Building MySQL from Source

Abstract

This is the Building MySQL from Source extract from the MySQL 8.0 Reference Manual.

For legal information, see the Legal Notices.

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

Document generated on: 2025-07-02 (revision: 82696)

Table of Contents

Preface and Legal Notices

This is the Building MySQL from Source extract from the MySQL 8.0 Reference Manual.

Licensing information—MySQL 8.0. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 8.0, see the MySQL 8.0 Commercial Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 8.0, see the MySQL 8.0 Community Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 1997, 2025, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Chapter 1 Installing MySQL from Source

Building MySQL from the source code enables you to customize build parameters, compiler optimizations, and installation location. For a list of systems on which MySQL is known to run, see https://www.mysql.com/support/supportedplatforms/database.html.

Before you proceed with an installation from source, check whether Oracle produces a precompiled binary distribution for your platform and whether it works for you. We put a great deal of effort into ensuring that our binaries are built with the best possible options for optimal performance. Instructions for installing binary distributions are available in Installing MySQL on Unix/Linux Using Generic Binaries.

If you are interested in building MySQL from a source distribution using build options the same as or similar to those use by Oracle to produce binary distributions on your platform, obtain a binary distribution, unpack it, and look in the `docs/INFO_BIN` file, which contains information about how that MySQL distribution was configured and compiled.

> **Warning**
>
> Building MySQL with nonstandard options may lead to reduced functionality, performance, or security.

The MySQL source code contains internal documentation written using Doxygen. The generated Doxygen content is available at https://dev.mysql.com/doc/index-other.html. It is also possible to generate this content locally from a MySQL source distribution using the instructions at Generating MySQL Doxygen Documentation Content.

Chapter 2 Installing MySQL Using a Standard Source Distribution

To install MySQL from a standard source distribution:

1. Verify that your system satisfies the tool requirements listed at Source Installation Prerequisites.

2. Obtain a distribution file using the instructions in How to Get MySQL.

3. Configure, build, and install the distribution using the instructions in this section.

4. Perform postinstallation procedures using the instructions in Postinstallation Setup and Testing.

MySQL uses `CMake` as the build framework on all platforms. The instructions given here should enable you to produce a working installation. For additional information on using `CMake` to build MySQL, see How to Build MySQL Server with CMake.

If you start from a source RPM, use the following command to make a binary RPM that you can install. If you do not have `rpmbuild`, use `rpm` instead.

```
$> rpmbuild --rebuild --clean MySQL-VERSION.src.rpm
```

The result is one or more binary RPM packages that you install as indicated in Installing MySQL on Linux Using RPM Packages from Oracle.

The sequence for installation from a compressed `tar` file or Zip archive source distribution is similar to the process for installing from a generic binary distribution (see Installing MySQL on Unix/Linux Using Generic Binaries), except that it is used on all platforms and includes steps to configure and compile the distribution. For example, with a compressed `tar` file source distribution on Unix, the basic installation command sequence looks like this:

```
# Preconfiguration setup
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
# Beginning of source-build specific instructions
$> tar zxvf mysql-VERSION.tar.gz
$> cd mysql-VERSION
$> mkdir bld
$> cd bld
$> cmake ..
$> make
$> make install
# End of source-build specific instructions
# Postinstallation setup
$> cd /usr/local/mysql
$> mkdir mysql-files
$> chown mysql:mysql mysql-files
$> chmod 750 mysql-files
$> bin/mysqld --initialize --user=mysql
$> bin/mysql_ssl_rsa_setup
$> bin/mysqld_safe --user=mysql &
# Next command is optional
$> cp support-files/mysql.server /etc/init.d/mysql.server
```

A more detailed version of the source-build specific instructions is shown following.

> **Note**
>
> The procedure shown here does not set up any passwords for MySQL accounts. After following the procedure, proceed to Postinstallation Setup and Testing, for postinstallation setup and testing.

- Perform Preconfiguration Setup

- Obtain and Unpack the Distribution

- Configure the Distribution

- Build the Distribution

- Install the Distribution

- Perform Postinstallation Setup

Perform Preconfiguration Setup

On Unix, set up the `mysql` user that owns the database directory and that should be used to run and execute the MySQL server, and the group to which this user belongs. For details, see Create a mysql User and Group. Then perform the following steps as the `mysql` user, except as noted.

Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it.

Obtain a distribution file using the instructions in How to Get MySQL.

Unpack the distribution into the current directory:

- To unpack a compressed `tar` file, `tar` can decompress and unpack the distribution if it has `z` option support:

```
$> tar zxvf mysql-VERSION.tar.gz
```

If your `tar` does not have `z` option support, use `gunzip` to decompress the distribution and `tar` to unpack it:

```
$> gunzip < mysql-VERSION.tar.gz | tar xvf -
```

Alternatively, `CMake` can decompress and unpack the distribution:

```
$> cmake -E tar zxvf mysql-VERSION.tar.gz
```

- To unpack a Zip archive, use `WinZip` or another tool that can read `.zip` files.

Unpacking the distribution file creates a directory named `mysql-VERSION`.

Configure the Distribution

Change location into the top-level directory of the unpacked distribution:

```
$> cd mysql-VERSION
```

Build outside of the source tree to keep the tree clean. If the top-level source directory is named `mysql-src` under your current working directory, you can build in a directory named `build` at the same level. Create the directory and go there:

```
$> mkdir bld
$> cd bld
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
$> cmake ../mysql-src
```

The build directory need not be outside the source tree. For example, you can build in a directory named `build` under the top-level source tree. To do this, starting with `mysql-src` as your current working directory, create the directory `build` and then go there:

```
$> mkdir build
$> cd build
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
$> cmake ..
```

If you have multiple source trees at the same level (for example, to build multiple versions of MySQL), the second strategy can be advantageous. The first strategy places all build directories at the same level, which requires that you choose a unique name for each. With the second strategy, you can use the same name for the build directory within each source tree. The following instructions assume this second strategy.

On Windows, specify the development environment. For example, the following commands configure MySQL for 32-bit or 64-bit builds, respectively:

```
$> cmake .. -G "Visual Studio 12 2013"
$> cmake .. -G "Visual Studio 12 2013 Win64"
```

On macOS, to use the Xcode IDE:

```
$> cmake .. -G Xcode
```

When you run `Cmake`, you might want to add options to the command line. Here are some examples:

- `-DBUILD_CONFIG=mysql_release`: Configure the source with the same build options used by Oracle to produce binary distributions for official MySQL releases.

- `-DCMAKE_INSTALL_PREFIX=dir_name`: Configure the distribution for installation under a particular location.

- `-DCPACK_MONOLITHIC_INSTALL=1`: Cause `make package` to generate a single installation file rather than multiple files.

- `-DWITH_DEBUG=1`: Build the distribution with debugging support.

For a more extensive list of options, see Chapter 4, *MySQL Source-Configuration Options*.

To list the configuration options, use one of the following commands:

```
$> cmake .. -L    # overview
$> cmake .. -LH   # overview with help text
$> cmake .. -LAH  # all params with help text
$> ccmake ..      # interactive display
```

If `CMake` fails, you might need to reconfigure by running it again with different options. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.

- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run these commands in the build directory on Unix before re-running `CMake`:

```
$> make clean
$> rm CMakeCache.txt
```

Or, on Windows:

```
$> devenv MySQL.sln /clean
$> del CMakeCache.txt
```

Before asking on the MySQL Community Slack, check the files in the `CMakeFiles` directory for useful information about the failure. To file a bug report, please use the instructions in How to Report Bugs or Problems.

Build the Distribution

On Unix:

```
$> make
$> make VERBOSE=1
```

The second command sets `VERBOSE` to show the commands for each compiled source.

Use `gmake` instead on systems where you are using GNU `make` and it has been installed as `gmake`.

On Windows:

```
$> devenv MySQL.sln /build RelWithDebInfo
```

If you have gotten to the compilation stage, but the distribution does not build, see Chapter 5, *Dealing with Problems Compiling MySQL*, for help. If that does not solve the problem, please enter it into our bugs database using the instructions given in How to Report Bugs or Problems. If you have installed the latest versions of the required tools, and they crash trying to process our configuration files, please report that also. However, if you get a `command not found` error or a similar problem for required tools, do not report it. Instead, make sure that all the required tools are installed and that your `PATH` variable is set correctly so that your shell can find them.

Install the Distribution

On Unix:

```
$> make install
```

This installs the files under the configured installation directory (by default, `/usr/local/mysql`). You might need to run the command as `root`.

To install in a specific directory, add a `DESTDIR` parameter to the command line:

```
$> make install DESTDIR="/opt/mysql"
```

Alternatively, generate installation package files that you can install where you like:

```
$> make package
```

This operation produces one or more `.tar.gz` files that can be installed like generic binary distribution packages. See Installing MySQL on Unix/Linux Using Generic Binaries. If you run `CMake` with `-DCPACK_MONOLITHIC_INSTALL=1`, the operation produces a single file. Otherwise, it produces multiple files.

On Windows, generate the data directory, then create a `.zip` archive installation package:

```
$> devenv MySQL.sln /build RelWithDebInfo /project initial_database
$> devenv MySQL.sln /build RelWithDebInfo /project package
```

You can install the resulting `.zip` archive where you like. See Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive.

Perform Postinstallation Setup

The remainder of the installation process involves setting up the configuration file, creating the core databases, and starting the MySQL server. For instructions, see Postinstallation Setup and Testing.

Note

The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in Postinstallation Setup and Testing.

Chapter 3 Installing MySQL Using a Development Source Tree

This section describes how to install MySQL from the latest development source code, which is hosted on GitHub. To obtain the MySQL Server source code from this repository hosting service, you can set up a local MySQL Git repository.

On GitHub, MySQL Server and other MySQL projects are found on the MySQL page. The MySQL Server project is a single repository that contains branches for several MySQL series.

- Prerequisites for Installing from Development Source

- Setting Up a MySQL Git Repository

Prerequisites for Installing from Development Source

To install MySQL from a development source tree, your system must satisfy the tool requirements listed at Source Installation Prerequisites.

