MySQL / MariaDB / Percona XtraDB (+ Galera Cluster)

    - Fehler bei ALTER TABLE Errno 150: Foreign key constraint is incorrectly formed
    - InnoDB - defekte FRM Files wiederherstellen - Incorrect information in file
    - Probleme mit Umlauten zwischen Datenbank und Webseite (UTF-8/latin1/ISO8859-1)

- Backup
    - Bestimmte Datenbanken vom Dump ausschließen
    - Dump einer einzelnen Datenbank
    - Extraktion einer einzelnen Datenbank oder Tabelle aus einem Fulldump
    - kompletten Dump eines MySQL-Servers erstellen
    - Mit netcat einen mysqldump übers Netzwerk schicken
    - komplettes MySQL-Dumpfile einspielen

- Replikation
    - MySQL Master/Slave-Replikation einrichten (online) (veraltet)
    - MySQL Master/Slave-Replikation einrichten (offline)
    - Master-Master-Replikation
    - Master-Slave-Replikation einrichten mit innobackupex

- kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.5
- kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.6)
- kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.7)
- MySQL mit anderem Datenverzeichnis auf speziellem Port starten

Percona XtraDB Cluster

Deadlock Monitoring mit pt-deadlock-logger

Unit-File anlegen:

```
[Unit]
Description=Start the Percona Toolkit Deadlock Logger
After=network.target
After=syslog.target
After=mysql.service

[Service]
Type=simple
StartLimitInterval=5
StartLimitBurst=10
#Environment=PTDEBUG=1
ExecStart=/usr/bin/pt-deadlock-logger --user root --defaults-file /root/.my.cnf --create-dest-table --dest D=percona_schema,t=deadlocks h=dbcluster1-ha1.mydomain.com
Restart=always
RestartSec=120

[Install]
WantedBy=multi-user.target
```

Aktivieren und Starten:

```
systemctl daemon-reload
systemctl enable pt-deadlock-logger.service
systemctl start pt-deadlock-logger.service
```

Alle 30 Sekunden wird nun geprüft, ob im SHOW INNODB STATUS ein Deadlock aufgelaufen ist. Falls ja, wird dieser in die angegebene Tabelle geloggt.

Vorbereitung zur Migration auf Percona XtraDB Cluster

Um eine vorhandene MySQL Datenbank zu Percona XtraDB Cluster zu migrieren müssen einige Voraussetzungen erfüllt sein (siehe hierzu auch die Limitations-Seite bei Percona.

- Alle Tabellen müssen InnoDB Engine sein
- Jede Tabelle sollte einen PRIMARY KEY haben

Alle Tabellen finden, die nicht die InnoDB Engine verwenden

```
USE information_schema;
SELECT CONCAT(TABLE_SCHEMA, '.', TABLE_NAME) AS TABLES_NOT_INNODB
FROM TABLES
WHERE ENGINE != 'InnoDB' AND
      TABLE_SCHEMA NOT IN('mysql','information_schema','performance_schema');
```

Tabellen ohne PRIMARY KEY finden

Dies verhindert folgenden Fehler beim Import eines Dumps:

```
[ERROR] WSREP: Percona-XtraDB-Cluster prohibits use of DML
command on a table (db.taxonomy_index) without an explicit primary key
with pxc_strict_mode = ENFORCING or MASTER

ERROR 1105 (HY000) at line 20655: Percona-XtraDB-Cluster prohibits use
of DML command on a table (db.taxonomy_index) without an explicit
primary key with pxc_strict_mode = ENFORCING or MASTER
```

```
USE information_schema;
SELECT CONCAT(TABLES.TABLE_SCHEMA, '.', TABLES.TABLE_NAME) AS TABLES_WITHOUT_PRIMARY_KEY
FROM TABLES
LEFT JOIN KEY_COLUMN_USAGE AS c
ON (TABLES.TABLE_NAME = c.TABLE_NAME AND
    c.CONSTRAINT_SCHEMA = TABLES.TABLE_SCHEMA AND
```

```
    c.CONSTRAINT_NAME = 'PRIMARY')
WHERE TABLES.TABLE_SCHEMA <> 'information_schema' AND
      TABLES.TABLE_SCHEMA <> 'performance_schema' AND
      TABLES.TABLE_SCHEMA <> 'mysql' AND
      c.CONSTRAINT_NAME IS NULL;
```

Über ALTER TABLE läßt sich diesen Tabellen ein PRIMARY KEY zuweisen. Vorher sollte allerdings sichergestellt sein, dass die Applikation auch damit zurecht kommt.

PRIMARY KEY hinzufügen:

```
USE mydb;
ALTER TABLE address ADD id INT PRIMARY KEY AUTO_INCREMENT;
```

wsrep Status überwachen

Als Script auf einer Cluster-Node ablegen. Gibt dauerhaft den aktuellen Wert der wsrep-Statusvariablen aus:

```
#!/bin/sh
watch -d -n1 -x mysql -B -e "SHOW STATUS WHERE variable_name ='wsrep_local_state_comment' OR
variable_name ='wsrep_cluster_size' OR variable_name ='wsrep_incoming_addresses' OR
variable_name ='wsrep_cluster_status' OR variable_name ='wsrep_connected' OR variable_name
='wsrep_ready' OR variable_name ='wsrep_local_state_uuid' OR variable_name
='wsrep_cluster_state_uuid' OR variable_name ='wsrep_last_committed' OR variable_name
='wsrep_received' OR variable_name ='wsrep_received_bytes';"
```

Alle wsrep%-Variablen ausgeben:

```
#!/bin/sh
watch -d -n1 -x mysql -B -e "SHOW STATUS WHERE variable_name like 'wsrep%'";
```

Percona APT Repository einrichten

Installationspaket für das Repository holen und installieren

```
apt update
apt install gnupg2
wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.deb
dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb
apt update
```

Percona Software installieren

```
apt install percona-toolkit xtrabackup percona-server-server
```

InnoDB Tablespace Cleanup mit pt-online-schema-change

Mit der Zeit wird das Datenfile (.ibd) einer InnoDB Tabelle durch das Hinzufügen und Löschen von Daten immer größer. Obwohl die Tabelle eigentlich viel kleiner ist wird der benötigte Plattenplatz nicht immer wieder freigegeben.

Die Voraussetzung für die folgenden Tipps ist, dass `innodb_file_per_table=on` gesetzt und beim PXC die `wsrep_OSU_method=TOI` eingestellt ist.

Bei Tabellen ohne große Last kann das schnell und einfach mit einem `mysqloptimize -A` behoben werden (alle Datenbanken und Tabellen werden "optimiert"). Dabei wird die Tabelle neu erstellt.

Handelt es sich um eine sehr aktive Tabelle und soll der Impact dieser Aktion nicht so groß sein (und zum Beispiel einen Percona Cluster sperren), dann kann man das Tool pt-online-schema-change aus dem Percona Toolkit nutzen. Das Tool kann die Tabellenstruktur ändern ohne Lese-/Schreibzugriffe zu blockieren. Normalerweise wird es verwendet, um z.B. ein neues Feld oder einen Index hinzuzufügen.

Ein `ALTER TABLE ENGINE=InnoDB` bewirkt hier aber das gleiche wie ein `mysqloptimize`. Ich habe mir dafür ein Script geschrieben, das alle Tabellen einer Datenbank durcharbeitet. Einziges Argument ist der Datenbankname.

```
#!/bin/bash


DB=${1}


TABLES=$(mysql -N -B ${DB} -e "show tables;")


for table in ${TABLES}; do
  echo "---------------- Processing ${DB} -- ${table} ----------------"
  pt-online-schema-change --execute --alter "ENGINE=InnoDB" --progress=time,5
D=${DB},t=${table}
  echo
done
```

Das sieht dann so aus:

```
# ./optimize-table-pt-osc.sh my_db
---------------- Processing my_db -- AuthAssignment ----------------
No slaves found.  See --recursion-method if host dbserver-01-a has slaves.
Not checking slave lag because no slaves were found and --check-slave-lag was not specified.
Operation, tries, wait:
```

```
  analyze_table, 10, 1
  copy_rows, 10, 0.25
  create_triggers, 10, 1
  drop_triggers, 10, 1
  swap_tables, 10, 1
  update_foreign_keys, 10, 1
Altering `my_db`.`AuthAssignment`...
Creating new table...
Created new table my_db._AuthAssignment_new OK.
Altering new table...
Altered `my_db`.`_AuthAssignment_new` OK.
2023-08-31T10:49:15 Creating triggers...
2023-08-31T10:49:15 Created triggers OK.
2023-08-31T10:49:15 Copying approximately 157 rows...
2023-08-31T10:49:15 Copied rows OK.
2023-08-31T10:49:15 Analyzing new table...
2023-08-31T10:49:15 Swapping tables...
2023-08-31T10:49:15 Swapped original and new tables OK.
2023-08-31T10:49:15 Dropping old table...
2023-08-31T10:49:15 Dropped old table `my_db`.`_AuthAssignment_old` OK.
2023-08-31T10:49:15 Dropping triggers...
2023-08-31T10:49:15 Dropped triggers OK.
Successfully altered `my_db`.`AuthAssignment`.
```

Bei einem Kunden konnte ich so den benötigten Platz von 450 GB auf 90 GB verringern.

Allgemein

Allgemein

Datenbank und Benutzer anlegen (veraltet)

Einen neuen Benutzer (User superadmin kommend von IP-Adresse 123.123.123.123) mit vollen Rechten auf alles erzeugen inkl. dem Recht wiederum neue User zu erzeugen (GRANT OPTION):

```
GRANT ALL PRIVILEGES ON *.* TO 'superadmin'@'123.123.123.123' IDENTIFIED BY 'password' WITH
GRANT OPTION;
FLUSH PRIVILEGES;
```

Einen neuen Benutzer (User user1 auf Datenbank userdb1 kommend von IP-Adresse 123.123.123.123) mit vollen Rechten auf seine eigene Datenbank erzeugen:

```
GRANT ALL PRIVILEGES ON userdb1.* TO 'user1'@'123.123.123.123' IDENTIFIED BY 'password';
CREATE DATABASE userdb1;
FLUSH PRIVILEGES;
```

Allgemein

Benutzertabelle als GRANT-Statements dumpen

> Eine einfacherere Alternative wäre das Tool `pt-show-grants` aus dem Percona-Toolkit.

