

# MariaDB Komplettkurs

## Agenda

### 1. Architektur MariaDB

- [Architecture Server](#)
- [Architecture Server \(Steps\)](#)
- [Storage Engines](#)

### 2. Installation

- [Installation Centos/RockyLinux](#)
- [Start/Status/Stop/Enable von MariaDB](#)
- [Does mariadb listen to the outside world](#)

### 3. Database Objects

- [Create Database](#)
- [Show structure of table](#)
- [Show all tables within db](#)

### 4. InnoDB - Storage Engine

- [InnoDB - Storage Engine - Structure](#)
- [Important InnoDB - configuration - options to optimized performance](#)

### 5. Training Data

- [Setup training data "contributions"](#)
- [Setup sakila test db](#)

### 6. Security and User Rights

- [Create User/Grant/Revoke - Management of users](#)
- [Change password of user](#)
- [Automatisches Einloggen ohne Passwort](#)

### 7. Binlog, Backup and Restore (Point-In-Time aka PIT)

- [binlog aktivieren und auslesen](#)
- [Backup with mysqldump - best practices](#)
- [PIT - Point in time Recovery - Exercise](#)
- [Flashback](#)
- [mariabackup](#)

### 8. Performance

- [Slow Query Log](#)
- [Percona-toolkit-Installation - Centos](#)

### 9. Monitoring

- [What to monitor?](#)

### 10. Replication

- [Slave einrichten - gtid \(mit mariabackup\)](#)
- [Binary Logs auf master bis slave-log-master-position ändern](#)

## 11. Tipps & Tricks

- [Set hostname on systemd-Systems](#)
- [Frisches Datenverzeichnis anlegen](#)
- [In den Root-Benutzer wechseln](#)

## 12. Dokumentation

- [MySQL - Performance - PDF](#)
- [Server System Variables](#)
- [Killing connection](#)
- [MariaDB - One Script Installation](#)
- [MariaDB - Information Schema Tables](#)
- [MariaDB - slow query log](#)
- [MariaDB - sys - vor 10.6](#)

# Backlog

## 1. Architecture of MariaDB

- [Query Cache Usage and Performance](#)
- [Optimizer-Basics](#)

## 2. Installation

- [Installation SLES15](#)
- [Installation \(Ubuntu\)](#)
- [Does mariadb listen to the outside world](#)

## 3. Configuration

- [Adjust configuration and restart](#)
- [Set global server system variable](#)

## 4. Backup

- [Use xtrabackup for MariaDB 5.5](#)
- [Ready-made-back-scripts](#)
- [Simple-Backup-Script](#)

## 5. Administration / Troubleshooting

- [Create fresh datadir \(Centos/Redhat\)](#)
- [Debug not starting service](#)

## 6. Galera

- [Installation and Configuration \(Centos/Redhat 8\)](#)
- [1. Node started nicht nach Crash, z.B. Stromausfall](#)

## 7. Information Schema / Status / Processes

- [Show server/session status](#)
- [Kill long running process](#)
- [Kill \(kickout user\) and stop server](#)

## 8. Security and User Rights

- [Debug and Setup External Connection](#)
- [Get Rights of user](#)

- [Secure with SSL server/client](#)
- [Auth with unix\\_socket](#)
- Authentication and Authorization
- [User- and Permission-concepts \(best-practice\)](#)
- [Setup external access](#)
- [Table encryption](#)

## 9. SELinux

- [Welche Ports sind freigegeben? \(MariaDb startet damit\)](#)
- [Neues Datenverzeichnis SELinux bekanntmachen - semanage fcontext](#)
- [Probleme mit SELinux erkennen und debuggen](#)

## 10. Database - Objects

- [Triggers](#)
- [Functions](#)
- [Stored Procedure](#)
- [Events](#)
- [Data Types](#)

## 11. Locking

- [Implicit Locks](#)
- [Identify Deadlocks in innodb](#)

## 12. Upgrade

- [MariaDB Upgrade 10.3 \(Centos\) -> 10.4 \(Mariadb.org\)](#)
- [MariaDB Upgrade 10.4 -> 10.5 \(Centos\)](#)
- [MariaDB Upgrade 5.5 -> 10.5](#)

## 13. Performance

- [io-Last/CPU-Last](#)
- [Views and performance](#)
- [Partitions and Explain](#)
- [3 Phases of DataSize](#)
- [Slow Query Log](#)

## 14. Optimal use of indexes

- Index-Types
- [Describe and indexes](#)
- [Find out indexes](#)
- [Index and Functions](#)
- [Index and Likes](#)
- [profiling-get-time-for-execution-of.query](#)
- [Find out cardinality without index](#)

## 15. Replication

- [Slave einrichten - Centos - old style \(master\\_pos\)](#)
- [MaxScale installieren](#)
- [Reference: MaxScale-Proxy mit Monitoring](#)
- [Walkthrough:Automatic Failover Master Slave](#)

## 16. Tools & Tricks

- [Percona-toolkit-Installation - Ubuntu](#)
- [pt-query-digest under Windows](#)
- [pt-query-digest - analyze slow logs](#)
- [pt-online-schema-change howto](#)
- [Ubuntu-with-Vagrant](#)
- [mysql-client](#)
- [Schweizer Such-Taschenmesser grep -r](#)
- [Set timezone in Centos 7/8](#)
- [Ist die Netzwerkkarte eingerichtet - nmtui](#)
- [User anlegen und passwort vergeben \(Centos/Redhat\)](#)
- [Scripts for deploying galera-cluster to Ubuntu 20.04](#)

#### 17. Extras

- [User Variables](#)
- [Installation sakila-db](#)

#### 18. Diagnosis and measurement of performance

- [Best practices to narrow down performance problems](#)

#### 19. Performance and optimization of SQL statements

- [Do not use '\\*' whenever possible](#)
- [Optimizer-hints \(and why you should not use them\)](#)

#### 20. Replication

- [Replikation Read/Write](#)

#### 21. Performance

- [Best Practices](#)
- [Example sys-schema and Reference](#)
- [Change schema online \(pt-online-schema-change\)](#)
- [Optimizer-Hints](#)

#### 22. Documentation / Literature

- [MySQL - Performance Blog](#)
- [Source-Code MariaDB](#)
- [Effective MySQL](#)
- [Last Training](#)
- [MariaDB Galera Cluster](#)
- [MySQL Galera Cluster](#)
- [Releases List - Long Time / Stable](#)

### 23. Questions and Answers

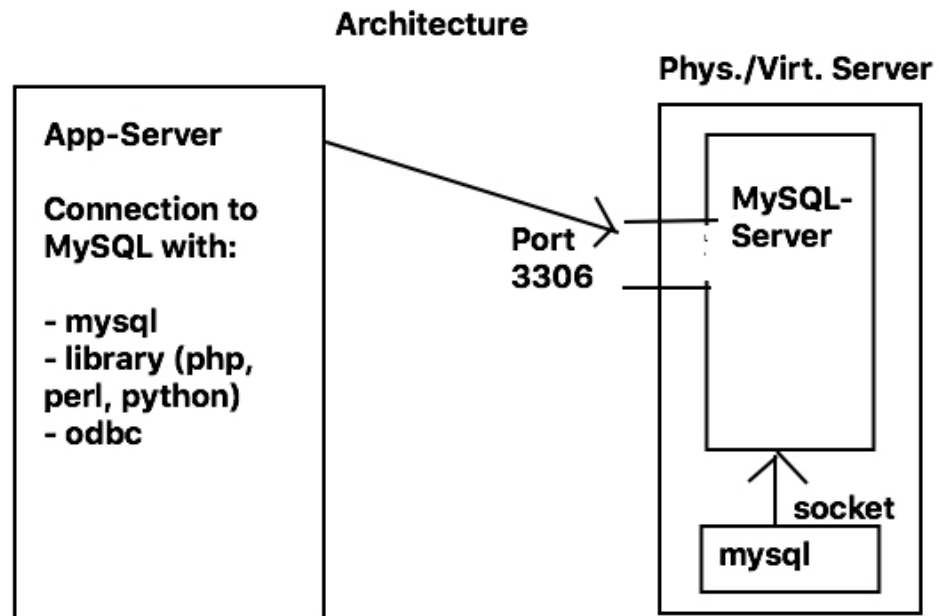
- [Questions and Answers](#)
- [Best filesystem for MariaDB](#)

### 24. MySQL Do-Nots

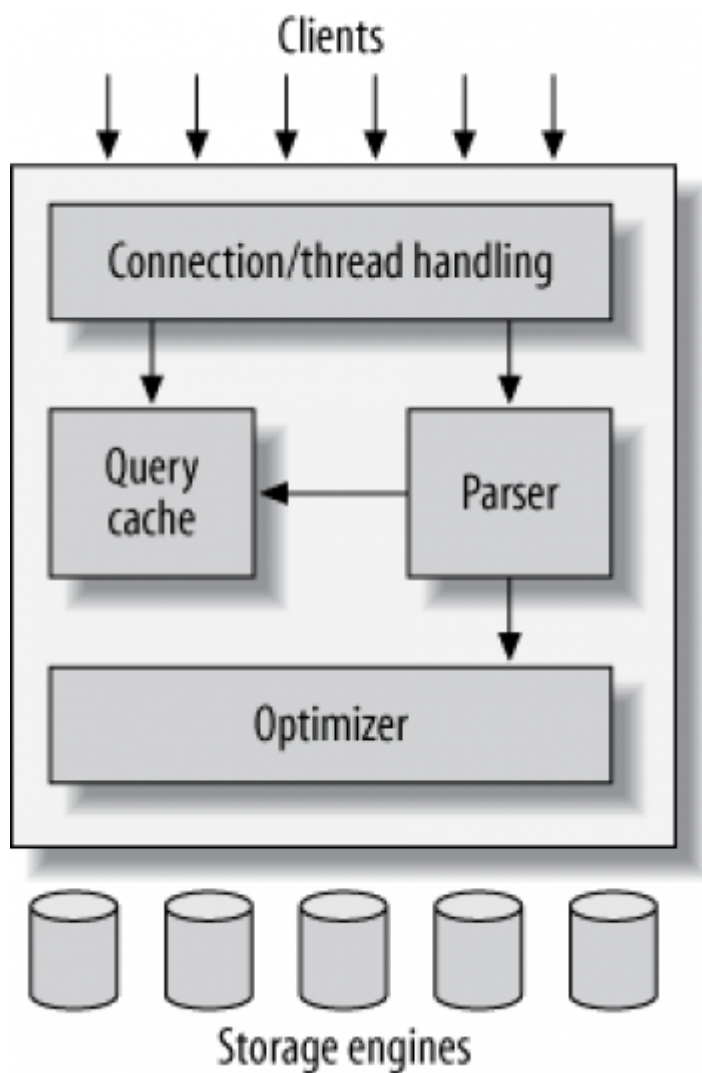
- [mysql-do-nots](#)

## Architektur MariaDB

### Architecture Server



### Architecture Server (Steps)



## Storage Engines

### Why ?

Let's you choose:  
How your data is stored

### What ?

- Performance, features and other characteristics you want

### Where ?

- Theoretically you can use a different engine for every table
- But: For performance optimization and future, it is better to concentrate on one

### What do they do ?

- In charge for: Responsible for storing and retrieving all data stored in MySQL
- Each storage engine has its:
  - Drawbacks and benefits

- Server communicates with them through the storage engine API
  - this interface hides differences
  - makes them largely transparent at query layer
  - api contains a couple of dozen low-level functions e.g. “begin a transaction”, “fetch the row that has this primary key”

### Storage Engine do not ....

- Storage Engines do not parse SQL
- Storage Engines do not communicate with each other

### They simply .....

- They simply respond to requests from the server

### Which are the most important one ?

- InnoDB (currently default engine)
- MyISAM/Aria
- Memory
- CSV
- Blackhole (/dev/null)
- Archive
- Partition
- (Federated/FederatedX)

## Installation

### Installation Centos/RockyLinux

#### Install from Distribution

```
dnf search mariadb
## find version
dnf info mariadb-server
dnf install -y mariadb-server
```

#### Install from MariaDB Foundation (Repo)

##### Find Repo Settings

- [https://mariadb.org/download/?t=mariadb&p=mariadb&r=10.9.3&os=windows&cpu=x86\\_64&pkg=msi&m=hs-esslingen](https://mariadb.org/download/?t=mariadb&p=mariadb&r=10.9.3&os=windows&cpu=x86_64&pkg=msi&m=hs-esslingen)

#### Setup Repo MariaDB - Server 10.6

```
## Setup repo
## nano /etc/yum.repos.d/MariaDB.repo
```

```
## MariaDB 10.6 CentOS repository list - created 2022-09-20 09:46 UTC
## https://mariadb.org/download/
[mariadb]
name = MariaDB
baseurl = https://mirror1.hs-esslingen.de/pub/Mirrors/mariadb/yum/10.6/centos8-amd64
module_hotfixes=1
```



```
gpgkey=https://mirror1.hs-esslingen.de/pub/Mirrors/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

```
## Install
sudo dnf install -y install MariaDB-server
sudo systemctl start mysql # always works - systemd - alias
sudo systemctl status mysql # Findout real service - name
## like Windows-Autostart
sudo systemctl enable mariadb
sudo systemctl status mariadb
```

## Secure installation

```
mariadb-secure-installation
## OR: if not present before 10.4
mysql_secure_installation
```

## Start/Status/Stop/Enable von MariaDB

### start/stop/status

```
## als root - user
systemctl status mariadb
systemctl stop mariadb
systemctl start mariadb

##
systemctl restart mariadb
```

### enable

```
## enable to be started after reboot
systemctl enable mariadb
```

## how is service configured / systemd-wise

```
systemctl cat mariadb
```

## Does mariadb listen to the outside world

### How to check ?

