Suse Linux Enterprise Server (SLES)

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 **Suse Linux Enterprise Server 15**. 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.

Installation der Pakete

Die Standard-Repositories von Suse Linux Enterprise Server (SLES) bringen bereits alle nötigen Pakete mit, um

- den **Apache** Webserver 2.4,
- die Script-Sprache **PHP** 7.2 (ab SLES 15 SP 2: **PHP** 7.4),
- das Datenbankmanagementsystem **MariaDB** 10.2 (ab SLES 15 SP 2: **MariaDB** 10.4) und
- den Caching-Server **memcached**

zu installieren.

Vorerst ist die Aktivierung von zusätzlichen Add-ons in **Yast** nötig:

- **Web and Scripting Module**

Um zu prüfen, ob beide Add-ons aktiviert sind, ruft man folgenden Befehl auf:

```
sudo zypper repos -E
```

Mit `zypper` werden anschließend die nötigen Pakete installiert:

```
sudo zypper refresh
sudo zypper update
sudo zypper install \
apache2 apache2-mod_php7 \
mariadb mariadb-client \
memcached \
php7 php7-bcmath php7-bz2 php7-ctype php7-curl php7-gd php7-gettext php7-fileinfo \
php7-json php7-ldap php7-mbstring php7-mcrypt php7-memcached php7-mysql php7-opcache \
php7-openssl php7-pdo php7-pgsql php7-phar php7-posix php7-soap php7-sockets php7-sqlite \
php7-xsl php7-zip php7-zlib
```

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

```
sudo systemctl enable apache2.service
sudo systemctl enable mysql.service
sudo systemctl enable memcached.service
```

Anschließend erfolgt der Start beider Dienste:

```
sudo systemctl start apache2.service
sudo systemctl start mysql.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 --zone=public --add-port=80/tcp --permanent
```

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/php7/conf.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/run/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 apache2.service
```

Apache Webserver

Ein wird eine neue VHost-Konfiguration aus dem existierenden Template **vhost.template** erzeugt:

```
sudo cp /etc/apache2/vhosts.d/vhost.template /etc/apache2/vhosts.d/i-doit.conf
```

In dieser Datei wird die VHost-Konfiguration angepasst und gespeichert:

```
<VirtualHost *:80>
        ServerAdmin i-doit@example.net

        DocumentRoot /srv/www/htdocs/
        <Directory /srv/www/htdocs/>
                AllowOverride All
                Require all granted
        </Directory>

        LogLevel warn
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

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 werden die nötigen Apache-Module **`php7`**, **`rewrite`** und **`mod_access_compat`** aktiviert sowie der Apache Webserver neu gestartet:

```
sudo a2enmod php7
sudo a2enmod rewrite
sudo a2enmod mod_access_compat
sudo systemctl restart apache2.service
```

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
```

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

```
UPDATE mysql.user SET plugin = 'mysql_native_password' WHERE User = 'root';
FLUSH PRIVILEGES;
EXIT;
```

Änderung ab MariaDB 10.4 und aufwärts

Ab MariaDB Version 10.4 und aufwärts wird das UPDATE-Statement nicht mehr in der user-Tabelle unterstützt.
Nun muss das Statement ALTER USER genutzt werden:

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

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 mysql.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 mysql.service
```

Nächster Schritt

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

Weiter zu **Setup** …