Migrating MySQL to Yugabyte using ysql_loader

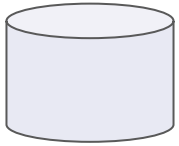
Mike Lee - Solutions Architect
Suranjan Kumar - Ecosystem Integration Engineer
Yugabyte

Agenda

- Pre-work
- What is ysql_loader?
- Where can I Git it?
- How to use it?
  - configuration options
- How to run it!

Workshop Configuration

On GCP

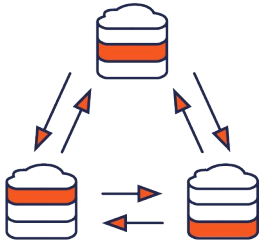


MySQL, MS SQL,
SQL Lite, Postgres

MacOS/VM

ysql_loader

Yugabyte Cloud



For production migrations, ysql_loader should be installed on a separate server.
For this workshop, ysql_loader will be installed on Mac

Prep for Workshop

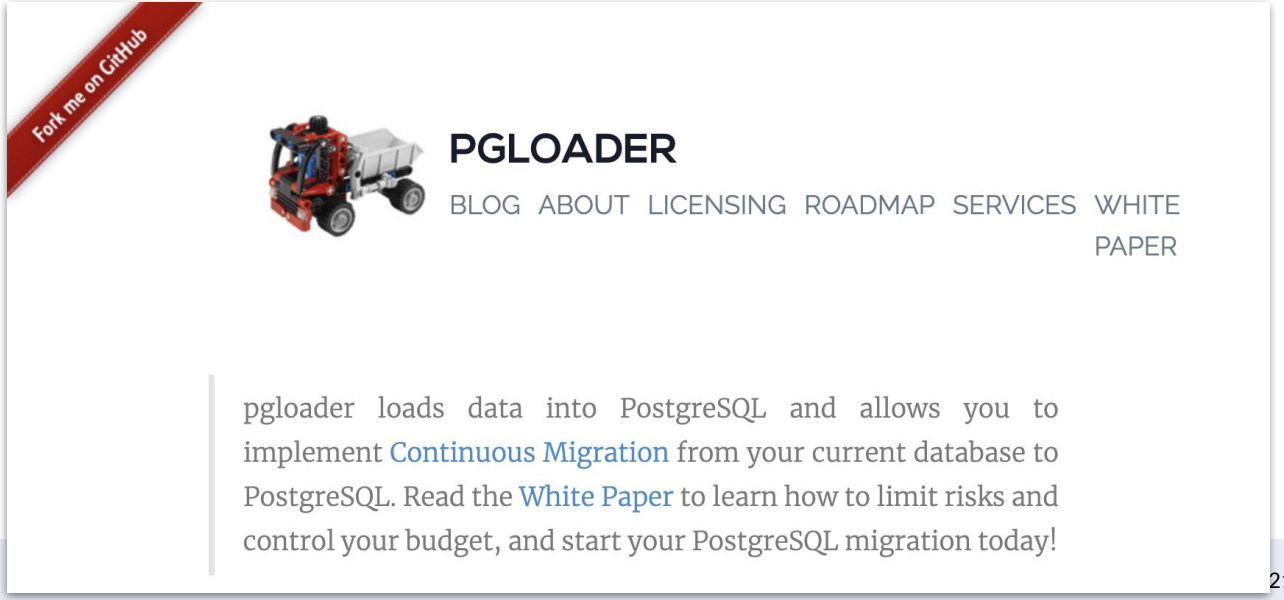
On your Mac/VM: git clone https://github.com/yugabyte/yugabyte-ysql-loader-workshop

Review Pre-Work PDF

- Create ybCloud instance
- Test connection to MySQL database
- Git ysql_loader

What is ysql_loader

- ysql_loader = pgloader (pgloader.io)
- Dimitri Fontaine: wrote and maintains of pgloader, Major Contributor to PostgreSQL, author of The Art of PostgreSQL
  - Open Source
  - ysql_loader forked from pgloader (Suranjan)



ysql_loader

- Migrates database objects (tables, indexes, sequences) and loads data from files (CSV, Fixed, DBF, IXF) or directly connecting to source databases such as MySQL, SQL Lite, MS SQL and PostgreSQL

