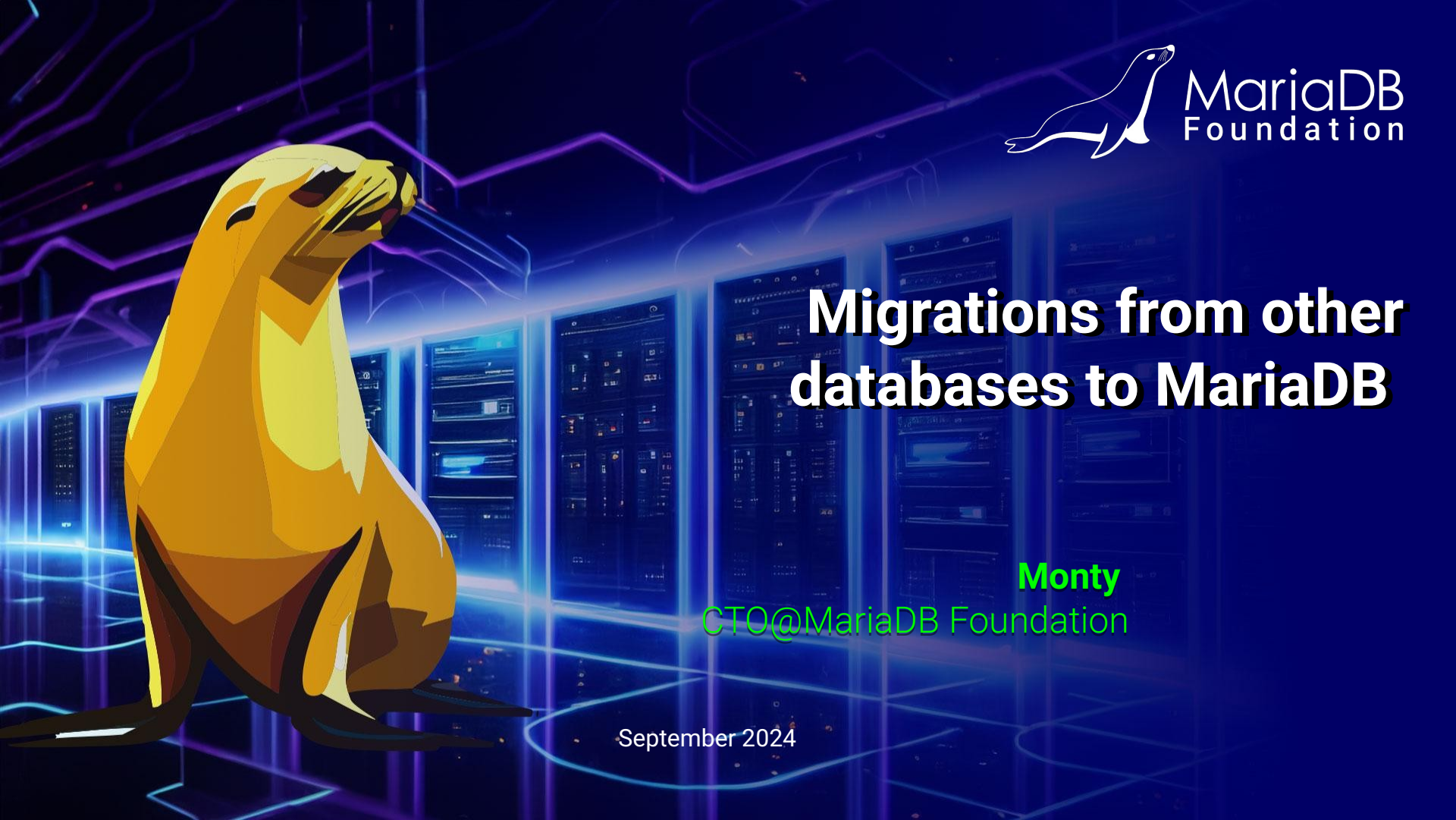

Migrations from other databases to MariaDB

Monty
CTO@MariaDB Foundation

September 2024

Points covered in this talk

- Migrations process goals
- Migration from ANSI SQL compliant databases
- Migration from Oracle
- Migration from MS-SQL (Microsoft SQL server)
- Migration from MySQL
- MariaDB Plc also have projects with migrations from DB2
- How to make the migration process better