```
lsof -i | grep mariadb
## localhost means it does NOT listen to the outside now
## mariabdd 5208          mysql  19u  IPv4  56942      0t0  TCP localhost:mysql
(LISTEN)

netstat -tupe
## or
netstat -an
```

## Database Objects

### Create Database

```
create schema training
create database training
```

### Show structure of table

#### show create table

```
use mysql;
show create table user
```

#### describe table

```
use mysql;
describe user;
```

### Show all tables within db

#### show all tables in database

```
## connect with db training
mysql training
mysql> show tables;
|training|
```

#### describe

```
MariaDB [training]> describe mitarbeiter;
+-----+-----+-----+-----+-----+-----+
| Field | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | tinyint(3) unsigned | NO   | PRI | NULL    |       |
| name  | varchar(50)         | YES  |     | NULL    |       |
| vorname | varchar(30)         | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```

#### show create

```
MariaDB [training]> show create table mitarbeiter;
```

```
+-----+-----+
+-----+-----+
+-----+-----+
| Table | Create Table
|
+-----+-----+
+-----+-----+
```

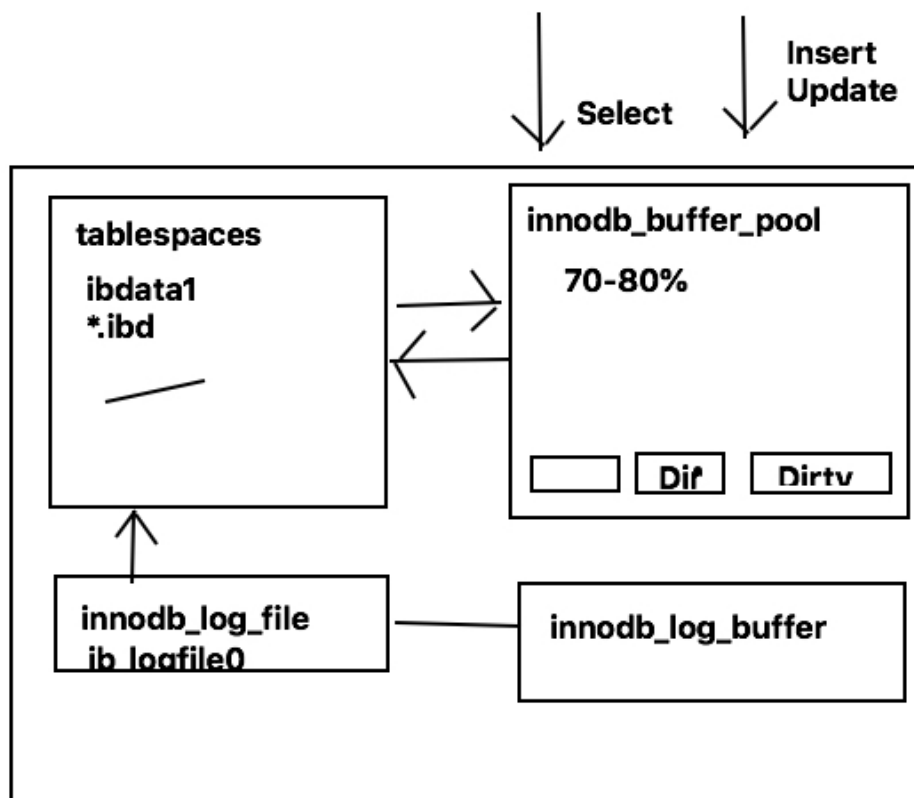
```

-----+
| mitarbeiter | CREATE TABLE `mitarbeiter` (
| `id` tinyint(3) unsigned NOT NULL,
| `name` varchar(50) DEFAULT NULL,
| `vorname` varchar(30) DEFAULT NULL,
| PRIMARY KEY (`id`)
| ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+-----+
-----+
1 row in set (0.000 sec)

```

## InnoDB - Storage Engine

### InnoDB - Storage Engine - Structure



### Important InnoDB - configuration - options to optimized performance

#### How big is the innodb buffer currently (setup) ?

```

mysql>select @@innodb_buffer_pool_size;
mysql>show variables like '%buffer%';

```

#### Innodb buffer pool

- How much data fits into memory
- Free buffers = pages of 16 Kbytes
- Free buffer \* 16Kbytes = free innodb buffer pool in KByte

```
## does not in windows -> pager grep
pager grep -i 'free buffers'
## does not work with workbench or heidisql because of formatting + \G only
works in client
show engine innodb status \G
Free buffers          7905
1 row in set (0.00 sec)
```

### Innodb buffer pool stats with status

```
## Also works in heidisql or workbench
show status like '%buffer%';
```

### Overview innodb server variables / settings

- <https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html>

### Change innodb\_buffer\_pool

```
## /etc/mysql/mysql.conf.d/mysqld.cnf
## 70-80% of memory on dedicated mysql
[mysqld]
innodb-buffer-pool-size=6G

##
systemctl restart mysql

##
mysql
mysql>show variables like 'innodb%buffer%';
```

### problems, when dynamically increasing buffer

- <https://www.percona.com/blog/2018/06/19/chunk-change-innodb-buffer-pool-resizing/>

### innodb\_flush\_method

Ideally O\_DIRECT on Linux, but please test it, if it really works well.

### innodb\_flush\_log\_at\_trx\_commit

When is flushing done from innodb\_log\_buffer to log.  
 Default: 1 : After every commit  
 -> best performance 2. -> once per second

```
## Good to use 2, if you are willing to loose 1 second of data on powerfail
```

### innodb\_flush\_neighbors

```
## on ssd disks set this to off, because there is no performance improvement
innodb_flush_neighbors=0

## Default = 1
```

## innodb\_log\_file\_size

```
## Should hold 60-120 min of data flow
## Calculate like so:
https://www.percona.com/blog/2008/11/21/how-to-calculate-a-good-innodb-log-file-size/
```

## skip-name-resolve.conf

```
## work only with ip's - better for performance
/etc/my.cnf
skip-name-resolve
```

- <https://nixcp.com/skip-name-resolve/>

## Ref:

- <https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool-resize.html>

## Privileges for show engine innodb status

```
show engine innodb status \G
ERROR 1227 (42000): Access denied; you need (at least one of) the PROCESS privilege(s)
for this operation
```

## Training Data

### Setup training data "contributions"

#### Walkthrough

- Complete process takes about 10 minutes

```
cd /usr/src
apt update; apt install -y git
git clone https://github.com/jmetzger/dedupe-examples.git
cd dedupe-examples
cd mysql_example
## Eventually you need to enter (in mysql_example/mysql.cnf)
## Only necessary if you cannot connect to db by entering "mysql"
## password=<your_root_pw>
./setup.sh
```

## Setup sakila test db

```
cd /usr/src
wget https://downloads.mysql.com/docs/sakila-db.tar.gz
tar xzvf sakila-db.tar.gz
```

```
cd sakila-db
mysql < sakila-schema.sql
mysql < sakila-data.sql
```

## Security and User Rights

### Create User/Grant/Revoke - Management of users

#### Create user

```
create user training@localhost identified by 'deinpassword';
```

#### Drop user (=delete user)

```
drop user training@localhost
```

#### Change User (e.g. change authentication)

```
## change pass
alter user training@localhost identified by 'newpassword';
```

#### Set global or db rights for a user

```
grant all on *.* to training@localhost
## only a specific db
grant all on mydb.* to training@localhost
```

#### Revoke global or revoke right from a user

```
revoke select on *.* from training@localhost
## only from a specific db
revoke select on training.* from training@localhost
```

#### Refs:

- <https://mariadb.com/kb/en/grant/#the-grant-option-privilege>
- <https://mariadb.com/kb/en/revoke/>

#### Change password of user

```
## you must be root or privileged to changed passwords
alter user training@localhost identified by 'password';
```

#### Automatisches Einloggen ohne Passwort

```
cat /home/kurs/.my.cnf
[mysql]
user=training
password=password
```

```
## einloggen als training
mysql
```

## Binlog, Backup and Restore (Point-In-Time aka PIT)

### binlog aktivieren und auslesen

#### Binlog - Wann ?

- PIT (Point-in-Time) - Recovery
- Master/Slave - Replication
- MariaDB Galera Cluster (meckert, wenn nicht aktiviert -> GUT !)

### Binlog aktivieren (Centos)

```
## /etc/my.cnf.d/server.cnf
[mysqld]
log-bin

## Server neu starten
systemctl restart mariadb
```

### Alte Logs automatisch löschen

- [https://mariadb.com/kb/en/replication-and-binary-log-system-variables/#expire\\_logs\\_days](https://mariadb.com/kb/en/replication-and-binary-log-system-variables/#expire_logs_days)

### Rowbasiertes Logging aktivieren

```
## Generell empfehlenswert da sicherer
## /etc/my.cnf.d/server.cnf
[mysqld]
log-bin
binlog-format=ROW

## Server neu starten
systemctl restart mariadb
```

### binlog auslesen

```
cd /var/lib/mysql
## Zeigt auch mit Kommentar die SQL-Statements an die bei ROW-basierten binlog
ausgeführt werden
mysqlbinlog -vv rechnername1-bin.000001
```

### Wie finde ich raus, welches binlog aktiv ist ?

```
## mysql -client starten
mysql> show master status;
```

### Backup with mysqldump - best practices

### Dumping (best option) without active binary log

```
mysqldump --all-databases --single-transaction > /usr/src/all-databases
## if you want to include procedures use --routines
## with event - scheduled tasks
mysqldump --all-databases --single-transaction --routines --events > /usr/src/all-databases
```

## Useful options for PIT

```
## --quick not needed, because included in --opt which is enabled by default

## on local systems using socket, there are no huge benefits concerning --compress
## when you dump over the network use it for sure
mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --events --flush-logs > /usr/src/all-databases.sql;
```

## With PIT\_Recovery you can use --delete-master-logs

- All logs before flushing will be deleted

```
mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --events --flush-logs --delete-master-logs > /usr/src/all-databases.sql;
```

## Flush binary logs from mysql

```
mysql -e "PURGE BINARY LOGS BEFORE '2013-04-22 09:55:22'";
```

## Version with zipping

```
mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --events --flush-logs --compress | gzip > /usr/src/all-databases.sql.gz
```

## Performance Test mysqldump (1.7 Million rows in contributions)

```
date; mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --events --flush-logs --compress > /usr/src/all-databases.sql; date
Mi 20. Jan 09:40:44 CET 2021
Mi 20. Jan 09:41:55 CET 2021
```

## Seperated sql-structure files and data-txt files including master-data for a specific database

```
# backups needs to be writeable for mysql
mkdir /backups
chmod 777 /backups
chown mysql:mysql /backups
mysqldump --tab=/backups contributions
mysqldump --tab=/backups --master-data=2 contributions
mysqldump --tab=/backups --master-data=2 contributions > /backups/master-data.tx
```

## Create new database base on sakila database



```
cd /usr/src
mysqldump sakila > sakila-all.sql
echo "create database mynewdb" | mysql
mysql mynewdb < sakila-all.sql
```

## PIT - Point in time Recovery - Exercise

### Problem coming up

```
## Step 1 : Create full backup (assuming 24:00 o'clock)
mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --
events --flush-logs --delete-master-logs > /usr/src/all-databases.sql;

## Step 2: Working on data
mysql>use sakila;
mysql>insert into actor (first_name,last_name) values ('john','The Rock');
mysql>insert into actor (first_name,last_name) values ('johanne','Johannson');

## Optional: Step 3: Looking into binary to see this data
cd /var/lib/mysql
## last binlog
mysqlbinlog -vv mariadb-bin.000005

## Step 4: Some how a guy deletes data
mysql>use sakila; delete from actor where actor_id > 200;
## now only 200 datasets
mysql>use sakila; select * from actor;
```

### Fixing the problem

```
## find out the last binlog
## Simple take the last binlog

cd /var/lib/mysql
## Find the position where the problem occurred
## and create a recover.sql - file (before apply full backup)
mysqlbinlog -vv --stop-position=857 mysqld-bin.000005 > /usr/src/recover.sql
```

```
## Step 1: Apply full backup
cd /usr/src/
mysql < all-databases.sql
```

```
-- im mysql-client durch eingeben des Befehls 'mysql'
-- should be 200 or 202
use sakila; select * from actor;
```

```
## auf der Kommandozeile
mysql < recover.sql
```

```
-- im mysql client
-- now it should have all actors before deletion
use sakila; select * from actor;
```

## Flashback

- Redoes insert/update/delete entries from binlog (binlog\_format = 'ROW')

