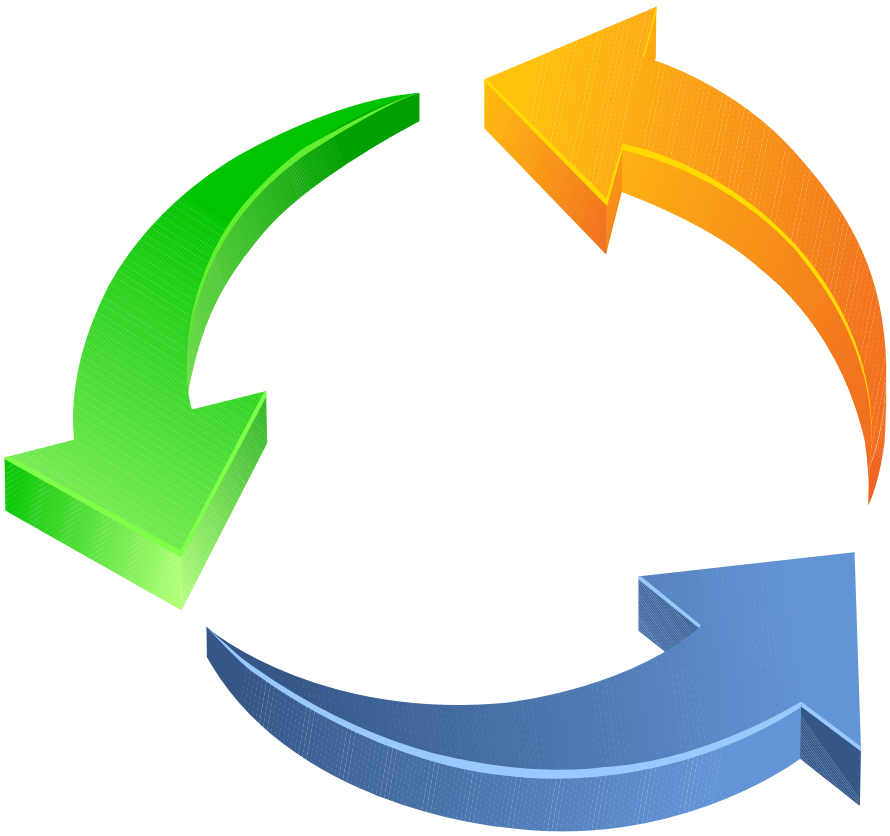
REPLIKATION MIT MYSQL 5.7

EINE SCHNELLANLEITUNG VON JÖRG REINHOLZ



INHALTSVERZEICHNIS

IMPRESSUM/RECHTLICHES

AUTOR UND INHALTLICH VERANTWORTLICHER

Jörg Reinholz
Hafenstraße 67
34125 Kassel
joerg.reinholz@gmail.com

http://www.fastix.org

FREMDE NAMEN UND MARKEN

- **“MySQL”** wird gegenwärtig von der Oracle weiterentwickelt und steht unter der GPL Version 2.
- **“MariaDB”** ist ein hochentwickelter Fork des ursprünglich hinter MySQL stehenden Hauptentwicklers Ulf Michael Widenius und der dahinter stehenden Community.
- “**Linux”** ist ein eingetragener Markenname von Linus Thorwald. Dieser entwickelt mit einer Community den Linux-Kernel, also die Basis dessen, was allgemein “Linux” genannt wird.
- “**Debian”** (vollständig: “Debian Gnu/Linux”) ist ein gemeinschaftlich entwickeltes freies Betriebssystem.
- **“Ubuntu”** ist eine kostenlose Linux-Distribution, die auf “Debian Gnu/Linux” basiert und von der Canonical Foundation, sowie der Ubuntu Community entwickelt und gepflegt wird.

FÜR DIESES HANDBUCH

GELTUNGSBEREICH DES HANDBUCHS

Dieses Handbuch gilt explizit für MySQL 5.7 unter Debian “Jessie” (Debian 8.9). Prinzipiell sind die dargestellten Vorgehensweisen und Schritte auch für ältere und neuere Versionen von MySQL – einschließlich MariaDB gültig. Es können aber Abweichungen auftreten.

Der Einsatz (und die Verwendung der Informationen aus diesem Handbuch) ist im Prinzip auch für MySQL bzw, MariaDB unter Windows gültig, wird dann aber teilweise (Installation, Ort der Konfigurationsdateien, kein ssh, kein rsync) erheblich abweichen.

DARSTELLUNGSVEREINBARUNG

Eingaben in den bereits in Betrieb befindlichen Server “s1.box” werden `blau umrahmt`.

Eingaben in den neuen Server “s2.box” werden auf den künftigen Seiten `rot umrahmt.`

ZIELE DER REPLIKATION

BACKUP

Hierfür wird regelmäßig eine Master-Client-Replikation von einem Server auf einen anderen MySQL-Server durchgeführt. Dieses Backup beinhaltet dann stets den aktuellen Datenbestand und der Backup-Server kann jederzeit gestoppt oder angehalten werden um z.B. den Datenbestand wie er zu einem bestimmten Zeitpunkt ist (war) entweder durch einen Dump (Export) oder auf Basis des Dateisystems zu sichern.