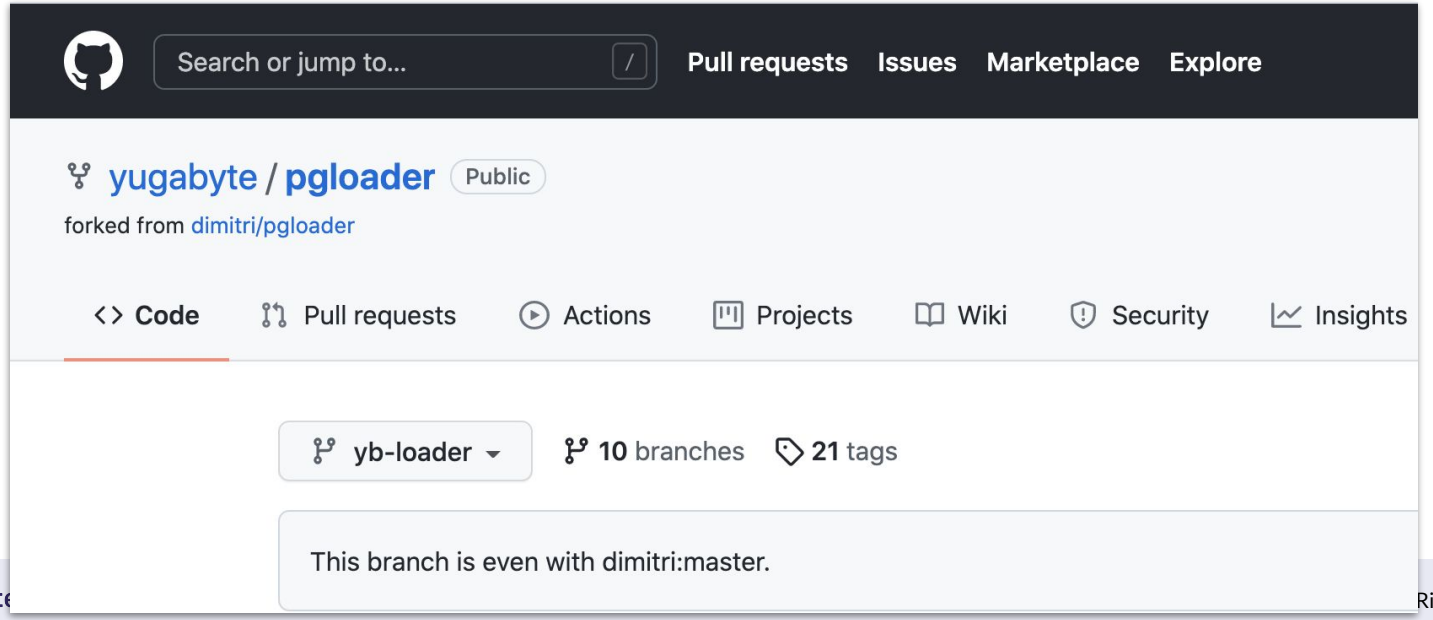
- Can load from flat files (CSV, Fixed Format) or directly connect and migrate entire databases to

- Follows a typical database migration workflow:
  - Create the target database on Yugabyte
  - Gather database objects from source database metadata catalog
  - Create Tables and Indexes
  - Copy table data using PostgreSQL COPY
  - Add the constraints, primary & foreign keys, and comments

- Has a rich set of options that allow you to customize ysql_loader to your exact needs.

Pgloader - how to Git it...

$git clone https://github.com/yugabyte/pgloader
$ cd pgloader
$ git checkout ysql_loader_dumpddl



pgloader - build it and they will migrate....

```
# on Mac/VM
$ 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
$ git checkout ysql_loader_dumpddl
$ make pgloader

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

pgloader(Docker) - build it and they will migrate

```
$ git clone https://github.com/yugabyte/pgloader
$ git checkout ysql_loader_dumpddl

$ cd /path/to/pgloader

$ sudo docker -t ysql-loader:v1.3 build .
```

pgloader command-file options

Configuration Sections

- **Database Source: FROM**
- **Migration Options: WITH**
- **Casting Rules: CAST**
- **Partial Migration: INCLUDING/EXCLUDING NAMES**
- Encoding Support
- Schema Transformations
- View Support

Check out: https://pgloader.readthedocs.io/en/latest/index.html

Pgloader command options - FROM

```
LOAD DATABASE
    FROM mysql://root:<password>@IP.Addr:3306/ml_migratedb
    INTO postgresql://yugabyte:<password>@IP.Addr:5433/ml_migratedb;
```

Determines the source and targets

Pre-migration checks: test the login and password and IP to make sure you can access both the source database and YugabyteDB from the wherever ysql_loader was installed.

Checkpoint

Have you created a ybCloud cluster?

Have you download mysqlsh and tested connectivity?

Have you git cloned yugabyte/ysql_loader and started a build?

If everything is ready…

1) Open config1.load, modify the hostname string on the postgresql line to be your ybCloud hostname
2) run pgloader using config1.load

Hint: ../pgloader/build/bin/pgloader --verbose config1.load

Pgloader command options - WITH

