# CONTENTS

1 Introduction ................................................. 3

2 Hardware and Software Requirements ......................... 5
   2.1 Hardware Requirements .................................. 6
   2.2 Software Requirements ................................... 6

3 Installation .................................................. 9

4 Updating ..................................................... 11
   4.1 Step 1: Stop All Relevant Services and the OTRS Daemon ..... 11
   4.2 Step 2: Backup Files and Database .......................... 12
   4.3 Step 3: Install the New Release ............................ 12
      4.3.1 Restore Old Configuration Files ......................... 13
      4.3.2 Restore Article Data ..................................... 13
      4.3.3 Restore Already Installed Default Statistics .......... 13
      4.3.4 Set File Permissions ..................................... 13
      4.3.5 Install Required Programs and Perl Modules ........... 13
   4.4 Step 4: Run the Migration Script .......................... 14
   4.5 Step 5: Update Installed Packages ......................... 15
   4.6 Step 6: Start your Services ............................... 15
   4.7 Step 7: Start the OTRS Daemon and Web Server ............ 15
   4.8 Step 8: Manual Migration Tasks and Changes ............... 16
      4.8.1 Upgrade Elasticsearch ................................... 17
   4.9 Recommendation: Convert Your MySQL Database ............ 18
      4.9.1 Allow Program Safe to Run ............................... 19

5 Backup and Restore ........................................ 21
   5.1 Backup .................................................... 21
   5.2 Restore .................................................... 22

6 Performance Tuning ....................................... 23
   6.1 Ticket Search Index ...................................... 23
   6.2 Document Search .......................................... 25
      6.2.1 Heap Size ............................................... 25
      6.2.2 Disk Allocation .......................................... 25
   6.3 Article Storage ........................................... 26
   6.4 Archiving Tickets ......................................... 27
   6.5 Tuning the Web Server .................................... 28
   6.6 Caching ..................................................... 28
   6.7 Clustering .................................................. 28
   6.8 Limits of Objects .......................................... 29
This manual is intended for use by system administrators. The chapters describe the updating of the OTRS software.

There is no graphical user interface for 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 who has root permissions. If a command starts with `otrs>`, you have to execute the command as the user created for OTRS.

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

We supposed that OTRS is installed to `/opt/otrs`. If OTRS is installed to a different directory, then you have to change the path in the commands or create a symbolic link to this directory.

```
root> ln -s /path/to/otrs /opt/otrs
```
HARDWARE AND SOFTWARE REQUIREMENTS

OTRS can be installed on Linux and on other Unix derivates (e.g. OpenBSD or FreeBSD). To run OTRS on Microsoft Windows is not possible.

To run OTRS, 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 OTRS machine.

Perl must be installed on the same machine as OTRS. 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 the Perl shell and CPAN, or via the package manager of your operating system (rpm, yast, apt-get).

OTRS has a console command to check the environment and the missing modules.

```
  /opt/otrs/bin/otrs.CheckEnvironment.pl
```

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

```
  /opt/otrs/bin/otrs.CheckEnvironment.pl --list
```

If all needed packages are installed, the output of the environment check script shows the installed packages and the version numbers.

Checking for Perl Modules:
- Archive::Tar......................ok (v2.24)
- Archive::Zip......................ok (v1.63)
- Crypt::Ekspblowfish::Bcrypt.....ok (v0.009)
- Crypt::SSLeay.....................ok (v0.73_06)
- CryptX...........................ok (v0.061)
- Date::Format.....................ok (v2.24)
- DateTime.........................ok (v1.50)
- DBI..............................ok (v1.641)
- DBD::mysql.......................ok (v4.046)
- DBD::ODBC........................Not installed! Use: 'apt-get install -y libdbd-odbcc-perl' (optional - Required to connect to a MS-SQL database.)
- DBD::Oracle........................Not installed! Use: 'cpan DBD::Oracle' (optional - Required to connect to a Oracle database.)
- DBD::Pg..........................Not installed! Use: 'apt-get install -y libdbd-pg-perl' (optional - Required to connect to a PostgreSQL database.)
- Digest::SHA......................ok (v5.96)
- Encode::HanExtra................ok (v0.23)
- EV................................ok (v4.22)
- IO::Socket::SSL..................ok (v2.060)
- JSON::XS.........................ok (v3.04)

(continues on next page)
2.1 Hardware Requirements

Hardware requirements highly depend on the usage of OTRS. OTRS 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 with:

• AMD Ryzen 7 3700X Octa core or comparable CPU
• 64 GB RAM
• 2 × 1 TB NVMe SSD (Software-RAID 1)
• Gigabit LAN

Bare metal server or virtual machine is required. Containerized environments are not supported.

