interpreter. Please check your Apache config files in the directory */etc/apache2/sites-enabled* and add the setting in case it is missing.

After finishing the installation tutorial, please login to the OTOBO Admin Area `Admin -> Packages` to install all required OTOBO OPM packages.

The following OPM packages and OTRS "Feature Addons" need NOT and should NOT be installed, as these featur

- OTRSHideShowDynamicField
- RotherOSSHideShowDynamicField
- TicketForms

- RotherOSS-LongEscalationPerformanceBoost
- Znuny4OTRS-AdvancedDynamicFields
- Znuny4OTRS-AutoSelect
- Znuny4OTRS-EscalationSuspend
- OTRSEscalationSuspend
- OTRSDynamicFieldAttachment
- OTRSDynamicFieldDatabase
- OTRSDynamicFieldWebService
- OTRSBruteForceAttackProtection
- Znuny4OTRS-ExternalURLJump
- Znuny4OTRS-QuickClose
- Znuny4OTRS-AutoCheckbox
- OTRSSystemConfigurationHistory

5.4 Step 2: Preparing the new OTOBO system and server

After installing OTOBO, please log in again to the OTOBO Admin Area `Admin -> System Configuration` and deactivate the config option `SecureMode`. Now log in on the server as user `root` and execute the following commands:

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

When OTOBO is running in Docker, you just need to stop the Docker container `otobo_daemon_1`:

```
docker_admin> cd /opt/otobo-docker
docker_admin> docker-compose stop daemon
```

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

:

We advise you to read the OTOBO *Backup and Restore* chapter.

5.4.1 Install sshpass and rsysnc if you want to migrate OTRS from another server

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

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

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

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

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

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

5.4.2 Docker: copy */opt/otrs* into the volume *otobo_opt_otobo*

In this section, we assume that */opt/otrs* is available on the Docker host.

In the case when the web application OTOBO runs inside the container `otobo_web_1`, OTOBO generally cannot access directories outside the container. There is an exception though: directories mounted as volumes into the container can be accessed. This means that for migration there are two possibilities:

1. copy */opt/otrs* into an existing volume
2. mount */opt/otrs* as an additional volume

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

For safe copying, we use `rsync`. But first we need to find out the correct target.

```
root> mountpoint_opt_otobo=$(docker volume inspect --format '{{ .Mountpoint }}' otobo_
↪opt_otobo)
root> echo $mountpoint_opt_otobo
root> rsync --recursive --safe-links --owner --group --chown 1000:1000 --perms --
↪chmod "a-wx,Fu+r,Du+rx" /opt/otrs/ $mountpoint_opt_otobo/tmp/otrs
```

This copied directory will be available as */opt/otobo/tmp/otrs* within the container.

5.5 Step 3: Preparing the OTRS / ((OTRS)) Community Edition system

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

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

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

5.5.1 Stop All Relevant Services and the OTRS Daemon

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

```
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
```

5.6 Step 4: Perform the Migration!

Please use the web migration tool at http://localhost/otobo/migration.pl (replace "localhost" with your OTOBO hostname) and follow the process.

: If OTOBO runs inside a Docker container, specify *localhost* for OTRS server and */opt/otobo/tmp/otrs* as the OTRS home directory.

: In the Docker case, a local database won't be reachable via `127.0.0.1` from within the Docker container. Pick one of the IP-addresses reported by `hostname --all-ip-addresses` instead for `OTRS Server`. In order to make sure that there is a database user who can read the data, it might be worthwhile to create a dedicated user. E.g. `CREATE USER 'otrs_migration'@'%' IDENTIFIED BY 'otrs_migration'` and `GRANT SELECT, SHOW VIEW ON otrs.* TO 'otrs_migration'@'%'`.

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

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

In the docker case:

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

5.7 Step 5: After Successful Migration!

1. Uninstall `sshpass` if you do not need it anymore.
2. Drop the databases user dedicated to the migration if you created one.
3. Have fun with OTOBO!