```
LOAD DATABASE
    FROM mysql://root:<password>@IP.Ad.dre.ss:3306/ml_migratedb
    INTO postgresql://yugabyte:<password>@IP.Ad.dres.ss:5433/ml_migratedb
WITH batch rows = 500, truncate
--BATCH ROWS best practice: 200-300.  Do not exceed 1000.
;
```

Default WITH options with MySQL:

- no truncate
- create tables
- include drop
- create indexes
- reset sequences
- foreign keys
- **downcase identifiers** - be mindful of tables with same name but differ by capitalization.
- **uniquify index names** - PG index names have to be unique per-schema (MySQL is per-table)

Pgloader command options - INCLUDING/EXCLUDING

```
LOAD DATABASE
    FROM mysql://root:<password>@IP.Ad.dres.ss:3306/ml_migratedb
    INTO postgresql://yugabyte:<password>@IP.Ad.dres.ss:5433/ml_migratedb
WITH batch rows = 1000, truncate
INCLUDING ONLY TABLE NAMES  MATCHING ~/ml_/, 'orders_orig'
EXCLUDING TABLE NAMES MATCHING ~<orig>
;
```

INCLUDING: comma separated list of table names or regular expression used to limit the tables.

EXCLUDING: comma separated list of table names or regular expressions used to limit the tables, HOWEVER, This filter only affects the result of the INCLUDING filter.

Pgloader command options - CAST

```
LOAD DATABASE
    FROM mysql://root:<password>@IP.Ad.dres.ss:3306/ml_migratedb
    INTO postgresql://yugabyte:<password>@IP.Ad.dres.ss:5433/ml_migratedb
WITH batch rows = 1000, truncate
INCLUDING ONLY TABLE NAMES  MATCHING ~/ml_/, 'orders_orig'
EXCLUDING TABLE NAMES MATCHING ~<orig>
CAST
  type tinyint to smallint drop typemod
;
```

MySQL TINYINT(1) supports values 0,1,2,3

PostgreSQL has BOOLEAN, but it only supports

- TRUE, 't', 'true', 'y', 'yes', 'on', '1' or FALSE, 'f', 'false', 'n', 'no', 'off', '0'
- To retain values 2 and 3, you must convert the data type to a SMALLINT.

CAST does this automatically for you!

ysql_loader command options - CAST

MYSQL

```
mysql> show full columns from ml_tinyint1;
+--------+-------------------------+-------------------+------+-----+---------+----------------
| Field  | Type                    | Collation         | Null | Key | Default | Extra
+--------+-------------------------+-------------------+------+-----+---------+----------------
| id     | int(10) unsigned zerofill | NULL              | NO   | PRI | NULL    | auto_increment
| ti_col | tinyint(1)              | NULL              | YES  |     | NULL    |
| name   | varchar(20)             | latin1_swedish_ci | NO   |     | NULL    |
+--------+-------------------------+-------------------+------+-----+---------+----------------
```

YugabyteDB

Converted to SMALLINT

```
ml_migratedb=# \d+ ml_tinyint1
                                                                    Table "ml_migratedb.ml_tinyint1"
 Column |          Type          | Collation | Nullable |                  Default                  |
--------+------------------------+-----------+----------+-------------------------------------------+
 id     | bigint                 |           | not null | nextval('ml_tinyint1_id_seq'::regclass) |
 ti_col | smallint               |           |          |                                           |
 name   | character varying(20)  |           | not null |                                           |
Indexes:
    "ml_tinyint1_pkey" PRIMARY KEY, lsm (id HASH)
```

Data type considerations

Data Type	MySQL	PostgreSQL
Integers(bytes)	TINYINT(1), SMALLINT(2), MEDIUMINT(3), INT(4), BIGINT(8)	SMALLINT(2), INTEGER(4), BIGINT(8)
Text	TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT	TEXT
Number	DECIMAL, NUMERIC	DECIMAL, NUMERIC
Double	DOUBLE	DOUBLE

ysql_loader command options - RESET SEQUENCES

MySQL

```
mysql> select max(id) from ml_order_line;
+---------+
| max(id) |
+---------+
|  924280 |
+---------+
```

MySQL: ml_order_line table has a sequence called **ml_order_line_id_seq**

Yugabyte

