



```
SELECT *  
FROM World  
WHERE "Someone"  
LIKE "%You%"
```

```
...  
/> no results !
```

mySQL Server

Installation und Konfiguration mySQL Server auf
Linux Mint

Jonas Hüsser

IS14BLf
Modul 141
25.02.16

Inhaltsverzeichnis

1	mySQL Server	1
1.1	Zweck dieses Dokuments	1
1.2	Allgemeine Daten	1
2	Installation & Konfiguration	2
2.1	Installation mysql-server	2
2.2	Konfiguration	2
2.2.1	automatischer Start	2
2.2.2	Absichern des Servers	2
2.2.3	Root-Passwort ändern	3
2.2.4	Initialisierung & Import	3
3	Wichtige Hinweise	5
4	Skripte	6
4.1	Restart Service	6
4.2	Backup	7
4.3	Recovery	10
4.4	Installation	13

1 MySQL Server

1.1 Zweck dieses Dokuments

Dieses Dokument dokumentiert die Installation, die Konfiguration sowie die Entwicklung der Scripts für einen MySQL Server auf GNU/Linux Basis. Es erfüllt die Anforderungen gemäss Moodle-Abgabe des Leistungsnachweis 1.

1.2 Allgemeine Daten

Arbeitsmaschine:	VM
Betriebssystem:	Linux Mint cinnamon 32bit
Basierend auf:	Debian
Paketmanager:	apt
Virtualisierungssoftware:	VirtualBox
Host:	iMac
Host-Betriebssystem:	OS 10.11.3 (OS X El Capitan)

2 Installation & Konfiguration

2.1 Installation mysql-server

Als erstes werden die Paketquellen aktualisiert. Danach das Paket `mysql-server` installiert. Während der Installation wird auch gleich das Passwort des root Benutzers festgelegt. Hier ist zu beachten, dass dies der root Nutzer des MySQL-Servers ist und nicht der, des Systems. Zwischen den beiden besteht kein direkter Zusammenhang. Auch haben die anderen SQL-Benutzer keinen Systemuser und vice versa. Um auf den MySQL-Server zugreifen zu können, muss der Befehl `mysql` eingegeben werden, gefolgt von den Parametern `-u` und `-p` für Username und Passwort. Nun können die MySQL spezifischen Befehle ausgeführt werden. Doch als allererstes muss der Server fertig konfiguriert werden.

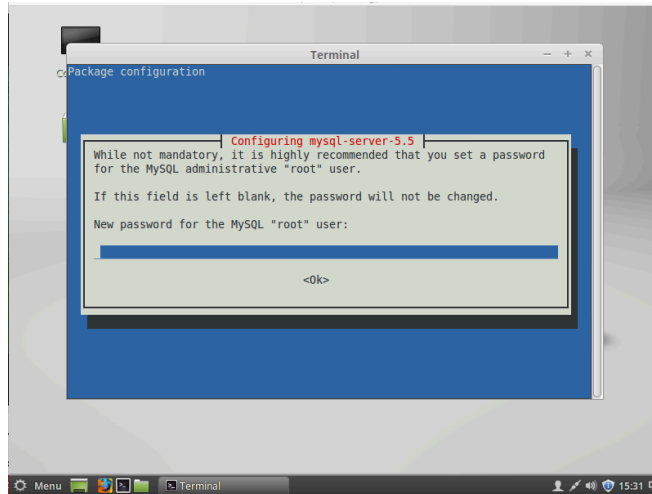


Abbildung 1 Festlegen des root-Passworts während der Installation

2.2 Konfiguration

2.2.1 automatischer Start

Normalerweise ist der Service so konfiguriert, dass er automatisch startet. Sollte dies nicht der Fall sein, kann dies mit dem Befehl behoben `sudo update-rc.d mysql enable` werden.

2.2.2 Absichern des Servers

Der Server wird mit dem Befehl `mysql_secure_installation`

abgesichert. Danach werden ein paar Fragen gestellt. Als erstes das Ändern des root-Passworts, dieses werden wir aber mit einem separaten Befehl erledigen. Als nächstes werden die anonymen Benutzer entfernt, dann das Remote-Login deaktiviert, die Testdatenbank sowie die dazugehörenden Berechtigungen entfernt und zuletzt die Berechtigungstabelle aktualisiert.

