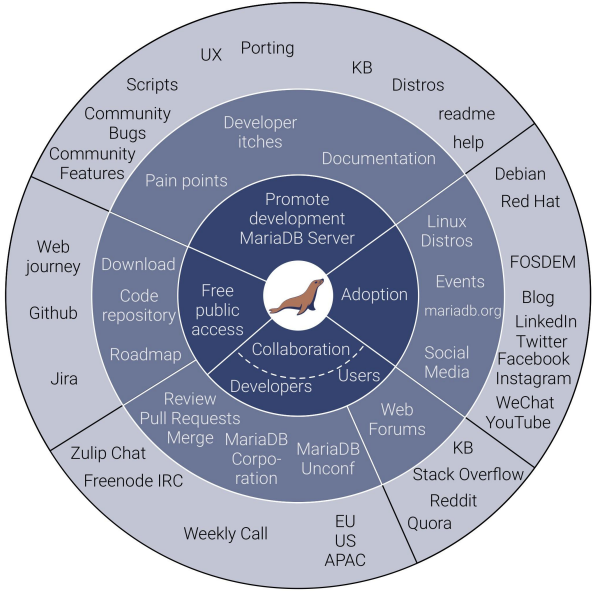


New Contributor Tutorial

Vicențiu Ciorbaru
Chief Development Officer
@ MariaDB Foundation

What is MariaDB?

- Created as a response to Oracle’s acquisition of MySQL (April 20, 2009)
- Named after Monty’s youngest daughter Maria (with My's approval :))
- First release in Oct 2009, MariaDB 5.1
- The development is guided by the MariaDB Foundation
- Strong focus on community development, not just in-house

How to contribute?

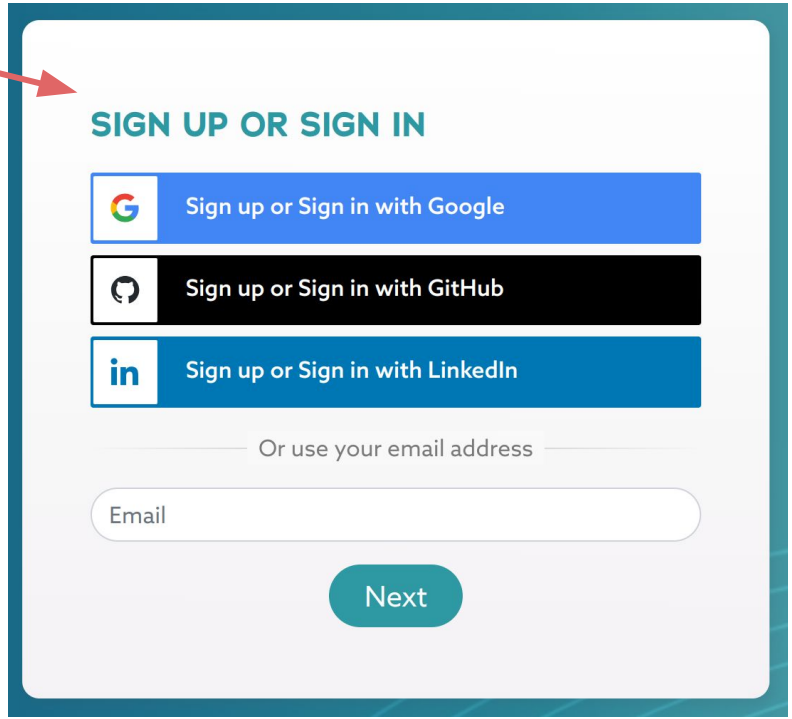
- Similar to many open source projects
- We have our code hosted at https://github.com/MariaDB
- We accept contributions in many forms
- Simplest are KB edits and pull requests (code)

How to contribute?

- Go to https://mariadb.com/kb/en/
- Create a new user or login

How to contribute?

- Go to https://mariadb.com/kb/en/
- Create a new user or login
- Then open any KB article
- Use Edit or Translate Menus on the left
- All contributions are reviewed and / or curated



Contributing code

- Easiest way is to submit patches via Github
  - You will need a Github account

- The MariaDB Server codebase can be found at:
  - https://github.com/MariaDB/server

- Fork the MariaDB Server code on Github to your own repository.
  - https://help.github.com/articles/fork-a-repo/

- Create a patch and submit a pull request:
  - https://help.github.com/articles/creating-a-pull-request-from-a-fork/

Step 1: Get the code

- We use a git repository. Download and install git.
- $ sudo apt install git
  - Fork the server on Github
- Clone the server fork you have created
  - $ git clone https://github.com/*<your-user>*/*<your-fork>*.git
- $ cd server

Step 2: Compile the server

- Install all required build dependencies:
  - $ sudo apt-get build-dep mariadb-server

- Use cmake to generate Makefiles
  - $ cmake . -DCMAKE_BUILD_TYPE=Debug

- Compile
  - $ make -j8

- You can also use build scripts
  - BUILD/compile-pentium64-debug-max

Step 3: Test the server

- We have finished building the server. We can run tests to see if it works properly.