Dieser Server sollte für sonstige Clients im Regelbetrieb nicht erreichbar sein, Falls doch ist darauf zu achten, dass Programme und Anwendungen nicht auf den Backup-Server zugreifen, insbesondere aber niemals Daten auf diesem verändern. In Umgebungen mit paranaoiden ökonomischen Einschränkungen kann zwar ein solcher Server auch für nur lesende Zugriffe verwendet werden – aber im Zweifelsfall wird dieser Backup-Server dann im Regelbetrieb hoch belastet und fällt dann kurz nach dem Ausfall des eigentlichen Servers auch aus, was dann mit Sicherheit zu einer teuren Down-Time führt.

Im Falle eines Ausfalls des replizierten Servers kann dieser entweder sehr schnell wieder hergestelllt werden oder durch Anpassung der IP-Adresse bzw. des DNS sogar als aktiver Server in Betrieb genommen werden, was Downtimes extrem verkürzt.

Bitte beachten Sie, dass bei einer Anpassung der IP-Adresse, des DNS oder des Switchings “Nebenwirkungen” eintreten. Das bestrifft insbesondere Schlüssel und Serverzertifikate für gesicherte Verbindungen. Hierfür sind Vorkehrungen zu treffen. Zudem wirkt bei einer Anpassung des DNS auf vielen Clients ein DNS-Cache entweder des Betriebssystems, des Clients oder der Clientanwendung nach.

LASTVERTEILUNG

Hierbei wird prinzipiell zwischen zwei Szenarios unterschieden:

SYSTEME MIT HOHER ABFRAGELAST

In einem solchen Szenario fragt die weit überwiegende Zahl der Clients einen Datenbestand lediglich ab. Hierfür wird regelmäßig eine Master-Client-Replikation von einem Server auf einen oder mehrere andere MySQL-Server (“Slaves”, “Replikanten”) durchgeführt, welche dann den Clients nur für lesende Abfragen zur Verfügung stehen. Die Lastverteilung erfolgt entweder

- durch Zufall ; z.B. bei DNS-Pooling oder “Round robin DNS” d.h. mehrere Replikanten haben einen gemeinsamen Hostnamen aber verschiedene IP-Adressen, die nach dem Zufallsprinzip ausgewählt werden. (Stoff: https://de.wikipedia.org/wiki/Lastverteilung_per_DNS)
- “geographische” DNS-Verteilung; lokale DNS-Server liefern den jeweiligen Clients unterschiediche IP-Adressen (z. B. in Abhängigkeit von der Niederlassung)
- Konfiguration der Anwendungen; die Anwendungen erhalten unterschiedliche Adressen für die Verbindungsaufnahme , eventuell wird der MySQL Router verwendet, oder
- “geographische” Zuordnung durch Level-3-Switches. Alle Replikanten haben für Verbindungsanfragen eine weitere, gemeinsame IP-Adresse. Welcher Server dann von welchem Client unter dieser IP-Adresse angesprochen wird, hängt von den Einstellungen am Switch ab.