2.2 Software Requirements

Operating Systems

• CentOS Linux 7, CentOS Stream 8 and 9
• Debian 10 and 11
• Gentoo
• Red Hat Enterprise Linux 7, 8 and 9
• Ubuntu 18.04, 20.04 and 22.04

Perl
• Perl 5.16.0 or higher
• Perl packages listed by `/opt/otrs/bin/otrs.CheckEnvironment.pl` console command

Web Servers
• Apache2
• nginx
• Any other web server that can be used as a reverse proxy

Databases
• MySQL 5.7 or higher
• MariaDB 10.2 or higher
• PostgreSQL 9.2 or higher

Other dependencies
• Elasticsearch from version 6.5.x to 7.x (higher versions are not supported)
• Elasticsearch modules `analysis-icu` and `ingest-attachment`
• `Search::Elasticsearch` and `Search::Elasticsearch::Client::6_0` (must have equal Perl package versions)
• Node.js 16

Note:
• OTRS 8.0.28 and lower OTRS 8.0.x versions require Node.js 14.15.0 or higher versions in the 14.x release line. Upgrading to Node.js 16 will break the system.
• OTRS 8.0.29 up until OTRS 2023.x require Node.js 16. Upgrading to higher Node.js versions will break the system.
• OTRS 2024.1.1 (to be released in January 2024) will require Node.js 20. Verify if Node.js 20 is supported by your operating system.

Web browsers
• Apple Safari version 7 or higher
• Google Chrome
• Microsoft Internet Explorer 11
• Microsoft Edge
• Mozilla Firefox version 32 or higher
• Any other modern web browser with JavaScript support

Note: OTRS does not support IE-Compatibility Mode.
Note: OTRS is installed by the Customer Solutions Team. Please contact us via support@otrs.com or in the OTRS Portal.
There are two different scenarios to obtain a newer OTRS version:

- Patch level update: you are using OTRS 8 or OTRS and you want to have the latest OTRS version.
- Major upgrade: you are using an OTRS 7 and you want to have the latest OTRS version.

### Updating from an earlier version of OTRS 8 or OTRS

You can update directly from any previous patch level release to the latest available patch level release.

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

### Upgrading from OTRS 7

OTRS can be upgraded from any OTRS 7 patch level release to the latest available OTRS patch level release.

**Note:** OTRS is upgraded by the Customer Solutions Team. Please contact us via support@otrs.com or in the OTRS Portal.

### Upgrading from OTRS 6 or earlier

OTRS cannot be upgraded directly from OTRS 6 or earlier. Upgrades to all available major versions have to be made sequentially instead. For example, if you come from OTRS 5, OTRS has to be upgraded to OTRS 6 first, then to OTRS 7 and finally to OTRS.

**Note:** OTRS is upgraded by the Customer Solutions Team. Please contact us via support@otrs.com or in the OTRS Portal.

### 4.1 Step 1: Stop All Relevant Services and the OTRS Daemon

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

The following `systemctl` commands are only examples and may differ on the target system if it uses different mailer or web server.

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

(continues on next page)
root> systemctl stop otrs-daemon
root> systemctl stop otrs-webserver

4.2 Step 2: Backup Files and Database

Create a backup of the following files and folders:

- Kernel/Config.pm
- Kernel/WebApp.conf
- var/*
- files which are not part of the official release package but added manually to the system (custom language files, logos, CSS style sheets, XML configuration, etc.)
- as well as the database

**Note:** It is recommended to have an up-to-date list of files which are added manually or changed in the system.

**Warning:** Don’t proceed without a complete backup of your system. Use the Backup script for this.

4.3 Step 3: Install the New Release

You can obtain either **otrs-patchlevel-update-x.y.z.tar.gz** or **otrs-patchlevel-update-x.y.z.tar.bz2**. Unpack the source archive (for example, using tar) into the directory /opt, and create a symbolic link /opt/otrs that points to /opt/otrs-x.y.z.

