Upgrading to a newer major version of MariaDB

What mysql_upgrade really does and what problems remain

Valerii Kravchuk, Principal Support Engineer, MariaDB

valerii.kravchuk@mariadb.com

Who am I and What Do I Do?

Valerii (aka **Valeriy**) **Kravchuk**:

- MySQL Support Engineer in MySQL AB, Sun and Oracle, 2005-2012
- Principal Support Engineer in Percona, 2012-2016
- Principal Support Engineer in MariaDB Corporation since March 2016
- **http://mysqlentomologist.blogspot.com** - my blog about MariaDB and MySQL (including some **HowTo**s, not just bugs marketing)
- **https://www.facebook.com/valerii.kravchuk** - my Facebook page
- **@mysqlbugs #bugoftheday** - links to interesting MySQL bugs, few per week
- **MySQL Community Contributor of the Year 2019**
- I speak about MySQL and MariaDB in public. Some slides from previous talks are here and there…
- "I solve problems", "I drink and I know things"

Disclaimers

- Since September, 2012 I act as an Independent Consultant providing services to different companies
- All views, ideas, conclusions, statements and approaches in my presentations and blog posts are mine and may not be shared by any of my previous, current and future employees, customers and partners
- All examples are either based on public information or are truly fictional and has nothing to do with any real persons or companies. Any similarities are pure coincidence :)
- The information presented is true to the best of my knowledge

Sources of information about MariaDB upgrades

- "**Upgrading MariaDB**" KB page and related links
- "**mysql_upgrade**" KB page
- Monty's "**Upgrading Between Major MariaDB Versions**" blog post of 2020 that caused interesting discussions
- My later "**What mysql_upgrade really does in MariaDB, Part I**" blog post
- Related bugs and tasks at **https://jira.mariadb.org**

- I am not going to speak about or refer to Enterprise Server (and its documentation) or SkySQL today...

What is this session about?

- Upgrade steps are clearly documented:
  - Take a backup (including **mysqldump** of the **mysql** database), do clean full (**innodb_fast_shutdown** = 0) shutdown
  - Upgrade/install new binaries, without server startup if possible
  - Start server with **--skip-grant-tables**, run **mysql_upgrade**
  - Restart and check the error log, there must be no errors
  - Upgrade slaves first in async replication setups
- But can we skip major versions and upgrade from 10.1 to 10.5 directly or should we do it one major versions at a time:
  10.1 → 10.2 → 10.3 → **10.4** → **10.5**
- What **mysql_upgrade** really does?
- What **mysql_upgrade** does NOT detect or resolve?
- What are the known bugs and problems in **mysql_upgrade** or upgrade process in general in MariaDB?

Can we skip major versions while upgrading?

- MySQL Manual is very clear, and most DBAs still trust it in case of MariaDB too:

  "Upgrade that skips versions is not supported. For example, upgrading directly from MySQL 5.5 to 5.7 is not supported."
- It's a lame way to say "we do not care to test or fix"
- The goal on Monty's post was to say that we in MariaDB care, to some extent…
- It's still an open discussion if doing it in steps really helps in general (it may be mandatory is case of Galera, it would make sense to test applications with less changes at a time, there are also bugs...)
- But **mysql_upgrade** in MariaDB is supposed to work for such cases, starting from major version **5.1** (?) at least...

What **mysql_upgrade** really does

- Read the manual
- Or just enable general query log, run it and see (as I did):
  - You may find out that it disables writing to the binary log or other Galera cluster nodes
  - It runs **CHECK TABLE … FOR UPGRADE …** for **mysql.*** tables, gets list of engines and **mysql.user** structure…
  - It gets the lists of all tables in all (real) databases
  - It does multiple **ALTER**s to **mysql.*** tables (including switching to Aria in case of 10.4+)
  - Missing and needed **mysql.*** tables are created (including InnoDB statistics ones)
  - **performance_schema** is (re-)created
  - **CHECK TABLE … FOR UPGRADE** is executed for all real tables in other databases
  - **FLUSH PRIVILEGES** is ececuted

What **mysql_upgrade** really does

- What if you do not want to enable general query log and make conclusions from a few test upgrade runs?
- Go read the source code (on GitHub if you prefer):
  - **https://github.com/MariaDB/server/blob/10.5/client/mysql_upgrade.c**
  - **static int run_sql_fix_privilege_tables(void)**
  - **https://github.com/MariaDB/server/blob/10.5/scripts/CMakeLists.txt**
  - **https://github.com/MariaDB/server/blob/10.5/scripts/mysql_system_tables_fix.sql**
- One day I'll write a blog post on the above...