## Referenz:

- <https://mariadb.com/kb/en/flashback/>

## mariabackup

### Installation von Repo mariadb (von mariadb.org)

```
dnf install MariaDB-backup
```

### Installation von Distri (Centos/Rocky/RHEL)

```
## Rocky 8
dnf install mariadb-backup
```

## Walkthrough (Ubuntu/Debian)

```
## user eintrag in /root/.my.cnf
[mariabackup]
user=root
## pass is not needed here, because we have the user root with unix_socket - auth

mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/20210120 --backup
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 -> 0 bytes
mariabackup --target-dir=/backups/20210120 --prepare

### Recover
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/backups/20200120 --copy-back
chown -R mysql:mysql /var/lib/mysql
chmod 755 /var/lib/mysql # otherwise socket for unprivileged user does not work
systemctl start mariadb
```

## Walkthrough (Redhat/Centos/Rocky Linux 8 mit mariadb for mariadb.org)

```
## user eintrag in /root/.my.cnf
[mariabackup]
user=root
## pass is not needed here, because we have the user root with unix_socket - auth
## or generic
```

```

## /etc/my.cnf.d/mariabackup.cnf
[mariabackup]
user=root

mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/20210120 --backup
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 -> 0 bytes
mariabackup --target-dir=/backups/20210120 --prepare

### Recover
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/backups/20200120 --copy-back
chown -R mysql:mysql /var/lib/mysql
chmod 755 /var/lib/mysql # otherwise socket for unprivileged user does not work
systemctl start mariadb

### important for selinux if it does not work
### does not start
restorecon -vr /var/lib/mysql
systemctl start mariadb

```

## Ref.

<https://mariadb.com/kb/en/full-backup-and-restore-with-mariabackup/>

## Performance

### Slow Query Log

#### Walkthrough

```

## Step 1
/etc/my.cnf.d/mariadb-server.cnf
[mysqld]
slow-query-log

## Step 2
mysql>SET GLOBAL slow_query_log = 1
mysql>SET slow_query_log = 1
mysql>SET GLOBAL long_query_time = 0.000001
mysql>SET long_query_time = 0.000001

## Step 3
## run some time / data
## and look into your slow-query-log
/var/lib/mysql/hostname-slow.log

```

### Show queries that do not use indexes

```
SET GLOBAL log_queries_not_using_indexes=0N;
```

## Geschwätzigkeit (Verbosity) erhöhen

```
SET GLOBAL log_slow_verbosity='query_plan,explain'
```

## Reference

- <https://mariadb.com/kb/en/slow-query-log-overview/>

## Percona-toolkit-Installation - Centos

### Walkthrough

```
## Howto
## https://www.percona.com/doc/percona-toolkit/LATEST/installation.html

## Step 1: repo installieren mit rpm -paket
dnf install -y https://repo.percona.com/yum/percona-release-latest.noarch.rpm; dnf
install -y percona-toolkit
```

## Monitoring

### What to monitor?

### What to monitor

#### System

- Last auf dem System (top)
- Festplatte (z.B. 85% voll ?) df /var/lib/mysql
- Swap (Wenn gewappt wird ist Hopfen und Malz verloren)

#### Erreichbarkeit

- Server per ping erreichen (mysqladmin ping -h ziel-ip)
- Einlogbar ? (myadmin ping -h ziel-ip -u control\_user)

#### Platte aka IO-Subsystem (iostats)

- <http://schulung.t3isp.de/documents/pdfs/mysql/mysql-performance.pdf>

| --                      | --   | -- |
|-------------------------|--|----|
| Read/Write requests     | IOPS (Input/Output operations per second)                | -- |
| Average IO wait         | Time that queue operations have to wait for disk access  | -- |
| Average Read/Write time | Time it takes to finish disk access operations (latency) | -- |
| Read/Write bandwidth    | Data transfer from and towards your disk                 | -- |

### Gneral mysql metrics

```
mysql -E -e "select variable_value from information_schema.session_status where
variable_name = 'uptime'";
```

```
# max connections
MariaDB [(none)]> show status like 'max_used_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Max_used_connections | 1 |
+-----+-----+
1 row in set (0.001 sec)

MariaDB [(none)]> show variables like 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
+-----+-----+
1 row in set (0.001 sec)

mysqladmin status
## you will find uptime here in seconds
```

| Metric               | Comments   | Suggested Alert                  |
|----------------------|--|----------------------------------|
| Uptime               | Seconds since the server was started. We can use this to detect respawns.  | When uptime is < 180. (seconds)  |
| Threads_connected    | Number of clients currently connected. If none or too high, something is wrong.  | None                             |
| Max_used_connections | Max number of connections at a time since server started. (max_used_connections / max_connections) indicates if you could run out soon of connection slots.              | When connections usage is > 85%. |
| Aborted_connects     | Number of failed connection attempts. When growing over a period of time either some credentials are wrong or we are being attacked. show status like 'Aborted_connects' | When aborted connects/min > 3.   |

## InnoDB

| Metric                       | Comments   | Suggested Alert |
|------------------------------|--|-----------------|
| Innodb_row_lock_waits        | Number of times InnoDB had to wait before locking a row.   | None            |
| Innodb_buffer_pool_wait_free | Number of times InnoDB had to wait for memory pages to be flushed. If too high, innodb_buffer_pool_size is too small for current write load. | None            |

## Query tracking

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

| Metric  | Comments   | Suggested Alert |
|---|--|-----------------|
| Slow_queries  | Number of queries that took more than long_query_time seconds to execute. Slow queries generate excessive disk reads, memory and CPU usage. Check slow_query_log to find them. | None            |
| Select_full_join  | Number of full joins needed to answer queries. If too high, improve your indexing or database schema.  | None            |
| Created_tmp_disk_tables   | Number of temporary tables (typically for joins) stored on slow spinning disks, instead of faster RAM.   | None            |
| (Full table scans) Handler_read% Number of times the system reads the first row of a table index. (if 0 a table scan is done - because no key was read). Sequential reads might indicate a faulty index. None |  |                 |

## Track Errors

```
journalctl -u mariadb | grep -i Error
```

## Monitoring with pmm (Percona Management Monitoring)

<https://pmmdemo.percona.com>

[Documentation](#)

## Replication

### Slave einrichten - gtid (mit mariabackup)

#### Step 0.5a: Installation on ubuntu/debian

```
apt update
apt install mariadb-backup
## check if available
mariabackup --version

## prepare for mariabackup if you use it with root and with unix_socket
/root/.my.cnf
[mariabackup]
user=root
```

#### Step 1: mariabackup on master

```
mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/20210121 --backup
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 -> 0 bytes
mariabackup --target-dir=/backups/20210121 --prepare
```

## Step 2: Transfer to new slave (from master)

```
## root@master:
rsync -e ssh -avP /backups/20210121 student@10.10.9.144:/home/student/
```

## Step 3: Setup replication user on master

```
## as root@master
##mysql>
CREATE USER repl@'10.10.9.%' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'10.10.9.%';
```

## Step 3a (Optional): Test repl user (connect) from slave

```
## as root@slave
## you be able to connect to
mysql -urepl -p -h10.10.9.110
## test if grants are o.k.
show grants
```

## Step 4a: Set server-id on master -> 1

```
[mysqld]
server-id=1

systemctl restart mariadb
###
```

## Step 4b: Set server-id on slave -> 3 + same config as server 1 + log\_slave\_update

```
[mysqld]
server-id          = 3
## activate master bin log, if this slave might be a master later
log_bin           = /var/log/mysql/mysql-bin.log
binlog_format = ROW
log_slave_update = 1

systemctl restart mariadb
### auf dem master config mit rsync rüberschrieben
### root@master
rsync -e ssh -avP /etc/mysql/mariadb.conf.d/z_uniruhr.cnf kurs@10.10.9.144:/home/kurs/
```

## Step 5: Restore Data on slave

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/home/student/20210121 --copy-back
chown -R mysql:mysql /var/lib/mysql
systemctl start mariadb
```

## Step 6: master.txt for change command

```
## root@slave
$ cat xtrabackup_binlog_info
mariadb-bin.000096 568 0-1-2

SET GLOBAL gtid_slave_pos = "0-1-2";
## /root/master.txt
## get information from master-databases.sql dump
CHANGE MASTER TO
  MASTER_HOST="10.10.9.110",
  MASTER_PORT=3306,
  MASTER_USER="repl",
  MASTER_PASSWORD="password",
  MASTER_USE_GTID=slave_pos;

mysql < master.txt
## or: copy paste into mysql>

## mysql>
start slave

## in mysql -> show slave status
mysql>show slave status
## Looking for
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

## Walkthrough

<https://mariadb.com/kb/en/setting-up-a-replication-slave-with-mariabackup/>

## Binary Logs auf master bis slave-log-master-position ändern

```
## Schritt 1: slave abfragen
mysql -u ext -p -h 192.168.56.103 --pager="grep -E 'Master_Log_File:'" -e "show slave
status \G"
## Beispiel Ausgabe
mariadb-bin.000003

## Schritt 2: logs bis dahin löschen auf master
mysql -e "purge logs to 'mariadb-bin.000003'"
```



## Tipps & Tricks

### Set hostname on systemd-Systems

```
## you have to be root
hostnamectl set-hostname mariadb1.training.local
## so that you will see it in your current prompt
su -
hostnamectl
```

### Frisches Datenverzeichnis anlegen

#### Walkthrough

```
## Schritt 1: Prepare
systemctl stop mariadb
cd /var/lib
## eventually delete old back dir
rm -fR /var/lib/mysql.bkup
##
mv mysql mysql.bkup

## Schritt 2: Fresh
mysql_install_db --user=mysql
chown mysql:mysql mysql
chmod g+rx,o+rx mysql
restorecon -rv /var/lib/mysql

## Schritt 3: Start
systemctl start mariadb
```

### In den Root-Benutzer wechseln

```
## einloggen als normaler Benutzer z.B. benutzer: kurs (wenn ich unter kurs eingeloggt bin)
sudo su -
## Eingeben des Passworts des Benutzers

### oder sudo -i (interaktiv)
sudo -i
## Eingeben des Passworts des normalen Benutzers
```

## Dokumentation

### MySQL - Performance - PDF

- <http://schulung.t3isp.de/documents/pdfs/mysql/mysql-performance.pdf>

### Server System Variables

- [https://mariadb.com/kb/en/server-system-variables/#bind\\_address](https://mariadb.com/kb/en/server-system-variables/#bind_address)

## Killing connection

- <https://mariadb.com/kb/en/kill/>

## MariaDB - One Script Installation

- <https://mariadb.com/kb/en/mariadb-package-repository-setup-and-usage/>

## MariaDB - Information Schema Tables

- <https://mariadb.com/kb/en/information-schema-tables/>

## MariaDB - slow query log

- <https://mariadb.com/kb/en/slow-query-log-overview/>

## MariaDB - sys - vor 10.6

- <https://github.com/FromDual/mariadb-sys>

# Architecture of MariaDB

## Query Cache Usage and Performance

### Performance query cache

- Always try to optimize innodb with disabled query cache first (innodb\_buffer\_pool)
- If you use query\_cache system can only use on CPU-Core. !!

### How to enable query cache

```
## have_query_cache means compiled in mysql
## query_cache_type off means not enable by config
-- query cache is disabled
mysql> show variables like '%query_cache%';
+-----+
| Variable_name          | Value |
+-----+
| have_query_cache       | YES   |
| query_cache_limit       | 1048576 |
| query_cache_min_res_unit | 4096   |
| query_cache_size        | 1048576 |
| query_cache_type        | OFF    |
| query_cache_wlock_invalidate | OFF    |
+-----+
6 rows in set (0.01 sec)
```

```
root@trn01:/etc/mysql/mysql.conf.d# tail mysqld.cnf
[mysqld]
pid-file           = /var/run/mysqld/mysqld.pid
socket             = /var/run/mysqld/mysqld.sock
datadir            = /var/lib/mysql
log-error          = /var/log/mysql/error.log
## By default we only accept connections from localhost
bind-address       = 0.0.0.0
## Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
```

```

query-cache-type=1

systemctl restart mysql

mysql> show variables like '%query_cache%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| have_query_cache       | YES   |
| query_cache_limit      | 1048576 |
| query_cache_min_res_unit | 4096   |
| query_cache_size       | 1048576 |
| query_cache_type       | ON     |
| query_cache_wlock_invalidate | OFF   |
+-----+-----+
6 rows in set (0.01 sec)

mysql> show status like '%Qcache%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Qcache_free_blocks    | 1      |
| Qcache_free_memory    | 1031832 |
| Qcache_hits           | 0      |
| Qcache_inserts        | 0      |
| Qcache_lowmem_prunes  | 0      |
| Qcache_not_cached     | 0      |
| Qcache_queries_in_cache | 0      |
| Qcache_total_blocks   | 1      |
+-----+-----+
8 rows in set (0.00 sec)

## status in session zurücksetzen.
mysql> flush status;
Query OK, 0 rows affected (0.00 sec)

```

## Performance bottleneck - mutex

<https://mariadb.com/de/resources/blog/flexible-mariadb-server-query-cache/>

## Something planned ?

- Nope ;o( Demand is new
- You might be able to use Demand together with maxscale
- Refer to: <https://mariadb.com/de/resources/blog/flexible-mariadb-server-query-cache/>

A mutual exclusion object (mutex) is a programming object that allows multiple program threads to share a resource (such as a folder) but not simultaneously. Mutex is set to unlock when the data is no longer needed or when a routine is finished. Mutex creates a bottleneck effect. The blocking means only one query can look at the Query Cache at

a time and other queries must wait. A query that must wait to look in the cache only to find it isn't in the cache will be slowed instead of being accelerated.