Bei der Einrichtung einer frischen Replikation kann es wichtig sein, dass vorhandene User nicht überschrieben werden. Mit dem folgenden Snippet lassen sich die Benutzer einer MySQL-DB als einfache GRANT-Statements inkl. gesonderten Rechten auf Datenbanken ausgeben und für den späteren Import passend bearbeiten.

```
mygrants()
{
  mysql -B -N $@ -e "SELECT DISTINCT CONCAT(
    'SHOW GRANTS FOR \'', user, '\'@\'', host, '\';'
    ) AS query FROM mysql.user" | \
  mysql $@ | \
  sed 's/\(GRANT .*\)/\1;/;s/^\(Grants for .*\)/## \1 ##/;/##/{x;p;x;}'
}
```

Ein Aufruf könnte dann so aussehen:

```
mygrants --host=prod-db1 --user=admin --password=secret
```

Allgemein

Datenbank und Benutzer löschen

Einen Benutzer inkl. Datenbank löschen (ab MySQL 5.0.2):

```
DROP USER user1;
DROP DATABASE userdb1;
```

Einen Benutzer inkl. Datenbank löschen (pre MySQL 4.1.1):

```
REVOKE ALL PRIVILEGES ON 'userdb1'.* FROM 'user1'@'123.123.123.123';

# und evtl.
REVOKE GRANT OPTION ON 'userdb1'.* FROM 'user1'@'123.123.123.123';

DELETE FROM mysql.user WHERE Host='123.123.123.123' AND User='root';
FLUSH PRIVILEGES;
DROP DATABASE userdb1;
```

Allgemein

InnoDB - ideale Logfile Größe herausfinden

von code.openark.com und MySQL Performance Blog

Um den idealen Wert für innodb_log_file_size für seine Datenbank herauszufinden, läßt sich sehr einfach dieser Code verwenden. Die Query erfasst über den Zeitraum von 60 Sekunden alle Schreiboperationen auf das InnoDB-Logfile. Idealer Zeitpunkt zur Ausführung ist während einer durchschittlichen Zugriffsmenge, nicht zur Peak-Zeit.

```
SELECT
  innodb_os_log_written_per_minute*60
    AS estimated_innodb_os_log_written_per_hour,
  CONCAT(ROUND(innodb_os_log_written_per_minute*60/1024/1024, 1), 'MB')
    AS estimated_innodb_os_log_written_per_hour_mb
FROM
  (SELECT SUM(value) AS innodb_os_log_written_per_minute FROM (
    SELECT -VARIABLE_VALUE AS value
      FROM INFORMATION_SCHEMA.GLOBAL_STATUS
      WHERE VARIABLE_NAME = 'innodb_os_log_written'
    UNION ALL
    SELECT SLEEP(60)
      FROM DUAL
    UNION ALL
    SELECT VARIABLE_VALUE
      FROM INFORMATION_SCHEMA.GLOBAL_STATUS
      WHERE VARIABLE_NAME = 'innodb_os_log_written'
  ) s1
) s2
;
```

Das sieht dann z.B. so aus:

```
+---------------------------------------+-------------------------------------------+
| estimated_innodb_os_log_written_per_hour | estimated_innodb_os_log_written_per_hour_mb |
+---------------------------------------+-------------------------------------------+
|                                 58920960 | 56.2MB                                    |
+---------------------------------------+-------------------------------------------+
```

Also sind es aufgerunden ca. 60 MB und da es standardmäßig 2 Logfiles gibt, lautet der ideale Wert:

```
innodb_log_file_size = 30M
```

Allgemein

Größe einer DB, Tabelle, aller DBs herausfinden

Ersetze 'YOUR_DATABASE_NAME' durch die gewünschte Datenbank.

Größe aller Tabellen einer Datenbank:

```
SELECT TABLE_SCHEMA AS 'Database_name', TABLE_NAME AS 'Table_Name',CONCAT(ROUND(((DATA_LENGTH
+ INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2)," Mb") AS Size FROM INFORMATION_SCHEMA.TABLES;
```

Größe bestimmter Tabellen:

```
SELECT TABLE_SCHEMA AS 'Database_name', TABLE_NAME AS 'Table_Name',CONCAT(ROUND(((DATA_LENGTH
+ INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2)," MB") AS Size FROM INFORMATION_SCHEMA.TABLES
where TABLE_SCHEMA = 'YOUR_DATABASE_NAME';
```

Größe einer bestimmten Datenbank:

```
SELECT CONCAT(sum(ROUND(((DATA_LENGTH + INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2))," MB") AS
Size FROM INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA =  'YOUR_DATABASE_NAME';
```

Größe aller Datenbanken:

```
SELECT table_schema AS "Database", sum( data_length + index_length ) / 1024 / 1024 AS "Size in
MB" FROM information_schema.TABLES GROUP BY table_schema;
```

Größe aller Datenbanken und wieviel Platz durch ein "OPTIMIZE TABLE" gewonnen werden kann:

```
SELECT table_schema "database name", round( sum( data_length + index_length ) / 1024 / 1024)
"database size in MB", round( sum( data_free )/ 1024 / 1024) "free space in MB" FROM
information_schema.TABLES GROUP BY table_schema;
```

Die 10 größten Tabellen anzeigen und wie viel Platz gewonnen werden könnte:

```
SELECT table_schema AS database_name,
       table_name,
       round( (data_length + index_length) / 1024 / 1024, 2) AS total_size,
       round( (data_length) / 1024 / 1024, 2) AS data_size,
       round( (index_length) / 1024 / 1024, 2) AS index_size,
```

```
       round( (data_free) / 1024 / 1024) AS free_space
FROM information_schema.tables
WHERE table_schema NOT IN ('information_schema', 'mysql', 'performance_schema' ,'sys')
      AND table_type = 'BASE TABLE'
      -- AND table_schema = '$i'
ORDER BY total_size DESC
LIMIT 10;
```

wo kann am meisten Platz gewonnen werden (sortiert nach free_space):

```
SELECT table_schema AS database_name,
       table_name,
       round( (data_length + index_length) / 1024 / 1024, 2) AS total_size,
       round( (data_length) / 1024 / 1024, 2) AS data_size,
       round( (index_length) / 1024 / 1024, 2) AS index_size,
       round( (data_free) / 1024 / 1024) AS free_space
FROM information_schema.tables
WHERE table_schema NOT IN ('information_schema', 'mysql', 'performance_schema' ,'sys')
      AND table_type = 'BASE TABLE'
      -- AND table_schema = '$i'
ORDER BY free_space DESC
LIMIT 10;
```

Allgemein

Passwort setzen oder ändern

mit „mysqladmin“ ein Passwort setzen oder ändern:

```
mysqladmin -h 'hostname' -u root password 'neues-passwort'

für localhost:
mysqladmin -u root password 'neues-passwort'
```

mittels SQL ein neues Passwort vergeben:

```
mysql -u root
mysql> GRANT ALL PRIVILEGES ON userdb1.* TO 'user1'@'123.123.123.123' IDENTIFIED BY 'neues-
passwort';
```

Allgemein

Passwort vergessen

Wenn das Kennwort für 'root' verloren ging / vergessen wurde und sonst auch keine Adminuser existieren, lassen sich die Kennwörter noch über den folgenden Weg zurücksetzen (getestet unter Debian).

Ist euer MySQL-Server direkt und öffentlich über das Internet erreichbar solltet ihr vorher den Zugriff darauf per iptables-Regel einschränken, da sich anschließend alle User ohne Kennwort einloggen können.

1. MySQL Server stoppen: /etc/init.d/mysql stop

2. in /etc/mysql/my.cnf wird diese Zeile hinzugefügt, damit werden beim Start die Berechtigungstabellen ignoriert und jeder(!) User kann sich ohne Kennwort einloggen.

```
--skip-grant-tables
```

3. MySQL Server wieder starten: /etc/init.d/mysql start

4. MySQL Client starten und Kennwörter neu setzen

```
# mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('newrootpassword') WHERE User='root';
exit
```

5. MySQL-Server wieder stoppen

6. „–skip-grant-tables“ wieder aus der my.cnf löschen

7. MySQL-Server wieder starten. Ihr solltet Euch nun mit dem neuen Kennwort einloggen können.

Allgemein

MySQL-Datenbank umbenennen

Der ganz sichere Weg ist, die DB per mysqldump zu exportieren und wieder in die neue DB zu importieren:

```
mysqldump alteDb | mysql neueDB
```

Für kurze Zeit gab es das MySQL-Kommando „RENAME DATABASE“, das aber wegen verschiedenen Problemen schnell wieder entfernt wurde.

