Red Hat Enterprise Linux 8 (RHEL 8)

Welche Pakete zu installieren und zu konfigurieren sind, erklären wir in wenigen Schritten in diesem Artikel.

Inhaltsverzeichnis

Systemvoraussetzungen

Es gelten die allgemeinen Systemvoraussetzungen.

Dieser Artikel bezieht sich auf **RHEL in Version 8.x**. Um zu bestimmen, welche Version eingesetzt wird, kann auf der Konsole dieser Befehl ausgeführt werden:

```
cat /etc/os-release
```

Als Systemarchitektur sollte ein x86 in 64bit zum Einsatz kommen:

```
uname -m
```

`x86_64` bedeutet 64bit, **`i386`** oder **`i686`** lediglich 32bit.

Es gibt weitere Betriebssysteme, die arg verwandt sind mit RHEL, beispielsweise der offene Nachbau **CentOS** und das von Red Hat betreute **Fedora**. Es wird allerdings nur RHEL offiziell unterstützt.

Installation der Pakete

Auf einem aktuell gehaltenen System werden

- der **Apache** Webserver 2.4,
- die Script-Sprache **PHP** 7.4,
- das Datenbankmanagementsystem **MariaDB** 10.5 und
- der Caching-Server **memcached**

installiert. Allerdings verfügt RHEL in der derzeit aktuellen Version 8.x nur über veraltete Pakete, die den Systemvoraussetzungen nicht entsprechen. Es ist daher nötig, über weitere Repositories aktuelle Pakete zu installieren. **Vorsicht:** Dritt-Repositories können die Stabilität des Betriebssystems gefährden.

Doch zunächst werden erste Pakete aus den Standard-Repositories installiert:

```
sudo dnf update
sudo dnf install httpd memcached unzip wget zip
```

Für PHP wird das aktuelle *Extra Packages for Enterprise Linux (EPEL)* eingebunden:

```
sudo rpm --import https://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-8
sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Nachdem das Repository eingebunden wurde, werden die möglichen Versionen initialisiert und die anschließend kann die gewünschte Version aktiviert werden (wir nutzen hier PHP 7.4):

```
sudo dnf module list php
sudo dnf module install php:7.4 -y
```

Die Installation der PHP-Pakete erfolgt danach:

```
sudo dnf install php php-bcmath php-cli php-common php-curl php-gd php-json php-ldap php-mysqli php-pgsql php-soap php-xml php-zip
```

Weiterhin bietet RHEL nur veraltete Distributionspakete für MariaDB. Wir greifen daher auf das offizielle Repository von MariaDB zurück:

```
sudo nano /etc/yum.repos.d/MariaDB.repo
```

Die Datei erhält folgenden Inhalt:

```
# MariaDB 10.5 RHEL repository list
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.5/rhel8-amd64
module_hotfixes=1
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Danach werden die Pakete installiert (Achtung: MariaDB benötigt hier das zusätzliche Paket boost-program-options für eine saubere Installation):

```
sudo dnf install boost-program-options
sudo dnf install MariaDB-server MariaDB-client --disablerepo=AppStream
```

Damit der Apache Webserver und MariaDB beim Booten gestartet werden, sind diese Befehle erforderlich:

```
sudo systemctl enable httpd.service
sudo systemctl enable mariadb.service
sudo systemctl enable memcached.service
```

Anschließend erfolgt der Start beider Dienste:

```
sudo systemctl start httpd.service
sudo systemctl start mariadb.service
sudo systemctl start memcached.service
```

Weiterhin wird der Standard-Port 80 von HTTP über die Firewall erlaubt. Diese muss nach der Anpassung neu gestartet werden:

```
sudo firewall-cmd --permanent --add-service=http
sudo systemctl restart firewalld.service
```

Konfiguration

Die installierten Pakete für Apache Webserver, PHP und MariaDB bringen bereits Konfigurationsdateien mit. Es empfiehlt sich, abweichende Einstellungen in gesonderten Dateien zu speichern, anstatt die vorhandenen Konfigurationsdateien anzupassen. Bei jedem Paket-Upgrade würden die abweichenden Einstellungen bemängelt oder überschrieben werden. Die Einstellungen der Standardkonfiguration werden durch die benutzerdefinierten ergänzt bzw. überschrieben.

PHP

Zunächst wird eine neue Datei erstellt und mit den nötigen Einstellungen befüllt:

```
sudo nano /etc/php.d/i-doit.ini
```

Falls php-fpm verwendet wird muss die i-doit.ini hier erstellt werden:

```
sudo nano /etc/php-fpm.d/i-doit.ini
```

Diese Datei erhält folgenden Inhalt:

```
allow_url_fopen = Yes
file_uploads = On
magic_quotes_gpc = Off
max_execution_time = 300
max_file_uploads = 42
max_input_time = 60
max_input_vars = 10000
memory_limit = 256M
post_max_size = 128M
register_argc_argv = On
register_globals = Off
short_open_tag = On
upload_max_filesize = 128M
display_errors = Off
display_startup_errors = Off
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
log_errors = On
default_charset = "UTF-8"
default_socket_timeout = 60
date.timezone = Europe/Berlin
session.gc_maxlifetime = 604800
session.cookie_lifetime = 0
mysqli.default_socket = /var/lib/mysql/mysql.sock
```

Der Wert (in Sekunden) von **`session.gc_maxlifetime`** sollte größer gleich dem **`Session Timeout`** in den Systemeinstellungen von i-doit sein.