## Optimizer-Basics

### General

- All optimizer today are cost-based

### Cost-Based

```
## How much costs are needed to get the information
```

## Installation

### Installation SLES15

- [https://downloads.mariadb.org/mariadb/repositories/#distro=SLES&distro\\_release=sles15-amd64--sles15&mirror=timo&version=10.5](https://downloads.mariadb.org/mariadb/repositories/#distro=SLES&distro_release=sles15-amd64--sles15&mirror=timo&version=10.5)

### Installation (Ubuntu)

#### Setup repo and install

- <https://downloads.mariadb.org/mariadb/repositories/>

```
### repo
sudo apt-get install software-properties-common
sudo apt-key adv --fetch-keys 'https://mariadb.org/mariadb_release_signing_key.asc'
## does an apt update after setting repo - automatically
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el]
https://mirror.dogado.de/mariadb/repo/10.5/ubuntu focal main'
sudo apt install mariadb-server
```

### Secure installation

```
mariadb-secure-installation
## OR: if not present before 10.4
mysql_secure_installation
```

### Does mariadb listen to the outside world

#### How to check ?

```
lsof -i | grep mariadb
## localhost means it does NOT listen to the outside now
## mariabdb 5208          mysql  19u  IPv4  56942      0t0  TCP localhost:mysql
(LISTEN)

netstat -tupel
## or
netstat -an
```

## Configuration

### Adjust configuration and restart

```
## change config in /etc/mysql/50-server.cnf
## After that restart server - so that it takes your new config
systemctl restart mariadb
echo $? # Was call restart succesful -> 0
```

### Set global server system variable

#### Find out current value

```
## show global variable
show global variables like '%automatic_sp%'
## or // variable_name needs to be in captitals
use information_schema
select * from global_variables where variable_name like '%AUTOMATIC_SP%';

## If you know the exact name
select @@global.automatic_sp_privileges;
select @@GLOBAL.automatic_sp_privileges;
```

### Set global Variable

```
## will be set like so till next restart of mysql server
set global automatic_sp_privileges = 0
```

### automatic\_sp\_privileges can only be set globally

```
## Refer to: server system variable doku

## Has same value in global an session scope
MariaDB [information_schema]> select @@automatic_sp_privileges; select
@@global.automatic_sp_privileges;
+-----+
| @@automatic_sp_privileges |
+-----+
|                          0 |
+-----+
1 row in set (0.000 sec)

+-----+
| @@global.automatic_sp_privileges |
+-----+
|                          0 |
+-----+
1 row in set (0.000 sec)
```

### Reference:

- [https://mariadb.com/kb/en/server-system-variables/#automatic\\_sp\\_privileges](https://mariadb.com/kb/en/server-system-variables/#automatic_sp_privileges)

## Backup

### Use xtrabackup for MariaDB 5.5

#### For mariadb 5.5 you can use xtrabackup instead of mariabackup

- <https://www.percona.com/doc/percona-xtrabackup/2.4/index.html>

### Ready-made-back-scripts

- <https://gist.github.com/skarllot/2576266>

### Simple-Backup-Script

#### Backup Script

```
cat backup-test.sh
##!/bin/bash

DATABASES=$(echo "select schema_name from information_schema.schemata where
schema_name != 'performance_schema' and schema_name != 'information_schema';" | mysql)
for i in $DATABASES
do
    mysqldump $i > /usr/src/dump_$i.sql
done
```

## Administration / Troubleshooting

### Create fresh datadir (Centos/Redhat)

#### Walkthrough

```
## Schritt 1: Prepare
systemctl stop mariadb
cd /var/lib
## eventually delete old back dir
rm -fR /var/lib/mysql.bkup
##
mv mysql mysql.bkup

## Schritt 2: Fresh
mysql_install_db --user=mysql
chown mysql:mysql mysql
chmod g+rx,o+rx mysql
restorecon -rv /var/lib/mysql

## Schritt 3: Start
systemctl start mariadb
```

### Debug not starting service

#### Walkthrough

```

## Service is not restarting - error giving
systemctl restart mariadb.service

## Step 1 : status -> what do the logs tell (last 10 lines)
systemctl status mariadb.service

## no findings -> step 2:
journalctl -xe

## no findings -> step 3:
journalctl -u mariadb.service
## or journalctl -u mariadb

## no findings -> step 4:
## search specific log for service
## and eventually need to increase the log level
## e.g. with mariadb (find through internet research)
less /var/log/mysql/error.log

## Nicht fündig -> Schritt 5
## Allgemeines Log
## Debian/Ubuntu
/var/log/syslog
## Redhat/Centos
/var/log/messages

```

## Find errors in logs quickly

```

cd /var/log/mysql
## -i = case insensitive // egal ob gross- oder kleingeschrieben
cat error.log | grep -i error

```

## Find configuration - option in config - files

```
grep -r datadir /etc
```

# Galera

## Installation and Configuration (Centos/Redhat 8)

### Setting up 1st - node

```

## Schritt 1: Create config

/etc/my.cnf.d/z_galera.cnf
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
## Set to 1 sec instead of per transaction

```

```
## for better performance // Attention: You might loose data on power
innodb_flush_log_at_trx_commit=0
## Galera Provider Configuration
wsrep_on=ON
## centos7 (x86_64)
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so
## Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.56.103,192.168.56.104,192.168.56.105"
wsrep_node_address=192.168.56.103
## Galera Synchronization Configuration
wsrep_sst_method=rsync
```

## Stop the server and bootstrap cluster

```
## setup first node in cluster
systemctl stop mariadb
galera_new_cluster # statt systemctl start mariadb
```

## Check if cluster is running

```
mysql> show status like 'wsrep%'\G
***** 38. row *****
Variable_name: wsrep_local_state_comment
Value: Synced
***** 56. row *****
Variable_name: wsrep_cluster_size
Value: 1
***** 57. row *****
Variable_name: wsrep_cluster_state_uuid
Value: 562e5455-a40f-11eb-b8c9-1f32a94e106e
***** 58. row *****
Variable_name: wsrep_cluster_status
Value: Primary
***** 59. row *****
Variable_name: wsrep_connected
Value: ON
```

## Setup firwealld for galera

```
firewall-cmd --add-port=3306/tcp --permanent
firewall-cmd --add-port=4567/tcp --permanent
firewall-cmd --add-port=4568/tcp --permanent
firewall-cmd --add-port=4444/tcp --permanent
firewall-cmd --reload

firewall-cmd --add-port=3306/tcp --permanent; firewall-cmd --add-port=4567/tcp --
permanent; firewall-cmd --add-port=4568/tcp --permanent; firewall-cmd --add-
port=4444/tcp --permanent; firewall-cmd --reload
```



## 1. Node started nicht nach Crash, z.B. Stromausfall

### Warum startet nicht ?

```
## node ist in einem nicht-geordneten Zustand.  
## und hat Angst ;o), dass die anderen Nodes u.U. weiter sind  
## Ziel sollte sein, die letzte Node als 1. zu starten mit -> galera_new_cluster
```

### Wie beheben ?

```
## Nach Informationen im Status gucken  
systemctl status mariadb  
  
## Nach Informationen in den Logs schauen  
journalctl -u mariadb  
## Speziell kann ich rausfiltern  
journalctl -u mariadb | grep -i error  
  
## In der Regel steht safe_to_bootstrap auf 0  
  ä Fixend  
/var/lib/grastate.dat  
safe_to_bootstrap = 1 # setzen  
  
## Immer nur ausführen, wenn es nur eine Node 1 !! git  
galera-new-cluster
```

## Information Schema / Status / Processes

### Show server/session status

#### Through mysql

```
## in mysql interface (client)  
mysql  
status;
```

#### With mysqladmin

```
mysqladmin status  
## or if you want to know more  
mysqladmin extended status
```

#### with mysql -> show status

```
## Status within session (status - counters)  
mysql> show status;  
## Status global (since last reboot/start of mariadb server)  
mysql> show global status;  
mysql> -- reset session status  
mysql> flush status;
```

```
## Show session status
mysql> show session status;
```

## Kill long running process

```
## Session 1
## sleep for 120 seconds
select sleep(120)

## Session 2
show processlist
## kill process you have identified for sleep(120)
MariaDB [(none)]> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Id | User | Host      | db      | Command | Time | State      | Info                                |
Progress |
+----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 36 | root | localhost | NULL    | Query   | 0    | starting   | show processlist                  |
0.000 |
| 37 | root | localhost | training | Query   | 4    | User sleep | select sleep(120)                  |
0.000 |
+----+-----+-----+-----+-----+-----+-----+-----+
+-----+
2 rows in set (0.000 sec)

## take 37
kill 37

## Session 1: query terminates
ERROR 2013 (HY000): Lost connection to MySQL server during query
```

## Kill (kickout user) and stop server

```
MariaDB [mysql]> show processlist;
+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Id | User      | Host      | db      | Command | Time | State      | Info
+-----+
| 30 | root      | localhost | mysql   | Sleep   | 10   |            | NULL
| 0.000 |
| 34 | root      | localhost | mysql   | Query   | 0    | starting   | show processlist
| 0.000 |
| 43 | training  | localhost | training | Sleep   | 5    |            | NULL
| 0.000 |
+-----+-----+-----+-----+-----+-----+-----+
+-----+
3 rows in set (0.000 sec)
```

```
MariaDB [mysql]> quit
Bye
root@its-lu20s04:~# mysql -e 'kill 43' && systemctl stop mariadb
root@its-lu20s04:~#
```

## Security and User Rights

### Debug and Setup External Connection

#### Prerequisites

```
client1: 192.168.56.104
server1: 192.168.56.103
```

#### Step 1: Be sure server is communicating to the outside

```
lsof -i
## should be
*:mysql
```

#### Step 2: Test connection from client

```
mysqladmin ping -h 192.168.56.103
## on succesful connection also without authentication
echo $?
0 # 0 was success also without proper authentication

## Bad news, if
echo $?
1
## Could not connect at all
```

#### Step 2a: No connection possible ? check Firewall ....

```
## Server 1
systemctl status firewalld
firewall-cmd --state
firewall-cmd --list-all # do we see mysql as a service

## no ?
firewall-cmd --get-services
firewall-cmd --add-service=mysql # only in runtime
firewall-cmd --runtime-to-permanent # config - works after reboot

### Recheck with Step 2
```

#### Step 3: Setup user without grants - Server1

```
## Server 1
mysql> create user ext@192.168.56.104 identified by 'topsecretpassword';
```

```
## Doing this twice triggers an weird error
```

### Step 3a: test connection from client - Client 1

```
mysql -uext -p -h 192.168.56.103
## on success
mysql>show grants
## should only be usage
mysql>show schemas
```

### Step 3b: Add privileges (testing giving all) - Server1

```
## *.* = all databases and all tables
mysql> GRANT ALL ON *.* TO ext@192.168.56.104
```

### Step 3c: See, if we have grants - Client 1

```
mysql>show grants
## grants will be shown but do not work yet
## we need to reconnect
mysql>quit
mysql -uext -p -h 192.168.56.103
mysql> -- now it works
```

## Get Rights of user

### Root can show rights of a specific user

```
## shows the right of the logged in user (you as a user)
show grants;

## show grants for a specific user
## no need for ' (quotes) if there are not special chars withing
## e.g.
show grants for training@localhost;
## if there are special chars, use quotes
show grants for 'mariadb.sys'@localhost;

## if you want to see rights of a user that has rights from everywhere
show grants for training@'%';
```

### If you cannot remember the exact user (user@host) look it up

```
## within mysql client
use mysql
select * from user \G
```

## Secure with SSL server/client

### Variant 1: Setup 1-way ssl encryption

## Create CA and Server-Key

```
## On Server - create ca and certificates
sudo mkdir -p /etc/my.cnf.d/ssl
sudo cd /etc/my.cnf.d/ssl

## create ca.
sudo openssl genrsa 4096 > ca-key.pem

## create ca-certificate
## Common Name: MariaDB CA
sudo openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.pem

## create server-cert
## Common Name: server1.training.local
## Password: --- leave empty ---
sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout server-key.pem -out
server-req.pem

## Next process the rsa - key
sudo openssl rsa -in server-key.pem -out server-key.pem

## Now sign the key
sudo openssl x509 -req -in server-req.pem -days 365000 -CA ca-cert.pem -CAkey ca-
key.pem -set_serial 01 -out server-cert.pem
```

## Verify certificates

```
openssl verify -CAfile ca-cert.pem server-cert.pem
```

## Configure Server

```
## create file
## /etc/my.cnf.d/z_ssl.cnf
[mysqld]
ssl-ca=/etc/my.cnf.d/ssl/ca-cert.pem
ssl-cert=/etc/my.cnf.d/ssl/server-cert.pem
ssl-key=/etc/my.cnf.d/ssl/server-key.pem
### Set up TLS version here. For example TLS version 1.2 and 1.3 ##
## Starts from mariadb 10.4.6 not possible before. !!!!
tls_version = TLSv1.2,TLSv1.3

## Set ownership
chown -vR mysql:mysql /etc/my.cnf.d/ssl/
```