5.8 Step 6: Known Migration Problems

5.8.1 1. Login after migration not possible

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

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

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

5.9 Step 7: Manual Migration Tasks and Changes

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

Password Policy Rule	Default
`PasswordMinSize`	8
`PasswordMin2Lower2UpperCharacters`	Yes
`PasswordNeedDigit`	Yes
`PasswordHistory`	10
`PasswordTTL`	30 days
`PasswordWarnBeforeExpiry`	5 days
`PasswordChangeAfterFirstLogin`	Yes

CHAPTER 6

Updating

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

6.1 Step 1: Stop All Relevant Services and the OTOBO Daemon

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

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

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

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

6.2 Step 2: Backup Files and Database

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

6.2.1 Example for a standard installation with Ubuntu and MySQL

```
root> mkdir /root/otobo-update                          # Create a update directory
root> cd /root/otobo-update                             # Change into the update directory
root> cp -pr /opt/otobo otobo-prod-old                  # Backup the hole OTOBO directory␣
↪to the update directory
root> mysql -otobo -p otobo > otobo-prod-old.sql        # Backup the otobo database and␣
↪save as dump otobo-prod-old.sql
```

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

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

6.3 Step 3: Install the New Release

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

```
root> cd /root/otobo-update                                                 # Change into␣
↪the update directory
root> wget https://ftp.otobo.org/pub/otobo/otobo-latest-10.0.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 Restore Old Configuration Files

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

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

6.3.2 Restore Article Data

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

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

6.3.3 Restore Already Installed Default Statistics

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

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

6.3.4 Set File Permissions

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

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

6.4 Step 4: Update Installed Packages

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

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

6.5 Step 5: Start your Services

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

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

Now you can log into your system.

CHAPTER 7

Updating OTOBO using Docker and Docker Compose

7.1 Updating to a new patch level release

First make sure that in *.env* the images have either the tag *latest* or the wanted version.

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

# Update OTOBO docker-compose repository
docker-admin> git checkout rel-10_0_0 # Please use the required version

# fetch the new images that are tagged a 'latest'
docker_admin> docker-compose pull

# stop and remove the containers, named volumes are kept
docker_admin> docker-compose down

# start again with the new images
docker_admin> docker-compose up --detach
```

After updating you need to reinstall all OTOBO packages and clear the cache.

```
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Admin::Package::ReinstallAll
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Admin::Package::UpgradeAll
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Maint::Config::Rebuild
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Maint::Cache::Delete
```

7.2 Force an update to or from a devel version

Images of devel versions are not upgraded automatically. But the upgrade can be forced. The source of the devel version can either be a local build or a devel image from Docker Hub. Here is an example using the devel image for the OTOBO 10.1.x branch from Docker Hub.

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

# force upgrade, skip reinstall
docker_admin> docker run -it --rm --volume otobo_opt_otobo:/opt/otobo rotheross/
↪otobo:devel-rel-10_1 upgrade

start again with the new version
docker_admin> docker-compose up -d
```

After updating you need to reinstall all OTOBO packages and clear the cache.

```
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Admin::Package::ReinstallAll
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Admin::Package::UpgradeAll
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Maint::Config::Rebuild
docker_admin> docker exec -it -uotobo otobo_web_1 bin/otobo.Console.pl␣
↪Maint::Cache::Delete
```

CHAPTER 8

Backup and Restore

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

8.1 Backup

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

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

The output of the script:

```
Backup an OTOBO system.

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

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


Long options:
 [--help]                      - same as -h
 --backup-dir                  - same as -d
```

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

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

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

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

8.2 Restore

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

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

The output of the script:

```
Restore an OTOBO system from backup.

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

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

8.3 Considerations for running OTOBO under Docker

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

First we need to make sure that the backup files are not created in the file system that is internal to the container. Because in that case all data would be lost when the container is stopped. Therefore the backup directory must be in a volume. For now we only consider the most simple case, where the backup dir is a local dir on the Docker host. The location of the backup dir in the container can be arbitrarily chosen. In this

example we choose the local dir `otobo_backup` as the location on the host, and `/otobo_backup` as the location in the container.

First we need to create the volume.

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

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

```
# create a backup
docker_admin> docker run -it --rm --volume 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

Backup and Restore using Docker

Please read the Backup and Restore description *Backup and Restore* for base informations about the backup and restore scripts.

9.1 Considerations for running OTOBO under Docker

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

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

First we need to create the volume.

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

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

```
# create a backup
docker_admin>docker run -it --rm --volume 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

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 that modules from CPAN are installed with the command line client `cpanm`. Alternatively, many Perl modules are also available as packages for your operating system. These packages can be installed via your systems regular package manager.

The utility `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.