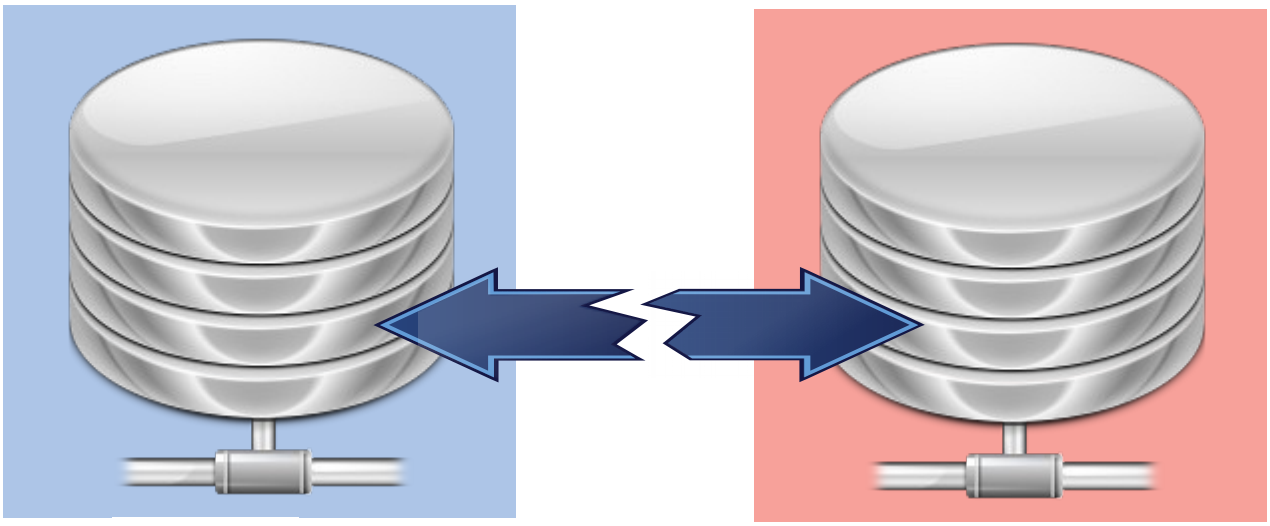
SYSTEME MIT HOHER ÄNDERUNGSLAST

In einem solchen Szeanrio gibt es eine Vielzahl von Clients welche häufig Änderungen im Datenbestand vornehmen. Beispiel hierfür wären Maschinen (Roboter) die in einem Produktionssteuerungssystem Daten nicht nur lesen sondern auch schreiben. Es ist augenscheinlich dass hier “Denkpausen” die Produktion direkt beeinträchtigen. In solchen Fällen wird die Master-Master-Replikation angewendet. Bequem ist, dass bei dieser Replikation auch Änderungen repliziert werden, so dass im Gegensatz zur Master-Client-Replikation für lesende und schreibende Abfragen keine unterschiedlichen Server adressiert werden müssen.

Die Lastverteilung erfolgt wie unter “Systeme mit hoher Abfragelast” beschrieben.

SZENARIO

BESCHREIBUNG



s1.box:: 192.168.1.111

s2.box:: 192.168.1.100

Es existiert bereits ein mit MySQL 5.7 betriebener Server (“s1.box”). Zum Zweck der Lastverteilung sollen nun ein zweiter Server (“s2.box”) aufgesetzt werden. Auf beiden Servern sollen die ursprünglichen Datenbestände des ersten Servers verfügbar sein. Auf beiden Servern sollen Datenbanken (Schemas), Tabellen, Spalten, Zeilen, Werte angelegt und gelöscht werden können und auf dem anderen Server verfügbar sein. (Master-Master-Replikation) Auch Veränderungen an der Nutzerdatenbank sollen übertragen werden. Die beiden Server sollen als Pool unter einem Hostnamen zur Verfügung stehen.

- **Die Vorgehensweise für einen dritten und weitere Server wird in diesem Handbuch nur kurz beschrieben.**
- **Die Vorgehensweise für die Master-Slave-Replikation ist ein Teil dieses Vorgehens – hören Sie einfach auf, wenn der Slave Daten repliziert. (siehe Seite 12 bzw. 13)**

Sie werden festellen, dass es durchaus sinnvoll ist, für die Server neben einem Namen (einer für alle) eine eindeutige Nummerierung zu vergeben.

VORARBEITEN

Für Datenbankserver wird empfohlen, ein minimales Betriebssystem zu verwenden, weil eine graphische Oberfläche lediglich Speicher verbraucht und keinen Nutzen erbringt. Vorliegend wird ein aktuelles Debian (Jessie) verwendet. Empfohlen wird die Netzwerkinstallation von einer Minimal-CD. Diese wird unter der Webadresse https://www.debian.org/CD/netinst/ für Debian beschrieben.

Es wird dringend empfohlen auf beiden Servern identische Versionen von MySQL zu betreiben!

INSTALLATION VON MYSQL 5.7 AUF DEM NEUEN SERVER (DEBIAN/UBUNTU)

Literatur im Web:

- https://dev.mysql.com/downloads/repo/apt/
- https://hilfreiche-server.tips/mysql-5-7-auf-debian-via-apt-installieren/

SCHRITTE:

1. Fügen Sie das offizielle MySQL-Archiv hinzu:

```
root@s2:/# echo “deb http://repo.mysql.com/apt/debian/$(lsb_release -sc) mysql-5.7” > /etc/apt/sources.list.d/mysql.list
```

(eine Zeile)

2. Sicherstellen, dass der Installer “apt” per https downloaden kann:

```
root@s2:/# apt install apt–transport–https
```

3. Schlüssel des Repositories importieren:

```
apt-key adv --keyserver pgp.mit.edu --recv-keys 5072E1F5
```

4. Aktuelle Informationen der Repositories abholen:

```
root@s2:/# apt update
```

5. Den Server voll aktualisieren:

```
root@s2:/# apt full-upgrade
```

6. Installieren des Servers, des Clients und der Pakete, von denen diese abhängig sind:

```
root@s2:/# apt install mysql–community–server
```

Bestätigen Sie die Installation der weiteren Pakete wie angezeigt mit “Y[es]” oder “J[a].

Hinweis: Auf älteren Debian-Installationen müssen Sie anstelle von **`apt`** `update|install|full-upgrade` das ältere **`apt-get`** `update|install|dist-upgrade` verwenden.

7. Halten Sie den neu installierten Server gleich wieder an:

```
root@s2:/# service mysql stop
```

INSTALLATION WEITERER SOFTWARE

Es ist hilfreich, weitere Software zur Verfügung zu haben:

[Optionaler Schritt:]

INSTALLATION VON RSYNC:

Für den Transport der binären Daten von s1.box nach s2.box kann das Programm “rsync” sehr hilfreich sein. Installieren dieses auf beiden Servern:

```
root@s2:/# apt install rsync
root@s1:/# apt install rsync
```

Für die Überprüfung der Erreichbarkeit des Servers kann das Programm nmap hilfreich sein. Installieren Sie dieses auf einem Rechner Ihrer Wahl, welcher beide Server unter allen interessierenden IP-Adressen erreichen kann: (Das kann auch einer der Server sein.)

```
root@s2:/# apt install nmap
```