Why to migrate to MariaDB

- To move to a fully open source database that is developed together with the community and for which one can get excellent support!
- To save on licenses fees from closed source databases (no per CPU prices)
  - MariaDB is 80% cheaper to run than most closed source databases, with equal (or better) support offering.
- MariaDB
  - Is supported by all major OS distributions and all major cloud vendors
  - Has probably the best replication support among all existing databases!
  - Works on most hardware and OS platforms
  - Easy to upgrade!
  - Has connectors (client libraries) for all major programming languages
  - Supports on rest encryption for your data
  - Allow you to access your old data indefinitely!
  - No incompatible format changes in old storage formats
  - Upgrade process can handle any old format for all supported engines

Unique features in MariaDB

- Based on threads (not processes like PostgreSQL)
  - Allows 20,000+ connected users (20,000 was already used in production in 2007)
- Has a thread pool (MySQL had one, but Oracle removed it 2007 from community server)
  - Greatly speeds up concurrent usage or many connections.
- Parallel and very flexible logical replication
  - Slaves can use different storage engines than the master
- System versioned tables (Allows you to compare current data to how it was 10 years ago)
- Efficient and closely integrated storage engine interface makes it possible to add new ways to store and manipulate data
  - This enables engines like ColumnStore and Cantian with very little code changes
- MariaDB plc has the ability to fix any bug or add any feature to MariaDB server required by customers!

Migration process goals

- The applications should be able to run "**unchanged**" after moved to MariaDB (expect changing the JDBC/ODBC drivers).
- Every new migration should be easier than the previous ones
  - This is achieved by working with MariaDB plc to add the application needed features not yet supported to MariaDB.

ANSI compliant databases

- MariaDB's SQL dialect follows ANSI SQL (large subset, with some extensions)
- Moving from any other ANSI SQL database is relatively easily, as long as MariaDB supports the subset the application uses
- These databases includes MySQL (more later), PostgreSQL, Sybase, DB2 and in some sense MS-SQL (more later).
- Migration process consist of:
  - Analyze what you may have to change in your SQL queries. SQLines: http://www.sqlines.com/download can help you with this!
  - Dump your database as SQL statements.
  - Import into MariaDB with 'mariadb database < dump.sql'
  - Change application to use a MariaDB connector
  - Test application for correctness and performance
  - Adjust privileges (as most databases does this a bit differently)

Migration from MySQL up to 5.7

- Up to MySQL 5.7, MariaDB is in most cases a drop in replacement (more details later):
  - Deinstall MySQL
  - Install MariaDB 10.5
  - Run ‘mariadb-upgrade’
  - Test the application (no need to upgrade any connectors)
- MariaDB can be a slave to MySQL 5.7 if:
  - MySQL native passwords are used (at least for the slave)
  - No JSON TYPE (JSON objects stored in TEXT works)

Migration from Oracle

To support Oracle's SQL syntax in MariaDB, the following feature was introduced:

SET SQL_MODE=**ORACLE**;

This changes the parser to be support a large part of Oracle's syntax and add new functionality needed by Oracle applications (like PACKAGES).

The ORACLE mode was developed together with DBS Bank allowing them to move their applications to MariaDB unchanged!

MariaDB plc is now working with other companies to do the same!

Migration from Oracle

Most of the Oracle compatibility was added in MariaDB 10.3

See https://mariadb.com/kb/en/sql_modeoracle/ for full list!