Setting Up a MySQL Git Repository

To set up a MySQL Git repository on your machine:

1. Clone the MySQL Git repository to your machine. The following command clones the MySQL Git repository to a directory named `mysql-server`. The initial download may take some time to complete, depending on the speed of your connection.

```
$> git clone https://github.com/mysql/mysql-server.git
Cloning into 'mysql-server'...
remote: Counting objects: 1198513, done.
remote: Total 1198513 (delta 0), reused 0 (delta 0), pack-reused 1198513
Receiving objects: 100% (1198513/1198513), 1.01 GiB | 7.44 MiB/s, done.
Resolving deltas: 100% (993200/993200), done.
Checking connectivity... done.
Checking out files: 100% (25510/25510), done.
```

2. When the clone operation completes, the contents of your local MySQL Git repository appear similar to the following:

```
~> cd mysql-server
~/mysql-server> ls
client            extra            mysys              storage
cmake             include          packaging          strings
CMakeLists.txt    INSTALL          plugin             support-files
components        libbinlogevents  README             testclients
config.h.cmake    libchangestreams router             unittest
configure.cmake   libmysql         run_doxygen.cmake  utilities
Docs              libservices      scripts            VERSION
Doxyfile-ignored  LICENSE          share              vio
Doxyfile.in       man              sql                win
doxygen_resources mysql-test       sql-common
```

3. Use the `git branch -r` command to view the remote tracking branches for the MySQL repository.

```
~/mysql-server> git branch -r
  origin/5.7
  origin/8.0
  origin/HEAD -> origin/trunk
  origin/cluster-7.4
  origin/cluster-7.5
  origin/cluster-7.6
  origin/trunk
```

4. To view the branch that is checked out in your local repository, issue the `git branch` command. When you clone the MySQL Git repository, the latest MySQL branch is checked out automatically. The asterisk identifies the active branch.

```
~/mysql-server$ git branch
* trunk
```

5. To check out an earlier MySQL branch, run the `git checkout` command, specifying the branch name. For example, to check out the MySQL 5.7 branch:

```
~/mysql-server$ git checkout 5.7
Checking out files: 100% (9600/9600), done.
Branch 5.7 set up to track remote branch 5.7 from origin.
Switched to a new branch '5.7'
```

6. To obtain changes made after your initial setup of the MySQL Git repository, switch to the branch you want to update and issue the `git pull` command:

```
~/mysql-server$ git checkout 8.0
~/mysql-server$ git pull
```

To examine the commit history, use the `git log` command:

```
~/mysql-server$ git log
```

You can also browse commit history and source code on the GitHub MySQL site.

If you see changes or code that you have a question about, ask on MySQL Community Slack.

7. After you have cloned the MySQL Git repository and have checked out the branch you want to build, you can build MySQL Server from the source code. Instructions are provided in Chapter 2, *Installing MySQL Using a Standard Source Distribution*, except that you skip the part about obtaining and unpacking the distribution.

Be careful about installing a build from a distribution source tree on a production machine. The installation command may overwrite your live release installation. If you already have MySQL installed and do not want to overwrite it, run `CMake` with values for the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options different from those used by your production server. For additional information about preventing multiple servers from interfering with each other, see Running Multiple MySQL Instances on One Machine.

Play hard with your new installation. For example, try to make new features crash. Start by running `make test`. See The MySQL Test Suite.

Chapter 4 MySQL Source-Configuration Options

The `CMake` program provides a great deal of control over how you configure a MySQL source distribution. Typically, you do this using options on the `CMake` command line. For information about options supported by `CMake`, run either of these commands in the top-level source directory:

```
$> cmake . -LH
$> ccmake .
```

You can also affect `CMake` using certain environment variables. See Environment Variables.

For boolean options, the value may be specified as `1` or `ON` to enable the option, or as `0` or `OFF` to disable the option.

Many options configure compile-time defaults that can be overridden at server startup. For example, the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options that configure the default installation base directory location, TCP/IP port number, and Unix socket file can be changed at server startup with the `--basedir`, `--port`, and `--socket` options for `mysqld`. Where applicable, configuration option descriptions indicate the corresponding `mysqld` startup option.

The following sections provide more information about `CMake` options.

- CMake Option Reference
- General Options
- Installation Layout Options
- Storage Engine Options
- Feature Options
- Compiler Flags
- CMake Options for Compiling NDB Cluster

CMake Option Reference

The following table shows the available `CMake` options. In the `Default` column, `PREFIX` stands for the value of the `CMAKE_INSTALL_PREFIX` option, which specifies the installation base directory. This value is used as the parent location for several of the installation subdirectories.

Table 4.1 MySQL Source-Configuration Option Reference (CMake)

Formats	Description	Default	Introduced	Removed
`ADD_GDB_INDEX`	Whether to enable generation of .gdb_index section in binaries		8.0.18	
`BUILD_CONFIG`	Use same build options as official releases			
`BUNDLE_RUNTIME_LIBRARIES`	Bundle runtime libraries with server MSI and Zip packages for Windows	`OFF`		
`CMAKE_BUILD_TYPE`	Type of build to produce	`RelWithDebInfo`		
`CMAKE_CXX_FLAGS`	Flags for C++ Compiler			

Formats	Description	Default	Introduced	Removed
CMAKE_C_FLAGS	Flags for C Compiler			
CMAKE_INSTALL_P...	Installation base directory	/usr/local/ mysql		
COMPILATION_COM...	Comment about compilation environment			
COMPILATION_COM...	Comment about compilation environment for use by mysqld		8.0.14	
COMPRESS_DEBUG_S...	Compress debug sections of binary executables	OFF	8.0.22	
CPACK_MONOLITHI...	Whether package build produces single file	OFF		
DEFAULT_CHARSET	The default server character set	utf8mb4		
DEFAULT_COLLATI...	The default server collation	utf8mb4_0900_ai_ci		
DISABLE_PSI_CON...	Exclude Performance Schema condition instrumentation	OFF		
DISABLE_PSI_DATA...	Exclude the performance schema data lock instrumentation	OFF		
DISABLE_PSI_ERR...	Exclude the performance schema server error instrumentation	OFF		
DISABLE_PSI_FIL...	Exclude Performance Schema file instrumentation	OFF		
DISABLE_PSI_IDL...	Exclude Performance Schema idle instrumentation	OFF		
DISABLE_PSI_MEMO...	Exclude Performance Schema memory instrumentation	OFF		
DISABLE_PSI_META...	Exclude Performance Schema metadata instrumentation	OFF		

Formats	Description	Default	Introduced	Removed
DISABLE_PSI_MUTEX	Exclude Performance Schema mutex instrumentation	OFF		
DISABLE_PSI_PS	Exclude the performance schema prepared statements	OFF		
DISABLE_PSI_RWLOCK	Exclude Performance Schema rwlock instrumentation	OFF		
DISABLE_PSI_SOCKET	Exclude Performance Schema socket instrumentation	OFF		
DISABLE_PSI_SP	Exclude Performance Schema stored program instrumentation	OFF		
DISABLE_PSI_STAGE	Exclude Performance Schema stage instrumentation	OFF		
DISABLE_PSI_STATEMENT	Exclude Performance Schema statement instrumentation	OFF		
DISABLE_PSI_STATEMENT_DIGEST	Exclude Performance Schema statements_digest instrumentation	OFF		
DISABLE_PSI_TABLE	Exclude Performance Schema table instrumentation	OFF		
DISABLE_PSI_THREAD	Exclude the performance schema thread instrumentation	OFF		
DISABLE_PSI_TRANSACTION	Exclude the performance schema transaction instrumentation	OFF		
DISABLE_SHARED	Do not build shared libraries, compile position-dependent code	OFF		8.0.18

Formats	Description	Default	Introduced	Removed
DOWNLOAD_BOOST	Whether to download the Boost library	OFF		
DOWNLOAD_BOOST_TIMEOUT	Timeout in seconds for downloading the Boost library	600		
ENABLED_LOCAL_INFILE	Whether to enable LOCAL for LOAD DATA	OFF		
ENABLED_PROFILING	Whether to enable query profiling code	ON		
ENABLE_DOWNLOADS	Whether to download optional files	OFF		8.0.26
ENABLE_EXPERIMENT_SYSVARS	Whether to enabled experimental InnoDB system variables	OFF		
ENABLE_GCOV	Whether to include gcov support			
ENABLE_GPROF	Enable gprof (optimized Linux builds only)	OFF		
FORCE_COLORED_OUTPUT	Whether to colorize compiler output	OFF	8.0.33	
FORCE_INSOURCE_BUILD	Whether to force an in-source build	OFF	8.0.14	
FORCE_UNSUPPORTED_COMPILER	Whether to permit unsupported compilers	OFF		
FPROFILE_GENERATE	Whether to generate profile guided optimization data	OFF	8.0.19	
FPROFILE_USE	Whether to use profile guided optimization data	OFF	8.0.19	
HAVE_PSI_MEMORY_INTERFACE	Enable performance schema memory tracing module for memory allocation functions used in dynamic storage of over-aligned types	OFF	8.0.26	
IGNORE_AIO_CHECK	With -DBUILD_CONFIG=mysql_release, ignore libaio check	OFF		

Formats	Description	Default	Introduced	Removed
`INSTALL_BINDIR`	User executables directory	`PREFIX/bin`		
`INSTALL_DOCDIR`	Documentation directory	`PREFIX/docs`		
`INSTALL_DOCREADMEDIR`	README file directory	`PREFIX`		
`INSTALL_INCLUDEDIR`	Header file directory	`PREFIX/include`		
`INSTALL_INFODIR`	Info file directory	`PREFIX/docs`		
`INSTALL_LAYOUT`	Select predefined installation layout	`STANDALONE`		
`INSTALL_LIBDIR`	Library file directory	`PREFIX/lib`		
`INSTALL_MANDIR`	Manual page directory	`PREFIX/man`		
`INSTALL_MYSQLKEYRINGDIR`	Directory for keyring_file plugin data file	`platform specific`		
`INSTALL_MYSQLSHAREDIR`	Shared data directory	`PREFIX/share`		
`INSTALL_MYSQLTESTDIR`	mysql-test directory	`PREFIX/mysql-test`		
`INSTALL_PKGCONFIGDIR`	Directory for mysqlclient.pc pkg-config file	`INSTALL_LIBDIR/pkgconfig`		
`INSTALL_PLUGINDIR`	Plugin directory	`PREFIX/lib/plugin`		
`INSTALL_PRIV_LIBDIR`	Installation private library directory		8.0.18	
`INSTALL_SBINDIR`	Server executable directory	`PREFIX/bin`		
`INSTALL_SECURE_FILE_PRIVDIR`	secure_file_priv default value	`platform specific`		
`INSTALL_SHAREDIR`	aclocal/mysql.m4 installation directory	`PREFIX/share`		
`INSTALL_STATIC_LIBRARIES`	Whether to install static libraries	`ON`		
`INSTALL_SUPPORTFILESDIR`	Extra support files directory	`PREFIX/support-files`		
`LINK_RANDOMIZE`	Whether to randomize order of symbols in mysqld binary	`OFF`		
`LINK_RANDOMIZE_SEED`	Seed value for LINK_RANDOMIZE option	`mysql`		

