MySQL Development Cycle

Abstract

This document explains the MySQL Server development cycle. The purpose of the document is to facilitate community involvement, for example by providing feedback on pre-releases and making contributions to upcoming releases, through explaining how MySQL Server versions are developed.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2018-06-08 (revision: 516)

Table of Contents

1 Preface and Legal Notices

This document explains the MySQL Server development cycle. The purpose of the document is to facilitate community involvement, for example by providing feedback on pre-releases and making contributions to upcoming releases, through explaining how MySQL Server versions are developed.

Legal Notices

2 Feature Development

Feature development happens as follows:

- A MySQL feature is specified in a Worklog entry.
- The Worklog entry undergoes specification, design, architecture and QA reviews (but not necessarily in a strict sequence).
- The MySQL feature is implemented in a **feature tree**.
- Feature trees are created from and kept in sync with the MySQL main development tree, which is called **TRUNK**.
- When a feature has been implemented, it undergoes a code review.
- When the code review is done, the feature tree is handed over to QA (quality assurance).
- QA tests the feature, the implementor fixes bugs, and QA eventually “signs off” the feature.
- Once the feature is signed off, it is merged into TRUNK.

This way, TRUNK will accumulate features and bug fixes over time. Extensive regression testing is performed on TRUNK all the time, keeping TRUNK close to Release Candidate (RC) quality at all times.

3 Feature Testing

New features in MySQL are developed and tested in separate feature trees before they are pushed to TRUNK. Quality goals for new features are the following:

- Complete functional and nonfunctional test coverage of changed and new functionality
- No regressions
- At least more than 80% code coverage

QA involvement starts as soon as the requirements and specifications of the feature are finalized by the development team. QA reviews available documents and provides feedback on the design, usability, testability, etc. A discussion follows with the developer and changes are made as needed to ensure that the feature can be tested.

Once the specifications and requirements are acceptable, QA creates the test plan which documents all scenarios that are to be tested. This includes standalone tests, integration tests, nonfunctional tests, etc. The test plan is reviewed by developers and peer QA team members.

While the developers are writing the product code, QA engineers start working on the automated tests, test infrastructure improvements, etc. The final round of testing starts after the feature has passed code reviews. This phase can last anywhere between a few days to months, depending on the complexity of the feature, stability of the code, number of bugs found, etc.

Features get signed off when the following conditions are met:

- No known bugs in the new feature – This is very strictly enforced and even minor bugs are not allowed. We believe that bugs are easiest to fix when the code is new, and hence this can help us deliver features that are of high quality.
- No known regressions – A feature gets developed on a tree which gets tested regularly through a continuous integration testing tool. Any regressions are detected and fixed before signoff.
- Acceptable code coverage numbers – A code coverage report is generated for the changed lines of code and the minimum expected coverage is 80%. Most features have a coverage of more than 90%. Any uncovered lines of code are analyzed and, wherever possible, new tests are added to increase code coverage.
- All new tests are added to the automated regression suite.

4 Performance Testing

We execute performance testing to assure that the performance profile of the products does not regress. Performance testing is done all throughout the development process (in the current TRUNK branch), and focuses on throughput testing and response time testing.

Throughput Testing

The focus of throughput testing are high concurrency scenarios, with concurrent connections ranging from 8 to 1024. Testing is performed with open source applications like sysbench, with an emphasis on simple queries and simple OLTP transactions, but also including custom workloads. Other tests offer a much deeper look into complex OLTP throughput. These tests use short durations per test (5 to 10 minutes per test), varying server configuration options, and varying dataset sizes for both batched and atomic transaction types. The primary purpose of those tests is to expose bottlenecks, eliminate regressions, and benchmark the throughput. Results are archived in a database in order to facilitate retrieval for comparisons, and automated comparisons are performed to standardize evaluation.

Response Time Testing

Response time testing is done for single threads, because the main focus is to ensure consistency regarding query plan and execution time performance. An open source standardized test is used to

improve and guard against regression in the parser or optimizer. We use two different configurations, one of them performing CPU-bound testing, and the other performing I/O-bound testing. Results are automatically compared with results from previous tests, and archived in a database. From those data, summaries and reports are created.

Apart from testing new features, all the above tests are performed in regular intervals (daily and weekly). These tests are run on Linux and Windows platforms. For interval testing, automation is used to pull the current code, execute testing, compare against previously established baselines, and report results. If any of the results exceed preset thresholds, this is tracked as a failure, and prompts further investigation. That research may reveal that the failure is due to recently introduced changes, in which case the development teams are alerted, who will either fix the issue or revert the change. For all tests, baselines are established during milestone revisions, and used to compare only against the same server, framework revision, server configuration, and environment. At the end of the development cycle, all tests are repeated with larger data sets on the release candidate to catch any regressions that were caused by last minute code changes, and any performance regressions caused by code integration.

