

# MariaDB Basics

## Agenda

### 1. Architecture of MariaDB

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

### 2. Installation / Configuration

- [Installation \(Ubuntu\)](#)
- [start/stop/status and logs](#)
- [Is mariadb listening to the outside world \(and how to fix\)?](#)

### 3. Administration

- [Debug configuration error](#)
- [Server System Variables](#)
- [Show structure of database](#)
- [Binary Logging](#)
- [Kill Session/User](#)

### 4. Training Data

- [Setup sakila test database](#)

### 5. Security and User Rights

- [Create User/Grant/Revoke - Management of users](#)
- [Getting rid of specific user after user permissions changes](#)
- [Secure with SSL server/client](#)
- [Secure with ssl for ubuntu/debian](#)
- [Table encryption](#)

### 6. InnoDB - Storage Engine

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

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

- [General](#)
- [Backup with mysqldump - best practices](#)
- [mariabackup](#)
- [mariadbbackup incremental](#)

### 8. Documentation

- [Mariadb Server System Variables](#)
- [MySQL - Performance - PDF](#)
- [MySQL Performance Blog](#)

## Add-Ons (Further read) / backlog

### 1. Architecture of MariaDB

- [Query Cache Usage and Performance](#)

### 2. Administration

- [Handling general\\_log](#)

### 3. Training Data

- [Setup training data "contributions"](#)

### 4. Optimal use of indexes

- Index-Types
  - [Describe and indexes](#)
  - [Find out indexes](#)
- [Index and Functions \(Cool new feature in MySQL 5.7\)](#)
- [Index and Likes](#)
- [profiling-get-time-for-execution-of-query](#)

- [Find out cardinality without index](#)

#### 5. Monitoring

- [What to monitor?](#)

#### 6. Replication

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

#### 7. Tools

- [Percona-toolkit-Installation](#)
- [pt-query-digest - analyze slow logs](#)
- [pt-online-schema-change howto](#)

#### 8. Diagnosis and measurement of performance

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

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

- [Do not use "\\*" whenever possible](#)
- [Be aware of subselects - Example 1](#)
- [Optimizer-hints \(and why you should not use them\)](#)

#### 10. Replication

- [Replikation Read/Write](#)

#### 11. Performance

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

#### 12. Upgrading / Patching

- [Upgrade vom 10.3 \(Distri Ubuntu 20.04\) -> 10.4 \(MariaDB-Foundation\)](#)

#### 13. Security and User Rights

- [Create User/Grant/Revoke - Management of users](#)
- [Getting rid of specific user after user permissions changes](#)
- [Disable unix\\_socket authentication for user](#)
- [Debug and Setup External Connection](#)
- [Get Rights of user](#)
- [Auth with unix\\_socket](#)
- [User- and Permission-concepts \(best-practice\)](#)
- [Setup external access](#)

#### 14. Backup and Restore (Point-In-Time aka PIT)

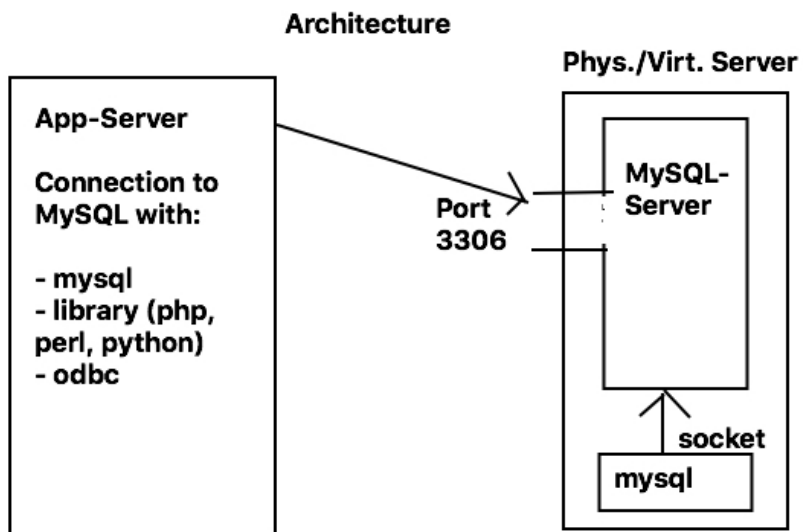
- [General](#)
- [Backup and Create new database based on backup](#)
- [PIT - Point-in-time-Recovery Exercise](#)
- [Backup / Recover to Network Destination](#)
- [Flashback](#)
- [Use xtrabackup for MariaDB 5.5](#)