2.2.3 Root-Passwort ändern

Mit dem Befehl `mysqladmin` wird der Server administriert. So wird auch das Passwort des root Benutzers geändert. Wir loggen uns wie bereits beschrieben ein und ändern gleich mit dem Befehl `password` das Passwort.

```
jhuesser@vmssql1 ~ $ mysqladmin -u root -p'1234' password 's!"41A45Çd1"#vd'
```

Abbildung 2 Ändern des mysql-root-Passworts

2.2.4 Initialisierung & Import

2.2.4.1 Initialisierung

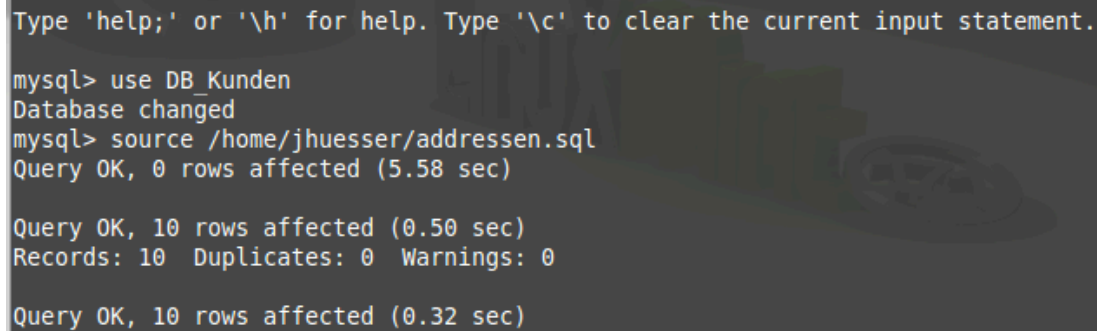
Als nächstes muss eine Datenbank mit dem Namen „Kunden“ erstellt werden. Dies geschieht mit dem Befehl `CREATE DATABASE Kunden;`. Man beachte, dass die SQL Syntax am Ende immer ein Semikolon hat.

```
mysql> CREATE DATABASE DB_Kunden;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> 
```

Abbildung 3 Erstellen der Datenbank "DB_Kunden"

2.2.4.2 Import

Die Beispieldaten von Moodle müssen nun in die DB importiert werden. Die Daten sind in einem SQL-File gespeichert, welches auch gleich die Tabelle „adressen“ erstellt. Ich habe die Datei modifiziert, sodass eine Tabelle „tbl_adressen“ erstellt wird. Dazu wird mit dem Befehl `use` die gewünschte DB ausgewählt und mit `source <path>` das Source-File angegeben



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use DB_Kunden
Database changed
mysql> source /home/jhuesser/adressen.sql
Query OK, 0 rows affected (5.58 sec)

Query OK, 10 rows affected (0.50 sec)
Records: 10  Duplicates: 0  Warnings: 0

Query OK, 10 rows affected (0.32 sec)
```

Abbildung 4 Import von Daten

3 Wichtige Hinweise

- Es muss zuerst das Installationsskript ausgeführt werden.
- Wegen der Sicherheit werden die Anmeldedaten mit einem Schlüsselpaar verschlüsselt
- Der einzige Sicherheitsfehler ist das hardgecodete Passwort zur Dateiverschlüsselung.
- Auch der obengenannte Schritt wäre theoretisch mit einem Schlüsselpaar machbar.
- Jedes Skript wurde auf obengenannten System getestet. Sie funktionierten stets.

4 Skripte

4.1 Restart Service

Den mysql-Service neuzutarten, ist nur ein Befehl. Um den Aufruf einfacher zu gestalten, heisst das Skript `remysql` und befindet sich unter `/bin/`. Hier das Skript:

```
#!/bin/bash
service mysql restart
service mysql status
```


4.2 Backup

```
#!/bin/bash

#Working Values
database=$1
workingdir=/tmp/sqlbak
configdir=/etc/mysql-bzu
backupdir=/backup
DATE=`date +%Y-%m-%d:%H:%M:%S`
keydir=${configdir}/keys
privkey=${keydir}/.private/private_key.pem
mysqluserfile=${configdir}/mysqluser.txt.encrypted
mysqlpassfile=${configdir}/mysqlpass.txt.encrypted
password="F45e$q%fh}" #unfortunately hardcoded