Per „RENAME TABLE“ lassen sich Tabellen umbenennen und dabei auch in ein anderes Schema verschieben. Da das für jede Tabelle einzeln gemacht werden muss, kann das über dieses Script auch automatisiert werden:

```
#!/bin/bash
#
# rename a mysql database - this works both for MyISAM and InnoDB
#
# usage: rename-mysql-db.sh old new
# dbname as in "SHOW DATABASES"
#

# mysql command and authorization info if needed
mysql="mysql"
#mysql="mysql -uroot -pblabla"
#mysql="mysql -uroot -pblabla -S /var/lib/mysql/mysql.sock -h localhost"

olddb=${1}
newdb=${2}

${mysql} -e "CREATE DATABASE ${newdb}"
tables=$(${mysql} -N -e "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE
table_schema='${olddb}'")

for name in ${tables}; do
      ${mysql} -e "RENAME TABLE $olddb.$name to $newdb.$name";
done;

${mysql} -e "DROP DATABASE $olddb"
```

Oder einfach die notwendigen Kommandos für die manuelle Ausführung ausgeben lassen:

```
ALTE_DB="nextcloud_test"
NEUE_DB="bla_test"
for table in `mysql -s -N -e "USE $ALTE_DB; SHOW TABLES;"`; do echo mysql -s -N -e "USE
$ALTE_DB; RENAME TABLE $ALTE_DB.$table to $NEUE_DB.$table;"; done;
```

Allgemein

mysqltuner - MySQL Performance Tuning Check-Script

MySQLTuner is ein Perlscript, das versucht die MySQL Konfiguration zu prüfen und Vorschläge zum Tuning auf Basis der laufenden Installation zu machen.

Installation:

```
wget -O mysqltuner.pl http://mysqltuner.pl,
chmod u+x mysqltuner.pl
```

Ausführung:

```
./mysqltuner.pl

                                                                                Di 27.
Jul 2021 14:48:07
 >>  MySQLTuner 1.8.1 - Major Hayden <major@mhtx.net>
 >>  Bug reports, feature requests, and downloads at http://mysqltuner.pl/
 >>  Run with '--help' for additional options and output filtering

[--] Skipped version check for MySQLTuner script
[OK] Logged in using credentials from Debian maintenance account.
[OK] Currently running supported MySQL version 8.0.25-15
[OK] Operating on 64-bit architecture

-------- Log file Recommendations ------------------------------------------------------------------
[OK] Log file /var/log/mysql/error.log exists
[--] Log file: /var/log/mysql/error.log(0B)
[--] Log file /var/log/mysql/error.log is empty. Assuming log-rotation. Use --server-
log={file} for explicit file

-------- Storage Engine Statistics -----------------------------------------------------------------
[--] Status: +ARCHIVE +BLACKHOLE +CSV -FEDERATED +InnoDB +MEMORY +MRG_MYISAM +MyISAM
+PERFORMANCE_SCHEMA
[--] Data in InnoDB tables: 525.5M (Tables: 650)
```

```
[--] Data in MyISAM tables: 28.9M (Tables: 160)
[OK] Total fragmented tables: 0

-------- Analysis Performance Metrics
----------------------------------------------------------------
[--] innodb_stats_on_metadata: OFF
[OK] No stat updates during querying INFORMATION_SCHEMA.

-------- Security Recommendations
-------------------------------------------------------------------
[--] Skipped due to unsupported feature for MySQL 8

-------- CVE Security Recommendations
-----------------------------------------------------------------
[--] Skipped due to --cvefile option undefined

-------- Performance Metrics
--------------------------------------------------------------------------
[--] Up for: 4d 12h 47m 32s (33M q [85.194 qps], 482K conn, TX: 155G, RX: 6G)
[--] Reads / Writes: 98% / 2%
[--] Binary logging is enabled (GTID MODE: OFF)
[--] Physical Memory     : 23.5G
[--] Max MySQL memory    : 9.9G
[--] Other process memory: 0B
[--] Total buffers: 3.4G global + 133.6M per thread (50 max threads)
[--] P_S Max memory usage: 72B
[--] Galera GCache Max memory usage: 0B
[OK] Maximum reached memory usage: 7.7G (32.72% of installed RAM)
[OK] Maximum possible memory usage: 9.9G (42.15% of installed RAM)
[OK] Overall possible memory usage with other process is compatible with memory available
[!!] Slow queries: 10% (3M/33M)
[OK] Highest usage of available connections: 66% (33/50)
[OK] Aborted connections: 0.27%  (1314/482261)
[--] Query cache have been removed in MySQL 8
[OK] Sorts requiring temporary tables: 0% (0 temp sorts / 435K sorts)
[!!] Joins performed without indexes: 16421
[OK] Temporary tables created on disk: 12% (78K on disk / 652K total)
[OK] Thread cache hit rate: 99% (49 created / 482K connections)
[OK] Table cache hit rate: 99% (37M hits / 37M requests)
[OK] table_definition_cache(2000) is upper than number of tables(1149)
```

```
[OK] Open file limit used: 0% (2K/1M)
[OK] Table locks acquired immediately: 100% (1M immediate / 1M locks)
[OK] Binlog cache memory access: 99.99% (463714 Memory / 463765 Total)

-------- Performance schema
--------------------------------------------------------------------------------
[--] Memory used by P_S: 72B
[--] Sys schema is installed.

-------- ThreadPool Metrics
--------------------------------------------------------------------------------
[--] ThreadPool stat is enabled.
[--] Thread Pool Size: 24 thread(s).
[OK] thread_pool_size between 16 and 36 when using InnoDB storage engine.

-------- MyISAM Metrics
--------------------------------------------------------------------------------
[--] MyISAM Metrics are disabled on last MySQL versions.

-------- InnoDB Metrics
--------------------------------------------------------------------------------
[--] InnoDB is enabled.
[--] InnoDB Thread Concurrency: 9
[OK] InnoDB File per table is activated
[OK] InnoDB buffer pool / data size: 3.0G/525.5M
[OK] Ratio InnoDB log file size / InnoDB Buffer pool size: 384.0M * 2/3.0G should be equal to
25%
[OK] InnoDB buffer pool instances: 3
[--] Number of InnoDB Buffer Pool Chunk : 24 for 3 Buffer Pool Instance(s)
[OK] Innodb_buffer_pool_size aligned with Innodb_buffer_pool_chunk_size &
Innodb_buffer_pool_instances
[OK] InnoDB Read buffer efficiency: 100.00% (3574568225 hits/ 3574619782 total)
[!!] InnoDB Write Log efficiency: 64.54% (3246059 hits/ 5029381 total)
[OK] InnoDB log waits: 0.00% (0 waits / 1783322 writes)

-------- Aria Metrics
--------------------------------------------------------------------------------
[--] Aria Storage Engine not available.

-------- TokuDB Metrics
```

```
-------------------------------------------------------------------------------
[--] TokuDB is disabled.

-------- XtraDB Metrics
-------------------------------------------------------------------------------
[--] XtraDB is disabled.

-------- Galera Metrics
-------------------------------------------------------------------------------
[--] Galera is disabled.

-------- Replication Metrics
-----------------------------------------------------------------------
[--] Galera Synchronous replication: NO
[--] No replication slave(s) for this server.
[--] Binlog format: MIXED
[--] XA support enabled: ON
[--] Semi synchronous replication Master: Not Activated
[--] Semi synchronous replication Slave: Not Activated
[--] This is a standalone server

-------- Recommendations
-------------------------------------------------------------------------------
General recommendations:
    We will suggest raising the 'join_buffer_size' until JOINs not using indexes are found.
              See https://dev.mysql.com/doc/internals/en/join-buffer-size.html
              (specially the conclusions at the bottom of the page).
Variables to adjust:
    join_buffer_size (> 4.0M, or always use indexes with JOINs)
```

Allgemein

Schnell Platz schaffen

Wenn dein Server binlogs schreibt und die Disk voll läuft kann es helfen schnell Platz zu schaffen indem man nicht mehr benötigte binlogs löscht.

Hier gibts zwei Möglichkeiten.

Mit der Variablen expire_logs_days (akzeptiert Integer-Werte von 0-99) läßt sich der Zeitraum einstellen wie lange Binlogs in Tagen aufgehoben werden sollen. Wenn man den Wert auf 0 setzt werden die Binlogs gar nicht mehr gelöscht. Default-Wert ist 7.

Platz schaffen läßt sich also z.B. so:

```
mysql> SHOW VARIABLES LIKE 'expire_logs_days';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| expire_logs_days | 3     |
+------------------+-------+
1 row in set (0.01 sec)

mysql> SET GLOBAL expire_logs_days=3;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.15 sec)
```

Der Wert wird nicht dauerhaft gespeichert und steht nach einem Restart der DB wieder auf Default, bzw. auf dem Wert, der in eurer Config gesetzt ist.

Man kann auch direkt alte Binlogs löschen, das läßt sich dann auch feiner steuern. In diesem Beispiel werden alle Binlogs älter als eine Stunde gelöscht:

```
mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.15 sec)

mysql> PURGE BINARY LOGS BEFORE NOW() - INTERVAL 1 HOUR;
Query OK, 0 rows affected (2.23 sec)
```

> Falls man ein Replication-Setup hat muss man natürlich darauf achten, dass der Slave auf aktuellem Stand ist und man nicht zuviele Logs löscht!

Allgemein

Ungenutzte Datenbanken finden

Dieses Snippet listet die neueste Datei in jedem Datenbankverzeichnis unterhalb von /var/lib/mysql auf:

```
cd /var/lib/mysql
for i in `mysql --skip-column-names -B -e "show databases" | grep -v -E "mysql|information_schema|sys|performance_schema|percona_schema"`; do echo -ne "\n--- $i ---\nSize: "; du -hs $i; find $i -type f -print0 | xargs -0 stat --format '%Y :%y %n' | sort -nr | cut -d: -f2- | head -n1; done
```

Beispiel:

```
--- database1 ---
Size: 121M      database1
2024-12-16 21:12:42.437939063 +0100 database1/Portal_PortalRequestLog.ibd

--- database2 ---
Size: 17G       database2
2024-12-10 06:24:49.902646396 +0100 database2/app_session.ibd

--- database3 ---
Size: 152M      database3
2024-10-10 14:20:10.301089103 +0200 database3/layout_file.frm
```

Fehlerbehandlung

Fehlerbehandlung

Replikationsfehler Errno: 1236

Replikationsfehler Errno: 1236: start replication from impossible position

```
Slave I/O: Got fatal error 1236 from master when reading data from binary log: 'Client requested master to start replication from impossible position; the first event mysql-bin.010346 at 5134523, the last event read from /var/lib/mysql/mysql-bin.000346 at 4, the last byte read from /var/lib/mysql/mysql-bin.010346 at 4., Error_code: 1236
```

Dieser Fehler kann mehrere Ursachen haben. Die häufigste Ursache bei meinen Server war ein harter Crash des Master-Servers. Dabei wurden die letzten Binlog-Einträge nicht auf die Disk geschrieben oder es landete Müll darin.

