

# SQL-SSH-Backup

Mit SQL-SSH-Backup lassen sich MySQL- und PostgreSQL-Pull-Sicherungen erstellen. Der Backupserver loggt sich dazu per SSH ein und führt die Befehle zur Sicherung auf dem zu sichernden Server aus. Der Backupserver benötigt dabei keinerlei MySQL- oder PostgreSQL-Module oder -Programme. Jede Datenbank kann einzeln gesichert, was eine Rücksicherung erleichtert.

## Copyright / Lizenz

SQL-SSH-Backup entstand hauptsächlich durch das Webserver-Team im Rahmen des Webhostings der FeM e.V. und steht unter der [GPL-2](#).

## Features (der aktuellen Version)

- Dump für MySQL und PostgreSQL via SSH
  - benötigt kein MySQL/PostgreSQL auf dem Backup-Server
  - sichere Übertragung/Nutzung der Datenbank-Zugangsdaten möglich
  - Einzel- oder Gesamtsicherung aller Datenbanken
  - Sicherung einzelner Datenbanken
- Kompression der Dumps mit bzip2/xz (MySQL) oder PostgreSQL-Dump-Format möglich
- mehrere Backups täglich möglich (Verzeichnisstruktur enthält dann Uhrzeit)
- automatische Bereinigung von alten Backups

## Installation

### Backupserver

Auf dem Backupserver wird neben SQL-SSH-Backup nur SSH (meist vorhanden) und ggf. bzip2/xz (für komprimierte MySQL-Backups) benötigt.

Das Skript steht unter

<http://subversion.fem.tu-ilmenau.de/repository/fem-overlay/trunk/app-backup/sql-ssh-backup/files/> oder im Gentoo-FeM-Overlay (***app-backup/sql-ssh-backup***) zur Verfügung.

Versionen:

- 1.3 - erstes kombiniertes MySQL/PgSQL-Release
- 1.4 - Zeitangabe in Sicherungsordner integrieren (erlaubt Mehrfach-Sicherungen an einem Tag)
- 1.5 - Konfigurationsdatei für Benutzer/Passwort bei MySQL (verhindert auslesen des Passworts via ps), SSH-Port-Angabe
- 1.6 - Konfigurationsdatei für Benutzer/Passwort bei PgSQL (verhindert auslesen des Passworts via ps)
- 1.7 - Gesamtdump aller Datenbanken statt Einzeldump ermöglichen
- 1.8 - Sicherung einer einzelnen Datenbank möglich

- 1.9 - Wiederholung von Dumps bei Fehlern ermöglichen (Retry), bereinigte Exit-Codes, kleinere Fehlerkorrekturen

- ***chown backup:backup /home/backup/.ssh/\****

Der Private Schlüssel für den Backup-Nutzer (/home/backup/.ssh/id\_rsa) muss nun sicher (z.B. per SCP) auf den Backup-Server übertragen werden. Anschließend kann er auf dem zu sichernden Server gelöscht werden.

- ***rm /home/backup/.ssh/id\_rsa***

## Datenbank-Benutzer anlegen

### MySQL

Der Backup-Benutzer benötigt lediglich folgende Rechte

- SELECT
- SHOW DATABASES
- LOCK TABLES

Benutzer anlegen - passendes PASSWORT wählen

```
CREATE USER 'backup'@'localhost' IDENTIFIED BY '<PASSWORT>';
GRANT USAGE ON * . * TO 'backup'@'localhost' IDENTIFIED BY '<PASSWORT>' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0 ;

REVOKE ALL PRIVILEGES ON * . * FROM 'backup'@'localhost';
REVOKE GRANT OPTION ON * . * FROM 'backup'@'localhost';

GRANT SELECT, SHOW DATABASES, LOCK TABLES ON * . * TO 'backup'@'localhost'
WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR
0 MAX_USER_CONNECTIONS 0;
```

### PostgreSQL

Der Backup-Benutzer benötigt folgende Rechte nicht:

- Datenbanken anlegen
- Rollen anlegen

Er benötigt die Rolle:

- Superuser

Folgendes als root auf dem zu sichernden Server ausführen und erst das Passwort (und Wiederholung) für den neuen Nutzer und dann ggf. das Passwort für den postgres-Benutzer eingeben:

- ***su - postgres***

- **`createuser -D -R -s backup -P`**

## Backupserver

Annahmen:

- Server: server1.example.org (Port 1022)
- Backup-Verzeichnis: /mnt/backup/server1.example.org/backupsql/
- SSH-Key: /mnt/backup/server1.example.org/identity

## SSH-Hostkey speichern

Zu allererst muss man sich einmal per SSH mit dem zu sichernden Server als der Benutzer auf dem Backupserver verbinden unter dem dann das Backup läuft. Der SSH-Key muss dem Nutzer gehören und „rw——“ (600) als Benutzerrechte haben.

- **`chmod 600 /mnt/backup/server1.example.org/identity`**
- **`ssh backup@server1.example.org -i /mnt/backup/server1.example.org/identity`**
  - Are you sure you want to continue connecting (yes/no)? **yes**

Ausgabe

```
The authenticity of host 'server1.example.org (10.200.2.211)' can't be
established.
RSA key fingerprint is c9:08:84:ea:1c:2c:84:5a:0f:cd:6e:1f:a1:bf:fe:4a.
Are you sure you want to continue connecting (yes/no)?
```

Nun sollte man als Nutzer Backup auf dem genannten Server eingeloggt sein.

## Backup

Auf dem Backupserver kann nun das Backup anstoßen werden. Anschließend kann man dies auch per Cronjob erledigen lassen. Der Parameter „-e“ gibt an, wieviele Tage Backups vorgehalten werden - im Beispiel sind das die letzten 120 Tage (Standard: 90).

## MySQL

- **`sql-ssh-backup -T mysql -l backup -s server1.example.org -P 1022 -d /mnt/backup/server1.example.org/backupsql -i /mnt/backup/server1.example.org/identity -e 120 -u backup -p PASSWORT`**

**PostgreSQL**

- ***sql-ssh-backup -T pgsql -l backup -s server1.example.org -P 1022 -d /mnt/backup/server1.example.org/backupsql -i /mnt/backup/server1.example.org/identity -e 120 -u backup -p PASSWORD***

From:

<https://wiki.fem.tu-ilmenau.de/> - **FeM-Wiki**

Permanent link:

<https://wiki.fem.tu-ilmenau.de/public/technik/howto/sql-ssh-backup?rev=1373111576>

Last update: **2013/07/06 13:52**