## Restart and check for errors

```
systemctl restart mariadb
journalctl -u mariadb
```

## Test connection on client

```
## only if we use option --ssl we will connect with ssl
mysql --ssl -uxyz -p -h <ip-of-server>
mysql>status
SSL:                               Cipher in use is TLS_AES_256_GCM_SHA384
```

## Force to use ssl

```
## on server
## now client can only connect, when using ssl
mysql> grant USAGE on *.* to remote@10.10.9.144 require ssl;
```

## Variant 2: 1-way ssl-encryption but checking server certificate

### Prerequisites

```
server1: 192.168.56.103
client1: 192.168.56.104
```

### Copy ca-cert to client

```
## on server1
cd /etc/my.cnf.d/ssl
scp ca-cert.pem kurs@192.168.56.104:/tmp

## on clien1
cd /etc/my.cnf.d
mkdir ssl
cd ssl
mv /tmp/ca-cert.pem .
```

### Configure client1 - client -config

```
sudo vi /etc/my.cnf.d/mysql-clients.cnf

Append/edit in [mysql] section:

### MySQL Client Configuration ##
ssl-ca=/etc/my.cnf.d/ssl/ca-cert.pem

### Force TLS version for client too
##tls_version = TLSv1.2,TLSv1.3
#### This option is disabled by default ####
#### ssl-verify-server-cert ###

## only works if you have no self-signed certificate
ssl-verify-server-cert
ssl

## domain-name in hosts setzen
## because in dns
```

```
vi /etc/hosts
192.168.56.103 server1.training.local

## now you to connect with hostname
## otherwise no check against certificate can be done
mysql -uext -p -h server1.training.local

## if it does not work, you get
ERROR 2026 (HY000): SSL connection error: Validation of SSL server certificate failed
```

### **Variant 3: 2-way - Security (Encryption) - validated on server and client**

#### **Client - Create certificate on server**

- we are using the same ca as on the server

```
## on server1
cd /etc/my.cnf.d/ssl
## Bitte Common-Name: MariaDB Client
openssl req -newkey rsa:2048 -days 365 -nodes -keyout client-key.pem -out client-req.pem

## process RSA - Key
## Eventually also works without - what does it do ?
## openssl rsa -in client-key.pem -out client-key.pem

## sign certificate with CA
openssl x509 -req -in client-req.pem -days 365 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out client-cert.pem
```

#### **Client - Zertifikate validieren**

```
openssl verify -CAfile ca-cert.pem client-cert.pem
```

#### **Zertifikate für Client zusammenpacken**

```
mkdir cl-certs; cp -a client* cl-certs; cp -a ca-cert.pem cl-certs ; tar cvfz cl-certs.tar.gz cl-certs
```

#### **Zertifikate auf Client transferieren**

```
scp cl-certs.tar.gz kurs@192.168.56.104:/tmp
```

#### **Zertifikate einrichten**

```
## on client1
## cleanup old config
rm /etc/my.cnf.d/ssl/ca-cert.pem

mv /tmp/cl-certs.tar.gz /etc/my.cnf.d/ssl
cd /etc/my.cnf.d; tar xzvf cl-certs.tar.gz
```

```
vi mysql-clients.cnf
[mysql]
ssl-ca=/etc/my.cnf.d/cl-certs/ca-cert.pem
ssl-cert=/etc/my.cnf.d/cl-certs/client-cert.pem
ssl-key=/etc/my.cnf.d/cl-certs/client-key.pem
```

## Test the certificate

```
## on server1 verify: X509 for user
select user,ssl_type from mysql.user where user='ext'

## connect from client1
## Sollte die Verbindung nicht klappen stimmt auf dem
## Client etwas mit der Einrichtung nicht
mysql -uext -p -h192.168.56.103
mysql> status
```

## Ref

- <https://www.cyberciti.biz/faq/how-to-setup-mariadb-ssl-and-secure-connections-from-clients/>

## Auth with unix\_socket

```
mysql>create user training@localhost identified via unix_socket
useradd training
passwd training

## testing
su - training
## mysql
## should not work without password
## Be sure, that user has access to socket
cd /var/lib/mysql
ls -la mysql.socket
```

## User- and Permission-concepts (best-practice)

```
## user should have as little permissions as possible
## so many as needed ;o)
MariaDB [mysql]> create database eventplanner;
Query OK, 1 row affected (0.000 sec)

MariaDB [mysql]> create user eventplanner@localhost identified by 'eventplanner';
Query OK, 0 rows affected (0.001 sec)

MariaDB [mysql]> grant all on eventplanner.* to eventplanner@localhost;
Query OK, 0 rows affected (0.003 sec)
```

## Setup external access

### Testing



```
## Where .104 is the server you want to connect to
## Variante 1
mysqladmin ping -h 192.168.56.104
echo $?
-> 0 // it is possible to reach mysql - server

## Variante 2
mysqladmin ping -h 192.168.56.104
echo $?
-> 1 // i cannot reach mysql-server -> port might close / firewall ?

## or use telnet
telnet 192.168.56.104 3306
```

## Checks on MariaDB (Theory)

- Is MariaDB - Server running ?
- Is 3306 port open (exposed to the outside)
- Is firewall open for port 3306
- Is there a valid user, who connect)

## Checks on MariaDB (Practical)

```
## Step 1: Running
systemctl status mariadb
## Step 2: Port open ?
lsof -i # does it listen to all interfaces. -> *
        # or an external interface
## Step 3: Firewall open -> see next block
## Step 4: User who can connect ?
```

## Checks on Firewall.

```
## Is firewall running and enabled
systemctl status firewalld
firewall-cmd --state

## Is interface setup for usage of firewalld
firewall-cmd --get-active-zones

## Is service "mysql" in zones
firewall-cmd --list-all-zones | less # is it within public - zone -> mysql

## To enable it, if not set
firewall-cmd --add-service=mysql --zone=public --permanent # writes to filesystem
config
firewall-cmd --reload # rereads settings from filesystem
```

## Setup valid user

```
## on server you want to connect to
mysql> create user extern@'192.168.56.%' identified by 'mysecretpass'
```

```
mysql> grant all on sakila.* to extern@'192.168.56.56'
```

## Now test from external with mysql

```
mysql -uextern -p -h 192.168.56.104  
mysql> show databases;
```

## Table encryption

### Step 1: Set up keys

```
mkdir -p /etc/mysql/encryption;  
echo "1;$(openssl rand -hex 32) > /etc/mysql/encryption/keyfile;  
  
openssl rand -hex 128 > /etc/mysql/encryption/keyfile.key;  
openssl enc -aes-256-cbc -md sha1 -pass file:/etc/mysql/encryption/keyfile.key -in  
/etc/mysql/encryption/keyfile -out /etc/mysql/encryption/keyfile.enc;  
  
rm -f /etc/mysql/encryption/keyfile;  
  
chown -R mysql:mysql /etc/mysql;  
chmod -R 500 /etc/mysql;
```

### Step 2: Verify data before encryption

```
cd /var/lib/mysql/mysql  
strings gtid_slave_pos.ibd
```

### Step 3: Setup configuration

```
## vi /etc/my.cnf.d/z_encryption.cnf  
  
[mysqld]  
plugin_load_add = file_key_management  
file_key_management_filename = /etc/mysql/encryption/keyfile.enc  
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key  
file_key_management_encryption_algorithm = AES_CTR  
  
innodb_encrypt_tables = FORCE  
innodb_encrypt_log = ON  
innodb_encrypt_temporary_tables = ON  
  
encrypt_tmp_disk_tables = ON  
encrypt_tmp_files = ON  
encrypt_binlog = ON  
aria_encrypt_tables = ON  
  
innodb_encryption_threads = 4  
innodb_encryption_rotation_iops = 2000
```

#### Step 4: Restart server

```
systemctl restart mariadb
```

#### Step 5: Verify encryption

```
cd /var/lib/mysql/mysql
strings gtid_slave_pos;

use information_schema;
select * from innodb_tablespace_encryption;
SELECT CASE WHEN INSTR(NAME, '/') = 0
            THEN '01-SYSTEM TABLESPACES'
            ELSE CONCAT('02-', SUBSTR(NAME, 1, INSTR(NAME, '/')-1)) END
        AS "Schema Name",
        SUM(CASE WHEN ENCRYPTION_SCHEME > 0 THEN 1 ELSE 0 END) "Tables Encrypted",
        SUM(CASE WHEN ENCRYPTION_SCHEME = 0 THEN 1 ELSE 0 END) "Tables Not Encrypted"
FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
GROUP BY CASE WHEN INSTR(NAME, '/') = 0
            THEN '01-SYSTEM TABLESPACES'
            ELSE CONCAT('02-', SUBSTR(NAME, 1, INSTR(NAME, '/')-1)) END
ORDER BY 1;
```

#### Step 6: disable encryption runtime

```
SET GLOBAL innodb_encrypt_tables = OFF;
```

#### Ref:

- <https://mariadb.com/de/resources/blog/mariadb-encryption-tde-using-mariadbs-file-key-management-encryption-plugin/>

## SELinux

### Welche Ports sind freigegeben? (MariaDb startet damit)

#### Welche Ports

```
semanage port -l | grep mysql
```

### Neues Datenverzeichnis SELinux bekanntmachen - semanage fcontext

```
mkdir /data
chown mysql:mysql /data
semanage fcontext -a -t mysqld_db_t "/data(/.*)?"
restorecon -vr /data
## type _t should mysqld_db_t
ls -laZ
```

### Probleme mit SELinux erkennen und debuggen

```

## Wenn mariadb nicht startet, dann zunächst Loganalyse
systemctl status mariadb
## Gibt es ERROR - Einträge ?
## Gibt es Permission / Access Denied - Einträge
## Wenn ansonsten alle Rechte stimmen, weiss das Probleme mit SELinux
journalctl -u mariadb | less

## Logs von selinux laufen über den Audit-Daemon
/var/log/audit/audit.log
## Dies können mit sealert analysiert werden
## Wichtig: Geduld haben, die Analyse dauert einen Moment
## auch nach 100% noch abwarten
sealert -a /var/log/audit/audit.log

## Allheilmittel ist meistens
## Setzt den richtigen Context, den SELinux braucht,
## damit mariadb starten kann
restorecon -rv /var/lib/mysql

```

## Database - Objects

### Triggers

#### Create the structure

```

create table countries (
    country_id int auto_increment,
    name varchar(50) not null,
    primary key(country_id)
);

INSERT INTO countries (name) values ('Germany'), ('Austria');

create table country_stats(
    country_id int,
    year int,
    population int,
    primary key (country_id, year),
    foreign key(country_id)
    references countries(country_id)
);

INSERT INTO country_stats (country_id, year, population) values (1,2020,100000);

create table population_logs(
    log_id int auto_increment,
    country_id int not null,
    year int not null,
    old_population int not null,
    new_population int not null,

```

```
    updated_at timestamp default current_timestamp,  
    primary key(log_id)  
);
```

## Create the trigger

```
create trigger before_country_stats_update  
  before update on country_stats  
  for each row  
  insert into population_logs(  
    country_id,  
    year,  
    old_population,  
    new_population  
  )  
  values(  
    old.country_id,  
    old.year,  
    old.population,  
    new.population  
  );
```

## Create trigger (the same) but with BEGIN/END - Block

```
create trigger before_country_stats_update  
  before update on country_stats  
  for each row  
  
  BEGIN  
  SET @anfang = 1;  
  insert into population_logs(  
    country_id,  
    year,  
    old_population,  
    new_population  
  )  
  values(  
    old.country_id,  
    old.year,  
    old.population,  
    new.population  
  );  
  END
```

## Run a test

```
update  
  country_stats  
set  
  population = 1352617399
```

```

where
    country_id = 1 and
    year = 2020;

-- what's the new result

select * from population_logs;

```

## Ref:

- <https://mariadb.com/kb/en/trigger-overview/>

## Ref with walkthrough

- <https://mariadb.com/kb/en/trigger-overview/>

## Functions

```

CREATE FUNCTION hello (s CHAR(20))
    RETURNS CHAR(50) DETERMINISTIC
    RETURN CONCAT('Hello, ',s,'!');

```

```

SELECT hello('world');

```

```

+-----+
| hello('world') |
+-----+

```

```

+-----+
| Hello, world! |
+-----+

```

```

+-----+

```

```

+-----+

```

```

+-----+

```

```

### Stored Procedure

```

```

### Example

```

```

USE sakila; DELIMITER //

```

```

CREATE PROCEDURE simpleproc (OUT param1 INT) BEGIN SELECT COUNT(*) INTO param1 FROM
actor; END; //

```

```

DELIMITER ;

```

```

CALL simpleproc(@a);

```

```

SELECT @a; +-----+ | @a | +-----+ | 1 | +-----+

```

```

### Reference

```

```

* https://mariadb.com/kb/en/create-procedure/

```

```

### Events

```

```

### Preparation

```

```
-- scheduler is not there SHOW PROCESSLIST;

-- Prüfen ob scheduler läuft show variables like '%event%'; set GLOBAL event_scheduler = on;

-- scheduler appears SHOW PROCESSLIST;

-- Events anzeigen show events;
```

```
### preparation
```

```
USE schulung; CREATE TABLE messages ( id INT PRIMARY KEY AUTO_INCREMENT, message
VARCHAR(255) NOT NULL, created_at DATETIME NOT NULL );
```

```
### One time event
```

```
USE schulung CREATE EVENT IF NOT EXISTS test_event_01 ON SCHEDULE AT CURRENT_TIMESTAMP
DO INSERT INTO messages(message,created_at) VALUES('Test MariaDB Event 1',NOW());
```

```
SELECT * FROM messages;
```

```
### Show all events from a specific database
```

```
SHOW EVENTS FROM schulung;
```

```
### Show all events in active database
```

```
USE schulung; SHOW EVENTS;
```

```
### One time event but preserved (so runs once every minute)
```