#Defines default database
if [[ -z $database ]];
then
    database="DB_Kunden"
fi

#decrypt credentials
openssl rsautl -decrypt -inkey $privkey -in $mysqluserfile -out ${configdir}/mysqluser.txt
openssl rsautl -decrypt -inkey $privkey -in $mysqlpassfile -out ${configdir}/mysqlpass.txt
sqluser=`cat ${configdir}/mysqluser.txt`
sqlpass=`cat ${configdir}/mysqlpass.txt`

#creates directories if not already exist
mkdir -p $workingdir
```

```

mkdir -p $backupdir
cd $backupdir

#counts files in $backupdir
filenmbr=`ls | wc -l`

#If files >6 delete oldest
if [ $filenmbr -gt 6 ];
then
    oldestfile=`ls -rt | head -1`
    rm -rf $oldestfile
fi

#change to workingdir
cd $workingdir
#filename is database and date
_file=${database}.sql.${DATE}
#login to mysql and get $database. save to $_file
mysqldump -u root -p$sqlpass --databases $database > $_file
#make archiv of backup, and encrypt it. output is $_file.encrypted
tar cvzf - $_file | openssl des3 -salt -k $password | dd of=${_file}.encrypted
#$_file is know encrypted file
_file=${_file}.encrypted
#move backupd file to $workingdir
mv $_file ${backupdir}/${_file}
#Clean up $workingdir
rm -rf $workingdir
rm ${configdir}/mysqluser.txt
rm ${configdir}/mysqlpass.txt

```

Das Skript ist in verschiedene Blöcke eingeteilt. Die eigentliche Aktion wird jeweils im zweitletzten Block gemacht. Doch beginnen wir von vorne. Zuerst werden alle Variablen definiert. Im Backupverzeichnis werden die Backups gesichert, im Arbeitsverzeichnis die Manipulationen gemacht und im Konfigurationsverzeichnis die Einstellungen gespeichert. Dazu zählen die MySQL-Credentials sowie Private- und Publickey. Dies wird alles mit dem Installationsskript durchgeführt, welches zwingend als erstes ausgeführt werden muss. Es besteht die Möglichkeit eine Datenbank per Parameter mitzugeben, dies habe ich implementiert, damit es einfach ist die Skripte auf ein anderes System zu migrieren. Allerdings wird die Datenbank, welche im Auftrag definiert ist, als Standarddatenbank verwendet, falls keine benutzerdefinierte DB angegeben wird. Im dritten Block wird der Benutzername sowie das Passwort des mysql-Benutzers entschlüsselt und in Variablen gespeichert. Einen Block weiter wird überprüft ob das Backup- und das Arbeitsverzeichnis vorhanden sind, ist dies nicht der Fall, werden diese erstellt. Nun werden die Anzahl Dateien im Backupverzeichnis gezählt. Falls diese grösser als 6, also 7 ist, wird die älteste Datei gelöscht. Im Arbeitsverzeichnis wird nun mit mysqldump ein Backup der gewünschten Datenbank erstellt. Danach wird es komprimiert und drei Mal mit des verschlüsselt. Jetzt wird das aktuelle Datum als Suffix angehängt und die Datei ins Backupverzeichnis verschoben. Im letzten Block werden schlussendlich alle Daten im Arbeitsverzeichnis, sowie die unverschlüsselten Anmeldedaten gelöscht.

Das Backup muss laut Auftrag Zeitgesteuert sein. Dies kann manuell mit dem Befehl `crontab -e` gemacht werden. In die nun geöffnete Datei, muss folgende Zeile zuunterst eingetragen werden. Allerdings wird auch dies vom Installationsskript übernommen.

