MySQL Konfiguration

Storage Engine

Theorie

Storage Engine bei einer neuen Tabelle wählen

Wenn man eine spezielle Storage Engine für eine Tabelle wählen möchte, kann man dies so machen

```
CREATE TABLE csvTest (csvID INT Primary Key, name) ENGINE = CSV;
CREATE TABLE csvMemory (i INT) ENGINE = MEMORY;
```

Benutzer konfigurieren

Alle user können so abgerufen werden.

```
USE mysql;
select * from user;

-- Schönere Ausgabe
select host,user,authentication_string from user;

-- Beschreibung der Attribute
DESCRIBE user;
```

!> Damit Berechtigungen übernommen werden, ist ein Befehl sehr wichtig!
`FLUSH PRIVILEGES;`

Root

1. Passwort für den Root user setzen

   ```
   ALTER USER 'root'@'localhost' IDENTIFIED BY 'root1234';
   FLUSH PRIVILEGES;
   ```

2. Überprüfen, von wo aus sich Root überall anmelden darf --> sollte nur vom localhost möglich sein!

   ```
   SELECT User, Host, plugin FROM mysql.user WHERE User = 'root';
   ```

```
mysql> SELECT User, Host, plugin FROM mysql.user WHERE User = 'root';
+------+-----------+-------------+
| User | Host      | plugin      |
+------+-----------+-------------+
| root | localhost | auth_socket |
+------+-----------+-------------+
1 row in set (0.00 sec)
```

Falls nicht, folgenden Command ausführen:

```
UPDATE mysql.user SET Host = 'localhost' WHERE User = 'root';
```

Benutzer konfigurieren

1. User erstellen

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'userPassword1234!';
```

2. User auf die gewünschten Tabellen Zugriff geben
   Hier gebe ich dem User nur SELECT & INSERT, weil er ein Anewdungsbenutzer ist, auf die Datenbank `demo`

```
GRANT SELECT,INSERT ON demo.* TO 'user'@'localhost';
```

Hier habe ich noch ein Admin Benutzer erstellt

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'adminPassword1234!';
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON demo.* TO
'admin'@'localhost';
```

```
mysql> select host,user,authentication_string from user;
+-----------+------------------+-------------------------------------------------------------------------+
| host      | user             | authentication_string                                                   |
+-----------+------------------+-------------------------------------------------------------------------+
| localhost | admin            | $A$005$1◆F)[XK9^HNu
/knT6vZ/Oh.3zZEFt0bEy7pM3MkT/22XDyg0tpg7FhmE5 |
| localhost | debian-sys-maint | $A$005$y       _A}%PB%RQyrHZ83hbs6e6.Q3AutaOPObtYUCgufNy3hBj67707FhG38 |
| localhost | mysql.infoschema | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.session    | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | mysql.sys        | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED |
| localhost | root             |                                                                         |
| localhost | user             | $A$005$3KKzBy-jl+(-r
                                                ddzvXNB2rQm7sUmbr9q/MhGiHWAMh4H.D3nwfnYAD81 |
+-----------+------------------+-------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```

Sever Konfiguration

Transaktions-Isolation

Um die Transaktions-Isolations-Stufe abzurufen, kann man folgenden Befehl verwenden.

```
SELECT @@transaction_isolation;
```

```
mysql> SELECT @@transaction_isolation;
+-------------------------+
| @@transaction_isolation |
+-------------------------+
| REPEATABLE-READ         |
+-------------------------+
1 row in set (0.00 sec)
```

Anomalie	Erklärung
Dirty Reads	Eine Transaktion kann keine Änderungen sehen, die von einer anderen Transaktion noch nicht bestätigt wurden.
Repeatable Reads	Wenn eine Transaktion eine Zeile zweimal liest, bleibt der Wert gleich, selbst wenn eine andere Transaktion ihn zwischenzeitlich ändert.
Phantom Reads	Neue Datensätze, die in einer anderen Transaktion eingefügt werden, könnten sichtbar werden, wenn eine erneute Abfrage durchgeführt wird.

System Variablen

Mit folgendem Befehl, können alle System Variablen ausgegeben werden.