To keep the event after it is expired, you use the ON COMPLETION PRESERVE clause.

```
CREATE EVENT test_event_02 ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MINUTE ON
COMPLETION PRESERVE DO INSERT INTO messages(message,created_at) VALUES('Test MariaDB Event
2',NOW());
```

```
### Same version, but with begin end block
```

```
DELIMITER / CREATE EVENT test_event_03 ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1
MINUTE ON COMPLETION PRESERVE DO BEGIN INSERT INTO messages(message,created_at)
VALUES('Test MariaDB Event 3',NOW()); END / DELIMITER ;
```

```
SELECT * FROM messages;
```

```
### Recurring Example
```

```
CREATE EVENT test_event_03 ON SCHEDULE EVERY 1 MINUTE STARTS CURRENT_TIMESTAMP ENDS
CURRENT_TIMESTAMP + INTERVAL 1 HOUR DO INSERT INTO messages(message,created_at)
VALUES('Test MariaDB recurring Event',NOW());
```

```
SELECT * FROM messages;
```

```
// after 1 minute SELECT * FROM messages;
```

```
### Drop an event
```

```
DROP EVENT IF EXIST test_event_03;
```

```
### Set event-scheduler in config / my.cnf / my.ini
```

```
[mysqld] event-scheduler
```

## after that restawrt

```
systemctl restart mariadb
```

```
### Fix timezone problem Linux (when time is displayed wrong)
```

## 09:32 UTC should be 11:32 CEST

## also root ausführen

```
timedatectl list-timezones | grep 'Europe/Berlin'; timedatectl set-timezone Europe/Berlin  
timedatectl date systemctl restart mariadb mysql mysql>select now(); mysql>--- time should ok now
```

```
### Data Types
```

```
* https://mariadb.com/kb/en/data-types/
```

```
## Locking
```

```
### Implicit Locks
```

```
### How do the work in general
```

```
* Implicit locks are done by InnoDB itself  
* We can only partly influence them.
```

```
### Who wants what ?
```

<who?, what?, how?, granted?>

```
### Explanation (a bit clumsy)
```

```
* IS and IX (intended share an intended write lock)
```



- \* IS and IX can be triggered on SQL
- \* IX -> SUFFIX -> FOR UPDATE (this triggers a IX lock)
- \* IX and IS are the first step (on table layer)
- \* After that IX -> tries to get a write lock on row-level -> X
- \* Works unless there is another X
- \* IX and IS is not retrieved on TABLE spaced operations (construction --- alter)

### Lock Type compability matrix

| X | IX | S | IS |
|---|----|---|----|
|---|----|---|----|

|             |            |             |            |
|-------------|------------|-------------|------------|
| X Conflict  | Conflict   | Conflict    | Conflict   |
| IX Conflict | Compatible | Conflict    | Compatible |
| S Conflict  | Compatible | IS Conflict | Compatible |
| IS Conflict | Compatible | Compatible  | Compatible |

### The best explanation across the internet ;o)

\* <http://stackoverflow.com/questions/25903764/why-is-an-ix-lock-compatible-with-another-ix-lock-in-innodb>|IX\_and\_IS-locks

Many people, both visitors and curators, enter the museum. The visitors want to view paintings, so they wear a badge labeled "IS". The curators may replace paintings, so they wear a badge labeled "IX". There can be many people in the museum at the same time, with both types of badges. They don't block each other.

During their visit, the serious art fans will get as close to the painting as they can, and study it for lengthy periods.

They're happy to let other art fans stand next to them before the same painting. They therefore are doing SELECT ... LOCK IN SHARE MODE and they have "S" lock, because they at least don't want the painting to be replaced while they're studying it.

The curators can replace a painting, but they are courteous to the serious art fans, and they'll wait until these viewers are done and move on. So they are trying to do SELECT ... FOR UPDATE (or else simply UPDATE or DELETE). They will acquire "X" locks at this time, by hanging a little sign up saying "exhibit being redesigned." The serious art fans want to see the art presented in a proper manner, with nice lighting and some descriptive placque. They'll wait for the redesign to be done before they approach (they get a lock wait if they try).

### Identify Deadlocks in innodb

### Prerequisite

2 sessions (connected to same server): Session 1 Session 2

sakila database is installed

### Session 1:

## Start transaction and lock row by updating it

```
mysql>use sakila; mysql>begin; mysql>update actor set last_name='Johnsson' where actor_id = 200
```

## Attention: not commit yet please, leave transaction open

```
### Session 2:
```

## Start transactio and try to update same row

```
mysql>use sakila; mysql>begin; mysql>update actor set last_name='John' where actor_id = 2000
```

## Now update cannot be done, because of lock from session one

```
### Session 1: / or new Session 3
```

## find out who blocks session 2

```
mysql>use information_schema;
```

## find out trx\_id of session 2

```
mysql>select * from innodb_trx;
```

## assuming we have trx\_id 1468;

## now we find out what is blocking this transaction

```
mysql>select * from innodb_locks_wait; MariaDB [information_schema]> select * from
innodb_lock_waits; +-----+-----+-----+-----+ | requesting_trx_id |
requested_lock_id | blocking_trx_id | blocking_lock_id | +-----+-----+-----+-----+
-----+ | 1469 | 1469:66:3:201 | 1468 | 1468:66:3:201 | +-----+-----+-----+
-----+-----+ 1 row in set (0.001 sec)
```

## either additional infos

```
select + from innodb_trx where trx_id = 1468;
```

## or directly kill this transaction

```
show processlist; kill 1468;
```

```
### Refs ( 3 important tables )
```

```
* https://mariadb.com/kb/en/information-schema-innodb\_lock\_waits-table/ (most
```

```
important one)
* https://mariadb.com/kb/en/information-schema-innodb_locks-table/
* https://mariadb.com/kb/en/information-schema-innodb_trx-table/
```

## Upgrade

### MariaDB Upgrade 10.3 (Centos) -> 10.4 (Mariadb.org)

### Prerequisites

Ubuntu 20.04 MariaDB-Server from Distri

Install new 10.4 from Mariadb.org

### Prepare

```
* Create backup of system (with mariabackup and/or mysqldump)
```

### Steps

**1. systemctl stop mariadb**

**2. dnf remove mariadb-\***

**3. Doublecheck if components left: apt list --installed | grep mariadb**

**4. Setup repo for mariadb**

**5. dnf install MariaDB-server**

**7. systemctl enable --now mariadb # enable for next reboot and start immediately**

**necessary for redhat**

**8. Perform mysql\_upgrade**

**On centos/redhat mysql\_upgrade need to be done**

mysql\_upgrade

**9. Check if it was successful**

cat /var/lib/mysql\_upgrade\_info

### Important - Check mysql - configuration structure

## Which directories are loaded in

/etc/mysql/my.cnf

## Eventually move files to the right directory

## As needed in migration from 10.3 (Distri) to 10.4 (mariadb.org) on Ubuntu 20.04

```
### Documentation

* https://mariadb.com/kb/en/upgrading-from-mariadb-103-to-mariadb-104/
* https://mariadb.com/kb/en/mysql_upgrade/

### MariaDB Upgrade 10.4 -> 10.5 (Centos)
```

### Step 0;

### Sicherung anlegen (mysqldump / mariabackup)

### Step 1:

### Change version in

### or where you have your repo definition

### Change 10.4 -> 10.5

/etc/yum.repos./MariaDB.repo

### Step 2:

systemctl stop mariadb

### Step 3

sudo yum remove MariaDB-server

### Step 4

sudo yum install MariaDB-server yum list --installed | grep MariaDB # sind alle Versionen gleich !  
Wichtig ! sudo yum update ## Achtung: abweichend von Doku MariaDB

### Step 4.5

## Check if old config files were saved as .rpmsave after delete of package 10.4

```
cd /etc/my.cnf.d ls -la
```

**e.g.**

```
mv server.cnf.rpmsave server.cnf
```

### Step 5:

systemctl start mariadb systemctl enable mariadb mysql\_upgrade # After that mysql\_upgrade\_info will be present in /var/lib/mysql with version-info

```
## Reference

* https://mariadb.com/kb/en/upgrading-from-mariadb-104-to-mariadb-105/

### MariaDB Upgrade 5.5 -> 10.5

* https://mariadb.com/kb/en/upgrading-between-major-mariadb-versions/

## Performance

### io-Last/CPU-Last

### IO-gebundene - Last (Input/Output)
```

Gegeben wenn:

- Hoher waiting wert in top (wa-wert in CPU-Liste)
- ◦ Hohelast 1,5, 15 min 1,2 1.5 2 (Load) -> top

### CPU-Gebundene - Last

```
Gegeben wenn:
- NUR: Hohe Last -> Wert in top -> 2 1.5 0.5 (Load)
- Waiting-wert: 0
```

## Views and performance

### General

```
SHOW CREATE VIEW
```

### Views and Algorithms

- Views can use 3 algorithms:

- merge
- simple rewrites (translates the query)
- temptable
  - Creates a temptable to retrieve information
  - In this case no indexes can be used
- Shows up explain with derived
- undefined
  - MySQL chooses, if to use merge or temptable
  - prefers merge over temptable if possible

## Example

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| id | select_type | table      | type | possible_keys | key  | key_len | ref  | rows |
| Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 1 | PRIMARY     | <derived2> | ALL  | NULL          | NULL | NULL    | NULL | 33   |
| NULL |
| 2 | DERIVED     | task       | ALL  | NULL          | NULL | NULL    | NULL | 33   |
| NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
```

## Handling (best practice)

- You can define the algorithm when creating the view
- If you define merge and mysql cannot handle it
  - you will get a warning

## Example of handling

```
mysql> CREATE ALGORITHM=MERGE VIEW priority_counts AS SELECT priority_id, COUNT(1) AS
quantity FROM task GROUP BY priority_id;
Query OK, 0 rows affected, 1 warning (0.12 sec)
```

```
mysql> SHOW WARNINGS;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Level  | Code | Message
|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Warning | 1354 | View merge algorithm can't be used here for now (assumed undefined
algorithm) |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
1 row in set (0.08 sec)
```

## Reference

- Ref: <https://dba.stackexchange.com/questions/54481/determining-what-algorithm-mysql-view-is-using>

## Partitions and Explain

### Walkthrough

```
-- EXPLAIN PARTITIONS

DROP TABLE IF EXISTS audit_log;
CREATE TABLE audit_log (
  yr    YEAR NOT NULL,
  msg   VARCHAR(100) NOT NULL)
ENGINE=InnoDB
PARTITION BY RANGE (yr) (
  PARTITION p0 VALUES LESS THAN (2010),
  PARTITION p1 VALUES LESS THAN (2011),
  PARTITION p2 VALUES LESS THAN (2012),
  PARTITION p3 VALUES LESS THAN MAXVALUE);
INSERT INTO audit_log(yr,msg) VALUES (2005,'2005'),(2006,'2006'),(2011,'2011'),
(2020,'2020');
EXPLAIN PARTITIONS SELECT * from audit_log WHERE yr in (2011,2012)\G
```

### Partitions sliced by hash of field

```
CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(30),
  lname VARCHAR(30),
  hired DATE NOT NULL DEFAULT '1970-01-01',
  separated DATE NOT NULL DEFAULT '9999-12-31',
  job_code INT,
  store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

## 3 Phases of DataSize

### Phase 1: Table content is small (only some rows)

```
## table scan is quicker than index search
## e.g. 10 entries

## so eventually index is not needed
```

### Phase 2: Index is good !!

```
## performance gain by using index
## Step 1: Obtaining id's from index (primary key id)
## Step 2: Retrieving data
```

### Phase 3: Index is not improve performance / or would makes performance worse

```
Step 1: lookup in index:
1
70
1040
2100
35000
-> there is a lot of space (other rows) in between.

Step 2: Lookup data, but a lot lookups needed

-> random reads
-> So mysql might be better off to do a table scan.
```

### Slow Query Log

#### Walkthrough

```
## Step 1
/etc/my.cnf.d/mariadb-server.cnf
[mysqld]
slow-query-log

## Step 2
mysql>SET GLOBAL slow_query_log = 1
mysql>SET slow_query_log = 1
mysql>SET GLOBAL long_query_time = 0.000001
mysql>SET long_query_time = 0.000001

## Step 3
## run some time / data
## and look into your slow-query-log
/var/lib/mysql/hostname-slow.log
```

