MySQL

Nützliche Codeschnipsel für die Verwaltung von MySQL Servern

Benutzer mit Adminrechten anlegen

```
## only local
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
## remote connection - not secure
CREATE USER 'admin'@'%' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Datenbanken ex- und importieren

Datenbanken exportieren (dumpen)

alle Datenbanken dumpen

```
mysqldump -uUSER -p --all-databases > my-mysql-dump.sql
```

eine bestimmte Datenbank dumpen

```
mysqldump -uUSER -p mydatabase1 > my-mysql-dump.sql
```

mehrere Datenbanken dumpen

```
mysqldump -uUSER -p --databases db_name1 db_name2 db_name_n > my-mysql-dump.sql
```

nur eine bestimmte Tabelle aus einer Datenbank dumpen

```
mysqldump -uUser -p mydatabase1 table_name > my-mysql-dump.sql
```

Größe eines Datenbankdumps reduzieren (bspw. für schnelleren Transfer zwischen zwei Servern)

```
mysqldump -uUser -p mydatabase1 table_name | gzip -c > my-mysql-dump.sql.gz
```

Dies kann man sogar automatisieren als Cronjob um eine Datenbank regelmäßig zu sichern

```
crontab -e
```

Dann ans Ende der Datei folgenden Code einfügen

```
0 */6 * * * mysqldump -u 'User' -p 'Password' mydatabase1 table_name | gzip -c > my-mysql-dump.s
```

Dies führt eine Datenbanksicherung alle 6 Stunden aus

Datenbankdump zurückspielen (importieren)

```
mysql -uUSER -p mydatabase1 < my-mysql-dump.sql
```

Hierbei muss die Datenbank unter dem angegebenem Datenbanknamen bereits existieren (in diesem Beispiel mydatabase1)

MySQL Tipps & Tricks

MySQL Backup per Konsole einspielen

```
pv export_db_name.sql |  mysql -uroot -p db_name
```

> **Hinweis:** Benötigt pv `(apt install pv)`

Neuen Benutzer erstellen und Rechte gewähren

```
CREATE USER 'new_user'@'localhost'  IDENTIFIED BY 'new_password';
GRANT ALL ON my_db.* TO 'new_user'@'localhost';
```

Prozessliste laufend aktualisiert anzeigen

```
mysqladmin -u root -p -i 1 processlist
```

Passwort neu setzen (für Useraccounts)

für MySQL

```
ALTER USER user IDENTIFIED BY 'auth_string';
```

für MariaDB

```
SET PASSWORD FOR 'username'@'localhost'  = PASSWORD('newpass');
```

Externen Zugriff zulassen

> In dieser Anleitung spreche ich immer von einem MySQL Server, gemeint ist damit natürlich auch der MariaDB Server. Sollten einzelne Schritte zwischen dem MySQL Server und dem MariaDB Server abweichend sein behandle ich diese seprarat

Um den externen Zugriff auf eine Datenbank bzw. einen MySQL Server freizuschalten bedarf es einer Änderung der Konfigurationsdatei des MySQL Servers.

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

Bei Installationen des MariaDB Servers findet man die Konfigurationsdatei abweichend hier

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Dort findet man in beiden Fällen eine umfangreiche Konfigurationsdatei vor, uns interessiert allerdings nur eine bestimmte Zeile

```
. . .
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1
. . .
```

Hier ändern wir den Inhalt folgendermaßen um

```
. . .
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
```

```
. . .
```

Datei speichern & schließen und anschließend den MySQL Server neu starten

```
sudo systemctl restart mysql
```

```
FLUSH PRIVILEGES;
```

Nun könnten theoretisch bereits alle Datenbanken extern erreicht werden, allerdings muss dieses Recht pro User und pro Datenbank explizit noch gesetzt werden. Hierzu loggen wir uns lokal auf dem MySQL Server ein und bearbeiten bzw. erstellen uns einen User mit den passenden Zugriffsrechten

```
mysql -u root -p

### Existierenden User für externen Zugriff von einer einzigen IP freischalten (für bspw. den Zugriff von einem Server zum nächsten bei gleichbleibender IP)
RENAME USER 'username'@'localhost' TO 'username'@'remote_ip';

### Wenn man jedoch von zuhause auf den Server zugreifen möchte eignet sich diese Methode nicht da i.d.R. die IP des heimischen Anschlusses regelmäßig geändert wird.
### Hierfür erlauben wir also den Zugriff von JEDER IP - Hinweis: Zu dem allgemein erhöhten Angriffsrisiko durch den externen Zugriff steigern wir die Gefahr erneut durch den Zugriff von JEDER IP aus.
### Grundsätzlich gilt daher: Immer ausreichend lange und komplizierte Passwörter verwenden um es pozentiellen Angreifern nicht allzu leicht zu machen
RENAME USER 'username'@'localhost' TO 'username'@'%';

### Neuen User erstellen für den externen Zugriff
CREATE USER 'username'@'remote_ip' IDENTIFIED BY 'password';
### ODER
CREATE USER 'username'@'%' IDENTIFIED BY 'password';
### Der neue Benutzer hat dann allerdings noch keine Zugriffsrechte um Aktionen an Datenbanken oder Tabellen auszuführen, dies muss separat erfolgen
GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD on *.* TO 'username'@'remote_ip' WITH GRANT OPTION;
### Hinweis: Nur die Rechte vergeben die auch wirklich benötigt werden
```

Zum Abschluss laden wir noch die Berechtigungen neu und verlassen den MySQL Server

```
FLUSH PRIVILEGES;
exit
```

Der Zugriff von Außerhalb sollte nun funktionieren

Abschließender Hinweis:

Sollte auf dem Server eine Firewall konfiguriert sein muss auch hier der Zugriff freigegeben werden, hier am Beispiel von `ufw`

```
sudo ufw allow from remote_ip to any port 3306
## Alternativ kann auch hier wieder der Zugriff von jeder IP erlaubt werden, jedoch wird auch hiervon abgeraten
sudo ufw allow 3306
```