- $ cd mysql-test && ./mtr --parallel=8 --mem
  - --parallel=n starts n tests in parallel (choose n to be roughly 2x #cpus)
  - --mem will use a memory filesystem instead of disk (you need ~2GB / thread)

Step 4: Write a patch

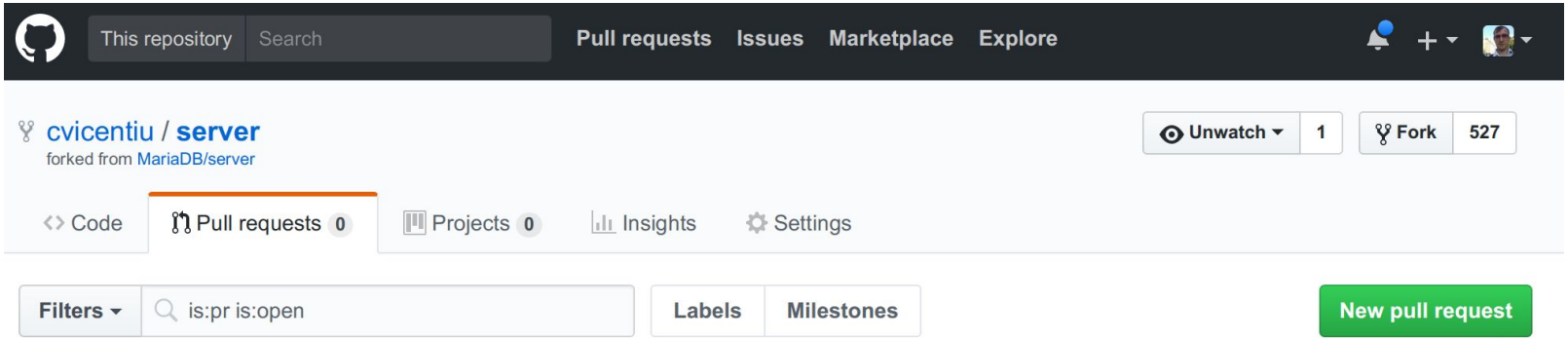
- For new features, make sure you are using main as a base:
  - ~/server/$ git checkout main && git pull

- Create a new branch based on newest main
  - ~/server/$ git branch main-patch # Create a branch
  - ~/server/$ git checkout main-patch # Change branch

- Write your patch, then add all changes and commit.
- Write a descriptive commit message.
  - ~/server/$ git add . && git commit

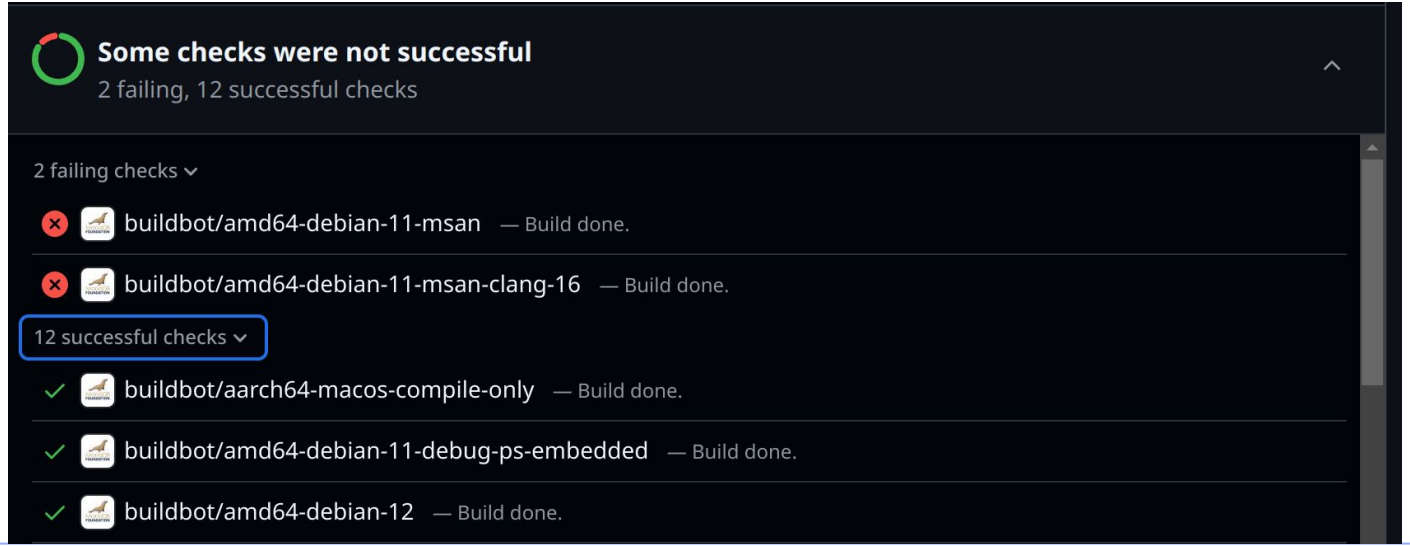
Step 5: Submit pull request

- Push your patch to your local github repository
- $ git push
- Go to your github fork page, pull-requests tab and start a new pull request.



