

### **Abstract**

This is the MySQL Installation Guide 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	
1 Installing MySQL	
2 General Installation Guidance	3
2.1 Supported Platforms	
2.2 Which MySQL Version and Distribution to Install	3
2.3 How to Get MySQL	4
2.4 Verifying Package Integrity Using MD5 Checksums or GnuPG	4
2.4.1 Verifying the MD5 Checksum	
2.4.2 Signature Checking Using GnuPG	
2.4.3 Signature Checking Using Gpg4win for Windows	7
2.4.4 Signature Checking Using RPM	
2.4.5 GPG Public Build Key for Archived Packages	
2.5 Installation Layouts	
2.6 Compiler-Specific Build Characteristics	
3 Installing MySQL on Unix/Linux Using Generic Binaries	
4 Installing MySQL from Source	
4.1 Source Installation Methods	
4.2 Source Installation Prerequisites	
4.3 MySQL Layout for Source Installation	
4.4 Installing MySQL Using a Standard Source Distribution	
4.5 Installing MySQL Using a Development Source Tree	
4.6 Configuring SSL Library Support	
4.7 MySQL Source-Configuration Options	
4.8 Dealing with Problems Compiling MySQL	
4.9 MySQL Configuration and Third-Party Tools	
4.10 Generating MySQL Doxygen Documentation Content	
5 Installing MySQL on Microsoft Windows	
5.1 MySQL Installation Layout on Microsoft Windows	
5.2 Choosing an Installation Package	
5.3 MySQL Installer for Windows	
5.3.1 MySQL Installer Initial Setup	
5.3.2 Setting Alternative Server Paths with MySQL Installer	
5.3.3 Installation Workflows with MySQL Installer	
5.3.4 MySQL Installer Product Catalog and Dashboard	
5.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive	
5.4.1 Extracting the Install Archive	
5.4.2 Creating an Option File	
5.4.3 Selecting a MySQL Server Type	
5.4.4 Initializing the Data Directory	
5.4.5 Starting the Server for the First Time	
5.4.6 Starting MySQL from the Windows Command Line	
5.4.7 Customizing the PATH for MySQL Tools	
5.4.8 Starting MySQL as a Windows Service	
5.4.9 Testing The MySQL Installation	
5.5 Troubleshooting a Microsoft Windows MySQL Server Installation	
5.6 Windows Postinstallation Procedures	
5.7 Windows Platform Restrictions	
6 Installing MySQL on macOS	
6.1 General Notes on Installing MySQL on macOS	
6.2 Installing MySQL on macOS Using Native Packages	126

6.3 Installing and Using the MySQL Launch Daemon	132
6.4 Installing and Using the MySQL Preference Pane	135
7 Installing MySQL on Linux	141
7.1 Installing MySQL on Linux Using the MySQL Yum Repository	142
7.2 Installing MySQL on Linux Using the MySQL APT Repository	
7.3 Installing MySQL on Linux Using the MySQL SLES Repository	
7.4 Installing MySQL on Linux Using RPM Packages from Oracle	
7.5 Installing MySQL on Linux Using Debian Packages from Oracle	
7.6 Deploying MySQL on Linux with Docker Containers	
7.6.1 Basic Steps for MySQL Server Deployment with Docker	
7.6.2 More Topics on Deploying MySQL Server with Docker	
7.6.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker	
7.7 Installing MySQL on Linux from the Native Software Repositories	
7.8 Installing MySQL on Linux with Juju	
7.9 Managing MySQL Server with systemd	
8 Installing MySQL on Solaris	
8.1 Installing MySQL on Solaris Using a Solaris PKG	
9 Postinstallation Setup and Testing	
9.1 Initializing the Data Directory	
9.2 Starting the Server	
9.2.1 Troubleshooting Problems Starting the MySQL Server	
9.3 Testing the Server	
9.4 Securing the Initial MySQL Account	
9.5 Starting and Stopping MySQL Automatically	
10 Upgrading MySQL	
10.1 Before You Begin	
10.2 Upgrade Paths	
10.3 Upgrade Best Practices	
10.4 What the MySQL Upgrade Process Upgrades	
10.5 Changes in MySQL 8.0	
10.6 Preparing Your Installation for Upgrade	
10.7 Upgrading MySQL Binary or Package-based Installations on Unix/Linux	
10.8 Upgrading MySQL with the MySQL Yum Repository	
10.9 Upgrading MySQL with the MySQL APT Repository	
10.10 Upgrading MySQL with the MySQL SLES Repository	
10.11 Upgrading MySQL on Windows	
10.12 Upgrading a Docker Installation of MySQL	
10.13 Upgrade Troubleshooting	
10.14 Rebuilding or Repairing Tables or Indexes	
10.15 Copying MySQL Databases to Another Machine	
11 Downgrading MySQL	
12 Environment Variables	
13 Perl Installation Notes	
13.1 Installing Perl on Unix	
13.2 Installing ActiveState Perl on Windows	
13.3 Problems Using the Perl DBI/DBD Interface	
10.0   100101110 U31119 tile   611 UUU/UUU IIIteliate	Z+0

# Preface and Legal Notices

This is the MySQL Installation Guide 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**

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

#### Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

#### **Use of This Documentation**

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

### **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

This chapter describes how to obtain and install MySQL. A summary of the procedure follows and later sections provide the details. If you plan to upgrade an existing version of MySQL to a newer version rather than install MySQL for the first time, see Chapter 10, *Upgrading MySQL*, for information about upgrade procedures and about issues that you should consider before upgrading.

If you are interested in migrating to MySQL from another database system, see MySQL 8.0 FAQ: Migration, which contains answers to some common questions concerning migration issues.

Installation of MySQL generally follows the steps outlined here:

#### 1. Determine whether MySQL runs and is supported on your platform.

Please note that not all platforms are equally suitable for running MySQL, and that not all platforms on which MySQL is known to run are officially supported by Oracle Corporation. For information about those platforms that are officially supported, see <a href="https://www.mysql.com/support/supportedplatforms/database.html">https://www.mysql.com/support/supportedplatforms/database.html</a> on the MySQL website.

#### 2. Choose which distribution to install.

Several versions of MySQL are available, and most are available in several distribution formats. You can choose from pre-packaged distributions containing binary (precompiled) programs or source code. When in doubt, use a binary distribution. Oracle also provides access to the MySQL source code for those who want to see recent developments and test new code. To determine which version and type of distribution you should use, see Section 2.2, "Which MySQL Version and Distribution to Install".

#### 3. Choose which track to install.

MySQL offers a bugfix track (such as MySQL 8.4), and an innovation track (today it's MySQL 9.3) and each track addresses different use cases. Both tracks are considered production-ready and include bug fixes, while innovation releases also include new features and potential for modified behavior.

A bugfix track upgrade includes point releases, such as MySQL 8.4.x upgrading to 8.4.y, while innovation track releases typically only have minor releases, such as MySQL 9.3.0 upgrading to 9.4.0. However, an innovation track does have the occasional point release.

#### 4. Download the distribution that you want to install.

For instructions, see Section 2.3, "How to Get MySQL". To verify the integrity of the distribution, use the instructions in Section 2.4, "Verifying Package Integrity Using MD5 Checksums or GnuPG".

#### 5. Install the distribution.

To install MySQL from a binary distribution, use the instructions in Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*. Alternatively, use the Secure Deployment Guide, which provides procedures for deploying a generic binary distribution of MySQL Enterprise Edition Server with features for managing the security of your MySQL installation.

To install MySQL from a source distribution or from the current development source tree, use the instructions in Chapter 4, *Installing MySQL from Source*.

#### 6. Perform any necessary postinstallation setup.

After installing MySQL, see Chapter 9, *Postinstallation Setup and Testing* for information about making sure the MySQL server is working properly. Also refer to the information provided in Section 9.4,

"Securing the Initial MySQL Account". This section describes how to secure the initial MySQL root user account, which has no password until you assign one. The section applies whether you install MySQL using a binary or source distribution.

7. If you want to run the MySQL benchmark scripts, Perl support for MySQL must be available. See Chapter 13, *Perl Installation Notes*.

Instructions for installing MySQL on different platforms and environments is available on a platform by platform basis:

#### • Unix, Linux

For instructions on installing MySQL on most Linux and Unix platforms using a generic binary (for example, a .tar.gz package), see Chapter 3, Installing MySQL on Unix/Linux Using Generic Binaries.

For information on building MySQL entirely from the source code distributions or the source code repositories, see Chapter 4, *Installing MySQL from Source* 

For specific platform help on installation, configuration, and building from source see the corresponding platform section:

- Linux, including notes on distribution specific methods, see Chapter 7, Installing MySQL on Linux.
- IBM AIX, see Chapter 8, Installing MySQL on Solaris.

#### Microsoft Windows

For instructions on installing MySQL on Microsoft Windows, using either the MySQL Installer or Zipped binary, see Chapter 5, *Installing MySQL on Microsoft Windows*.

For details and instructions on building MySQL from source code using Microsoft Visual Studio, see Chapter 4, *Installing MySQL from Source*.

#### macOS

For installation on macOS, including using both the binary package and native PKG formats, see Chapter 6, *Installing MySQL on macOS*.

For information on making use of an macOS Launch Daemon to automatically start and stop MySQL, see Section 6.3, "Installing and Using the MySQL Launch Daemon".

For information on the MySQL Preference Pane, see Section 6.4, "Installing and Using the MySQL Preference Pane".

## Chapter 2 General Installation Guidance

### Table of Contents

2.1	Supported Platforms	3
2.2	Which MySQL Version and Distribution to Install	. 3
2.3	How to Get MySQL	4
2.4	Verifying Package Integrity Using MD5 Checksums or GnuPG	. 4
	2.4.1 Verifying the MD5 Checksum	5
	2.4.2 Signature Checking Using GnuPG	5
	2.4.3 Signature Checking Using Gpg4win for Windows	. 7
	2.4.4 Signature Checking Using RPM	11
	2.4.5 GPG Public Build Key for Archived Packages	12
2.5	Installation Layouts	
	Compiler-Specific Build Characteristics	

The immediately following sections contain the information necessary to choose, download, and verify your distribution. The instructions in later sections of the chapter describe how to install the distribution that you choose. For binary distributions, see the instructions at Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries* or the corresponding section for your platform if available. To build MySQL from source, use the instructions in Chapter 4, *Installing MySQL from Source*.

### 2.1 Supported Platforms

MySQL platform support evolves over time; please refer to https://www.mysql.com/support/supportedplatforms/database.html for the latest updates.

### 2.2 Which MySQL Version and Distribution to Install

When preparing to install MySQL, decide which version and distribution format (binary or source) to use.

First, decide whether to install from a bugfix series like MySQL 8.4, or use an innovation release like MySQL 9.3. Both tracks include bug fixes while an innovation release includes the newest features. Both bugfix and innovation releases are meant for production use.

The naming scheme in MySQL 8.0 uses release names that consist of three numbers and an optional suffix (for example, **mysql-8.0.34**). The numbers within the release name are interpreted as follows:

- The first number (8) is the major version number.
- The second number (0) is the minor version number. Taken together, the major and minor numbers constitute the release series number. The series number describes the stable feature set.
- The third number (34) is the version number within the release series. This is incremented for each new bugfix release; for an innovation release, it will likely always be 0. For a bugfix series such as MySQL 8.0, the most recent version within the series is the best choice.

After choosing which MySQL version to install, decide which distribution format to install for your operating system. For most use cases, a binary distribution is the right choice. Binary distributions are available in native format for many platforms, such as RPM packages for Linux or DMG packages for macOS. Distributions are also available in more generic formats such as Zip archives or compressed tar files. On Windows, you can use the MySQL Installer to install a binary distribution.

Under some circumstances, it may be preferable to install MySQL from a source distribution:

- You want to install MySQL at some explicit location. The standard binary distributions are ready to run at any installation location, but you might require even more flexibility to place MySQL components where you want.
- You want to configure mysqld with features that might not be included in the standard binary distributions. Here is a list of the most common extra options used to ensure feature availability:
  - -DWITH\_LIBWRAP=1 for TCP wrappers support.
  - -DWITH\_ZLIB={system|bundled} for features that depend on compression
  - -DWITH\_DEBUG=1 for debugging support

For additional information, see Section 4.7, "MySQL Source-Configuration Options".

- You want to configure mysqld without some features that are included in the standard binary distributions.
- You want to read or modify the C and C++ code that makes up MySQL. For this purpose, obtain a source distribution.
- Source distributions contain more tests and examples than binary distributions.

### 2.3 How to Get MySQL

Check our downloads page at https://dev.mysql.com/downloads/ for information about the current version of MySQL and for downloading instructions.

For RPM-based Linux platforms that use Yum as their package management system, MySQL can be installed using the MySQL Yum Repository. See Section 7.1, "Installing MySQL on Linux Using the MySQL Yum Repository" for details.

For Debian-based Linux platforms, MySQL can be installed using the MySQL APT Repository. See Section 7.2, "Installing MySQL on Linux Using the MySQL APT Repository" for details.

For SUSE Linux Enterprise Server (SLES) platforms, MySQL can be installed using the MySQL SLES Repository. See Section 7.3, "Installing MySQL on Linux Using the MySQL SLES Repository" for details.

To obtain the latest development source, see Section 4.5, "Installing MySQL Using a Development Source Tree".

### 2.4 Verifying Package Integrity Using MD5 Checksums or GnuPG

After downloading the MySQL package that suits your needs and before attempting to install it, make sure that it is intact and has not been tampered with. There are three means of integrity checking:

- MD5 checksums
- Cryptographic signatures using GnuPG, the GNU Privacy Guard
- For RPM packages, the built-in RPM integrity verification mechanism

The following sections describe how to use these methods.

If you notice that the MD5 checksum or GPG signatures do not match, first try to download the respective package one more time, perhaps from another mirror site.

### 2.4.1 Verifying the MD5 Checksum

After you have downloaded a MySQL package, you should make sure that its MD5 checksum matches the one provided on the MySQL download pages. Each package has an individual checksum that you can verify against the package that you downloaded. The correct MD5 checksum is listed on the downloads page for each MySQL product; you should compare it against the MD5 checksum of the file (product) that you download.

Each operating system and setup offers its own version of tools for checking the MD5 checksum. Typically the command is named md5sum, or it may be named md5, and some operating systems do not ship it at all. On Linux, it is part of the **GNU Text Utilities** package, which is available for a wide range of platforms. You can also download the source code from http://www.gnu.org/software/textutils/. If you have OpenSSL installed, you can use the command openssl md5 package\_name instead. A Windows implementation of the md5 command line utility is available from http://www.fourmilab.ch/md5/. winMd5Sum is a graphical MD5 checking tool that can be obtained from http://www.nullriver.com/index/products/winmd5sum. Our Microsoft Windows examples assume the name md5.exe.

Linux and Microsoft Windows examples:

```
$> md5sum mysql-standard-8.0.42-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945 mysql-standard-8.0.42-linux-i686.tar.gz

$> md5.exe mysql-installer-community-8.0.42.msi
aaab65abbec64d5e907dcd41b8699945 mysql-installer-community-8.0.42.msi
```

You should verify that the resulting checksum (the string of hexadecimal digits) matches the one displayed on the download page immediately below the respective package.

#### Note

Make sure to verify the checksum of the *archive file* (for example, the .zip, .tar.gz, or .msi file) and not of the files that are contained inside of the archive. In other words, verify the file before extracting its contents.

### 2.4.2 Signature Checking Using GnuPG

Another method of verifying the integrity and authenticity of a package is to use cryptographic signatures. This is more reliable than using MD5 checksums, but requires more work.

We sign MySQL downloadable packages with <code>GnuPG</code> (GNU Privacy Guard). <code>GnuPG</code> is an Open Source alternative to the well-known Pretty Good Privacy (<code>PGP</code>) by Phil Zimmermann. Most Linux distributions ship with <code>GnuPG</code> installed by default. Otherwise, see <a href="http://www.gnupg.org/">http://www.gnupg.org/</a> for more information about <code>GnuPG</code> and how to obtain and install it.

To verify the signature for a specific package, you first need to obtain a copy of our public GPG build key, which you can download from http://pgp.mit.edu/. The key that you want to obtain is named mysql-build@oss.oracle.com. The keyID for MySQL 8.0.36 packages and higher, and MySQL 8.3.0 and higher, is A8D3785C. After obtaining this key, you should compare it with the key following value before using it verify MySQL packages. Alternatively, you can copy and paste the key directly from the text below.

#### Note

The public GPG build key for earlier MySQL release packages (keyID 5072E1F5 or 3A79BD29), see Section 2.4.5, "GPG Public Build Key for Archived Packages".

```
----BEGIN PGP PUBLIC KEY BLOCK----
Version: SKS 1.1.6
Comment: Hostname: pgp.mit.edu
```

mQINBGU2rNoBEACSi5t0nL6/Hj3d0PwsbdnbY+SqLUIZ3uWZQm6tsNhvTnahvPPZBGd199iW YxownnrRkpiQUM/yD7/JmVENgwWb6akZeGYrXch9jd4XV3t8OD6TGzTedTki0TDNr6YZYhC7 jUm9fK9Zs299pzOXSxRRNGd+3H9gbXizrBu4L/3lUrNf//rM7OvV9Ho7u9YYyAQ3L3+OABK9 FKHNhrpi8Q0cbhvWkD4oCKJ+YZ54XrOG0YTg/YUAs5/3//FATI1sWdtLjJ5pSb0onV3LIbar RTN81C4Le/5kd31cot9J8b3EMXL5p9OGW7wBfmNVRSUI74Vmwt+v9qyp0Hd0keRCUn8lo/1V 0YD9i92KsE+/IqoYTjnya/5kX41jB8vr1ebkHFuJ404+G6ETd0owwxq64jLIcsp/GBZHGU0R KKAO9DRLH7rpQ7PVlnw8TDNlOtWt5EJlBXFcPL+NgWbqkADAyA/XSNeWlqonvPlYfmasnAHA pMd9NhPQhC7hJTjCiAwG8UyWpV8Dj07DHFQ5xBbkTnKH2OrJtguPqSNYtTASbsWz09S8ujoT DXFT17NbFM2dMIiq0a4VQB3SzH13H2io9Cbg/TzJrJGmwgoXgwARAQABtDZNeVNRTCBSZWxl YXN1IEVuZ2luZWVyaW5nIDxteXNxbClidWlsZEBvc3Mub3JhY2xlLmNvbT6JAlQEEwEIAD4W  ${\tt IQS8pDQXw7SF3RKOxtS3s7eIqNN4XAUCZTas2gIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgID}$ AQIeAQIXgAAKCRC3s7eIqNN4XLzoD/9PlpWtfHl18eQTHwGsGIwFA+fgipyDElapHw3MO+K9 VOEYRZCZSuBXHJe9kjGEVCGUDrfImvgTuNuqYmVUV+wyhP+w46W/cWVkqZKAW0hNp0TTvu3e Dwap7gdk80VF24Y2Wo0bbiGkpPiPmB59oybGKaJ756J1KXIL4hTtK3/hjIPFnb64Ewe4YLZy  ${\tt oJu0fQ0yA8gXuBoalHhUQTbRpXI0XI3tpZiQemNbfBfJqXo6LP3/LgChAuOfHIQ8alvnhCwx}$ hNUSYGIRqx+BEbJw1X99Az8XvGcZ36VOQAZztkW7mEfH9NDPz7MXwoEvduc61xw1MvEsUIaS fn6SGLFzWPC1A98UMSJgF6sKb+JNoNbzKaZ8V5w13msLb/pq7hab72HH99XJbyKNliYj3+KA 3q0YLf+Hgt4Y4EhIJ8x2+g690Np7zJF4KXNFbi1BGloLGm78akY1rQlzpndKSpZq5KWw8FY/ 1PEXORezg/BPD3Etp0AVKff4YdrDl0kNB7zoHRfFHAvEuuqti8aMBrbRnRSG0xunMUOEhbYS /wOOT10g3bF9NpAkfU1Fun57N96Us2T9gKo9AiOY5DxMe+IrBg4zaydEOovgqNi2wbU0MOBQ b23Puhj7ZCIXcpILvcx9ygjk0Nr75w+XQrFDNeux4Znzay3ibXtAPqEykPMZHsZ2sbkCDQR1 NqzaARAAsdvBo8WRqZ5WVVk61ReD8b6Zx83eJUkV254YX9zn5t8KDRjYOySwS75mJIaZLsv0 YQjJk+5rt10tejyCrJIFo9CMvCmjUKtVbgmhfS5+fUDRrYCEZBBSa0Dvn68EBLiHugr+SPXF 601hXEUqdMCpB6oVp6X45JVQroCKIH5vsCtw2jU8S2/IjjV0V+E/zitGCiZaoZ1f6NG7ozyF ep1CSAReZu/sssk0pCL1fCebRd9Rz3QjSrQhWYuJa+eJmiF4oahnpUGktxMD632I9aG+IMfj  $\verb|tnJNtX32MbO+Se+cCtVc3cxSa/pR+89a3cb9IBA5tFF2Qoekhqo/1mmLi93Xn6uDUh15tVxT||$ nB217dBT27tw+p0hjd9hXZRQbrIZUTyh3+8EMfmAjNSIeR+th86xRd9XFRr9EOqrydnALOUr  ${\tt 9cT7TfXWGEkFvn61jQX7f4RvjJOTbc4jJgVFyu8K+VU6ulNnFJgDiNGsWvnYxAf7gDDbUSXE}$ uC2anhWvxPvpLGmsspngge4y1+3nv+UqZ9sm6LCebR/7UZ67tYz3p6xzAOVgYsYcxoIUuEZX jHQtsYfTZZhrjUWBJ09jrMvlKUHLnS437SLbgoXVYZmcqwAWpVNOLZf+fFm4IE5aGBG5Dho2 CZ6ujngW9Zkn98T1d4N0MEwwXa2V6T1ijzcqD7GApZUAEQEAAYkCPAQYAQgAJhYhBLykNBfD tIXdEo7G1Lezt4io03hcBQJlNqzaAhsMBQkDwmcAAAoJELezt4io03hcXqMP/01aPT3A3Sg7 oTQoHdCxj04ELkzrezNWGM+YwbSKrR2LoXR8zf2tBFzc2/T198V0+68f/eCvkvqCuOtq4392 Ps23j9W3r5XG+GDOwDsx0gl0E+Qkw07pwdJctA6efsmnRkjF2YVO0N9MiJA1tc8NbNXpEEHJ Z7F8Ri5cpQrGUz/AY0eae2b7QefyP4rpUELpMZPjc8Px39Fe1DzRbT+5E19TZbrpbwlSYs1i CzS5YGFmpCRyZcLKXo3zS6N22+82cnRBSPPipiO6WaQawcVMlQO1SX0giB+3/DryfN9VuIYd 1EWCGQa300MVu6o5KVHwPg19R1P6xPZhurkDpAd0b1s4fFxin+MdxwmG7Rs1ZA9CXRPpzo7/ fCMW8sYOH15DP+YfUckoEreBt+zezBxbIX2CGGWEV9v3UBXadRtwxYQ6sN9bqW4jm1b41vNA 17b6CVH6sVgtU3eN+5Y9an1e5jLD6kFYx+OIeqIIId/TEqwS61csY9aav4j4KLOZFCGNU0FV ji7NQewSpepTcJwfJDOzmtiDP4vol1ApJGLRwZZZ9PB6wsOgDOoP6sr0YrDI/NNX2RyXXbg1 nQ1yJZVSH3/3eo6knG2qTthUKHCRDNKdy9Qqc1x4WWWtSRjh+zX8AvJK2q1rVLH2/3i1xe9w cAZUlaj3id3TxquAlud4lWDz =h5nH ----END PGP PUBLIC KEY BLOCK----

To import the build key into your personal public GPG keyring, use <code>gpg --import</code>. For example, if you have saved the key in a file named <code>mysql\_pubkey.asc</code>, the import command looks like this:

```
$> gpg --import mysql_pubkey.asc
gpg: key B7B3B788A8D3785C: public key "MySQL Release Engineering
<mysql-build@oss.oracle.com>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

You can also download the key from the public keyserver using the public key id, A8D3785C:

```
$> gpg --recv-keys A8D3785C
gpg: requesting key A8D3785C from hkp server keys.gnupg.net
gpg: key A8D3785C: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
1 new user ID
gpg: key A8D3785C: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
53 new signatures
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg: new user IDs: 1
gpg: new signatures: 53
```

If you want to import the key into your RPM configuration to validate RPM install packages, you should be able to import the key directly:

```
$> rpm --import mysql_pubkey.asc
```

If you experience problems or require RPM specific information, see Section 2.4.4, "Signature Checking Using RPM".

After you have downloaded and imported the public build key, download your desired MySQL package and the corresponding signature, which also is available from the download page. The signature file has the same name as the distribution file with an <code>.asc</code> extension, as shown by the examples in the following table.

Table 2.1 MySQL Package and Signature Files for Source files

File Type	File Name	
	mysql-8.0.42-linux-glibc2.28-x86_64.tar.xz	
	mysql-8.0.42-linux-glibc2.28- x86_64.tar.xz.asc	

Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file:

```
$> gpg --verify package_name.asc
```

If the downloaded package is valid, you should see a Good signature message similar to this:

The Good signature message indicates that the file signature is valid, when compared to the signature listed on our site. But you might also see warnings, like so:

```
$> gpg --verify mysql-8.0.42-linux-glibc2.28-x86_64.tar.xz.asc
gpg: Signature made Fri 15 Dec 2023 06:55:13 AM EST
gpg: using RSA key BCA43417C3B485DD128EC6D4B7B3B788A8D3785C
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: BCA4 3417 C3B4 85DD 128E C6D4 B7B3 B788 A8D3 785C
```

That is normal, as they depend on your setup and configuration. Here are explanations for these warnings:

- gpg: no ultimately trusted keys found: This means that the specific key is not "ultimately trusted" by you or your web of trust, which is okay for the purposes of verifying file signatures.
- WARNING: This key is not certified with a trusted signature! There is no indication that the signature
  belongs to the owner.: This refers to your level of trust in your belief that you possess our real public key.
  This is a personal decision. Ideally, a MySQL developer would hand you the key in person, but more
  commonly, you downloaded it. Was the download tampered with? Probably not, but this decision is up to
  you. Setting up a web of trust is one method for trusting them.

See the GPG documentation for more information on how to work with public keys.

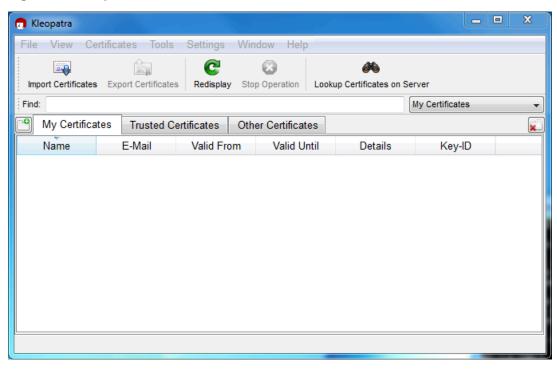
### 2.4.3 Signature Checking Using Gpg4win for Windows

The Section 2.4.2, "Signature Checking Using GnuPG" section describes how to verify MySQL downloads using GPG. That guide also applies to Microsoft Windows, but another option is to use a GUI tool like

Gpg4win. You may use a different tool but our examples are based on Gpg4win, and utilize its bundled Kleopatra GUI.

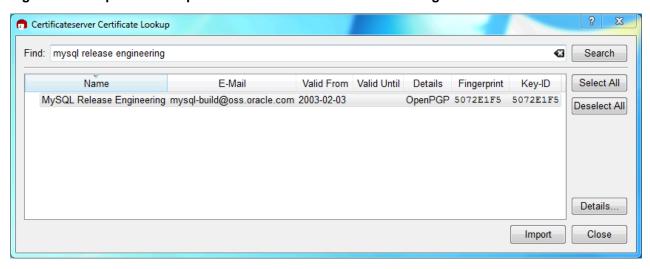
Download and install Gpg4win, and then load Kleopatra. The dialog should look similar to:

Figure 2.1 Kleopatra: Initial Screen



Next, add the MySQL Release Engineering certificate. Do this by clicking **File**, **Lookup Certificates on Server**. Type "Mysql Release Engineering" into the search box and press **Search**.

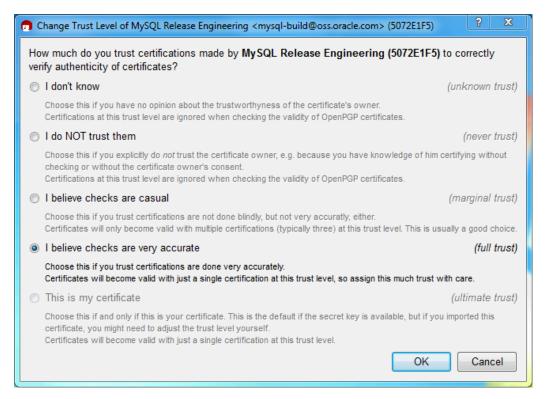
Figure 2.2 Kleopatra: Lookup Certificates on Server Wizard: Finding a Certificate



Select the "MySQL Release Engineering" certificate. The Fingerprint and Key-ID must be "3A79BD29" for MySQL 8.0.28 and higher or "5072E1F5" for MySQL 8.0.27 and earlier, or choose **Details...** to confirm the certificate is valid. Now, import it by clicking **Import**. When the import dialog is displayed, choose **Okay**, and this certificate should now be listed under the **Imported Certificates** tab.

Next, configure the trust level for our certificate. Select our certificate, then from the main menu select **Certificates**, **Change Owner Trust...**. We suggest choosing **I believe checks are very accurate** for our certificate, as otherwise you might not be able to verify our signature. Select **I believe checks are very accurate** to enable "full trust" and then press **OK**.

Figure 2.3 Kleopatra: Change Trust level for MySQL Release Engineering



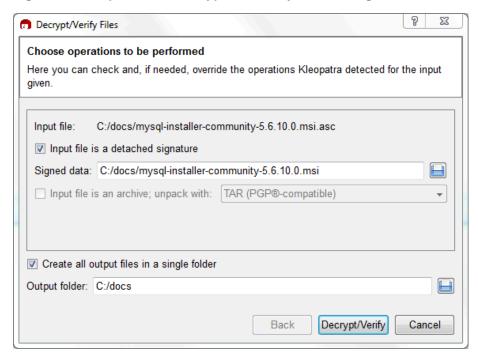
Next, verify the downloaded MySQL package file. This requires files for both the packaged file, and the signature. The signature file must have the same name as the packaged file but with an appended .asc extension, as shown by the example in the following table. The signature is linked to on the downloads page for each MySQL product. You must create the .asc file with this signature.

Table 2.2 MySQL Package and Signature Files for MySQL Installer for Microsoft Windows

File Type	File Name	
Distribution file	mysql-installer-community-8.0.42.msi	
Signature file	mysql-installer-community-8.0.42.msi.asc	

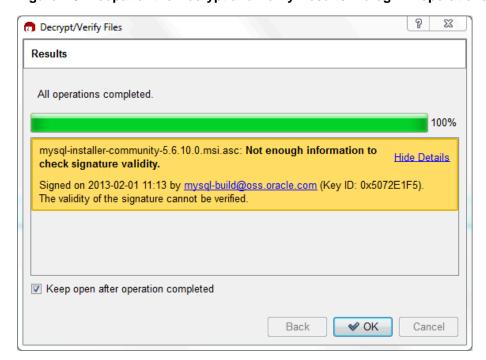
Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file. Either drag and drop the signature (.asc) file into Kleopatra, or load the dialog from File, Decrypt/Verify Files..., and then choose either the .msi or .asc file.

Figure 2.4 Kleopatra: The Decrypt and Verify Files Dialog



Click **Decrypt/Verify** to check the file. The two most common results look like the following figure; although the yellow warning may look problematic, the following means that the file check passed with success. You may now run this installer.

Figure 2.5 Kleopatra: the Decrypt and Verify Results Dialog: All operations completed



Seeing a red The signature is bad error means the file is invalid. Do not execute the MSI file if you see this error.

Results

All operations completed.

Invalid signature.

Signed with unknown certificate 0x8C718D3B5072E1F5.

The signature is bad.

Weep open after operation completed

Back

WOK

Cancel

Figure 2.6 Kleopatra: the Decrypt and Verify Results Dialog: Bad

The Section 2.4.2, "Signature Checking Using GnuPG", section explains why you do not see a green Good signature result.

### 2.4.4 Signature Checking Using RPM

For RPM packages, there is no separate signature. RPM packages have a built-in GPG signature and MD5 checksum. You can verify a package by running the following command:

```
$> rpm --checksig package_name.rpm

Example:

$> rpm --checksig mysql-community-server-8.0.42-1.el8.x86_64.rpm
mysql-community-server-8.0.42-1.el8.x86_64.rpm: digests signatures OK
```

#### Note

If you are using RPM 4.1 and it complains about (GPG) NOT OK (MISSING KEYS: GPG#3a79bd29), even though you have imported the MySQL public build key into your own GPG keyring, you need to import the key into the RPM keyring first. RPM 4.1 no longer uses your personal GPG keyring (or GPG itself). Rather, RPM maintains a separate keyring because it is a system-wide application and a user's GPG public keyring is a user-specific file. To import the MySQL public key into the RPM keyring, first obtain the key, then use rpm --import to import the key. For example:

```
$> gpg --export -a 3a79bd29 > 3a79bd29.asc
$> rpm --import 3a79bd29.asc
```

Alternatively, rpm also supports loading the key directly from a URL:

```
$> rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023
```

You can also obtain the MySQL public key from this manual page: Section 2.4.2, "Signature Checking Using GnuPG".

### 2.4.5 GPG Public Build Key for Archived Packages

The following GPG public build key (keyID 3A79BD29) can be used to verify the authenticity and integrity of MySQL packages versions 8.0.28 through 8.0.35, 8.1.0, and 8.2.0. For signature checking instructions, see Section 2.4.2, "Signature Checking Using GnuPG". It expired on December 14, 2023.

### GPG Public Build Key for MySQL 8.0.28 through 8.0.35, and 8.1.0/8.2.0 Packages

----BEGIN PGP PUBLIC KEY BLOCK--mQINBGG4urcBEACrbsRa7tSSyxSfFkB+KXSbNM9rxYqoB78u107skReefq4/+Y72 TpDvlDZLmdv/lK0IpLa3bnvsM9IE1trNLrfi+JES62kaQ6hePPgn2RqxyIirt2se  $\verb|Si3Z3n3jleg+mSdhAvW+b+hFnqxo+TY0U+RBwDi4oO0YzHefkYPSmNPdlxRPQBMv|\\$ 4GPTNfxERx6XvVSPcL1+jQ4R2cQFBryNhidBFIkoCOszjWhm+WnbURsLheBp7571 qEyrpCufz77zlq2gEi+wtPHItfqsx3rzxSRqatztMGYZpNUHNBJkr13npZtGW+kd N/xu980QLZxN+bZ88pNoOuzD6dKcpMJ0LkdUmTx5z9ewiFiFbUDzZ7PECOm2g3ve Jrwr79CXDLE1+39Hr8rDM2kDhSr9tAlPTnHVDcaYIGgSNIBcYfLmt91133klHQHB IdWCNVtWJjq5YcLQJ9TxG9GQzgABPrm6NDd1t9j7w1L7uwBvMB1wgpirRTPVfnUS Cd+025PEF+wTcBhfnzLtFj5xD7mNsmDmeHkF/sDfNOfAzTE1v2wq0ndYU60xbL6/ yl/Nipyr7WiQjCGOm3WfkjjVDTfs7/DXUqHFDOu4WMF9v+oqwpJXmAeGhQTWZC/Q hWtrjrNJAgwKpp263gDSdW70ekhRzsok1HJwX1SfxHJYCMFs2aH6ppzNsQARAQAB tDZNeVNRTCBSZWxlYXNlIEVuZ2luZWVyaW5nIDxteXNxbClidWlsZEBvc3Mub3Jh Y2x1LmNvbT6JAlQEEwEIAD4WIQSFm+jXxYb10EMLGcJGe5Qt0nm9KQUCYbi6twIb AwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRBGe5QtOnm9KUewD/99 2sS31WLGoUQ6NoL7qOB4CErkqXtMzpJAKKq2jtBGG3rKE1/0VAq1D8AwEK4LcCO4 07wohnH0hNiUbeDck5x20pgS5SplQpuXX1K9vPzHeL/WNTb98S3H2Mzj4o9obED6 Ey52tTupttMF8pC9TJ93LxbJ1CHIKKwCA1cXud3GycRN72eqSqZfJGdsaeWLmFmH f6oee27d8XLoNjbyAxna/4jdWoTqmp8oT3bgv/TBco23NzqUSVPi+7ljS1hHvcJu oJYqaztGrAEf/lWIGdfl/kLEh8IYx8OBNUojh9mzCDlwbs83CBqoUdlzLNDdwmzu 34Aw7xK14RAVinGFCpo/7EWoX6weyB/zqevUIIE89UABTeFoGih/hx2jdQV/NQNt hWTW0jH0hmPnajBVAJPYwAu082rx2pnZCxDATMn0el0kTue3PCmzHBF/GT6c65aQ C4aojj0+Veh787QllQ9FrWbwnTz+4fNzU/MBZtyLZ4JnsiWUs9eJ2Vlg/A+RiIKu 3570qylytLqlqYiWfzHFlYjdtbPYKjDaScnvtY8VO2Rktm7XiV4zKFKiaWp+vuVY pR0/7Adgnlj5Jt91QQGOr+Z2VYx8SvBcC+by3XAtYkRHtX5u4MLlVS3gcoWfDiWw CpvqdK21EsXjQJxRr3dbSn0HaVj4FJZX0QQ7WZm6WLkCDQRhuLq3ARAA6RYjqfC0 YcLGKvHhoBnsX29vy9Wn1y2JYpEnPUIB8X0VOyz5/ALv4Hqtl4THkH+mmMuhtndo q2BkCCk508jWBvKS1S+Bd2esB45BDDmIhuX3ozu9Xza4i1FsPnLkQ0uMZJv301s2 pXFmskhYyzmo6aOmH2536LdtPSlXtywfNV1HEr69V/AHbrEzfoQkJ/qvPzELBOjf jwtDPDePiVgW9LhktzVzn/Bj07XlJxw4PGcxJG6VApsXmM3t2fPN9eIHDUq8ocbH dJ4en8/bJDXZd9ebQoILUuCg46hE3p6nTXfnPwSRnIRnsgCzeAz4rxDR4/Gv1Xpz v5wqpL21XQi3nvZK1cv7J1IRVdphK66De9GpVQVTqC102gqJUErdjGmxmyCA1000 RqEPfKTrXz5YUGsWwpH+4xCuNQP0qmreRw3ghrH8potIr0iOVXFic5vJfBTgtcuE B6E6ulAN+3jqBGTaBML0jxgj3Z5VC5HKVbpg2DbB/wMrLwFHNAbzV5hj2Os5Zmva 0ySP1YHB26pAW8dwB38GBaQvfZq3ezM4cRAo/iJ/GsVE98dZEBO+Ml+0KYj+ZG+v yxzo20sweun7ZKT+9qZM90f6cQ3zqX6IfXZHHmQJBNv73mcZWNhDQOHs4wBoq+FG  ${\tt QWNqLU9xaZxdXw80r1viDAwOy13EUtcVbTkAEQEAAYkCPAQYAQgAJhYhBIWb6NfF}$ hvU4QwsZwkZ71C06eb0pBQJhuLq3AhsMBQkDwmcAAAoJEEZ71C06eb0pSi8P/iy+ dNnxrtiENn9vkkA7AmZ8RsvPXYVeDCDSsL7UfhbS77r2L1qTa2aB3gAZUDIOX1n5 11SxMeeLtOequLMEV2Xi5km70rdtnja5SmWfc9fyExunXnsOhq6UG872At5CGEZU Oc2Nt/hlGtOR3xbt3O/Uwl+dErQPA4BUbW5K1T7OC6oPvtlKfF4bGZFloHgt2yE9 YSNWZsTPe6XJSapemHZLPOxJLnhs3VBirWE31QS0bR15Azl0/fg7ia65vQGMOCOT  $\verb|LpgChTbcZHtozeFqva4IeEgE4xN+6r8WtgSYeGGDRmeMEVjPM9dzQObf+SvGd58u| \\$ 2z9f2agPK1H32c69RLoA0mHRe7Wkv4izeJUc5tumUY0e8OjdenZZjT3hjLh6tM+m rp2oWnQIoed4LxUw1dhMOj0rYXv6laLGJ1FsW5eSke7ohBLcfBBTKnMCBohROHy2 E63Wqqfsdn3UYzfqZ8cfbXetkXuLS/OM3MXbiNjq+E1YzjqWrkayu7yLakZx+mx6 sHPIJYm2hzkniMG29d5mG17ZT9emP9b+CfqGUxoXJkjs0gnD144bwGJ0dmIBu3aj VAaHODXyY/zdDMGjskfEYbNXCAY2FRZSE58tgTvPKD++Kd2KGp1MU2EIFT7JYfKh HAB5DGMkx92HUMidsTSKHe+QnnnoFmu4gnmDU31i =Xqbo