INSTALLATION VON OPENSSH

Falls der ssh-Server wirklich noch nicht installiert sein sollte installieren Sie diesen wie folgt:

```
root@s2:/# apt install openssh-server openssh-client
root@s1:/# apt install openssh-server openssh-client
```

INSTALLATION DES PROGRAMMS UUID (OPTIONAL)

Für einige Anwendungszwecke (wie das nachfolgend gezeigte Erzeugen einer UUID) ist openssl sehr dienlich:

```
root@s1:/# apt install uuid
```

INSTALLATION VON FAIL2BAN (OPTIONAL)

Sie können die Sicherheit Ihrer Server erhöhen in dem Sie zusätzlich fail2ban installieren.

```
root@s2:/# apt install fail2ban
root@s1:/# apt install fail2ban
```

KONFIGURATIONSARBEITEN FÜR OPENSSH

Wenn Sie die MySQL-Datenbanken per rsync oder scp übertragen wollen ist dieser Schritt unumgänglich.

ÜBERPRÜFUNG DER SSH-KONFIGURATION

Zunächst einmal muss sich sich für die Folgearbeiten der administrative Benutzer “root “auf beiden Rechnern per ssh anmelden können.

Überprüfen Sie deshalb, ob die Konfiguaration des ssh-Servers auf beiden Rechnern folgenden Eintrag enthält: (es werden vermutlich noch Kommentare ausgegeben)

```
root@s2:/# grep “PermitRootLogin” /etc/ssh/sshd.config
PermitRootLogin without-password
```

```
root@s1:/# grep “PermitRootLogin” /etc/ssh/sshd.config
PermitRootLogin without-password
```

Falls nicht sollten Sie diesen Eintrag in der Datei “/etc/ssh/sshd.config” mit einem Editor Ihrer Wahl genau so setzen, sonst müssen Sie auf das deutlich weniger sichere root-login mit einem Passwort setzen.

Wenn Sie an der Datei “/etc/ssh/sshd.config” Änderungen vorgenommen haben müssen Sie den openssh-Server anweisen, diese neue Konfiguration zu verwenden:

```
root@s2:/# service ssh reload
```

```
root@s1:/# service ssh reload
```

ERZEUGEN DER SCHLÜSSELDATEIEN

Legen Sie mit

```
seminar@s1:/~ ssh-keygen -b 4096
seminar@s1:~/.ssh cat < ./ssh/id_rsa.pub >> .ssh/authorized_keys
```

die Schlüsseldateien an und kopieren diese in die Datei “authorized_keys”

WICHTIG:

Sie sollten bei der Ausführung von “ssh-keygen” unbedingt ein Passwort für die Sicherung des Keys vergeben.

Kopieren das verzeichnis .ssh mit allen Datei en anschließend als Root in das Root-Verzeichnis:

```
seminar@s1:~$ su
Passwort:
root@s1:/home/seminar# cd /root
root@s1:~# cp -r /home/seminar/.ssh ./
root@s1:~# chown -R root:root .shh
root@s1:~# chmod 700 .ssh
```

Wechseln Sie jetzt zum Server “s2.box” und kopieren Sie die Schlüssel erst auf den Server und danach (genau so wie eben) für den root:

```
seminar@s2:~$ mkdir .ssh
seminar@s2:~$ chmod 700 .ssh
seminar@s2:~$ cd .ssh
seminar@s2:~./ssh$ scp -r seminar@s1.box:/home/seminar/.ssh/*
seminar@s1:~$ su
Passwort:
root@s2:/home/seminar# cd /root
root@s2:~# cp -r /home/seminar/.ssh ./
root@s2:~# chown -R root:root .shh
root@s2:~# chmod 700 .ssh
```

Hinweis: Wenn Sie Putty benutzen, dann können Sie den Key mit dem Programm Puttygen erzeugen und anschließend auf beide Server kopieren. Am besten so, dass Sie sich zunächst nur als normaler Benutzer anmelden können. Erzeugen in diesem Fall für den Zugriff via ssh mit dem root-Konto einen weiteren Key wie hier vorgestellt. Das Sie ein Passwort für den Key vergeben ist extrem wichtig weil dieser in fremde Hände gelangen kann.

ABSCHLIESSENDER TEST:

Jetzt sollten Sie sich als root auf dem jeweils entfernten Server anmelden können:

```
seminar@s2:~$ ssh root@s1.box
Enter passphrase for key '/home/seminar/.ssh/id_rsa':

[Viel Text entfernt]

Last login: Mon Sep 18 19:05:22 2017 from 192.168.1.100
root@s1:~#
```

Falls das nicht klappt haben Sie was falsch gemacht. Überprüfen Sie den Inhalt der “.ssh” Verzeichnisse auf allen Servern für den Standard-Benutzer und den root.

[“fast” optionaler Schritt:]

ÜBERTRAGUNG VON SERVER-SCHLÜSSELN UND -ZERTIFIKATEN

Falls Sie das Server-Pooling via DNS (Round Robin, bei mobilen Geräten auch “geografisch” unterschiedliche DNS-Einträge oder Level-3-Switching), verwenden wollen stoßen sich einige Clients (zu Recht!) daran, wenn sich der (vermeintlich) gleiche Server mit unterschiedlichen Server-Keys bzw. Zertifikaten ausweist, denn dieses ist ein Hinweis auf eine “Man-in-the-middle”-Attacke.