Some known problems of **mysql_upgrade**

- MDEV-22323 - "**Upgrading MariaDB**" - collection of tasks related to upgrading (major/minor) versions of MariaDB, between MariaDB and MySQL/Percona etc
- MDEV-24540 - "**Detect incompatible MySQL virtual columns and report in error log**"
- MDEV-24453 - "**mysql_upgrade does not honor --verbose parameter**" - that would help with debugging
- MDEV-24316 - "**cross upgrade from MySQL - have precheck tool**"
- MDEV-24093 - "**Detect during mysql_upgrade if type_mysql_json.so is needed and load it**"
- MDEV-23962 - "**Remove arc directory support**", What is it for? :)
- MDEV-23008 - "**store mysql_upgrade version info in system table instead of local file**"

Problems that **mysql_upgrade** does not resolve

- Problems to start some engine or plugin in a new version
- Problems with tables using storage engines that are “not there” in a new version, for whatever reason
- Problems that **mysqlcheck**:

  ```
  mysqlcheck ... --check-upgrade ... --auto-repair
  ```

  does not detect or resolve
- For example, old datetime data types in InnoDB tables pre-MySQL 5.6 or MariaDB 10.0. They are not converted to a new format and in 10.3+ this may cause *problems*...
- Check for the tip of the iceberg here (the case oif MySQL 5.7.x is discussed), one day I’ll write a lot more...

Some known bugs in **mysql_upgrade**

- MDEV-24566 - "**mysql_upgrade failed with "('mariadb.sys'@'localhost') does not exist" and mariadb 10.4/10.5 on docker**"
- MDEV-24555 - "**mysql.event table check fails after an upgrade**"
- MDEV-24452 - "**ALTER TABLE mysql.event takes infinite time which for example breaks mysql_upgrade**", 10.3 to 10.5 upgrade
- MDEV-24172 - "**data structure tests for innodb_table_stats and innodb_index_stats incorrect**", see also MDEV-13274 and MDEV-13360 - "t**oo long values in mysql.innodb_table_stats.table_name**", still in 10.1.38
- MDEV-24122 - "**Minor updates to MariaDB 10.2/10.3 seems to be adding a DEFAULT ROLE to the show grants command that is invalid.**", in review

Some known bugs in **mysql_upgrade** or upgrade

- MDEV-22683 - "**mysql_upgrade misses some changes to mysql schema**"
- MDEV-22655 - **"CHECK TABLE ... FOR UPGRADE fails to report old datetime format"** - my bug report
- MDEV-22645 - "**default_role gets removed when migrating from 10.1 to 10.4**" - something that **mysql_upgrade** does not fix
- MDEV-22477 - "**mysql_upgrade fails with sql_notes=OFF**"

- MDEV-22357 - "**Clearing InnoDB autoincrement counter when upgrading from MariaDB < 10.2.4**" - something **mysql_upgrade** can hardly fix, engine issue
- MDEV-22167 - "**auto_increment counters changed weirdly after direct 10.1 -> 10.3 upgrade**"
- MDEV-22220 - "**upgrade fails from 10.3 to 10.4**" - error on RocksDB startup, we can not fix that

Summary

- Running **mysql_upgrade** after any major version upgrade is a must. MariaDB may work for years without that, but you may end up with all kinds of troubles
- I was not able to find any confirmed bug reports about simple cases when direct upgrade skipping some major version(s) fails while the same upgrade done step by step does not fail.
- Missing tables etc are really created at later stages of the process, no matter what was the version we upgrade from. In SQL code I see no checks for what the older version was. Looks like SQL is executed no matter if we get the error, and we just continue to the next statement that is going to fix the problem, if any.
- The SQL code executed, with checks etc, is complex enough. **mysql_upgrade** tries to deal with many details and fix many things, so one can not exclude bugs (many!).
- It seems to be safe to run **mysql_upgrade** multiple times (**--force**).
- Assuming that **CHECK TABLE ... FOR UPGRADE** from MariaDB 10.x is able to find and fix all problems in tables originating from MariaDB or MySQL version x.y.z so that the result is usable, the process of direct upgrade skipping intermediate major versions may work well

Q & A

- Thank you! Thanks to **MariaDB Foundation**!
- Please, report bugs to **https://jira.mariadb.org**!