#### 15. Documentation / Literature

- [Effective MySQL](#)
- [MariaDB Galera Cluster](#)
- [MySQL Galera Cluster](#)
- [Alternative password authentication \(salting\)](#)
- [User statistics](#)

## Architecture of MariaDB

### Architecture Server



### Storage Engines

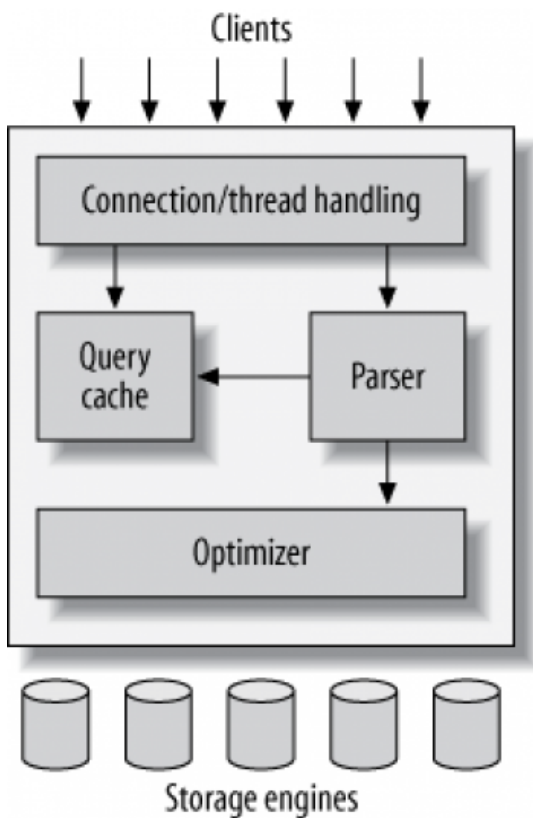
#### Why ?

Let's you choose:  
How your data is stored

#### What ?

- Performance, features and other characteristics you want

#### Looks like



#### 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 ?

- MyISAM/Aria
- InnoDB
- Memory
- CSV
- Blackhole (/dev/null)
- Archive
- Federated/FederatedX

## Installation / Configuration

### Installation (Ubuntu)

#### Install version from distribution (older version)

```
apt update
apt install mariadb-server
```

## Install Newest version from mariadb

```
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'
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el] https://mirror.dogado.de/mariadb/repo/10.5/ubuntu focal
main'

apt update
apt install mariadb-server
```

## Secure installation

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

## start/stop/status and logs

```
## How to find out if it is running
systemctl status mariadb

## To stop it
systemctl stop mariadb

## To start it
systemctl start mariadb

## to restart it
systemctl restart mariadb

## How it the configuration of the service
systemctl cat mariadb

## Logs
## last 10 lines
systemctl status mariadb
journalctl -u mariadb
```

## Is mariadb listening to the outside world (and how to fix)?

### not the case

```
lsof -i
## or
netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
## <- in this case not - localhost
```

### Yes !

```
## ubuntu 20.04
## change to listen on all interfaces
## vi /etc/mariadb-conf.d/50-server.cnf
## this is only for the mysqld standalone daemon
[mysqld]
bind-address = 0.0.0.0
```

```
## restart
systemctl restart mariadb

lsof -i
## connect to the server by external interface (e.g. eth0 )
mysql -h 10.0.3.3
```

## Administration

### Debug configuration error

#### 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

## Didn't find something -> step 5
## General Log
## Debian/Ubuntu
/var/log/syslog
## REdhat/Centos
/var/log/messages
```

### Find errors in logs quickly

```
cd /var/log/mysql
## -i = case insensitive // no matter if capital or lower letters
cat error.log | grep -i error
```

### Server System Variables

```
MariaDB [(none)]> show global variables like '%long%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| deadlock_search_depth_long | 15 |
| deadlock_timeout_long | 50000000 |
| long_query_time | 10.000000 |
| max_long_data_size | 16777216 |
| performance_schema_events_stages_history_long_size | -1 |
| performance_schema_events_statements_history_long_size | -1 |
| performance_schema_events_waits_history_long_size | -1 |
+-----+-----+
7 rows in set (0.001 sec)

MariaDB [(none)]> select @@long_query_Time
-> ;
+-----+
| @@long_query_Time |
```

