Migrating MySQL to YugabyteDB Using ysql_loader Workshop

V1.0 Original Author: Eric Pratt, Taylor Mull, Stanley Sung
V1.1 Mike Lee

We understand that database migrations can be painful. We have helped users successfully migrate from MySQL to YugabyteDB, a PostgreSQL-compatible distributed SQL database. A very popular tool to accomplish this task is pgloader. In this post, we will cover how to migrate both your MySQL schema and data to YugabyteDB.

Prerequisites

Before starting the migration there are a few prerequisites you'll need to address for the Workshop.

Source Database:
You will need a “source” database to access. Supported databases are: MySQL, MS SQL, SQL Lite, or PostgreSQL. For this workshop we will be migrating from a MySQL database.

To get started creating a MySQL database use this link:
https://dev.mysql.com/doc/mysql-getting-started/en/

Yugabyte Database:
You can quickly install and create a single node Yugabyte cluster here:
https://docs.yugabyte.com/latest/quick-start/

Ysqlsh command line tool:
Download and install `ysqlsh` command line tool with connectivity to a running YugabyteDB cluster that you are going to migrate into.

Create a database on the YugabyteDB cluster that you will migrate into from the source cluster. This database needs to match the database name from MySQL. There would not be any tables or data behind the database but pgloader requires the database to exist on the target cluster. You can create the database by following these steps:

```
# Use ysqlsh to connect to any node of the YugabyteDB cluster
ysqlsh --host=<ip>
```

```
# Create the database
create database <name>;
```

Preparing to Migrate

For the workshop, we will install the ysql_loader on the same server as MySQL.

Verify Connectivity to Source MySQL DB
Next, ensure that the migration machine is able to connect to the MySQL server, because pgloader will need to communicate with MySQL over port 3306. Also make sure that the username/password used by pgloader and the IP address from which pgloader is connecting has permissions to connect to the MySQL source database.

1. To verify network connectivity you can use `telnet <MySQL_ip> 3306`

2. To grant the pgloader ip the permissions to access the database you can run the following command.

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'<pgloader_instance_ip>' WITH
GRANT OPTION;

flush PRIVILEGES;
```

3. To add a password for the ip/user combination, use the following command:

```
SET PASSWORD FOR 'root'@'<pgloader_instance_ip>' =
PASSWORD('<password>');
```

If you see a "*failed to connect*" message like this after doing the above steps check the access with your MySQL DBA:

```
2021-04-22T17:33:33.232901Z ERROR #1=mysql: Failed to connect to
#1# at "172.161.20.87" (port 3306) as user "root": MySQL Error
[1045]: "Access denied for user
'root'@'ip-172-161-27-195.us-east-2.compute.internal' (using
password: YES)"
2021-04-22T17:33:33.232990Z LOG report summary reset
       table name     errors       rows      bytes      total time
-----------------  ---------  ---------  ---------  --------------
  fetch meta data          0          0                     0.000s
-----------------  ---------  ---------  ---------  --------------
-----------------  ---------  ---------  ---------  --------------
```

Create a test database on MySQL

Verify Connectivity to Target YugabyteDB Cluster

The final check is to make sure the pgloader instance is able to reach the YugabyeDB cluster. For this we'll need to check to make sure the pgloader instance is able to communicate with one of the YugabyteDB nodes across port 5433.

1. To verify connectivity run `telnet <YugabyteDB_node_ip> 5433`

Installing pgloader (command line)

```
$ git clone https://github.com/yugabyte/pgloader
#
# GET ALL LIBRARIES
$ apt-get install sbcl unzip libsqlite3-dev make curl gawk
freetds-dev libzip-dev
$ cd /path/to/pgloader
$ make pgloader
```

```
##purposely renamed, but you could replace pgloader
$ sudo cp build/bin/pgloader /usr/bin/ysql_loader
```

For pgloader command flags:

```
ysql_loader --help
```

```
pgloader [ option ... ] command-file ...
pgloader [ option ... ] SOURCE TARGET
  --help -h                       boolean  Show usage and exit.
  --version -V                    boolean  Displays pgloader version and exit.
  --quiet -q                      boolean  Be quiet
  --verbose -v                    boolean  Be verbose
  --debug -d                      boolean  Display debug level information.
  --client-min-messages           string   Filter logs seen at the console (default:
"warning")
  --log-min-messages              string   Filter logs seen in the logfile (default:
"notice")
  --summary -S                    string   Filename where to copy the summary
  --root-dir -D                   string   Output root directory. (default:
#P"/tmp/pgloader/")
  --upgrade-config -U             boolean  Output the command(s) corresponding to
.conf file for v2.x
  --list-encodings -E             boolean  List pgloader known encodings and exit.
  --logfile -L                    string   Filename where to send the logs.
  --load-lisp-file -l             string   Read user code from files
  --dry-run                       boolean  Only check database connections, don't
load anything.
  --on-error-stop                 boolean  Refrain from handling errors properly.
  --no-ssl-cert-verification      boolean  Instruct OpenSSL to bypass verifying
certificates.
  --context -C                    string   Command Context Variables
  --with                          string   Load options
  --set                           string   PostgreSQL options
  --field                         string   Source file fields specification
  --cast                          string   Specific cast rules
  --type                          string   Force input source type
  --encoding                      string   Source expected encoding
  --before                        string   SQL script to run before loading the data
  --after                         string   SQL script to run after loading the data
  --self-upgrade                  string   Path to pgloader newer sources
  --regress                       boolean  Drive regression testing
```

Setting Up pgloader Command File

Instead of using the pgloader command line SOURCE TARGET with options, we use a command file to simplify the docker command.

```
load database
```

```
 from mysql://root:password@<IP>:3306/testdb
 into postgresql://yugabyte:yugabyte@<IP>:5433/testdb
 WITH
    max parallel create index=1, batch rows = 1000;
```

Running pgloader

pgloader Schema and Data Migration

The docker image is based on centos7. We can “bash” into the container and run the pgloader command (/usr/local/bin/pgloader). In this doc, we store the pgloader command-file in “centos” home directory and use docker volume to map the configuration from centos home directory to docker container directory. The docker flag “--rm” will make sure the container is removed once the pgloader job is done.

```
docker run --rm --name <name_for_container> \
         -v <local_dir pgloader_config_dir>:<mount_path_in_container> \
         yugabytedb/pgloader:v1.1 pgloader \
         <mount_path_in_container>/<pgloader_config_file>

example:
sudo docker run --rm --name pgloader \
-v /home/ubuntu/Mlee/PgloaderDSS:/tmp \
ybloader:v1.2 pgloader \
/tmp/mysql_yb.cmd
```

Ex. pgloader command

```
[root@ip-172-161-27-195 centos]# pwd
/home/centos
[root@ip-172-161-27-195 centos]# ls
pgloader.conf
[root@ip-172-161-27-195 centos]# docker run --rm --name pgloader1
-v /home/centos:/tmp yugabytedb/pgloader:v1.1 pgloader -v -L
/tmp/pgloader.log /tmp/pgloader.conf
```

We should get the below output when the pgloader is running.

```
2021-04-22T18:49:00.000672Z LOG pgloader version "3.6.3~devel"
2021-04-22T18:49:00.264485Z LOG Migrating from #<MYSQL-CONNECTION
mysql://root@172.161.20.87:3306/testdb #x302001D3B50D>
```

