

Restore MySQL Master-Master Replication

Introduction

This page describes the process of rebuilding a Master-Master DB & Replication. Since the MySQL Dump command locks the tables, there is no need to create it when there is no traffic on the machine. It can be done during operational hours. With --master-data the command mysqldump stores the correct position for inserting the replication on the slave server.

Procedure

Determine Good Master

In a system with Master-Master replication, one of the masters is active and processing queries from the other jtel cluster members. This is the server from which the MySQL Dump is taken, since this master has the latest data and is up to date.

Further down in this page, the good master-database and server will be referred to as the



GOOD MASTER

GOOD SLAVE

Further down in this page, the broken server will be referred to as the

BROKEN MASTER

BROKEN SLAVE

Haproxy configuration

If there is a HAPROXY, then remove the servers on the **BROKEN MASTER** side from the distribution (also the **BROKEN SLAVE** on this side).

STOP SLAVE


Login to MySQL on both the **GOOD MASTER** and the **BROKEN MASTER** to stop the slave SQL. Leave MySQL again afterwards. Use the following commands for this:

```
mysql -uUSER -pPWD
STOP SLAVE;
QUIT;
```


Phase 1 - MySQL Dump

A MySQL Dump of the **GOOD MASTER** is now created. Perform the following steps to create a MySQL Dump and save it to the STORE:

mysqldump command

 The mysqldump command is different, depending on the jtel portal release, as well as the MySQL software release installed on the databases. All different options and how to find out which one to choose is specified below.

Master-Master Replication

 The mysqldump commands on this page can NOT be used to realign a master-slave replication. Visit the following page for that description [Role DATA - Simple Master / Slave](#)

jtel Portal software release

Log in to the Load Balancer of the cluster and execute the following commands as the jtel user


```
# Find out which software release is installed
cd /srv/jtel/shared/JTELCarrierPortal
git status

# If /srv/jtel/.. does not exist on the load balancer, attempt this
cd /home/jtel/shared/JTELCarrierPortal
git status

# Expected output
release-stable/3.XX
```

Create Backup Directory


The following commands are designed to be executed on the load balancer as jtel user

 The following cd commands depend on the variable JT_DATE_TIME, which is set at the beginning of the next part. If the variable is not set, commands will fail.

```
# Create backup directory SLAVE MySQL Dump
JT_DATE_TIME=$(date +%F)
mkdir /srv/jtel/shared/backup/${JT_DATE_TIME}
# If /srv/jtel/.. does not exist on the load balancer, attempt this
JT_DATE_TIME=$(date +%F)
mkdir /home/jtel/shared/backup/${JT_DATE_TIME}
```

Create MySQL Dump


CAUTION - CREDENTIALS+IP-Adresses

 Credentials and IP-Adresses need to be changed before the following mysqldump commands can be executed
The following commands are designed to be executed on the load balancer as jtel user

MySQL Dump - Until jtel Portal release 3.12

```
# Change to backup directory
cd /srv/jtel/shared/backup/${JT_DATE_TIME}
# Create MySQL Dump
mysqldump -uUSER -pPWD -h<IP-Address-OR-Alias-GOOD-MASTER> --single-transaction --master-data=2 --databases JTELWeb JTELStats JTELLog --add-drop-database --add-drop-table --events --routines --triggers > /srv/jtel/shared/backup/${JT_DATE_TIME}/acd-dbm_${JT_DATE_TIME}.sql
```

In versions 3.12, 3.14 and 3.15

 If someone logs on to the portal while the dump is being pulled, it will go wrong. Enclosed a SQL query. If the time changes after executing the query, a login has taken place.

If this happens, the dump has to be pulled again and in the meantime it has to be permanently checked if a login has taken place. Only if this is not the case, the dump can be replicated error-free to the slave.

```
SELECT Max(dtAcdLoggedIn) FROM Users;
```

In versions 3.11 and below and version 3.16 this problem does not exist.

MySQL Dump - From jtel Portal release 3.12 until latest release

```
# Change to backup directory
cd /srv/jtel/shared/backup/${JT_DATE_TIME}
# Create MySQL Dump
mysqldump -uUSER -pPWD -h<IP-Address-OR-Alias-GOOD-MASTER> --single-transaction --master-data=2 --databases JTELWeb JTELStats JTELStats2 JTELLog --add-drop-database --add-drop-table --events --routines --triggers > /srv/jtel/shared/backup/${JT_DATE_TIME}/acd-dbm_${JT_DATE_TIME}.sql
```

MySQL Dump - From Release 3.32

```
# change to jtel user
su jtel
# Change to backup directory
cd /srv/jtel/shared/backup/${JT_DATE_TIME}
# Create MySQL Dump
mysqldump -uUSER -pPWD -h<IP-Address-OR-Alias-GOOD-MASTER> --single-transaction --master-data=1 --databases JTELWeb JTELStats JTELStats2 JTELLog JTELCustomer --add-drop-database --add-drop-table --events --routines --triggers > /srv/jtel/shared/backup/${JT_DATE_TIME}/acd-dbm_<yyyymmdd>.sql
```