Das kann man herausfinden, wenn man sich das letzte Binlog (auf dem Master) ansieht. Sollte das Log hier zu Ende sein, muss man dem Slave das darauf folgende Logfile und die neue Logposition mitteilen.

```
$ mysqlbinlog --no-defaults --base64-output=decode-rows --verbose --verbose --start-position=5134523 mysql-bin.010346
```

Das nächste Logfile wäre in diesem Fall /var/lib/mysql/mysql-bin.000347, bei Position 4.

```
mysql> STOP SLAVE;
mysql> CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000347', MASTER_LOG_POS=4;
mysql> START SLAVE;
```

Die Replikation sollte nun wieder normal weiterlaufen.

Um dem Fehler entgegen zui wirken, kann man „sync_binlog = 1“ in der my.cnf im [mysqld]-Bereich setzen. Damit führt MySQL nach jedem commit einen fsync auf der Disk aus. Bei entsprechend schnellen Platten bedeutet diese Option nur einen minimalen Performanceverlust.

Fehlerbehandlung

Replikationsfehler Errno: 1580 ALTER Logtable

Nach einem Update von MySQL bei einer Master-Slave-Replikation konnte die Replikation auf einem Slave-Server kürzlich nicht mehr gestartet werden.

```
Last_Errno: 1580
Last_Error: Error 'You cannot 'ALTER' a log table if logging is enabled' on query. Default
database: 'mysql'. Query: 'ALTER TABLE slow_log
```

Offenbar hat der Slave Probleme, das „`ALTER TABLE`“ anzuwenden, sofern das Logging noch aktiv ist (in diesem Fall das Slow-Query Log).

Der Fehler läßt sich relativ einfach beheben (über die MySQL CLI):

```
STOP SLAVE;
SET GLOBAL slow_query_log = 'OFF';
START SLAVE;
SHOW SLAVE STATUS\G
```

Der Fehler sollte nun verschwunden sein, der Slave konnte die Query also auf seine DB anwenden. Nun kann auch das Slow-Query Log wieder aktiviert werden:

```
mysql> SET GLOBAL slow_query_log = 'ON';
```

Fehlerbehandlung

Replikationsfehler Errno: 1594 Relay log read failure

Wenn ein Server, auf dem ein MySQL-Replication-Slave läuft abstürzt, kann es passieren, dass die Master-Slave-Replikation nicht mehr funktioniert. Beim Absturz landen korrupte Daten in den Relay-Logs, die von MySQL nach dem Neustart nicht verarbeitet werden können. Im SLAVE STATUS steht dann folgendes:

Last_Errno: 1594
Last_Error: Relay log read failure: Could not parse relay log event entry. The possible reasons are: the master's binary log is corrupted (you can check this by running 'mysqlbinlog' on the binary log), the slave's relay log is corrupted (you can check this by running 'mysqlbinlog' on the relay log), a network problem, or a bug in the master's or slave's MySQL code. If you want to check the master's binary log or slave's relay log, you will be able to know their names by issuing 'SHOW SLAVE STATUS' on this slave.
In diesem Fall reicht es häufig die Relay-Logs vom Slave zu löschen und die Replikation neu zu starten. Ein erneuter Dump der Datenbank ist dabei nicht notwendig. Wir sagen dem Server einfach, wo er unterbrochen wurde und setzen die Replikation an dieser Stelle neu auf.

1. Slave-Prozesse stoppen: `STOP SLAVE;`
2. Aus `SLAVE STATUS` die folgenden Werte merken: „Relay_Master_Log_File“ und „Exec_Master_Log_Pos“
3. Slave zurücksetzen, dabei werden die korrupten Logs gelöscht: `RESET SLAVE;`
4. jetzt sagen wir dem Slave an welcher Stelle die Replikation unterbrochen wurde mit den Werten aus dem zweiten Schritt. Master-Host, Username usw. müssen wir dabei nicht nochmal eingeben: `CHANGE MASTER TO MASTER_LOG_FILE='Relay_Master_Log_File', MASTER_LOG_POS=Exec_Master_Log_Pos;`
5. nun kann der Slave wieder gestartet werden und sollte an der richtigen Stelle weitermachen. Das zuvor korrupte Binlog wird erneut vom Master übertragen: `START SLAVE;`

Fehlerbehandlung

Fehler bei ALTER TABLE Errno 150: Foreign key constraint is incorrectly formed

Dieser Fehler trat bei einem „`ALTER TABLE `table` ENGINE = InnoDB`“ auf (die Tabelle sollte von MyISAM auf InnoDB umgestellt werden). Eine Spalte war von einer anderen Tabelle (bereits InnoDB) referenziert. Nach dem der Contraint von der anderen Tabelle entfernt worden war, ließ sich die Tabelle konvertieren. Anschließend kann der Constraint wieder hinzugefügt werden.

Mit diesem Query lassen sich Referenzen von externen Tabellen auf die zu konvertierende Tabelle anzeigen:

```
SELECT
  TABLE_NAME,COLUMN_NAME,CONSTRAINT_NAME, REFERENCED_TABLE_NAME,REFERENCED_COLUMN_NAME
FROM
  INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
  REFERENCED_TABLE_SCHEMA = '<database>' AND
  REFERENCED_TABLE_NAME = '<table>';
```

Fehlerbehandlung

InnoDB - defekte FRM Files wiederherstellen - Incorrect information in file

Sollten InnoDB-Datenbanken korrupt sein und auch keinen Dump mehr erlauben, da diese mit folgendem Fehler abbrechen „Incorrect information in file: ‘<table name> .frm“, kann man sich vielleicht noch mit diesem Vorgehen helfen (InnoDB erlaubt auch kein „repair tables“):

File Backup erstellen:

```
# /etc/init.d/mysqld stop
# mkdir /root/mysql_backup
# cp -rv /var/lib/mysql/* /root/mysql_backup/
```

In die MySQL Konfigurationsdatei my.cnf unter [mysqld] innodb_force_recovery = 4 setzen und die Datenbank wieder starten via:

```
mysqldump -uroot -pPASSWORD -A > dump.sql
```

einen Dump erstellen, sollte dieser fehlschlagen, stoppen wir die Datenbank und erhoehen den innodb_force_recovery auf 5 oder im naechsten Schritt auf 6. Laeuft der Dump durch, wird die Datenbank gestoppt und alle Datenbanken und Tabellen geloescht:

```
rm -rf /var/lib/mysql/*
```

Nun wird innodb_force_recovery in der my.cnf auskommentiert und die Datenbank wieder gestartet. Jetzt koennen wir die Datenbanken wieder einspielen:

```
mysql -uroot -pPASSWORD < dump.sql
```

Nachdem die Datenbanken nun wieder laufen erstellen wir uns einen Cronjob fuer das taegliche Sichern aller Datenbanken.

Probleme mit Umlauten zwischen Datenbank und Webseite (UTF-8/latin1/ISO8859-1)

Bei der Wiederherstellung oder dem Umzug einer Datenbank kann es manchmal zu Problemen kommen. Sonderzeichen und Umlaute werden dann auf der Webseite nicht mehr korrekt dargestellt. Eine Ursache ist z.B. wenn auf dem alten Server die Datenbank als Latin1 gedumpt wird und auf dem neuen ohne vorherige Konvertierung in eine UTF-8 DB importiert wird. MySQL und PHP gehen dann davon aus, dass die Daten in UTF-8 sind, obwohl diese eigentlich im Latin1 Zeichensatz vorliegen.

Dieses Problem läßt sich wie folgt beheben.

Dump der Datenbank, wir sagen dem mysqldump explizit, dass es sich um Latin1 handelt:

```
mysqldump --skip-extended-insert --default-character-set=latin1 <database> -r <database>-latin1.sql
```

- -skip-extended-insert fasst mehrere Zeilen nicht mehr in einem einzelnen INSERT zusammen. Das erleichtert einen Diff, verlangsamt allerdings bei großen Datenbanken den Reimport deutlich (optional)
- -r speichert den Output direkt in eine Datei. Es soll schon vorgekommen sein, dass durch die Umleitung von STDOUT über > in eine Datei komische Sachen passiert sind

Mit dem Tool iconv wird nun das Dumpfile tatsächlich auf UTF-8 gesetzt. Der Parameter „-c“ ist dabei optional. Er läßt Zeichen, die nicht konvertiert werden können einfach weg (Achtung: möglicher Datenverlust).

```
iconv -f UTF-8 -t UTF-8 -c <database>-latin1.sql > <database>-utf8.sql
```

Wenn wir nun die beiden Dateien vergleichen, sollten in der utf8-Datei die Umlaute korrekt angezeigt werden:

```
diff <database>-latin1.sql <database>-utf8.sql
```

Im Dumpfile müssen eventuelle Datenbanken/Tabellen noch auf UTF-8 gestellt werden:

```
# grep latin1 <database>-utf8.sql
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
  `name` varchar(255) CHARACTER SET latin1 NOT NULL,
  `description` text CHARACTER SET latin1 NOT NULL,
  `source` varchar(255) CHARACTER SET latin1 NOT NULL,

# sed -i 's/latin1/utf8/g' <database>-utf8.sql
```

Der MySQL-Server und die verbindenden Clients sollten nun natürlich auch UTF-8 liefern/schreiben. Dazu brauchts ein paar Ergänzungen an den entsprechenden Stellen in der my.cnf:

```
[client]
[...]
default-character-set   = utf8

[...]

[mysqld]
[...]
character-set-server    = utf8
collation-server        = utf8_unicode_ci
```