Formats	Description	Default	Introduced	Removed
MAX_INDEXES	Maximum indexes per table	64		
MEMCACHED_HOME	Path to memcached; obsolete	[none]		8.0.23
MSVC_CPPCHECK	Enable MSVC code analysis.	OFF	8.0.33	
MUTEX_TYPE	InnoDB mutex type	event		
MYSQLX_TCP_PORT	TCP/IP port number used by X Plugin	33060		
MYSQLX_UNIX_ADDR	Unix socket file used by X Plugin	/tmp/ mysqlx.sock		
MYSQL_DATADIR	Data directory			
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF		
MYSQL_PROJECT_NAME	Windows/macOS project name	MySQL		
MYSQL_TCP_PORT	TCP/IP port number	3306		
MYSQL_UNIX_ADDR	Unix socket file	/tmp/ mysql.sock		
NDB_UTILS_LINK_DYNAMIC	Cause NDB tools to be dynamically linked to ndbclient		8.0.22	
ODBC_INCLUDES	ODBC includes directory			
ODBC_LIB_DIR	ODBC library directory			
OPTIMIZER_TRACE	Whether to support optimizer tracing			
OPTIMIZE_SANITIZER_BUILDS	Whether to optimize sanitizer builds	ON	8.0.34	
REPRODUCIBLE_BUILD	Take extra care to create a build result independent of build location and time			
SHOW_SUPPRESSED_COMPILER_WARNINGS	Whether to show suppressed compiler warnings and not fail with -Werror.	OFF	8.0.30	
SYSCONFDIR	Option file directory			

Formats	Description	Default	Introduced	Removed
`SYSTEMD_PID_DIR`	Directory for PID file under systemd	`/var/run/ mysqld`		
`SYSTEMD_SERVICE_NAME`	Name of MySQL service under systemd	`mysqld`		
`TMPDIR`	tmpdir default value			
`USE_LD_GOLD`	Whether to use GNU gold linker	`ON`		8.0.31
`USE_LD_LLD`	Whether to use LLVM lld linker	`ON`	8.0.16	
`WIN_DEBUG_NO_INLINE`	Whether to disable function inlining	`OFF`		
`WITHOUT_SERVER`	Do not build the server; internal use only	`OFF`		
`WITHOUT_xxx_STORAGE_ENGINE`	Exclude storage engine xxx from build			
`WITH_ANT`	Path to Ant for building GCS Java wrapper			
`WITH_ASAN`	Enable AddressSanitizer	`OFF`		
`WITH_ASAN_SCOPE`	Enable AddressSanitizer -fsanitize-address-use-after-scope Clang flag	`OFF`		
`WITH_AUTHENTICATION_CLIENT_PLUGINS`	Enabled automatically if any corresponding server authentication plugins are built		8.0.26	
`WITH_AUTHENTICATION_LDAP`	Whether to report error if LDAP authentication plugins cannot be built	`OFF`		
`WITH_AUTHENTICATION_PAM`	Build PAM authentication plugin	`OFF`		
`WITH_AWS_SDK`	Location of Amazon Web Services software development kit			
`WITH_BOOST`	The location of the Boost library sources			

Formats	Description	Default	Introduced	Removed
WITH_BUILD_ID	On Linux systems, generate a unique build ID	ON	8.0.31	
WITH_BUNDLED_LIBEVENT	Use bundled libevent when building ndbmemcache; obsolete	ON		8.0.23
WITH_BUNDLED_MEMCACHED	Use bundled memcached when building ndbmemcache; obsolete	ON		8.0.23
WITH_CLASSPATH	Classpath to use when building MySQL Cluster Connector for Java. Default is an empty string.			
WITH_CLIENT_PROTOCOL_TRACING	Build client-side protocol tracing framework	ON		
WITH_CURL	Location of curl library			
WITH_DEBUG	Whether to include debugging support	OFF		
WITH_DEFAULT_COMPILER_OPTIONS	Whether to use default compiler options	ON		
WITH_DEFAULT_FEATURE_SET	Whether to use default feature set	ON		8.0.22
WITH_DEVELOPER_ENTITLEMENTS	Whether to add the 'get-task-allow' entitlement to all executables on macOS to generate a core dump in the event of an unexpected server halt	OFF	8.0.30	
WITH_EDITLINE	Which libedit/ editline library to use	bundled		
WITH_ERROR_INSERT	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF		

Formats	Description	Default	Introduced	Removed
`WITH_FIDO`	Type of FIDO library support	`bundled`	8.0.27	
`WITH_GMOCK`	Path to googlemock distribution			8.0.26
`WITH_ICU`	Type of ICU support	`bundled`		
`WITH_INNODB_EXTRA_DEBUG`	Whether to include extra debugging support for InnoDB.	`OFF`		
`WITH_INNODB_MEMCACHED`	Whether to generate memcached shared libraries.	`OFF`		
`WITH_JEMALLOC`	Whether to link with -ljemalloc	`OFF`	8.0.16	
`WITH_KEYRING_TEST`	Build the keyring test program	`OFF`		
`WITH_LIBEVENT`	Which libevent library to use	`bundled`		
`WITH_LIBWRAP`	Whether to include libwrap (TCP wrappers) support	`OFF`		
`WITH_LOCK_ORDER`	Whether to enable LOCK_ORDER tooling	`OFF`	8.0.17	
`WITH_LSAN`	Whether to run LeakSanitizer, without AddressSanitizer	`OFF`	8.0.16	
`WITH_LTO`	Enable link-time optimizer	`OFF`	8.0.13	
`WITH_LZ4`	Type of LZ4 library support	`bundled`		
`WITH_LZMA`	Type of LZMA library support	`bundled`		8.0.16
`WITH_MECAB`	Compiles MeCab			
`WITH_MSAN`	Enable MemorySanitizer	`OFF`		
`WITH_MSCRT_DEBUG`	Enable Visual Studio CRT memory leak tracing	`OFF`		
`WITH_MYSQLX`	Whether to disable X Protocol	`ON`		
`WITH_NDB`	Build MySQL NDB Cluster, including NDB storage	`OFF`	8.0.31	

Formats	Description	Default	Introduced	Removed
	engine and all NDB programs			
WITH_NDBAPI_EXAMPLES	Build API example programs.	OFF		
WITH_NDBCLUSTER	NDB 8.0.30 and earlier: Build NDB storage engine. NDB 8.0.31 and later: Deprecated; use WITH_NDB instead	OFF		
WITH_NDBCLUSTER_STORAGE_ENGINE	Prior to NDB 8.0.31, this was for internal use only. NDB 8.0.31 and later: toggles (only) inclusion of NDBCLUSTER storage engine	ON		
WITH_NDBMTD	Build multithreaded data node binary	ON		
WITH_NDB_DEBUG	Produce a debug build for testing or troubleshooting.	OFF		
WITH_NDB_JAVA	Enable building of Java and ClusterJ support. Enabled by default. Supported in MySQL Cluster only.	ON		
WITH_NDB_PORT	Default port used by a management server built with this option. If this option was not used to build it, the management server's default port is 1186.	[none]		
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_NUMA	Set NUMA memory allocation policy			
WITH_PACKAGE_FLAGS	For flags typically used for RPM/DEB packages, whether to add them to standalone builds on those platforms		8.0.26	
WITH_PLUGIN_NDBCLUSTER	For internal use; may not work as		8.0.13	8.0.31

Formats	Description	Default	Introduced	Removed
	expected in all circumstances. Instead, users should employ WITH_NDBCLUSTER			
`WITH_PROTOBUF`	Which Protocol Buffers package to use	`bundled`		
`WITH_RAPID`	Whether to build rapid development cycle plugins	`ON`		
`WITH_RAPIDJSON`	Type of RapidJSON support	`bundled`	8.0.13	
`WITH_RE2`	Type of RE2 library support	`bundled`		8.0.18
`WITH_ROUTER`	Whether to build MySQL Router	`ON`	8.0.16	
`WITH_SASL`	Internal use only			
`WITH_SSL`	Type of SSL support	`system`		
`WITH_SYSTEMD`	Enable installation of systemd support files	`OFF`		
`WITH_SYSTEMD_DEBUG`	Enable additional systemd debug information	`OFF`	8.0.22	
`WITH_SYSTEM_LIBS`	Set system value of library options not set explicitly	`OFF`		
`WITH_TCMALLOC`	Whether to link with -ltcmalloc. BUNDLED is supported on Linux only	`OFF`	8.0.22	
`WITH_TEST_TRACE_PLUGIN`	Build test protocol trace plugin	`OFF`		
`WITH_TSAN`	Enable ThreadSanitizer	`OFF`		
`WITH_UBSAN`	Enable Undefined Behavior Sanitizer	`OFF`		
`WITH_UNIT_TESTS`	Compile MySQL with unit tests	`ON`		
`WITH_UNIXODBC`	Enable unixODBC support	`OFF`		
`WITH_VALGRIND`	Whether to compile in Valgrind header files	`OFF`		

Formats	Description	Default	Introduced	Removed
`WITH_WIN_JEMALLOC`	Path to directory containing jemalloc.dll		8.0.29	
`WITH_ZLIB`	Type of zlib support	`bundled`		
`WITH_ZSTD`	Type of zstd support	`bundled`	8.0.18	
`WITH_xxx_STORAGE_ENGINE`	Compile storage engine xxx statically into server			

General Options

- `-DBUILD_CONFIG=mysql_release`

  This option configures a source distribution with the same build options used by Oracle to produce binary distributions for official MySQL releases.

- `-DWITH_BUILD_ID=bool`

  On Linux systems, generates a unique build ID which is used as the value of the `build_id` system variable and written to the MySQL server log on startup. Set this option to `OFF` to disable this feature.

  Added in MySQL 8.0.31, this option has no effect on platforms other than Linux.

- `-DBUNDLE_RUNTIME_LIBRARIES=bool`

  Whether to bundle runtime libraries with server MSI and Zip packages for Windows.

- `-DCMAKE_BUILD_TYPE=type`

  The type of build to produce:

  - `RelWithDebInfo`: Enable optimizations and generate debugging information. This is the default MySQL build type.

  - `Release`: Enable optimizations but omit debugging information to reduce the build size. This build type was added in MySQL 8.0.13.