```
2021-04-22T18:49:00.264662Z LOG Migrating into #<PGSQL-CONNECTION
pgsql://yugabyte@172.161.20.43:5433/testdb #x302001D3B3AD>
```

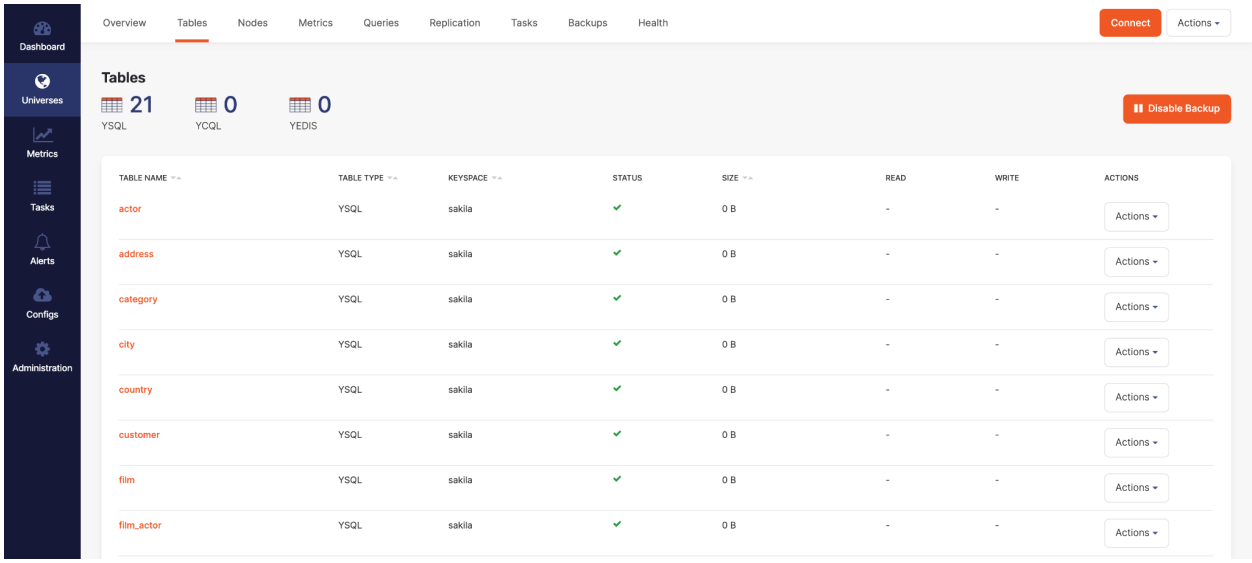
We can also double check the pgloader is running by running docker ps

```
[centos@ip-172-161-27-195 ~]$ sudo docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS
NAMES
2dc611d4d412   yugabytedb/pgloader:v1.1   "pgloader /tmp/pgloa…"   16 minutes ago   Up 16 minutes
pgloader1
```

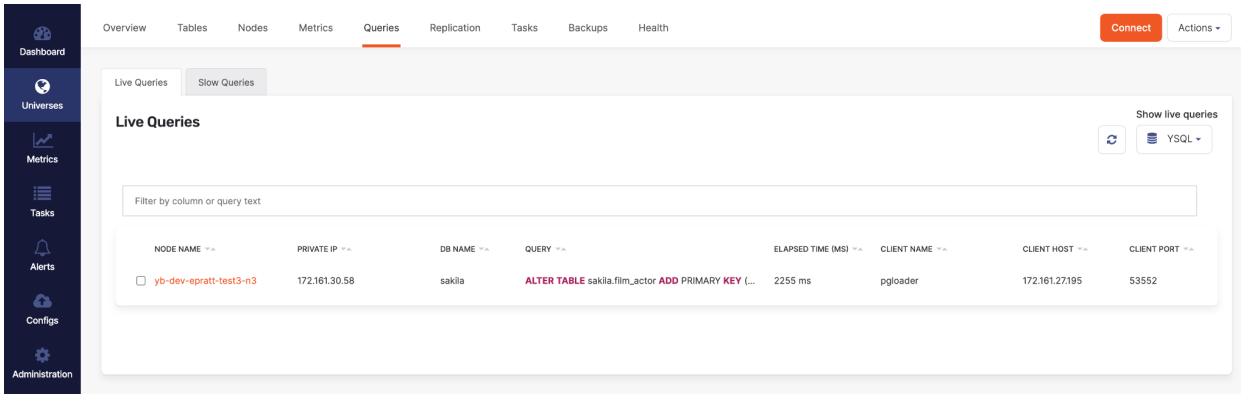
We can also tail the pgloader.log we specified in the docker command.