Danach den MySQL-Server neu starten (Achtung: Das nachträgliche Umstellen des Zeichensatzes kann unter Umständen bei bestehenden Datenbanken für Probleme sorgen).

```
# service mysql restart
```

Die Einstellungen überprüfen:

```
mysql> show variables like '%character_set_%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

Nun kann man den Dump einspielen:

```
mysql <datenbank> < <database>-utf8.sql
```

Backup

Backup

Bestimmte Datenbanken vom Dump ausschließen

```
DBEXCLUDE=will_nicht
MYSQL=$(mysql -N <<<"SHOW DATABASES" | grep -v ${DBEXCLUDE} | tr "\n" " ")
mysqldump --databases ${MYSQL} > mysqldump.sql
```

Dump einer einzelnen Datenbank

```
mysqldump -p --create-options -Q -c --add-drop-table --add-locks -F DATENBANKNAME > db-dump.sql
```

wenn der Dump gleich mit bzip2 gepackt werden soll:

```
mysqldump -p --create-options -Q -c --add-drop-table --add-locks -F DATENBANKNAME | bzip2 > db-dump.sql
```

Sollte die Datenbank eine Authentifizierung benötigen sind noch folgende Parameter zu ergänzen:

```
-u <username>
-p
```

Das Passwort wird dann nach dem Drücken von Eingabe abgefragt (könnte aber auch gleich als Parameter mit übergeben werden (z.B. `-p"meinpasswort"`).

Backup

Extraktion einer einzelnen Datenbank oder Tabelle aus einem Fulldump

Eine einzelne Datenbank aus einem Fulldump direkt wiederherzustellen ist ganz einfach:

```
# mysql -u root -p --one-database MeineDB < fulldump.sql
```

Wenn aber nur der SQL-Dump der Datenbank aus dem Fulldump gezogen werden soll hilft dieses sed-Kommando:

```
# sed -n '/^-- Current Database: `MeineDB`/,/^-- Current Database: `/p' fulldump.sql > MeineDB.sql
```

und nur eine einzelne Tabelle einer DB aus dem Dump zu ziehen geht so:

```
# sed -n '/^-- Current Database: `MeineDB`/,/^-- Current Database: `/p' fulldump.sql | sed -n '/^CREATE TABLE `MeineTabelle`/,/^DROP TABLE /p' | sed -n -e :a -e '1,6!{P;N;D;};N;ba' > MeineDB.MeineTabelle.sql
```

Backup

kompletten Dump eines MySQL-Servers erstellen

folgender Befehl erstellt einen kompletten Dump des MySQL-Servers (Parameter p weglassen, wenn localhost/root kein Passwort hat):

```
mysqldump -p -A -a -Q -c --add-drop-table --add-locks -F > db-dump.sql
```

Das ganze kann auch gleich gepackt werden, falls die größe des Dumps etwa die Platte sprengen würde:

```
mysqldump -p -A -a -Q -c --add-drop-table --add-locks -F | bzip2 > db-dump.sql
```

Backup

Mit netcat einen mysqldump übers Netzwerk schicken

Wenn mal der Diskspace knapp wird, aber man trotzdem einen Fulldump der MySQL-DB braucht, lässt sich netcat dafür gut einsetzen, um den Dump auf einen entfernten Server zu kopieren. Zu beachten ist, dass die Verbindung unverschlüsselt ist!

Zuerst wird auf dem Zielserver der Listener gestartet, die Standardausgabe wird in eine Datei umgeleitet. Die Daten werden beim Übertragen direkt durch gzip geschickt, die Datenmenge verringert sich dadurch normalerweise deutlich.

```
nc -l 2000 > fulldump.sql.gz
# oder gleich wieder entpacken:
nc -l 2000 | gunzip > fulldump.sql
```

Auf dem Quellserver wird die Ausgabe von mysqldump durch gzip gepiped an netcat geschickt, dieser überträgt es an den entfernten Listener (in diesem Beispiel 123.123.123.123). Es sollte entweder ein MySQL-Account ohne Passwort (unschön, könnte aber auch nur ein temporärer Account sein) verwendet werden oder das Passwort mit -p„<Passwort>“ gleich auf der Kommandozeile mitgegeben werden (eigentlich auch unschön). Wird nur -p verwendet, wird der Passwortprompt schon an den Zielserver übertragen, da ja die Standardausgabe umgeleitet ist. Es hilft schon ein Leerzeichen an den Anfang der mysqldump-Zeile zu setzen, damit diese nicht in der Bash-History auftaucht.

```
mysqldump -u root -p"<password>" -A | gzip | nc -w1 123.123.123.123 2000
```

Es lassen sich mit den üblichen mysqldump-Optionen natürlich auch nur einzelne Datenbanken übertragen.

Backup

komplettes MySQL-Dumpfile einspielen

Über das folgende Kommando läßt sich ein komplettes MySQL-Backup einspielen. Idealerweise wird der Vorgang in screen oder tmux gestartet, damit er nicht abgebrochen wird, wenn mal die SSH-Session stirbt.

Für die Dauer des Einspielens werden dabei automatische Commits und die Checks auf foreign keys deaktiviert. Sofern das Backup in sich konsistent ist sollte es hierbei auch zu keinen Fehlern kommen und das Einspielen geht deutlich schneller.

```
( echo "SET AUTOCOMMIT=0;"; echo "SET FOREIGN_KEY_CHECKS=0;"; cat backup.sql; echo "SET FOREIGN_KEY_CHECKS=1;"; echo "COMMIT;"; echo "SET AUTOCOMMIT=1;"; ) | mysql
```

Replikation

MySQL Master/Slave-Replikation einrichten (online) (veraltet)

> Achtung, dieses Feature wird von aktuellen MySQL Versionen nicht mehr unterstützt.

Auf dem Master-Server die Datei /etc/my.cnf ändern und in der [mysqld]-Sektion folgendes einfügen:

```
server-id = 1
log-bin
```

Auf dem Slave-Server die Datei /etc/my.cnf ändern und in der [mysqld]-Sektion folgendes einfügen:

```
server-id = 2
#replicate-do-db = database1 # um nur bestimmte Datenbanken zu replizieren
```

Master- und Slave-Server neu starten

```
service mysqld restart
```

Auf dem Master-Server einen User für die Replikation einrichten:

```
mysql> GRANT SUPER,REPLICATION CLIENT,REPLICATION SLAVE,RELOAD ON *.* TO repl@"slave.host"
IDENTIFIED BY 'password';
mysql> STOP SLAVE; # wenn bereits gestartet
mysql> START SLAVE;
```

Auf dem Slave-Server:

```
mysql> LOAD DATA FROM MASTER;
```

Replikation

MySQL Master/Slave-Replikation einrichten (offline)

Einfach und umkompliziert eine MySQL Master/Slave-Replikation einrichten (OFFLINE, empfohlen bei InnoDB-Datenbanken)

- User für Replikation einrichten:

```
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%.domain.de' IDENTIFIED BY 'slavepass';
```

- Auf dem Master-Server /etc/my.cnf bearbeiten, in der [mysqld]-Sektion ergänzen und danach den MySQL-Server neu starten:

```
# replikation
log-bin=mysql-bin
server-id=1
innodb_flush_log_at_trx_commit=1
sync_binlog=1
```

- Auf dem Slave-Server /etc/my.cnf bearbeiten, in der [mysqld]-Sektion ergänzen und danach den MySQL-Server neu starten:

```
# replikation
server-id=2
```

- Auf dem Master-Server die Tables speeren, die Datenbank anhalten und die binary-Dateien auf das Slave-System kopieren:

```
mysql -p
mysql> FLUSH TABLES WITH READ LOCK;
( eventuell bei InnoDB mit Transaktionen den Server ganz stoppen: service mysqld stop )
```

- Achtung: Die Shell jetzt NICHT beenden, da der Schreibschutz sonst wieder aufgehoben wird!

```
cd /var/lib
rsync -avr -e ssh mysql sqlserver2.domain.de:/var/lib/
```

- Nach dem Kopieren wird der Master wieder freigegeben:

```
( wurde der Server vorher gestoppt, so muss er jetzt wieder gestartet werden: service mysqld start )
mysql> SHOW MASTER STATUS;
mysql> UNLOCK TABLES;
```

Die Werte von File und Position merken, die werden gleich gebraucht.

- Auf dem Slave-Server die Datenbank starten, die Replikation konfigurieren und starten:

```
service mysqld start
mysql -u root -p
mysql> CHANGE MASTER to MASTER_HOST='sqlserver1.domain.de’,MASTER_PORT=3306,
MASTER_USER='repl',MASTER_PASSWORD='slavepass',MASTER_LOG_FILE='<logfilename>',MASTER_LOG_POS=
<logfileposition>;
mysql> START SLAVE;
```

Master-Master-Replikation

Dieser Artikel zeigt, wie man mit 2 (oder mehr) MySQL-Servern eine echte Master-Master-Replikation aufsetzt, um die Server HA-tauglich zu machen.

Es gibt 2 Server:

- **mysql1** 192.168.1.10
- **mysql2** 192.168.1.11

Als Heartbeat-IP kann man irgendeine IP verwenden, da MySQL automatisch auf allen konfigurierten IP-Adressen lauscht.

- /etc/my.cnf auf **mysql1** anpassen

```
[mysqld]
...
server-id=1
log-bin
...
```

- auf **mysql1** den Useraccount für die Replikation anlegen

```
mysql> grant replication slave on *.* to 'replication'@192.168.1.11 identified by 'slave';
```

- MySQL-Server auf **mysql1** neu starten
- Nun auf **mysql2** die Datei /etc/my.cnf anpassen