  - `Debug`: Disable optimizations and generate debugging information. This build type is also used if the `WITH_DEBUG` option is enabled. That is, `-DWITH_DEBUG=1` has the same effect as `-DCMAKE_BUILD_TYPE=Debug`.

  The option values `None` and `MinSizeRel` are not supported.

- `-DCPACK_MONOLITHIC_INSTALL=bool`

  This option affects whether the `make package` operation produces multiple installation package files or a single file. If disabled, the operation produces multiple installation package files, which may be useful if you want to install only a subset of a full MySQL installation. If enabled, it produces a single file for installing everything.

- `-DFORCE_INSOURCE_BUILD=bool`

  Defines whether to force an in-source build. Out-of-source builds are recommended, as they permit multiple builds from the same source, and cleanup can be performed quickly by removing the build directory. To force an in-source build, invoke `CMake` with `-DFORCE_INSOURCE_BUILD=ON`.

- `-DFORCE_COLORED_OUTPUT=`*`bool`*

  Defines whether to enable colorized compiler output for `gcc` and `clang` when compiling on the command line. Defaults to `OFF`.

Installation Layout Options

The `CMAKE_INSTALL_PREFIX` option indicates the base installation directory. Other options with names of the form `INSTALL_`*`xxx`* that indicate component locations are interpreted relative to the prefix and their values are relative pathnames. Their values should not include the prefix.

- `-DCMAKE_INSTALL_PREFIX=`*`dir_name`*

  The installation base directory.

  This value can be set at server startup using the `--basedir` option.

- `-DINSTALL_BINDIR=`*`dir_name`*

  Where to install user programs.

- `-DINSTALL_DOCDIR=`*`dir_name`*

  Where to install documentation.

- `-DINSTALL_DOCREADMEDIR=`*`dir_name`*

  Where to install `README` files.

- `-DINSTALL_INCLUDEDIR=`*`dir_name`*

  Where to install header files.

- `-DINSTALL_INFODIR=`*`dir_name`*

  Where to install Info files.

- `-DINSTALL_LAYOUT=`*`name`*

  Select a predefined installation layout:

  - `STANDALONE`: Same layout as used for `.tar.gz` and `.zip` packages. This is the default.

  - `RPM`: Layout similar to RPM packages.

  - `SVR4`: Solaris package layout.

  - `DEB`: DEB package layout (experimental).

  You can select a predefined layout but modify individual component installation locations by specifying other options. For example:

  ```
  cmake . -DINSTALL_LAYOUT=SVR4 -DMYSQL_DATADIR=/var/mysql/data
  ```

  The `INSTALL_LAYOUT` value determines the default value of the `secure_file_priv`, `keyring_encrypted_file_data`, and `keyring_file_data` system variables. See the descriptions of those variables in Server System Variables, and Keyring System Variables.

- `-DINSTALL_LIBDIR=`*`dir_name`*

  Where to install library files.

- `-DINSTALL_MANDIR=`*`dir_name`*

Where to install manual pages.

- `-DINSTALL_MYSQLKEYRINGDIR=`*`dir_path`*

The default directory to use as the location of the `keyring_file` plugin data file. The default value is platform specific and depends on the value of the `INSTALL_LAYOUT CMake` option; see the description of the `keyring_file_data` system variable in Server System Variables.

- `-DINSTALL_MYSQLSHAREDIR=`*`dir_name`*

Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=`*`dir_name`*

Where to install the `mysql-test` directory. To suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_MYSQLTESTDIR=`).

- `-DINSTALL_PKGCONFIGDIR=`*`dir_name`*

The directory in which to install the `mysqlclient.pc` file for use by `pkg-config`. The default value is `INSTALL_LIBDIR/pkgconfig`, unless `INSTALL_LIBDIR` ends with `/mysql`, in which case that is removed first.

- `-DINSTALL_PLUGINDIR=`*`dir_name`*

The location of the plugin directory.

This value can be set at server startup with the `--plugin_dir` option.

- `-DINSTALL_PRIV_LIBDIR=`*`dir_name`*

The location of the dynamic library directory.

Default location. For RPM builds, this is `/usr/lib64/mysql/private/`, for DEB it is `/usr/lib/mysql/private/`, and for TAR it is `lib/private/`.

Protobuf. Because this is a private location, the loader (such as `ld-linux.so` on Linux) may not find the `libprotobuf.so` files without help. To guide the loader, `RPATH=$ORIGIN/../$INSTALL_PRIV_LIBDIR` is added to `mysqld` and `mysqlxtest`. This works for most cases but when using the Resource Group feature, `mysqld` is `setsuid`, and the loader ignores any `RPATH` which contains `$ORIGIN`. To overcome this, an explicit full path to the directory is set in the DEB and RPM versions of `mysqld`, since the target destination is known. For tarball installs, patching of `mysqld` with a tool like `patchelf` is required.

This option was added in MySQL 8.0.18.

- `-DINSTALL_SBINDIR=`*`dir_name`*

Where to install the `mysqld` server.

- `-DINSTALL_SECURE_FILE_PRIVDIR=`*`dir_name`*

The default value for the `secure_file_priv` system variable. The default value is platform specific and depends on the value of the `INSTALL_LAYOUT CMake` option; see the description of the `secure_file_priv` system variable in Server System Variables.

- `-DINSTALL_SHAREDIR=`*`dir_name`*

Where to install `aclocal/mysql.m4`.

- `-DINSTALL_STATIC_LIBRARIES=`*`bool`*

  Whether to install static libraries. The default is `ON`. If set to `OFF`, these library files are not installed: `libmysqlclient.a`, `libmysqlservices.a`.

- `-DINSTALL_SUPPORTFILESDIR=`*`dir_name`*

  Where to install extra support files.

- `-DLINK_RANDOMIZE=`*`bool`*

  Whether to randomize the order of symbols in the `mysqld` binary. The default is `OFF`. This option should be enabled only for debugging purposes.

- `-DLINK_RANDOMIZE_SEED=`*`val`*

  Seed value for the `LINK_RANDOMIZE` option. The value is a string. The default is `mysql`, an arbitrary choice.

- `-DMYSQL_DATADIR=`*`dir_name`*

  The location of the MySQL data directory.

  This value can be set at server startup with the `--datadir` option.

- `-DODBC_INCLUDES=`*`dir_name`*

  The location of the ODBC includes directory, which may be used while configuring Connector/ODBC.

- `-DODBC_LIB_DIR=`*`dir_name`*

  The location of the ODBC library directory, which may be used while configuring Connector/ODBC.

- `-DSYSCONFDIR=`*`dir_name`*

  The default `my.cnf` option file directory.

  This location cannot be set at server startup, but you can start the server with a given option file using the `--defaults-file=`*`file_name`* option, where *`file_name`* is the full path name to the file.

- `-DSYSTEMD_PID_DIR=`*`dir_name`*

  The name of the directory in which to create the PID file when MySQL is managed by systemd. The default is `/var/run/mysqld`; this might be changed implicitly according to the `INSTALL_LAYOUT` value.

  This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DSYSTEMD_SERVICE_NAME=`*`name`*

  The name of the MySQL service to use when MySQL is managed by `systemd`. The default is `mysqld`; this might be changed implicitly according to the `INSTALL_LAYOUT` value.

  This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DTMPDIR=`*`dir_name`*

  The default location to use for the `tmpdir` system variable. If unspecified, the value defaults to `P_tmpdir` in `<stdio.h>`.

Storage Engine Options

Storage engines are built as plugins. You can build a plugin as a static module (compiled into the server) or a dynamic module (built as a dynamic library that must be installed into the server using the `INSTALL PLUGIN` statement or the `--plugin-load` option before it can be used). Some plugins might not support static or dynamic building.

The `InnoDB`, `MyISAM`, `MERGE`, `MEMORY`, and `CSV` engines are mandatory (always compiled into the server) and need not be installed explicitly.

To compile a storage engine statically into the server, use `-DWITH_engine_STORAGE_ENGINE=1`. Some permissible *engine* values are `ARCHIVE`, `BLACKHOLE`, `EXAMPLE`, and `FEDERATED`. Examples:

```
-DWITH_ARCHIVE_STORAGE_ENGINE=1
-DWITH_BLACKHOLE_STORAGE_ENGINE=1
```

To build MySQL with support for NDB Cluster, use the `WITH_NDB` option. (*NDB 8.0.30 and earlier*: Use `WITH_NDBCLUSTER`.)

> **Note**
>
> It is not possible to compile without Performance Schema support. If it is desired to compile without particular types of instrumentation, that can be done with the following `CMake` options:
>
> ```
> DISABLE_PSI_COND
> DISABLE_PSI_DATA_LOCK
> DISABLE_PSI_ERROR
> DISABLE_PSI_FILE
> DISABLE_PSI_IDLE
> DISABLE_PSI_MEMORY
> DISABLE_PSI_METADATA
> DISABLE_PSI_MUTEX
> DISABLE_PSI_PS
> DISABLE_PSI_RWLOCK
> DISABLE_PSI_SOCKET
> DISABLE_PSI_SP
> DISABLE_PSI_STAGE
> DISABLE_PSI_STATEMENT
> DISABLE_PSI_STATEMENT_DIGEST
> DISABLE_PSI_TABLE
> DISABLE_PSI_THREAD
> DISABLE_PSI_TRANSACTION
> ```
>
> For example, to compile without mutex instrumentation, configure MySQL using `-DDISABLE_PSI_MUTEX=1`.

To exclude a storage engine from the build, use `-DWITH_engine_STORAGE_ENGINE=0`. Examples:

```
-DWITH_ARCHIVE_STORAGE_ENGINE=0
-DWITH_EXAMPLE_STORAGE_ENGINE=0
-DWITH_FEDERATED_STORAGE_ENGINE=0
```

It is also possible to exclude a storage engine from the build using `-DWITHOUT_engine_STORAGE_ENGINE=1` (but `-DWITH_engine_STORAGE_ENGINE=0` is preferred). Examples:

```
-DWITHOUT_ARCHIVE_STORAGE_ENGINE=1
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1
-DWITHOUT_FEDERATED_STORAGE_ENGINE=1
```

If neither `-DWITH_engine_STORAGE_ENGINE` nor `-DWITHOUT_engine_STORAGE_ENGINE` are specified for a given storage engine, the engine is built as a shared module, or excluded if it cannot be built as a shared module.

Feature Options

- `-DADD_GDB_INDEX=`*`bool`*

  This option determines whether to enable generation of a `.gdb_index` section in binaries, which makes loading them in a debugger faster. The option is disabled by default. `lld` linker is used, and is disabled by It has no effect if a linker other than `lld` or GNU `gold` is used.