Support for Oracle syntax was added to:

- Stored procedures
- Cursors
- LOOP (like FOR i IN 1..10 LOOP ... END LOOP)
- Variable handling (like var1 table_name.column_name%TYPE)
- Exceptions
- Oracle function names (some aliases and some new ones)
- Synonyms for Oracle Basic SQL types
- Packages
- Rudimentary support for that NULL = “”
  - With **sql_mode=EMPTY_STRING_IS_NULL**
- We also added new functionality, like **SEQUENCES** !

Stored Procedures and Stored Functions

Oracle syntax	Description
CREATE PROCEDURE p1 (param OUT INT)	ANSI uses (OUT param INT)
CREATE PROCEDURE p1 (a IN OUT INT)	ANSI uses (INOUT param INT)
AS before function body	CREATE FUNCTION f1 RETURN NUMBER AS BEGIN...
IS before function body	CREATE FUNCTION f1 RETURN NUMBER IS BEGIN...
If function has no parameters then parentheses must be omitted	Example: CREATE PROCEDURE p1 AS BEGIN NULL; END;
CREATE PROCEDURE p1 AS BEGIN END p1;	Optional routine name after END keyword. MDEV-12089
CREATE FUNCTION f1(a VARCHAR)	VARCHAR can be used without length for routine parameters and RETURN clause.
CREATE AGGREGATE FUNCTION f1( )	Creates an aggregate function.
No CALL needed in Stored Procedures	In Oracle mode one can call other stored procedures with name only.
RETURN. Can also be used in stored procedures	ANSI uses RETURNS.

Cursors

Oracle syntax	Description
CREATE PROCEDURE p1 AS CURSOR cur IS (SELECT a, b FROM t1); BEGIN FOR rec IN cur ...	Explicit cursor with FOR loop. MDEV-10581
CREATE PROCEDURE p1 AS rec IN (SELECT a, b FROM t1)	Implicit cursor with FOR loop. MDEV-12098
CURSOR c(prm_a VARCHAR2, prm_b VARCHAR2) ... OPEN c(1,2)	Cursor with parameters. MDEV-10597
CURSOR c(prm_a VARCHAR2, prm_b VARCHAR2) ... FOR rec in c(1,2)	Cursor with parameters and FOR loop. MDEV-12314
Cursor with parameters and FOR loop. MDEV-12314	Example: CREATE PROCEDURE p1 AS BEGIN NULL; END;
s %ISOPEN, %ROWCOUNT, %FOUND, %NOTFOUND	Explicit cursor attributes. MDEV-10582

LOOP

Oracle syntax	Description
FOR i IN 1..10 LOOP ... END LOOP	Numeric FOR loop. MDEV-10580
GOTO	GOTO statement. MDEV-10697
<<label>> used with GOTO	ANSI uses label:. MDEV-10697
To leave loop block: EXIT [label] [WHEN bool_expr]	ANSI syntax is IF bool_expr THEN LEAVE label
[<<label>>] WHILE boolean_expression LOOP statement... END LOOP [label] ;	Oracle style WHILE loop
CONTINUE [label] [WHEN boolean_expression]	CONTINUE is only valid inside a loop

VARIABLES

Oracle syntax	Description
var:= 10; Can also be used with MariaDB systemvariables	MariaDB uses SET var= 10;
var INT := 10	Default variable value
var1 table_name.column_name%TYPE	Take data type from a table column. MDEV-10577
var2 var1%TYPE	Take data type from another variable
rec1 table_name%ROWTYPE	Take ROW structure from a table. MDEV-12133
rec2 rec1%ROWTYPE	Take ROW structure from ROW variable
CURSOR c1 IS SELECT a,b FROM t1; rec1 c1%ROWTYPE;	Take ROW structure from a cursor. MDEV-12011
Variables can be declared after cursor declarations	In MariaDB mode, variables must be declared before cursors. MDEV-10598
Triggers uses :NEW and :OLD	ANSI uses NEW and OLD. MDEV-10579