```
[mysqld]
...
server-id=2
master-host = 192.168.128.1
master-user = replication
master-password = slave
master-port = 3306
...
```

- auf **mysql2** den MysSQL-Server neu starten und in der MySQL-Konsole den Slave-Status ausgeben

```
mysql> show slave status\G;


*************************** 1. row ***************************
```

```
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.1.10
Master_User: replica
Master_Port: 3306
...
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
1 row in set (0.00 sec)
```

- Auf dem **mysql1** kontrolliert man den Master-Status

```
mysql> show master status;

+-----------------------+----------+--------------+------------------+
| File                  | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----------------------+----------+--------------+------------------+
|MysqlMYSQL01-bin.000016 |      132 |              | foobar           |
+-----------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

- nun haben wir bereits ein funktionierenden Master-Slave-Setup, in den weiteren Schritten konfigurieren wir uns ein echtes Master-Master-Setup
- auf **mysql2** /etc/my.cnf anpassen

```
[mysqld]
...
server-id=2
log-bin
binlog-ignore-db=mysql
master-host = 192.168.1.10
master-user = replication
master-password = slave
master-port = 3306
```

- den Replikationsaccount auf **mysql2** anlegen

```
mysql> grant replication slave on *.* to 'replication'@192.168.1.10 identified by 'slave';
```

- MySQL-Server auf **mysql2** neu starten
- /etc/my.cnf auf **mysql1** anpassen

```
[mysqld]
...
master-host = 192.168.128.2
master-user = replication
master-password = slave
master-port = 3306
...
```

- jetzt auch auf **mysql1** den MySQL-Server neu starten
- mit den altbekannten Kommandos „show slave status“ und „show master status“ den aktuellen Stand der Replikation kontrollieren. Auf beiden Systemen sollten folgende Zeilen erscheinen:

```
Slave_IO_Running:□Yes
Slave_SQL_Running:□Yes
```

- im Falle eines Fehlers kann man in /var/log/mysqld.log nachlesen, was den Server grade stört

Replikation

Master-Slave-Replikation einrichten mit innobackupex

Mit innobackupex läßt sich auf einfache Weise im Live-Betrieb eine Master-Slave-Replikation einrichten. Die Datenbank muss ggf. nur für Änderungen an der Konfiguration neu gestartet werden (sofern nicht schon vorher entsprechend konfiguriert).

Voraussetzungen:

- ein Masterserver mit MySQL und vollen Benutzerrechten (root)
- auf dem Master müssen binlogs eingeschaltet und die server-id auf 1 gesetzt sein
- ein Slaveserver mit MySQL und vollen Benutzerrechten (root)
- auf dem Slaveserver müssen binlogs eingeschaltet (ist ggf. wichtig, wenn man einen Slave aus dem Slave erstellen will) und die server-id auf 2 gesetzt sein
- rsync auf beiden Servern

Zuerst installieren wir das Tool innobackupex, bzw. das Paket percona-xtrabackup. Das Toolkit läßt sich entweder direkt als .deb Paket herunterladen oder über das Percona-Repository installieren. Ich verwende das Repository.

Am einfachsten läßt sich das Percona Repo über das zur Verfügung gestellt .deb installieren. Dies fügt den Percona-GPG Key hinzu und legt ein entsprechendes sources.list-File an:

Percona Repository anlegen

```
wget https://repo.percona.com/apt/percona-release_0.1-4.$(lsb_release -sc)_all.deb
dpkg -i percona-release_0.1-4.$(lsb_release -sc)_all.deb
apt-get update
apt-get -y install percona-xtrabackup qpress
# oder für die neuere Version 2.4
apt-get -y install percona-xtrabackup-24 qpress
```

`qpress` wird für die komprimierten Backups benötigt. Dazu unten mehr.

Um in den nächsten Schritten mit weniger Passwortabfragen auszukommen, lege ich eine Datei mit den MySQL-Zugangsdaten an, die von allen Tools ausgelesen wird.

/root/.my.cnf

```
[client]
user=root
pass=secret
```

Jetzt ziehen wir eine Binärkopie/Snapshot des aktuellen MySQL-Datadirs (/var/lib/mysql) in unser Sicherungsverzeichnis. Darauf wird dann das aktuelle Binlog angewendet, um einen definitiven Zustand zu erhalten. Der erste Befehl legt das Backup in ein Unterverzeichnis von /backup mit dem aktuellen <TIMESTAMP>, das muss im zweiten Befehl berücksichtigt werden:

```
mkdir /backup
innobackupex --compress --compress-threads=6 --rsync --slave-info /backup
```

Da das Backup komprimiert wird (kann auch weggelassen werden), werden wir das apply-log erst auf dem Ziel durchführen. Siehe unten. Wenn man nicht komprimiert, kann man –apply-log direkt ausführen.

Jetzt muss noch der Slaveserver vorbereitet werden. Dazu wird der evtl. schon laufende MySQL-Server gestoppt und das Datendir gelöscht:

```
systemctl stop mysql.service
rm -rf /var/lib/mysql
```

Nun wird das Backup vom Master auf den Slave kopiert:

```
rsync -av -e ssh /backup/<TIMESTAMP> slave:/var/lib/mysql
```

Auf dem Slave muss das Backup jetzt entpackt und vorbereitet werden:

```
innobackupex --decompress /var/lib/mysql/
innobackupex --apply-log /var/lib/mysql/
```

Auf dem Slave setzen wir nun noch passende Berechtigungen:

```
chown -R mysql.mysql /var/lib/mysql
```

Der „debian-sys-maint“-User auf dem Slave hat nun das Kennwort vom Master. Das müssen wir in der Konfigurationsdatei anpassen (/etc/mysql/debian.cnf), damit u.a. das Init-Script mit dem MySQL-Server kommunizieren kann.

Danach kann der MySQL auf dem Slave mit den Daten vom Master gestartet werden:

```
systemctl start mysql.service
```

Für die Replikation brauchen wir noch einen User mit entsprechenden Rechten, root sollte dafür natürlich nicht verwendet werden. Auf dem Master legen wir den User wie folgt an (IP ist dabei die IP-Adresse des Slave-Servers, ein sicheres Passwort sollte gewählt werden):

```
mysql> GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'IP' IDENTIFIED BY '$PASS';
```

Vom Slave aus testen, ob der Zugang funktioniert:

```
mysql -h master-server -u repl -p$PASS
```

Im letzten Schritt sagen wir dem Slave wer sein Herr ist und an welcher Position im Binlog die Replikation beginnen soll. Diese Infos stehen auf dem durch innobackupex angelegten File (xtrabackup_binlog_info, in diesem Beispiel lief der Master schon länger, daher die hohen Werte).

/var/lib/mysql/xtrabackup_binlog_info

```
mysql-bin.002284 40083139
```

In der MySQL-Shell konfigurieren wir den Slave:

```
mysql> change master to master_host='$MASTER-IP', master_user='repl', master_password='$PASS',
master_log_file='mysql-bin.002284', master_log_pos=40083139;

mysql> start slave;
```

Den Status der Replikation kann man mit diesem Befehl überprüfen:

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: $MASTER-IP
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.002289
          Read_Master_Log_Pos: 16815558
               Relay_Log_File: mysql-relay-bin.000016
                Relay_Log_Pos: 15880380
        Relay_Master_Log_File: mysql-bin.002289
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 15880234
              Relay_Log_Space: 16815903
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
```

```
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 49
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
1 row in set (0.00 sec)
```

Dabei sollte Slave_IO_Running und Slave_SQL_Running auf YES stehen. Seconds_Behind_Master zeigt ungefähr wieviele Sekunden der Slave dem Master-Server hinterherhängt.

kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.5

Diese Anleitung wird nicht weiter aktualisiert. Bitte wechselt auf die Versionen 5.7 oder 8.0

Dieses Beispiel ist für kleine bis mittelgroße Setups geeignet. Die Vorlage kann einfach nach /etc/mysql/my.cnf kopiert werden. Sofern die Datenbankengine MyISAM nicht benötigt wird, kann diese über „skip-myisam“ im [mysqld]-Bereich entfernt werden.

Um die neuen Einstellungen zu aktiveren, müssen sehr wahrscheinlich einmal die InnoDB-Logfiles gelöscht werden (MySQL läßt sich sonst nicht starten). Diese werden dann beim nächsten Start der Datenbank neu angelegt. Je nach Größe kann das auch länger dauern, das Init-Script beendet sich dann gerne mit einem Fehler. Den aktuellen Status sieht man im MySQL-Logfile (bzw. /var/log/syslog). Die Logfiles werden wie folgt gelöscht:

```
service mysql stop
rm -f /var/lib/mysql/ib_logfile*
service mysql start
```

> **Sämtliche Einstellungen sind von Dir dann natürlich nochmals zu überprüfen**, insbesondere Caches und Speicherverbrauch. Dabei kann dann auch das Tool mysqltuner.pl helfen. Mein Beispiel ist auf ein Hot-Dataset von 2GB InnoDB Daten ausgelegt.

Bei **Debian Wheezy**: Anpassung der Systemlimits, diese Zeilen am Ende vor der letzten Zeile ergänzen:

/etc/security/limits.conf

```
*    soft    nofile    65535
*    hard    nofile    65535
```

Ab **Debian Jessie**, bzw. Distributionen mit Systemd, muss das Systemd unit file angepasst werden (danach „systemctl daemon-reload“ ausführen):

/lib/systemd/system/mysql.service

```
LimitNOFILE=infinity
LimitMEMLOCK=infinity
```

/etc/mysql/my.cnf

```
#
# The MySQL database server configuration file
# compact version from https://wiki.magenbrot.net/datenbanken/mysql/kompakte_mysql_my.cnf_v
#

[client]
port                           = 3306
socket                         = /var/run/mysqld/mysqld.sock
```

```
default_character_set           = utf8

[mysqld_safe]
socket                          = /var/run/mysqld/mysqld.sock
nice                            = 0