```
----END PGP PUBLIC KEY BLOCK----
```

The following GPG public build key (keyID 5072E1F5) can be used to verify the authenticity and integrity of MySQL 8.0.27 packages and earlier. For signature checking instructions, see Section 2.4.2, "Signature Checking Using GnuPG".

### GPG Public Build Key for MySQL 8.0.27 Packages and Earlier

----BEGIN PGP PUBLIC KEY BLOCK----Version: SKS 1.1.6 Comment: Hostname: pgp.mit.edu mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bP1UWC3RODjQRey CITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZfw2vOUgCmYv2hW0h yDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3BqOxRznNCRCRxAuAuVztHRcE AJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNWhxwR9pRWVArNYJdDRT+rf2RUe3vpquKN QU/hnEIUHJRQqYH08qTxvxXNQc7fJYLVK2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sB gACyP/Vb7hiPwxh6rDZ7ITnEkYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgt obZX9qnrAXw+uNDIQJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAn Wqcyefeprv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q2TX1TUUwgUmVs ZWFzZSBFbmdpbmVlcmluZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iEYEEBECAAYF AlldBJ4ACqkQvcMmpx2w8a2MYQCqqa9wXfwOe/52xq0RTkhsbDQhvdAAn30njwoLBhKdDBxk hVmwZQvzdYYNiGYEExECACYCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAUCTnc+KgUJE/sC FQAKCRCMcY07UHLh9SbMAJ411+qBz2BZNSGCZwwA6YbhGPC7FwCgp8z5TzIw4YQuL5NGJ/sy 0oSazqmIZgQTEQIAJgUCTnc9dgIbIwUJEPPzpwYLCQgHAwIEFQIIAwQWAgMBAh4BAheAAAoJ EIxxjTtQcuH1Ut4AoIKjhdf70899d+7JFq3LD7zeeyI0AJ9Z+YyE1HZSnzYi73brScilbIV6  $\verb|syhpbbmragapahsjBgsJCAcDagQVaggDBBYCAwECHgECF4ACGQEFalGUkToFCRU31aoACgkQ| \\$ jHGN01By4fWLQACfV6wP8ppZqMz2Z/gPZbPP7sDHE7EAn2kDDatXTZIR9pMgcnN0cff1tsX6  $\verb|iGkeexeCaCkCGyMGCwkiBwMCBBUCCAMefgiDaQieaQiXgaizaQuCuwhuZgUJGmbLywAKCRCM|\\$ cY07UHLh9V+DAKCjS1gGwgVI/eut+5L+12v3ybl+ZgCcD7ZoA341HtoroV3U6xRD09fUgeqI bAQTEQIALAIbIwIeAQIXgAIZAQYLCQgHAwIGFQoJCAIDBRYCAwEABQJYpXsIBQkeKT7NAAoJ EIxxiTtOcuH1wrMAnRGuZVbriMR077KTGAVhJF2uKJiPAJ9rCpXYFve2IdxST2i7w8nyqefV a4hsBBMRAqAsAhsjAh4BAheAAhkBBqsJCAcDAqYVCqkIAqMFFqIDAQAFAlinBSAFCR4qyRQA CgkQjHGNO1By4fVXBQCeOqVM1XfAWdq+QqaTAtbZskN3HkYAn1T8L1bIktFREeV1KrQEA7fg 6HrQiGwEExECACwCGyMCHgECF4ACGQEGCwkIBwMCBhUKCQgCAwUWAgMBAAUCXEBY+wUJI87e 5AAKCRCMcY07UHLh9RZPAJ9uvm0zlzfCN+DHxHVaoFLFjdVYTQCfborsC9tmEZYawhhogjeB kZkorbyJARwEEAECAAYFAlAS6+UACgkQ8aIC+GoXHivrWwf/dtLk/x+NC2VMDlg+vOeM0qgG 111hXZfiNsEisvvGaz4m8fSFRGe+1bvvfDoKRhxiGXU48RusjixzvBb6KTMuY6JpOVfz9Dj3 H9spYriHa+i6rYySXZIpOhfLiMnTy7NH2OvYCyNzSS/ciIUACIfH/2NH8zNT5CNF1uPNRs7H  $\verb|shzzz| ?pOltjttWiF4cq/Ij6Z6CNrmdj+SiMvjYN9u6sdEKGtoNtpycgD5HGKR+I7Nd/7v56y| | Shzzz| ?pOltjtWiF4cq/Ij6Z6CNrmdj+SiMvjYN9u6sdEKGtoNtpycgD5HGKR+I7Nd/7v56y| | Shzzz| | Shzz| | Shzzz| | Shzzz| | Shzzz| | Shzzz| | Shzzz| | Shzzz| | Shzz| | Shzzz| | Shzz| | Shzz$ haUe4FpuvsNXiq86K9tI6MUFS8CUyy7Hj3kVBZOUWVBM053knGdALSyq0r50DA3jMGKV14Zn Hje2RVWRmFTr5YWoRTMxUSQPMLpBN1kBHAQQAQ1ABqUCU1B+vQAKCRAohbcD0zcc8dWwCACW XXWDXIcAWRUw+j3ph8dr9u3SItljn3wBc7clpclKWPuLvTz7lGgzlVB0s8hH4xgkSA+zLz16 u56mpUzskF17f1I3Ac9GGpM40M5vmmR9hwlD1HdZtGfbD+wkjlqgitNLoRcGdRf/+U7x09Gh SS7Bf339sunIX6sMgXSC4L32D3zDjF5icGdb0kj+31CrRmp853dGyA3ff9yUiBkxcKNawpi7 Vz3D2ddUpOF3BP+8NKPg4P2+srKgkFbd4HidcISQCt3rY4vaTkEkLKg0nNA6U4r0YgOa7wIT SsxFlntMMzaRg53QtK0+YkH0KuZR3GY8B7pi+tlgycyVR7mIFo7riQEcBBABAgAGBQJcSESc AAoJENwpi/UwTWr2X/YH/0JLr/qBW7cDIx9admk5+vjPoUl6U6SGzCkIlfK24j90kU0oJxDn FVwc9tcxGtxK8n6AEc5G0FQzjuXeYQ1SAHXquZ9CeGjidmsrRLVKXwOIcFZPBmfS9JBzdHa9 W1b99NWHOehWWnyIITVZ1KeBLbI7uoyXkvZgVp0REd37XWGgYEhT0JwAXnk4obH6djY3T/Hf D70piuvFU7w84IRAqevUcaDppU/1QluDiOnViq6MAki85Z+uoM6ojUZtwmqXDSYIPzRHctfx Vdv3HS423RUvcfpMUGG94r7tTOSXhHS9rcs61zLnK184J0xzI5bWS/Fw+5h40Gpd4HTR/kiE Xu2JARwEEAEIAAYFAlaBV3QACgkQRm7hv+CThQqT0wf9Ge3sRxw+NIkLkKsHYBTktjY0yv49 48ja5s9awR0bzapKOMaluEgfwtKD8/NCgYeIVYyaZlYmS1FP51yAtuzdvZXAI0DAITyM4d1S RCESjCCiZ028eIEcoeM/j+UXrwo4+I7/abFhiSakzsFZ/eQHnsMnkJOLf8kug3vMXjSoiz+n T14++fBK2mCVtu1Sftc877X8R7xUf0KYAGibnY+RAi7E2JVTMtWfdtJaqt315y6ouTrLOM9d 3ZeEMdYL1PCmXrwZ4+u7oTNC26yLSbpL+weAReqH8jGsV1UmWWMXvkm+ixmrnN66WvSLqQ6K P5jWnowV9+KEhNnWBOaT4Iu8rYkBIgQQAQIADAUCTndBLgUDABJ1AAAKCRCXELibyletfAnx B/9t79Q72ap+hzawzKHAyk3j990FbB8uQDXYVdAM5Ay/Af0eyYSOd9SBgpexyFlGL404dd7U /uXwbZpAu5uEGxB/16Mq9EVPO5YxCR0ir7oqi6XG/qh+QJy/d3XG07ZbudvnLFy1UE+tF8YU Z5sm9lrnwPKYI2DIa0BToA7Pi95q82Yjb4YgNCxjrr61g09n4LHDN1i74cNX0eas19zp14zS acGftJGOrPEk+ChNCGKFNq/qr9Hn/ank29D8fzg6BLoaOix8ZzZ25QPMI/+SF4xEp/07IoI4 dA+0m4iPz76B+ke0RTsqNRfVKjdz2fQ9214G9yWwNulGcI3FBZTiYGi3iQEiBBABAqAMBQJO iLYZBQMAEnUAAAoJEJcQuJvKV618tkAH/2hGrH40L3xRAP/CXEJHK3O+L8y4+duBBQ8scRqn XS28SLfdL8f/ENH+1wah9jhyMC+jmyRldd5ar3cC/s8AJRvOSDRfR5KvagvrDLrrF+i/vYDB K5f6JQrryq0poupEuK0zTbLxo1FX+CAq+3tQy8aY6+znItpiWhvK8ZoULYKV+Q063YyVWdBk KadgELA6S08aQTGK7bJkyJ9xgbFBykcpUUbn0p4XZwzZ3jFgzwcmqRIYZbfTosVVLJ5HAb7B

u22AukPlsz9PZvd8X8nfmtoJIwt15qtFOrxrKA+X5czswzZ5H3jprDqOY6yA0EStu+8h1CPo LtY50oXmh5D/KdphSKDM33Z9b/3MHzK5CWeCQUkaJ1gxtyLW1HWyLOIhUkW6xHdmieoA8Yr9 JS1r1jopYuGZztz1ScQeSWr8190xnZZVIjKReVy2rDSxtv7PV5wR3gby72PmKWUw7UHfqtBr JgA+h5ctfx1jhXIUtUZpDTStZAFgVmunDXoBNZtYYk/ffY1J8KTjNmrqRcRbTurSy3dgGAAA Z01DIR5kJrh3ikFFJfrXz0qODoYOchxqI4Xoc7o8uv19GUuvk5sKBT4b2ASF+JXAMRX0T7v8 Gralhn3CGGQGpZDN2ldM1Mzbi5oSETTUQ87nN4I7bXirqYkBIgQQAQIADAUCTqumAQUDABJ1 AAAKCRCXELibyletfMCHB/9/0733PXrdjkVlUjF7HKpdD8xy324oe5cRWdEVhsDj11AsPhLv c37M3uCf2MV5BwGjjDypVRX3hT+1r9VsuR201ETKmU8zhdjxgTlZ931t/KDerU9sSJWOT33m wEX7b50j31hgqy2Bc+qOUfSNR8TIOZ7E6P6GynxFzreS+QjHfpUFrg41FgV58YCEoMyKAvZg CFzVSQa2QZO4uaUIbAhXqW+INkPdEl/nfvlUWdoe/t5d/BDELAT4HEbcJRGuN/GNrExOYw/I VC40iQEiBBABAgAMBQJOvNkcBQMAEnUAAAoJEJcQuJvKV618xSkH/izTt1ERQsgGcDUPqqvd 8exAk1mpsC7IOW+AYYtbOjIQOz7UkwUWVpr4R4sijXfzoZTYNqaYMLbencgHv25CE14PZnVN xWDhwDrhJ8X8Idxrlyh5FKt0CK53NT9yAsa1cg/85oVqZeB0zECGWgsVtIc8JmTJvTSmFVrz vmJ0f/BIp+cUp10AnOyx0JzWNkQZWTmsVhxY6skBEd4/+2ydv9TEoESw207t7c3Z7+stWcTK RUg7TrqHPvFkr9U0FKnHeTeqPhc8rjUgfLaJASIEEAECAAwFAk70o7wFAwASdQAACgkQlxC4 m8pXrXza3Af/QjONcvE3jme8h8SMLvlr6L1l1uWpHyWwcvgakRJwUojRrSVPghUAhjZEob4w CzZ4ebRR8q7AazmOW5Fn1GoqtzrWxjRdBX3/vOdj0NvXqCFfTgmOSc4qz98+Lzuu8qQH9DE1 ZLyptv96tGZb5w82NtHFMU9LkkjAVYcDXqJ4USm90CApXqd+81VOrWuM8NycgD01k3ZKZQXH 1DHdJFzohNtqbWGMWdjqwKHoBSHEsjZ/WarXEf0+oTLjZSbrymtGpPInsijHWD9QMOR55RwC DtPW+JPPu5elLdaurjPOjjI6lol8sNHekjmDZmRI0ZMyjprJITg4AG3yLU9zU+boCYkBIgQQ  ${\tt AQIADAUCTv18VgUDABJ1AAAKCRCXELibyletfNeIB/0Wtd7SWBw8z61g5YwuG/mBcmLZVQFollow} \\$ vGnJFeb+QlybEicqrUYJ3fIPj8Usc27dlwLP+6SU8BtldYjQ7p7CrQtaxG2SWYmNaJ50f6Eb JpO/31WSWiNEgF3ycFonoz3yuWMwEdMXBa+NAVV/gUtE1BmoeW+NwKSrYN30FYmkZe+v+Ckq SYwlg0r9+191FwKFvfk0jX1ZGk6GP27zTw49yopW9kFw/AUZXlwQHOYAL3gnslwPz5LwiTyJ QkxAYYvdByZk4GjOi+HzqGPspNIQEeUteXzfbPz0fWEt64tudegYu/fN5QVLGS/WHfkuFkuo gwNBFcu5TPEYcwGkuE/IZZEniQEiBBABAgAMBQJPBAkXBQMAEnUAAAoJEJcQuJvKV618AG8H /OLLrlyM6osbLAhOzcKmD//jSOZ2FIBXGKUmK8/onlu5sUcMmVVPHnjUO/mHiFMbYFC655Di rVUKlIZb6sdx2E/K+ZkZHPWvF1BAaHUO/QGh3Zzc8lVJg9KtFLAJkmQkc61VEF2MriaRlvlo VPNr50iv2TH0PgVxdV3goBL6EdAdgdwCvy23Z44v0p0QVNQt4aJKg2f49XO/N1+Gd2mEr7wX  $\verb|aN9DZQq5zTU7uTRif3F1XHQ4bp8TWBK3Mu/sLlqZYtF3z7GH4w3QbwyA2CWkGgTGwQwyU8Fh| \\$ JQdrqXGl0w0y6JusjJWdwTlfxA6Eia3wrSw2f8R1u6V0k0ZhsMu3s7iJASIEEAECAAwFAk8V 1NwFAwASdQAACgkQlxC4m8pXrXzijAf7Bn+4ul7NedLGKB4fWyKDvZARcys13kNUcIl2KDdu j4rliaY3vXT+bnP7rdcpQRal3r+SdqM5uByROHNZ+014rVJIVAY+ahhk/0RmdJTsv791JSkT FuPzjYbkthqCsLIwa2XFHLBYSZuLvZMpL8k4rSMuI529XL48et1K7QNNVDtwmHUGY+xvPvPP GOZwjmX7sHsrtEdkerjmcMughpvANpyPsFe8ErQCOrPhDIkZBSNcLur7zwj6m0+85eUTmcj8 1uIIk4wjp39tY3UrBisLzR9m4VrOd9AVw/JRoPDJFq6f4reQSOLbBd5yr7IyYtQSnTVMqxR4 4vnQcPqEcfTtb4kBigQQAQIADAUCTzltCwUDABJ1AAAKCRCXELibyletfAo9CACWRtSxOvue Sr6Fo6TSMqlodYRtEwQYysEjcXsT5EM7pX/zLgm2fTgRgNzwaBkwFqH6Y6B4g2rfLyNExhXm NW11e/YxZgVRyMyRUEp6qGL+kYSOZR2Z23cOU+/dn58xMxGYChwj3zWJj+Cjw9U+D/6etHpw UrbHGc5HxNpyKQkEV5J+SQ5GDW0POONi/UHlkgSSmmV6mXlqEkEGrtyliFN1jpiTRLPQnzAR  $198 \pm Jo3G \pm G5Y \pm GFb \\ N1 \pm un1 \\ sXN9 \\ v/s4 \\ dzb \\ V0 \\ mcHvAq/1W + 2AT6OJDD \\ 204 \\ pp/mFxKBFi4XqF6$ 74HbmBzlS7zyWjjT2ZnujFDqEMKfske/OHSuGZI34qJ3iQEiBBABAgAMBQJPSpCtBQMAEnUA AAoJEJcQuJvKV618L1QH/ijaCAlgzQIvESk/QZTxQo6Hf7/ObUM3tB7iRjaIK0XWmUodBpOC 3kWWBEIVqJdxW/tbMbP8WebGidHWV4uX6R9GXDI8+egj8BY8LL807gKXkqeOxKax0NSk5vBn gpix2KVlHtWIm7azB0AiCdcFTCuVElHsIrhMAqtN6idGBVKtXHxW3//z9xiPvcIuryhj8orS IeJCtLCjji7KF2IUgCyyPJefr/YT7DTOC897E1I01E4dDymNur41NjobAogaxp6PdRNHBDum y8pfPzLvF3OY4Cv+SEa/EHmCOTHTamKaN6Jry/rpofqtueiMkwCi81RLqQd0ee6W/iui8Lwp  $/\,2KJASIEEAECAAwFAk9V2xoFAwASdQAACgkQlxC4m8pXrXy9UQgAsVc8HNwA7VKdBqsEvPJg$ xVlm6Y+9JcqdQcA77qSMClc8n6oVF1RpI2yFnFUpj1mvJuW7iiX98tRO3QKWJIMjEPovgZcS bhVhgKXiU87dtWwmcYhMsXBAYczbsSaNWhOIPwKHuQ+rYRevd0xGD0013P7pocZJR850tM9e 5809bzdsRYZpFW5MkrD7Aity5GpD65xYmAkbBwTjN4eNlp0nHVdSbVf4Fsjve6JC6yzKOGFB VU1TtAR2uPK6xxpn8ffzCNTA1vKXEM8Hgjyq4LWSdDTBIevuAqkz4T2eGJLXimhGpTXy7vz+ yletffBEB/9RmWSSkUmPWib2EhHPuBL6Xti9NopLOmj5MFzHcLtqoommKvpOUwr1xv0cZMej ZenU3cW1AvvY287oJwmkFRFu9LJviLSGub9hxtQLhjd5qNaGRFLeJV8Y0Vtz+se2FWLPSvpj mWFdfXppWQO/kIgVZoXcGJQrQWcetmLLgU9pxRcLASO/e5/wynFXmgSajxWzWHhMvehvJTOq siYWsQxqT/XaWQTyJHkpYJoXx4XKXnocvc8+X3QkxAFfOHCwWhYI+7CN8znDqxYuX//PKfDG 2Un0JHP1za8rponwNG7c58Eo3WKIRw0TKeSwOc1cSufnFcrPenmlh2p70EvNRAINiQEiBBAB AgAMBQJPeKdGBQMAEnUAAAoJEJcQuJvKV618YwoIAMn3uqSB4Ge1D61m0pIXJfOcC6BhCZvM 5gRvDtkD6w8b/+r2/eQiSu7ey/riYwB6dm3GzKR7FEbIK6bEuPOUBwvV2tYkZRgTYqXq7NBL uNv7c80GWhC/PqdvdhFn4KAvL0PjVIgr5+mdXyviKqG7uvguYBDtDUMX1qgZpi+fb7EsbJYf  ${\tt EkBR63jGQw04unqT1EXWds17gj+yp4IHbkJmEJMS8d2NIZMPbIlHmN+haTA73DwNkbVD1ata}$ qSLiFIGXRyZy87fikLVIljOJASIEEAECAAwFAk+KdAUFAwASdQAACgkQlxC4m8pXrXwIUQgA

mnkFtxXv4kExFK+ShRwBYOql1/a6D3MbDkUHwn3Q8N58pYIqz1ONrJ/ZO8zme2rkMT1IZpdu WgjBrvgWhmWCqWExngClj0Gv6jI8nlLzjjCkCZYwVzo2cQ8VodCRD5t0lilFU132XNqAk/br U/dL5L1PZR4dV04kGBYir0xuziWdnNayd19DguzPRo+p7jy2RTyHD6d+VvL33iojA06WT+74 j+uls3PnMNj3WixxdNGXaNXWoGApjDAJfHIHeP1/JWlGX7tCeptNZwIgJUUv665ik/QeN2go 2qHMSC4BRBAs4H2aw9Nd9raEb7fZliDmnMjlXsYIerQo7q7kK2PdMYkBIgQQAQIADAUCT5xA QQUDABJ1AAAKCRCXELibyletfOLsCADHzAnM10PtSWB0qasAr/9ioftqtKyxvfdd/jmxUcO1 RUDjngNd4GtmmL7MS6jTejkGEC5/fxzB9uRXqM3WYLY3QV1+nLi/tHEcotivu2vqv4NGfUvW CJfnJvEKBjR8sDGTCxxZQoYoAFbGTP1v9t4Rdo7asy37sMFR2kA4/kU1FDxYtFYFwwZCJpNL hhw0MCI2StI/wIwtA/7TiFCNqHHAKAGeSzKVyKrPdjn8yt7Js2dM6t2NUOwXQ563S4s6JZdR lXUV9oYhlv+gFAuD57UHvinn6rdoXxgj3uoBmk9rWqJDNYgNfwtf1BcQXJnea+rMavGQWihx eV40+BZPx9G6iQEiBBABAGAMBQJPrg39BQMAEnUAAAoJEJcQuJvKV618M4YIAIp9yNCVLGta URSthhmmgE/sMT5h2Uga6a3mXq8GbGa3/k4SGqv51bC6iLILm2b0K81u5m6nxqdZ8XNNMmY9 E+yYTjPsST7cI0xUzbAjKews63WlEUrj/lE2NEtvAjoS2gJB+ktxkn/9IHnqwrgOgUofbw6T hymURI+eqyoDdBp91IOD8Uuq91X+I+C1PPu+NCOyCtcAhOzh+8p7eJeOATEZe2aB1cdUWqqY evEnYNNK8zv/X30MY167YyEgofKoSYKTqEuPHIITmkAfn0qVsBA4/VtLbzGVGyQECmbbA34s 51bMLrYeERF5DnSKcIa665srQ+pRCfJhz6VQXGsWlyWJASIEEAECAAwFAk+/2VUFAwASdQAA CgkQlxC4m8pXrXwDOAf+JEUUKLiqO+iqOLV+LvI09lU4ww7YfXcqz4B9yNG0e5VprfS7nQ0P tMf5dB7rJ6tNqkuHdoCb+w0/31pPEi7BFKXIoSgOz3f5dVKBGo8GBsX+/G/TKSiTenov0PEU 7/DlwvwmsGExmgmsSQgEWTA3y1aVxc9EVC9x0Fi/czcNN1Spj5Qec7Ee9L0yX4snRL1dx30L lu9h9puZgm8bl5FLemPUv/LdrrLDqG9j4m2dACS3TlN14cwiBAf/NvxX3DEPOYTS6fwvKgLY nHlOmKRCwlJ6PArpvdyjFUGWeCS7r4KoMCKY5tkvDof3FhggrQWgmzuPltBkTBQ7s4sGCNww 6okBiqOOAOIADAUCT9GlzwUDABJ1AAAKCRCXELibyletfDj1B/9N01u6faG1D5xFZquzM7Hw EsSJb/Ho9XJRClmdX/Sq+ErOUlSMz2FA9wDQCw6OGq0I3oLLwpdsr9O8+b0P82TodbAPU+ib OslUWTbLAYUi5NH6WW4pKnubObnKbTAmzlw+rvfUibfVFRBTyd2Muur1g5/kVUvw2qZw4BTg Tx3rwFuZUJALkwyvT3TUUrArOdKF+nLtVq3bn8EBKPx2GfKcFhASupOq4kHoKd0mF1OVt9Hh KKuoBhlmDdd6oaEHLK0QcTXHsUxZYViF022ycBWFqFtaoDMGzyUX010yFp/RVBT/jPXSBWtG lctH+LGsKL4/hwz985CSp3qnCpaRpe3qiQEiBBABAgAMBQJP43EgBQMAEnUAAAoJEJcQuJvK V618UEEIALr7RNQkNw1qo7E4bUpWJjopiD00IvynA0r5Eo0r83VX5YY1AfuoMzBGg6ffKiCs drHjEh45aIguu8crQ7p2tLUOOzKYiFFKbZdsT/yliYRu4n28eHdv8VMKGZIA7t0ONIp1YPd2 9pjyVKy4MOo91NfwXM5+tcIzbYL9g+DuhQbYDmy8TVv7KKyY/gqZU1YB6kS49lycQw8WCine FoeD1fb6aP9u0MFivqn2QCAhjXueKC01M2O0jR0wu7jdojN50Jgeo6U0eIHTj2OQmznh8wYG MX2o+1ybSTjjHIp3X8ldYx01Sa3AqwKEBclLdg5yIyAjHq2phROd2s/gjqrWt+uJASIEEAEC AAwFAk/1PVUFAwASdQAACgkQlxC4m8pXrXwn3AgAjWUh311xsQcXo8pdF7XniUSlqnmKYxT+ UZOP711xeaV/yjY+gwyZvf8TWT4R1Rp5IGg6aNLwLaDB31cXBGuXAANGUr+kb1ewviHnCY3Z +PWiuiusle+ofjbs8tFAr3LN3Abj70dME7GOhLyplP2mXIoAlnMDJ0AyrKx5EeA2jS8zCWCu ziiOj4ZwUZAesXchpSO9V9Q86YiPtp+ikV0hmYqZpIXRNcHOpxnVyEW/95MFwi4qpG+VoN57 kWBXv6csfaco4BEIu9X/7y4OLbNuvzcinnHa0Pde5RnRlbEPQBBZyst2YZviWTFsbG8K2xok dotdZDabvrRGMhRzBUwQEokBIgQQAQIADAUCUAZhawUDABJ1AAAKCRCXELibyletfDJUCAC+ 68SXrK4aSeJY6W+4cS6xS//7YYIGDqpX4gSlW1tMIKCIWNhHkZqxKnWClnmvgGhw6VsZ2N0k YdOnIrzEPWL7qp1ZRiE1GDY85dRXNw0SXaGGi7A8s6J9yZPAApTvpMS/cv1JO+IveFaBRHbI RRndS3QgZVXq48RH2OlHep3o7c964WTB/41oZPJ7i0KgsDLdpjC1kJRf09iY0s/3QrjL7nJq 5m14uY16rbqaIoL81C7iyc0UKU9sZGMcPV7H0oOIAy206A3hYSruytOtiC1PnfVZjh14ek2C g+Uc+4B8LQf5Lpha4xuB9xvp1X5Gt3wiPrMzcH89yOaxhR8490+0iQEiBBABAgAMBQJQGC19 BQMAEnUAAAoJEJcQuJvKV618CbcIAJCXDbUt96B3xGYghOx+cUb+x8zcy91yNV8QC2xjd9Mr 02LJTQHfJfQ9Td6LfuoRb7nQHOqJK1/lWE28t9tlH7I+i7ujYwA/fWardRzqCulNXrgFEiQK ZFaDjRYyM0jWG/sA3/Rq2CMBNhBeCcTDuZ8VvRdm0xMPpyavP8D2dM9WBkPHOik4yAIILVkr hWmr0Up0JhRoelfeyqcN/6ClUgeRMIyBYthA55fk2X5+CerommlpDfJJ1FQOv64VSzS68NG8 j9yf66uuL3bB0OdzOMW6Yq/P9wskCDlMbYm/UnHfB5wAuxWpDeAvt/u+vU4xqqEjkUQGp03b 0v1x179maSuJASIEEgEKAAwFAlWg3HIFgweGH4AACgkQSjPs1SbI/EsPUQf/Z6Htrj7wDWU8 vLYv3Fw23ZuJ8t8U/akSNwbq6UGqwqke+5MKC1fpk90ekzu5Q6N78XUII3Qq8HnfdTU0ihYq qd3AlQmO6CG2hEz5xoxR1jJziRCbblJ7qEw8N/KzBcTkHB4+ag6bjFY9U4f9xU3TjPIu7F2V BklAX+cmDo8yzPjDnP4ro0Yabbg0Q9xzvaK/7pFRz+vL/u/lxW7iE7n6vXTiaY1XnIt5xAXX dwfLYmWeAgdc9KXFNlt4lfuqrETtNCHme+JI+B2Tz2gHmMVLHiDV59eLC0uU/uVsOXEd26ib JC4f3KqY9kxuQm325kNzxnMxiwMPCVzsEh7lsYp+OokBMwQQAQgAHRYhBADTXowDFGilEoOK 6kPAyq+7WPawBQJasiYMAAoJEEPAyq+7WPawox0H/i96nkg1ID61ux+i20cOhVZylNJ770Vv 0zfXddWRN/67SuMVjLLiD/WfnDpw6ow6NM7vfEwbmvo1qeFF7rWWTPLm57uZfTk73un3fbaL JiDZyrUStQKK/yhGAZmwulOQq7XBm+u8G9UcFi4XQxuoc5I/v/lUgbxXBADlxlfzpkIDwOaB s23RDiMcWZGcosUkYHX1m8scU0tRANVLQ/PHgttlU13x2PLzrdQm3YUDKUJ9+ynO2jN2sYwt laSohj4UbLnq6pI4CXWZR7XWQs+NX7P3R359FDtw7OhyKoVuIkRFZ1jY0i3wQgwl/Sm2DAg9 31sZDVc/avEUaOO+VuJuvJ+JATMEEAEIABOWIQQGFx4znGT7HFjpuwT3iPLIbOWZfAUCXJ7Q KwAKCRD3iPLIbOWZfGoXB/wN0P3m27fY/6UXTl0Ua3H+24ueUdLipsvR8ZTwEfnwkhLrbgqE 0Em7ZuhZkzv7j856gv/t0ekYYqWGg1CLalD3y371LAGq1tjY3k/g2RWLxLXNdzgXEyFvaNQA oQa9aC2Q7F0yEMwVkkXrGa4MML7IBkrtMds9QPKtfipachPf6tQOFc12zHRjXMZi0eRWyQue 0sLLiJZPn7N8bBAJyZ9IJEpkhNrKS+9J5D1Refj++DwBKDh04kQXZFEZZhxcungQW5oMBQgr uW2hULTLeiEV+C5160nwWJOz6XKJpOJp8PY0bO8pGgToGIYHkoX2x64yoROuZasFDv7sFGX6 7QxyiQEzBBABCAAdFiEEEN0MfMPATUAxIpzAoiiOmODCOrwFAlv/EJIACqkQoiiOmODCOrwq uAf+IVXpOb2S3UQzWJLSQyWG0wQ51go4IBVpHv6hKUhDFj47YdUbYWO+cgGNBjC7FVz54PUM

PIdxImGHE1NHH+DNR8hvvAi+YpnqqdT3g+OgZ6XoYevret5B2b5fRgN1/HWUjaJ/n5g6SMsC +3DrmdMu1FEDnKv/1HwQvOQXKt/U2rXE1ILOmVdMavRJEwkrk2SVwbdeass2EInZVsmWL+ot 9dU5hrkmLA16iHUoK6zF6WaI1oi7UU2kgUF2DNyZG/5AumsNhxE608EAs1zEdN8wibXL48vq  ${\tt Z4Ue9GvImokdlq/r/4BMUdFlqLEZHBkbaklK1zXxl7uMiW3ZIcqpg5HgwYkBMwQQAQgAHRYh}$ BBTHGHD/tHbAjAF4NhhrZPE15/iCBQJZ+o/oAAoJEBhrZPE15/iCyfMH/3YP3ND8jFqIWkmG JaITHP9GhAQda73q7BFIrBHeL033tcLtUbEHXvnIZzulo7jiu9oQBjQvqGqIl5AqH1m7lHaD iAL3VmuUFZ4wys7SODHvSZUW1aPLEdOoLKeiG9J6elu0d/xWZmj86IaHMHrUEm1itMoo0m+U  ${\tt MwVNLFNZrAjCn82DiS6sS0A52t0lpq/jR4v9AYfMZSnd1MLm/CZaZpzWq6aqm7ef7CDfsUvU} \\$ w7VsL3p1s+Jgo6+8RwQ1W2Lgt5ORthvpjPKE1z0qgDpoXTkPOi8M20taD5UZbpByzMZPJXXr +LBrRbs481cPVHx8sxHMh1HsQCiXHDGiTNSaJ1qJATMEEAEIAB0WIQQazDqcUxAL9VrKN9zD LyvJ+reoRgUCW4YZiAAKCRDDLyvJ+reoRptWCACoIgFrvhbr3c1WVq16LJ8UmQLk/6uFFZPN  $\verb|CiR6ZbvzOd+a3gk1G8AhDEW2zoNhfg9+I7yqUBGqn+B1nDZ6psyu8d5EoRUFTm3PghqEccy5||$ KixqoPxBTquzkKGbN8PDLUY5KvpTOLL1YZx1HzSHw4roPsU4rxZtxyu98sSW0cm47VPr069p 91p9rCoHY8Fng7r3w28tVfvLuZ1SK4jtykIvw+M/pVBk9rQVCAJ0JjkAHkT0pkHqsVBYhtu7 mzsXfkQZkeuxdNx6X1fMrbJofzH0GYTT8Knn75Ljhr3hozrsL4Kz4J9gsLHCjkD5XKzLwCFK  ${\tt R6UhhZZr7uhufbqZIyTLiQEzBBABCAAdFiEELLeCvUfxyJI8qMqHHSPVZ6Jn8NcFAltZjFMA}$ CgkQHSPVZ6Jn8NfKSggApk065wFrxq2uqkZKfJGw2mdsGeDVjGq9tMKUWeYVxTNxjiYly8Dc /jrOS3AU6q7X7tAAcmvaXoBfW3xEIXMSH73GeinVG7wnlab6GKPDRKJzXfJ88rF07pX8R1pc ZH+eikiFsN9bcnEycH82bonS7dzyoo6yg2zBqNtsmWYLDg2hcoTw4UHAPwdX6+n99m3VzOqO 8ThQI9hqpUYGvP5qyYahFf+39HSViof+Kq5KKhvSoiS9NzFzYZ0ZszYt+2jozUpAM6XqtEGu TMzXHkE+/V4yI3hIsvHNkXKgDrqjwA+UmT1R4/gBoiRhZ8r4mn1gYI08darQmkppf9MEbcDz /R8c5xY96ntPI2u6hwn5i0BGD/2IdO+VdnBUnyE4k9t2fXKDRtq6LAR2PAD00ehSe4qiR6hw  ${\tt ldaC8yiyg+zgpZusbCLGxbsBdYEqMwTIeFsa8DyPMANpJ0XLkGGf8oC7+6RuAJvlm6DRlurral} \\$ U93/QIG6M2SNsmnPgSZWYV4Y5/G7Xxyj0Fc3gNjjjGGP61CBR01W6rgNPn35sZ9GYCZcGlQA GGrT8mSVoUhPqPCXKz2dZDzsmDHn7rULB6bXcsHiC/nW/wFBpoVOIFIxND0rb1SYyJzPdPtO K6S+o+ancZct8ed/4fUJPBGqrBsuFS1SKzvJfPXjHGtZBitqOE7h57SJATMEEAEIABOWIQQt 9h/1MHY0zPQ0K+NHN096zf003AUCXK2H5QAKCRBHN096zf003OJtB/wKbQN4IjVNkmWxSaBc JABRu/WSbNjoTo/auJV6IRUBpwR130izMw239w5suuWx1phjPq3Pdg1BaKKeQNdeRoiudUjd hydON1cq2wh90073wU2GHeZLi48MopUNksrhHfd/XWV//0LcSpERsqIBVIUi+8DHwFvpCzCz zIRg9lOcQmEtJAFFUtkF9FEeZgO2NPO3fEwkjKDeJYUiB+mD9BliyxhU8apUx/c2zaFGQOCr MllN/gHztAWDcIadK/tujqRWR4wnJ0+ny/HP+bWd18+YjhcWzUQ8FytG+DA3oylQld0w0emt  $\verb|qfn0zqiFkJQdG0M4qtItJYEYH1YpG2yoQHcCiQEzBBABCAAdFiEERVx3fry8YaOOhcAGjZrN| \\$ vi2vIgUFAlnScGAACgkQjZrNvi2vIgW5IQf8DKjeoHF9ChDcb4T01uJJiAUu61xewSRD7iwD 6MjCsaxgMifTD7Bzvdem4finoOul2YAPtlLfIfVtVRtGG97R/Wvs3yjI9NSzxkDGuuE7/IIi 4dKlcKkvijg7G6A8+MGXaQTw8iOePI/44IyG5yogKjno7L4h0f3WguGzmCRUJcgYm23IsaTh Pvdq39ARyHAlrk0hXZ+OqsYBrlW7KLyPrbPA3N+/2RkMz6m+T8ZksOrEdF/90nC9Rky4Wbg4 SJqWQNNSMfgT0rQL2Qvne598FKmltrTJuwBtIrSeuL/dbKt+hkLgnRjnmtA5yPaf0gXvMtfU  ${\tt P9goQMWD+A2BU/bXJokBMwQQAQgAHRYhBFBgHh7ZZZpG0pg7f1ToXvZveJ/LBQJblegpAAoJ}$ EFTOXvZveJ/LS0YH/jpcVprmEGnqlC0mYG2MlRqeK4T8Y6UnHE2zBPc125P4QcQfhgUJ98m4 0B5UkzljreFr9Zebk3pE8r4NBsamlJvi8sGbZONTsX4D3oW9ks0eicKOcTZJgtX5RmSNFh63 +EHbqTneK/NTQIuqRSCOufqCOH6QY1PVsICBlFZUPMfuxRl07EwHKNIHPVBZNlM7AXxdjCMU kCyXn1fVnbCyJWsXQT9polJRnIAJMAtykcYVLNS/IS65U+K1cMshcF+Gil9BuGyckbRuNaSJ ATMEEAEIAB0WIQRh2+o6RdTFb7cSlWG3d+zE2Q5m7gUCWdJutAAKCRC3d+zE2Q5m7rgJB/9k c+prmrnjsq/Lt6d90LqYoavvIeFkAoDhhWgQeEOAD1wgyHIpS6qoMKgvBlvda2r0bmk1kUL2 xQaiDj36wB5yJHauOnFX+3ZJ6QCYUaeoWtqO2ROHvTiuyUdVKC5NtKaHpM1/1P/j1/1ZRWay idggH7EnwDMt+900xD02n5J29Vp9uP01GtMVsVSiJCGcOxwNBgNiXX1BpZbN4bRm5F8DAGiN v4ZI69QZFWbpj8wFVJ/rV4ouvCFPlutVEAuIlKpAj35joXDFJhMvPpnPj84iocGqYPZHKR6j a90+o8dZw3hXObFowjcxsJuQUTVkPuhzqr6kEulampaQ8OGpXCZHiQEzBBABCAAdFiEEZ/mR  ${\tt TQQxCZjglXUwgzhtKKq2evsFAltbmWkACgkQgzhtKKq2evsdragAubfuGlvWX3TTG/VYYrfM}$ laS1Roc034ePoJHK5rLT00/Tnnn0bw38kJM1juyu4Ebfou+ZAlspiWgHad62R1B29Kys/6uC qG2Jvbf716da4oLXeLYd9eb+IKVEiSb2yfbsLtLLB0c/kBdcHUp6A1zz0HV811HWj1Wx8cFU MV7aAQoOfnNBbnNWLzNXXLYGHh47/QmjifE5V8r6UJZGsyv/1hP4JHsQ2nqcM8Vfj+K+HEuu nnxzgWAcQXP/01h111VwoWhsJ1HW+4kwW02DDopdBfLTzCtzcdOkfBcCg8hsmC4Jpxww5eHm saY6sIB32keCpikVOGwdGDbRH7+da8knzokBMwQQAQgAHRYhBG4VA/IlW5kLV/VchhLcHkBr mersBQJaX4N4AAoJEBLcHkBrmersksUH/3M0cypXBnyGIl/yE576MDa0G1xJvciup0ELeyhj 48Y7IAr7XiqDtiPt8t1IiPFF8iaw56vJw5H6UKraOcjZHOH1SwDr5gAWJgMqnq1FX/DxVKif USt81KX0tHN6t6oMESgm2jRKvcWjh6PvEZ1IArxZG4IjrErqWIJjUJR86xzkLyhRVTkUL/Yk uNlli013AlaD/0CGuAnjrluUUXypadtNr7/qsBx8dG6B/VMLWToEDEon76b8BzL/Cqr0eRyg Qz6KWi3hmsK+mE4+2VoDGwuHquM90R0uS9Z+7LUws24mX5QE7fz+AT9F5pthJQzN9BTVqvGc kpI2sz3PNvzBL5WJATMEEAEIAB0WIQR00X0/mB27LBoNhwQL60sMns+mzQUCWoyYfgAKCRAL 60sMns+mzYgnB/9y+G1B/9tGDC+9pitnVtCL2yCHGpGAg+TKhQsabXzzQfyykTgzCHhvqRQc XHz5NSgR0Io+kbGMUUqCaen6OlcORVxYIuivZekJOAG+9kiqWRbyTv4aR6zvh8O5wCyEhhyi ifi65PM7y9lD6i22qTt/JoDnFkP5Ri6Af/fZ9iaIaluQKJCU5xY1Lt/BorGlrGvX5KiZD8xc AjhJRATZ0CJ21qbxISSxELAfH42KzGAvJw/0hARrMk1/eK0HVDpD47mcmC5h/O/HlwPYi0hn xB+6/nuwwtRgMDBufNV0StU43njxCYmGI9/I1z5Vs+zhz8ypw/xCr1U7aAPZQdSSsfEViQEz

BBABCAAdFiEEelR8OpStCJs7bhrK1TniJxBsvzsFAlv+8d0ACqkQ1TniJxBsvzsiFwf/a3lt OuSrFs4M03YVp6LoCM6CwZfvcFl+6B0TAurOiCja9lsNmbusSx0ad7bZy6/kHDXH/eqomXeu O4hkxxBvGK3gZt7iQsr9vsUSbbJnc1zMyOZKlhdxAOLOskttqtPs6hiJ9kUHFGZe47V3c77G GMgi/akIU5PkxhK7+/bbAsW0iK60aXCZ5nAbWlzTQLgJnYrlk4b920rzGe8nDTGzGmSjIGnb YvuD9ZI40DZRWVf1tXqCY643AXFYoOhRxj54uHnMLYhc0I65u2ZGwRiTI0g/en5E8i7WoejA /sR0+cYs711IJwlNRwfqmnJWRGREEHcJ3N52k3X7ayq3qmr3K4kBMwQQAQqAHRYhBJSRYHFB cqf4Tl2vzE+YN4Ly8sn+BQJae/KHAAoJEE+YN4Ly8sn+5ckH/juc2h7bC4OGmRHcZBLAG2vW WEMTc8dAr9ZyJYXzR25W1/Cz/JXgJgMjSrE6m9ptycpvWc6IRlrQM/IqG+ywYFPwNp3PYsc0 1N33yC15W7DPRDTtJE+9yUbSY9FeYraV4ghxiBxD1cDwtd7DFNGNRvBDH7yQHmXBW0K8x6yX Mwl1gj2/MvdFUKmz8Lku940mrbD0i83cnAjUNbN15Wle7hWAIRALt3P1VusjV/XyzxvcSffb mt3CqBCyK9CNyEr27CVkhZ8pcabITx9afMd1UTEii90+qzqcJwcR46bJPZBdavMt56kVCeC0 kG440300k+0ahKXzw4YspZM0046gYRKJATMEEAEIAB0WIQSm5fcyEkLUw6FcN0ZJ1MJhNZ28 bgUCXTJMCQAKCRBJ1MJhNZ28bsgCB/96P1BUdsKgnh/RpmPB+piFQf6Og+97L4fxHuQbzKOe UNCSWNF7saVa5VaPxbV/9jDCTPZI5vBtnJebXtkmLoWFSZaXCYb49SijfvRsRAeX5OSgIRd4 3KMu07nAvbPVYtMChC0/g1T3riF2icC6pgvmNZWm5Nu4pkLzRmQv8U33BAkL7EYIjZZaC/9h o4Sh41/gLNItOxMdsD34sJwBLvEi1pQOa1xNJ4kfQSRD/8ufakE5wfSie/s04w/2Cp7RD9H0 VlD+7FwPO1HQ3XJjONvOzj6uVdwCC5fcmbXbb2bbJ/xe4YVL3xmwWz5m2w+kBSpaZ6VHNocB 8S20mIIPpr70iQEzBBABCAAdFiEEp6WxZJrn5Z0o967I/htVRVZtQSYFAlqnkGEACgkQ/htV RVZtQSYV2Af9E7FLIUi8lqOyYyZuX6skkNf5rNSew+7i5NsiNpQzZMdscJh9eJzyLrePLp7q 9HUOhMF/Fc0SgbDtKSWbfSidXkeaQ2twPj4rP1xxYBc0OY0OX4fNVA5O/pTI9nxIVQCDTljl /WIY+fnj881CkaKWoRJITaotjFmYt+gbJMBn3MMYf0VODeIRozV7//NdkzFXKmJ3fsCDGXXF CVWM1Fn3M91o1fh3FSgKd+0sexUDn5afwWCqjGgiXDsE7fEdwsbnz1rDzWvuqCoZyIh1RXQf QVbiakpzfvtDytC3Vo6F2KzpZ9d69Adhfn2ydAYxL/Xuvk9pWdEBNF4T+HfS9Z30BokBMwQQ AQGAHRYhBPJCF6TG7RrucA13q1lkfneVsjZHBQJawgLrAAoJEF1kfneVsjZHgNsIAIaSJ3gF tBtf0WLxYIo5zhNc1XOnfqUUNjGrXHm5NxoI4Eulpx9dQYCJ++whMFbxpZQTgFAUq8q342EZ raLCWwALZEZmkZjv+FX6bk8sgqZESpU0LJAIqpobKpaaw0Q7LS+XW00SchH1oLFAgDyBeIDZ N/LiTlIdkJelxpDQDtgUHawksqMCbIaBe60B5xvmlNkhnrmnM1p+e3LUd4j+XxACdcY5LSqV zVT4OyD1WkKzk8EAASUI8xysNBEeX9/8/EXaAciECQb3MkYxTQZ4WqCLU0GCG16Sx2fY5zI6 4Y1j/Sfn3JHikJots8eR1D/UxrXOuG5n9VUY/4tTa0UGPuCJAU4EEAEIADgWIQRLXddYAQ10 69GnwU+qS4a3H5yDGqUCX6xjqBoUqAAAAAAAAAX2V1AZ251cGcub3JnYW5uaQAKCRCqS4a3 H5yDGkRfB/9z/5MuAWLwoRLJtnJQzEOW7jsfzYpepL3ocT9tdGcs8jJTH3vh2x4Kp2d0Zaxx Zs7R8ehZO5XJQ/DWdhH+7cifoeXmAEqDnlKSXZQZY/bG054tM6zes3tFTH3dCrn7LF59fQOG OaZHgbFRQJO6F++90Mj9WAgeqGxyEhAlFIxFw4Cuul8OZAUIfq7YISnpkg2Tm/Q0SRRDJE4i /7WJE/HVMB0Rf9KJXuk2BJ1RIpQz8Cf+GVZ5aGI1XdM58QknprnollxoTKhrE74rAGHW7nRD xIxOoP8odiXbLzn//g2m123usqncCKWZONDdVupax3RQ7xsIuFc9Kx4OtjwPQftziQFOBBAB CAA4FiEE6hBKAqPbyqqOC7fUwpbDMFwG9MsFAl8u+m8aFIAAAAAADQAEcmVtQGdudXBnLm9y Z2FubmkACgkQwpbDMFwG9MsIvggAhRfd2Z5WLR6hGxOHu+A+ysjX6xKjcqshCYr8jRuOf1FN vxugQQoFM5pQr15TyhokaU78aDUoIbLnKcxxmH114hXxcRtg/9Y22TidOVN4jjNbc69KvCC4 uANYuAJaI3o5fb1jv8Lx820iRDMhtRqyTdSGdU5//8X5FXCt+HhhzpSNoNtpxyhsKP0PAWao zuETqvxy7t0uy0f1OTbZLI5nb52DxjBdZ1ThnJ2L9RwR2nSGhxjhTFg8LrZWgWNtY5HG+vk9 qbCwaC6ovNJ0G98i0DMrlbyGCbxa4Rv332n1xPf1/EPYWmNPlMu0V3bSCqxVa5u3etA5fw3r qIm333vgFlkBswQQAQoAHRYhBJTatFFgHAZYHkTw9GcRGDP/RljgBQJa7LubAAoJEGcRGDP/ RljgNu8L/jN8j4HSggpnzJ0+3dFjVg7FUHJF6BZ84tv9huhmyrByaIrEfFf9ARn8OizKgdpC /wJT1+KXarvsxdnEDlYSat3HS/sEw3BmZjAeTwPi0ShloiSjYgYRbg3irDskqUHML4hhvMx0 x9nZIag2XoSSH7kPEd5jOb8cd7jJeoGg6Z9Z9lMHuyqTGi0T/EbnhjQfVTxWkSkcDvdxbSuW D96mvZrbRnrMebXKkISb0uVUn3/o11iUo9jXs+Q/03Tb9i0H3eOliP1kcB/kggu9xblIPM+J VaK5Z+zAVLPKTQJi+sP/ayEux0xZzfbZ96WERnzT4E7Wwv8MvaLbybtID28Oy9YoBBYv7CrC tyfrHh1t4v2AedRSZcTPKAaQ5NtLAvIdex0kOvvofaGi+7nmgV00vCZFBSXetvBMZkCapW09 vF7wcahaXpF+0Spl9vE2JiesST7uQobCUm1EjxJP0vMDcO1vIfJHlbIhB/f3PE3rXZIzYTdL s3Kb40ONaUfNy9jYtYkCHAQQAQIABqUCVJqcUqAKCRB3MepTnaVyot2+D/9wAQ+p031VMpYS gMWMNLgjq3z7QrN0NYNpxUXAonxECjUzZKSUPGci+fPKx13ZUenk+ruLgtgJmjmUOR6u1Dov BpDFzhfqbIpjgtMDrnY5sWqxJ+CH2Rb5okEEDJ5qE9DwIMP5iXbf4xjnB0yPiq3sp983PLvy 8ttidWe9FDf8JuhWLHRJHODQjc6LufcHSWKG9fLmCjL2KSPN1696MwR+N95EKCivLL2PlG8c f08Xd8lW1S0cJLh/6TEuZtAnVeo0NUOGUXOPPyhTPP/xhfLeKbkxjtm6rg/jBaIjuuQgUyNN hKnP96/GRWWRHvio6eBPalhUcvImSrCHnqLRpdyMxmK67ZzKZS3YsH0ixozJYE0mNevZ2hEY wB+O5HllqK22YwvJnCLH2ZZWTu2TCUjGZP8hbo2nSoyENlxZio9Gl/v4ypjdlgwrjnnZvxoM yOFeuc47AuzP5QjhtlrWv12C4hYi3YLZvkLVFD0CxAE/CDuHk/4eFG4UC4Mor6+BXwVG7NE1 4qQWrAHjLQ2/sHMpsUqY/5X7+StG/78PLP0HP+PIBCDDTa7W0+6kf0EaGVHKW43IIkVNI2Ps b44tTT+Xhc2mHk44LuzL4Axlywv+CxP9NcKLNFwK4Ck1M8Np6cAKlu+Dw6gjOY1aGHgtdsBQ cIqZj/+ETD0+9NkDXEoeDIkCHAQSAQIABgUCUliwpAAKCRCiKuTrQynFRXZdD/9vb+690GSR t456C6wMLqB1+Ocv9XeaCTiJjLqAL2G6bRH2q2VcNHnU/VMTD2YLVu0eP7ubsirVrmR7nAqL sQ1mKKWvTI+p5aAvn4sL3x3P8vzmGoDAigZ458yGuVpVsBkSPjJBMAkMDfm9kdWxCanzuKXS b591fTg4EtcHPDzoSgABntASgfioVxP2TVPfre282cibeYS+RDlaMTVH25yE1rWDuF2U1CVW SMWY9mskr1+XjPnoO2jz0+jhKB7jyMMfSmJqzgcBNgezFbzX2fPmNnMZzEucVFFHmIhNVmL2 rOwc/s1tSHerG5YIdL3HOJek5xJljzjzFfDrdjmMMl+nO6nO78oePoLNdglQQSqnOyW6gZv8 EIIQ/N1nSi/LEW60z8FFxzoO8TqxMMX9QRLbVE6p+7C0nqolhZf6UEiDIIm+PihF1vPFSV54 +70oLObCshe2g4pbRGWPhIJ4X3ILBQwFMZbn+cIuY3h3B/UpbZE/YSDgRFu5TLtCfBE/1QKX

