DAT151 – Oblig7

Backup and recovery

Adrian Mortensen & Kamil Sosna

Contents

Task 1: Backup 2
Script: 2
Crontab: 2
Results: 2
Task 2: Recovery 3
Setup 3
From dump 4
Binary logs 4
Results 4
Task 3: Replication 5
Configuration 5
Setup 5
Server1 5
Server2 6
Testing 6

Task 1: Backup

Script:

```
#!/bin/bash

#Faste variabler
BACKUPDIR="/extra/backup/DB"
LOGDIR="/var/lib/mysql"
NOW=$( date '+%F_%H:%M:%S' )
TIMESTAMP=$( date '+%s')


mkdir $BACKUPDIR/$TIMESTAMP
#Backup av databaser
    # Finner liste av databaser som kan kjøre vanlig prosedyre.
    # Bruker brukeren backupAdmin som har rettigheter til å gjøre backup.
set -- $(mysql -u backupAdmin --skip-column-names -e "SHOW DATABASES WHERE
\`Database\` NOT IN ('mysql','information_schema','performance_schema')")
for db
do
    mysqldump --user backupAdmin --master-data=2 --single-transaction -F --
databases $db > $BACKUPDIR/$TIMESTAMP/$db'_'$NOW.sql
done
    # Egen backup av mysql databasen
    mysqldump --user backupAdmin --single-transaction -F mysql --databases
> $BACKUPDIR/$TIMESTAMP/'mysql_'$NOW.sql

#Binær log backup:
    #Flytter binærloggene
mv $LOGDIR/*-bin.0* $BACKUPDIR/$TIMESTAMP/


# lager tar.gz arkiv av backup for å spare plass.
tar -czvf $BACKUPDIR/$TIMESTAMP'_Backup_'$NOW.tar.gz $BACKUPDIR/$TIMESTAMP
# Sletter ukomprimert mappe
rm -r $BACKUPDIR/$TIMESTAMP
exit 0
```

Crontab:

```
$ sudo EDITOR=nano crontab -e
no crontab for root - using an empty one

0 14 * * * /home/admo/git/DAT151/Oblig7/Scripts/runBackup.sh
```

Adding a crontab for running the backup at 14:00 every day. (the time 14:00 is for testing purposes because it was added around 13:50.

Got this file:

```
Backup: 1582570801_Backup_2020-02-24_14:00:01.tar.gz
```

Results:

The result is archive files that contains the sql dumps of each database, a with timestamp both unix timestamp and human readable time for backup.

Task 2: Recovery

Setup

Following the task:

1. Take backup of database where table Passing has 100000 rows
   a. Done with the script
2. Insert 1000 rows into Passing
   a. Code for insertion:

```
MariaDB [Oblig5]> INSERT INTO Passing SELECT DISTINCT regno,now(),3
FROM Car LIMIT 1000;
Query OK, 1000 rows affected (0.019 sec)
Records: 1000  Duplicates: 0  Warnings: 0
```

3. Delete table Passing
   - Code for deletion

```
MariaDB [Oblig5]> DROP TABLE Passing;
```

4. Create empty table Passing
   - Code for table creation

```
CREATE TABLE IF NOT EXISTS  Passing  (
   regno  CHAR(7) NOT NULL,
   timestamp  TIMESTAMP NOT NULL,
   tollstation  SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY ( regno ,  timestamp ),
  CONSTRAINT  fk_Passing_1
    FOREIGN KEY ( tollstation )
    REFERENCES  Tollstation  ( idTollstation )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT  fk_Passing_Subscription1
    FOREIGN KEY ( regno )
    REFERENCES Car (regno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

5. Insert 100 rows into Passing
   - Code for adding 100 rows

```
MariaDB [Oblig5]> INSERT INTO Passing SELECT DISTINCT
regno,now(),3 FROM Car LIMIT 100;
Query OK, 100 rows affected (0.016 sec)
Records: 100  Duplicates: 0  Warnings: 0
```

From dump

First we get the latest backup, and extract it.

Then after a quick inspection of the file we determine the insert values are between the lines 695 and 890 in the backup sql dump.

By writing a small command we extract only the relevant sql lines into their own sql file.