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

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

results in:

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

Special care must be taken when OTOBO runs within Docker. In this case, it seems an easy option to install the wanted module into the system Perl. However, due to how Docker works, this change would be lost when the container is restarted. Therefore the modules must be installed into a location that survives a restart. The local install location can be specified with the option `--local-lib`. Installed modules will be found by Perl because of the environment variables `PERL5LIB` and `PATH`, which are set-up accordingly in the Docker image.

```
otobo> cd /opt/otobo
otobo> cpanm --local-lib local 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
```

```
otobo> perldoc -l Acme::Dice
/opt/otobo/local/lib/perl5/Acme/Dice.pm

otobo: echo $PERL5LIB
/opt/otobo_install/local/lib/perl5:/opt/otobo/local/lib/perl5

otobo: 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

Performance Tuning

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

11.1 Ticket Index Module

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

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

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

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

11.2 Ticket Search Index

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

To create an initial index, use this command:

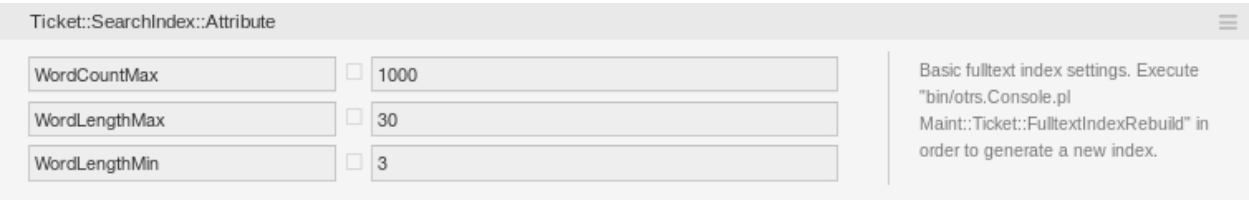
```
otobo> /opt/otobo/bin/otobo.Console.pl Maint::Ticket::FulltextIndex --rebuild
```

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

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

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

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



11.1: `Ticket::SearchIndex::Attribute` Setting

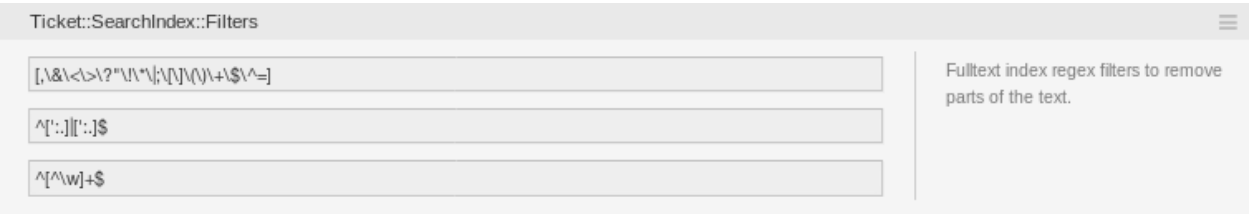
: Run the following command in order to generate a new index:

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

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

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

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



11.2: `Ticket::SearchIndex::Filters` Setting

There are three default filters defined:

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

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

11.3: `Ticket::SearchIndex::StopWords###en` Setting

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

:

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

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

11.3 Document Search

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

11.3.1 Heap Size

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

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

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

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

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

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

For more information and good rules of thumb about the heap size, please follow the official documentation.

11.3.2 Disk Allocation

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

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

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

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

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

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

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

For more information about disk watermarks and disk-based shard allocation, please follow the official documentation.

11.4 Article Storage

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

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

: Don't use this with large setups.

`Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS` Use this module to store attachments on the local file system. It is fast, but if you have multiple front end

servers, you must make sure the file system is shared between the servers. Place it on an NFS share or preferably a SAN or similar solution.

: Recommended for large setups.

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

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

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

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

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

11.5 Archiving Tickets

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

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

To use the archive feature:

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

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

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

To search for archived tickets:

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

11.6 Caching

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

11.6.1 Install a Redis Cache Server

1. Install Redis Server

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

2. Install Perl module Redis or Redis::Fast

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

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

3. Configure OTOBO for Redis

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

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

11.6.2 RamDisk Caching

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

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

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

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

11.7 Clustering

For very high loads, it can be required to operate OTOBO on a cluster of multiple front end servers. This is a complex task with many pitfalls. Therefore, Rother OSS provides support for clusters in its managed OTOBO environment exclusively.

CHAPTER 12

Documentation History

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

Published by: Rother OSS GmbH, (https://otobo.de), Oberwalting 31, 94339 Leiblfing, Germany. Authors: OTRS AG (original version), Rother OSS GmbH (https://rother-oss.com)