Step 6: Make sure tests pass in buildbot

- Pull requests get a dedicated CI test run
  - Protected branch builders are **mandatory**
  - There are other builders that need to also pass.
  - https://buildbot.mariadb.org/#/grid?branch=refs/pull/3810/merge



Tips & tricks

- Before attempting any particular task, make sure that:
  - **You** understand what the task's goal is
  - Your proposed solution has the right approach
    - Ask in a comment / in JIRA before spending too much time coding it.

- Before submitting the PR:
  - Make sure your code is well documented
    - Commit message explains **why** and **what** is changed. Sometimes **how**, but clear code is better.
  - Add test cases.
  - Make sure code follows CODING_STANDARDS.md
    - https://github.com/MariaDB/server/blob/main/CODING_STANDARDS.md

Tips & tricks

- Reviews will come as comments on GitHub.
- Address any concerns and update your branch.
- It’s strongly recommended to include test cases as part of your patch.

10 sql/sys_vars.ic — Show comments — View

```
@@ -2493,7 +2493,14 @@ public:
 class Sys_var_vers_asof: public sys_var
-{
+// XXX now I'm really starting to dislike your ... QUERY FOR SYSTEM_TIME
+// extension. A *third* way to do the same thing? It's SQL, not Perl.
+//
+//   SELECT * FROM (SELECT ....) FOR SYSTEM_TIME AS OF xxx
+//   SET STATEMENT vers_current_time=xxx SELECT ...
+//   SELECT ... QUERY FOR SYSTEM_TIME AS OF xxx
+//
+// that's a bit too many alternatives.
```

midenok on Jul 20 • edited

You don't like it because you didn't type versioned queries much. When you have to type queries form (2) is almost unusable because impossible to memorize. Form (1) is weird and not evident. Do you really think 3 ways is much? I would say diversity is versatility. If I love it, why someone else wouldn't love it? Do we want our product to be maximum popular? What harm (3) can do?

vuvova on Jul 22 Owner

I didn't type versioned queries so far. But I've typed many SQL queries. And often I wanted for SQL to be less verbose. But SQL is a very verbose language. And new syntax should be consistent with existent language style and logic. Harm is 1. added confusion (many different inconsistent ways of doing the same thing, one needs to remember them, because feature C is not a logical combination of features A and B, but must be remembered separately). and 2. we have to maintain new syntax for many years.

But ok, if you strongly prefer to keep it, let's keep it for now, I'll see what others think about it.

midenok on Jul 28

I agree about 2., but not about 1. I will explain more if you will ask me to. But for now, thanks for giving time out!

midenok on Sep 4

Closed.

```
 public:
   Sys_var_vers_asof(
     const char *name_arg,
```

Tips & tricks

- mysql-test-run.pl script will search for files ending in *.test in
  - mysql-test/main/
  - mysql-test/suite/

- It will run all statements inside it as SQL queries to the server.
  - There is a special syntax for mtr commands.

- It will compare output to that found in <test-case>.result file.

Tips & tricks

- Inside mysql-test/main/ directory create a test file.
  - ~/server/$ cd mysql-test && touch t/hello.test
- Add a statement within the test file
  - $ echo 'SELECT "Hello World!";' > t/hello.test
- Run mysql-test-run on the new test.
  - $ ./mtr hello

Jira & Zulip Chat & the Community

- We use Jira https://jira.mariadb.org to track
  - Bugs, Pull Requests and New Feature requests
  - You can contribute by filing bugs too!

- Find us on Zulip Chat:
  - https://mariadb.zulipchat.com

- Mailing Lists: https://mariadb.com/kb/en/mailing-lists/
  - Maria Developers - For MariaDB development discussions
  - Maria Discuss - For users and general discussions
  - Announce - Low volume announce list (read only)

How do I find tasks to work on?

- JIRA label **beginner-friendly** https://jira.mariadb.org/issues/?filter=31469
  - labels = beginner-friendly AND status = open

- Other projects also have similar labels:
  - **TypeScript** https://github.com/microsoft/TypeScript/issues?q=is%3Aissue%20state%3Aopen%20label%3A%22Good%20First%20Issue%22
  - **Python**
    - https://bugs.python.org -> Select easy issues

How do I find tasks to work on?