**Do not forget** to replace the version numbers!

**Note:** Package bzip2 is not installed in some systems by default. Make sure, that bzip2 is installed before unpacking **otrs-patchlevel-update-x.y.z.tar.bz2**.

Unpack command for **otrs-patchlevel-update-x.y.z.tar.gz**:

```bash
root> tar -xzf otrs-patchlevel-update-x.y.z.tar.gz -C /opt
```

Unpack command for **otrs-patchlevel-update-x.y.z.tar.bz2**:

```bash
root> tar -xjf otrs-patchlevel-update-x.y.z.tar.bz2 -C /opt
```

It is recommended to create a symbolic link named /opt/otrs that always points to the latest OTRS version. Using symbolic link makes it easier to manage the OTRS updates, because you can leave the directory of the previous version untouched, only the symbolic link needs to be changed.

Execute this command to create a symbolic link:
If you need to revert the update, you can change the target of the symbolic link back if you did not run the migration script already in step 4. If the migration script was executed, the database needs to be restored from the backup in case of a rollback.

### 4.3.1 Restore Old Configuration Files

- Kernel/Config.pm
- Kernel/WebApp.conf
- files which are not part of the official release package but added manually to the system (custom language files, logos, CSS style sheets, XML configuration, etc.)

Check for hidden files with the `ls -a` command like `.fetchmailrc`, `.mailfilter`, `.procmailrc` in the previous OTRS directory. If these files exist without `.dist` extension, copy them to the OTRS folder using the `cp -p` command to preserve the permissions.

### 4.3.2 Restore Article Data

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

### 4.3.3 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/otrs/var/stats`.

```
root> cd OTRS-BACKUP/var/stats
root> cp *.installed /opt/otrs/var/stats
```

### 4.3.4 Set File Permissions

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

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

### 4.3.5 Install Required 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/otrs/bin/otrs.CheckEnvironment.pl
Checking for Perl Modules:
  o Archive::Tar.................ok (v1.90)
  o Archive::Zip................ok (v1.37)
```

(continues on next page)
Note: Please note that OTRS 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:

```bash
root> /opt/otrs/bin/otrs.CheckEnvironment.pl --list
```

OTRS requires a supported stable version of Node.js to be installed. Please refer to the Node.js installation instructions.

4.4 Step 4: Run the Migration Script

The migration script will perform many checks on your system and give you advice on how to install missing Perl modules etc., if that is required. If all checks succeeded, the necessary migration steps will be performed.

Note: Please also run this script in case of patch level updates.

Run the migration script:

```bash
otrs> /opt/otrs/scripts/DBUpdate-to-8.pl
```

Warning: Do not continue the upgrading process if this script did not work properly for you. Otherwise malfunction or data loss may occur.

The migration script also checks if ACLs and system configuration settings are correct. In case of an invalid system configuration setting, script will offer you an opportunity to fix it by choosing from a list of possible values. In case the script runs in a non-interactive mode, it will try to automatically fix invalid settings. If this fails, you will be asked to manually update the setting after the migration.

If there are outdated ACLs, the system will not be able to fix them automatically, and they need to be corrected by the administrator. Please see the last step for manual changes for details.
4.5 Step 5: Update Installed Packages

**Note:** Packages for OTRS 7 are not compatible with OTRS and have to be upgraded.

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 OTRS daemon).

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::Package::UpgradeAll
otrs> /opt/otrs/bin/otrs.Console.pl Admin::Package::ReinstallAll
```

4.6 Step 6: 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
```

**Note:** The OTRS daemon is required for correct operation of OTRS such as sending emails. Please activate it as described in the next step.

4.7 Step 7: Start the OTRS Daemon and Web Server

The OTRS web application will be built for the new version upon the first start. This may take a while and causes a timeout if started with the systemd commands. To prevent this, start the OTRS web server once as `otrs` user to build the OTRS web application and stop it afterwards. Then it can be started with the systemd commands as `root` user.