[mysqld]
# basic settings
user                            = mysql
pid_file                        = /var/run/mysqld/mysqld.pid
socket                          = /var/run/mysqld/mysqld.sock
bind_address                    = 127.0.0.1
port                            = 3306
basedir                         = /usr
datadir                         = /var/lib/mysql
tmpdir                          = /tmp
event_scheduler                 = 0
skip_external_locking
skip_name_resolve
#skip_myisam

# charset and collation
character_set_server            = utf8
collation_server                = utf8_general_ci

# os related
open_files_limit                = 8192                          # should be around 3x table_c

# networking
max_connections                 = 250
max_allowed_packet              = 128M
connect_timeout                 = 10
wait_timeout                    = 180                           # could cause problems if too
net_read_timeout                = 30
net_write_timeout               = 60
back_log                        = 50

# fine tuning
key_buffer_size                 = 16M
thread_stack                    = 256K
thread_cache_size               = 8
sort_buffer_size                = 2M
read_buffer_size                = 128K
read_rnd_buffer_size            = 256K
join_buffer_size                = 128K
query_cache_limit               = 1M
query_cache_size                = 16M
table_open_cache                = 256
tmp_table_size                  = 32M
max_heap_table_size             = 32M
bulk_insert_buffer_size         = 16M
auto_increment_increment        = 1
auto_increment_offset           = 1
concurrent_insert               = 2

# MyISAM options
myisam_recover_options          = FORCE,BACKUP
myisam_sort_buffer_size         = 8M

# logging
#general_log_file               = /var/log/mysql/mysql.log
#general_log                    = 1
log_error                       = /var/log/mysql/error.log
slow_query_log                  = 1
slow_query_log_file             = /var/log/mysql/mysql-slow.log
long_query_time                 = 2
log_queries_not_using_indexes  = 1

# replication settings
server_id                       = 1                             # set to 2 or higher in repli
log_bin                         = mysql-bin
#binlog_format                  = statement
binlog_format                   = mixed
log_slave_updates               = false
expire_logs_days                = 3
max_binlog_size                 = 100M
binlog_cache_size               = 32K
sync_binlog                     = 1
```

```
#relay_log                      = mysqld-relay-bin            # enable on replication slave
#read_only                      = true                        # only on replication slaves

# InnoDB options
innodb_data_home_dir            = /var/lib/mysql
innodb_log_group_home_dir       = /var/lib/mysql
innodb_table_locks              = true
innodb_lock_wait_timeout        = 60
innodb_thread_concurrency       = 9
innodb_commit_concurrency       = 0
innodb_support_xa               = true
innodb_buffer_pool_size         = 2G
innodb_buffer_pool_instances    = 2
innodb_log_file_size            = 30M
innodb_additional_mem_pool_size = 128M
innodb_data_file_path           = ibdata1:10M:autoextend
innodb_flush_log_at_trx_commit  = 1
innodb_flush_method             = O_DIRECT
innodb_log_buffer_size          = 8M
innodb_open_files               = 8192
innodb_file_per_table

[mysqldump]
quick
quote_names
max_allowed_packet              = 128M

[mysql]

[isamchk]
key_buffer_size                 = 16M
myisam_sort_buffer_size         = 16M
myisam_max_sort_file_size       = 2147483648
myisam_repair_threads           = 1
myisam_recover_options          = FORCE,BACKUP

# include *.cnf files, overwriting settings from here
!includedir /etc/mysql/conf.d/
```

kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.6)

kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.6)

> Diese Anleitung wird nicht weiter aktualisiert. Bitte wechselt auf die Versionen 5.7 oder 8.0

Dieses Beispiel ist für kleine bis mittelgroße Setups geeignet. Die Vorlage kann einfach nach /etc/mysql/my.cnf kopiert werden. Sofern die Datenbankengine MyISAM nicht benötigt wird, kann diese über „skip-myisam“ im [mysqld]-Bereich entfernt werden.

MariaDB 10.0 entspricht etwa MySQL 5.5 mit Backports von MySQL 5.6.

Um die neuen Einstellungen zu aktiveren, müssen sehr wahrscheinlich (bei Percona Server 5.6 und 5.7 ist es nicht notwendig!) einmal die InnoDB-Logfiles gelöscht werden (MySQL läßt sich sonst nicht starten). Diese werden dann beim nächsten Start der Datenbank neu angelegt. Je nach Größe kann das auch länger dauern, das Init-Script beendet sich dann gerne mit einem Fehler. Den aktuellen Status sieht man im MySQL-Logfile (bzw. /var/log/syslog). Die Logfiles werden wie folgt gelöscht:

```
service mysql stop
rm -f /var/lib/mysql/ib_logfile*
service mysql start
```

> **Sämtliche Einstellungen sind von Dir dann natürlich nochmals zu überprüfen**, insbesondere Caches und Speicherverbrauch. Dabei kann dann auch das Tool mysqltuner.pl helfen. Mein Beispiel ist auf ein Hot-Dataset von 2GB InnoDB Daten ausgelegt.

Bei **Debian Wheezy**: Anpassung der Systemlimits, diese Zeilen am Ende vor der letzten Zeile ergänzen:

/etc/security/limits.conf

```
*    soft    nofile    65535
*    hard    nofile    65535
```

Ab **Debian Jessie**, bzw. Distributionen mit Systemd, muss das Systemd unit file angepasst werden (danach „systemctl daemon-reload“ ausführen). Sollte es dieses File nicht geben (z.B. bei Debian Jessie mit MariaDB 10) siehe nächster Absatz.

/lib/systemd/system/mysql.service

```
LimitNOFILE=infinity
LimitMEMLOCK=infinity
```

MariaDB / Percona in Debian Jessie hat kein eigenes Service file. Folgendes funktioniert allerdings:

```
# mkdir /etc/systemd/system/mysql.service.d
# vi /etc/systemd/system/mysql.service.d/limits.conf
  [Service]
  LimitNOFILE=infinity
  LimitMEMLOCK=infinity
# systemctl daemon-reload
# systemctl status mysql.service

es sollten folgende Zeilen auftauchen:
  Drop-In: /etc/systemd/system/mysql.service.d
           ??limits.conf
```

Da sich die Pfade zu den Logfiles in meiner Config vom Standard unterscheiden, sollte noch der Inhalt von /etc/logrotate.d/mysql ausgetauscht werden:

/etc/logrotate.d/mysql

```
/var/log/mysql/*.log {
        daily
        rotate 7
        missingok
        create 640 mysql adm
        compress
        sharedscripts
        postrotate
          test -x /usr/bin/mysqladmin || exit 0
          if [ -f `my_print_defaults --mysqld | grep -oP "pid-file=\K[^$]+"` ]; then
            # If this fails, check debian.conf!
            mysqladmin --defaults-file=/etc/mysql/debian.cnf flush-logs
          fi
        endscript
}
```

/etc/mysql/my.cnf

```
#
# The MySQL database server configuration file
#
# compact version from https://wiki.magenbrot.net/datenbanken/mysql/kompakte_mysql_my.cnf_v
#
# 2016 Oliver Völker <info@ovtec.it>
#

[client]
port                            = 3306
socket                          = /var/run/mysqld/mysqld.sock
default_character_set           = utf8

[mysqld_safe]
socket                          = /var/run/mysqld/mysqld.sock
nice                            = 0

[mysqld]
# basic settings
```

```
user                            = mysql
pid_file                        = /var/run/mysqld/mysqld.pid
socket                          = /var/run/mysqld/mysqld.sock
bind_address                    = 127.0.0.1
port                            = 3306
basedir                         = /usr
datadir                         = /var/lib/mysql
tmpdir                          = /tmp
event_scheduler                 = 0
explicit_defaults_for_timestamp = true
skip_external_locking
skip_name_resolve
#skip_myisam

# charset and collation
character_set_server            = utf8
collation_server                = utf8_general_ci

# os related
open_files_limit                = 8192                          # should be around 3x table_

# networking
max_connections                 = 250
max_connect_errors              = 1000000
max_allowed_packet              = 128M
connect_timeout                 = 10
wait_timeout                    = 180                           # could cause problems if too
net_read_timeout                = 30
net_write_timeout               = 60
interactive_timeout             = 600
slave_net_timeout               = 60
back_log                        = 50

# fine tuning
key_buffer_size                 = 16M
thread_stack                    = 256K
thread_cache_size               = 8
thread_pool_size                = 16
sort_buffer_size                = 1M
read_buffer_size                = 128k
read_rnd_buffer_size            = 256k
join_buffer_size                = 128k
query_cache_type                = 0
query_cache_limit               = 0
query_cache_size                = 0
table_open_cache                = 256
tmp_table_size                  = 32M
max_heap_table_size             = 32M
bulk_insert_buffer_size         = 16M
auto_increment_increment        = 1
auto_increment_offset           = 1
concurrent_insert               = 2
ft_min_word_len                 = 3

# MyISAM options
myisam_recover_options          = FORCE,BACKUP
myisam_sort_buffer_size         = 8M

# logging
general_log_file                = /var/log/mysql/mysql.log
general_log                     = 0
log_error                       = /var/log/mysql/error.log
slow_query_log                  = 1
slow_query_log_file             = /var/log/mysql/slow.log
long_query_time                 = 2
log_queries_not_using_indexes   = 1
log_slow_admin_statements       = 1

# replication settings
server_id                       = 1                             # set to 2 or higher in repli
log_bin                         = mysql-bin
#binlog_format                  = statement
binlog_format                   = mixed
log_slave_updates               = false
expire_logs_days                = 3
max_binlog_size                 = 1G
binlog_cache_size               = 32K
sync_binlog                     = 1
```

```
#relay_log                      = mysqld-relay-bin            # enable on replication slave
#read_only                      = true                        # only on replication slaves

