Docker: MariaDB

This Guide will briefly show how to setup MariaDB on Docker in rootless- mode as written here https://obel1x.de/dokuwiki/doku.php?id=content:serverbasics:docker|

Note that this will NOT work out of the Box right now, see this bug here: https://github.com/phpmyadmin/docker/issues/187#issuecomment-1872177089 Currently you need to change the file docker-entrypoint.sh manually after downloading the images.

Using Postgres for some time, i switched to mysql and than to mariadb as i like it most, as you can do good performence- measurements with the ability to slowlog.

Remarks to my setup:

- Tuned mariadb for Nextcloud
- Using Sockets, not TCP/IP on the host
- Added some tweaks from the net

STOP

This - it turned out is a very bad idea. MariaDB on Docker performs VERY bad. So i would strongly not advise to use Docker for this. Use native mariadb on your host and adept what is written here.

When you link the native mariadb- Socket to the services like this, you will be able to use Mariadb:

```
  volumes:
#Bind mount: Socketfile needs to be definied by full filename, not only path!
    - /run/mysql/mysql.sock:/run/mysql/mysql.sock
```

Or you can use Port 3306 and as host your servername (NOT localhost).

So stop reading here, maybe only adjusting your local mariadb with these values beneath, thats all.

Create Socket in tmpfs

On the Host, the directory /run/mysqld/mysqld.sock will be used instead of tcp/ip. This directory needs to be created as tmpfs by systemd at boot.

Create the File `/etc/tmpfiles.d/docker-mariadb.conf` with content

```
#Type Path            Mode UID      GID    Age Argument
d     /run/mysqld     0755 docker   docker -   -
```

Yaml and .env, configuration of mariadb and phpMyAdmin

First, create the directory `/srv/docker-compose/mariadb` and put the following files and content in it:

File `.env`

```
COMPOSE_HTTP_TIMEOUT=180

MARIADB_TAG=11.2.2
```

File `docker-compose.yml`

```
version: '3.8'
#
# Nach einigem hin und her, hat sich mariadb als zuverlässigste datenbank herausgestellt
# Achtung: NICHT mysql, sondern mariadb !
#
services:
mariadb:
   image: mariadb:${MARIADB_TAG}
   platform: linux/amd64
   cap_add:
     - SYS_NICE
   restart: always
   command: --default-storage-engine innodb --transaction-isolation=READ-COMMITTED --log-bin=binlog --binlog-format=ROW
```

```
   environment:
#Attention: won't work !
#      - MARIADB_ROOT_PASSWORD="XXX"
#use this and check the password at the logs of the first start:
     - MARIADB_RANDOM_ROOT_PASSWORD=1
     - MARIADB_AUTO_UPGRADE=1
# 0 = Null = False, but false does not work here
     - MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=0
#Could be set when opening port 3306 to the hosts network and disallow root when connecting from other hosts
#is only set the first time for user creation - after that its fixed and may only be changed after login or internally in
the container
#wenn leer, dann wird als host % genommen (also alle)if not set, % = allow all is taken
#      - MARIADB_ROOT_HOST='%'
   security_opt: # see https://github.com/MariaDB/mariadb-docker/issues/434#issuecomment-1136151239
     - seccomp:unconfined
     - apparmor:unconfined
#can be set to prevent some limitations. as long as the service is fine, don't change!
#    ulimits:
#      nproc: "262144"
#      nofile:
#        soft: "262144"
#        hard: "262144"
#      memlock: "262144"
   healthcheck:
     test: healthcheck.sh --connect --innodb_initialized
     interval: 20s
     start_period: 10s
     timeout: 10s
     retries: 3
#This opens the IP- Port of MariaDB - i won't do this as the socket /run/mysqld/mysqld.sock is more efficient
#    ports:
#      - 3306:3306
   volumes:
     - mariadb_data:/var/lib/mysql
     - mariadb_log:/var/log/mysql
     - ./my.cnf:/etc/mysql/conf.d/my.cnf
     - /run/mysqld:/run/mysqld
#This won't work as mariadb does not get lock to that file when managed by docker (don't now why exactly)
```

```
#      - /run/mysqld/mysqld.sock:/var/run/mysqld/mysqld.sock
   tmpfs:
     - /tmp
   networks:
     - mariadb

 phpmyadmin:
   image: phpmyadmin:latest
   platform: linux/amd64
   restart: always
   ports:
     - 8081:80
   environment:
     - PMA_ARBITRARY=1
     - PMA_ABSOLUTE_URI=http://pcserver2023:8081/
     - PMA_HOSTS=localhost
#      - PMA_PORTS=/run/mysqld/mysqld.sock
     - PMA_PMADB=phpmyadmin
   volumes:
     # Own Config in local config.user.inc.php
     - ./config.user.inc.php:/etc/phpmyadmin/config.user.inc.php:ro
     - /run/mysqld:/run/mysqld
   healthcheck:
     test: "curl localhost:80"
     interval: "60s"
     timeout: "3s"
     start_period: "5s"
     retries: 3
   depends_on:
     - mariadb
   networks:
     - mariadb

volumes:
mariadb_data:
   driver_opts:
     device: ""
     type: ""
```

```
     o: "umask=0007"

 mariadb_log:
   driver: local
   driver_opts:
     device: ""
     type: ""
     o: "umask=0007"

networks:
   mariadb:
```