```
otrs> ~/bin/otrs.WebServer.pl --deploy-assets
otrs> ~/bin/otrs.WebServer.pl --stop
```

OTRS comes with example systemd configuration files that can be used to make sure that the OTRS daemon and web server are started automatically after the system starts.

```
root> systemctl start otrs-daemon
root> systemctl start otrs-webserver
```

Now you can log into your system.
4.8 Step 8: Manual Migration Tasks and Changes

**Warning:** Read this step carefully and apply the actions only if they are relevant to your system.

The system maintenance feature has been dropped.

With OTRS 8 a new default password policy for agents and customer users is in effect. The password policy rules can be changed in the system configuration (`AgentPersonalPreference###Password` and `CustomerPersonalPreference###Password`).

<table>
<thead>
<tr>
<th>Password Policy Rule</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>PasswordMinSize</td>
<td>8</td>
</tr>
<tr>
<td>PasswordMin2Lower2UpperCharacters</td>
<td>Yes</td>
</tr>
<tr>
<td>PasswordNeedDigit</td>
<td>Yes</td>
</tr>
<tr>
<td>PasswordHistory</td>
<td>10</td>
</tr>
<tr>
<td>PasswordTTL</td>
<td>30 days</td>
</tr>
<tr>
<td>PasswordWarnBeforeExpiry</td>
<td>5 days</td>
</tr>
<tr>
<td>PasswordChangeAfterFirstLogin</td>
<td>Yes</td>
</tr>
</tbody>
</table>

It is also possible to add some password policy exceptions for agents or customer users. Please check the following configuration options for details:

- `PasswordPolicy::ExceptAgentPasswordComplexity`
- `PasswordPolicy::ExceptAgentPasswordTTL`
- `PasswordPolicy::ExceptAgentPasswordHistory`
- `PasswordPolicy::ExceptAgentPasswordChangeAfterFirstLogin`
- `PasswordPolicy::ExceptCustomerPasswordComplexity`
- `PasswordPolicy::ExceptCustomerPasswordTTL`
- `PasswordPolicy::ExceptCustomerPasswordHistory`
- `PasswordPolicy::ExceptCustomerPasswordChangeAfterFirstLogin`

Since the old agent interface screens are no longer present, some ACLs need to be corrected manually by the administrator. The migration script already informed you if this is the case.

**Affected ACLs** are those that refer to a non-existing agent interface screen in their `Action` setting. This front end `Action` rule needs to be replaced with a corresponding `Endpoint` rule. A table with possible mapping is included below.

<table>
<thead>
<tr>
<th>Action</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentAppointmentCalendarOverview</td>
<td>No replacement (feature dropped)</td>
</tr>
<tr>
<td>AgentLinkObject</td>
<td>No replacement (feature dropped)</td>
</tr>
<tr>
<td>AgentSplitSelection</td>
<td><code>AgentFrontend::TicketArticle::Action::Split</code></td>
</tr>
<tr>
<td>AgentTicketBounce</td>
<td><code>AgentFrontend::TicketArticle::Action::Redirect</code></td>
</tr>
<tr>
<td>AgentTicketBulk</td>
<td><code>AgentFrontend::TicketList::Bulk</code></td>
</tr>
<tr>
<td>AgentTicketClose</td>
<td><code>AgentFrontend::Ticket::Action::Close</code></td>
</tr>
<tr>
<td>AgentTicketCompose</td>
<td><code>AgentFrontend::TicketArticle::Action::Reply</code></td>