SSH verbindungen sollten deshalb nicht zu der Pool-Adresse aufgebaut werden! Kopieren Sie, falls Sie nicht (wie nachfolgend beschrieben) ohnenhin die gesamten Verzeichnisse */var/lib*mysql mit rsync “gleichschalten”, also sämtliche Keys und Zertifikate aus /var/lib/mysql auf alle am Pool teilnehmenden Server. Das sind folgende Dateien:

```
ca-key.pm, ca.pem, client-cert.pem, client-ley.pem, public-key.pem,server-cert.pem,
server-key.pem
```

ERSTER SCHRITT: EINRICHTEN DER MASTER-SLAVE-REPLIKATION

EINRICHTEN DES SLAVES BZW. DES ZWEITEN MASTERS

Falls nicht schon geschehen halten Sie den zweiten Server an und ergänzen Sie die Konfigurationsdatei für den MySQL-Server, z.B. mit

```
root@s2:/# service mysql stop
root@s2:/# vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

Es sind folgende Zeilen zu ändern bzw. nachzutragen, verwenden Sie ggf. Für die IP-Adressen andere Werte:

```
#bind-address   = 127.0.0.1
## joerg.reinholz@fastix.org wrote:
bind-address    = 0.0.0.0
```

Wenn Sie den Server “s2.box” **ausschließlich als Slave** betreiben wollen können Sie ihn komplett schreibschützen. Tragen Sie an der gleichen Stelle noch

```
read_only = 1
```

ein.
Wenn Sie aber eine **Master-Master-Replikation** als Ziel haben, dann fügen Sie auch diese Zeilen gleich mit ein:

```
## joerg.reinholz@fastix.org wrote:
# Setting up as master for replication:
log-bin         = mysql-bin
server-id       = 2
innodb_flush_log_at_trx_commit = 1
sync_binlog     = 1
```

Wenn Sie mehrere Master haben wollen, dann müssen diese eine eineindeutige Server-Id erhalten. Es ist sinnvoll, diese auch im Namen des Servers zu führen statt sich hier ein komplizierten Buchhaltung zu ergeben.

Löschen Sie jetzt jede Datei innerhalb von “/var/lib/mysql”:

```
root@s2:/# rm -rf /var/lib/mysql
```

Jetzt folgt die Aufgabe, die Datenbanken vom ersten Server zu kopieren. Das kann “zu Fuß” oder über das Netz geschehen.

EINRICHTEN DES ERSTEN MASTERS

Da es beim nachfolgenden Kopieren der Datenbanken zwingend zu einer Downtime kommt richten wir den ersten Server auch gleich als Master ein:

Es ist eine wirklich gute Idee, zunächst einen MySQL-Benutzer einzurichten, der Replizieren darf:

```
mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> GRANT REPLICATION SLAVE ON *.* TO ‘replikation’@`%` IDENTIFIED BY
```

```
'PASSWORT';
```

Sie benutzen selbstverständlich ein eigenes Passwort. Einige Quellen geben an, man möge den Host in der Form `replikation'@`%.domain.tld`` angeben. Das Funktioniert aber nur bei einem sauberen Reverse-Lookup der IPs – und der ist nicht immer gegeben. Sie können das ggf. mit dem Befehl "`nslookup 192.168.1.100`" prüfen.

```
root@s1:/# vi /etc/mysql/mysql.conf.d/mysqld.cnf
```

Es sind folgende Zeilen zu ändern bzw. nachzutragen, verwenden Sie ggf. Für die IP-Adressen andere Werte:

```
#bind-address   = 127.0.0.1
## joerg.reinholz@fastix.org wrote:
bind-address    = 0.0.0.0
```

Auf dem Master fügen Sie auch diese Zeilen in jedem Fall ein:

```
## joerg.reinholz@fastix.org wrote:
# Setting up as master for replication:
log-bin         = mysql-bin
server-id       = 1
innodb_flush_log_at_trx_commit = 1
sync_binlog     = 1
expire_logs_days = 10
max_binlog_size = 100M
```

Denken Sie an die einzigartige/eineindeutige Server-ID! Die log-bin werden nach 10 Tagen gelöscht. Tut man das nicht füllen diese irgendwann die Platte.

Sie müssen jetzt den Master einmal neu starten:

```
root@s1:/# service mysql restart
```

Klappt das nicht, dann haben Sie sich wahrscheinlich vertippt. "`less /var/log/mysql/error.log`" hilft Ihnen den Fehler zu finden.

Wenn der Server läuft verbinden Sie sich wieder als "root", stoppen die schreibenden Zugriffe und holen sich danach die aktuelle Zeile (Position) im Binären Log, sowie den Name der binären Log-Datei, an welcher der Server gerade arbeitet.