  This option was added in MySQL 8.0.18.

- `-DCOMPILATION_COMMENT=`*`string`*

  A descriptive comment about the compilation environment. As of MySQL 8.0.14, `mysqld` uses `COMPILATION_COMMENT_SERVER`. Other programs continue to use `COMPILATION_COMMENT`.

- `-DCOMPRESS_DEBUG_SECTIONS=`*`bool`*

  Whether to compress the debug sections of binary executables (Linux only). Compressing executable debug sections saves space at the cost of extra CPU time during the build process.

  The default is `OFF`. If this option is not set explicitly but the `COMPRESS_DEBUG_SECTIONS` environment variable is set, the option takes its value from that variable.

  This option was added in MySQL 8.0.22.

- `-DCOMPILATION_COMMENT_SERVER=`*`string`*

  A descriptive comment about the compilation environment for use by `mysqld` (for example, to set the `version_comment` system variable). This option was added in MySQL 8.0.14. Prior to 8.0.14, the server uses `COMPILATION_COMMENT`.

- `-DDEFAULT_CHARSET=`*`charset_name`*

  The server character set. By default, MySQL uses the `utf8mb4` character set.

  `charset_name` may be one of `binary`, `armscii8`, `ascii`, `big5`, `cp1250`, `cp1251`, `cp1256`, `cp1257`, `cp850`, `cp852`, `cp866`, `cp932`, `dec8`, `eucjpms`, `euckr`, `gb2312`, `gbk`, `geostd8`, `greek`, `hebrew`, `hp8`, `keybcs2`, `koi8r`, `koi8u`, `latin1`, `latin2`, `latin5`, `latin7`, `macce`, `macroman`, `sjis`, `swe7`, `tis620`, `ucs2`, `ujis`, `utf8mb3`, `utf8mb4`, `utf16`, `utf16le`, `utf32`.

  This value can be set at server startup with the `--character-set-server` option.

- `-DDEFAULT_COLLATION=`*`collation_name`*

  The server collation. By default, MySQL uses `utf8mb4_0900_ai_ci`. Use the `SHOW COLLATION` statement to determine which collations are available for each character set.

  This value can be set at server startup with the `--collation_server` option.

- `-DDISABLE_PSI_COND=`*`bool`*

  Whether to exclude the Performance Schema condition instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_FILE=`*`bool`*

  Whether to exclude the Performance Schema file instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_IDLE=`*`bool`*

  Whether to exclude the Performance Schema idle instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_MEMORY=`*`bool`*

Whether to exclude the Performance Schema memory instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_METADATA=`*bool*

Whether to exclude the Performance Schema metadata instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_MUTEX=`*bool*

Whether to exclude the Performance Schema mutex instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_RWLOCK=`*bool*

Whether to exclude the Performance Schema rwlock instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_SOCKET=`*bool*

Whether to exclude the Performance Schema socket instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_SP=`*bool*

Whether to exclude the Performance Schema stored program instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STAGE=`*bool*

Whether to exclude the Performance Schema stage instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STATEMENT=`*bool*

Whether to exclude the Performance Schema statement instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STATEMENT_DIGEST=`*bool*

Whether to exclude the Performance Schema statement digest instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_TABLE=`*bool*

Whether to exclude the Performance Schema table instrumentation. The default is `OFF` (include).

- `-DDISABLE_SHARED=`*bool*

Whether to disable building build shared libraries and compile position-dependent code. The default is `OFF` (compile position-independent code).

This option is unused, and was removed in MySQL 8.0.18.

- `-DDISABLE_PSI_PS=`*bool*

Exclude the Performance Schema prepared statements instances instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_THREAD=`*bool*

Exclude the Performance Schema thread instrumentation. The default is `OFF` (include).

Only disable threads when building without any instrumentation, because other instrumentations have a dependency on threads.

- `-DDISABLE_PSI_TRANSACTION=`*bool*

Exclude the Performance Schema transaction instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_DATA_LOCK=`*bool*

  Exclude the performance schema data lock instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_ERROR=`*bool*

  Exclude the performance schema server error instrumentation. The default is `OFF` (include).

- `-DDOWNLOAD_BOOST=`*bool*

  Whether to download the Boost library. The default is `OFF`.

  See the `WITH_BOOST` option for additional discussion about using Boost.

- `-DDOWNLOAD_BOOST_TIMEOUT=`*seconds*

  The timeout in seconds for downloading the Boost library. The default is 600 seconds.

  See the `WITH_BOOST` option for additional discussion about using Boost.

- `-DENABLE_DOWNLOADS=`*bool*

  Whether to download optional files. For example, with this option enabled, `CMake` downloads the Google Test distribution that is used by the test suite to run unit tests, or Ant and JUnit, required for building the GCS Java wrapper.

  As of MySQL 8.0.26, MySQL source distributions bundle the Google Test source code used to run unit tests. Consequently, as of that version the `WITH_GMOCK` and `ENABLE_DOWNLOADS CMake` options are removed and are ignored if specified.

- `-DENABLE_EXPERIMENTAL_SYSVARS=`*bool*

  Whether to enable experimental `InnoDB` system variables. Experimental system variables are intended for those engaged in MySQL development, should only be used in a development or test environment, and may be removed without notice in a future MySQL release. For information about experimental system variables, refer to `/storage/innobase/handler/ha_innodb.cc` in the MySQL source tree. Experimental system variables can be identified by searching for "PLUGIN_VAR_EXPERIMENTAL".

- `-DENABLE_GCOV=`*bool*

  Whether to include `gcov` support (Linux only).

- `-DENABLE_GPROF=`*bool*

  Whether to enable `gprof` (optimized Linux builds only).

- `-DENABLED_LOCAL_INFILE=`*bool*

  This option controls the compiled-in default `LOCAL` capability for the MySQL client library. Clients that make no explicit arrangements therefore have `LOCAL` capability disabled or enabled according to the `ENABLED_LOCAL_INFILE` setting specified at MySQL build time.

  By default, the client library in MySQL binary distributions is compiled with `ENABLED_LOCAL_INFILE` disabled. If you compile MySQL from source, configure it with `ENABLED_LOCAL_INFILE` disabled or enabled based on whether clients that make no explicit arrangements should have `LOCAL` capability disabled or enabled, respectively.

  `ENABLED_LOCAL_INFILE` controls the default for client-side `LOCAL` capability. For the server, the `local_infile` system variable controls server-side `LOCAL` capability. To explicitly cause the server to refuse or permit `LOAD DATA LOCAL` statements (regardless of how client programs and libraries are configured at build time or runtime), start `mysqld` with `--local-infile` disabled or enabled,

respectively. `local_infile` can also be set at runtime. See Security Considerations for LOAD DATA LOCAL.

- `-DENABLED_PROFILING=`*bool*

  Whether to enable query profiling code (for the `SHOW PROFILE` and `SHOW PROFILES` statements).

- `-DFORCE_UNSUPPORTED_COMPILER=`*bool*

  By default, `CMake` checks for minimum versions of supported compilers; to disable this check, use `-DFORCE_UNSUPPORTED_COMPILER=ON`.

- `-DFPROFILE_GENERATE=`*bool*

  Whether to generate profile guided optimization (PGO) data. This option is available for experimenting with PGO with GCC. See `cmake/fprofile.cmake` in the MySQL source distribution for information about using `FPROFILE_GENERATE` and `FPROFILE_USE`. These options have been tested with GCC 8 and 9.

  This option was added in MySQL 8.0.19.

- `-DFPROFILE_USE=`*bool*

  Whether to use profile guided optimization (PGO) data. This option is available for experimenting with PGO with GCC. See the `cmake/fprofile.cmake` file in a MySQL source distribution for information about using `FPROFILE_GENERATE` and `FPROFILE_USE`. These options have been tested with GCC 8 and 9.

  Enabling `FPROFILE_USE` also enables `WITH_LTO`.

  This option was added in MySQL 8.0.19.

- `-DHAVE_PSI_MEMORY_INTERFACE=`*bool*

  Whether to enable the performance schema memory tracing module for memory allocation functions (`ut::aligned_`*name* library functions) used in dynamic storage of over-aligned types.

- `-DIGNORE_AIO_CHECK=`*bool*

  If the `-DBUILD_CONFIG=mysql_release` option is given on Linux, the `libaio` library must be linked in by default. If you do not have `libaio` or do not want to install it, you can suppress the check for it by specifying `-DIGNORE_AIO_CHECK=1`.

- `-DMAX_INDEXES=`*num*

  The maximum number of indexes per table. The default is 64. The maximum is 255. Values smaller than 64 are ignored and the default of 64 is used.

- `-DMYSQL_MAINTAINER_MODE=`*bool*

  Whether to enable a MySQL maintainer-specific development environment. If enabled, this option causes compiler warnings to become errors.

- `-DWITH_DEVELOPER_ENTITLEMENTS=`*bool*

  Whether to add the `get-task-allow` entitlement to all executables to generate a core dump in the event of an unexpected server halt.

  On macOS 11+, core dumps are limited to processes with the `com.apple.security.get-task-allow` entitlement, which this CMake option enables. The entitlement allows other processes to attach and read/modify the processes memory, and allows `--core-file` to function as expected.

  This option was added in MySQL 8.0.30.

- `-DMUTEX_TYPE=type`

  The mutex type used by `InnoDB`. Options include:

  - `event`: Use event mutexes. This is the default value and the original `InnoDB` mutex implementation.

  - `sys`: Use POSIX mutexes on UNIX systems. Use `CRITICAL_SECTION` objects on Windows, if available.

  - `futex`: Use Linux futexes instead of condition variables to schedule waiting threads.

- `-DMYSQLX_TCP_PORT=port_num`

  The port number on which X Plugin listens for TCP/IP connections. The default is 33060.

  This value can be set at server startup with the `mysqlx_port` system variable.

- `-DMYSQLX_UNIX_ADDR=file_name`

  The Unix socket file path on which the server listens for X Plugin socket connections. This must be an absolute path name. The default is `/tmp/mysqlx.sock`.

  This value can be set at server startup with the `mysqlx_port` system variable.

- `-DMYSQL_PROJECT_NAME=name`

  For Windows or macOS, the project name to incorporate into the project file name.

- `-DMYSQL_TCP_PORT=port_num`

  The port number on which the server listens for TCP/IP connections. The default is 3306.

  This value can be set at server startup with the `--port` option.

- `-DMYSQL_UNIX_ADDR=file_name`

  The Unix socket file path on which the server listens for socket connections. This must be an absolute path name. The default is `/tmp/mysql.sock`.