VARIABLES

Oracle syntax	Description
SQLCODE	Returns the number code of the most recent exception. Can only be used in Stored Procedures. MDEV-10578
SQLERRM	Returns the error message associdated to it's error number argument or SQLCODE if no argument is given. Can only be used in Stored Procedures. MDEV-10578
SQL%ROWCOUNT	Almost same as ROW_COUNT(). MDEV-10583
ROWNUM	Returns number of accepted rows

Exceptions

Oracle syntax	Description
BEGIN ... EXCEPTION WHEN OTHERS THEN BEGIN .. END; END;	Exception handlers are declared at the end of a block
TOO_MANY_ROWS, NO_DATA_FOUND, DUP_VAL_ON_INDEX	Predefined exceptions. MDEV-10839
RAISE TOO_MANY_ROWS ; .... EXCEPTION WHEN TOO_MANY_ROWS THEN ...	Exception can be used with RAISE and EXCEPTION...WHEN. MDEV-10840
CREATE OR REPLACE FUNCTION f1 (a INT) RETURN INT AS e1 EXCEPTION...	User defined exceptions. MDEV-10587

BEGIN blocks

Oracle syntax	Description
BEGIN to start a block	MariaDB uses BEGIN NOT ATOMIC for anyonymous blocks. MDEV-10655
DECLARE is used before BEGIN	DECLARE a INT; b VARCHAR(10); BEGIN v:= 10; END;
WHEN DUP_VAL_ON_INDEX THEN NULL ; NULL; WHEN OTHERS THEN NULL	Do not require BEGIN..END in multi-statement exception handlers in THEN clause. MDEV-12088

Simple Syntax Compatibility

Oracle syntax	Description
ELSIF	ANSI uses ELSEIF
SELECT UNIQUE	Same as SELECT DISTINCT. MDEV-12086
TRUNCATE TABLE t1 [DROP STORAGE] or [REUSE STORAGE]	DROP STORAGE and REUSE STORAGE are allowed as optional keywords for TRUNCATE TABLE. MDEV-10588
Subqueries in a FROM clause without an alias	SELECT * FROM (SELECT 1 FROM DUAL), (SELECT 2 FROM DUAL)
UNION, EXCEPT and INTERSECT all have the same precedence.	INTERSECT has higher precedence than UNION and EXCEPT in non-Oracle modes.
MINUS	MINUS is a synonym for EXCEPT.

Functions

Oracle syntax	Description
ADD_MONTHS()	Added as a wrapper for DATE_ADD() to enhance Oracle compatibility. All modes.
CAST(expr as VARCHAR(N))	Cast expression to a VARCHAR(N). MDEV-11275
DECODE	In Oracle mode, compares and matches search expressions
LENGTH() is same as CHAR_LENGTH()	MariaDB translates LENGTH() to OCTET_LENGTH(). In all modes one can use LENGTHB() as a synonym to OCTET_LENGTH()

Functions

Oracle syntax	Description
substr('abc',0 ,3) same as substr('abc', 1 ,3)	Position 0 for substr() is same as position 1
SYS_GUID	Generates a globally unique identifier. Similar to UUID but without the -. All modes.
TO_CHAR	Added to enhance Oracle compatibility. All modes.
TRIM, LTRIM, RTRIM, LPAD and RPAD	Returns NULL instead of an empty string if returning an empty result. These functions can also be accessed outside of ORACLE mode by suffixing _ORACLE onto the end of the function name, such as TRIM_ORACLE.