```
show variables\G;
```

Netzwerkkonfiguration

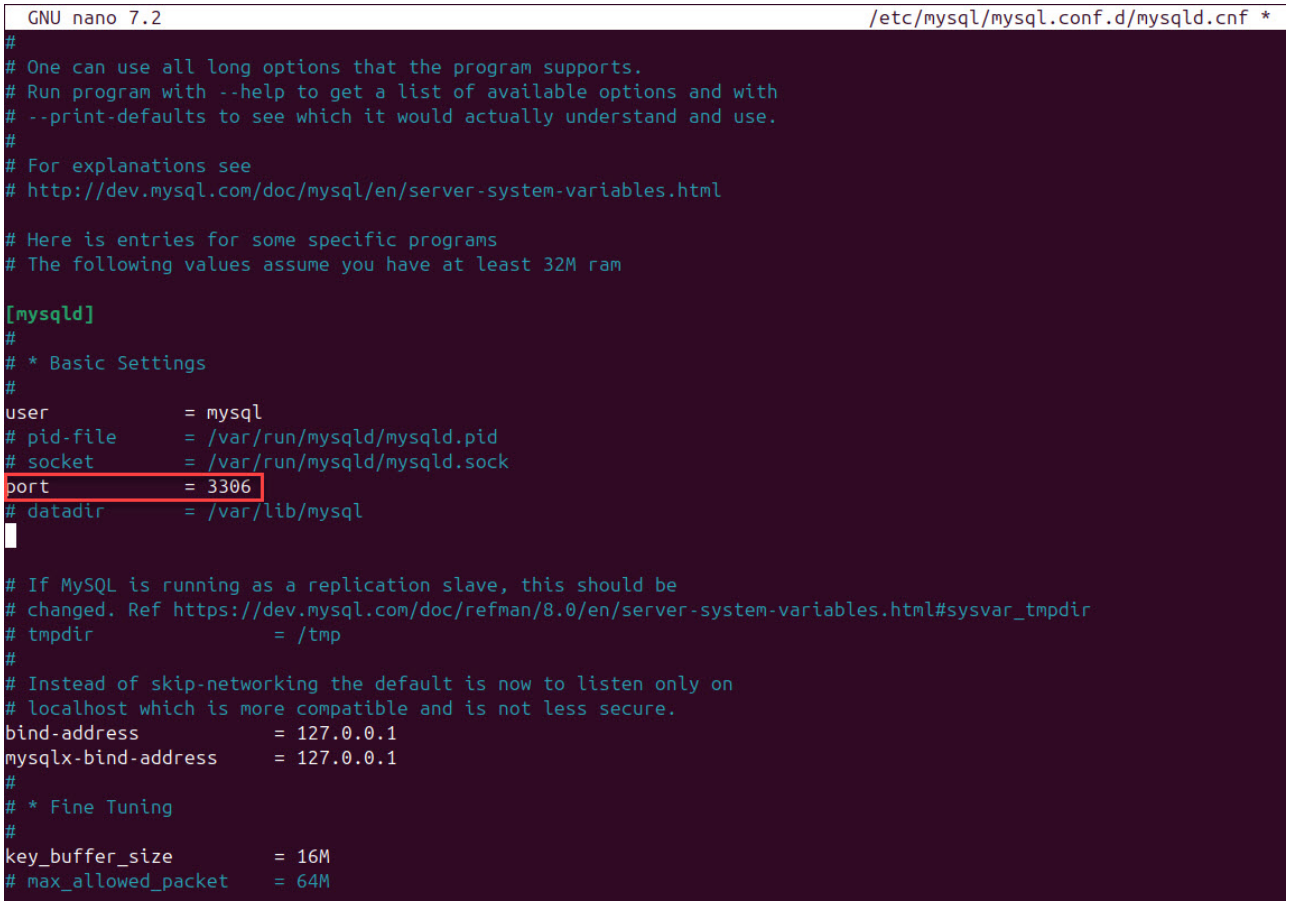
1. Konfigurationsdatei öffnen

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf;
```