  This value can be set at server startup with the `--socket` option.

- `-DOPTIMIZER_TRACE=bool`

  Whether to support optimizer tracing. See Tracing the Optimizer.

- `-DREPRODUCIBLE_BUILD=bool`

  For builds on Linux systems, this option controls whether to take extra care to create a build result independent of build location and time.

  This option was added in MySQL 8.0.11. As of MySQL 8.0.12, it defaults to `ON` for `RelWithDebInfo` builds.

- `-DSHOW_SUPPRESSED_COMPILER_WARNINGS=bool`

  Show suppressed compiler warnings, and do so without failing with `-Werror`. Defaults to `OFF`.

  This option was added in MySQL 8.0.30.

- `-DUSE_LD_GOLD=bool`

  GNU `gold` linker support was removed in MySQL 8.0.31; this CMake option was also removed.

CMake causes the build process to link with the GNU `gold` linker if it is available and not explicitly disabled. To disable use of this linker, specify the `-DUSE_LD_GOLD=OFF` option.

- `-DUSE_LD_LLD=`*bool*

  CMake causes the build process to link using the LLVM `lld` linker for Clang if it is available and not explicitly disabled. To disable use of this linker, specify the `-DUSE_LD_LLD=OFF` option.

  This option was added in MySQL 8.0.16.

- `-DWIN_DEBUG_NO_INLINE=`*bool*

  Whether to disable function inlining on Windows. The default is `OFF` (inlining enabled).

- `-DWITH_ANT=`*path_name*

  Set the path to Ant, required when building GCS Java wrapper. Set `WITH_ANT` to the path of a directory where the Ant tarball or unpacked archive is saved. When `WITH_ANT` is not set, or is set with the special value `system`, the build process assumes a binary `ant` exists in `$PATH`.

- `-DWITH_ASAN=`*bool*

  Whether to enable the AddressSanitizer, for compilers that support it. The default is `OFF`.

- `-DWITH_ASAN_SCOPE=`*bool*

  Whether to enable the AddressSanitizer `-fsanitize-address-use-after-scope` Clang flag for use-after-scope detection. The default is off. To use this option, `-DWITH_ASAN` must also be enabled.

- `-DWITH_AUTHENTICATION_CLIENT_PLUGINS=`*bool*

  This option is enabled automatically if any corresponding server authentication plugins are built. Its value thus depends on other `CMake` options and it should not be set explicitly.

  This option was added in MySQL 8.0.26.

- `-DWITH_AUTHENTICATION_LDAP=`*bool*

  Whether to report an error if the LDAP authentication plugins cannot be built:

  - If this option is disabled (the default), the LDAP plugins are built if the required header files and libraries are found. If they are not, `CMake` displays a note about it.

  - If this option is enabled, a failure to find the required header file and libraries causes CMake to produce an error, preventing the server from being built.

  For information about LDAP authentication, see LDAP Pluggable Authentication.

- `-DWITH_AUTHENTICATION_PAM=`*bool*

  Whether to build the PAM authentication plugin, for source trees that include this plugin. (See PAM Pluggable Authentication.) If this option is specified and the plugin cannot be compiled, the build fails.

- `-DWITH_AWS_SDK=`*path_name*

  The location of the Amazon Web Services software development kit.

- `-DWITH_BOOST=`*path_name*

  The Boost library is required to build MySQL. These `CMake` options enable control over the library source location, and whether to download it automatically:

- `-DWITH_BOOST=`*`path_name`* specifies the Boost library directory location. It is also possible to specify the Boost location by setting the `BOOST_ROOT` or `WITH_BOOST` environment variable.

  `-DWITH_BOOST=system` is also permitted and indicates that the correct version of Boost is installed on the compilation host in the standard location. In this case, the installed version of Boost is used rather than any version included with a MySQL source distribution.

- `-DDOWNLOAD_BOOST=`*`bool`* specifies whether to download the Boost source if it is not present in the specified location. The default is `OFF`.

- `-DDOWNLOAD_BOOST_TIMEOUT=`*`seconds`* the timeout in seconds for downloading the Boost library. The default is 600 seconds.

For example, if you normally build MySQL placing the object output in the `bld` subdirectory of your MySQL source tree, you can build with Boost like this:

```
mkdir bld
cd bld
cmake .. -DDOWNLOAD_BOOST=ON -DWITH_BOOST=$HOME/my_boost
```

This causes Boost to be downloaded into the `my_boost` directory under your home directory. If the required Boost version is already there, no download is done. If the required Boost version changes, the newer version is downloaded.

If Boost is already installed locally and your compiler finds the Boost header files on its own, it may not be necessary to specify the preceding `CMake` options. However, if the version of Boost required by MySQL changes and the locally installed version has not been upgraded, you may have build problems. Using the `CMake` options should give you a successful build.

With the above settings that allow Boost download into a specified location, when the required Boost version changes, you need to remove the `bld` folder, recreate it, and perform the `cmake` step again. Otherwise, the new Boost version might not get downloaded, and compilation might fail.

- `-DWITH_CLIENT_PROTOCOL_TRACING=`*`bool`*

  Whether to build the client-side protocol tracing framework into the client library. By default, this option is enabled.

  For information about writing protocol trace client plugins, see Writing Protocol Trace Plugins.

  See also the `WITH_TEST_TRACE_PLUGIN` option.

- `-DWITH_CURL=`*`curl_type`*

  The location of the `curl` library. *`curl_type`* can be `system` (use the system `curl` library) or a path name to the `curl` library.

- `-DWITH_DEBUG=`*`bool`*

  Whether to include debugging support.

  Configuring MySQL with debugging support enables you to use the `--debug="d,parser_debug"` option when you start the server. This causes the Bison parser that is used to process SQL statements to dump a parser trace to the server's standard error output. Typically, this output is written to the error log.

  Sync debug checking for the `InnoDB` storage engine is defined under `UNIV_DEBUG` and is available when debugging support is compiled in using the `WITH_DEBUG` option. When debugging support

is compiled in, the `innodb_sync_debug` configuration option can be used to enable or disable `InnoDB` sync debug checking.

Enabling `WITH_DEBUG` also enables Debug Sync. This facility is used for testing and debugging. When compiled in, Debug Sync is disabled by default at runtime. To enable it, start `mysqld` with the `--debug-sync-timeout=N` option, where N is a timeout value greater than 0. (The default value is 0, which disables Debug Sync.) N becomes the default timeout for individual synchronization points.

Sync debug checking for the `InnoDB` storage engine is available when debugging support is compiled in using the `WITH_DEBUG` option.

For a description of the Debug Sync facility and how to use synchronization points, see MySQL Internals: Test Synchronization.

- `-DWITH_DEFAULT_FEATURE_SET=bool`

  Whether to use the flags from `cmake/build_configurations/feature_set.cmake`. This option was removed in MySQL 8.0.22.

- `-DWITH_EDITLINE=value`

  Which `libedit`/`editline` library to use. The permitted values are `bundled` (the default) and `system`.

- `-DWITH_FIDO=fido_type`

  The `authentication_fido` authentication plugin is implemented using a FIDO library (see FIDO Pluggable Authentication). The `WITH_FIDO` option indicates the source of FIDO support:

  - `bundled`: Use the FIDO library bundled with the distribution. This is the default.

    As of MySQL 8.0.30, MySQL includes `fido2` version 1.8.0. (Prior releases used `fido2` 1.5.0).

  - `system`: Use the system FIDO library.

  `WITH_FIDO` is disabled (set to `none`) if all authentication plugins are disabled.

  This option was added in MySQL 8.0.27.

- `-DWITH_GMOCK=path_name`

  The path to the googlemock distribution, for use with Google Test-based unit tests. The option value is the path to the distribution zip file. Alternatively, set the `WITH_GMOCK` environment variable to the path name. It is also possible to use `-DENABLE_DOWNLOADS=1`, so that CMake downloads the distribution from GitHub.

  If you build MySQL without the Google Test unit tests (by configuring without `WITH_GMOCK`), CMake displays a message indicating how to download it.

  As of MySQL 8.0.26, MySQL source distributions bundle the Google Test source code. Consequently, as of that version, the `WITH_GMOCK` and `ENABLE_DOWNLOADS` CMake options are removed and are ignored if specified.

- `-DWITH_ICU={icu_type|path_name}`

  MySQL uses International Components for Unicode (ICU) to support regular expression operations. The `WITH_ICU` option indicates the type of ICU support to include or the path name to the ICU installation to use.

  - `icu_type` can be one of the following values:

    - `bundled`: Use the ICU library bundled with the distribution. This is the default, and is the only supported option for Windows.

    - `system`: Use the system ICU library.

  - `path_name` is the path name to the ICU installation to use. This can be preferable to using the `icu_type` value of `system` because it can prevent CMake from detecting and using an older or incorrect ICU version installed on the system. (Another permitted way to do the same thing is to set `WITH_ICU` to `system` and set the `CMAKE_PREFIX_PATH` option to `path_name`.)

- `-DWITH_INNODB_EXTRA_DEBUG=bool`

  Whether to include extra InnoDB debugging support.

  Enabling `WITH_INNODB_EXTRA_DEBUG` turns on extra InnoDB debug checks. This option can only be enabled when `WITH_DEBUG` is enabled.

- `-DWITH_INNODB_MEMCACHED=bool`

  Whether to generate memcached shared libraries (`libmemcached.so` and `innodb_engine.so`).

- `-DWITH_JEMALLOC=bool`

  Whether to link with `-ljemalloc`. If enabled, built-in `malloc()`, `calloc()`, `realloc()`, and `free()` routines are disabled. The default is `OFF`.

  `WITH_JEMALLOC` and `WITH_TCMALLOC` are mutually exclusive.

  This option was added in MySQL 8.0.16.

- `-DWITH_KEYRING_TEST=bool`

  Whether to build the test program that accompanies the `keyring_file` plugin. The default is `OFF`. Test file source code is located in the `plugin/keyring/keyring-test` directory.

- `-DWITH_LIBEVENT=string`

  Which `libevent` library to use. Permitted values are `bundled` (default) and `system`. Prior to MySQL 8.0.21, if you specify `system`, the system `libevent` library is used if present, and an error occurs otherwise. In MySQL 8.0.21 and later, if `system` is specified and no system `libevent` library can be found, an error occurs regardless, and the bundled `libevent` is not used.

  The `libevent` library is required by `InnoDB` memcached, X Plugin, and MySQL Router.

- `-DWITH_LIBWRAP=bool`

  Whether to include `libwrap` (TCP wrappers) support.