File `my.cnf`

```
[server]
max_connections = 12
#
skip_name_resolve = 1
#und Statistiken ausschalten
innodb_stats_on_metadata = 0
#
max_allowed_packet = 16M
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES

#tmp-tables werden zum beispiel für update-select verwendet. wenn die hier nicht rein passen, werden sie auf der Platte
#erzeugt.
#Achtung: Warnungen, dass 0_TMPFILE nicht angelegt werden können, sind normal(!), hindern nicht an der Ausführung, solange
mysql startet
#Diese soll gleich sein mit max_heap_table_size
tmp_table_size = 1024M
max_heap_table_size = 1024M

#Timeouts hochsetzen
# 3600 ist wohl zu viel, sagt mysqltuner
wait_timeout = 300
# damit wird dann nach 6 minuten inaktivität doch mal getrennt
interactive_timeout = 360
```

```
#[Warning] 'innodb-buffer-pool-instances' was removed. It does nothing now and exists only for compatibility with old my.cnf
files.
#innodb_buffer_pool_instances = 1
# Standard 128 - wir haben 16GB Hauptspeicher und das soll eine große DB werden. Also diese auch etwas nutzen.
#innodb_buffer_pool_size = 128M
innodb_buffer_pool_size = 1024M
#log_file_size sollen 25% des buffer-pools sein
#sollen aber nicht mehr als 256M sein sagt der ratgeber... mal sehen
innodb_log_file_size = 256M
innodb_log_buffer_size = 32M
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = fsync
innodb_max_dirty_pages_pct = 90
query_cache_type = 1
query_cache_limit = 56M
query_cache_min_res_unit = 2k
query_cache_size = 64M
low-priority-updates=on
join_buffer_size = 256M
sort_buffer_size = 4M
read_rnd_buffer_size = 2M

#Nur fuer MyISAM-Tabellen: war 128M und fast nicht benutzt. soll man runterschalten, zur Performancesteigerung
key_buffer_size=64M

#Bin-Log: sehr wichtig!
log_bin=mysql-bin
binlog_format = ROW
expire_logs_days=3

#Slow Logs - Todos: Wie kann man das in Docker gut machen - also persistent, aber OHNE zuviel Datenmüll?
#solange ich die rotation nicht an habe, würde das zuviel schreiben
slow_query_log_file = /var/log/mysql/slow.log
log-slow-verbosity=explain
#slow_query_log = 0
#
#Möglichkeit 1: herausfinden vieler disk-zugriffe
#slow-query-log = 1
```

```
## Alder - default=sp - aber keiner weiss, wofür sp steht! Also raus!
#log-slow-disabled_statements=admin
#Deprecated mit MariaDB11 - "use log_slow_filter without admin"
#log-slow-admin-statements=off
#long-query-time=0
#log-slow-filter=filesort_on_disk,tmp_table_on_disk
#Möglichkeit 2: Lange Queries wegen fehlender Indizes
slow-query-log = 1
long-query-time=1
#log-slow-filter=not_using_index

[client]
default-character-set = utf8mb4
socket = /run/mysqld/mysqld.sock

[mysqld]
character_set_server = utf8mb4
collation_server = utf8mb4_general_ci
transaction_isolation = READ-COMMITTED
innodb_file_per_table=1
#[Warning] 'innodb-large-prefix' was removed. It does nothing now and exists only for compatibility with old my.cnf files.
#innodb_large_prefix=on
#[Warning] 'innodb-defragment' was removed. It does nothing now and exists only for compatibility with old my.cnf files.
#innodb-defragment=1
```

