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

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"`).

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
```

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
```

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.

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
```