Prepared Statements

Oracle syntax	Description
PREPARE stmt FROM 'SELECT :1, :2'	ANSI uses ?. MDEV-10801
EXECUTE IMMEDIATE 'INSERT INTO t1 SELECT (:x,:y) FROM DUAL' USING 10,20	Dynamic placeholders. MDEV-10801

Synonyms for Basic SQL Types

Oracle syntax	Description
VARCHAR2	VARCHAR
NUMBER	DECIMAL
DATE (with time portion)	MariaDB DATETIME
RAW	VARBINARY
CLOB	LONGTEXT
BLOB	LONGBLOB

Packages

- CREATE PACKAGE
- CREATE PACKAGE BODY
- DROP PACKAGE
- DROP PACKAGE BODY
- SHOW CREATE PACKAGE
- SHOW CREATE PACKAGE BODY

NULL Handling in Oracle mode

NULL As a Statement:

- IF a=10 THEN NULL; ELSE NULL; END IF

Translating Empty String Literals to NULL (In Oracle, empty string ('') and NULL are the same thing). By using **sql_mode=EMPTY_STRING_IS_NULL** you can get a similar experience in MariaDB:

- SET sql_mode=EMPTY_STRING_IS_NULL;
- SELECT '' IS NULL; -- returns TRUE
- INSERT INTO t1 VALUES (''); -- inserts NULL
- Concat Operator Ignores NULL
  - CONCAT() and || ignore NULL in Oracle mode. Can also be accessed outside of ORACLE mode by using CONCAT_OPERATOR_ORACLE. MDEV-11880 and MDEV-12143.

New Oracle features in 10.6

In MariaDB 10.6 there are several new Oracle features, thanks to contributions from **Woqtech**:

- **ROWNUM**, with LIMIT optimization
  - In MariaDB normal mode one can use ROWNUM()
  - **SELECT * from t1 WHERE ROWNUM() <= 10;**
- **SYS_GUID()**
  - Like UUID() but without '-'
- **MINUS** as an alias for **EXCEPT**

More with be added with future migration work:

- Features paid for by one customer will be available to everyone in next major MariaDB version
- Customers paying for features will get access to them at once they are developed (before anyone else)!

New features in development for migrations

- MDEV-32101 **CREATE PACKAGE [BODY] for sql_mode=DEFAULT (Done in 11.4.1)**
  - This is a preparatory task for IBM DB2's CREATE MODULE compatibility.
- MDEV-12252 **ROW data type for stored function return values**
  - In review. Needs QA when review is done. Hopefully, for 11.7
- MDEV-20034 sql_mode="oracle" does not support stored code r**eturning SYS_REFCURSOR**
  - In a stage tree. Needs review and QA. Hopefully, for 11.7
- MDEV-32380 **Array data type for stored routines**
  - In a stage tree. Needs review and QA. Hopefully, for 11.7
  - Later, the VARRAY Oracle's PL/SQL data type will be done on top of it.

New features in development for migrations

- MDEV-10152 Add support for TYPE .. IS REF CURSOR
- MDEV-10654 IN, OUT, INOUT parameters in CREATE FUNCTION
- MDEV-10862 Stored procedures: default values for parameters (optional parameters)
- MDEV-13139 CREATE PACKAGE: package-wide declarations
- MDEV-13648 Add FULL OUTER JOIN to MariaDB
- MDEV-13817 add support for oracle's left join syntax - the (+)
- MDEV-19149 System package SYS.DBMS_OUTPUT
- MDEV-19635 System package SYS.DBMS_SQL
- MDEV-20238 "DEFAULT" parameters for functions/procedures
- MDEV-20649 RAISE_APPLICATION_ERROR()"
- MDEV-34316 sql_mode=ORACLE: Ignore the NOCOPY keyword in stored routine parameters
- MDEV-34317 DECLARE TYPE type_name IS RECORD (..) with scalar members in stored routines
- MDEV-34319 DECLARE TYPE .. TABLE OF .. INDEX BY in stored routines
- MDEV-34320 CREATE PACKAGE: PRAGMA RESTRICT_REFERENCES
- MDEV-34324 System package SYS.DBMS_AQ
- MDEV-34325 System package SYS.DBMS_CRYPTO
- MDEV-34326 System package SYS.DBMS_LOB
- MDEV-34327 System package SYS.DBMS_RANDOM
- MDEV-34328 System package SYS.DBMS_SESSION
- MDEV-34329 System package SYS.DBMS_TRANSACTION
- MDEV-34330 System package SYS.DBMS_UTILITY
- MDEV-34331 System package SYS.UTL_I18N
- MDEV-34332 System package SYS.UTL_RAW
- MDEV-34333 System package SYS.UTL_ENCODE
- MDEV-34391 SET PATH statement
- MDEV-34484 Overloading in package routines