File `config.user.inc.php`

```
<?php

declare(strict_types=1);

# Settings for phpMyAdmin

# $cfg['ShowPhpInfo'] = true; // Adds a link to phpinfo() on the home page
$cfg['AllowArbitraryServer'] = false;

// Array starts at 1 ...
```

```
$i = 1;
$cfg['Servers'][$i]['host'] = 'localhost';
$cfg['Servers'][$i]['socket'] = '/run/mysqld/mysqld.sock';
#$cfg['Servers'][$i]['auth_type'] = 'http';
```

Backup

Backing up MariaDB on Docker works like this:

```
#!/bin/bash
# This would be Postgres - i don't like postgres, so see beneath for mariadb:
# rm /backup/nextcloud_pgdumpall.zstd
# docker exec -t postgresql-pgsql_db-1 pg_dumpall -c -U postgres  | zstd -19 -o /backup/nextcloud_pgdumpall.zstd --no-
progress
# For mariadb use mariadb-dump and root
# For mysql use mysqldump and admin
BACKUPFILE=/backup/nextcloud_mariadbdump.zstd
MARIADB_PASS='verysecretpassword'
echo "Backup of mariadb to ${BACKUPFILE}"
rm ${BACKUPFILE}
docker exec -t mariadb-mariadb-1 mariadb-dump --all-databases --single-transaction --quick --lock-tables=false -u root -
p${MARIADB_PASS}  | zstd -19 -o ${BACKUPFILE} --no-progress
echo 'Backup done.'
```

No Docker Service

If you have MariaDB as native Host- Service installed, use:

```
#!/bin/bash
# Makes a Backup of the whole Mariadb
BACKUPFILE=/root/backup/mariadb_dump_all.zstd
echo "Fullbackup of MariaDB into ${BACKUPFILE}..."
# Keep one Copy of the old Backup
```

```
if [ -f ${BACKUPFILE} ]; then
    if [ -f ${BACKUPFILE}.back ]; then
        rm ${BACKUPFILE}.back
    fi
    mv ${BACKUPFILE} ${BACKUPFILE}.back
else
    echo "File ${BACKUPFILE} was not found, not removing ${BACKUPFILE}.back"
fi
# for mysql use mysqldump...
/usr/bin/mariadb-dump --all-databases --single-transaction --quick --lock-tables=true -u root -p'SECRETPASSPHRASE' | zstd
-19 -o ${BACKUPFILE} --no-progress
if [ ! -f ${BACKUPFILE} ]; then
    echo "ERROR: File ${BACKUPFILE} was not created - NO BACKUP PERFORMED. Please check the Backup."
    exit 1
fi
```