```

+-----+
|          10.000000 |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> select @@long_query_time
-> ;
+-----+
| @@long_query_time |
+-----+
|          10.000000 |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> select @@GLOBAL.long_query_time
-> ;
+-----+
| @@GLOBAL.long_query_time |
+-----+
|          10.000000 |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> select @@global.long_query_time
-> ;
+-----+
| @@global.long_query_time |
+-----+
|          10.000000 |
+-----+
1 row in set (0.000 sec)

```

```

## Within server
SET GLOBAL general_log = 1;

```

## Show structure of database

```

mysql>use mysql;
mysql>describe columns_priv;
mysql>show create table columns_priv;

```

## Binary Logging

### General

- It is disabled by default

## Why and when to use it ?

- Needed Galera Cluster (3 - Node - Cluster)
- Replication
- PIT (Point-In-Time) - Recovery (e.g. recover to start from 4 a.m. with full backup + binary log)

## How to enable it ?

```

## Ubuntu
## vi /etc/mysql/mariadb.conf.d/50-server.cnf
[mysqld]
log-bin

## Restart server
systemctl restart mariadb

```

## How to view the binary-log

```
cd /var/lib/mysql

mysqlbinlog -vv mysqld-bin.000001
## in the special configuration from /etc/mysql/... gets in the way
mysqlbinlog --no-defaults -vv mysqld-bin.000001
```

## Kill Session/User

```
MariaDB [(none)]> show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+
| Id | User      | Host      | db  | Command | Time | State      | Info              | Progress |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 37 | root      | localhost | NULL | Query   | 0    | Init       | show processlist | 0.000    |
| 38 | root      | localhost | NULL | Query   | 10   | User sleep | select sleep(1000) | 0.000    |

# kill thread 38. Connection will be interrupted. User session will be cancelled
kill 38
```

## Training Data

### Setup sakila test database

```
cd /usr/src
wget https://downloads.mysql.com/docs/sakila-db.tar.gz
tar xvf sakila-db.tar.gz
cd sakila-db/
ls -la
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 'yourpassword';
```

#### 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
```



### Useful command to find out users:

```
select user,host from mysql.user;
```

### Refs:

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

### Getting rid of specific user after user permissions changes

#### Why ?

- You might have changed the grants, but they only reflect after a reconnect

#### Howto

```
## step 1: get thread_id id of user
MariaDB [information_schema]> select id,user,host,command from processli
st where user='training';
+----+-----+-----+-----+
| id | user   | host               | command |
+----+-----+-----+-----+
| 75 | training | jochen-wt6y:42026 | Sleep   |
+----+-----+-----+-----+
1 row in set (0.001 sec)

## step 2: kill thread_id = connection_id = id
kill 75
```

### 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

```

#### Setup user to use client-certificate

```

## Client certificate needs to be there
ALTER USER 'alice'@'%'
    REQUIRE X509;

## Client certificate needs to be a specific one

```

```
ALTER USER 'alice'@'%'
  REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland';

## Reference:
https://mariadb.com/kb/en/securing-connections-for-client-and-server/
```

### 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/>

### Secure with ssl for ubuntu/debian

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

##### Create CA and Server-Key

```
## On Server - create ca and certificates
mkdir -p /etc/mysql/ssl; cd /etc/mysql/ssl

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

## create ca-certificate
## Common Name: MariaDB CA
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 ----
openssl req -newkey rsa:2048 -days 365000 -nodes -keyout server-key.pem -out server-req.pem

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

## Now sign the key
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/mysql/mariadb.conf.d/z_ssl.cnf
[mysqld]
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/server-cert.pem
ssl-key=/etc/mysql/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/mysql/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;
```

## 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
## show content - is there readable content ?
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;
```

```
## Create a user that is not allowed to do so .... no set global
create user noroot@'localhost' identified by 'password';
grant all on *.* to noroot@'localhost';
revoke super on *.* from noroot@'localhost';
```

