Replikation

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

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;
```

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

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.