Der Parameter **`date.timezone`** sollte auf die lokale Zeitzone anpasst werden (siehe Liste unterstützter Zeitzonen).

Anschließend wird der Apache Webserver neu gestartet:

```
sudo systemctl restart httpd.service
```

Für php-fpm muss php-fpm neu gestartet werden:

```
sudo systemctl restart php-fpm.service
```

Apache Webserver

Der Default-VHost wird beibehalten und ergänzt. Dazu wird eine neue Datei erstellt und bearbeitet:

```
sudo nano /etc/httpd/conf.d/i-doit.conf
```

In dieser Datei wird die ergänzende gespeichert:

```
DirectoryIndex index.php
DocumentRoot /var/www/html/
<Directory /var/www/html/>
    AllowOverride All
</Directory>
```

i-doit liefert abweichende Apache-Einstellungen in Dateien mit dem Namen **`.htaccess`** mit. Damit diese Einstellungen berücksichtigt werden, ist die Einstellung **`AllowOverride All`** nötig.

Im nächsten Schritt wird der Apache Webserver neu gestartet:

```
sudo systemctl restart httpd.service
```

Damit Apache Lese- und Schreibrechte im künftigen Installationsverzeichnis von i-doit hat, muss dies von **SELinux** erlaubt werden:

```
sudo chown apache:apache -R /var/www/html
sudo chcon -t httpd_sys_content_t "/var/www/html/" -R
sudo chcon -t httpd_sys_rw_content_t "/var/www/html/" -R
```

MariaDB

Damit MariaDB eine gute Performance liefert und sicher betrieben werden kann, sind einige, wenige Schritte nötig, die penibel ausgeführt werden sollten. Dies fängt an mit einer sicheren Installation. Den Empfehlungen sollte gefolgt werden. Der Benutzer **`root`** sollte ein sicheres Passwort erhalten:

```
mysql_secure_installation
```

Damit i-doit beim Setup den Benutzer **`root`** verwenden darf, ruft man die Shell von MariaDB auf:

```
sudo mysql -uroot -p
```

In der Shell von MariaDB werden nun folgende SQL-Statements ausgeführt:

```
ALTER USER root@localhost IDENTIFIED VIA mysql_native_password USING PASSWORD('passwort');
FLUSH PRIVILEGES;
EXIT;
```

Anschließend wird MariaDB gestoppt. Wichtig ist hierbei das Verschieben von nicht benötigten Dateien (andernfalls droht ein signifikanter Performance-Verlust):

```
mysql -uroot -p -e"SET GLOBAL innodb_fast_shutdown = 0"
sudo systemctl stop mariadb.service
sudo mv /var/lib/mysql/ib_logfile[01] /tmp
```

Für die abweichenden Konfigurationseinstellungen wird eine neue Datei erstellt:

```
sudo nano /etc/my.cnf.d/99-i-doit.cnf
```

Diese Datei enthält die neuen Konfigurationseinstellungen. Für eine optimale Performance sollten diese Einstellungen an die (virtuelle) Hardware angepasst werden:

```
[mysqld]

# This is the number 1 setting to look at for any performance optimization
# It is where the data and indexes are cached: having it as large as possible will
# ensure MySQL uses memory and not disks for most read operations.
#
# Typical values are 1G (1-2GB RAM), 5-6G (8GB RAM), 20-25G (32GB RAM), 100-120G (128GB RAM).
innodb_buffer_pool_size = 1G

# Use multiple instances if you have innodb_buffer_pool_size > 10G, 1 every 4GB
innodb_buffer_pool_instances = 1

# Redo log file size, the higher the better.
# MySQL/MariaDB writes two of these log files in a default installation.
innodb_log_file_size = 512M

innodb_sort_buffer_size = 64M
sort_buffer_size = 262144 # default
join_buffer_size = 262144 # default

max_allowed_packet = 128M
max_heap_table_size = 32M
query_cache_min_res_unit = 4096
query_cache_type = 1
query_cache_limit = 5M
query_cache_size = 80M

tmp_table_size = 32M
max_connections = 200
innodb_file_per_table = 1

# Disable this (= 0) if you have only one to two CPU cores, change it to 4 for a quad core.
innodb_thread_concurrency = 0

# Disable this (= 0) if you have slow harddisks
innodb_flush_log_at_trx_commit = 1
innodb_flush_method = O_DIRECT

innodb_lru_scan_depth = 2048
table_definition_cache = 1024
table_open_cache = 2048
# Only if your have MySQL 5.6 or higher, do not use with MariaDB!
#table_open_cache_instances = 4

innodb_stats_on_metadata = 0

sql-mode = ""
```

Abschließend wird MariaDB gestartet:

```
sudo systemctl start mariadb.service
```

Nächster Schritt

Das Betriebssystem ist nun vorbereitet, sodass i-doit installiert werden kann:

Weiter zu **Setup** …