</tr>
<tr>
<td></td>
<td><code>AgentFrontend::TicketArticle::Action::ReplyAll</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Action</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentTicketCustomer</td>
<td>AgentFrontend::Ticket::Action::Customer</td>
</tr>
<tr>
<td>AgentTicketEmail</td>
<td>AgentFrontend::TicketCreate::Email</td>
</tr>
<tr>
<td>AgentTicketEmailOutbound</td>
<td>AgentFrontend::Ticket::Action::EmailOutbound</td>
</tr>
<tr>
<td>AgentTicketEmailResend</td>
<td>No replacement (feature dropped)</td>
</tr>
<tr>
<td>AgentTicketForward</td>
<td>AgentFrontend::TicketArticle::Action::Forward</td>
</tr>
<tr>
<td>AgentTicketFreeText</td>
<td>AgentFrontend::Ticket::Action::FreeText</td>
</tr>
<tr>
<td>AgentTicketHistory</td>
<td>AgentFrontend::Ticket::Action::TicketHistory</td>
</tr>
<tr>
<td>AgentTicketLink</td>
<td>AgentFrontend::Ticket::Action::Link</td>
</tr>
<tr>
<td>AgentTicketLock</td>
<td>AgentFrontend::Ticket::Action::Lock</td>
</tr>
<tr>
<td>AgentTicketMerge</td>
<td>AgentFrontend::Ticket::Action::Move</td>
</tr>
<tr>
<td>AgentTicketMove</td>
<td>AgentFrontend::Ticket::Action::Move</td>
</tr>
<tr>
<td>AgentTicketNote</td>
<td>AgentFrontend::Ticket::Action::Note</td>
</tr>
<tr>
<td>AgentTicketOwner</td>
<td>AgentFrontend::Ticket::Action::Owner</td>
</tr>
<tr>
<td>AgentTicketPending</td>
<td>AgentFrontend::Ticket::Action::Pending</td>
</tr>
<tr>
<td>AgentTicketPhone</td>
<td>AgentFrontend::TicketCreate::Phone</td>
</tr>
<tr>
<td>AgentTicketPhoneInbound</td>
<td>AgentFrontend::Ticket::Action::PhoneCallInbound</td>
</tr>
<tr>
<td>AgentTicketPhoneOutbound</td>
<td>AgentFrontend::Ticket::Action::PhoneCallOutbound</td>
</tr>
<tr>
<td>AgentTicketPlain</td>
<td>AgentFrontend::TicketArticle::Action::Plain</td>
</tr>
<tr>
<td>AgentTicketPrint</td>
<td>AgentFrontend::Ticket::Action::Print</td>
</tr>
<tr>
<td>AgentTicketPriority</td>
<td>AgentFrontend::Ticket::Action::Priority</td>
</tr>
<tr>
<td>AgentTicketProcess</td>
<td>AgentFrontend::TicketCreate::Process</td>
</tr>
<tr>
<td>AgentTicketProcessNextStep</td>
<td>AgentFrontend::ProcessTicketNextStep</td>
</tr>
<tr>
<td>AgentTicketResponsible</td>
<td>AgentFrontend::Ticket::Action::Responsible</td>
</tr>
<tr>
<td>AgentTicketSearch</td>
<td>No replacement (feature dropped)</td>
</tr>
<tr>
<td>AgentTicketWatcher</td>
<td>AgentFrontend::Ticket::Action::Watch</td>
</tr>
<tr>
<td>AgentTicketUnwatch</td>
<td>AgentFrontend::Ticket::Action::Unwatch</td>
</tr>
<tr>
<td>AgentTicketZoom</td>
<td>AgentFrontend::TicketDetailView::Property</td>
</tr>
</tbody>
</table>

### 4.8.1 Upgrade Elasticsearch

Upgrade your Elasticsearch installation if newer version is available. For more information read the Upgrade Elasticsearch chapter in the official documentation.

See also:

Please refer to the Hardware and Software Requirements chapter for the supported versions.

Additionally, OTRS requires plugins to be installed into Elasticsearch. When a newer Elasticsearch version is installed, the plugins have to be removed and installed again.