Migration from MS-SQL

MS-SQL was originally developed by Sybase. It's largely ANSI SQL, but has some differences that can cause problems with migrations.

We have started to handle some of the issues already in 10.2.3, but there is still more work to do:

SET SQL_MODE=MSSQL;

'[' and ']' can be used to quite identifiers

CREATE TABLE [t 1] ([a b] INT);

Work with MariaDB devs for better compatibility

What to do if you are planning to move a large set of applications to MariaDB from another database that uses constructs, functionality or syntax that MariaDB doesn't currently understand:

- Start a migration process with MariaDB Plc that includes changing MariaDB to understand the new syntax/functionality!
- Have your developers or employ a company to extend MariaDB with the new functionality and contribute the code to the MariaDB foundation to be included in the current and all future versions of MariaDB.

The above is the surest way to ensure that all future migration projects will be easier than the ones before!

MariaDB usage at DBS

- DBS has since 2017 migrated > 80% of their Oracle applications to MariaDB with no changes in their applications!
- They started migrating their biggest, most complex application to MariaDB.
  - They wanted to ensure management and all other departments that MariaDB will work for them!
- In general the performance with MariaDB has been better than with Oracle.
  - A few really complex queries are a bit slower, should be fixed in MariaDB 11.0 thanks to the new optimizer.
- I, and other people from MariaDB plc, visit DBS 1-2 times a year to ensure that everything works smoothly and to discuss new features needed by DBS.
  - We are constantly adding new features to help DBS use MariaDB even more efficient!

MariaDB usage at DBS

DBS is using

- MariaDB Enterprise, to get longer life cycles for their products
  - MariaDB Enterprise is supported at least 8 years
  - They are trying to avoid upgrades, like many customers
- Master-Slave and Master-delayed-slave, for HA
- S3 storage engine, for archiving and giving access to archives to other parts of the organization
- MaxScale as a proxy
  - MaxScale will in the near future get a new features like “multiplex, copy workload and compare results” that will allow DBS to compare current version with future version. This will greatly simplify future upgrades as DBS can verify beforehand that the upgrade will not cause any issues!

New features to make migrations easier

MySQL compatibility:

- Extended mariadb-upgrade to handle:
  - Conversion of MySQL JSON binary format to MariaDB TEXT format (2021)
  - MySQL 5.6/5.7 new partition format in InnoDB (2022)

MySQL compatibility features in development

- Tool to convert MySQL config files to work with both MySQL and MariaDB
  - Tool to check if config file is compatible already exists
- Tool to check a command will give syntax errors in MariaDB
  - Can be used to check that all queries in a general log can be parsed (and thus probably be executed) in MariaDB
- LATERAL tables
- SHA password plugin (to be able to handle MySQL stored passwords)

New features to make migrations easier

Percona server features ported and enhanced to MariaDB:

- Binlog space limit (11.4)
- Storage engine metrics in slow log (11.4)
- More table and index usage statistics in userstat (11.4)
  - Columns added to INDEX_STATISTICS
    - QUERIES
  - Columns added to TABLE_STATISTICS
    - ROWS_INSERTED, ROWS_DELETED, ROWS_UPDATED, KEY_READ_HITS and KEY_READ_MISSES.
  - Columns added to CLIENT_STATISTICS and USER_STATISTICS:
    - KEY_READ_HITS and KEY_READ_MISSES.
- Warnings in sql_error_log (11.4)
- SENT_ROWS in information_schema.process_list (11.4)
- slow_query_log_always_write_time variable (implemented as log_slow_always_query_time) 11.4