5 Lab Releases

A Lab release is a **preview** of one or more features under development which have normally not been integrated into TRUNK yet. Lab release features are not part of the last DMR (development milestone release). Lab releases are created when the MySQL community voices interest in evaluating early work, and they are **snapshots** of the feature tree at a given point in time. Lab releases can have beta quality, or they can be available only on a limited set of platforms, or documentation can not be available. Finally, there is no guarantee that the features in a lab release will appear in future DMR or GA releases. Lab releases can be found at labs.mysql.com.

6 Development Milestone Releases

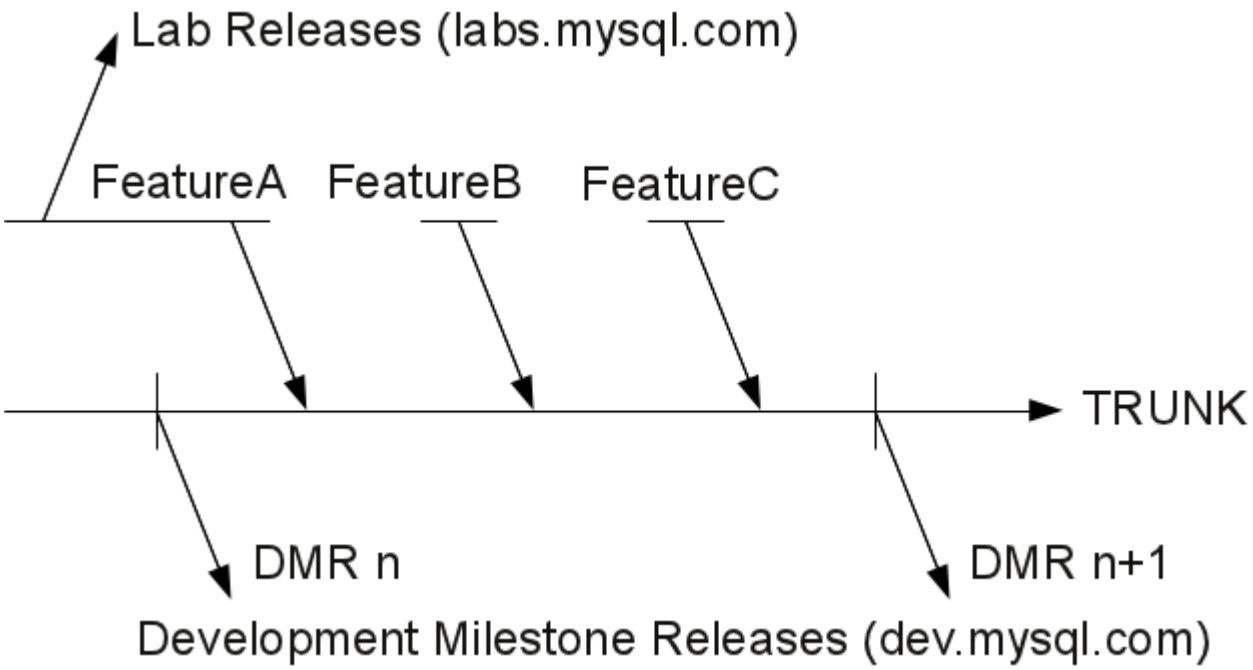
A development milestone release (DMR) is produced from TRUNK every 3-6 months and includes everything that is new in TRUNK since the previous DMR. The DMRs are made available on all supported platforms. The goal is to produce DMRs with RC (release candidate) quality.

The purpose of a DMR is to “release often”, as a service for early adopters, and to collect feedback from customers and users. That feedback is used to improve the next DMR. DMRs can be found on the MySQL Developer Zone.

Development milestone releases are produced as follows:

- A release manager sets a DMR cut-off date for new features.
- The DMR will include all features that make it to TRUNK before the cut-off date.
- Features are developed and tested on independent feature trees and pushed to TRUNK only once approved by QA.
- After all features are pushed, TRUNK is locked and automated regression tests are run over the weekend.
- QA signs off TRUNK once test runs are green.
- The DMR is cloned off and put in a separate tree, and TRUNK is opened up again for normal work.
- QA runs extended tests on the cloned-off DMR; bugs found in the process are fixed, and eventually QA approves the DMR.
- The DMR is released and announced.

Figure 1 Milestone releases



7 General Availability (GA) Releases

GA releases are releases recommended for production systems. The overall goal is to produce a new GA release every 18 to 24 months. GA releases can be found on the MySQL Developer Zone.

GA releases are produced as follows:

- A GA release is based on a DMR.
- The selected DMR undergoes additional testing and bug fixing; then a first RC (release candidate) is produced and made public.
- The release candidate undergoes evaluation by customers and users; this can cause a need for more RC cycles until GA quality will be reached.
- The GA release is released and announced.

Figure 2 GA Releases