### Show queries that do not use indexes

```
SET GLOBAL log_queries_not_using_indexes=ON;
```

### Geschwätzigkeit (Verbosity) erhöhen

```
SET GLOBAL log_slow_verbosity='query_plan,explain'
```

### Reference

- <https://mariadb.com/kb/en/slow-query-log-overview/>

### Optimal use of indexes



## Describe and indexes

### Walkthrough

#### Step 1:

```
## Database and Table with primary key
create database descindex;
use descindex;
create table people (id int unsigned auto_increment, first_name varchar(25), last_name
varchar(25), primary key (id), passcode mediumint unsigned);
## add an index
## This will always !! translate into an alter statement.
create index idx_last_name_first_name on people (last_name,first_name)
##
create unique index idx_passcode on people (passcode)

desc people;
```

| Field      | Type                  | Null | Key | Default | Extra          |
|------------|-----------------------|------|-----|---------|----------------|
| id         | int(10) unsigned      | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(25)           | YES  |     | NULL    |                |
| last_name  | varchar(25)           | YES  |     | NULL    |                |
| passcode   | mediumint(8) unsigned | YES  |     | NULL    |                |

4 rows in set (0.01 sec)

#### Step 2:

```
## Add simple combined index on first_name, last_name
create index idx_first_name_last_name on people (first_name, last_name);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0
desc people;
```

-- show the column where the combined index starts (MUL = Multi)

| Field      | Type                  | Null | Key | Default | Extra          |
|------------|-----------------------|------|-----|---------|----------------|
| id         | int(10) unsigned      | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(25)           | YES  | MUL | NULL    |                |
| last_name  | varchar(25)           | YES  |     | NULL    |                |
| passcode   | mediumint(8) unsigned | YES  |     | NULL    |                |

4 rows in set (0.01 sec)

#### Step 3:

```
## Add a unique index on passcode
create index idx_passcode on people (passcode)
mysql> desc people;
```

-- Line with UNI shows this indexes.

| Field      | Type                  | Null | Key | Default | Extra          |
|------------|-----------------------|------|-----|---------|----------------|
| id         | int(10) unsigned      | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(25)           | YES  | MUL | NULL    |                |
| last_name  | varchar(25)           | YES  |     | NULL    |                |
| passcode   | mediumint(8) unsigned | YES  | UNI | NULL    |                |

4 rows in set (0.01 sec)

#### Step 4:

```
## Get to know all your indexes on a table
show indexes for people
mysql> show index from people;
```

| Table  | Non_unique | Key_name                 | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
|--------|------------|--------------------------|--------------|-------------|-----------|-------------|----------|--------|------|------------|---------|---------------|
| people | 0          | PRIMARY                  | 1            | id          | A         |             |          |        |      |            |         |               |
| people | 0          | idx_passcode             | 1            | passcode    | A         |             |          |        |      |            |         |               |
| people | 1          | idx_first_name_last_name | 1            | first_name  | A         |             |          |        |      |            |         |               |
| people | 1          | idx_first_name_last_name | 2            | last_name   | A         |             |          |        |      |            |         |               |

4 rows in set (0.01 sec)

#### Find out indexes

##### Show index from table

```
create database showindex;
use showindex;
CREATE TABLE `people` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(25) DEFAULT NULL,
  `last_name` varchar(25) DEFAULT NULL,
  `passcode` mediumint(8) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_passcode` (`passcode`),
```

```
KEY `idx_first_name_last_name` (`first_name`,`last_name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
show index from people
```

Show create table

```
show create table peple
```

show index from

```
show index from contributions
```

## Index and Functions

**No function can be used on an index:**

```

explain select * from actor where upper(last_name) like 'A%';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
| rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | actor | NULL | ALL | NULL | NULL | NULL | NULL |
| 200 | 100.00 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Workaround with generated columns

```
## 1. Create Virtual Column with upper  
MariaDB [sakila](45) AS (upper(last_name_upper)) VIRTUAL;  
Query OK, 0 rows affected (0.006 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [sakila](last_name_upper);  
Query OK, 0 rows affected (0.008 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [sakila]> explain select * from actor where last_name_upper like 'A%';  
+-----+-----+-----+-----+-----+-----+-----+-----+  
--+-+-----+-----+  
| id | select_type | table | type | possible_keys | key | key_len | ref  
| rows | Extra |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
--+-+-----+-----+  
| 1 | SIMPLE | actor | range | idx_upper | idx_upper | 183 | NUL  
L | 7 | Using where |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
--+-+-----+-----+  
1 row in set (0.001 sec)
```

## Now we try to search the very same

```
explain select * from actor where last_name_upper like 'A%';
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key |
| key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | actor | NULL | range | idx_last_name_upper |
idx_last_name_upper | 183 | NULL | 7 | 100.00 | Using where |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

## Reference

- <https://mariadb.com/kb/en/generated-columns/>
- <https://mariadb.com/kb/en/slow-query-log-overview/>

## Index and Likes

### 1. like 'Will%' - Index works

```
explain select last_name from donors where last_name like 'Will%';
```

### 2. like '%iams' - Index does not work

```
-- because like starts with a wildcard
explain select last_name from donors where last_name like '%iams';
```

### 3. How to fix 3, if you are using this often ?

```
## Walkthrough
## Step 1: modify table
alter table donors add last_name_reversed varchar(70) GENERATED ALWAYS AS
(reverse(last_name));
create index idx_last_name_reversed on donors (last_name_reversed);

## besser - Variante 2 - untested
alter table donors add last_name_reversed varchar(70) GENERATED ALWAYS AS
(reverse(last_name)), add index idx_last_name_reversed on donors (last_name_reversed);

## Step 2: update table - this take a while
update donors set last_name_reversed = reversed(last_name)
## Step 3: work with it
select last_name,last_name_reversed from donor where last_name_reversed like
reverse('%iams');

## Version 2 with pt-online-schema-change
```

## profiling-get-time-for-execution-of.query

- Get better values, how long queries take

## Example

```

set profiling = 1
## Step 2 - Execute query
select last_name as gross from donors where last_name like lower('WILLI%')

## Step 3 - Show profiles
show profiles;
+-----+-----+-----+
+-----+
| Query_ID | Duration | Query
|
+-----+-----+-----+
+-----+
|          1 | 0.01993525 | select last_name as gross from donors where last_name like
lower('WILLI%') |
4 rows in set, 1 warning (0.00 sec)

## Step 4 - Show profile for a specific query
mysql> show profile for query 1;
+-----+-----+
+-----+-----+
| Status | Duration |
+-----+-----+
| starting | 0.000062 |
| checking permissions | 0.000006 |
| Opening tables | 0.000021 |
| init | 0.000017 |
| System lock | 0.000007 |
| optimizing | 0.000007 |
| statistics | 0.000083 |
| preparing | 0.000012 |
| executing | 0.000004 |
| Sending data | 0.022251 |
| end | 0.000005 |
| query end | 0.000008 |
| closing tables | 0.000007 |
| freeing items | 0.001792 |
| cleaning up | 0.000016 |
+-----+-----+
15 rows in set, 1 warning (0.00 sec)

```

## Find out cardinality without index

## Find out cardinality without creating index

```

select count(distinct donor_id) from contributions;

select count(distinct vendor_city) from contributions;
+-----+
| count(distinct vendor_city) |

```

```
+-----+
|                  1772 |
+-----+
1 row in set (4.97 sec)
```

## Replication

### Slave einrichten - Centos - old style (master\_pos)

#### Configure master

```
## /etc/my.cnf.d/server.cnf
## /etc/my.cnf.d/mariadb-server.cnf
[mysqld]
log-bin=mariadb-bin
server_id=1
log-basename=master1
binlog-format=row

systemctl stop mariadb
systemctl start mariadb
```

#### Setup replication user on master

```
CREATE USER 'replication_user'@'%' IDENTIFIED BY 'bigs3cret';
GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'%';
```

#### Slave aufsetzen

```
### Wichtig: möglichst gleiche Version

## repo einrichten von mariadb

## /etc/yum.repos.d/mariadb.repo
## MariaDB 10.4 CentOS repository list - created 2022-01-14 08:34 UTC
## https://mariadb.org/download/
[mariadb]
name = MariaDB
baseurl = https://mirror.kumi.systems/mariadb/yum/10.4/centos8-amd64
module_hotfixes=1
gpgkey=https://mirror.kumi.systems/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
## MariaDB 10.4 CentOS repository list - created 2022-01-14 08:34 UTC
## https://mariadb.org/download/
[mariadb]
name = MariaDB
baseurl = https://mirror.kumi.systems/mariadb/yum/10.4/centos8-amd64
module_hotfixes=1
gpgkey=https://mirror.kumi.systems/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

```

## MariaDB - Server installieren
dnf install -y MariaDB-server

## server - config von master rüberspielen
## auf server
cd /etc
tar cvfz my.cnf.d.tar.gz my.cnf.d
scp my.cnf.d.tar.gz kurs@192.168.56.104:/tmp

## auf slave ausrollen
cd /etc
mv my.cnf.d my.cnf.d.bkup
mv /tmp/my.cnf.d.tar.gz .
tar cvfz my.cnf.d.tar.gz

## config anpassen
## /etc/my.cnf.d/server.cnf
## pr: /etc/my.cnf.d/mariadb-server.cnf
[mysqld]

innodb-buffer-pool-size=3G
innodb-flush-method=0_DIRECT

## Enable slow-query-log
slow-query-log

server_id=2

## only necessary, if you want the slave to
## become master later on
log-bin=mariadb-bin
binlog-format=row
log-slave-updates=1
log-basename=slave1

## server restarten
systemctl stop mariadb
systemctl start mariadb

```

## backup auf master ausspielen und auf slave kopieren

```

mysqldump --all-databases --single-transaction --events --routines --master-data=2 --
flush-logs --delete-master-logs > /usr/src/master-dump.sql

scp /usr/src/master-dump.sql kurs@192.168.56.104:/tmp

```

## auf slave backup einspielen einspielen und konfigurieren

```
## vi /root/.my.cnf
[client]
password=mysupersecret

mv /tmp/master-dump.sql /usr/src
mysql < master-dump.sql

## vi /root/master.sql
CHANGE MASTER TO
  MASTER_HOST='192.168.56.103',
  MASTER_USER='replication_user',
  MASTER_PASSWORD='big3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master1-bin.000002',
  MASTER_LOG_POS=389,
  MASTER_CONNECT_RETRY=10;

mysql < /root/master.sql
```

## slave starten

```
mysql>slave start
mysql>show slave status \G
```

## Ref:

- <https://mariadb.com/kb/en/setting-up-replication/>

## Skip-Counter

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
START SLAVE;
```

## MaxScale installieren

### Why do Loadbalancing with MaxScale ?

- Cluster node transparent to application
  - Application does not see single nodes
- If one node fails you will have no downtime
  - In opposite: To talking to this node directly

### License Implications since 2.x

- MariaDB MaxScale >= 2.0 is licensed under MariaDB BSL.
- maximum of three servers in a commercial context.
  - Any more, and you'll need to buy their commercial license.
- MariaDB MaxScale 2.1.0 will be released under BSL 1.1 from the start
- Each release transitions in about max 4 years to GPL



## The MaxScale load-balancer and its components

- Routers
- Listeners
- Filters
- Servers (backend database server)

### Filters

- Logging Filters
- Statement rewriting filters
- Result set manipulation filters
- Firewall filter
- Pipeline control filters
  - e.g. tee and send to a second server
- Ref: <https://mariadb.com/kb/en/mariadb-maxscale-25-regex-filter/>

### Documentation - maxctrl

- <https://mariadb.com/kb/en/mariadb-maxscale-25-maxctrl/>

## Installation and Setup

### Installation

```
apt update
apt install apt-transport-https curl

## Setting up the repos
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
## Installing maxscale
apt install maxscale
```

### Setup (Part 1: MaxScale db-user)

- Do this on one of the galera nodes
- Adjust IP !!

```
## IP FROM MAXSCALE
## Setup privileges on cluster nodes
## It is sufficient to set it on one node, because
## it will be synced to all the other nodes
## on node 1
CREATE USER 'maxscale'@'10.10.11.139' IDENTIFIED BY 'P@ssw0rd';
##
GRANT SELECT ON mysql.db TO 'maxscale'@'10.10.11.139';
GRANT SELECT ON mysql.user TO 'maxscale'@'10.10.11.139';
GRANT SELECT ON mysql.tables_priv TO 'maxscale'@'10.10.11.139';
##
GRANT SELECT ON mysql.columns_priv TO 'maxscale'@'10.10.11.139';
GRANT SELECT ON mysql.proxies_priv TO 'maxscale'@'10.10.11.139';
```

```
##
GRANT SHOW DATABASES ON *.* TO 'maxscale'@'10.10.11.139';
## Needed for maxscale
GRANT SELECT ON mysql.procs_priv TO 'maxscale'@'10.10.11.139';
GRANT SELECT ON mysql.roles_mapping TO 'maxscale'@'10.10.11.139';

## Additionally for cluster operations (rejoin, switchover, failover for master/slave
replications
## these permissions are needed
GRANT super, reload, process, show databases, event on *.* to
'maxscale'@'10.10.11.139';
## GRANT select on mysql.user to 'maxscale'@'10.10.11.139';
```

```
## On maxscale - server
apt update
apt install mariadb-client
## Test the connection
## Verbindung sollte aufgebaut werden
mysql -u maxscale -p -h <ip-eines-der-nodes>
mysql>show databases
```