```
ml_migratedb=# select nextval('ml_order_line_id_seq');
 nextval
---------
  924281
(1 row)
```

In ybdb: the max value is preserved and next value ready to use.

Let’s run this!

Run pgloader using config2.load and config3.load

Config2.load - shows including/excluding functionality
Config3.load - shows CAST capabilities (tinyint to smallint)

Migrating MySQL to Yugabyte best practices

The workflow is similar, but we suggest the following extra steps:

- Use: DUMPDDL ONLY option
  - Provide a way to add YugabyteDB constructs and e.g provide #tablets, colocated table, geo partitioned table
- Use the DATA ONLY option to move data after database and tables have been created

Best Practices

- Create your database COLOCATED - `CREATE DATABASE company WITH COLOCATED = true;`
- Use Geo-Partitioning to pin data to a geographic location PARTITION BY LIST (geo_partition)
- Large tables (> 1M rows) should add SPLIT INTO x TABLETS to the CREATE TABLE statement

Please see https://docs.yugabyte.com/latest/

ysql_loader command options - DUMPDDL ONLY

```
LOAD DATABASE
    FROM mysql://root:P8ssw0rd2@10.142.0.2:3306/ml_migratedb
    INTO postgresql://yugabyte:yugabyte@10.204.0.5:5433/ml_migratedb
WITH dumpddl only
including only table names matching ~/ml_/, 'orders_orig'
CAST
  type tinyint to smallint drop typemod
;
```

DDL extracted to file ddl.sql !

```
2021-09-17T22:23:22.396000Z LOG report summary reset
               table name     errors       read   imported       bytes      total time       read      write
-----------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
          fetch meta data          0          6          6                    0.095s
           Create Schemas          0          0          0                    0.007s
         Create SQL Types          0          0          0                    0.007s
            Create tables          0          8          8                   17.277s
           Set Table OIDs          0          4          4                    0.012s
       create primary key          0          3          3                   16.375s
-----------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
-----------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
Index Build Completion          0          2          2                   10.008s
            Create Indexes          0          2          2                    8.731s
           Reset Sequences          0          0          0                    1.076s
              Primary Keys          0          0          0                    0.000s
      Create Foreign Keys          0          0          0                    0.000s
           Create Triggers          0          0          0                    0.001s
         Install Comments          0          1          1                    0.026s
-----------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
        Total import time          ✓          0          0                   19.842s
```

DUMPDDL ONLY - ddl.sql

```
DROP TABLE IF EXISTS ml_migratedb.ml_order_line CASCADE;
CREATE TABLE ml_migratedb.ml_order_line
(
  id            bigserial not null,
  ol_o_id       bigint not null,
  ol_d_id       smallint not null,
  ol_w_id       smallint not null,
  ol_number     smallint not null,
  ol_i_id       bigint,
  ol_supply_w_id smallint,
  ol_delivery_d  timestamptz,
  ol_quantity    smallint,
  ol_amount      decimal(6,2),
  ol_dist_info   char(24)
) SPLIT INTO 24 TABLETS;

ALTER TABLE ml_migratedb.ml_order_line ADD PRIMARY KEY (id, ol_w_id, ol_d_id, ol_o_id, ol_number);
ALTER TABLE ml_migratedb.ml_orders ADD PRIMARY KEY (id, o_w_id, o_d_id, o_id);
ALTER TABLE ml_migratedb.ml_tinyint1 ADD PRIMARY KEY (id);
CREATE UNIQUE INDEX idx_ml_order_line_order_line_i1 ON ml_migratedb.ml_order_line (id);
CREATE UNIQUE INDEX idx_ml_orders_orders_i1 ON ml_migratedb.ml_orders (id);
```

ysql_loader command options - DATA ONLY

```
LOAD DATABASE
    FROM mysql://root:P8ssw0rd2@10.142.0.2:3306/ml_migratedb
    INTO postgresql://yugabyte:yugabyte@10.204.0.5:5433/ml_migratedb
WITH data only
including only table names matching ~/ml_/, 'orders_orig'
CAST
  type tinyint to smallint drop typemod
;
```