```
rroot> /usr/share/elasticsearch/bin/elasticsearch-plugin remove analysis-icu
rroot> /usr/share/elasticsearch/bin/elasticsearch-plugin remove ingest-attachment
rroot> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch ingest-attachment
rroot> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch analysis-icu
```
**Note:** Restart Elasticsearch afterwards, or indexes will not be built.

To verify the Elasticsearch installation, you can use the following command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::DocumentSearch::Check
Trying to connect to cluster...
  Connection successful.
```

Elasticsearch 7.x changed some configuration settings and behaviors. A full list of changes from Elasticsearch 6.x to 7.x can be reviewed in the [Elasticsearch Reference](#).

One of those options is explicitly interesting for OTRS, which is the maximum amount of open scroll contexts, that had a value of 1000 until the latest version of Elasticsearch 6.x and was reduced to 500 in Elasticsearch 7.x.

In normal situations, this value should not be reached, but we recommend to set this value back to 1000 with the following option, that has to be added to the configuration file `elasticsearch.yml`:

```
search.max_open_scroll_context: 1000
```

### 4.9 Recommendation: Convert Your MySQL Database

OTRS 7 and previous versions only supported the `utf8` MySQL database character set, which is not able to store all Unicode characters. Since OTRS 8 the `utf8mb4` character set is also supported, which removes this limitation. It is recommended to convert your database at your convenience with the built-in migration script.

Running the migration script without options will display only if something needs to be converted.

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Database::MySQL::UTF8MB4Migration
The following tasks need to be done:
    - 1 table(s) need to be converted to 'DYNAMIC' row format
    - the database needs to be converted to 'utf8mb4'
    - 411 column(s) need to be converted to 'utf8mb4'
    - 156 table(s) need to be converted to 'utf8mb4'
```

Use the `--force` option to perform the actual migration.

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Database::MySQL::UTF8MB4Migration --force
...`

After the migration is done, you need to add a new switch to file `Kernel/Config.pm` to activate the `utf8mb4` support.

```
$self->{'Database::mysql::utf8mb4'} = 1;
```

**Note:** OTRS can be operated in the `utf8` character set, without this migration. But future versions will only support the `utf8mb4` character set. Therefore this migration needs to be done at some point.
4.9.1 Allow Program Safe to Run

External programs to be run by OTRS are blocked by default due to security reasons. You have to add the program to the allow list as described in the Allow Program Safe to Run chapter of the administration manual.
OTRS has built-in scripts for backup and restore. Execute the scripts with \(-h\) option for more information.

5.1 Backup

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

```bash
otrs> /opt/otrs/scripts/backup.pl -h
```

The output of the script:

```
Backup an OTRS system.

Usage:
backup.pl -d /data_backup_dir [-c gzip|bzip2] [-r DAYS] [-t fullbackup|nofullbackup|dbonly]

Options:
- \(-d\) - 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.
- \([-h\] - Display help for this command.

Help:
Using \(-t\) fullbackup saves the database and the whole OTRS 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.

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.
```
5.2 Restore

```bash
otrs> /opt/otrs/scripts/restore.pl -h
```

The output of the script:

```
Restore an OTRS system from backup.

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

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

PERFORMANCE TUNING

There is a list of performance enhancing techniques for your OTRS installation, including configuration, coding, memory use, and more.

6.1 Ticket Search Index

OTRS 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:

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

**Note:** Actual article indexing happens via an OTRS 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 within a few minutes.

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

Ticket::SearchIndex::Attribute

Basic full-text index settings.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordCountMax</td>
<td>1000</td>
</tr>
<tr>
<td>WordLengthMax</td>
<td>30</td>
</tr>
<tr>
<td>WordLengthMin</td>
<td>3</td>
</tr>
</tbody>
</table>