# InnoDB options
innodb_data_home_dir            = /var/lib/mysql
innodb_log_group_home_dir       = /var/lib/mysql
innodb_table_locks              = true
innodb_lock_wait_timeout        = 50
innodb_thread_concurrency       = 9
innodb_commit_concurrency       = 0
innodb_support_xa               = true
innodb_buffer_pool_size         = 2G
innodb_buffer_pool_instances    = 2
innodb_log_file_size            = 512M
innodb_data_file_path           = ibdata1:10M:autoextend
innodb_flush_log_at_trx_commit  = 1
innodb_flush_method             = O_DIRECT
innodb_log_buffer_size          = 8M
innodb_open_files               = 8192
innodb_file_per_table           = on

[mysqldump]
quick
quote_names
max_allowed_packet              = 128M

[mysql]

[isamchk]
key_buffer_size                 = 16M
myisam_sort_buffer_size         = 16M
myisam_max_sort_file_size       = 2147483648
myisam_repair_threads           = 1
myisam_recover_options          = FORCE,BACKUP

# include *.cnf files, overwriting settings from here
!includedir /etc/mysql/conf.d/
```

kompakte MySQL my.cnf Vorlage für neue Server (MySQL-/MariaDB-/Percona-Server 5.7)

Dieses Beispiel ist für kleine bis mittelgroße Setups geeignet. Die Vorlage kann einfach nach /etc/mysql/my.cnf kopiert werden. Sofern die Datenbankengine MyISAM nicht benötigt wird, kann diese über „skip-myisam“ im [mysqld]-Bereich entfernt werden.

MariaDB 10.1 entspricht etwa MySQL 5.6 mit Backports von MySQL 5.7.

Um die neuen Einstellungen zu aktiveren, müssen sehr wahrscheinlich (bei Percona Server 5.6 und 5.7 ist es nicht notwendig!) einmal die InnoDB-Logfiles gelöscht werden (MySQL läßt sich sonst nicht starten). Diese werden dann beim nächsten Start der Datenbank neu angelegt. Je nach Größe kann das auch länger dauern, das Init-Script beendet sich dann gerne mit einem Fehler. Den aktuellen Status sieht man im MySQL-Logfile (bzw. /var/log/syslog). Die Logfiles werden wie folgt gelöscht:

```
service mysql stop
rm -f /var/lib/mysql/ib_logfile*
service mysql start
```

> **Sämtliche Einstellungen sind von Dir dann natürlich nochmals zu überprüfen**, insbesondere Caches und Speicherverbrauch. Dabei kann dann auch das Tool mysqltuner.pl helfen. Mein Beispiel ist auf ein Hot-Dataset von 2GB InnoDB Daten ausgelegt.

Bei **Debian Wheezy**: Anpassung der Systemlimits, diese Zeilen am Ende vor der letzten Zeile ergänzen:

/etc/security/limits.conf

```
*    soft    nofile    65535
*    hard    nofile    65535
```

Zum anpassen der Limits in Systemd ist ein Drop-In Unitfile anzulegen (das mitgelieferte Unitfile muss dafür nicht angepasst werden):

```
# mkdir /etc/systemd/system/mysql.service.d && vi
/etc/systemd/system/mysql.service.d/limits.conf
# bzw. für MariaDB:
# mkdir /etc/systemd/system/mariadb.service.d && vi
/etc/systemd/system/mariadb.service.d/limits.conf

[Service]
LimitNOFILE=infinity
```

```
LimitMEMLOCK=infinity


# systemctl daemon-reload

# systemctl status mysql.service


es sollten folgende Zeilen auftauchen:

  Drop-In: /etc/systemd/system/mysql.service.d

           └─limits.conf
```

Da sich die Pfade zu den Logfiles in meiner Config vom Standard unterscheiden, sollte noch der Inhalt von /etc/logrotate.d/mysql ausgetauscht werden:

/etc/logrotate.d/mysql

```
/var/log/mysql/*.log {
 daily
 rotate 7
 missingok
 create 640 mysql adm
 compress
 sharedscripts
 postrotate
  test -x /usr/bin/mysqladmin || exit 0
  # If this fails, check debian.conf!
  MYADMIN="/usr/bin/mysqladmin --defaults-file=/etc/mysql/debian.cnf"
  if [ -z "`$MYADMIN ping 2>/dev/null`" ]; then
    # Really no mysqld or rather a missing debian-sys-maint user?
    # If this occurs and is not a error please report a bug.
    #if ps cax | grep -q mysqld; then
    if killall -q -s0 -umysql mysqld; then
       exit 1
    fi
  else
    $MYADMIN flush-logs
  fi
 endscript
}
```

/etc/mysql/my.cnf

```
#
# The MySQL database server configuration file
#
# compact version from https://wiki.magenbrot.net/datenbanken/mysql/kompakte_mysql_my.cnf_v
#
# 2016 Oliver Völker <info@ovtec.it>
#

[client]
port                           = 3306
socket                         = /var/run/mysqld/mysqld.sock
default_character_set          = utf8

[mysqld_safe]
socket                         = /var/run/mysqld/mysqld.sock
nice                           = 0

[mysqld]
# basic settings
user                           = mysql
pid_file                       = /var/run/mysqld/mysqld.pid
socket                         = /var/run/mysqld/mysqld.sock
bind_address                   = 127.0.0.1
port                           = 3306
basedir                        = /usr
datadir                        = /var/lib/mysql
```

```
tmpdir                          = /tmp
event_scheduler                 = 0
explicit_defaults_for_timestamp = true
skip_external_locking
skip_name_resolve
#skip_myisam

# charset and collation
character_set_server            = utf8
collation_server                = utf8_general_ci

# os related
open_files_limit                = 8192                      # should be around 3x table_

# networking
max_connections                 = 250
max_connect_errors              = 1000000
max_allowed_packet              = 128M
connect_timeout                 = 10
wait_timeout                    = 180                       # could cause problems if too
net_read_timeout                = 30
net_write_timeout               = 60
interactive_timeout             = 600
slave_net_timeout               = 60
back_log                        = 50

# fine tuning
key_buffer_size                 = 16M
thread_stack                    = 256K
thread_cache_size               = 8
thread_pool_size                = 16
sort_buffer_size                = 1M
read_buffer_size                = 128k
read_rnd_buffer_size            = 256k
join_buffer_size                = 128k
query_cache_type                = 0
query_cache_limit               = 0
query_cache_size                = 0
table_open_cache                = 256
tmp_table_size                  = 32M
max_heap_table_size             = 32M
bulk_insert_buffer_size         = 16M
auto_increment_increment        = 1
auto_increment_offset           = 1
concurrent_insert               = 2
ft_min_word_len                 = 3

# MyISAM options
myisam_recover_options          = FORCE,BACKUP
myisam_sort_buffer_size         = 8M

# logging
general_log_file                = /var/log/mysql/mysql.log
general_log                     = 0
log_error                       = /var/log/mysql/error.log
slow_query_log                  = 1
slow_query_log_file             = /var/log/mysql/slow.log
long_query_time                 = 2
log_queries_not_using_indexes   = 1
log_slow_admin_statements       = 1

# replication settings
server_id                       = 1                         # set to 2 or higher in repli
log_bin                         = mysql-bin
#binlog_format                  = statement
binlog_format                   = mixed
log_slave_updates               = false
expire_logs_days                = 3
max_binlog_size                 = 1G
binlog_cache_size               = 32K
sync_binlog                     = 1
#relay_log                      = mysqld-relay-bin          # enable on replication slave
#read_only                      = true                      # only on replication slaves

# InnoDB options
innodb_data_home_dir            = /var/lib/mysql/
innodb_log_group_home_dir       = /var/lib/mysql
innodb_table_locks              = true
```

```
innodb_lock_wait_timeout        = 50
innodb_thread_concurrency       = 9
innodb_commit_concurrency       = 0
innodb_support_xa               = true
innodb_buffer_pool_size         = 2G
innodb_buffer_pool_instances    = 2
innodb_log_file_size            = 50M
innodb_data_file_path           = ibdata1:10M:autoextend
innodb_flush_log_at_trx_commit  = 1
innodb_flush_method             = O_DIRECT
innodb_log_buffer_size          = 8M
innodb_open_files               = 8192
innodb_file_per_table           = true
innodb_large_prefix             = true
innodb_file_format              = barracuda

[mysqldump]
quick
quote_names
max_allowed_packet              = 128M

[mysql]

[isamchk]
key_buffer_size                 = 16M
myisam_sort_buffer_size         = 16M
myisam_max_sort_file_size       = 2147483648
myisam_repair_threads           = 1
myisam_recover_options          = FORCE,BACKUP

# include *.cnf files, overwriting settings from here
!includedir /etc/mysql/conf.d/
```

MySQL mit anderem Datenverzeichnis auf speziellem Port starten

Um MySQL z.B. aus einem xtrabackup-Restore parallel zu einer laufenden Datenbank auf einem anderen Port zu starten kann dieses Kommando verwendet werden:

```
mysqld --socket=/run/mysqld/mysqld3311.sock --datadir=/restore/var/lib/mysql --basedir=/usr --user=mysql --port=3311 --pid-file=/run/mysqld/mysqld3311.pid
```

Verbinden mit der Instanz auf Port 3311:

```
mysql -h 127.0.0.1 -P3311 -uroot -pPasswort
```

Oder mit mysqldump eine bestimmte Datenbank/Tabelle extrahieren:

```
# verbinden entweder über den Port 3311 (-P) oder den Socket in /run/mysqld/mysqld3311.sock (-S)

# einmal alles sichern
mysqldump -h 127.0.0.1 -P3311 -A > /root/all_database.sql

# eine bestimmte Datenbank
mysqldump -h 127.0.0.1 -P3311 database > /root/single_database.sql

# bestimmte Tabelle(n) aus einer bestimmten Datenbank
mysqldump -h 127.0.0.1 -P3311 database tabelle1 tabelle2 [...] > /root/tables_from_database.sql
```