```
00 00 * * * /bin/sqlbak
```

4.3 Recovery

```
#!/bin/bash

#Working Values
database=$1
workingdir=/tmp/sqlbak
configdir=/etc/mysql-bzu
backupdir=/backup
keydir=${configdir}/keys
privkey=${keydir}/.private/private_key.pem
mysqluserfile=${configdir}/mysqluser.txt.encrypted
mysqlpassfile=${configdir}/mysqlpass.txt.encrypted
password="F45e$q%fh}" #unfortunately hardcoded

#Use default DB name if empty
if [[ -z $database ]];
then
    database="DB_Kunden"
fi

#decrypt credentials
openssl rsautl -decrypt -inkey $privkey -in $mysqluserfile -out ${configdir}/mysqluser.txt
openssl rsautl -decrypt -inkey $privkey -in $mysqlpassfile -out ${configdir}/mysqlpass.txt
sqluser=`cat ${configdir}/mysqluser.txt`
sqlpass=`cat ${configdir}/mysqlpass.txt`

#Create and change directory
mkdir -p $workingdir
cd $backupdir
```

```

#search latest backup
latest=`ls -t | head -1`

#copy encrypted backup to workingdir
cp $latest ${workingdir}/$latest
cd $workingdir

#decrypt and unpack the backup
dd if=$latest | openssl des3 -d -k $password | tar xvfz -

#remove encrypted backup copy
rm -f $latest

#select latest file (=decrypted backup)
decryptedFile=`ls -t | head -1`

#create commands file and fill in with commands
touch commands.sql
echo "CREATE DATABASE IF NOT EXISTS ${database};" >> commands.sql
echo "USE ${database};" >> commands.sql
echo "SOURCE ${workingdir}/${decryptedFile}" >> commands.sql

#execute the commands
mysql -u $sqluser -p$sqlpass < commands.sql
#clean up working directory
rm -rf $workingdir
rm ${configdir}/mysqluser.txt
rm ${configdir}/mysqlpass.txt

```

Auch hier werden im ersten Block die Arbeitsvariablen festgelegt. Es sind wieder der Backupspeicherort, das Arbeitsverzeichnis, die Datenbank und das Passwort, sowie der Privatekey. Um mit dem Backupskript kompatibel zu sein, wurde ebenfalls die Möglichkeit gegeben eine andere Datenbank zu wählen, wird diese nicht genutzt, kommt hier ebenfalls die gleiche Standard-Datenbank zum Zuge. Auch in den nächsten Blöcken bleibt das Skript noch ziemlich gleich. Es wird das benötigte Verzeichnis erstellt, und ins Backupverzeichnis gewechselt, nachdem die MySQL-Usercredentials entschlüsselt und in die Variablen gespeichert wurden. Im Backupverzeichnis dann, wird das neuste Backup eruiert und ins Arbeitsverzeichnis kopiert. Dort wird es entschlüsselt und entpackt. Die Kopie des verschlüsselten Backups wird daraufhin entfernt. Im nächsten Schritt werden alle MySQL-Befehle in eine Datei geschrieben. Bei einem Backup ist in der SQL-Datei normalerweise die erste Zeile bereits vorhanden. Falls dies aber aus einem Grund nicht der Fall sein sollte, habe ich sie ins Recovery-Script implementiert. Theoretisch wäre auch ein direkter Import, mit einer Zeile Code möglich. Da ich allerdings die erwähnte Zeile zur Sicherheit eingefügt habe, habe ich den Weg eines externen Files gewählt. Die anderen beiden Zeilen kennen wir bereits aus dem Backup-Script. Zuletzt logt sich das Script wieder auf dem MySQL-Server an und führt die Befehle aus der Datei aus. Danach wird das Arbeitsverzeichnis und die unverschlüsselten Anmeldedaten gelöscht

4.4 Installation

```
#!/bin/bash

#Script starts at bottom.

function instellation {
    #Starts full instellation
    #make scripts executable
    chmod +x sqlbak.sh
    chmod +x restart.sh
    chmod +x recovery.sh
    #copy to /bin for easy call
    mv sqlbak.sh /bin/sqlbak
    mv restart.sh /bin/remysql
    mv recovery.sh /bin/mysql-recovery
    #make config directories for config & key pairs
    mkdir -p $configdir
    mkdir -p $keydir
    mkdir -p $privkeydir
    mkdir -p $pubkeydir
}

function encryptCredentials {
    #Encrypts mysql-User credentials
    #File to encrypt
    file=$1
    #encrypt file with public key
    openssl rsautl -encrypt -inkey $pubkey -pubin -in $file -out ${file}.encrypted
}
}
```