7QhJknJhQhJ+Dx+Y8h1Cx61Qr0KP5DmOkHYZfAQtdacgrqEr/qNen4QYRdKp0gTne8AV7svB12fAQtdacqrqEr/qNen4QYRdKp0gTne8AV7svB12fAQtdacqrqEr/qNen4QYRdKp0gTne8AV7svB12fAQtdacqrqEr/qNen4QYRdKp0gTne9AV7svB12fAQtdacqrqEr/qNen4QYRdKp0gTne9AV8eI/8PkzvUPaHrax0g6ZSbeWbvEw6czm0qUGJX7iMlJSauIJPrbOjvXT7qIsaqZRRiUSWXo+ m+jzK5qdeRhEIUmlJI/tU/RsGokCMwQQAQgAHRYhBEW+vuyVCr0Fzw71w1CgTQw7ZRfyBQJd RylnDu/SFXPUS01QBj7/Bc2k18934+pUtte+B5KZI2s/28Gn98C2IjxxU+YZ1X1LbUkx0cPA jFWjUh/JSfu6Hif2J0NAG3meySnlmpxl6oZeTojeWolt39PF4N/ay7S2TqIjGSBfxvDlpeIU bnziKsyM5ULbkMdgHssQvyZvrVzQxacRzPK424jXtKR6B2oA0wqMcP4c69UmVKEKIzJNYrn4 Kjs+An8vZvJYAVbiWEyEseTTo3XJePdBNs1xxK2vWLA5PeLkE8bmzHr8iQ3hA0NaY7jSJp3e GrhWIdXV+nfclrFUPghYr5z+ljCSK5sow+aRiED39qd1Y+0iUAy94cqY3MQ4ayGgnB/+YuSx B5jNjCBYJetFWWSJXnkbiYRLjU88dflXCrTbhkSuCu3agOjsBJYUyg/c1Z4eCQgpTWB2cjYQ OucKOsWt8U6qsl12qwYLrORfcP2aCwTTnWIxqIN9F6iMafOsG+za8JY+B8PDJxxwWWz8vCvX  $\verb|ChTYrfiFei8oUqoHYTbw07cxaxkDd2CgXsQMmOczSoXZZPAe8AhsUibDl+BZs/vLZT7HrXtt| \\$ /ggz8LzVCcyQqwmCHurvgjauwjk6IcyZ5CzHFUTYWUjvFqYfAoN15xUZbvPYiQIzBBABCAAd FiEERsRGITzmkUU5TZu635zONxKwpCkFAlxFLcAACgkQ35zONxKwpClKVw/+PfrtIVHFsOdl 2crWBSo5Hifvx9Vn2nPiNKErygB+tPWDS4UwzVUnpZfXCM7bKJFFPeKbitYxN3BlDmVhZMkc  $1 \\ DZMAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm102MAtIPSstO2oX7Tv/C0WOZP1AWkp5m0DPV3iGbGZjwmy5wy5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fNtaWyx5wy8fWfNtaWyx5wy8ffNtaWyy8ffWffNtaWyy8ffWffNtaWyy8ffWffNtaWyy8ffWffWffWffWffWffWffWffWffWffWffWffW$ Wf11LMvgwnFsQ74xobnCpssLgmogXfoLFQNF/VUfRveJ2Ci8raWyAdXFBdAIrejawAx5MMhO /lefq3W3f9bqtJZ5DzLbxQ3Xtqs+RY1ihv1y12lr9vLpgKKGmZ92KDvjv2UXHd7XZ90aPMj7 Rx0MQ1d+5d/tNQ8rLJGuj1I7NqHmLHMz67TvRtP14aNP7Mss80HiEKLYq23kGqXN+6cjG3UM i290uJZaAnTno65Cgsyn7JFKyXDdTOmp3TSoyVsPFq92qgd/jFBf3dJj8c+mZEVXkUFeeUEK  $31 \\ EMGFCH + oE8 \\ un7 \\ nu + XWqFyFSw5wn + PGYDXkSd6z/NyIN5DXa326KV + qpUmIWO1cymm7cmZ4$ KJQt7zgWCxh2DuWQzRlTjeQd8Iw62V8tIOBokWP9Thes18Qk2GOUeCnvczLdevT41qr8IzvV nSwX/LQyxmmz2/dmPhzJ6kA6KQKGOSF6WnV/WuD4kESFKwtABFi6mYQi1F6CynpVw/nu535C  $\tt 4fFG4d+A5G6sKJx//hjoCgmJAjMEEAEIAB0WIQRGxEYhPOaRRTlNm7rfnM43ErCkKQUCXa6eIndexComplete and the temperature of the complete of the complete$ YgAKCRDfnM43ErCkKfNXD/0cTEjvQlgyy3UI3xfhYtRng8fsRXcACjMajnrvYCoRceWwF6D+ Ekvh5hNQqrZsxrD6nozY+iJhkkaQitIj4qw7i4KY03fo613FjeLFXWqf4sfLTANSsRNxawEo  $/\mathtt{JxP1JeOToOgYTkikWOkgZWSs/mqvHAxJZrVq/Zhz06OugfOYVGmGZonU7zP12toiwParIZ9}$ hcZ/byxfNoXEtsQyUHO1Tu8Fdypmk0zYUgZK2kGwXslfOGj5m0M5nfUuVWq5C5mWtO16ZngT LPJ32tRW526KIXXZMTc0PzrQqQvTFHEWRLdc3MAOI1gumHzSE9fgIBjvzBUvs665ChAVE7p2 BU6nx1tC4DojuwXWECVMlqL0HKjC5xvmil12QhseV7Da341I0k5TcLRcomkbkv8IhcCI5q08 lgUq1YwZAMflienJt4zRPVSPyYKa4sfPuIzlPYxXB01lGEpuE5UKJ94ld+BJu04alQJ6jKz2 DUdH/Vg/1L7YJNALV2cHKsis2z9JBaRg/AsFGN139XqoOatJ8yDs+FtSy1t12ulwaT33TqJ0 nHZ8nuAfyUmpdG74RC0twbv94EvCebmqVg2lJIxcxaRdU0ZiSDZJNbXjcgVA4gvIRCYbadl9 OTHPTKUYrOZ2hN1LUKVoLmWkpsO4J2D1T5wXgcSH5DfdToMd88RGhkhH7YkCMwQQAQgAHRYh BH+P4y2Z05oUXOVHZQXCWLGt3v4UBQJhrDYPAAoJEAXCWLGt3v4Uh2oQAMS3sK0MEnTPE+gu 71Li9rMbD/305nlAxBJLX4MzLi2xP1648YV5nq9WMMt6qyp+OVwDXefneYNMgfU2/uu/Wi/o  ${\tt XTHBJuU361mFzhRWPj2h/vtfgDIYG2wio0DNJyaUQwLEi6gqPm0AHhKS4td69R+7qyQsbUIa}$  ${\tt BFgoytxFzxDb5o2hicEOXa573m4myfAdCx5ucYfq+j1XJW9Wgw7ERnF1v9xQDXiuryXWFRdv} \\$ UOOWzVPu9T0gPkcG8NABwqxs280c7n9A19HM2FtDAkD0Licm/I4ZEhFVqvG6Hj966+FeuICw OaefFhthOoi3ycO+pkj1IePz/TmnsplTvvZOXH+6XEMPpPRQpvf5IZKJyrvuzoU8vkXYY2h/ gJHi9HiSIIQ/BVEpvp6UjXvIbNP1K31II88qx9EfT/tv434wlZpC6V1FzE2LtxyNcj/+OUvj 9hKOJ7lKOVpsnBbGiWg809s4sCIZ/ifLfWAKOJgxAEk/GcRkkkCqGNx7HA+coteNHqXLa/Lb 2/r8gGn6kH9YhQootJsGhhSsY+6CW5TM5E+FhSRJU7MFHRpA94N7Hn6OFUK2OXtHyRhxE867 R+ChJaZXbtoQJVNv2Rv9yoZrBki3RoQ6/6/fcnR1x2moTMYg7K8AMMv7ZCfaP6AjPOjTVnMV CpNy1Ao7smOzLAfKbbeXiQIzBBABCAAdFiEEjy2YV7IZJ8NHv36cSrDCiwqTaaEFAmF9XbsA CgkQSrDCiwqTaaFUGw//WSU022Csa6016VN8yJQmf0wCo9sieWDXCdHZ+CB0+gu013EMYR2a gL81qCd6M79fpP8DiLKOJvn9mhXCsjYjTJQUsuNi5kQ/O9gwarRsr7EjJ7R8u81pSh9YP1MS yN6XXf0a4Qy5H0w9idJdb3owKAXSjuRdi/hUExjA8TWliyWrfwiVDQi/aCoLZ4b9p6SfGR3Y  ${\tt gE8UIZLZtdWgsPJHkvdvntTPi4fwMsadBfa2f+m4Wq2CAU5KSfYsVpKAwSQ1OsdUZUK7g+Ui}$ jy//ad7eZ+BAc75blHs7ua2iiF8Sc7MC55ZM5ldkv+0lqJ7td5vOCT1LKJq5PKKUC7YTTh9U PH1ERJ/SWcHNES1YhwLvUO2VRO1PN9H1QkPnEMBOObpmYkNQyLBfFwioJ3ilptYY0IUX5qBM 5UkwgyqMsdyrL+2ozIYc+/A8KUnZXozOAG9LP8gBE5jBJSIkbqsi9Fumf7Q63++g4ojcYpOZ F92X6kQMGqBvkvs8UajR5f/n6QH0je4XFPj4141VM/PPfZSShNGdOOi41+KwozICnQ1+fhwh N0VG4eALSJ6XQEEfJ18PrBRS3sdC70VEMLevEC8ojSQeZE11CLe1qAUoEcmgmXjsODaJn2tt qNYYUxcFOycFnzgWL679C9FVp+DAg9jzDMKsqWo/Lt3IDNF19ZUc93WJAjMEEAEKAB0WIQSC piWCWP+fBOH/9bx9bbut3FAu7gUCW8ygHQAKCRB9bbut3FAu7mOaD/9QJ1MiyKvw9rYqTvkU  ${\tt OSDSLu88g6NP5R9ozgGZegInZ/NzT8u5emYccflnLlfvRQZPnT7YIH4+h25CCGQ5HzXUGENx}$ ndeuG4dm3B10A8hxv+abEM9VYDGqSIvF6z1xObvENOpMgmlmFdDi9O9d6jFFy4Hd6/BWejbU 4M3kfuD39RxaT10EWfqvTVf4GKiLqM71glNB8WrTqxt2t/Mo2h6UPCF7/wPF/idMAbKEn0ye  $\verb|b1WDCaZVXXAQETfNo129hPb2qxPGoCWGw24ySpGrM5We4Nd3bbdGItSZ0mATNM1+m9FY9j30| \\$ vpePFzzYGZ+23EcpxWU+7jWbjZ42ssCW6kx2/ERLVma7FuneEAqUc3qZr/3ZdZOVMvseq8c0 n66D/NRLgMcpOQK62qJfSrxQj6sJCGRY4dxAfdTZWrcxu8UvvcINezGIToQ0y+Mc5LM1vMOd GPmumGkejDstWGmnHi5IogN8ibzyywsbNsO+qDdlUFA2bmVhh2uK7M95kyuMH3GnWbz4IiMx RyUVEyK8yKnEmgOmLG4WiJjksP1jIPf3ztTEVVDJxy1gT3R36lsxd+OabnPOgiz1oFewKaur aWX1e0E6eBWJ95ufookCMwQQAQoAHRYhBM8z5mfkMwAXdpGlbLdWs0L0i1qEBQJcBMl7AAoJ  $\verb|ELdWs0L0i1qEmxwP/jDweTwTh1s+7Pp39L6aLB7nuQzdMleTksPGgmtguRBZipbOYOryEozK||$ 

9hI3Hq/ymV/loINv6GZhieDoZvxrv9eEKqO2eUE0IletSy7znlhV6MB7PBOc29dbCMf5L4qo  $\verb|xUG/f+XfhkRZEkjZRWMlitlERlDU5ghAQ3skLuT9bu3aZkGdBgw0U5qjVvGzYxp2LFpNHXlf|$ Trln3RZoDbRI+E9BPILqZFIZczp/fxRRNkXyogkrGD+0PANFsjySQKd/rr8/Z4isl3AM8CZ7  ${\tt s4tMWM4EVJ20ygnrcMuIEJdXVsR0Ln1gJLuQ9HpWehve0d7/cIZkN7a0fqgE7bMvSPyxWL3minds} \\$ yTA4FwdbrebBr2y7ix1XZ6WtX/rqTvo2HTDFLle0ZwMbbfAtoFX0M01PtXTLmJA15w1G8Nj8 bthWdN4KVFyOpqPt7OXc/G1YNLzcyYQXX5e8Uskmq40OH5cQV5OFEG8qpxTq53wANDdxXGzs NUQe84Qkoyk75nwzVfsi00/OhTZmfIC48esXcs0kTrkSPrFcHktSMoYPmHfV3dTF17ifjz5a C2SL22R+RokWuzGxxpvEaQAWIyCt6izf1a+CjnXPD2Jw3yDC/Oeg68XYiSrbeFdCRzQbS9YP ipUFIlHuCiNZeGg3rFL2N2JodXg2LGORJz1RKazT7uAfRr5z7W1FtDtNeVNRTCBQYWNrYWd1 IHNpZ25pbmcga2V5ICh3d3cubXlzcWwuY29tKSA8YnVpbGRAbXlzcWwuY29tPohGBBARAgAG BQI/rOOvAAoJEK/FI0h4q3QP9pYAoNtSISDDAAU2HafyAY1LD/yUC4hKAJ0czMsBLbo0M/xP aJ60x9Q5Hmw2uIhGBBARAgAGBQI/tEN3AAoJEIWWr6swc05mxsMAnRag9X61Ygu1kbfBiqDk u4czTd9pAJ4q5W8KZ0+2ujTrEPN55NdWtnXj4YhGBBARAgAGBQJDW7PqAAoJEIvYLm8wuUtc f3OAnRCyqF0CpMCTdIGc7bDO517CIMhTAJ0UTGx0O1d/VwydDiKWj45N2tNbYIhGBBARAqAG BQJEgG8nAAoJEAssGH1MQ+b1g3AAn0LFZP1xoiExchVUNyEf91re86gTAKDYbKP3F/FVH7Ng c8T77xkt8vuUPYhGBBARAgAGBQJFMJ7XAAoJEDiOJeizQZWJMhYAmwXMOYCIotEUwybHTYri Q3LvzT6hAJ4kqvYk2i44BR2W2os1FPGq7FQgeYhGBBARAgAGBQJFoaNrAAoJELvbtoQbsCq+ m48An2u2Sujv15k9PEsrIOAxKGZyuC/VAKC1oB7mIN+cG2WMfmVE4ffHYh1P5ohGBBMRAgAG BQJE8TMmAAoJEPZJxPRgk1MMCnEAoIm2pP0sIcVh9Yo0YYGAqORrTOL3AJwIbcy+e8HMNSoN V5u51RnrVKie34hMBBARAgAMBQJBgcsBBYMGItmLAAoJEBhZ0B9ne6HsQo0AnA/LCTQ3P5kv JvDhg1DsfVTFnJxpAJ49WFjg/kIcaN5iP1JfaBAITZI3H4hMBBARAgAMBQJBgcs0BYMGIt1Y AAoJEIHC9+viE7aSIiMAnRVTVVAfMXvJhV6D5uHfWeeD046TAJ4kjwP2bHyd6DjCymg+BdED z63axohMBBARAgAMBQJBgctiBYMGItkqAAoJEGtw7Nldw/RzCaoAmwWM6+Rj1z14D/PIys5n  $\\ \mathtt{W48Hq13hAJ0bL0Bthv96g+7oUy9Uj09Uh411F4hMBBARAgAMBQJB0JMkBYMF1BFoAAoJEH01}$ ygrBKafCYlUAoIb1r5D6qMLMPMO1krHk3MNbX5b5AJ4vryx5fw6iJctC5GWJ+Y8ytXab34hM BBARAqAMBQJCK1u6BYMFeUjSAAoJEOYbpIkV67mr8xMAoJMy+UJC0sqXMPSxh3BUsdcmtFS+ AJ9+Z15LpoOnAidTT/K9iODXGViK6ohMBBIRAgAMBQJAK1k6BYMHektSAAoJEDyhHzSU+vhh JlwAnA/gOdwOThjO8O+dFtdbpKuImfXJAJOTL53QKp92EzscZSz49lD2YkoEqohMBBIRAgAM BQJAPfq6BYMHZqnSAAoJEPLXXGPjnGWcst8AoLQ3MJWqttMNHDblxSyzXhFGhRU8AJ4ukRzf NJqElQHQ00ZM2WnCVNzOUIhMBBIRAqAMBQJBDqqEBYMGlpoIAAoJEDnKK/Q9aopf/N0AniE2 fcCKOlwDIwusuGVlC+JvnnWbAKDDoUSEYuNn5qzRbrzWW5zBno/Nb4hMBBIRAgAMBQJCgKU0 BYMFI/9YAAoJEAQNwIV8g5+o4yQAnA9QOFLV5POCddyUMqB/fnctuO9eAJ4sJbLKP/Z3SAiT pKrNo+XZRxauqIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAnlmxHijft00b KXvucSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJYiKJ AAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8JEg2aLos+5zEY rB/LsohMBBMRAqAMBQI+TU2EBYMJV1cIAAoJEC27dr+t1MkzBQwAoJU+RuTVSn+TI+uWxUpT 82/ds5NkAJ9bnNodffyMMK7GyMiv/TzifiTD+4hMBBMRAgAMBQJB14B2BYMFzSQWAAoJEGbv 28jNgv0+P7wAn13uu8YkhwfNMJJhWdpK2/qM/4AQAJ40drnKW2qJ5EEIJwtxpwapgrzWiYhM BBMRAGAMBQJCGIEOBYMFjCN+AAoJEHbBAxyiMW6hoO4An0Ith3Kx5/sixbjZR9aEjoePGTNK AJ94SldLiESaYaJx2lGIlD9bbVoHQYhdBBMRAgAdBQI+PqMMBQkJZgGABQsHCgMEAxUDAgMW AgECF4AACgkQjHGN01By4fVxjgCeKVTBNefwxq1A61bRr9s/Gu8r+AIAniiKdI11Fh0duUKH AVpr03s8XerMiF0EExECAB0FAkes1LQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAAKCRCMcY07 UHLh9a6SAJ9/PgZQSPNeQ6LvVVzCALEBJOBt7QCffgs+vWP18JutdZc7XiawgAN9vmmIXQQT EQIAHQUCR6yUzwUJDTBYqAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQcuH1dCoAoLC6RtsD 9K3N7N0xcp3PYOzH2oqzAKCFHn0jSqxk7E8by3sh+Ay8yVv0BYhdBBMRAgAdBQsHCgMEAxUD AgMWAgECF4AFAkequSEFCQ0ufRUACgkQjHGNO1By4fUdtwCfRNcueXikBMy7tE2BbfwEyTLB TFAAnifQGbkmcARVS7nqauGhe1ED/vdgiF0EExECAB0FCwcKAwQDFQMCAxYCAQIXgAUCS3Au ZQUJEPPyWQAKCRCMcY07UHLh9aA+AKCHDkOBKBrGb8tOg9BIub3LFhMvHQCeIOOot1hHHUls TIXAUrD8+ubleZaIZQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAABIJEIxx jTtQcuH1B2VHUEcAAQFxjqCeKVTBNefwxq1A6IbRr9s/Gu8r+AIAniiKdI11FhOduUKHAVpr O3s8XerMiGUEExECAB0FAkeslLQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAASCRCMcY07UHLh 9QdlR1BHAAEBrpIAn38+BlBI815Dou9VXMIAsQEk4G3tAJ9+Cz69Y/Xwm6111zteJrCAA32+ aYhlBBMRAgAdBQsHCgMEAxUDAgMWAgECF4AFAktwL8oFCRDz86cAEgdlR1BHAAEBCRCMcY07 UHLh9bDbAJ4mKWARqsvx4TJ8N1hPJF2oTjkeSgCeMVJljxmD+Jd4SscjSvTgFG6Q1WCIbwQw EQIALwUCTnc9rSqdIGJ1aWxkQG15c3FsLmNvbSB3aWxsIHN0b3Aqd29ya21uZyBzb29uAAoJ EIxxjTtQcuH1tT0An3EMrSjEkUv290X05JkLiVfQr0DPAJwKtL1ycnLPv15pGMvSzav8JyWN 31h7BDARAgA7BQJCdzX1NB0AT29wcy4uLiBzaG91bGQgaGF2ZSBiZWVuIGxvY2FsISBJJ20g KnNvKiBzdHVwaWQuLi4ACgkQOcor9Dlqil/vRwCdFo08f66oKLiuEAqzlf9iDlPozEEAn2Eg vCYLCCHjfGosrkrU3WK5NFVgiI8EMBECAE8FAkVvAL9IHQBTaG91bGQgaGF2ZSBiZWVuIGEg bG9jYWwgc2lnbmF0dXJ1LCBvciBzb21ldGhpbmcgLSBXVEYgd2FzIEkgdGhpbmtpbmc/AAoJ EDnKK/Q9aopfoPsAn3BVqKOalJeF0xPSvLR90PsRlnmGAJ44oisY7Tl3NJbPqZal8W32fbqq bikBHAQSAQIABgUCS8IiAwAKCRDc9Osew28OLx5CB/91LHRH0qWjPPyIrv3DTQ06x2gljQ1r Q1MWZNuoeDfRcmgbrZxdiBzf5Mmd361iFiLmDIGLEX8vyT+Q9U/Nf1bRh/AKFkOx9PDSINWY bE6zCI2PNKjSWFarzr+cQvfQqGX0CEILVcU1HDxZlir1nWpRcccnasMBFp52+koc6PNFjQ13 HpHbM3IcPHaaV8JD3ANyFYS410C/S4etDQdX37GruVb9Dcv9XkC5TS2KjDIBsEs89isHrH2+ 3ZlxdLsE7LxJ9DWLxbZAND9OiiuThjAGK/pYJb+hyLLuloCq85ZX81/ZLqE0Kyl55xuTvCq1 tSPmSUObCuWAH+OagBdYSduxiQEiBBABAgAMBQJJKmigBQMAEnUAAAoJEJcQuJvKV618U4wI

AKk/45VnuUf9w1j7fvdzgWdIjT9Lk9dLQAGB13gEVZEVYqtYF5cEZzyx18c7NUTCTNX3qLId  $\verb"ull114A4CQQDg5U9bUwwUKaUfGLaz380mtKtM9V9A4f19H2Gfsdumr8RPDQihfUUqju+d0ycd" and a substitution of the s$ mcUScj48Nctx0xhCCWNjOFPERHi9hjRQq7x6RKyFTLjM5ftdInHCo9S+mzyqz9O+iMgX68Mm +AVgdWSC9L6yGnw6H97GD28oRMGWBTzsmCyqf9I3YutH8mGXRot3QbSJD7/AeZVh1BQwVoJn CT8Eo1pc/OYZkRRndE1thrX0yjuFwTeOzvqeHlgzEW/FtOCBW7iR0WSJASIEEAECAAwFAkoz TogFAwASdQAACqkQlxC4m8pXrXwXiAf+Ked6Mqd98YyTyNiLHhllPulboCnKqj430jLzkfqv 7ytVCu1xMfKrRWRw3fA9LC19mzNQX/So/o/ywsk0nUG2sfEs5FiMk+aC957Ic/MDagmXqKap ZROJbzbZ/KNj9rPCG9kXPGa9sUn6vk39nnv4hri30tNKpM0fMxRhpcoNoCrNl4rs/QTpdRpp 7KBuNaMEtDU7R7OjMDL4qT+BcCmYMIYW4dIV7tmaC0VxtcszZcVCkxSigRMPZHwxSx37GdCx 9/+ Tq1A4vGL6NQSxZKv+Kqa+WTqBngO16YGO6FxdiXEliNRpf1mafmz6h8XgYXFGpehjuX1n601z0BffuWbpL4kBiqQQAQIADAUCSkRyCqUDABJ1AAAKCRCXELibyletfPaaB/9FCSmYwz7m vzOfHZOlEAYeLnCS290XGW89o4FYTbw0PBOulygyqj2TMCK68RCNU2KFs/bXBHeS+dDzitMA fSaULYi7LJuCCmrDM5SX5aLSj6+TxkDQDR1K1ZE3y6qd4Kx3VeeoN7Wu+oLj/3Jjbbe0uYCQ+/PniRra9f0Z0neTExZ7CGtVBIsKS1CnKBTR26MZMOom2eTRZwGFUX1PzuW/dbZ4Z0+J6XMd Tm2td7OYYWPbV3noblkUrxyjtGtO3ip3Oe3zSCWHUFMaaEuXOMw8tN51wy6ybcPVAH0hOiBw b3iCFJ/20QqaZEno6edYzkqf0pwvrcTmiPb+Vj0fnjBJiQEiBBABAgAMBQJKVj5HBQMAEnUA AAOJEJcQuJvKV61845AH/R3IkGIGOB/7x3fI0gOkOS0uFljDxysiM8FV06BfXbFpRgFMZxAh NFUdKCDN98MDkFBd5S5aGkvhAHS7PVwQ8/BIyJaJeUG3AXmrpFV/c9kYn1+YW50Q9E7tKu51 5UOj1Y/weNtC04u6Rh/nrp6CvMBhH2nvhSBZ+2kO2auqtFOhuK6+wUHGixt5EK8RAKs3Sf6n kP2EJUHzy1Q8ec5YDiaV24AVkPFBZMCkpD3Z+seIGrL4zUkV7PPY4zd9g34Oqj8JvtnA4AD/  ${\tt Z1vBLujLixcQdt9aieOySA9DAVgHbe2wVS4zi5nBURsmD5u96CUOwNK1sOV+ACtdIv/T5qSU}$ VweJASIEEAECAAwFAkpoCoQFAwASdQAACgkQlxC4m8pXrXysfQf+IJyIPhTphk0kGPQY3v9e 3znW30VahyZxoL6q25eeQWGmVeTF1U4JThUEyzgYGip8i9qBsFPJ9XgOL5bxTGv7/WOK7eX8  $\verb|e+gXHB3A2QYbrM0GFZKN3BCkbA++HmvJXU58tf+aBCB00bG+rPn6QUNSPibu4tp65TaPVPSV| \\$ HjNTTICxu3sneHB+okJcc5z1ubme8nAytKb6x0JM/keNSXAev2ZN7zG5m+Pqw7/DQ/qCoqzG ML1bu1P2rSh8bYpJPC3vAVuHTmxsbhRBg417j5KiHf4qMBrVzRy+YiHhwpf2p8JbCGF141+H yletfBjrCACDd/zvoveoNlNiUUBazelcGXwaxSvUMSROUQNkxkoMzfA+aFpYFHWEwDfLqndp  $\verb|ojtikgkESd5f0Djt26olfeklvx3mpzfGz8l39KzDM1i6+7Mtg7DnA3kvfVIuZBNDwqoTS6hH| \\$ KcGa0MJDqzZQqJ9Ke/7T7eY+HzktUBLjzUY2kv5VV8Ji0p6xY27jT73xiDov00ZbBFN+xBtx 2iRmjjgnPtjt/zU5sLiv9fUOA+Pb53gBT+mXMNx2tsg07Kmuz7vfjR5ydoY7guyB3X1vUK9y AmCWlGq67eRG934SujZFikO/oZUrwRrQu2jj5v8B7xwtcCFCdpZAIRabD4BTglvPiQEiBBAB AgAMBQJKjl+9BQMAEnUAAAoJEJcQuJvKV618DTwH/3DzIl1zwr6TTtTfTBH9FSDdhvaUEPKC bLT3WZWzIHREaLEEncQ85cGoYoBeJXVBIwBczZUpGy4pqFjYcWQ9vKFm2Nt1Nrs+v9tKc+9G ECH0Y1a+9GDYqnepcN2O/3HLASCEpXFwQhVe01G+lupGgqYfMgTG9RByTkMzVXB9ER5gijGC zjTf1YAOFUx2eBBLYa3w/ZZpT+nwRmEUaDpfwq06UPrzMZuho17SGPZUNz41z4p2NF8Td9bk hOiJ3+gORRohbq0HdaRdvSDoP/aGsQltfeF5p0KEcpIHx5B05H1twIkOGFTxyx3nTWqauEJy 2a+W15ZB10hB2TqwAE9Z54KJASIEEAECAAwFAkqgEkcFAwASdQAACgkQlxC4m8pXrXwyXwf/ UPzz+D+n19JWivha7laUxuDzMQCKTcEjFCu4QVZ1rqcBFPoz0Tt74/X75QdmxZizqX1E61bF  $\verb|EsbVjL2Mt5zZjedS1vbSbrmn4hV4pHZr08dbf1ZkNX105g8ZlpsqQ7VyUt5YtWCn0tGNn4B5| \\$ Eb6WMeqxQteujV3B7AtMH+CD0ja+A2/p0rHIpqScz8aupksBMCrYqhoT+7/qXNEVkjNmcu2N mHxfv6dL5Xy/0iJjie2umStu8WTfRTpYmnv2gEhbCdb/zhFvG61GgTBJqv9MvBVGRxnJFd41 NqlucsadD+UM7WjV3v5VuN2r9KD9wocd/s22ELCRA2wKccvR/nWBkIkBIgQQAQIADAUCSqgQ AAUDABJ1AAAKCRCXELibyletfAT8B/9cPhH8DlHoiv+cK8rAJMomZqVqOyy4BwsRrakycVlg 7/yvMs74anynSoUf0LgsXADQ29Hmrpf+zC5E5/jPGWNK81x2VBVoB8nZkMSAnkZfOw+mWu9I Aj2NLcsvt9JYNmAq5R7RrirHsDQ2DIYxRgaE/5CVEVry9YQEj18A13/SYyoB4FWpDI4fRfUW JbUJrYmfg0p+4zL0YS9F11UhsHUu+g1W1c83N54ozI1v013HUwVayzII4E/YNrIkpOaO+o8R  $\verb"MAXw4DsJZcpViQEiBBABAgAMBQJKuceJBQMAEnUAAAoJEJcQuJvKV6188KEH/24QK2LV1142"$ 4Wx3T9G4bJFRWWuuEkTpYJw6ss72lqus9t7BsoGaNLMHQzKAlca9wLTqY826q4nv9anEqwWZ +Di8kE+UAMUq2BFTL0EvOMJ6i1ZyE8cUFVb1+09tpBWJJS7t3z00uMMMznGuHzSm4MgCnGhA  $\verb|sOgiuHdPWSlnHnqNJa/SB6UVQxtcDOaqQlLIvhd2HVqrOBRtER3td/YgLO6HSxXpXtz8DBa2| \\$ NYQYSwAdlqJAPLBnBsLXwbCswuIDMZZv8BJwUNBEJkokOMv5CXxhPrP5kxWvyBvs1hTk8ph2 GIh/ZRVNDAsChbuU1EJBACpwaMrcgwjPtI7/KTgeZVSJASIEEAECAAwFAkreCMYFAwASdQAA CgkQlxC4m8pXrXyOQQf7BvRm/3PvFCCksyjBW4EVBW7z/Ps/kBK6bIE9Q7f7QlXFIcGGUIpA rufXWbV+G4a3Z8LFeFJTovNePfquwpFjneUZn1CG+oVS1AfddvYhAsgkLhQqMbaNJIJ1y4D/  ${\tt H3xvCna/s7Teufud0JLXoLBedFXeB5Cg2K1EoxINqMo+lm/VGJmbykwqoRvxZLDfnbFag5zG}$ 59+OWw4TC8nzlIQYIBn22YiWRk5zsCJA40O+KL1vwBiFDrREhALQc/YBJKYrRX3ZV4U/EeYD KB0NCBk1W1tXGCee3uhM0S5VFc1j7Pg58ECuntH5xOy+KMNF1jiQwvWfbaFTJvCjFQS+Op1X b4kBIgQQAQIADAUCSu86VAUDABJ1AAAKCRCXELibyletfGs8CACteI2BmKs24GF80JeWTOQI cvHnCdV7hKZOltbNPBbDv6qTt3iX2GVa10iYhI5Eq30jt/hKFJTMlfYZyI1peFodGjv7Lk51 u7zaNBvT1pBCP+eJspi6rGpSuhtMSb405jPc1RBmbY+w9wctLyZf1zG+s1Sdw8adcRXQNFqr vVIZYOmu2S8FunqLfxpjewiFiDPzAzmbWzMoO2PLCYFhwV6Eh2jO33OGbvBmyHNFZBfX5F/+ kiyeT47MEhrfhytJ6ZOdpxtX8HvbvzPZcDLOI80W6rPTG76KW06ZiZrJ81YCa6a7D01y7BYy W2HoxzYcuumjRkGF4nqK4Mw+wefCp0H/iQEiBBABAgAMBQJLAF3aBQMAEnUAAAoJEJcQuJvK V618/q0H/ibXDQG2WQmC1LoT4H+ezXjPqDq8aiuz6f4xibTmrO+L4ScMX+zK0KZVwp6Kau28 Nx+g00oAUW8mNxhd+c10ZaY+7RIkxEvkooKKsArBmZT+xrE6CgHlAs3D4Mc+14nfD0aZaUbE

 $\verb|iobWvX1YL127MELLcWyeMlgbeNoucc473JddvmHSRRM5F9Qp28CvWDEXYqhq11aoaho8+cei|\\$ pvzyuO3OTwjuAOqhefOHzAvFrRli99ML8xzF1ZOvBct+36SuYxDXyIhkSd7aG9Us01W6W5Si JYt4cDyI0JDhbhZN0tzWYKcKMZMxf8w3jW4sfOL0prhHrARqqPiU8OTUH/VNX5CJASIEEAEC AAwFAksRgasFAwASdQAACgkQlxC4m8pXrXydogf/a3lofmYFMoE3p9SqGt/v28iyO0j9A1Lm  $\verb|qKwEhJkxff/X/Qa7pafGQ9J90JQkxYKMxydWPspTbDFMccZWkBK132vZp9Q3FHKpnDPDLK2S||$ 25miTReeAAQNqMMFLeyy7ZHi5YsKwLbKxcSo7/m0jlitNYlmt94imFNpq/mHGsy6O+rLeQTA opuIzP3VwN6ItL5gIFxqWPmf/V0xh/vxTwLqJ66vECD8vyHrHblUzgiXHgyYbZPxAa2SRRd3 4V38phaZ/QsTkss+Sd/QeHChWyU9d6KengWwcr/nDO+K/hhmnO5Oqz02Upwyxrgi6484HQUN /Smf44VBsSD1DBjaAKjMr4kBIgQQAQIADAUCSyNN1AUDABJ1AAAKCRCXELibyletfCWiB/9c EZtdFVcsxpE3hJzM6PBPf+1QKuJORve/7MqNEb3TMWFgBxyOfvD7uMpCJyOrqq5AbUQfZfj9 K7qmzWUMuoYceGIlbdmHFBJwtmaF0BiyHaobqY/9RbdCNcbtzrW34feiW9aDZyvCoLHEVkCC QACSv3FwdYVkkRB5eihvpwJk5tpScdIA12YLqzmVTFdhrZuYvtDdQHjgoLMO8B9s9kok7D2T SpveVzXXPH68Z3JkVubhHT7cs+n+9PRvcaVJtsX2VTUY5eFVqmGuAUVrvp2aN8cKQ+mVcCQr VVIhT908YB5925MUx2VJml0y0nkBQuMZyzMEOVGkuU/G+pVrRmmAiQEiBBABAqAMBQJLJyaS BQMAEnUAAAoJEJcQuJvKV618eU0IAKnVh6ymId9C3ZqVyxwTnOB8RMQceJzwCLqk2RT0dPhN 5ZwUcQN71Cp9hymMutC8FdKRK/ESK21vJF2/576Pln4fIeOIbycBAEvqrL14epATj53uBizo NOTuwb1kximFERuW3MP4XiFUJB0tPws2vR5UU3t6GoQJJwNoIbz9DK2L6X/Qz3Tb9if6bPSK U6JR1Yn3Hos9ogg21vWCxgMTKUuPAYhmYjSvkqH3BihXi+c17MVvE7W5GJbQHuJo+MgSxu04 4qnvDHZpf4Mzc30XcGlohjxefNyeiY2bzdI2yCaCtmW0lCW1Sc2oiE0zw06lD4hY5XmC2Xql MLsKB5VNXJGJASIEEAECAAwFAks4Ze4FAwASdQAACgkQlxC4m8pXrXyWXggAon2abiNvRzx9 7364Mjx4IlFvM1tVebzNbOkDwZS1ABqTDGgq/ffZA/VZrU+h2eL97cQyGxJEQ5kkm/v1iobE ZEFMT0pv9WMzfidqzhdKdcpbbxdaErIjD5fBACKdjazAUeH7zce2v+bBN019LZoRiXbNugG9 381kJ2E4ZTYYfvftL/e4RzOgqR9VD/A5MzxfXFbCVharHbeT8OwZy4Oz2UDaDszHsNKoG1WN pOSf2HTMBPNcsOSY/hIBRWNxnzdY0kWt7laeLNmN1eUEwzk4J7GnlambPIctOdoEUriMSaey TkLZGejKnwi/PqARyDW1FsReKNHD753ZMViUnAsq2IkBIqQQAQIADAUCS0oyJwUDABJ1AAAK CRCXELibyletfGodCAC5hjmxwquHSb8ZL0RifIL3j3iU6U7qLK1TQKkTqqELfUzeF9f8NuNR txLmzNk1T7YI9iji6NAtnuy43v610MbqlkV8x69qNP36Owv408wXxEt0s5ViZuVOZJAY075c YRhopgfmhkh4hbkAoKCLajOROWUEEsDHsqqj8XLJuGRREURy8TJWaB/cotXsgiJf99gt+gIw In8tyb3+WVIUHWfw2+Drpd3nfcMqgeO54PePJo0BWWjaar+wgC/76Se286IHcYMrml/Adnvx ZaIKmxZmkTmDMCfMnVjRYSKBGjQ9Uu7dws7SMsbbd34f8Jt9nyuRqMc14INAXthWY/S3Sdil iQEiBBABAgAMBQJLW/5mBQMAEnUAAAoJEJcQuJvKV6181L8IAKq3ZOQHzqaOoz5wnvj51YG8 nZoW5RG7HOb3mL1D9b+FTTzaIxsLf7STagPwKtM57rU/7ehHIuO/9QQNQ3Mudw17ZiwD015X 7iG8/AflWnc6bXfTz18IplRuqyVc0qQeJZhT7MBpklcS4ZGZHPQdtAh4Aw5YXihrbbq6jV7j CzUmFz4XcT8CkJHIUGoFR0vTmFqlAt2K1imwGMh2IEamPOJ0wsTbBfZbhmkB03RToEjIipGZ M+NtKS/NL2RJYWZ+FCCcEMoRgmlVmATWw3natgLWwN4Z6K4rGXONWi/OwyFgxZpmjdHmjcXa Iqz8EroVsLbnaV/8yG7cqK5e6M0Fk1iJASIEEAECAAwFAkttIfqFAwASdQAACqkQ1xC4m8pX rXyR3QgAksvAMfqC+ACUEWSVAlepDFR1xI45UwBa2UeBY7KjOOCiZlkGREvx20IOv1gExyPl zNxDeqmYsl2mleEoH6QlXaJRd8MxIVfAnjAt8izwU2dfDwflTTWgGQYf8q7qeAv1XC34yNge 0JaTD1C55QpmcO51f2ojMsAi36bBJO4Dr59jhVYiDjQADS/d7FpAznlhH9SGUq6ekYb2jxCS rvt0wRtMyk6YGgts4xEHcN0wC9VTobaXo9xvsqhtUK44Gdvptq1cBFX8byzD6fN8nXp+v8qh tlPYDqb4muqTh2UXXiWMtvPXo7kkZQ8CvI3YbZ10F1IDLt20VJWFZaJYL2fzyokCIqQQAQIA DAUCQYHLhQWDBiLZBwAKCRCq4+bOZqFEaKgvEACCErnaHGyUYa0wETjj6DLEXsqe0iXad4i9 aBQxnD35GUgcFofC/nCY4XcnCMMEnmdQ9ofUuU3OBJ6BNJIbEusAabgLooebP/3KEaiCIiyh HYU5jarpZAh+Zopgs30c11m01tJaS69iJxrGTLodkAsAJAeEUwTPg9fHFFzC1eGBysovFWg4 bljz/zClI+qyTbFA5g6tRoiXTo8ko7QhY2AA5UGEg+83Hdb6akC04Z2QRErxKAqrphHzj8Xp jVOsQAdAi/qVKQeNKROlJ+iq6+YesmcWGfzeb87dGNweVFDJIGA0qY27pTb21ExYjsRFN4Cb 13NfodAbMTOxcAWZ7jAPCxAP1HUG++mHMrhQXEToZnBFE4nbnC7vOBNgWdjUgXcpkUCkop4b 17BFpR+k8ZtYLSS8p2LLz4uAeCcSm2/msJxT7rC/FvoH8428oHincqs2ICo9zO/Ud4Hmm000 +SsZdVKIIjinGyOVWb400zkAlnnhEZ3o6hAHcREIsBgPwEYVTj/9ZdC0AO44Nj9cU7awaqgt rnwwfr/o4V2q18bLSkltZU27/29HeuOeFGj1FeOYrDd/aRNsxbyb2O28H4sG1CVZmC5uK1iQ BDiSyA7Q0bbdofCWoQzm5twlpKWnY8Oe0ub9XP5p/sVfck4FceWFHwv+/PC9RzS1331Q6vM2 wIkCIgQTAQIADAUCQp8KHAWDBQWacAAKCRDYwgoJWiRXzyE+D/9uc7z6fIsalfOYoLN60ajA bQbI/uRKBFugyZ5RoaItusn9Z2rAtn61WrFhu4uCSJtFN1ny2RERg40f56pTghKrD+YEt+Nz e6+FKQ5AbGIdFsR/2bUk+ZZRSt83e14Lcb6ii/fJfzkoIox9ltkifQxqY7Tvk4noKu4oLSc8 OlWsfc/y0B9sYUUCmUfcnq58DEmGie9ovUslmyt5NPnveXxp5UeaRc5Rqt9tK2B4A+7/cqEN rdZJbAMSunt2+2fkYiRunAFPKPBdJBsY1sxeL/A9aKeOviKEXQdAWqdNZKNCi8rd/oOP99/9 lMbFudAbX6nL2DSb10G2Z7NWEqgIAzjmpwYYPCKeVz5Q8R+if9/fe5+STY/550aI33fJ2H3v +U435VjYqbrerWe36xJItcJeqUzW71fQtXi1CTEl3w2ch7VF5oj/QyjabLnAlHgSlkSi6p7B y5C2MnbCHlCfPnIinPhFoRcRGPjJe9nFwGs+QblvS/Chzc2WX3s/2SWm4gEUKRX4zsAJ5ocy fa/vkxCkSxK/erWlCPf/J1T70+i5waXDN/E3enSet/WL7h94pQKpjz8OdGL4JSBHuAVGA+a+ dknqnPF0KMKLhjrqV+L7084FhbmAP7PXm3xmiMPriXf+el5fZZequQoIaqf8rdRHHhRJxQqI 0HNknkaOqs8dtrkCDQQ+PqMdEAgA7+GJfxbMdY4ws1PnjH9rF4N2qfWsEN/1xaZoJYc3a6M0 2WCnH16ahT2/tBK2w1QI4YFteR47gCvtgb601JHff0o2HfLmRDRiRjd1DTCHqeyX7CHhcghj /dNR1W2Z015QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE7zaD5cH9J7yv/6xuZVw411x0h4Uq sTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fmLe11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRs KTfozBu74F47D8Ilbf5vSYHbuE5p/loIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnl LzKUb/F5GwADBQf+Lwqqa8CGrRfsOAJxim63CHfty5mUc5rUSnTs1GYEIOCR1BeQauyPZbPD

SDD9MZ1ZaSafanFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGo
TbOWI39Ecyzgs1zBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42LmuQT5N
dKHGwHsXTPt10klk4bQk40ajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt926s/ymfdf5Hkd
Qlcyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZWhe70YGNPwlyjWJT11hM
BBgRAgAMBQI+PqMdBQkJZgGAAAOJEIxxjTtQcuH17p4An3r1QpVC9yhnW2cSAjq+kr72GX0e
AJ4295kl6NxYEuFApmr1+0uUq/SlsYhMBBgRAgAMBQJHrJT8BQkNMFjfAAOJEIxxjTtQcuH1
pc4An0I965H3JY2GTrizp+dCezxbhexaAJ48FhocFYvfhZtgeUWb6aPvgQZHT4hUBBgRAgAM
BQI+PqMdBQkJZgGAABIJEIxxjTtQcuH1B2VHUEcAAQHungCfevVClUL3KGdbZxICOr6SvvYZ
fR4Anjb3mSXo3FgS4UCmavX7S5Sr9KWxiFQEGBECAAwFAk53Pe0FCRP7AbgAEgdlR1BHAAEB
CRCMcY07UHLh9RSbAJsFivb5sESf8vYE5yfDln9AVa6FEwCgpWAIWbl9plDcB+L5RCUBw6mG
uck=
=yia9
----END PGP PUBLIC KEY BLOCK-----

### 2.5 Installation Layouts

The installation layout differs for different installation types (for example, native packages, binary tarballs, and source tarballs), which can lead to confusion when managing different systems or using different installation sources. The individual layouts are given in the corresponding installation type or platform chapter, as described following. Note that the layout of installations from vendors other than Oracle may differ from these layouts.

- Section 5.1, "MySQL Installation Layout on Microsoft Windows"
- Section 4.3, "MySQL Layout for Source Installation"
- Table 3.1, "MySQL Installation Layout for Generic Unix/Linux Binary Package"
- Table 7.5, "MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone"
- Table 6.1, "MySQL Installation Layout on macOS"

### 2.6 Compiler-Specific Build Characteristics

In some cases, the compiler used to build MySQL affects the features available for use. The notes in this section apply for binary distributions provided by Oracle Corporation or that you compile yourself from source.

### icc (Intel C++ Compiler) Builds

A server built with icc has these characteristics:

SSL support is not included.

# Chapter 3 Installing MySQL on Unix/Linux Using Generic Binaries

Oracle provides a set of binary distributions of MySQL. These include generic binary distributions in the form of compressed tar files (files with a .tar.xz extension) for a number of platforms, and binaries in platform-specific package formats for selected platforms.

This section covers the installation of MySQL from a compressed tar file binary distribution on Unix/Linux platforms. For Linux-generic binary distribution installation instructions with a focus on MySQL security features, refer to the Secure Deployment Guide. For other platform-specific binary package formats, see the other platform-specific sections in this manual. For example, for Windows distributions, see Chapter 5, Installing MySQL on Microsoft Windows. See Section 2.3, "How to Get MySQL" on how to obtain MySQL in different distribution formats.

MySQL compressed tar file binary distributions have names of the form mysql-VERSION-OS.tar.xz, where VERSION is a number (for example, 8.0.42), and OS indicates the type of operating system for which the distribution is intended (for example, pc-linux-i686 or winx64).

There is also a "minimal install" version of the MySQL compressed tar file for the Linux generic binary distribution, which has a name of the form mysql-VERSION-OS-GLIBCVER-ARCH-minimal.tar.xz. The minimal install distribution excludes debug binaries and is stripped of debug symbols, making it significantly smaller than the regular binary distribution. If you choose to install the minimal install distribution, remember to adjust for the difference in file name format in the instructions that follow.

#### Warnings

• If you have previously installed MySQL using your operating system native package management system, such as Yum or APT, you may experience problems installing using a native binary. Make sure your previous MySQL installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check for configuration files such as /etc/myscnf or the /etc/myscnl directory and delete them.

For information about replacing third-party packages with official MySQL packages, see the related APT guide or Yum guide.

MySQL has a dependency on the libaio library. Data directory initialization
and subsequent server startup steps fail if this library is not installed locally. If
necessary, install it using the appropriate package manager. For example, on
Yum-based systems:

```
$> yum search libaio # search for info
$> yum install libaio # install library
```

Or, on APT-based systems:

```
$> apt-cache search libaio # search for info
$> apt-get install libaio1 # install library
```

• Oracle Linux 8 / Red Hat 8 (EL8): These platforms by default do not install the file /lib64/libtinfo.so.5, which is required by the MySQL client bin/mysql for packages mysql-VERSION-el7-x86\_64.tar.gz and mysql-VERSION-linux-glibc2.12-x86\_64.tar.xz. To work around this issue, install the ncurses-compat-libs package:

\$> yum install ncurses-compat-libs

 If no RPM or .deb file specific to your distribution is provided by Oracle (or by your Linux vendor), you can try the generic binaries. In some cases, due to library incompatibilities or other issues, these may not work with your Linux installation. In such cases, you can try to compile and install MySQL from source. See Chapter 4, Installing MySQL from Source, for more information and instructions

To install a compressed tar file binary distribution, unpack it at the installation location you choose (typically /usr/local/mysql). This creates the directories shown in the following table.

Table 3.1 MySQL Installation Layout for Generic Unix/Linux Binary Package

Directory	Contents of Directory
bin	mysqld server, client and utility programs
docs	MySQL manual in Info format
man	Unix manual pages
include	Include (header) files
lib	Libraries
share	Error messages, dictionary, and SQL for database installation
support-files	Miscellaneous support files

Debug versions of the mysqld binary are available as mysqld-debug. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. See Chapter 4, *Installing MySQL from Source*.

To install and use a MySQL binary distribution, the command sequence looks like this:

```
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
$> cd /usr/local
$> tar xvf /path/to/mysql-VERSION-OS.tar.xz
$> ln -s full-path-to-mysql-VERSION-OS mysql
$> cd mysql
$> mkdir mysql-files
$> chown mysql:mysql mysql-files
$> chown for mysql-files
$> bin/mysqld --initialize --user=mysql
$> bin/mysql_ssl_rsa_setup
$> bin/mysql_safe --user=mysql &
# Next command is optional
$> cp support-files/mysql.server /etc/init.d/mysql.server
```

#### Note

This procedure assumes that you have root (administrator) access to your system. Alternatively, you can prefix each command using the sudo (Linux) or pfexec (Solaris) command.

The mysql-files directory provides a convenient location to use as the value for the secure\_file\_priv system variable, which limits import and export operations to a specific directory. See Server System Variables.

A more detailed version of the preceding description for installing a binary distribution follows.

### Create a mysql User and Group

If your system does not already have a user and group to use for running <code>mysqld</code>, you may need to create them. The following commands add the <code>mysql</code> group and the <code>mysql</code> user. You might want to call the user and group something else instead of <code>mysql</code>. If so, substitute the appropriate name in the following instructions. The syntax for <code>useradd</code> and <code>groupadd</code> may differ slightly on different versions of <code>Unix/Linux</code>, or they may have different names such as <code>adduser</code> and <code>addgroup</code>.

```
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
```

#### Note

Because the user is required only for ownership purposes, not login purposes, the useradd command uses the -r and -s /bin/false options to create a user that does not have login permissions to your server host. Omit these options if your useradd does not support them.

### **Obtain and Unpack the Distribution**

Pick the directory under which you want to unpack the distribution and change location into it. The example here unpacks the distribution under /usr/local. The instructions, therefore, assume that you have permission to create files and directories in /usr/local. If that directory is protected, you must perform the installation as root.

```
$> cd /usr/local
```

Obtain a distribution file using the instructions in Section 2.3, "How to Get MySQL". For a given release, binary distributions for all platforms are built from the same MySQL source distribution.

Unpack the distribution, which creates the installation directory. tar can uncompress and unpack the distribution if it has z option support:

```
$> tar xvf /path/to/mysql-VERSION-OS.tar.xz
```

The tar command creates a directory named mysql-VERSION-OS.

To install MySQL from a compressed tar file binary distribution, your system must have GNU xz Utils to uncompress the distribution and a reasonable tar to unpack it.

#### Note

The compression algorithm changed from Gzip to XZ in MySQL Server 8.0.12; and the generic binary's file extension changed from .tar.gz to .tar.xz.

GNU tar is known to work. The standard tar provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU tar, or if available, use a preinstalled version of GNU tar. Usually this is available as <code>gnutar</code>, <code>gtar</code>, or as tar within a GNU or Free Software directory, such as <code>/usr/sfw/bin</code> or <code>/usr/local/bin</code>. GNU tar is available from <a href="http://www.gnu.org/software/tar/">http://www.gnu.org/software/tar/</a>.

If your tar does not support the xz format then use the xz command to unpack the distribution and tar to unpack it. Replace the preceding tar command with the following alternative command to uncompress and extract the distribution:

```
$> xz -dc /path/to/mysql-VERSION-OS.tar.xz | tar x
```

Next, create a symbolic link to the installation directory created by tar:

\$> ln -s full-path-to-mysql-VERSION-OS mysql

\$> export PATH=\$PATH:/usr/local/mysql/bin

### **Perform Postinstallation Setup**

The remainder of the installation process involves setting distribution ownership and access permissions, initializing the data directory, starting the MySQL server, and setting up the configuration file. For instructions, see Chapter 9, *Postinstallation Setup and Testing*.

# Chapter 4 Installing MySQL from Source

### **Table of Contents**

4.1 Source Installation Metho	ods	27
4.2 Source Installation Prere-	quisites	28
4.3 MySQL Layout for Sourc	e Installation	29
4.4 Installing MySQL Using a	Standard Source Distribution	30
4.5 Installing MySQL Using a	Development Source Tree	34
4.6 Configuring SSL Library	Support	36
4.7 MySQL Source-Configura	ation Options	
	ompiling MySQL	
4.9 MySQL Configuration and	d Third-Party Tools	72
4.10 Generating MySQL Dox	ygen Documentation Content	72
,	•	

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 <a href="https://www.mysql.com/support/supportedplatforms/database.html">https://www.mysql.com/support/supportedplatforms/database.html</a>.

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 Chapter 3, *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 Section 4.10, "Generating MySQL Doxygen Documentation Content".

### 4.1 Source Installation Methods

There are two methods for installing MySQL from source:

 Use a standard MySQL source distribution. To obtain a standard distribution, see Section 2.3, "How to Get MySQL". For instructions on building from a standard distribution, see Section 4.4, "Installing MySQL Using a Standard Source Distribution".

Standard distributions are available as compressed tar files, Zip archives, or RPM packages. Distribution files have names of the form mysql-VERSION.tar.gz, mysql-VERSION.zip, or mysql-VERSION.rpm, where VERSION is a number like 8.0.42. File names for source distributions can be distinguished from those for precompiled binary distributions in that source distribution names are generic and include no platform name, whereas binary distribution names include a platform name indicating the type of system for which the distribution is intended (for example, pc-linux-i686 or winx64).

• Use a MySQL development tree. For information on building from one of the development trees, see Section 4.5, "Installing MySQL Using a Development Source Tree".

### 4.2 Source Installation Prerequisites

Installation of MySQL from source requires several development tools. Some of these tools are needed no matter whether you use a standard source distribution or a development source tree. Other tool requirements depend on which installation method you use.

To install MySQL from source, the following system requirements must be satisfied, regardless of installation method:

- CMake, which is used as the build framework on all platforms. CMake can be downloaded from http://www.cmake.org.
- A good make program. Although some platforms come with their own make implementations, it is highly
  recommended that you use GNU make 3.75 or later. It may already be available on your system as
  gmake. GNU make is available from http://www.gnu.org/software/make/.

On Unix-like systems, including Linux, you can check your system's version of make like this:

```
$> make --version
GNU Make 4.2.1
```

- As of MySQL 8.0.26, MySQL 8.0 source code permits use of C++17 features. To enable the necessary level of C++17 support across all supported platforms, the following minimum compiler versions apply:
  - Linux: GCC 10 or Clang 5
  - macOS: XCode 10
  - Solaris: (MySQL 8.0.40 and earlier) GCC 10; (MySQL 8.0.41 and later) GCC 11.4
  - · Windows: Visual Studio 2019
- Building MySQL on Windows requires Windows version 10 or later. (MySQL binaries built on recent versions of Windows can generally be run on older versions.) You can determine the Windows version by executing WMIC.exe os get version in the Windows Command Prompt.
- The MySQL C API requires a C++ or C99 compiler to compile.
- An SSL library is required for support of encrypted connections, entropy for random number generation, and other encryption-related operations. By default, the build uses the OpenSSL library installed on the host system. To specify the library explicitly, use the WITH\_SSL option when you invoke CMake. For additional information, see Section 4.6, "Configuring SSL Library Support".
- The Boost C++ libraries are required to build MySQL (but not to use it). MySQL compilation requires a particular Boost version. Typically, that is the current Boost version, but if a specific MySQL source distribution requires a different version, the configuration process stops with a message indicating the Boost version that it requires. To obtain Boost and its installation instructions, visit the official Boost web site. After Boost is installed, tell the build system where the Boost files are placed according to the value set for the WITH\_BOOST option when you invoke CMake. For example:

```
cmake . -DWITH_BOOST=/usr/local/boost_version_number
```

Adjust the path as necessary to match your installation.

• The ncurses library.

- Sufficient free memory. If you encounter build errors such as internal compiler error when
  compiling large source files, it may be that you have too little memory. If compiling on a virtual machine,
  try increasing the memory allocation.
- Perl is needed if you intend to run test scripts. Most Unix-like systems include Perl. For Windows, you can use ActiveState Perl. or Strawberry Perl.

To install MySQL from a standard source distribution, one of the following tools is required to unpack the distribution file:

• For a .tar.gz compressed tar file: GNU gunzip to uncompress the distribution and a reasonable tar to unpack it. If your tar program supports the z option, it can both uncompress and unpack the file.

GNU tar is known to work. The standard tar provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU tar, or if available, use a preinstalled version of GNU tar. Usually this is available as gnutar, gtar, or as tar within a GNU or Free Software directory, such as /usr/sfw/bin or /usr/local/bin. GNU tar is available from https://www.gnu.org/software/tar/.

- For a .zip Zip archive: WinZip or another tool that can read .zip files.
- For an .rpm RPM package: The rpmbuild program used to build the distribution unpacks it.

To install MySQL from a development source tree, the following additional tools are required:

- The Git revision control system is required to obtain the development source code. GitHub Help provides
  instructions for downloading and installing Git on different platforms.
- bison 2.1 or later, available from http://www.gnu.org/software/bison/. (Version 1 is no longer supported.) Use the latest version of bison where possible; if you experience problems, upgrade to a later version, rather than revert to an earlier one.

bison is available from http://www.gnu.org/software/bison/. bison for Windows can be downloaded from http://gnuwin32.sourceforge.net/packages/bison.htm. Download the package labeled "Complete package, excluding sources". On Windows, the default location for bison is the C:\Program Files \GnuWin32 directory. Some utilities may fail to find bison because of the space in the directory name. Also, Visual Studio may simply hang if there are spaces in the path. You can resolve these problems by installing into a directory that does not contain a space (for example C:\GnuWin32).

On Solaris Express, m4 must be installed in addition to bison. m4 is available from http://www.gnu.org/software/m4/.

#### Note

If you have to install any programs, modify your PATH environment variable to include any directories in which the programs are located. See Setting Environment Variables.

If you run into problems and need to file a bug report, please use the instructions in How to Report Bugs or Problems.

### 4.3 MySQL Layout for Source Installation

By default, when you install MySQL after compiling it from source, the installation step installs files under / usr/local/mysql. The component locations under the installation directory are the same as for binary distributions. See Table 3.1, "MySQL Installation Layout for Generic Unix/Linux Binary Package", and

Section 5.1, "MySQL Installation Layout on Microsoft Windows". To configure installation locations different from the defaults, use the options described at Section 4.7, "MySQL Source-Configuration Options".

### 4.4 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 Section 4.2, "Source Installation Prerequisites".
- 2. Obtain a distribution file using the instructions in Section 2.3, "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 Chapter 9, *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 Section 7.4, "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 Chapter 3, *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 Chapter 9, *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 Section 2.3, "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 -</pre>
```

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 Section 4.7, "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 Section 4.8, "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 Chapter 3, *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 Section 5.4, "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 Chapter 9, *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 Chapter 9, *Postinstallation Setup and Testing*.