- Ask on Zulip (https://mariadb.zulipchat.com)
  - GSoC - We have participated in the past 10 years
  - You can help with packaging too!

- We have other more beginner-friendly projects
  - AI Frameworks integrations
    https://mariadb.org/announcing-the-mariadb-vector-bounty-program

  - MariaDB Jupyter Kernel
    https://mariadb.com/kb/en/using-the-mariadb-jupyter-kernel/

License

- MariaDB Contributor Agreement
  - https://mariadb.org/get-involved/getting-started-for-developers/mca/
  - https://mariadb.org/get-involved/getting-started-for-developers/mca-faq/
- BSD-new
  - https://en.wikipedia.org/wiki/BSD_licenses

- **Why is it needed?**
  - So that MariaDB Foundation can defend your Open Source code in court!
- **FSF requires code copyright assignation as well.**
  - **https://www.gnu.org/licenses/why-assign.html**

Git best practices

- How do I do X?
- Problem: Multiple ways of doing X
- Everybody has their own way, hard for somebody new to learn.

How to start making sense of everything?

Create a simplified mental model of the system.

Use the model to make sense of each git command.

Adjust model when it's not complex enough.

The git commit

Everything in git revolves around the "commit".

What is a commit?

```
commit eff9c198e32a828f610b93fad3a0f0eb63b3ded2 (HEAD -> main)
Author: Sergei Golubchik <serg@mariadb.org>
Date:   Thu Nov 14 19:09:01 2024 +0100

    11.8 branch
```

Commit Message

Commit Date

Commit Author

Commit Hash

The git commit

```
diff --git a/sql/create_options.cc b/sql/create_options.cc
index 3c010189f07..91211fbdda3 100644
--- a/sql/create_options.cc
+++ b/sql/create_options.cc
@@ -266,8 +266,8 @@ bool extend_option_list(THD* thd, st_plugin_int *plugin, bool create,
                                        value, opt->type != HA_OPTION_TYPE_ULL);
             if (!extended)
             {
-              void *pos= *option_list ? &(last->next) : option_list;
-              thd->register_item_tree_change((Item**)pos);
+              if (*option_list)
+                thd->register_item_tree_change((Item**)&(last->next));
              extended= true;
             }
             val->link(option_list, &last);
```

The git branch

A branch is a pointer to a commit.

Whenever a new commit gets created, the current working branch gets updated to the new commit id.

Rule #1: Basic Commit Rules

- Every commit should compile.
- Every commit should have all tests pass.
- Every commit should be a self-contained, logically valid change.
- In short: "DO NOT BREAK THE BUILD!"

Rule #2: Commit content

- Commit often, small changes. (Respect Rule #1)

- Make commit messages relevant.
  - One should be able to tell roughly what a commit does without reading the code.
  - One should be able to understand the reasoning for a commit from its message.

Rule #2: Commit content

Commit messages must follow the pattern:

- Commit title (max 80ish chars)
- Empty line
- Commit message (max 80 chars)

- When making a commit that fixes a particular Jira Bug, the format should be:
  - MDEV-XXXXX <MDEV-XXXXX-Title>

  - Tools that work with git expect this format. (Jira)
  - Makes life easier for everybody.

Rule #3: Branches are cheap, use them

```
$ git checkout main


$ git pull main   # Now we are up to date

$ git checkout -b bb-10.6-MDEV-XXXXX

# Now we are working on a separate branch.
```

Rule #4: Prefer Rebasing Over Merging

When working on a bugfix for a certain version, somebody may push something in the meantime.

When trying to merge to the main branch, you will need to update.

If you don't rebase, an extra merge commit will be generated.

Rule #4: Prefer Rebasing Over Merging

Merge commits are usually not desireable

Integrate a large set of changes directly.

Difficult to pinpoint which change causes a bug.

Merges contains all conflict resolution information. => Impossible to read diffs.

Rule #4: Prefer Rebasing Over Merging

```
$ git checkout main

$ git pull main   # Now we are up to date

$ git checkout -b bb-main-MDEV-XXXXX

$ git rebase main

$ git checkout main

$ git merge bb-main-MDEV-XXXXX

# This will just update main branch pointer
# to be the same as bb-main-MDEV-XXXXX
```

Rule #5: Rebase interactive to clean-up

When your commit history is "dirty"

Rewrite it with git rebase --interactive

- Reorder commits
- Change commit messages
- Merge multiple commits into one
- Drop unneeded commits

Try to not break Rule #1 && Rule #2
Commits must be self contained & not break tests.
Each commit must compile and be a standalone logical change.

Thank you!

Contact details:

vicentiu@mariadb.org

About:

https://mariadb.org/vicentiu