MySQL Dump - From MySQL 8.0.27

To Check for MySQL Version, log in to the master database server and execute the following command

```
mysql --version
```

```
# Change to backup directory
cd /srv/jtel/shared/backup/${JT_DATE_TIME}
# Create MySQL Dump
mysqldump -uUSER -pPWD -h<IP-Address-OR-Alias-GOOD-MASTER> --single-transaction --source-data --databases JTELWeb JTELStats JTELStats2 JTELLog --add-drop-database
--add-drop-table --events --routines --triggers > /srv/jtel/shared/backup/${JT_DATE_TIME}/acd-dbm_${JT_DATE_TIME}.sql
```

Brief Explanation of the mysqldump command

- *<Name LogFile>* and *<Position LogFile>* are saved through *--master-data* or *--source-data* in the dump
- use only the JTEL databases '*--databases*'
- delete all databases before import '*--add-drop-database*'
- delete all tables of the databases before import '*--add-drop-table*'
- Implement all routines and procedures in the dump '*--events --routines*'
- Additionally the parameter *--default_character_set utf8* can be used

Phase 2 - Import Dump on **BROKEN MASTER**

The following commands are designed to be run from the Load Balancer as root user. After the source *<acd-dbm_<yyyymmdd>.sql>* is executed, the dump will start importing into the master database. Once it is finished, the Slave SQL is started on the BROKEN MASTER, and the replication parameters are updated on the GOOD MASTER to realign the master-master replication.

Brief explanation - tmux: tmux is a tool used to create sub-sessions within a ssh terminal. The tool is a good option for this specific operation, since the sub-session can be exited but the processes which were started will still be running in the background. This prevents failure by accidental loss of the ssh terminal, and provides the ability to view the MySQL process list to check the progress of the import

Tmux cheat sheet: <https://tmuxcheatsheet.com/>

Import Dump - tmux

```
# switch to root user
su root
# start tmux sub-session
tmux new
# change to the directory where the dump file is located
cd /srv/jtel/shared/backup/${JT_DATE_TIME}/
# copy the name
ls -als
# log in to mysql and import the dump
mysql -uroot -p<password>
source <acd-dbm_<yyyymmdd>.sql>
```

Import Dump - No tmux

```
# switch to root user
su root
# change to the directory where the dump file is located
cd /srv/jtel/shared/backup/${JT_DATE_TIME}/
# copy the name
ls -als
# log in to mysql and import the dump
mysql -uroot -p<password>
source <acd-dbm_<yyyymmdd>.sql>
```

Update Replication Parameters on **BROKEN MASTER**

These commands are executed on the **BROKEN MASTER**

Determine the replication parameters from the **GOOD MASTER** and reinitialize the **BROKEN MASTER**

```
# Determine replication parameters (Write down the parameters after taking the information out of the dump file) The information is contained in a line starting
with "CHANGE MASTER TO MASTER_HOST"
cd /srv/jtel/shared/backup/${JT_DATE_TIME}/
less <acd-dbm_<yyyymmdd>.sql>
# Initialize and start slave SQL on BROKEN MASTER
mysql -uUSER -pPWD
CHANGE MASTER TO MASTER_HOST = '<IP-Address-OR-Alias-GOOD-MASTER>', MASTER_USER = 'repl', MASTER_PASSWORD = '<PASSWORD>', MASTER_LOG_FILE='<NAME_LOGFILE>',
MASTER_LOG_POS=<POSITION_LOGFILE>;
START SLAVE;
```

Check the slave on the **BROKEN MASTER**

```
SHOW SLAVE STATUS\G
```

Only if everything is OK, and the replication is up to date, then continue.

The status can be monitored with the following command:

```
watch 'mysql -u root -p<PASSWORD> -e "SHOW SLAVE STATUS\G" 2>/dev/null'
```

Lock Tables on **BROKEN MASTER**

```
FLUSH TABLES WITH READ LOCK;
# Determine replication parameters (Write down the parameters after taking the information out of the dump file) The information is contained in a line starting
with "CHANGE MASTER TO MASTER_HOST"
SHOW MASTER STATUS;
```

On the **GOOD MASTER**, reposition and start the replication.

```
# Initialize and start slave SQL on BROKEN MASTER
mysql -uUSER -pPWD
CHANGE MASTER TO MASTER_HOST = '<IP-Address-OR-Alias-BROKEN-MASTER>', MASTER_USER = 'repl', MASTER_PASSWORD = '<PASSWORD>', MASTER_LOG_FILE='<NAME_LOGFILE>',
MASTER_LOG_POS=<POSITION_LOGFILE>;
START SLAVE;
```

Unlock the tables on the **BROKEN** master server

```
UNLOCK TABLES;
```

Final Checks - All Servers

Check the **GOOD MASTER**, THE **GOOD SLAVE**, THE **BROKEN MASTER** AND THE **BROKEN SLAVE**

It is usually not necessary to restore the slaves attached to both masters. If it is, they can be re-initialized with the normal slave recovery procedure.

```
SHOW SLAVE STATUS\G
```

Final Checks - Haproxy configuration

If there is a HAPROXY, then add the servers on the **BROKEN MASTER** side from the distribution (also the **BROKEN SLAVE** on this side).