```
2021-09-17T22:54:10.801000Z LOG report summary reset
                  table name     errors       read   imported      bytes      total time       read      write
----------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
                fetch meta data         0          4          4                   0.086s
              Drop Foreign Keys         0          0          0                   0.000s
----------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
ml_migratedb.ml_order_line              0     800000     800000    56.9 MB       2m32.413s  2m11.162s  2m29.125s
    ml_migratedb.ml_orders              0      80000      80000     3.5 MB          21.583s     0.632s     17.456s
  ml_migratedb.ml_tinyint1              0        201        201     4.3 kB          21.906s     0.002s
  ml_migratedb.orders_orig              0      30000      30000     1.2 MB           3.024s     0.214s      0.121s
----------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
   COPY Threads Completion              0          4          4                  2m32.395s
              Reset Sequences           0          0          0                   2.063s
          Create Foreign Keys           0          0          0                   0.000s
             Install Comments           0          1          1                   0.025s
----------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
            Total import time           ✓     910201     910201    61.6 MB       2m34.483s
```

Let’s run this!

Run pgloader using config4.load and config5.load

Config4.load - shows DUMPDDL ONLY (look for the ddl.sql)
Let’s modify the ddl.sql file

Config5.load - shows DATA ONLY

Running pgloader options

pgloader can be run by passing arguments or by passing a command-file

```
ybpgloader --help
ybpgloader [ option ... ] command-file ...
ybpgloader [ 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
```

Running pgloader

Using the command file:

```
ysql_loader --verbose <cmd_file>
```

Or by running it from Docker...

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

example:
sudo docker run --rm --name pgloader \
-v /home/ubuntu/Mlee/PgloaderDSS:/tmp \
mleeyb/ysql-loader:v1.3  /bin/bash -c 'cd /tmp; pgloader' --verbose \
 /tmp/schmonly.conf'
```

The migrated database summary

```
2021-09-17T22:10:25.494000Z LOG report summary reset
                  table name     errors       read   imported      bytes      total time       read      write
--------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
            fetch meta data          0          6          6                    0.100s
             Create Schemas          0          0          0                    0.006s
           Create SQL Types          0          0          0                    0.008s
             Create tables          0          8          8                   18.274s
             Set Table OIDs          0          4          4                    0.012s
         create primary key          0          3          3                   16.418s
--------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
ml_migratedb.ml_order_line          0     800000     800000    56.9 MB       1m10.368s     57.023s   1m7.093s
    ml_migratedb.ml_orders          0      80000      80000     3.5 MB        7.942s       0.666s     5.001s
  ml_migratedb.orders_orig          0      30000      30000     1.2 MB       10.493s       0.295s     0.078s
  ml_migratedb.ml_tinyint1          0        201        201     4.3 kB        0.038s       0.002s     0.001s
--------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
   COPY Threads Completion          0          4          4                   1m10.339s
    Index Build Completion          0          2          2                    1m3.443s
             Create Indexes          0          2          2                    1m0.745s
           Reset Sequences          0          0          0                     2.386s
              Primary Keys          0          0          0                     0.000s
       Create Foreign Keys          0          0          0                     0.000s
           Create Triggers          0          0          0                     0.001s
          Install Comments          0          1          1                     0.025s
--------------------------  ---------  ---------  ---------  ---------  --------------  ---------  ---------
         Total import time          ✓     910201     910201    61.6 MB       3m16.939s
```

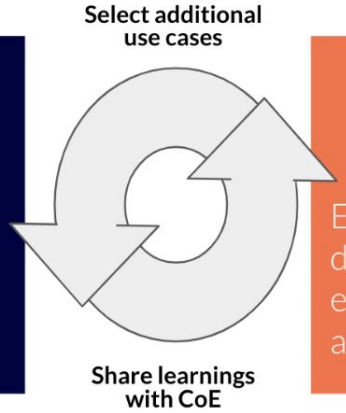
Phases of the Yugabyte Database Modernization Journey

Discovery & Planning

Assess your data infrastructure to determine best services based on business needs. Create an incremental plan for modernization, starting with 1-2 workloads based on effort and business impact.

Database Migration

Use automated migration tools and transformation management processes to **replatform** or **refactor** the database footprint to modern cloud native RDBMS.



Center of Excellence

Ensure a smooth operational transition during and after migrations through enablement, health assessments, onboarding, and process management.

Summary

- Migrate MySQL, MS SQL, SQL Lite, and Postgres databases to YugabyteDB
- Flexible command options to customize the migration
- Proven, Open Source migration tool
- Optimized to work with YugabyteDB

References/Acknowledgements

- YSQL Loader: https://docs.yugabyte.com/latest/integrations/ysql-loader/#root
- pgloader documentation: https://pgloader.readthedocs.io/en/latest/intro.html
- Yugabyte Branch of pgloader: https://github.com/yugabyte/pgloader

Thank You

Join us on Slack: yugabyte.com/slack

Star us on Github: github.com/yugabyte/yugabyte-db