```

function generateKey {
    #Generates keypair
    #Ask if key pair exist.
    read -p "Do you need a keypair (only on first installation)? [Y/n] " needkey
    if [[ -z "$needkey" ]] || [[ $needkey == "Y" ]] || [[ $needkey == "y" ]]; then
        #Generate private key
        openssl genpkey -algorithm RSA -out ${privkeydir}/private_key.pem -pkeyopt
rsa_keygen_bits:2048

        #Generate public key
        openssl rsa -pubout -in ${privkeydir}/private_key.pem \
-out ${pubkeydir}/public_key.pem

        elif [[ $needkey == "n" ]] || [[ $needkey == "N" ]]; then
            #Do nothing
            echo "No keys generated"
            return

        else
            #Ask again
            generateKey
            return
        fi
    fi

}

function credentials {

```



```

#Save mysql credentials
#Generates keys
generateKey
#set key values
privkey=${privkeydir}/private_key.pem
pubkey=${pubkeydir}/public_key.pem
#ask for credentials and saves to value
read -p "Enter a valid mySQL-Username: " sqluser
read -s -p "Enter the mySQL Passwordfor user $sqluser: " sqlpass
echo ''
#save credentials to file
echo $sqluser > ${configdir}/$userfile
echo $sqlpass > ${configdir}/$passfile
#encrypt credentials
encryptCredentials ${configdir}/$userfile
encryptCredentials ${configdir}/$passfile
rm ${configdir}/$userfile
rm ${configdir}/$passfile

}

function registerCronjob {
#make file
touch mycron
#write cronjob to file
echo "00 00 * * * /bin/sqlbak" > mycron
#register cronjob
crontab mycron
#remove file

```

```

        rm -f mycron
    }
    function _start {
        #starts here
        #full instellation or just new mysql-credentials'
        read -p "Do you want a complete instellation or just update the credentials? [C/u]" \
choice
        if [[ -z "$choice" ]] || [[ $choice == "c" ]] || [[ $choice == "C" ]]; then
            #perform full isntellation
            instellation
            credentials
            registerCronjob

        elif [[ $choice == "u" ]] || [[ $choice == "U" ]]; then
            #change credentials
            credentials

        else
            #start again
            _start
            return
        fi
    }

    #SCRIPT STARTS HERE

    #working values
    #configdir with credentials and keys stored
    configdir=/etc/mysql-bzu
    #file with username
    userfile=mysqluser.txt

```

```
#file with password
passfile=mysqlpass.txt
#dir with keypair
keydir=${configdir}/keys
#dir with the privatekey
privkeydir=${keydir}/.private
#dir with the publickey
pubkeydir=${keydir}/public
#Starts the actual script
_start
```

Ich testete die Skripte auf verschiedenen Systemen. Da es mühsam war diese immer zu übernehmen habe ich ein kleines Installationsskript geschrieben. Am Anfang war dies wirklich nur eine Hilfe, mittlerweile muss es zwingend ausgeführt werden.

Das Skript ist hier in mehrere Funktionen aufgeteilt. Dies da unterschieden wird zwischen einer Vollinstallation und einer Anpassung der mysql-Usercredentials. Im Skript wird zuerst das Konfigurationsverzeichnis angegeben. Danach die Dateien welche den Benutzernamen sowie das Passwort speichern. Als letztes wird festgelegt wo der Private- und der Publickey abgelegt werden. Danach wird die Funktion `_start` aufgerufen.

In dieser Funktion wird der Benutzer gefragt ob er die Vollinstallation oder nur eine Anpassung der Anmeldedaten durchführen will. Je nachdem werden einige Funktionen weggelassen.

In Der Funktion `instellation`, wird jedem Skript Ausführrechte gegeben, sowie mit einem einzigartigen Namen nach `/bin` verschoben. Ausserdem werden das Konfigurationsverzeichnis sowie die Schlüsselverzeichnisse erstellt.

Danach wird die Funktion `Credentials` aufgerufen. Diese wiederum ruft die Funktion `generateKeys` auf. Dort wird gefragt ob das Schlüsselpaar bereits existiert. Falls nicht wird eins erstellt. Danach wird der User nach den mySQL-Credentials gefragt, diese werden in die Dateien geschrieben und dann verschlüsselt.

In der Letzen Funktion wird via ein externes File noch der Cronjob für das Backup registriert.