- `-DWITH_LOCK_ORDER=bool`

  Whether to enable LOCK_ORDER tooling. By default, this option is disabled and server builds contain no tooling. If tooling is enabled, the LOCK_ORDER tool is available and can be used as described in The LOCK_ORDER Tool.

  > **Note**
  >
  > With the `WITH_LOCK_ORDER` option enabled, MySQL builds require the `flex` program.

  This option was added in MySQL 8.0.17.

- `-DWITH_LSAN=bool`

  Whether to run LeakSanitizer, without AddressSanitizer. The default is `OFF`.

  This option was added in MySQL 8.0.16.

- `-DWITH_LTO=bool`

  Whether to enable the link-time optimizer, if the compiler supports it. The default is `OFF` unless `FPROFILE_USE` is enabled.

  This option was added in MySQL 8.0.13.

- `-DWITH_LZ4=lz4_type`

  The `WITH_LZ4` option indicates the source of `zlib` support:

  - `bundled`: Use the `lz4` library bundled with the distribution. This is the default.

  - `system`: Use the system `lz4` library. If `WITH_LZ4` is set to this value, the `lz4_decompress` utility is not built. In this case, the system `lz4` command can be used instead.

- `-DWITH_LZMA=lzma_type`

  The type of LZMA library support to include. `lzma_type` can be one of the following values:

  - `bundled`: Use the LZMA library bundled with the distribution. This is the default.

  - `system`: Use the system LZMA library.

  This option was removed in MySQL 8.0.16.

- `-DWITH_MECAB={disabled|system|path_name}`

  Use this option to compile the MeCab parser. If you have installed MeCab to its default installation directory, set `-DWITH_MECAB=system`. The `system` option applies to MeCab installations performed from source or from binaries using a native package management utility. If you installed MeCab to a custom installation directory, specify the path to the MeCab installation, for example, `-DWITH_MECAB=/opt/mecab`. If the `system` option does not work, specifying the MeCab installation path should work in all cases.

  For related information, see MeCab Full-Text Parser Plugin.

- `-DWITH_MSAN=bool`

  Whether to enable MemorySanitizer, for compilers that support it. The default is off.

  For this option to have an effect if enabled, all libraries linked to MySQL must also have been compiled with the option enabled.

- `-DWITH_MSCRT_DEBUG=`*`bool`*

  Whether to enable Visual Studio CRT memory leak tracing. The default is `OFF`.

- `-DMSVC_CPPCHECK=`*`bool`*

  Whether to enable MSVC code analysis. The default is `OFF`.

- `-DWITH_MYSQLX=`*`bool`*

  Whether to build with support for X Plugin. The default is `ON`. See Using MySQL as a Document Store.

- `-DWITH_NUMA=`*`bool`*

  Explicitly set the NUMA memory allocation policy. `CMake` sets the default `WITH_NUMA` value based on whether the current platform has `NUMA` support. For platforms without NUMA support, `CMake` behaves as follows:

  - With no NUMA option (the normal case), `CMake` continues normally, producing only this warning: `NUMA library missing or required version not available`.

  - With `-DWITH_NUMA=ON`, `CMake` aborts with this error: `NUMA library missing or required version not available`.

- `-DWITH_PACKAGE_FLAGS=`*`bool`*

  For flags typically used for RPM and Debian packages, whether to add them to standalone builds on those platforms. The default is `ON` for nondebug builds.

  This option was added in MySQL 8.0.26.

- `-DWITH_PROTOBUF=`*`protobuf_type`*

  Which Protocol Buffers package to use. *`protobuf_type`* can be one of the following values:

  - `bundled`: Use the package bundled with the distribution. This is the default. Optionally use `INSTALL_PRIV_LIBDIR` to modify the dynamic Protobuf library directory.

  - `system`: Use the package installed on the system.

  Other values are ignored, with a fallback to `bundled`.

- `-DWITH_RAPID=`*`bool`*

  Whether to build the rapid development cycle plugins. When enabled, a `rapid` directory is created in the build tree containing these plugins. When disabled, no `rapid` directory is created in the build tree. The default is `ON`, unless the `rapid` directory is removed from the source tree, in which case the default becomes `OFF`.

- `-DWITH_RAPIDJSON=`*`rapidjson_type`*

  The type of RapidJSON library support to include. *`rapidjson_type`* can be one of the following values:

  - `bundled`: Use the RapidJSON library bundled with the distribution. This is the default.

  - `system`: Use the system RapidJSON library. Version 1.1.0 or later is required.

  This option was added in MySQL 8.0.13.

- `-DWITH_RE2=re2_type`

  The type of RE2 library support to include. `re2_type` can be one of the following values:

  - `bundled`: Use the RE2 library bundled with the distribution. This is the default.

  - `system`: Use the system RE2 library.

  As of MySQL 8.0.18, MySQL no longer uses the RE2 library, and this option has been removed.

- `-DWITH_ROUTER=bool`

  Whether to build MySQL Router. The default is `ON`.

  This option was added in MySQL 8.0.16.

- `-DWITH_SASL=value`

  Internal use only. This option was added in 8.0.20. Not supported on Windows.

- `-DWITH_SSL={ssl_type|path_name}`

  For support of encrypted connections, entropy for random number generation, and other encryption-related operations, MySQL must be built using an SSL library. This option specifies which SSL library to use.

  - `ssl_type` can be one of the following values:

    - `system`: Use the system OpenSSL library. This is the default.

      On macOS and Windows, using `system` configures MySQL to build as if CMake was invoked with `path_name` points to a manually installed OpenSSL library. This is because they do not have system SSL libraries. On macOS, *brew install openssl* installs to `/usr/local/opt/openssl` so that `system` can find it. On Windows, it checks `%ProgramFiles%/OpenSSL`, `%ProgramFiles%/OpenSSL-Win32`, `%ProgramFiles%/OpenSSL-Win64`, `C:/OpenSSL`, `C:/OpenSSL-Win32`, and `C:/OpenSSL-Win64`.

    - `yes`: This is a synonym for `system`.

    - `opensslversion`: (*MySQL 8.0.30 and later:*) Use an alternate OpenSSL system package such as `openssl11` on EL7, or `openssl3` on EL8.

      Authentication plugins, such as LDAP and Kerberos, are disabled as they do not support these alternative versions of OpenSSL.

  - `path_name` is the path name to the OpenSSL installation to use. This can be preferable to using the `ssl_type` value of `system` because it can prevent CMake from detecting and using an older or incorrect OpenSSL version installed on the system. (Another permitted way to do the same thing is to set `WITH_SSL` to `system` and set the `CMAKE_PREFIX_PATH` option to `path_name`.)

  For additional information about configuring the SSL library, see Configuring SSL Library Support.

- `-DWITH_SYSTEMD=bool`

  Whether to enable installation of `systemd` support files. By default, this option is disabled. When enabled, `systemd` support files are installed, and scripts such as `mysqld_safe` and the System V initialization script are not installed. On platforms where `systemd` is not available, enabling `WITH_SYSTEMD` results in an error from `CMake`.

  For more information about using `systemd`, see Managing MySQL Server with systemd. That section also includes information about specifying options otherwise specified in `[mysqld_safe]`

option groups. Because `mysqld_safe` is not installed when `systemd` is used, such options must be specified another way.

- `-DWITH_SYSTEM_LIBS=`*bool*

  This option serves as an "umbrella" option to set the `system` value of any of the following `CMake` options that are not set explicitly: `WITH_CURL`, `WITH_EDITLINE`, `WITH_FIDO`, `WITH_ICU`, `WITH_LIBEVENT`, `WITH_LZ4`, `WITH_LZMA`, `WITH_PROTOBUF`, `WITH_RE2`, `WITH_SSL`, `WITH_ZSTD`.

  `WITH_ZLIB` was included here priot MySQL 8.0.30.

- `-DWITH_SYSTEMD_DEBUG=`*bool*

  Whether to produce additional `systemd` debugging information, for platforms on which `systemd` is used to run MySQL. The default is `OFF`.

  This option was added in MySQL 8.0.22.

- `-DWITH_TCMALLOC=`*bool*

  Whether to link with `-ltcmalloc`. If enabled, built-in `malloc()`, `calloc()`, `realloc()`, and `free()` routines are disabled. The default is `OFF`.

  Beginning with MySQL 8.0.38, a `tcmalloc` library is included in the source; you can cause the build to use the bundled version by setting this option to `BUNDLED`. `BUNDLED` is supported on Linux systems only.

  `WITH_TCMALLOC` and `WITH_JEMALLOC` are mutually exclusive.

  This option was added in MySQL 8.0.22.

- `-DWITH_TEST_TRACE_PLUGIN=`*bool*

  Whether to build the test protocol trace client plugin (see Using the Test Protocol Trace Plugin). By default, this option is disabled. Enabling this option has no effect unless the `WITH_CLIENT_PROTOCOL_TRACING` option is enabled. If MySQL is configured with both options enabled, the `libmysqlclient` client library is built with the test protocol trace plugin built in, and all the standard MySQL clients load the plugin. However, even when the test plugin is enabled, it has no effect by default. Control over the plugin is afforded using environment variables; see Using the Test Protocol Trace Plugin.

  > **Note**
  >
  > Do *not* enable the `WITH_TEST_TRACE_PLUGIN` option if you want to use your own protocol trace plugins because only one such plugin can be loaded at a time and an error occurs for attempts to load a second one. If you have already built MySQL with the test protocol trace plugin enabled to see how it works, you must rebuild MySQL without it before you can use your own plugins.

  For information about writing trace plugins, see Writing Protocol Trace Plugins.

- `-DWITH_TSAN=`*bool*

  Whether to enable the ThreadSanitizer, for compilers that support it. The default is off.

- `-DWITH_UBSAN=`*bool*

  Whether to enable the Undefined Behavior Sanitizer, for compilers that support it. The default is off.

- `-DWITH_UNIT_TESTS={ON|OFF}`

  If enabled, compile MySQL with unit tests. The default is `ON` unless the server is not being compiled.

- `-DWITH_UNIXODBC=1`

  Enables unixODBC support, for Connector/ODBC.

- `-DWITH_VALGRIND=bool`

  Whether to compile in the Valgrind header files, which exposes the Valgrind API to MySQL code. The default is `OFF`.

  To generate a Valgrind-aware debug build, `-DWITH_VALGRIND=1` normally is combined with `-DWITH_DEBUG=1`. See Building Debug Configurations.