#### working with mysqlbinlog and encryption

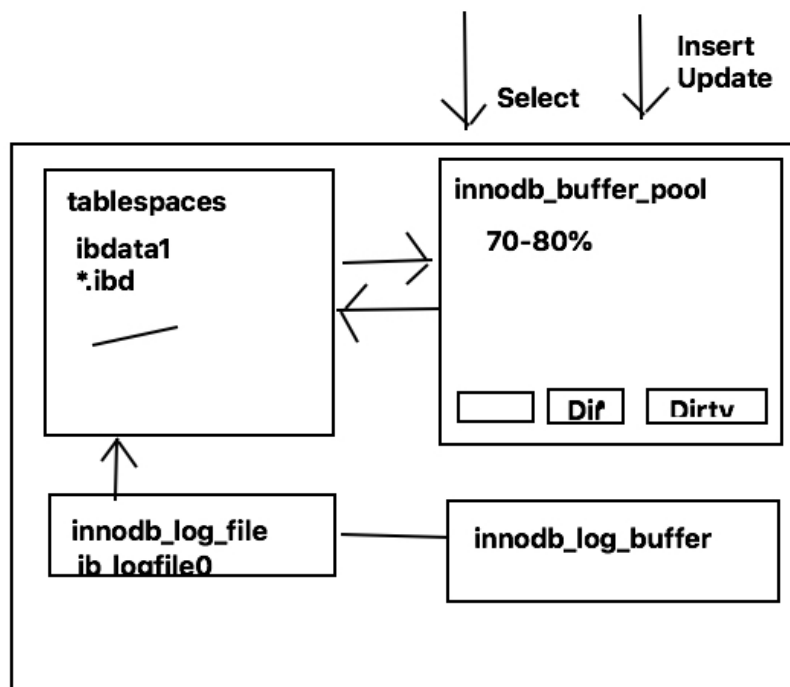
```
mysqlbinlog -vv --read-from-remote-server --socket /run/mysqld/mysqld.sock mysqld-bin.000003 | less
```

#### Ref:

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

## InnoDB - Storage Engine

### InnoDB - Storage Engine - Structure



## Important InnoDB - configuration - options to optimized performance

### InnoDB buffer pool

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

```
pager grep -i 'free buffers'
show engine innodb status \G
Free buffers          7905
1 row in set (0.00 sec)
```

```
show status like '%free%';
```

### 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%';
```

### 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
```

## skip-name-resolv.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
```

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

### General

#### Define your goal

- Full backup of database-server (specific to PIT - point-in-time)
- Simply backup some specific databases (with data) - ( e.g. 1 database out of 20)
- Backup Structure and Data separately in multiple files - (For further work - e.g. for developers)
- Extract data from a specific table (because of problems that came up)

### Backup with mysqldump - best practices

#### best practice minimal options

```
mysqldump --all-databases --events --routines
```

#### 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 (not using replication)

- All logs before flushing will be deleted

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

#### Alternative - flushing logs

- <https://mariadb.com/kb/en/purge-binary-logs/>



## 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
```

## mariabackup

### Installation (Ubuntu/Debian)

```
apt install mariadb-backup
```

### Walkthrough (Backup)

```
## 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
```

### Walkthrough (Recover)

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/backups/20200120 --copy-back
chmod -R mysql:mysql /var/lib/mysql
systemctl start mariadb
```

### Ref.

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

## mariadbbackup incremental

### Prerequisites: Setup user to be used in /root/.my.cnf

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

## Backup-Phase

### Day 1: First full backup (directory always needs to be empty)

```
## create some data
mysql -e "create schema if not exists backuptest"
mysql -e "create table if not exists data (id int, content varchar(50), primary key(id))" backuptest
mysql -e "insert into data (id, content) values (1, 'day1 - dataset 1'),(2, 'day 1 - dataset 2')" backuptest
mysql -e "select * from data" backuptest

## create a folder for our backup
mkdir -p /var/mariadb

## Day 1: let us do the full backup
mariabackup --backup --target-dir=/var/mariadb/backup/
```

### Day 2: Let us add some data and then do the incremental backup

```
mysql -e "insert into data (id, content) values (3, 'day2 - dataset 1'),(4, 'day 2 - dataset 2')" backuptest
mysql -e "select * from data" backuptest
## now do the backup - folder inc1 needs to be empty !!!
mariabackup --backup \
  --target-dir=/var/mariadb/inc1/ \
  --incremental-basedir=/var/mariadb/backup/
```

### Day 3: Let us even add more more data and the do the incremental backup

```
mysql -e "insert into data (id, content) values (5, 'day3 - dataset 1'),(6, 'day 3 - dataset 2')" backuptest
mysql -e "select * from data" backuptest

## now we do the backup based on the last incremental backup (so basedir is inc1)

mariabackup --backup \
  --target-dir=/var/mariadb/inc2/ \
  --incremental-basedir=/var/mariadb/inc1/
```

## Recovery Phase

### Prepare

```
## Step 1: Apply the changes from recovery/redo log of full backup
mariabackup --prepare --target-dir=/var/mariadb/backup