TCP Port öffnen

1. Port Zeile einkommentieren --> Der Standardport ist 3306

```
port            = 3306
```



2. SQL server neu starten

```
sudo systemctl restart mysql
```

Server Betrieb

Protokollierung langsamer Abfragen aktivieren

1. Konfigurationsdatei öffnen

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf;
```

2. Folgende 4 Zeilen auskommentieren

```
[mysqld]
slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow.log
long_query_time = 2
log_queries_not_using_indexes = 1
```

```
  GNU nano 7.2                                    /etc/mysql/mysql.conf.d/mysqld.cnf *

# thread_cache_size       = -1

# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover-options  = BACKUP

# max_connections        = 151

# table_open_cache       = 4000

#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
#
# Log all queries
# Be aware that this log type is a performance killer.
# general_log_file        = /var/log/mysql/query.log
# general_log             = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
slow_query_log          = 1
slow_query_log_file     = /var/log/mysql/mysql-slow.log
long_query_time = 2
log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
# server-id             = 1
```

3. SQL server neu starten

```
sudo systemctl restart mysql
```

Inhalt des Data-Directories auflisten

Im Data Directory, werden alle Datenbankdaten gespeichert!

1. Nachschauen, wo sich das Data Directory befindet

```
SHOW VARIABLES LIKE 'datadir';
```

2. Pfad öffnen

```
sudo ls /var/lib/mysql/
```

```
vagrant@m141vm:~$ sudo ls /var/lib/mysql/
auto.cnf         binlog.000004   binlog.000008   binlog.index      client-key.pem      '#ib_16384_1.dblwr'  '#innodb_redo'   mysql.ibd            server-cert.pem   undo_002
binlog.000001    binlog.000005   binlog.000009   ca-key.pem        debian-5.7.flag     ib_buffer_pool       '#innodb_temp'   performance_schema   server-key.pem
binlog.000002    binlog.000006   binlog.000010   ca.pem            demodb              ibdata1               m141vm.pid      private_key.pem      sys
binlog.000003    binlog.000007   binlog.000011   client-cert.pem  '#ib_16384_0.dblwr'  ibtmp1                mysql           public_key.pem       undo_001
```

Default Datenbanken

In MySQL gibt es mehrere Default Datenbanken, welche bei der Installation von MySQL automatisch installiert werden.

MySQL

```
use mysql;
```

Hier gibt es folgende drei wichtige Tabellen:

- user

  ```
  Select * from user\G;
  ```

  Hier werden alle Benutzerkontoinformationen gespeichert.
- db

  ```
  Select * from db\G;
  ```

  Hier wird für jede Datenbank beschrieben, welcher User, welche Rechte auf jeder Datenbank hat.
- tables_priv

  ```
  Select * from tables_priv\G;
  ```

  Hier wird für jede Tabelle von der Datenbank beschrieben, welcher User, welche Rechte auf jede Tabelle hat.

SYS

In dieser Datenbank, findet man viele Infos für die Überwachung der Performance vom MySQL Server.

```
use sys;
```

- host_summary

  ```
  Select * from host_summary\G;
  ```

  Hier sieht man alle Verbindungen, welche aktuell mit dem MySQL Server bestehen und wie viele Ressourcen diese verbrauchen.

performance_schema

Detaillierte Information zur aktuellen Leistung von MySQL

```
use performance_schema;
```

- events_statements_summary_by_digest

  ```
  Select * from events_statements_summary_by_digest\G;
  ```

  Zeigt die häufigsten Abfragen an, welche gemacht wurden
- threads

  ```
  Select * from threads\G;
  ```

  Zeigt alle aktuellen Threads an.