## SETUP (PART 2: CONFIGURATION)

```
## /etc/maxscale.cnf

[maxscale]

threads=auto
syslog=0
maxlog=1
log_warning=1
log_notice=1
log_info=0
log_debug=0

[TheMonitor]
type=monitor
module=mariadbmon
servers=server1,server2,server3
user=maxscale
password=P@ssw0rd
auto_rejoin=true
auto_failover=true

[RW-Split-Router]
type=service
router=readwritesplit
servers=server1,server2,server3
user=maxscale
password=P@ssw0rd
max_slave_connections=100%
```

```
[RW-Split-Listener]
type=listener
service=RW-Split-Router
protocol=MariaDBClient
port=3306
```

```
[server1]
type=server
address=142.93.98.60
port=3306
protocol=MariaDBBackend
```

```
[server2]
type=server
address=142.93.103.153
port=3306
protocol=MariaDBBackend
```

```
[server3]
type=server
address=142.93.103.246
port=3306
protocol=MariaDBBackend
```

```
## Start
```

```
systemctl start maxscale
```

```
## What does the log say ?
```

```
## /var/log/maxscale/maxscale.log
```

## maxctrl

```
maxctrl list servers
maxctrl show server server1
maxctrl list services
maxctrl show service ReadWrite-Split-Router
```

## Reference: MaxScale-Proxy mit Monitoring

[MaxScale MariaDB-Monitor](#)

## Walkthrough:Automatic Failover Master Slave

<https://mariadb.com/kb/en/mariadb-maxscale-25-automatic-failover-with-mariadb-monitor/>

## Tools & Tricks

### Percona-toolkit-Installation - Ubuntu

## Walkthrough

```
## Howto
## https://www.percona.com/doc/percona-toolkit/LATEST/installation.html

## Step 1: repo installieren mit deb -paket
wget https://repo.percona.com/apt/percona-release_latest.focal_all.deb;
apt update;
apt install -y curl;
dpkg -i percona-release_latest.focal_all.deb;
apt update;
apt install -y percona-toolkit;
```

## pt-query-digest under Windows

### Attention about download

url is wrong in Reference document, us:  
<https://www.percona.com/get/pt-query-digest>

### Reference

- <http://www.jonathanlevin.co.uk/2012/01/query-digest-on-windows.html>

## pt-query-digest - analyze slow logs

### Requires

- Install percona-toolkit

### Usage

```
## first enable slow_query_log
set global slow_query_log = on
set global long_query_time = 0.2
## to avoid, that i have to reconnect with new session
set session long_query_time = 0.2

## produce slow query - for testing
select * from contributions where vendor_last_name like 'W%';
mysql > quit

##
cd /var/lib/mysql
## look for awhile with -slow.log - suffix
pt-query-digest mysql-slow.log > /usr/src/report-slow.txt
less report-slow.txt
```

## pt-online-schema-change howto

### Requirements

- Install percona-toolkit

## What does it do ?

```
## Altering table without blocking them
## Do a dry-run beforehand
pt-online-schema-change --alter "ADD INDEX idx_city (city)" --dry-run
D=contributions,t=donors
##
pt-online-schema-change --alter "ADD INDEX idx_city (city)" --execute
D=contributions,t=donors
```

## Problems -> high cpu load

```
## fine - tune params
## e.g. --max-load
## refer to docs
https://www.percona.com/doc/percona-toolkit/3.0/pt-online-schema-
change.html#:~:text=pt%2Donline%2Dschema%2Dchange%20works%20by%20creating%20an%20empty,.
```

## Ubuntu-with-Vagrant

### Walkthrough

```
## Step 1: Download git for windows
https://git-scm.com/downloads
## Step 2: Install Virtualbox
https://download.virtualbox.org/virtualbox/6.1.18/VirtualBox-6.1.18-142142-Win.exe
## Step 3: Auf dem Desktop, rechte Maustaste -> git bash here
## in the bash
mkdir myvirtualmachine
vagrant init ubuntu/focal64
vagrant up
## and the you are in the machine (shell)
vagrant ssh
## within machine switch from vagrant user to root without password
sudo su -
## there you go - install whatever
```

## Include provisioning in Vagrantfile

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y mysql-server-5.7 wget
  cd /usr/src
  touch foo
  wget https://downloads.mysql.com/docs/sakila-db.tar.gz
  tar xzvf sakila-db.tar.gz
  cd sakila-db
  mysql < sakila-schema.sql
  mysql < sakila-data.sql
```

```
SHELL
end
```

## Destroy machine

```
vagrant destroy -f
```

## mysql-client

### \G Spezialausgabe

```
## Spalten werden als Zeilen angezeigt
## nur im mysql-client
mysql

mysql> show variables like 'bind%' \G
```

## Pager

```
## pager innerhalb von mysql verwenden
mysql> pager less
mysql> -- Jetzt wird der Linux Pager less verwendet
mysql> -- so schalte ich ihn wieder ab
mysql> pager
```

## Schweizer Such-Taschenmesser grep -r

```
grep -r "PermitRootLogin" /etc
```

## Set timezone in Centos 7/8

```
## as root
timedatectl list-timezones | grep 'Europe/Berlin'
timedatectl set-timezone 'Europe/Berlin'
timedatectl
```

## Ist die Netzwerkkarte eingerichtet - nmtui

```
## Grafische Oberfläche auf der Kommandozeile
nmtui
```

## User anlegen und passwort vergeben (Centos/Redhat)

```
## als root ausführen
useradd training
passwd training
```

## Scripts for deploying galera-cluster to Ubuntu 20.04

- <https://github.com/jmetzger/ansible-galera-cluster-maxscale>

## Extras

### User Variables

```
## only valid within one session
set @host='localhost';

## You can use it in select
select @host;

## You can use it in the where clause
select mysql.user where host=@host;

## not possible to use it within create user
## DOES NOT WORK !
set @mypass='password';
create user someuser@somehost identified by @mypass;
```

### Installation sakila-db

```
cd /usr/src
wget https://downloads.mysql.com/docs/sakila-db.tar.gz
tar xzvf sakila-db.tar.gz

cd sakila-db
mysql < sakila-schema.sql
mysql < sakila-data.sql
```

## Diagnosis and measurement of performance

### Best practices to narrow down performance problems

#### Pre-Requisites

- System is slow

#### Analyze - Checklist - Step 1

```
## Are there slow queries ?
## look for time
show full processlist

### or time - in seconds
select * from information_schema.processlist where time > 10;
```

#### Re-Execute SELECT or where from UPDATE / DELETE

```
## Is it still slow ?
## Eventually kill
mysql>show processlist
mysql>--kill <Thread-id>
```

```
mysql>-- example
mysql>kill 44
```

## Explain what is going on

```
Explain Select....
```

## Performance and optimization of SQL statements

### Do not use '\*' whenever possible

#### Why ?

- You are adding .. to the server:
  - I/O
  - memory
  - CPU
- You are preventing covering indexes

### Walkthrough. (Look at the time)

#### Using '\*'

```
## using '*'
pager grep "rows in set";
select * from donors where last_name like 'Willia%'; select * from donors where
last_name like 'Willia%';
-- time between 0.02 and 0.04 secs
-- 2424 rows in set (0.02 sec)
-- reset pager
pager
```

```
## corresponding Explain (QEP)
```

```
explain select * from donors where last_name like 'Willia%';
```

```
+---+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key |
| key_len | ref | rows | filtered | Extra |
+---+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | donors | NULL | range | donors_donor_info |
donors_donor_info | 213 | NULL | 4748 | 100.00 | Using index condition |
+---+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

#### using specific fields

```
pager grep 'rows in set'; select last_name,first_name from donors where last_name like
'Willia%'; pager;
PAGER set to 'grep 'rows in set''
2424 rows in set (0.01 sec)
```



```

explain select last_name,first_name from donors where last_name like 'Willia%';
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key |
| key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | donors | NULL | range | donors_donor_info | donors_donor_info | 213 | NULL | 4748 | 100.00 | Using where; Using index |
+----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

```

- Uses cover index (indicator in Extra: using index)

#### Ref:

- <https://www.oreilly.com/library/view/high-performance-mysql/9780596101718/ch04.html>

### Optimizer-hints (and why you should not use them)

#### Tell the optimizer what to do and what not to do

- <https://dev.mysql.com/doc/refman/5.7/en/optimizer-hints.html#optimizer-hints-syntax>

## Replication

### Replikation Read/Write

- <https://proxysql.com/blog/configure-read-write-split/>

## Performance

### Best Practices

#### Indexes

##### 2 Indexes vs. Combined Index

- In most cases a combined index is better than 2 indexes.

#### Joins

##### Field-Type

- Do not use varchar() or char() aka string types of join field
- better: integer (unsigned) && same size
  - e.g. actor\_id id int unsigned

#### Views

##### General

- Only use views with merge
- NO temptable please, these CANNOT be indexed.

#### Where

##### No functions in where please

- Why ? Index cannot be used.
- example:
  - select first\_name from actor where upper(first\_name) like 'A%'

#### Alternative solution

- use a virtual field and index virtual field (possible from mysql > 5.7)
- Massive improvements in mysql 8

#### Example sys-schema and Reference

#### Examples

```
mysql> select * from sys.host_summary\G
***** 1. row *****
      host: localhost
    statements: 1347
 statement_latency: 7.55 m
statement_avg_latency: 336.50 ms
      table_scans: 15
        file_ios: 612857
    file_io_latency: 1.66 m
current_connections: 1
  total_connections: 7
      unique_users: 1
    current_memory: 0 bytes
total_memory_allocated: 0 bytes
1 row in set (0.01 sec)
```

#### Ref:

- <https://github.com/mysql/mysql-sys/blob/master/README.md>

#### Change schema online (pt-online-schema-change)

- <https://www.percona.com/doc/percona-toolkit/3.0/pt-online-schema-change.html>

#### Optimizer-Hints

#### Tell the optimizer what to do and what not to do

- <https://dev.mysql.com/doc/refman/5.7/en/optimizer-hints.html#optimizer-hints-syntax>

### Documentation / Literature

#### MySQL - Performance Blog

- <https://www.percona.com/blog/>

#### Source-Code MariaDB

- <https://github.com/MariaDB/server>

#### Effective MySQL

- <https://www.amazon.com/Effective-MySQL-Optimizing-Statements-Oracle/dp/0071782796>

#### Last Training

- <https://github.com/jmetzger/training-mysql-developers-basics>

## MariaDB Galera Cluster

- <http://schulung.t3isp.de/documents/pdfs/mariadb/mariadb-galera-cluster.pdf>

## MySQL Galera Cluster

- <https://galeracluster.com/downloads/>

## Releases List - Long Time / Stable

- <https://mariadb.com/kb/en/mariadb-server-release-dates/>

## Questions and Answers

### Questions and Answers

#### 1. Do you recommend Aurora

In my current humble opinion Aurora is a double edged sword. Aurora looks promising for scalability, but a lot of stuff is modified mysql-stuff and in my opinion has a lot of restrictions.

You should be aware, that moving to Aurora might be a task and reverting back even more.

- Refer to: <https://ahmedahamid.com/aurora-mysql/>

I would like to point you to a performance measurement report here:

- <https://galeracluster.com/2019/09/everdata-reports-galera-cluster-outshines-amazon-aurora-and-rds/>

#### 2. Get rid of unattended - upgrades problem (dirty hack)

```
ps aux | grep unatt  
kill <process-id-von-unattended-upgrades>
```

#### 3. Archive Data

<https://www.percona.com/doc/percona-toolkit/LATEST/pt-archiver.html>

#### 4. Does innodb do defragmentation by itself ?

```
## Some background while doing research.  
## Nil performance benefits of defragmentation in index.  
https://stackoverflow.com/questions/48569979/mariadb-table-defragmentation-using-optimize
```

#### 5. Defragmentation

```
## Optimize table  
ALTER TABLE contributions engine = InnoDB
```

```
## mariadb has a patch for defragmentation
https://mariadb.org/defragmenting-unused-space-on-innodb-tablespace/

## alter table xyz engine=InnoDB - defragmentations
## but is also invasive.
## with ibdata1 innodb_file_per_table it lets the size grow
```

## 6. Is it possible to do select, update, deletes without using innodb\_buffer in specific

No, this is not possible

## 7. Unit test framework in MySQL

No, there is no testing framework with MySQL

## 8. MariaDB - Advantages

- flashback
- Verschlüsselung von Tabellen // mariabackup
- Einige Storage Engine (Aria -> MyISAM - crash-recovery)
- JSON anders implementiert
- galera
- feature: defragmentation

```
Mysql 8 does not:
decode
set profiling (still available but deprecated )
```

## 9. Select without locking

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED ;
BEGIN ;
SELECT * FROM TABLE_NAME ;
COMMIT ;
```

## Best filesystem for MariaDB

- <https://mariadb.com/de/resources/blog/what-is-the-best-linux-filesystem-for-mariadb/>

## MySQL Do-Not's

### mysql-do-nots

#### 1. No function in where (column\_name)

```
## Never use a function for the column name in where
## e.g.
select * from donors where upper(last_name) like 'Willia%'
```

### Why ?

- Not index can be used

```
## Not filtering possible by indx -> possible_keys -> NULL
explain select last_name from donors where upper(last_name) like 'WILLI%';
+----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key |
key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | donors | NULL | index | NULL | donors_donor_info |
687 | NULL | 701948 | 100.00 | Using where; Using index |
+----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```