```
mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0,02 sec)
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      154 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0,00 sec)
```

Notieren Sie die Daten für File und Position. Jetzt ist es Zeit, den Server ganz anzuhalten, denn wenn alles gut ging folgt der nächste Schritt:

```
mysql> quit
root@s1:/# service mysql stop
```

ZUM VERSTÄNDNIS: WAS DAS “BINLOG” ÜBERHAUPT IST UND WIE DIE REPLIKATION FUNKTIONIERT

MySQL schreibt die Daten in binäre Dateien mit dem Name “ibdataX” in /var/lib/mysql. Im “binlog” hält MySQL fest, was es an diesen Dateien verändert hat. Man kann das, grob gesagt, mit “diff” und “patch” vergleichen. “diff” liefert den Unterschied zwischen zwei Versionen einer Datei. “patch” kann den gelieferten Unterschied (Einfügungen, Löschungen) in eine alte Version einfügen. Der “Master” liefert den “Slaves” also nummerierte “diffs”. Die Slaves nehmen diese “diffs” entgegen, schauen auf die Positions-Nummer, ob diese schon verarbeitet wurden, und pflegen die unverarbeiteten Änderungen in den eigenen Datenbestand ein.

- Deshalb funktioniert die Replikation nur mit dem Format “innodb” und womöglich nicht zwischen verschiedenen MySQL / MariaDB-Versionen.
- Deshalb ist es eine gute Idee, zu Beginn eine binäre Kopie der Daten des Masters auf die Slaves zu übertragen. Was wir im nächsten Abschnitt tun werden.

DIE DATENBANKEN VOM MASTER AUF DEN SLAVE KOPIEREN

Die Datenbanken müssen jetzt kopiert werden. Wenn der bestehende Server (erster Master, “s1.box”) bereits viele Gigabyte an Daten enthält, dann kann es vorteilhaft sein, diese Daten anders (z.B. durch Kopieren auf eine tragbare Festplatte), also “zu Fuß” zu kopieren. Wie Sie das genau machen ist nahezu egal: Hauptsache alle Daten aus dem Ordner “/var/lib/mysql” werden übertragen! Hier verwende ich das Programm “rsync”. Da ich gerade auf diesem bin, vom ersten Master (“s1.box”) aus:

```
root@s1:/# cd /var/lib/mysql
root@s1:/var/lib/mysql/# rsync -avu ./* root@s2.box:/var/lib/mysql/
receiving incremental file list
[etliche Zeilen mit Dateien und Quittung nicht aufgeführt]
```

Das Übertragen von mehreren hundert Megabyte hat hier in wenigen Minuten geklappt.

Jetzt sollte man zum Slave bzw. “s2.box” wechseln und sich als “root” das Ergebnis ansehen:

```
root@s2:/# cd /var/lib/mysql
root@s2:/var/lib/mysql# ls -l
insgesamt 3835968
drwxr-xr-x 1 mysql mysql        320 Jan  8  2017 geo
-rw-r--r-- 1 mysql mysql 3827302400 Sep 18 11:39 ibdata1
-rw-rw---- 1 mysql mysql   50331648 Sep 18 11:39 ib_logfile0
-rw-rw---- 1 mysql mysql   50331648 Jan  7  2017 ib_logfile1
drwx------ 1 mysql mysql       2798 Aug  5 10:35 mysql
-rw-rw---- 1 mysql mysql        125 Sep 18  2017 mysql-bin.000001
-rw-rw---- 1 mysql mysql        171 Sep 18  2017 mysql-bin.index
-rw-r--r-- 1 mysql mysql         15 Aug  5 10:35 mysql_upgrade_info
drwx------ 1 mysql mysql       3296 Aug  5 10:35 performance_schema
drwx------ 1 mysql mysql        950 Apr  1 20:18 sakila
drwx------ 1 mysql mysql       8750 Mai  5  2016 test
```

Das Ergebnis kann so ähnlich aussehen wie hier gezeigt.

Wenn Sie sich die Daten für File und Position wie auf Seite 10 dargestellt unten gut gemerkt oder bessser noch notiert haben können wir jetzt wir auf dem Master die Tabellen wieder zum Schreiben freigeben:

```
mysql -u root -p
Enter password:
```

```
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> UNLOCK TABLES;
```

Wir lassen den Master so stehen und laufen wie er ist. “s1.box” kann ab jetzt wieder lesend und schreibend benutzt werden.

DEN ZWEITEN SERVER ZUM SLAVE BESTIMMEN

FÜR MYSQL 5.7 UND SPÄTER: UUID DES SERVERS ÄNDERN

Ab MySQL 5.7 gibt es für jeden Server eine UUID, die einzigartig bzw, eineindeutig sein muss. Lax ausgedrückt handelt es sich hierbei um eine sehr große Zahl die entweder zufällig gebildet wird oder aus den Komponenten Zeit und Zufall besteht und ein paar Minus-Zeichen. Diese, zwecks der Vermeidung zufälliger Übereinstimmungen gigantisch große Zahl ist, nicht ganz fachgerecht, in einer Datei gespeichert, die sich in /var/lib/mysql befindet und und auto.cnf heisst. Wir müssen diese ändern.

Eine neue UUID kann man mit dem Programm uuid bauen:

```
root@s2:/# uuid
722bbb96-9ca8-11e7-9fac-b7ff557e9ba4
```

Sie können diese UUID in die Zwischenablage kopieren und dann anstelle der bestehenden in die Datei /var/lib/mysql/auto.cnf einfügen - oder wie folgt vorgehen:

```
root@s2:/# echo “[auto]” > /var/lib/mysql/auto.cnf
root@s2:/# echo "server-uuid=$(uuid)" >> /var/lib/mysql/auto.cnf
```

Danach sollte diese Datei (bis auf die natürlich andere UUID) folgenden Inhalt haben:

```
[auto]
server-uuid=722bbb96-9ca8-11e7-9fac-b7ff557e9ba4
```

DEN SLAVE-MODUS STARTEN

Der MySQL-Server auf dem Slave wird zunächst ganz normal gestartet. Es sollte sich niemand anders als der root einloggen. Insbesondere darf an den Datenbanken nichts schreibend geändert werden:

```
root@s2:/# service mysql start
```

Jetzt ist es Zeit, sich als root mit der Datenbank zu verbinden und diesem mitzuteilen, dass er als Slave zu arbeiten hat:

```
root@s2:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> CHANGE MASTER TO MASTER_HOST=”s1.box”,
           MASTER_USER=”replikation”,
           MASTER_PASSWORD=”PASSWORT”,
           MASTER_LOG_FILE=”mysql-bin.000001”,
           MASTER_LOG_POS=154
;
Query OK, 0 rows affected (0,02 sec)
mysql> START SLAVE;
```

Das kann natürlich auch in einer Zeile geschehen. Die Daten für den USER und das PASSWORT haben Sie sich natürlich gemerkt. Die Daten für MASTER_LOG_FILE und MASTER_LOG_POS haben Sie sich wie auf Seite 10 stehend gemerkt oder aufgeschrieben…

Falls Sie nur eine Master-Slave-Replikation wollten (z.B. für einen Backup-Server), dann sind Sie schon fertig! Sie müssen nichts weiter ändern, Der Slave merkt sich diese Angaben auch über einen kompletten Neustart hinaus und beginnt zu replizieren sobald er einen Master findet und dieser ihm Daten sendet.

MEHRERE SLAVES (MULTI SLAVES)

Bei mehreren Slaves gegen Sie identisch vor. Halten Sie den Master an, ermitteln Sie wie auf Seite 10 gezeigt binlog -Datei und Position, Richten Sie den Slave durch Kopieren der Datenbank-Dateien und Ändern der UUID ein. Starten Sie den SLAVE-Modus wie auf Seite 12 gezeigt.

MASTER-MASTER-REPLIKATION EINRICHTEN

ZUM VERSTÄNDNIS

Die Crux an der Master-Master-Replikation ist, dass es sich eigentlich um eine “Kreisreplikation” handelt. Vorliegend werden die Daten wie folgt repliziert:

“s1.box” schickt die Änderungen (abgeschlossene Transaktionen) zu “s2.box”, “s2.box” wiederum schickt seine Daten (abgeschlossene Transaktionen) an “s1.box”. Also ist jeder der Master zugleich Slave eines anderen.

Aus diesem Grund muss der zukünftige zweite Master vor den weiteren Schritten als Slave laufen.

VOM SLAVE ZUM MASTER

Wenn Sie es noch nicht (wie auf Seite 9 vorgeschlagen) getan haben sollten, dann richten Sie den zweiten Server (“s2.box”) wie folgt als Master ein.

Ändern Sie dazu die Datei “/etc/mysql/mysql.conf.d/mysqld.cnf” in dem Sie folgendes einfügen.

```
## joerg.reinholz@fastix.org wrote:
# Setting up as master for replication:
log-bin         = mysql-bin
server-id       = 2
innodb_flush_log_at_trx_commit = 1
sync_binlog     = 1
expire_logs_days = 10
max_binlog_size = 100M
```

Starten Sie dann den MySQL-Server neu, sorgen Sie aber dafür, dass niemand außer Ihnen schreibend zugreift:

```
root@s2:/# service mysql restart
```

Loggen Sie sich in MySQL ein und ermitteln Sie wie auf Seite 10 beschrieben das bin-log-file und die Position – aber auf dem bisherigen Slave:

```
mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0,02 sec)
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      364 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0,00 sec)
```

Notieren Sie die Daten für File und Position. Wechseln Sie zum bisherigen reinen Master, loggen Sie sich in MySQL ein und initialisieren Sie, dass dieser als Slave arbeitet:

```
seminar@s1:/~ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [Viel Text entfernt]
mysql> CHANGE MASTER TO MASTER_HOST=”s2.box”,
          MASTER_USER=”replikation”,
          MASTER_PASSWORD=”PASSWORT”,
          MASTER_LOG_FILE=”mysql-bin.000001”,
          MASTER_LOG_POS=364
;
Query OK, 0 rows affected (0,02 sec)
mysql> START SLAVE;
```

Die Daten stammen aus der Ausgabe von Seite 13 , Benutzername und Passwort für den Replikations-User haben Sie an den “s2.box” mit übertragen, als Sie die Dateien aus dem Verzeichnis /var/lib/mysql übertragen haben (Seite 11).

Jetzt sollte die “Kreisleplikation” laufen.

TESTS

MASTER-SLAVE-REPLIKATION

Loggen Sie sich auf dem Master als root ein. Legen Sie auf dem Master eine neue Datenbank (seit einiger Zeit korrekt neues “Schema”) an.