```
[centos@ip-172-161-27-195 ~]$tail -f pgloader.log
```

We can check where the pgloader is currently at by going to the platform UI and checking the tables section to see the tables start to load in.



In addition you can check the live queries and see current queries/ddl changes being made on the cluster at that time.



pgloader Schema Migration Only

If you only wanted to migrate the schema from MySQL to YugabyteDB and not include the data you can add `WITH schema only` to the pgloader command file and pgloader will only load the schema.

```
load database
 from mysql://root:password@172.161.20.87:3306/testdb
 into postgresql://yugabyte:yugabyte@172.161.30.169:5433/testdb
 WITH
    max parallel create index=1, batch rows = 1000, schema only;
```

Validation

Once pgloader finishes the migration, you will get a summary of the migration steps which includes how long each step took and the number of rows inserted.

```
2021-09-21T19:42:05.153000Z LOG report summary reset
                  table name     errors       read   imported      bytes      total time       read      write
-------------------------- ---------  ---------  ---------  ---------  --------------  ---------  ---------
                fetch meta data          0          4          4                   0.105s
              Drop Foreign Keys          0          0          0                   0.000s
                       Truncate          0          4          4                   0.535s
-------------------------- ---------  ---------  ---------  ---------  --------------  ---------  ---------
ml_migratedb.ml_order_line          0     800000     800000    56.9 MB         59.317s    49.590s    56.979s
     ml_migratedb.ml_orders          0      80000      80000     3.5 MB          6.410s     0.803s     5.392s
   ml_migratedb.ml_customer          0      30000      30000    16.2 MB         11.446s     1.169s    10.882s
   ml_migratedb.ml_tinyint1          0        201        201     4.3 kB          0.033s     0.002s     0.002s
-------------------------- ---------  ---------  ---------  ---------  --------------  ---------  ---------
    COPY Threads Completion          0          4          4                  59.306s
                Reset Sequences          0          0          0                   1.863s
          Create Foreign Keys          0          0          0                   0.000s
             Install Comments          0          1          1                   0.022s
-------------------------- ---------  ---------  ---------  ---------  --------------  ---------  ---------
           Total import time          ✓     910201     910201    76.7 MB        1m1.191s
```

Alternatively you can also check the platform to make sure all the tables are present by looking under the tables tab and making sure there are no active queries migration queries against the cluster (index creation, index backfill, copy from, create table, etc.)

Using ysqlsh -h <IP> <dbname> you can check out the tables that have been migrated.

```
ysqlsh (11.2-YB-2.7.2.0-b0)
Type "help" for help.

ml_migratedb=# \d
                    List of relations
    Schema     |          Name          |   Type   |  Owner
---------------+------------------------+----------+----------
 ml_migratedb | ml_customer            | table    | yugabyte
 ml_migratedb | ml_order_line          | table    | yugabyte
 ml_migratedb | ml_order_line_id_seq   | sequence | yugabyte
 ml_migratedb | ml_orders              | table    | yugabyte
 ml_migratedb | ml_orders_id_seq       | sequence | yugabyte
 ml_migratedb | ml_tinyint1            | table    | yugabyte
 ml_migratedb | ml_tinyint1_id_seq     | sequence | yugabyte
 ml_migratedb | orders_orig            | table    | yugabyte
(8 rows)

ml_migratedb=#
```

Conclusion

The ability to migrate seamlessly to a new database helps to take the pressure off moving to a new technology. With the changes we've made to pgloader at Yugabyte to enable users to easily move your current MySQL database onto YugabyteDB, we relieve that headache. As always we are here to help and answer any questions you may have. Join us on our community Slack channel, and star us on GitHub.