- `-DWITH_WIN_JEMALLOC=string`

  On Windows, pass in a path to a directory containing `jemalloc.dll` to enable jemalloc functionality. The build system copies `jemalloc.dll` to the same directory as `mysqld.exe` and/or `mysqld-debug.exe` and utilizes it for memory management operations. Standard memory functions are used if `jemalloc.dll` is not found or does not export the required functions. An INFORMATION level log message records whether or not jemalloc is found and used.

  This option is enabled for official MySQL binaries for Windows.

  This option was added in MySQL 8.0.29.

- `-DWITH_ZLIB=zlib_type`

  Some features require that the server be built with compression library support, such as the `COMPRESS()` and `UNCOMPRESS()` functions, and compression of the client/server protocol. The `WITH_ZLIB` option indicates the source of `zlib` support:

  In MYSQL 8.0.32 and later, the minimum supported version of `zlib` is 1.2.13.

  - `bundled`: Use the `zlib` library bundled with the distribution. This is the default.

  - `system`: Use the system `zlib` library. If `WITH_ZLIB` is set to this value, the `zlib_decompress` utility is not built. In this case, the system `openssl zlib` command can be used instead.

- `-DWITH_ZSTD=zstd_type`

  Connection compression using the `zstd` algorithm (see Connection Compression Control) requires that the server be built with `zstd` library support. The `WITH_ZSTD` option indicates the source of `zstd` support:

  - `bundled`: Use the `zstd` library bundled with the distribution. This is the default.

  - `system`: Use the system `zstd` library.

  This option was added in MySQL 8.0.18.

- `-DWITHOUT_SERVER=bool`

  Whether to build without MySQL Server. The default is OFF, which does build the server.

  This is considered an experimental option; it is preferred to build with the server.

  This option also prevents building of the `NDB` storage engine or any `NDB` binaries including management and data node programs.

Compiler Flags

- `-DCMAKE_C_FLAGS="flags"`

  Flags for the C compiler.

- `-DCMAKE_CXX_FLAGS="flags"`

  Flags for the C++ compiler.

- `-DWITH_DEFAULT_COMPILER_OPTIONS=bool`

  Whether to use the flags from `cmake/build_configurations/compiler_options.cmake`.

  > **Note**
  >
  > All optimization flags are carefully chosen and tested by the MySQL build team. Overriding them can lead to unexpected results and is done at your own risk.

- `-DOPTIMIZE_SANITIZER_BUILDS=bool`

  Whether to add `-O1 -fno-inline` to sanitizer builds. The default is `ON`.

To specify your own C and C++ compiler flags, for flags that do not affect optimization, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` CMake options.

When providing your own compiler flags, you might want to specify `CMAKE_BUILD_TYPE` as well.

For example, to create a 32-bit release build on a 64-bit Linux machine, do this:

```
$> mkdir build
$> cd build
$> cmake .. -DCMAKE_C_FLAGS=-m32 \
  -DCMAKE_CXX_FLAGS=-m32 \
  -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

If you set flags that affect optimization (`-Onumber`), you must set the `CMAKE_C_FLAGS_build_type` and/or `CMAKE_CXX_FLAGS_build_type` options, where `build_type` corresponds to the `CMAKE_BUILD_TYPE` value. To specify a different optimization for the default build type (`RelWithDebInfo`) set the `CMAKE_C_FLAGS_RELWITHDEBINFO` and `CMAKE_CXX_FLAGS_RELWITHDEBINFO` options. For example, to compile on Linux with `-O3` and with debug symbols, do this:

```
$> cmake .. -DCMAKE_C_FLAGS_RELWITHDEBINFO="-O3 -g" \
  -DCMAKE_CXX_FLAGS_RELWITHDEBINFO="-O3 -g"
```

CMake Options for Compiling NDB Cluster

To compile with support for NDB Cluster, you can use `-DWITH_NDB`, which causes the build to include the NDB storage engine and all NDB programs. This option is enabled by default. To prevent building of the NDB storage engine plugin, use `-DWITH_NDBCLUSTER_STORAGE_ENGINE=OFF`. Other aspects of the build can be controlled using the other options listed in this section.

The following options apply when building the MySQL sources with NDB Cluster support.

- `-DMEMCACHED_HOME=dir_name`

  `NDB` support for memcached was removed in NDB 8.0.23; thus, this option is no longer supported for building `NDB` in this or later versions.

- `-DNDB_UTILS_LINK_DYNAMIC={ON|OFF}`

Controls whether NDB utilities such as `ndb_drop_table` are linked with `ndbclient` statically (`OFF`) or dynamically (`ON`); `OFF` (static linking) is the default. Normally static linking is used when building these to avoid problems with `LD_LIBRARY_PATH`, or when multiple versions of `ndbclient` are installed. This option is intended for creating Docker images and possibly other cases in which the target environment is subject to precise control and it is desirable to reduce image size.

Added in NDB 8.0.22.

- `-DWITH_BUNDLED_LIBEVENT={ON|OFF}`

  `NDB` support for memcached was removed in NDB 8.0.23; thus, this option is no longer supported for building `NDB` in this or later versions.

- `-DWITH_BUNDLED_MEMCACHED={ON|OFF}`

  `NDB` support for memcached was removed in NDB 8.0.23; thus, this option is no longer supported for building `NDB` in this or later versions.

- `-DWITH_CLASSPATH=path`

  Sets the classpath for building MySQL NDB Cluster Connector for Java. The default is empty. This option is ignored if `-DWITH_NDB_JAVA=OFF` is used.

- `-DWITH_ERROR_INSERT={ON|OFF}`

  Enables error injection in the `NDB` kernel. For testing only; not intended for use in building production binaries. The default is `OFF`.

- `-DWITH_NDB={ON|OFF}`

  Build MySQL NDB Cluster; build the NDB plugin and all NDB Cluster programs.

  Added in NDB 8.0.31.

- `-DWITH_NDBAPI_EXAMPLES={ON|OFF}`

  Build NDB API example programs in `storage/ndb/ndbapi-examples/`. See NDB API Examples, for information about these.

- `-DWITH_NDBCLUSTER_STORAGE_ENGINE={ON|OFF}`

  NDB 8.0.30 and earlier: For internal use only; may not always work as expected. To build with `NDB` support, use `WITH_NDBCLUSTER` instead.

  NDB 8.0.31 and later: Controls (only) whether the `NDBCLUSTER` storage engine is included in the build; `WITH_NDB` enables this option automatically, so it is recommended that you use `WITH_NDB` instead.

- `-DWITH_NDBCLUSTER={ON|OFF}` (DEPRECATED)

  Build and link in support for the `NDB` storage engine in `mysqld`.

  This option is deprecated as of NDB 8.0.31, and subject to eventual removal; use `WITH_NDB` instead.

- `-DWITH_NDBMTD={ON|OFF}`

  Build the multithreaded data node executable `ndbmtd`. The default is `ON`.

- `-DWITH_NDB_DEBUG={ON|OFF}`

  Enable building the debug versions of the NDB Cluster binaries. This is `OFF` by default.

- `-DWITH_NDB_JAVA={ON|OFF}`

  Enable building NDB Cluster with Java support, including support for ClusterJ (see MySQL NDB Cluster Connector for Java).

  This option is `ON` by default. If you do not wish to compile NDB Cluster with Java support, you must disable it explicitly by specifying `-DWITH_NDB_JAVA=OFF` when running `CMake`. Otherwise, if Java cannot be found, configuration of the build fails.

- `-DWITH_NDB_PORT=port`

  Causes the NDB Cluster management server (`ndb_mgmd`) that is built to use this `port` by default. If this option is unset, the resulting management server tries to use port 1186 by default.

- `-DWITH_NDB_TEST={ON|OFF}`

  If enabled, include a set of NDB API test programs. The default is `OFF`.

- `-DWITH_PLUGIN_NDBCLUSTER={ON|OFF}`

  For internal use only; may not always work as expected. This option was removed in NDB 8.0.31; use `WITH_NDB` instead to build MySQL NDB Cluster. (*NDB 8.0.30 and earlier*: Use `WITH_NDBCLUSTER`.)

Chapter 5 Dealing with Problems Compiling MySQL

The solution to many problems involves reconfiguring. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.

- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run the following commands before re-running `CMake`:

On Unix:

```
$> make clean
$> rm CMakeCache.txt
```

On Windows:

```
$> devenv MySQL.sln /clean
$> del CMakeCache.txt
```

If you build outside of the source tree, remove and recreate your build directory before re-running `CMake`. For instructions on building outside of the source tree, see How to Build MySQL Server with CMake.

On some systems, warnings may occur due to differences in system include files. The following list describes other problems that have been found to occur most often when compiling MySQL:

- To define which C and C++ compilers to use, you can define the `CC` and `CXX` environment variables. For example:

  ```
  $> CC=gcc
  $> CXX=g++
  $> export CC CXX
  ```

  While this can be done on the command line, as just shown, you may prefer to define these values in a build script, in which case the `export` command is not needed.

  To specify your own C and C++ compiler flags, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` CMake options. See Compiler Flags.

  To see what flags you might need to specify, invoke `mysql_config` with the `--cflags` and `--cxxflags` options.

- To see what commands are executed during the compile stage, after using `CMake` to configure MySQL, run `make VERBOSE=1` rather than just `make`.

- If compilation fails, check whether the `MYSQL_MAINTAINER_MODE` option is enabled. This mode causes compiler warnings to become errors, so disabling it may enable compilation to proceed.

- If your compile fails with errors such as any of the following, you must upgrade your version of `make` to GNU `make`:

  ```
  make: Fatal error in reader: Makefile, line 18:
  Badly formed macro assignment
  ```

  Or:

```
make: file `Makefile' line 18: Must be a separator (:
```

Or:

```
pthread.h: No such file or directory
```

Solaris and FreeBSD are known to have troublesome `make` programs.

GNU `make` 3.75 is known to work.

- The `sql_yacc.cc` file is generated from `sql_yacc.yy`. Normally, the build process does not need to create `sql_yacc.cc` because MySQL comes with a pregenerated copy. However, if you do need to re-create it, you might encounter this error:

```
"sql_yacc.yy", line xxx fatal: default action causes potential...
```

This is a sign that your version of `yacc` is deficient. You probably need to install a recent version of `bison` (the GNU version of `yacc`) and use that instead.

Versions of `bison` older than 1.75 may report this error:

```
sql_yacc.yy:#####: fatal error: maximum table size (32767) exceeded
```

The maximum table size is not actually exceeded; the error is caused by bugs in older versions of `bison`.

For information about acquiring or updating tools, see the system requirements in Chapter 1, *Installing MySQL from Source*.