```
mysql> create schema test;
Query OK, 1 row affected (0,09 sec)
```

Loggen Sie sich auf dem Slave ebenfalls als root in MySQL ein. Prüfen Sie auf dem Slave, ob Sie in dieses wechseln können:

```
mysql> use test
Database changed
```

Erstellen Sie auf dem Master eine einfache Tabelle und fügen Sie Daten ein:

```
mysql> CREATE TABLE test (id int, string varchar(20));
Query OK, 0 rows affected (0,05 sec)
mysql> insert into test (id, string) values(1, "hallo");
Query OK, 1 row affected (0,00 sec)
```

Fragen Sie die Tabelle auf dem Client ab:

```
mysql> show tables;
+-----------------+
| Tables_in_test |
+-----------------+
| test            |
+-----------------+
1 row in set (0,00 sec)
mysql> select * from test;
+------+--------+
| id   | string |
+------+--------+
|    1 | hallo  |
+------+--------+
1 row in set (0,00 sec)
```

Wenn das so funktioniert, dann funktioniert die Master-Slave Replikation.

MASTER-MASTER-REPLIKATION:

Loggen Sie sich auf dem “s1.box” als root ein. Legen Sie auf dem Master eine neue Datenbank (seit einiger Zeit korrekt neues “Schema”) an.

```
mysql> create schema test;
Query OK, 1 row affected (0,09 sec)
```

Loggen Sie sich auf dem Slave ebenfalls als root in MySQL ein. Prüfen Sie auf dem Slave, ob Sie in dieses wechseln können:

```
mysql> use test
Database changed
```

Erstellen Sie auf dem zweiten Maser (s2.box) eine einfache Tabelle und fügen Sie Daten ein:

```
mysql> CREATE TABLE test (id int, string varchar(20));
Query OK, 0 rows affected (0,05 sec)
mysql> insert into test (id, string) values(1, "hallo");
Query OK, 1 row affected (0,00 sec)
```

Fragen Sie die Tabelle auf dem ersten Master (“s1.box”) ab:

```
mysql> use test;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_test |
+-----------------+
| test            |
+-----------------+
1 row in set (0,00 sec)

mysql> select * from test;
+------+--------+
| id   | string |
+------+--------+
|    1 | hallo  |
+------+--------+
1 row in set (0,00 sec)
```

Wenn das so funktioniert, dann funktioniert die Master-Master Replikation.

Dann sind Sie fertig und können direkt zum Server-Pooling schreiten:

SERVER-POOLING MIT ROUND-ROBIN

Voraussetzung für das Server-Pooling mit Round-Robin ist, dass Sie die DNS-Server Ihrer Domain selbst verwalten.

Für den Testfall habe ich folgende Daten verwendet: (Auszug aus /etc/bind/db.box:)

```
s1              IN A            192.168.1.111
s2              IN A            192.168.1.100

db-pool         IN A            192.168.1.111
db-pool         IN A            192.168.1.100
```

nslookup liefert folgende Ergebnisse:

```
~> nslookup db-pool.box
Server:         192.168.1.251
Address:        192.168.1.251#53

Name:   db-pool.box
Address: 192.168.1.111
Name:   db-pool.box
Address: 192.168.1.100


~> nslookup db-pool.box
Server:         192.168.1.251
Address:        192.168.1.251#53

Name:   db-pool.box
Address: 192.168.1.100
Name:   db-pool.box
Address: 192.168.1.111
```

Dementsprechend verbindet sich der MySQL abwechselnd mit einem zufälligem Host. Ist einer ausgefallen/heruntergefahren, dann wählt der MySQL-Client so lange eine andere IP aus dem Pool bis ein erreichbarer Host gefunden wurde. Der Login dauert dann etwas länger. Während der Sitzung wird der Host aber nicht gewechselt.

MEHR ALS ZWEI MASTER

Bei einer höheren Zahl von Mastern richten Sie die Replikation wie folgt ein:

Genau wie bei zwei Mastern wird jeder Server wird als Master eingerichtet,

Server 1 wird Slave von Server 3; Server 2 wird Slave von Server 1; Server 3 wird Slave von Server 2;

Sollten zusätzlich noch weitere, reine Slave-Server erwünscht sein, so können diese an einen beliebigen Master gebunden werden.

WEITERFÜHRENDE LITERATUR / WEBLINKS

- MySQL 5.7 Reference Manual, Chapter 16 Replication (englisch)
  https://dev.mysql.com/doc/refman/5.7/en/replication.html
- Die Übersicht über alle Server-Variablen von MySQL 5.7 (englisch) liefert erhellende Einsichten.
  https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html
- Thomas Krenn: MySQL Replikation (deutsch)
  https://www.thomas-krenn.com/de/wiki/MySQL_Replikation
- Debian Handbücher (deutsch, englisch, ...)
  https://www.debian.org/doc/user-manuals.de.html
- Handbuch zur Middleware “Mysql-Router”
  https://dev.mysql.com/doc/mysql-router/2.1/en/

LIZENZ