Optimizing Tables

Doing this will remove defragmentation and repair some stuff, so maybe you want to do this once a month or so `mariadb_optimize.sh`:

```
#!/bin/bash
#Optimize Database
# 28.11.2023: Docker-Version
# 13.12.2023: Replaced mysql by mariadb
#
echo "Check and optimize DB Mysql"

RUN_SQL=/srv/backupscripts/backupfiles/check_all_tables.sql
RUN_LOG=/srv/backupscripts/backupfiles/check_all_tables.log

#SQL="SELECT CONCAT('ANALYZE LOCAL TABLE \`',table_schema,'\`.\`',table_name,'\`;')"
SQL="SELECT CONCAT('CHECK TABLE \`',table_schema,'\`.\`',table_name,'\` EXTENDED;')"
SQL="${SQL} FROM information_schema.tables WHERE table_schema NOT IN"
SQL="${SQL} ('information_schema','performance_schema','mysql','sys','innodb')"
SQL="${SQL} AND engine IS NOT NULL"
```

```
MARIADB_USER='root'
MARIADB_PASS='verysecretpassword'
CONTAINERNAME='mariadb_mariadb_1'

RUN_CMD="docker exec -i ${CONTAINERNAME} mariadb -u${MARIADB_USER} -p${MARIADB_PASS}"

# Create SQL Commands to run
${RUN_CMD} -ANe"${SQL}" --raw --silent | grep TABLE> ${RUN_SQL}
# Execute CHECK TABLE Commands
${RUN_CMD} --raw --silent --table <${RUN_SQL}> ${RUN_LOG} 2>&1

#cat "${RUN_LOG}" | grep check
#RUN_ERROR=(`cat "${RUN_LOG}" | grep check`)
RUN_ERROR=(`cat "${RUN_LOG}" | grep error`)
if [[ ! -z "${RUN_ERROR}" |]]; then
  echo "************************************************"
  echo ""
  echo "!!!! ERROR- STOP OPTIMIZE: Some MySQL Databases are corrupt, please check output in ${RUN_LOG}:"
  cat "${RUN_LOG}"
  echo ""
  echo "************************************************"
  exit 1
fi
echo "Check MysqlDB was sucessful, no errors found"

exit 0

#For me, i choose to run this only every 6th of the month
ifStart=`date '+%d'`
if [ $ifStart == 06 ]
then
  echo "Optimize DB Mysql"
  RUN_SQL=/srv/backupscripts/backupfiles/optimize_all_tables.sql
  RUN_LOG=/srv/backupscripts/backupfiles/optimize_all_tables.log

  SQL="SELECT CONCAT('OPTIMIZE TABLE \`',table_schema,'\`.\`',table_name,'\`;')"
  SQL="${SQL} FROM information_schema.tables WHERE table_schema NOT IN"
  SQL="${SQL} ('information_schema','performance_schema','mysql','sys','innodb')"
```

```
  SQL="${SQL} AND engine IS NOT NULL"

  # Create SQL Commands to run
  ${RUN_CMD} -ANe"${SQL}" --raw --silent | grep TABLE> ${RUN_SQL}

  # Execute Commands
  ${RUN_CMD} --raw --silent --table <"${RUN_SQL}"> ${RUN_LOG} 2>&1
  cat "${RUN_LOG}"
  RUN_ERROR=(`cat "${RUN_LOG}" | grep error`)
  if [[ ! -z "${RUN_ERROR}" |]]; then
    echo "************************************************"
    echo ""
    echo "!!!! ERROR- STOP OPTIMIZE: Some MySQL Databases are corrupt, please check output in ${RUN_LOG}:"
    echo ""
    echo "************************************************"
    exit 1
  fi
  # Useless for Docker...
  #  /sbin/btrfs filesystem defragment /srv/faststorage/mysql -r
fi
echo "Optimize finished sucessfully"
exit 0

#Diese Version geht leider nicht, weil die KOmmandos im Docker nicht vorhanden sind...
#docker exec -t mysql-mysqldb-1 mysqlcheck -u admin -p'Ps.xz!)6JLn/YoGL' --check --auto-repair --all-databases 1>/dev/null
#docker exec -t mysql-mysqldb-1 mysqlcheck -u admin -p'Ps.xz!)6JLn/YoGL' --check --auto-repair --all-databases
```