## Step 2: Add the changes from inc1
mariabackup --prepare --target-dir=/var/mariadb/backup --incremental-dir=/var/mariadb/inc1

## Step 3: Add the changes from inc2
mariabackup --prepare --target-dir=/var/mariadb/backup --incremental-dir=/var/mariadb/inc2
```

### Copy-Back

```
systemctl stop mariadb
cd /var/lib/
mv mysql mysql.mybkup
mariabackup --copy-back --target-dir=/var/mariadb/backup
chown -R mysql:mysql mysql
systemctl start mariadb

## Check if we have all data again
mysql -e "select * from data" backuptest
```

### Ref:

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

## Documentation

### Mariadb Server System Variables

- [https://mariadb.com/kb/en/server-system-variables/#long\\_query\\_time](https://mariadb.com/kb/en/server-system-variables/#long_query_time)

## MySQL - Performance - PDF

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

## MySQL Performance Blog

- [https://www.percona.com/blog/choosing-innodb\\_buffer\\_pool\\_size/](https://www.percona.com/blog/choosing-innodb_buffer_pool_size/)

## 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.

## Administration

### Handling general\_log

#### Activate during runtime

```

## Hint hostname: myserver
mysql>set global general_log = 1

ls -la /var/lib/mysql/myserver.log

```

## Implications

- By default
- Will massively increase in size, because all queries are documented

## Truncate while running

```

## will be empty that
cd /var/lib/mysql
> myserver.log

## and keeps on writing in there

## Attention
## Delete logfile does not work, needs restart
## or
## set global general_log = 0; set global general_log = 1 # after deletion

```

## 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
```

## Optimal use of indexes

### Index and Functions (Cool new feature in MySQL 5.7)

No index 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 virtual columns (possible since mysql 5.7)

```
## 1. Create Virtual Column with upper
alter table sakila add idx_last_name_upper varchar(45) GENERATED ALWAYS AS upper(last_name);
## 2. Create an index on that column
create index idx_last_name_upper on actor (last_name_upper);
```

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)

### Preview MySQL 8

- MySQL 8 support functional indexes

### 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)
```

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.	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
```

#### Ref

- <https://blog.serverdensity.com/how-to-monitor-mysql/>

#### Monitoring with pmm (Percona Management Monitoring)

<https://pmmdemo.percona.com>

[Documentation](#)

## Replication

### Slave einrichten -gtid

#### 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/mysqldumpdir/20210121 kurs@10.10.9.144:/home/kurs/
```

#### 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
```

#### 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

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

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/
### root@slave
mv /home/kurs/z_uniruhr.cnf /etc/mysql/mariadb.conf.d/
chown root:root /etc/mysql/mariadb.conf.d
systemctl restart mariadb
```

#### Step 5: Restore Data on slave

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup4
mariabackup --target-dir=/backups/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/>

### Slave einrichten - master\_pos

#### Step 1: mysqldump on master

```
mkdir -p /backups/mysqldumpdir
## in version 5.5. there is not --git so use it without --gtid
mysqldump --all-databases --single-transaction --master-data=2 --routines --events --compress >
/backups/mysqldumpdir/master-databases.sql;
```

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

```
## root@master:
rsync -e ssh -avP /backups/mysqldumpdir/master-databases.sql kurs@10.10.9.144:/home/kurs/
```

#### Step 3 (Optional): Be sure that slave is really fresh (no data yet)

```
## if old not wanted data is present, e.g. other databases, start with fresh-installation by so:
## as root
cd /var/lib
mv mysql mysql.bkup
mariadb-install-db --user=mysql
```

#### Step 4: 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
```

#### Step 4a (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 5a: Set server-id on master -> 1

```
[mysqld]
server-id=1

systemctl restart mariadb
###
```

#### Step 5b: Set server-id on slave -> 2 + same config as server 1

```
[mysqld]
server-id          = 2
## activate master bin log, if this slave might be a master later
log_bin           = /var/log/mysql/mysql-bin.log

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/
### root@slave
mv /home/kurs/z_uniruhr.cnf /etc/mysql/mariadb.conf.d/
chown root:root /etc/mysql/mariadb.conf.d
systemctl restart mariadb
```

## Step 6: Restore Data on slave

```
## root@slave
cd /home/kurs
mysql < master-databases.sql
```

## Step 7: master.txt for change command

```
## root@slave
## /root/master.txt
## get information from master-databases.sql dump
CHANGE MASTER TO
  MASTER_HOST="10.10.9.110",
  MASTER_PORT=3310,
  MASTER_USER="repl",
  MASTER_PASSWORD="password",
  MASTER_LOG_FILE='mysqld-bin.000001',
  MASTER_LOG_POS=568;
## Version 1
mysql < master.txt
## or: copy paste into mysql>

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

## Step 8: not working on 5.5.