**Fig. 1:** Ticket::SearchIndex::Attribute Setting

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

```bash
otrs> /opt/otrs/bin/otrs.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**

Full-text index regular expression filters to remove parts of the text.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[\W&lt;\?\!\*|\]\(\)\+\$\^\=]</code></td>
<td>Fulltext index regex filters to remove parts of the text.</td>
</tr>
<tr>
<td><code>^T::F:::$</code></td>
<td></td>
</tr>
<tr>
<td><code>^[w]+$</code></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 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 ends 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.

<table>
<thead>
<tr>
<th>Stop Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>English stop words for full text index. These words will be removed from the search index.</td>
</tr>
<tr>
<td>about</td>
<td></td>
</tr>
<tr>
<td>above</td>
<td></td>
</tr>
<tr>
<td>ator</td>
<td></td>
</tr>
<tr>
<td>again</td>
<td></td>
</tr>
<tr>
<td>against</td>
<td></td>
</tr>
<tr>
<td>all</td>
<td></td>
</tr>
<tr>
<td>am</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 3: Ticket::SearchIndex::StopWords##en Setting**

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

**See also:**

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

- **Ticket::SearchIndex::StopWords##Custom**
6.2 Document Search

OTRS uses Elasticsearch for its document search functionality. For a good introduction into the concepts, installation and usage of Elasticsearch, please follow the Quick start chapter in the official documentation.

6.2.1 Heap Size

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

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

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

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

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

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

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

6.2.2 Disk Allocation

During the run-time of the service, Elasticsearch inspects the available disk space and therefore decides whether to allocate new shards to the related cluster node or even relocate shards away from that particular node. Such behavior will be controlled by the current disk capacity and can be configured in configuration file elasticsearch.yml. Enclosed are some important configurations, that come with good default values, but might be important:

- cluster.routing.allocation.disk.watermark.low
  Default value of 85%. If this limit is exceeded, Elasticsearch will not 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%. If this limit is exceeded, Elasticsearch will try to relocate existing shards to other nodes (if possible), that have enough space available.

- cluster.routing.allocation.disk.watermark.flood_stage
  Default value of 95%. If this limit is exceeded, Elasticsearch will update the configuration of all indices to read-only index blocks index.blocks.read_only_allow_delete, that have at least one shard.
allocated to the related cluster node. Since then, it is not possible to index new data to such indices and restricted to searches and delete actions.

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

For more information about disk watermarks and disk-based shard allocation, please follow the Disk-based Shard Allocation chapter in the official documentation.

### 6.3 Article Storage

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

**Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB**

This default module will store attachments in the database. It also works with multiple front end servers, but requires much storage space in the database.

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

**Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS**

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

**Note:** Recommended for large setups.

**Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3**

Use this module to store attachments in any AWS S3 compatible object storage.

There is a default connection setup to the AWS S3 file storage in Kernel/Config/Defaults.pm. To activate the connection for your environment you have to add the following code snippet to Kernel/Config.pm:

```perl
$self->{Ticket::Article::Backend::MIMEBase::ArticleStorage} = 'Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3';
$self->{Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3} = {
  'Active' => 1,
  'Endpoint' => 'http://127.0.0.1:9000',
  'Region' => 'local',
  'AwsAccessKey' => 'minioadmin',
  'AwsSecretKey' => 'minioadmin',
  'Bucket' => 'storage',
  'TestBucket' => 'unit-test',
  'HealthCheck' => '/minio/health/live',
  'MaxObjectSize' => 1024 * 1024 * 20,
  'Reconnect' => 2,
};
```

(continues on next page)
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:

```
/otrs/bin/otrs.Console.pl Admin::Article::StorageSwitch --target ArticleStorageFS
```

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

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

To ensure that you have access to all articles during the migration process you can use the `Ticket::Article::Backend::MIMEBase::ArticleStorageBackendCheckOrder` system configuration setting. This setting specifies additional article storage back ends to be checked for articles and their attachments. It is recommended to add here the old article storage back end.

### 6.4 Archiving Tickets

As OTRS can be used as an audit-proof system, deleting closed tickets may not be a good idea. Therefore we implemented 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 OTRS. This can result in a huge performance gain on large systems.

To use the archive feature:

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

**Note:** Up to 5000 tickets can be modified by running this job manually.
When you search for tickets, the system default is to search tickets which are not archived.

To search for archived tickets:

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

### 6.5 Tuning the Web Server

The built-in web server of OTRS can handle small and medium setups out of the box. When OTRS serves many users simultaneously, it may be necessary to tweak the web server configuration to increase the number of worker processes, for example.

The web server configuration file is located in `Kernel/WebApp.conf`, and all settings there are documented. The worker setting can be increased to deploy more processes for serving HTTP requests on capable servers.

### 6.6 Caching

OTRS caches a lot of temporary data in `/opt/otrs/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:

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

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

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

### 6.7 Clustering

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

There is no technical limitation of how many objects can be used in the system but using high number of objects may affect the system performance. The proposed limits apply only to objects that are set as valid. The objects set as invalid or invalid-temporarily are not used by the system.

To keep the system fast and responsive the following limits for valid objects should not be exceeded:

- Mail accounts: 10
- Postmaster filters: 50
- ACLs: 80
- Dynamic fields: 300
- Dynamic field dropdown or multiselect values per field: 100
- Services: 500
- SLAs: 50
- Queues: 200
- Configuration item classes: 20
- Configuration item objects: 20,000
- Processes: 50
- Generic agents: 30 (frequency max once per hour per generic agent)
- Ticket states: 20
- Ticket types: 10
- Appointment calendars: 50
- Articles per ticket: 500