New features in development for migrations

Extending the KB documentation with step-by-step instructions for doing migrations from different MySQL setups:

- Single server
- MySQL replication cluster
- Percona XTRADB (Galera) cluster

Step by step MySQL → MariaDB migration

First fix configuration files so that they **work on both MySQL and MariaDB.**

- If mariadb binaries is installed, one can use the script at https://jira.mariadb.org/browse/MDEV-32745 to find the not supported options.
  - Fix is done by moving all MySQL specific options to a section: **[mysqld-5.7]**
  - This includes all options that uses MySQL specific directories for logging or replication(in other words, paths that has 'mysql' as part of the path).
  - Add all MariaDB specific options to the section: **[mariadbd]**
- Make a copy of the options with path's in the [mysqld-5.7] section and change the 'mysql' path part to 'mariadb' (except binary and relay logs!)
- Create the directories needed for the options and ensure that the owner is 'mariadbd'
- Optionally add to **[mariadbd]**
  - **skip-slave-start**

Step by step MySQL → MariaDB migration

Things to do in the MySQL client to check that we are ready to start migration

- log all queries and result
  - **tee /tmp/mysql-migration.log**
- Check that there are no incompatible MySQL features used
  - **SELECT user, plugin FROM mysql.user where plugin like "%sha%";**
  - **select table_schema, table_name from information_schema.columns where data_type="JSON";**
  - **select table_name, create_options from information_schema.tables where create_options like "%comp%";**
  - **select table_schema, table_name, create_options from information_schema.tables where create_options like '%ENCRYPTION%';**
  - **select @@Innodb_file_per_table,@@Innodb_fast_shutdown\G**
    - The last query should return 1 and 0
  - exit

Step by step MySQL → MariaDB migration

- Stop slave, REMEMBER(!) master_log_file and exec_master_log_pos!
  - **STOP SLAVE;**
  - **SHOW SLAVE STATUS;**
- Ensure fast shutdown is 0
  - **set @@global.Innodb_fast_shutdown=0;**
- **exit**

Step by step MySQL → MariaDB migration

Things to do from shell:

- Stop MySQL
  - **systemctl stop mysql.service**
  - **systemctl disable mysql.service**
- Change owner of datadir
  - **chown -R mariadb:mariadb #mariadb-datadir#**
- Change owner of all other mysql related files and directories. You can find them with
  - **my_print_defaults --mysqld | grep '/'**
  - **chown -R mariadb:mariadb #all** directories and important files from above
- start mariadbd on the node data
  - **systemctl start mariadb.service**
  - **mariadb-upgrade**

Step by step MySQL → MariaDB migration

Things to do in the MySQL client:

- **tee /tmp/mysql-migration-2.log**
- **show slave status;**
- **start all slaves;**

Or alternatively:

- **CHANGE MASTER 'channel-name' MASTER_USE_GTID=no …**

In case of problems try doing

- **RESET SLAVE ; START ALL SLAVES;**

Step by step MySQL → MariaDB migration

Last checks and configurations:

- Check log files for errors/warnings
- Remove **skip-slave-start** from your configuration file

New features to optionally enable in the configuration file for MariaDB

- **aria-pagecache-buffer-size=#** #Should be at least same as key-buffer-size was before
- **key-buffer-size=64K** # Assuming that MyISAM files are not used anymore
- **thread_handling=pool_of_threads** # For setups with a lot of concurrent queries
- **slave-parallel-threads=8** # For slaves with a lot of replicated queries
- **Log-basename=###** # Basename for all log files (simplifies name handling)

Time for questions

If you any questions left about migrating to MariaDB,

now is your chance!