```
Switch to using gtid later on:

show slave status; # look for using_gtid
stop slave;
CHANGE MASTER TO MASTER_USE_GTID = slave_pos;
show slave status; # look for using_gtid
start slave;
```

## Walkthrough

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

## 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

### Percona-toolkit-Installation

#### 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 - 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%2Dschem%2Dchange%20works%20by%20creating%20an%20empty,it%20with%20the%20new%20one.
```

## 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>

#### Be aware of subselects - Example 1

#### 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>

## Upgrading / Patching

### Upgrade vom 10.3 (Distri Ubuntu 20.04) -> 10.4 (MariaDB-Foundation)

### 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. apt remove mariadb-*
## 3. Doublecheck if components left: apt list --installed | grep mariadb
## 4. Setup repo for mariadb
## 5. apt update
## 6. apt install mariadb-server

## 7. systemctl enable --now mariadb # enable for next reboot and start immediately
## necessary for redhat
```

```
## 8. Doublecheck if mysql_upgrade was done
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/>

## Security and User Rights

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

#### Create user

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

#### 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
```

#### Useful command to find out users:

```
select user,host from mysql.user;
```

#### Refs:

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

### Getting rid of specific user after user permissions changes

#### Why ?

- You might have changed the grants, but they only reflect after a reconnect

#### Howto

```
## step 1: get thread_id id of user
MariaDB [information_schema]> select id,user,host,command from processli
st where user='training';
+----+-----+-----+-----+
| id | user      | host      | command |
+----+-----+-----+-----+
```

```
+-----+-----+-----+
| 75 | training | jochen-wt6y:42026 | Sleep |
+-----+-----+-----+
1 row in set (0.001 sec)

## step 2: kill thread_id = connection_id = id
kill 75
```

## Disable unix\_socket authentication for user

## Debug and Setup External Connection

## Get Rights of user

## Auth with unix\_socket

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

## Setup external access

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

## General

### Define your goal

- Full backup of database-server (specific to PIT - point-in-time)
- Simply backup some specific databases (with data) - ( e.g. 1 database out of 20)
- Backup Structure and Data separately in multiple files - (For further work - e.g. for developers)
- Extract data from a specific table (because of problems that came up)

### Backup and Create new database based on backup

```
mysqldump sakila > sakila.sql
mysql -e 'create schema sakilanew'
## or
echo "create schema sakilanew" | mysql

mysql sakilanew < sakila.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 --no-defaults -vv mysqldbin.000005

## Step 3: 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 recovery.sql - file (before apply full backup)
mysqlbinlog --no-defaults -vv --stop-position=857 mysqld-bin.000005 > /usr/src/recover.sql

## Step 1: Apply full backup
cd /usr/src/
mysql < all-databases.sql
mysql> should be 200 or 202
mysql> use sakila; select * from actor;
mysql < recover.sql
mysql> -- now it should have all actors before deletion
mysql> use sakila; select * from actor;
```

## Backup / Recover to Network Destination

### Assumptions

```
Server 1: 192.168.1.1
Server 2: 192.168.1.2

Create new db -> sakilaremote on server 1
Backup data from sakila on server2 and send to server 1
```

### Preparation (on server 1)

```
## is server listening to the outside world
lsof -i | grep mysql

## create user on server
mysql>create user ext@'%' identified by 'mysecretpass'
mysql>grant all on *.* to ext@'%'
```

### Testing (on server 1)

```
mysql -uext -p -h 192.168.1.1
mysql>create schema sakilaremote
```

### Executing (on server 2)

```
mysqldump sakila | mysql -uext -p -h 192.168.1.1 sakilaremote
```

### Validating (on server 2)

```
mysql -uext -p -h 192.168.1.1
mysql> use sakilaremote;
mysql> show tables;
```

### Flashback

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

### Referenz:

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

### 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>

## Documentation / Literature

### Effective MySQL

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

### MariaDB Galera Cluster

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

#### **MySQL Galera Cluster**

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

#### **Alternative password authentication (salting)**

#### **User statistics**

- <https://mariadb.com/kb/en/user-statistics/>