```
sudo sed -n -e "695,890p" "/extra/backup/DB/1582571328_Backup_2020-02-
24_14:08:48/extra/backup/DB/1582571328/Oblig5_2020-02-24_14:08:48.sql" >
/home/admo/git/DAT151/recovered.sql
```

Then I source these commands in the sql database

SOURCE /home/admo/git/DAT151/recovered.sql

Binary logs

During our testing we didn't get the binary logs to show any of the changes that was done to the database. From searching we couldn't find any relevant settings except adding

```
log_bin = mysql-bin
```

to the mariadb config file. This at least enabled binary logs. But from our testing it still does not log the changes.

We will still explain how I would proceed if I would be able to see the changes in the file.

1. Flush the binary logs

   ```
   MariaDB [Oblig5]> FLUSH LOGS;
   ```

2. Find the most relevant log

   ```
   MariaDB [Oblig5]> SHOW MASTER STATUS;
   ```

3. Use mysqlbinlog to look through the log and find the last commit before dropping of the table
   a. This position will then be noted for later
   b. Command
      ```
      $sudo mysqlbinlog /var/lib/mysql/mysql-bin.000001
      ```
4. Pipe the mysqlbinlog with a stop position into mysql.
   a. Command

```
mysqlbinlog --stop-position=TheFoundPosition /var/lib/mysql/mysql-bin.000001 |
mysql -u Adrian -p Oblig5
```

Results

From the results of our testing the script works and pulls all the generated binary files. Aswell as the sql dumps for each database. Where the biggest difficulty in the dump was to find the relevant data to re-insert. This could maybe have been easier if we had done a dump for each table and rather grouped the databases by themselves. Or if instead of looking for the relevant inserts we had just run the entire dump.

Task 3: Replication

Configuration

In this assignment 2 servers (on separate networks (but with correct routing)) replicate a database between them(MASTER – MASTER). SERVER1 will be on ip address 10.0.0.6 and SERVER2 on 192.168.88.241.

On both servers there was some configuration work that had to be done.

In the mariadb-server.conf file

```
log-bin

server_id=$SERVER-NR

replicate-do-db=replicate

bind-address=$HOST-IP
```

was added.

On both servers we created a master-user and a database “replicated”.

Setup

Server1

```
#Server1 - IP 10.0.0.6

MariaDB [replicated]> GRANT REPLICATION SLAVE ON *.* TO 'masterAdmo'@'%';
Query OK, 0 rows affected (0.001 sec)

MariaDB [replicated]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.002 sec)

MariaDB [replicated]> SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mysqld-bin.000001  |     1240 |              |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.001 sec)

MariaDB [replicated]>
```

Here we give a master-user permissions and check the binary logs for replication and a point in time to replicate from. The file name and Current Position in the file will be used on Server2

Server2

```
#Server2 - IP 192.168.88.241

MariaDB [replicated]> GRANT REPLICATION SLAVE ON *.* TO 'masterAdmo'@'%';
Query OK, 0 rows affected (0.001 sec)

MariaDB [replicated]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.002 sec)

MariaDB [replicated]> STOP SLAVE;
CHANGE MASTER TO MASTER_HOST='10.0.0.6',
MASTER_USER='masterAdmo',MASTER_PASSWORD='master_password',MASTER_LOG_FILE=
'mysqld-bin.000001',MASTER_LOG_POS=1240;

MariaDB [replicated]> START SLAVE;

MariaDB [replicated]> SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mysqld-bin.000001  |     1075 |              |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.001 sec)

# Going back to finish up on Server1
```

Here we see the information gathered from SERVER1 used while setting up the MASTER_HOST

Also generating the info for SERVER2 for replication.

```
ariaDB [replicated]> STOP SLAVE;

CHANGE MASTER TO MASTER_HOST='192.168.88.241',
MASTER_USER='masterAdmo',MASTER_PASSWORD='master_password',MASTER_LOG_FILE=
'mysqld-bin.000001',MASTER_LOG_POS=1075;


MariaDB [replicated]> START SLAVE;
```

Here we use the SERVER2 info to setup the same thing on SERVER1.

Testing

```
#TESTING (Done on SERVER2)
MariaDB [replicated]> CREATE TABLE sample (`name` varchar(18));


#Run on SERVER1
MariaDB [replicated]> SHOW TABLES IN replicated;
+-----------------------+
| Tables_in_replicated |
+-----------------------+
| sample                |
+-----------------------+
1 row in set (0.001 sec)
```