# 4.5 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 Section 4.2, "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.
```

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
                                         packaging
cmake
                   include
                                                             strings
components libbinlogevents README config.h.cmake libchangestreams router configure.cmake libmysql run doxyge
                                                             support-files
                                                            testclients
                                                              unittest
                                          run_doxygen.cmake utilities
                  libmysql
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 Section 4.4, "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.

# 4.6 Configuring SSL Library Support

An SSL library is required for support of encrypted connections, entropy for random number generation, and other encryption-related operations.

If you compile MySQL from a source distribution, CMake configures the distribution to use the installed OpenSSL library by default.

To compile using OpenSSL, use this procedure:

- 1. Ensure that OpenSSL 1.0.1 or newer is installed on your system. If the installed OpenSSL version is older than 1.0.1, CMake produces an error at MySQL configuration time. If it is necessary to obtain OpenSSL, visit http://www.openssl.org.
- 2. The WITH\_SSL CMake option determines which SSL library to use for compiling MySQL (see Section 4.7, "MySQL Source-Configuration Options"). The default is -DWITH\_SSL=system, which uses OpenSSL. To make this explicit, specify that option. For example:

```
cmake . -DWITH_SSL=system
```

That command configures the distribution to use the installed OpenSSL library. Alternatively, to explicitly specify the path name to the OpenSSL installation, use the following syntax. This can be useful if you have multiple versions of OpenSSL installed, to prevent CMake from choosing the wrong one:

```
cmake . -DWITH_SSL=path_name
```

Alternative OpenSSL system packages are supported as of MySQL 8.0.30 by using WITH\_SSL=openss111 on EL7 or WITH\_SSL=openss13 on EL8. Authentication plugins, such as LDAP and Kerberos, are disabled since they do not support these alternative versions of OpenSSL.

3. Compile and install the distribution.

To check whether a mysqld server supports encrypted connections, examine the value of the have\_ssl system variable:

If the value is YES, the server supports encrypted connections. If the value is DISABLED, the server is capable of supporting encrypted connections but was not started with the appropriate --ssl-xxx options to enable encrypted connections to be used; see Configuring MySQL to Use Encrypted Connections.

# 4.7 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 Chapter 12, 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_	Bundle runtime libraries with server MSI and Zip packages for Windows	OFF		
CMAKE_BUILD_TYP	Type of build to produce	RelWithDebInfo		

Formats	Description	Default	Introduced	Removed
CMAKE_CXX_FLAGS	Flags for C++ Compiler			
CMAKE_C_FLAGS	Flags for C Compiler			
CMAKE_INSTALL_P	Installation base directory	/usr/local/ mysql		
COMPILATION_COM	©mment about compilation environment			
COMPILATION_COM	Comment about compilation environment for use by mysqld		8.0.14	
COMPRESS_DEBUG_	©ompress 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_COLLATION	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_MEM	Exclude Performance Schema memory instrumentation	OFF		

Formats	Description	Default	Introduced	Removed
DISABLE_PSI_MET.	Exclude Performance Schema metadata instrumentation	OFF		
DISABLE_PSI_MUT	Exclude Performance Schema mutex instrumentation	OFF		
DISABLE_PSI_PS	Exclude the performance schema prepared statements	OFF		
DISABLE_PSI_RWL	Exclude Performance Schema rwlock instrumentation	OFF		
DISABLE_PSI_SOC	Exclude Performance Schema socket instrumentation	OFF		
DISABLE_PSI_SP	Exclude Performance Schema stored program instrumentation	OFF		
DISABLE_PSI_STA	Exclude Performance Schema stage instrumentation	OFF		
DISABLE_PSI_STA	Exclude Performance Schema statement instrumentation	OFF		
DISABLE_PSI_STA	Performance Schema statements_digest instrumentation	OFF		
DISABLE_PSI_TAB	Exclude Performance Schema table instrumentation	OFF		
DISABLE_PSI_THR	Exclude the performance schema thread instrumentation	OFF		
DISABLE_PSI_TRA	Exclude the performance	OFF		

Formats	Description	Default	Introduced	Removed
	schema transaction instrumentation			
DISABLE_SHARED	Do not build shared libraries, compile position-dependent code	OFF		8.0.18
DOWNLOAD_BOOST	Whether to download the Boost library	OFF		
DOWNLOAD_BOOST_	Timeout in seconds for downloading the Boost library	600		
ENABLED_LOCAL_I	Whether to enable LOCAL for LOAD DATA	OFF		
ENABLED_PROFILI	Whether to enable query profiling code	ON		
ENABLE_DOWNLOAD	Whether to download optional files	OFF		8.0.26
ENABLE_EXPERIME	Whiether 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_O	Whether to colorize compiler output	OFF	8.0.33	
FORCE_INSOURCE_	Whether to force an in-source build	OFF	8.0.14	
FORCE_UNSUPPORT	Whether to permit unsupported compilers	OFF		
FPROFILE_GENERA	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	Enable performance schema memory tracing module for	OFF	8.0.26	

Formats	Description	Default	Introduced	Removed
	memory allocation functions used in dynamic storage of over-aligned types			
IGNORE_AIO_CHEC	With - DBUILD_CONFIG=n ignore libaio check	OFF nysql_release,		
INSTALL_BINDIR	User executables directory	PREFIX/bin		
INSTALL_DOCDIR	Documentation directory	PREFIX/docs		
INSTALL_DOCREAD	README file directory	PREFIX		
INSTALL_INCLUDE	⊞eader 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_MYSQLKE	Directory:for keyring_file plugin data file	platform specific		
INSTALL_MYSQLSH.	Shared data directory	PREFIX/share		
INSTALL_MYSQLTE	mysql-test directory	PREFIX/mysql- test		
INSTALL_PKGCONF	Directory for mysqlclient.pc pkg-config file	INSTALL_LIBDIR/ pkgconfig		
INSTALL_PLUGIND	Rlugin directory	PREFIX/lib/ plugin		
INSTALL_PRIV_LI	Installation private library directory		8.0.18	
INSTALL_SBINDIR	Server executable directory	PREFIX/bin		
INSTALL_SECURE_	secu <u>re</u> rfile <u>p</u> riiv default value	platform specific		
INSTALL_SHAREDI	aclocal/mysql.m4 installation directory	PREFIX/share		
INSTALL_STATIC_	Whether to install static libraries	ON		
INSTALL_SUPPORT	Extrasupport files directory	PREFIX/support-files		
	-			

Formats	Description	Default	Introduced	Removed
LINK_RANDOMIZE	Whether to randomize order of symbols in mysqld binary	OFF		
LINK_RANDOMIZE_	Seed value for LINK_RANDOMIZE option	mysql		
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_ADD	Unix socket file used by X Plugin	/tmp/ mysqlx.sock		
MYSQL_DATADIR	Data directory			
MYSQL_MAINTAINE	Whether to enable MySQL maintainer-specific development environment	OFF		
MYSQL_PROJECT_N	AWindows/macOS project name	MySQL		
MYSQL_TCP_PORT	TCP/IP port number	3306		
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock		
NDB_UTILS_LINK_	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_SANITI	Whether to optimize sanitizer builds	ON	8.0.34	
REPRODUCIBLE_BU	Take extra care to create a build result independent of build location and time			

Formats	Description	Default	Introduced	Removed
SHOW_SUPPRESSED	Whether to show IN suppressed compiler warnings and not fail with - Werror.	<b>O</b> FF	8.0.30	
SYSCONFDIR	Option file directory			
SYSTEMD_PID_DIR	Directory for PID file under systemd	/var/run/mysqld		
SYSTEMD_SERVICE	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_IN	Whether to disable function inlining	OFF		
WITHOUT_SERVER	Do not build the server; internal use only	OFF		
WITHOUT_xxx_STO	Excluderstorage 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_AUTHENTICA	Enabled IENT_PLUG automatically if any corresponding server authentication plugins are built	INS	8.0.26	
WITH_AUTHENTICA	Whether to report error if LDAP authentication plugins cannot be built	OFF		

Formats	Description	Default	Introduced	Removed
WITH_AUTHENTICA	Boild_PAM authentication plugin	OFF		
WITH_AWS_SDK	Location of Amazon Web Services software development kit			
WITH_BOOST	The location of the Boost library sources			
WITH_BUILD_ID	On Linux systems, generate a unique build ID	ON	8.0.31	
WITH_BUNDLED_LI	libevent when building ndbmemcache; obsolete	ON		8.0.23
WITH_BUNDLED_ME	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_PRO	Build client side protocol tracing framework	ON		
WITH_CURL	Location of curl library			
WITH_DEBUG	Whether to include debugging support	OFF		
WITH_DEFAULT_CO	Whether to usens default compiler options	ON		
WITH_DEFAULT_FE	Whetherno use default feature set	ON		8.0.22
WITH_DEVELOPER_	Whethento add the 'get-task-allow' entitlement to all executables on macOS to generate a core dump in	OFF	8.0.30	

Formats	Description	Default	Introduced	Removed
	the event of an unexpected server halt			
WITH_EDITLINE	Which libedit/ editline library to use	bundled		
WITH_ERROR_INSE	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF		
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_EXT	Whether to include extra debugging support for InnoDB.	OFF		
WITH_INNODB_MEMO	Whether to generate memcached shared libraries.	OFF		
WITH_JEMALLOC	Whether to link with -ljemalloc	OFF	8.0.16	
WITH_KEYRING_TE	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		

Formats	Description	Default	Introduced	Removed
WITH_LZMA	Type of LZMA library support	bundled		8.0.16
WITH_MECAB	Compiles MeCab			
WITH_MSAN	Enable MemorySanitizer	OFF		
WITH_MSCRT_DEBU	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 engine and all NDB programs	OFF	8.0.31	
WITH_NDBAPI_EXA	BuildsAPI 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	BriortocNDBNGINE 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	[none]		

Formats	Description	Default	Introduced	Removed
	option was not used to build it, the management server's default port is 1186.			
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_NUMA	Set NUMA memory allocation policy			
WITH_PACKAGE_FL	used for RPM/DEB packages, whether to add them to standalone builds on those platforms		8.0.26	
WITH_PLUGIN_NDB	may not work as expected in all circumstances. Instead, users should employ WITH_NDBCLUSTE	R	8.0.13	8.0.31
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_DE	Enable additional systemd debug information	OFF	8.0.22	
WITH_SYSTEM_LIB	Set system value of library options not set explicitly	OFF		

Formats	Description	Default	Introduced	Removed
WITH_TCMALLOC	Whether to link with -ltcmalloc. BUNDLED is supported on Linux only	OFF	8.0.22	
WITH_TEST_TRACE	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		
WITH_WIN_JEMALL	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_STORAG	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 <code>mysql-test</code> directory. To suppress installation of this directory, explicitly set the option to the empty value (<code>-DINSTALL\_MYSQLTESTDIR=</code>).

• -DINSTALL\_PKGCONFIGDIR=dir\_name

The directory in which to install the <code>mysqlclient.pc</code> file for use by <code>pkg-config</code>. The default value is <code>INSTALL\_LIBDIR/pkgconfig</code>, unless <code>INSTALL\_LIBDIR</code> ends with <code>/mysql</code>, 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 <code>secure\_file\_priv</code> system variable. The default value is platform specific and depends on the value of the <code>INSTALL\_LAYOUT</code> CMake option; see the description of the <code>secure\_file\_priv</code> 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 Innode, 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_STAGE
DISABLE_PSI_STATEMENT
DISABLE_PSI_STATEMENT
DISABLE_PSI_STATEMENT_DIGEST
DISABLE_PSI_TABLE
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. 11d linker is used, and is disabled by It has no effect if a linker other than 11d 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 goov 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 <code>ENABLED\_LOCAL\_INFILE</code> disabled. If you compile MySQL from source, configure it with <code>ENABLED\_LOCAL\_INFILE</code> disabled or enabled based on whether clients that make no explicit arrangements should have <code>LOCAL</code> 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 guery profiling code (for the SHOW PROFILE and SHOW PROFILES statements).

• -DFORCE\_UNSUPPORTED\_COMPILER=boo1

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 <code>cmake/fprofile.cmake</code> file in a MySQL source distribution for information about using <code>FPROFILE\_GENERATE</code> and <code>FPROFILE\_USE</code>. 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.

• -DMYSOL 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=boo1

CMake causes the build process to link using the LLVM 11d 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=boo1 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 <code>cmake/build\_configurations/feature\_set.cmake</code>. 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 <code>-DENABLE\_DOWNLOADS=1</code>, 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=1z4\_type

The WITH\_LZ4 option indicates the source of zlib support:

- bundled: Use the 1z4 library bundled with the distribution. This is the default.
- system: Use the system 1z4 library. If WITH\_LZ4 is set to this value, the 1z4\_decompress utility is not built. In this case, the system 1z4 command can be used instead.
- -DWITH\_LZMA=1zma\_type

The type of LZMA library support to include. 1zma\_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 <code>-DWITH\_MECAB=system</code>. The <code>system</code> 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, <code>-DWITH\_MECAB=/opt/mecab</code>. If the <code>system</code> 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. rapid json\_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 <code>path\_name</code> points to a manually installed OpenSSL library. This is because they do not have system SSL libraries. On macOS, <code>brew install openssl</code> installs to <code>/usr/local/opt/openssl</code> so that system can find it. On Windows, it checks <code>%ProgramFiles%/OpenSSL</code>, <code>%ProgramFiles%/OpenSSL-Win32</code>, <code>%ProgramFiles%/OpenSSL-Win32</code>, and <code>C:/OpenSSL-Win64</code>.

- 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 Section 4.6, "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 Section 7.9, "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\_LZMA, 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 <code>jemalloc.dll</code> to enable jemalloc functionality. The build system copies <code>jemalloc.dll</code> to the same directory as <code>mysqld.exe</code> and/or <code>mysqld-debug.exe</code> and utilizes it for memory management operations. Standard memory functions are used if <code>jemalloc.dll</code> 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 -01 -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 (<code>-Onumber</code>), you must set the <code>CMAKE\_C\_FLAGS\_build\_type</code> and/or <code>CMAKE\_CXX\_FLAGS\_build\_type</code> options, where <code>build\_type</code> corresponds to the <code>CMAKE\_BUILD\_TYPE</code> value. To specify a different optimization for the default build type (<code>RelWithDebInfo</code>) set the <code>CMAKE\_C\_FLAGS\_RELWITHDEBINFO</code> and <code>CMAKE\_CXX\_FLAGS\_RELWITHDEBINFO</code> options. For example, to compile on Linux with <code>-O3</code> and with debug symbols, do this:

```
$> cmake .. -DCMAKE_C_FLAGS_RELWITHDEBINFO="-03 -g" \
```

-DCMAKE\_CXX\_FLAGS\_RELWITHDEBINFO="-03 -g"

# **CMake Options for Compiling NDB Cluster**

To compile with support for NDB Cluster, you can use <code>-DWITH\_NDB</code>, 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 <code>-DWITH\_NDBCLUSTER\_STORAGE\_ENGINE=OFF</code>. 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 <code>-DWITH\_NDB\_JAVA=OFF</code> when running <code>CMake</code>. 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.)

# 4.8 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 recreate 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

For information about acquiring or updating tools, see the system requirements in Chapter 4, *Installing MySQL from Source*.

# 4.9 MySQL Configuration and Third-Party Tools

Third-party tools that need to determine the MySQL version from the MySQL source can read the MYSQL\_VERSION file in the top-level source directory. The file lists the pieces of the version separately. For example, if the version is MySQL 8.0.36, the file looks like this:

```
MYSQL_VERSION_MAJOR=8
MYSQL_VERSION_MINOR=0
MYSQL_VERSION_PATCH=36
MYSQL_VERSION_EXTRA=
MYSQL_VERSION_STABILITY="LTS"
```

#### Note

In MySQL 5.7 and earlier, this file was named VERSION.

To construct a five-digit number from the version components, use this formula:

MYSQL\_VERSION\_MAJOR\*10000 + MYSQL\_VERSION\_MINOR\*100 + MYSQL\_VERSION\_PATCH

# 4.10 Generating MySQL Doxygen Documentation Content

The MySQL source code contains internal documentation written using Doxygen. The generated Doxygen content is available at <a href="https://dev.mysql.com/doc/index-other.html">https://dev.mysql.com/doc/index-other.html</a>. It is also possible to generate this content locally from a MySQL source distribution using the following procedure:

1. Install doxygen 1.9.2 or later. Distributions are available here at http://www.doxygen.nl/.

After installing doxygen, verify the version number:

```
$> doxygen --version
1.9.2
```

2. Install PlantUML.

When you install PlantUML on Windows (tested on Windows 10), you must run it at least once as administrator so it creates the registry keys. Open an administrator console and run this command:

```
$> java -jar path-to-plantuml.jar
```

The command should open a GUI window and return no errors on the console.

3. Set the PLANTUML\_JAR\_PATH environment to the location where you installed PlantUML. For example:

```
$> export PLANTUML_JAR_PATH=path-to-plantuml.jar
```

4. Install the Graphviz dot command.

After installing Graphviz, verify dot availability. For example:

```
$> which dot
/usr/bin/dot
$> dot -V
dot - graphviz version 2.40.1 (20161225.0304)
```

5. Change location to the top-level directory of your MySQL source distribution and do the following:

First, execute cmake:

```
$> cd mysql-source-directory
$> mkdir build
$> cd build
$> cmake ..
```

Next, generate the doxygen documentation:

```
$> make doxygen
```

Inspect the error log, which is available in the doxyerror.log file in the top-level directory. Assuming that the build executed successfully, view the generated output using a browser. For example:

```
$> firefox doxygen/html/index.html
```

# Chapter 5 Installing MySQL on Microsoft Windows

## **Table of Contents**

5.1 MySQL Installation Layout on Microsoft Windows	78
5.2 Choosing an Installation Package	78
5.3 MySQL Installer for Windows	80
5.3.1 MySQL Installer Initial Setup	81
5.3.2 Setting Alternative Server Paths with MySQL Installer	
5.3.3 Installation Workflows with MySQL Installer	
5.3.4 MySQL Installer Product Catalog and Dashboard	94
5.3.5 MySQL Installer Console Reference	. 100
5.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive	. 110
5.4.1 Extracting the Install Archive	
5.4.2 Creating an Option File	. 110
5.4.3 Selecting a MySQL Server Type	111
5.4.4 Initializing the Data Directory	
5.4.5 Starting the Server for the First Time	. 112
5.4.6 Starting MySQL from the Windows Command Line	
5.4.7 Customizing the PATH for MySQL Tools	
5.4.8 Starting MySQL as a Windows Service	. 114
5.4.9 Testing The MySQL Installation	. 117
5.5 Troubleshooting a Microsoft Windows MySQL Server Installation	
5.6 Windows Postinstallation Procedures	
5.7 Windows Platform Restrictions	121

#### **Important**

MySQL 8.0 Server requires the Microsoft Visual C++ 2019 Redistributable Package to run on Windows platforms. Users should make sure the package has been installed on the system before installing the server. The package is available at the Microsoft Download Center. Additionally, MySQL debug binaries require Visual Studio 2019 to be installed.

MySQL is available for Microsoft Windows 64-bit operating systems only. For supported Windows platform information, see https://www.mysql.com/support/supportedplatforms/database.html.

There are different methods to install MySQL on Microsoft Windows.

# **MySQL Installer Method**

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure a specific version of MySQL Server as follows:

1. Download MySQL Installer from https://dev.mysql.com/downloads/installer/ and execute it.

#### Note

Unlike the standard MySQL Installer, the smaller web-community version does not bundle any MySQL applications, but downloads only the MySQL products you choose to install.

2. Determine the setup type to use for the initial installation of MySQL products. For example:

- **Developer Default**: Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench.
- **Server Only**: Provides a setup for the selected version of MySQL Server without other products.
- Custom: Enables you to select any version of MySQL Server and other MySQL products.
- 3. Install the server instance (and products) and then begin the server configuration by following the onscreen instructions. For more information about each individual step, see Section 5.3.3.1, "MySQL Server Configuration with MySQL Installer".

MySQL is now installed. If you configured MySQL as a service, then Windows automatically starts the MySQL server every time you restart the system. Also, this process installs the MySQL Installer application on the local host, which you can use later to upgrade or reconfigure MySQL server.

#### Note

If you installed MySQL Workbench on your system, consider using it to check your new MySQL server connection. By default, the program automatically start after installing MySQL.

## **Additional Installation Information**

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see Section 5.4.8, "Starting MySQL as a Windows Service".

To accommodate the RESTART statement, the MySQL server forks when run as a service or standalone, to enable a monitor process to supervise the server process. In this case, there are two mysqld processes. If RESTART capability is not required, the server can be started with the --no-monitor option. See RESTART Statement.

Generally, you should install MySQL on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the PATH environment variable or accessing the Service Control Manager. When installed, MySQL does not need to be executed using a user with Administrator privileges.

For a list of limitations on the use of MySQL on the Windows platform, see Section 5.7, "Windows Platform Restrictions".

In addition to the MySQL Server package, you may need or want additional components to use MySQL with your application or development environment. These include, but are not limited to:

To connect to the MySQL server using ODBC, you must have a Connector/ODBC driver. For more
information, including installation and configuration instructions, see MySQL Connector/ODBC
Developer Guide.

#### Note

MySQL Installer installs and configures Connector/ODBC for you.

To use MySQL server with .NET applications, you must have the Connector/NET driver. For more
information, including installation and configuration instructions, see MySQL Connector/NET Developer
Guide.

#### Note

MySQL Installer installs and configures MySQL Connector/NET for you.

MySQL distributions for Windows can be downloaded from https://dev.mysql.com/downloads/. See Section 2.3, "How to Get MySQL".

MySQL for Windows is available in several distribution formats, detailed here. Generally speaking, you should use MySQL Installer. It contains more features and MySQL products than the older MSI, is simpler to use than the compressed file, and you need no additional tools to get MySQL up and running. MySQL Installer automatically installs MySQL Server and additional MySQL products, creates an options file, starts the server, and enables you to create default user accounts. For more information on choosing a package, see Section 5.2, "Choosing an Installation Package".

 A MySQL Installer distribution includes MySQL Server and additional MySQL products including MySQL Workbench, and MySQL for Visual Studio. MySQL Installer can also be used to upgrade these products in the future (see https://dev.mysql.com/doc/mysql-compat-matrix/en/).

For instructions on installing MySQL using MySQL Installer, see Section 5.3, "MySQL Installer for Windows".

The standard binary distribution (packaged as a compressed file) contains all of the necessary files that
you unpack into your chosen location. This package contains all of the files in the full Windows MSI
Installer package, but does not include an installation program.

For instructions on installing MySQL using the compressed file, see Section 5.4, "Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive".

• The source distribution format contains all the code and support files for building the executables using the Visual Studio compiler system.

For instructions on building MySQL from source on Windows, see Chapter 4, *Installing MySQL from Source*.

# **MySQL** on Windows Considerations

#### Large Table Support

If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use MAX\_ROWS and AVG\_ROW\_LENGTH when you create tables. See CREATE TABLE Statement.

#### MySQL and Virus Checking Software

Virus-scanning software such as Norton/Symantec Anti-Virus on directories containing MySQL data and temporary tables can cause issues, both in terms of the performance of MySQL and the virus-scanning software misidentifying the contents of the files as containing spam. This is due to the fingerprinting mechanism used by the virus-scanning software, and the way in which MySQL rapidly updates different files, which may be identified as a potential security risk.

After installing MySQL Server, it is recommended that you disable virus scanning on the main directory (datadir) used to store your MySQL table data. There is usually a system built into the virus-scanning software to enable specific directories to be ignored.

In addition, by default, MySQL creates temporary files in the standard Windows temporary directory. To prevent the temporary files also being scanned, configure a separate temporary directory for

MySQL temporary files and add this directory to the virus scanning exclusion list. To do this, add a configuration option for the tmpdir parameter to your my.ini configuration file. For more information, see Section 5.4.2, "Creating an Option File".

# 5.1 MySQL Installation Layout on Microsoft Windows

For MySQL 8.0 on Windows, the default installation directory is C:\Program Files\MySQL\MySQL Server 8.0 for installations performed with MySQL Installer. If you use the ZIP archive method to install MySQL, you may prefer to install in C:\mysql. However, the layout of the subdirectories remains the same.

All of the files are located within this parent directory, using the structure shown in the following table.

Table 5.1 Default MySQL Installation Layout for Microsoft Windows

Directory	Contents of Directory	Notes
bin	mysqld server, client and utility programs	
%PROGRAMDATA%\MySQL\MySQL Server 8.0\	Log files, databases	The Windows system variable %PROGRAMDATA% defaults to C: \ProgramData.
docs	Release documentation	With MySQL Installer, use the Modify operation to select this optional folder.
include	Include (header) files	
lib	Libraries	
share	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

### **Silent Installation Methods**

Use MySQL Installer, see Section 5.3.5, "MySQL Installer Console Reference".

# 5.2 Choosing an Installation Package

For MySQL 8.0, there are multiple installation package formats to choose from when installing MySQL on Windows. The package formats described in this section are:

- MySQL Installer
- MySQL noinstall ZIP Archives
- MySQL Docker Images

Program Database (PDB) files (with file name extension pdb) provide information for debugging your MySQL installation in the event of a problem. These files are included in ZIP Archive distributions (but not MSI distributions) of MySQL.

# **MySQL** Installer

This package has a file name similar to mysql-installer-community-8.0.42.0.msi or mysql-installer-commercial-8.0.42.0.msi, and utilizes MSIs to install MySQL server and other products automatically. MySQL Installer downloads and applies updates to itself, and to each of the installed products. It also configures the installed MySQL server (including a sandbox InnoDB cluster test setup) and MySQL Router. MySQL Installer is recommended for most users.

MySQL Installer can install and manage (add, modify, upgrade, and remove) many other MySQL products, including:

- Applications MySQL Workbench, MySQL for Visual Studio, MySQL Shell, and MySQL Router (see https://dev.mysql.com/doc/mysql-compat-matrix/en/)
- Connectors MySQL Connector/C++, MySQL Connector/NET, Connector/ODBC, MySQL Connector/Python, MySQL Connector/J, MySQL Connector/Node.js
- Documentation MySQL Manual (PDF format), samples and examples

MySQL Installer operates on all MySQL supported versions of Windows (see https://www.mysql.com/support/supportedplatforms/database.html).

#### Note

Because MySQL Installer is not a native component of Microsoft Windows and depends on .NET, it does not work with minimal installation options like the Server Core version of Windows Server.

For instructions on how to install MySQL using MySQL Installer, see Section 5.3, "MySQL Installer for Windows".

# **MySQL noinstall ZIP Archives**

These packages contain the files found in the complete MySQL Server installation package, with the exception of the GUI. This format does not include an automated installer, and must be manually installed and configured.

The noinstall ZIP archives are split into two separate compressed files. The main package is named mysql-VERSION-winx64.zip. This contains the components needed to use MySQL on your system. The optional MySQL test suite, MySQL benchmark suite, and debugging binaries/information components (including PDB files) are in a separate compressed file named mysql-VERSION-winx64-debugtest.zip.

If you choose to install a noinstall ZIP archive, see Section 5.4, "Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive".

# **MySQL Docker Images**

For information on using the MySQL Docker images provided by Oracle on Windows platform, see Section 7.6.3, "Deploying MySQL on Windows and Other Non-Linux Platforms with Docker".

### Warning

The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk.

# 5.3 MySQL Installer for Windows

MySQL Installer is a standalone application designed to ease the complexity of installing and configuring MySQL products that run on Microsoft Windows. It is downloaded with and supports the following MySQL products:

MySQL Servers

MySQL Installer can install and manage multiple, separate MySQL server instances on the same host at the same time. For example, MySQL Installer can install, configure, and upgrade separate instances of MySQL 5.7 and MySQL 8.0 on the same host. MySQL Installer does not permit server upgrades between major and minor version numbers, but does permit upgrades within a release series (such as 8.0.36 to 8.0.37).

#### Note

MySQL Installer cannot install both *Community* and *Commercial* releases of MySQL server on the same host. If you require both releases on the same host, consider using the ZIP archive distribution to install one of the releases.

MySQL Applications

MySQL Workbench, MySQL Shell, and MySQL Router.

MySQL Connectors

These are not supported, instead install from <a href="https://dev.mysql.com/downloads/">https://dev.mysql.com/downloads/</a>. These connectors include MySQL Connector/NET, MySQL Connector/Python, MySQL Connector/ODBC, MySQL Connector/J, MySQL Connector/Node.js, and MySQL Connector/C++.

#### Note

The connectors were bundled before MySQL Installer 1.6.7 (MySQL Server 8.0.34), and MySQL Installer could install each connector up to version 8.0.33 until MySQL Installer 1.6.11 (MySQL Server 8.0.37). MySQL Installer now only detects these old connector versions to uninstall them.

# **Installation Requirements**

MySQL Installer requires Microsoft .NET Framework 4.5.2 or later. If this version is not installed on the host computer, you can download it by visiting the Microsoft website.

To invoke MySQL Installer after a successful installation:

- 1. Right-click Windows Start, select **Run**, and then click **Browse**. Navigate to Program Files (x86) > MySQL > MySQL Installer for Windows to open the program folder.
- 2. Select one of the following files:
  - MySQLInstaller.exe to open the graphical application.
  - MySQLInstallerConsole.exe to open the command-line application.
- 3. Click **Open** and then click **OK** in the Run window. If you are prompted to allow the application to make changes to the device, select Yes.

Each time you invoke MySQL Installer, the initialization process looks for the presence of an internet connection and prompts you to enable offline mode if it finds no internet access (and offline mode is

disabled). Select Yes to run MySQL Installer without internet-connection capabilities. MySQL product availability is limited to only those products currently in the product cache when you enable offline mode. To download MySQL products, click the offline mode **Disable** quick action shown on the dashboard.

An internet connection is required to download a manifest containing metadata for the latest MySQL products that are not part of a full bundle. MySQL Installer attempts to download the manifest when you start the application for the first time and then periodically in configurable intervals (see MySQL Installer options). Alternatively, you can retrieve an updated manifest manually by clicking **Catalog** in the MySQL Installer dashboard.

#### Note

If the first-time or subsequent manifest download is unsuccessful, an error is logged and you may have limited access to MySQL products during your session. MySQL Installer attempts to download the manifest with each startup until the initial manifest structure is updated. For help finding a product, see Locating Products to Install.

# **MySQL Installer Community Release**

Download software from https://dev.mysql.com/downloads/installer/ to install the Community release of all MySQL products for Windows. Select one of the following MySQL Installer package options:

- Web: Contains MySQL Installer and configuration files only. The web package option downloads only the MySQL products you select to install, but it requires an internet connection for each download. The size of this file is approximately 2 MB. The file name has the form mysql-installer-community-web-VERSION.N.msi in which VERSION is the MySQL server version number such as 8.0 and N is the package number, which begins at 0.
- Full or Current Bundle: Bundles all of the MySQL products for Windows (including the MySQL server). The file size is over 300 MB, and the name has the form mysql-installer-community-VERSION.N.msi in which VERSION is the MySQL Server version number such as 8.0 and N is the package number, which begins at 0.

# **MySQL Installer Commercial Release**

Download software from <a href="https://edelivery.oracle.com/">https://edelivery.oracle.com/</a> to install the Commercial release (Standard or Enterprise Edition) of MySQL products for Windows. If you are logged in to your My Oracle Support (MOS) account, the Commercial release includes all of the current and previous GA versions available in the Community release, but it excludes development-milestone versions. When you are not logged in, you see only the list of bundled products that you downloaded already.

The Commercial release also includes the following products:

- Workbench SE/EE
- MySQL Enterprise Backup
- MySQL Enterprise Firewall

The Commercial release integrates with your MOS account. For knowledge-base content and patches, see My Oracle Support.

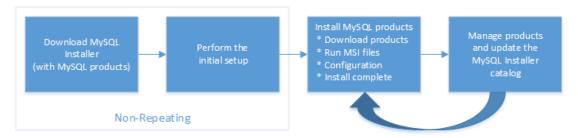
# 5.3.1 MySQL Installer Initial Setup

Choosing a Setup Type

- · Path Conflicts
- Check Requirements
- MySQL Installer Configuration Files

When you download MySQL Installer for the first time, a setup wizard guides you through the initial installation of MySQL products. As the following figure shows, the initial setup is a one-time activity in the overall process. MySQL Installer detects existing MySQL products installed on the host during its initial setup and adds them to the list of products to be managed.

Figure 5.1 MySQL Installer Process Overview



MySQL Installer extracts configuration files (described later) to the hard drive of the host during the initial setup. Although MySQL Installer is a 32-bit application, it can install both 32-bit and 64-bit binaries.

The initial setup adds a link to the Start menu under the **MySQL** folder group. Click **Start**, **MySQL**, and **MySQL Installer - [Community | Commercial]** to open the community or commercial release of the graphical tool.

## **Choosing a Setup Type**

During the initial setup, you are prompted to select the MySQL products to be installed on the host. One alternative is to use a predetermined setup type that matches your setup requirements. By default, both GA and pre-release products are included in the download and installation with the **Client only** and **Full** setup types. Select the **Only install GA products** option to restrict the product set to include GA products only when using these setup types.

#### Note

Commercial-only MySQL products, such as MySQL Enterprise Backup, are available to select and install if you are using the Commercial version of MySQL Installer (see MySQL Installer Commercial Release).

Choosing one of the following setup types determines the initial installation only and does not limit your ability to install or update MySQL products for Windows later:

- **Server only**: Only install the MySQL server. This setup type installs the general availability (GA) or development release server that you selected when you downloaded MySQL Installer. It uses the default installation and data paths.
- Client only: Only install the most recent MySQL applications (such as MySQL Shell, MySQL Router, and MySQL Workbench). This setup type excludes MySQL server or the client programs typically bundled with the server, such as mysql or mysqladmin.
- Full: Install all available MySQL products, excluding MySQL connectors.

 Custom: The custom setup type enables you to filter and select individual MySQL products from the MySQL Installer catalog.

Use the Custom setup type to install:

- A product or product version that is not available from the usual download locations. The catalog
  contains all product releases, including the other releases between pre-release (or development) and
  GA.
- An instance of MySQL server using an alternative installation path, data path, or both. For instructions
  on how to adjust the paths, see Section 5.3.2, "Setting Alternative Server Paths with MySQL Installer".
- Two or more MySQL server versions on the same host at the same time (for example, 5.7 and 8.0).
- A specific combination of products and features not offered as a predetermine setup type. For example, you can install a single product, such as MySQL Workbench, instead of installing all client applications for Windows.

#### **Path Conflicts**

When the default installation or data folder (required by MySQL server) for a product to be installed already exists on the host, the wizard displays the **Path Conflict** step to identify each conflict and enable you to take action to avoid having files in the existing folder overwritten by the new installation. You see this step in the initial setup only when MySQL Installer detects a conflict.

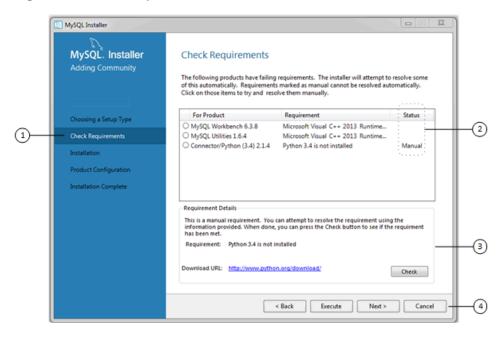
To resolve the path conflict, do one of the following:

- Select a product from the list to display the conflict options. A warning symbol indicates which path is in conflict. Use the browse button to choose a new path and then click **Next**.
- Click Back to choose a different setup type or product version, if applicable. The Custom setup type enables you to select individual product versions.
- Click **Next** to ignore the conflict and overwrite files in the existing folder.
- Delete the existing product. Click Cancel to stop the initial setup and close MySQL Installer. Open MySQL Installer again from the Start menu and delete the installed product from the host using the Delete operation from the MySQL Installer dashboard.

### **Check Requirements**

MySQL Installer uses entries in the package-rules.xml file to determine whether the prerequisite software for each product is installed on the host. When the requirements check fails, MySQL Installer displays the **Check Requirements** step to help you update the host. Requirements are evaluated each time you download a new product (or version) for installation. The following figure identifies and describes the key areas of this step.

Figure 5.2 Check Requirements



### **Description of Check Requirements Elements**

- 1. Shows the current step in the initial setup. Steps in this list may change slightly depending on the products already installed on the host, the availability of prerequisite software, and the products to be installed on the host.
- 2. Lists all pending installation requirements by product and indicates the status as follows:
  - A blank space in the Status column means that MySQL Installer can attempt to download and install
    the required software for you.
  - The word *Manual* in the **Status** column means that you must satisfy the requirement manually. Select each product in the list to see its requirement details.
- Describes the requirement in detail to assist you with each manual resolution. When possible, a
  download URL is provided. After you download and install the required software, click Check to verify
  that the requirement has been met.
- 4. Provides the following set operations to proceed:
  - Back Return to the previous step. This action enables you to select a different the setup type.
  - Execute Have MySQL Installer attempt to download and install the required software for all items without a manual status. Manual requirements are resolved by you and verified by clicking Check.
  - Next Do not execute the request to apply the requirements automatically and proceed to the
    installation without including the products that fail the check requirements step.
  - Cancel Stop the installation of MySQL products. Because MySQL Installer is already installed, the
    initial setup begins again when you open MySQL Installer from the Start menu and click Add from
    the dashboard. For a description of the available management operations, see Product Catalog.

## **MySQL Installer Configuration Files**

All MySQL Installer files are located within the C:\Program Files (x86) and C:\ProgramData folders. The following table describes the files and folders that define MySQL Installer as a standalone application.

#### Note

Installed MySQL products are neither altered nor removed when you update or uninstall MySQL Installer.

Table 5.2 MySQL Installer Configuration Files

File or Folder	Description	Folder Hierarchy
MySQL Installer for Windows	This folder contains all of the files needed to run MySQL Installer and MySQLInstallerConsole.exe, a command-line program with similar functionality.	C:\Program Files (x86)
Templates	The Templates folder has one file for each version of MySQL server. Template files contain keys and formulas to calculate some values dynamically.	C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest
package-rules.xml	This file contains the prerequisites for every product to be installed.	C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest
products.xml	The products file (or product catalog) contains a list of all products available for download.	C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest
Product Cache	The Product Cache folder contains all standalone .msi files bundled with the full package or downloaded afterward.	C:\ProgramData\MySQL \MySQL Installer for Windows

# 5.3.2 Setting Alternative Server Paths with MySQL Installer

You can change the default installation path, the data path, or both when you install MySQL server. After you have installed the server, the paths cannot be altered without removing and reinstalling the server instance.

#### Note

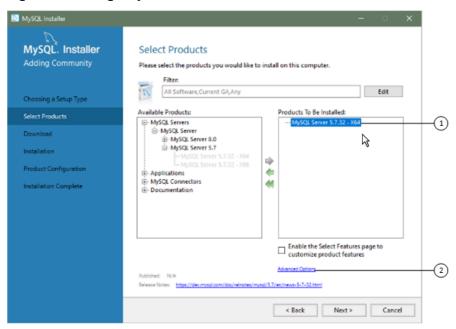
Starting with MySQL Installer 1.4.39, if you move the data directory of an installed server manually, MySQL Installer identifies the change and can process a reconfiguration operation without errors.

### To change paths for MySQL server

- 1. Identify the MySQL server to change and enable the Advanced Options link as follows:
  - a. Navigate to the **Select Products** page by doing one of the following:
    - i. If this is an initial setup of MySQL Installer, select the Custom setup type and click **Next**.

- If MySQL Installer is installed on your computer, click Add from the dashboard.
- b. Click **Edit** to apply a filter on the product list shown in **Available Products** (see Locating Products to Install).
- With the server instance selected, use the arrow to move the selected server to the **Products To** Be Installed list.
- d. Click the server to select it. When you select the server, the **Advanced Options** link is enabled below the list of products to be installed (see the following figure).
- 2. Click **Advanced Options** to open a dialog box where you can enter alternative path names. After the path names are validated, click **Next** to continue with the configuration steps.

Figure 5.3 Change MySQL Server Path



# 5.3.3 Installation Workflows with MySQL Installer

MySQL Installer provides a wizard-like tool to install and configure new MySQL products for Windows. Unlike the initial setup, which runs only once, MySQL Installer invokes the wizard each time you download or install a new product. For first-time installations, the steps of the initial setup proceed directly into the steps of the installation. For assistance with product selection, see Locating Products to Install.

#### Note

Full permissions are granted to the user executing MySQL Installer to all generated files, such as my.ini. This does not apply to files and directories for specific products, such as the MySQL server data directory in <code>%ProgramData%</code> that is owned by <code>SYSTEM</code>.

Products installed and configured on a host follow a general pattern that might require your input during the various steps. If you attempt to install a product that is incompatible with the existing MySQL server version (or a version selected for upgrade), you are alerted about the possible mismatch.

MySQL Installer provides the following sequence of actions that apply to different workflows:

• Select Products. If you selected the Custom setup type during the initial setup or clicked Add from the MySQL Installer dashboard, MySQL Installer includes this action in the sidebar. From this page, you can apply a filter to modify the Available Products list and then select one or more products to move (using arrow keys) to the Products To Be Installed list.

Select the check box on this page to activate the Select Features action where you can customize the products features after the product is downloaded.

• **Download.** If you installed the full (not web) MySQL Installer package, all .msi files were loaded to the Product Cache folder during the initial setup and are not downloaded again. Otherwise, click **Execute** to begin the download. The status of each product changes from Ready to Download, to Downloading, and then to Downloaded.

To retry a single unsuccessful download, click the **Try Again** link.

To retry all unsuccessful downloads, click Try All.

• Select Features To Install (disabled by default). After MySQL Installer downloads a product's .msi file, you can customize the features if you enabled the optional check box previously during the Select Products action.

To customize product features after the installation, click Modify in the MySQL Installer dashboard.

• Installation. The status of each product in the list changes from Ready to Install, to Installing, and lastly to Complete. During the process, click **Show Details** to view the installation actions.

If you cancel the installation at this point, the products are installed, but the server (if installed) is not yet configured. To restart the server configuration, open MySQL Installer from the Start menu and click **Reconfigure** next to the appropriate server in the dashboard.

Product configuration. This step applies to MySQL Server, MySQL Router, and samples only.
The status for each item in the list should indicate Ready to Configure. Click Next to start the configuration wizard for all items in the list. The configuration options presented during this step are specific to the version of database or router that you selected to install.

Click **Execute** to begin applying the configuration options or click **Back** (repeatedly) to return to each configuration page.

• Installation complete. This step finalizes the installation for products that do not require configuration. It enables you to copy the log to a clipboard and to start certain applications, such as MySQL Workbench and MySQL Shell. Click Finish to open the MySQL Installer dashboard.

## 5.3.3.1 MySQL Server Configuration with MySQL Installer

MySQL Installer performs the initial configuration of the MySQL server. For example:

- It creates the configuration file (my.ini) that is used to configure the MySQL server. The values written to this file are influenced by choices you make during the installation process. Some definitions are host dependent.
- By default, a Windows service for the MySQL server is added.
- Provides default installation and data paths for MySQL server. For instructions on how to change the default paths, see Section 5.3.2, "Setting Alternative Server Paths with MySQL Installer".

• It can optionally create MySQL server user accounts with configurable permissions based on general roles, such as DB Administrator, DB Designer, and Backup Admin. It optionally creates a Windows user named Mysqlsys with limited privileges, which would then run the MySQL Server.

User accounts may also be added and configured in MySQL Workbench.

• Checking **Show Advanced Options** enables additional **Logging Options** to be set. This includes defining custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log.

During the configuration process, click **Next** to proceed to the next step or **Back** to return to the previous step. Click **Execute** at the final step to apply the server configuration.

The sections that follow describe the server configuration options that apply to MySQL server on Windows. The server version you installed will determine which steps and options you can configure. Configuring MySQL server may include some or all of the steps.

## **Type and Networking**

Server Configuration Type

Choose the MySQL server configuration type that describes your setup. This setting defines the amount of system resources (memory) to assign to your MySQL server instance.

- **Development**: A computer that hosts many other applications, and typically this is your personal workstation. This setting configures MySQL to use the least amount of memory.
- **Server**: Several other applications are expected to run on this computer, such as a web server. The Server setting configures MySQL to use a medium amount of memory.
- **Dedicated**: A computer that is dedicated to running the MySQL server. Because no other major applications run on this server, this setting configures MySQL to use the majority of available memory.

#### Manual

Prevents MySQL Installer from attempting to optimize the server installation, and instead, sets the default values to the server variables included in the my.ini configuration file. With the Manual type selected, MySQL Installer uses the default value of 16M for the tmp\_table\_size variable assignment.

Connectivity

Connectivity options control how the connection to MySQL is made. Options include:

- TCP/IP: This option is selected by default. You may disable TCP/IP Networking to permit local host connections only. With the TCP/IP connection option selected, you can modify the following items:
  - Port for classic MySQL protocol connections. The default value is 3306.
  - X Protocol Port shown when configuring MySQL 8.0 server only. The default value is 33060
  - Open Windows Firewall port for network access, which is selected by default for TCP/IP connections.

If a port number is in use already, you will see the information icon ( ^\_ ) next to the default value and **Next** is disabled until you provide a new port number.

• Named Pipe: Enable and define the pipe name, similar to setting the named\_pipe system variable. The default name is MySQL.

When you select **Named Pipe** connectivity, and then proceed to the next step, you are prompted to set the level of access control granted to client software on named-pipe connections. Some clients require only minimum access control for communication, while other clients require full access to the named pipe.

You can set the level of access control based on the Windows user (or users) running the client as follows:

- Minimum access to all users (RECOMMENDED). This level is enabled by default because it is the most secure.
- Full access to members of a local group. If the minimum-access option is too restrictive for the client software, use this option to reduce the number of users who have full access on the named pipe. The group must be established on Windows before you can select it from the list. Membership in this group should be limited and managed. Windows requires a newly added member to first log out and then log in again to join a local group.
- Full access to all users (NOT RECOMMENDED). This option is less secure and should be set only when other safeguards are implemented.
- **Shared Memory**: Enable and define the memory name, similar to setting the shared\_memory system variable. The default name is MySOL.
- Advanced Configuration

Check **Show Advanced and Logging Options** to set custom logging and advanced options in later steps. The Logging Options step enables you to define custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log. The Advanced Options step enables you to set the unique server ID required when binary logging is enabled in a replication topology.

• MySQL Enterprise Firewall (Enterprise Edition only)

The **Enable MySQL Enterprise Firewall** check box is deselected by default. Select this option to enable a security list that offers protection against certain types of attacks. Additional post-installation configuration is required (see MySQL Enterprise Firewall).

### **Authentication Method**

The **Authentication Method** step is visible only during the installation or upgrade of MySQL 8.0.4 or higher. It introduces a choice between two server-side authentication options. The MySQL user accounts that you create in the next step will use the authentication method that you select in this step.

MySQL 8.0 connectors and community drivers that use libmysqlclient 8.0 now support the caching\_sha2\_password default authentication plugin. However, if you are unable to update your clients and applications to support this new authentication method, you can configure the MySQL server to use mysql\_native\_password for legacy authentication. For more information about the implications of this change, see caching\_sha2\_password as the Preferred Authentication Plugin.

If you are installing or upgrading to MySQL 8.0.4 or higher, select one of the following authentication methods:

Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8.0 supports a new authentication based on improved, stronger SHA256-based password methods. It is recommended that all new MySQL server installations use this method going forward.

#### **Important**

The caching\_sha2\_password authentication plugin on the server requires new versions of connectors and clients, which add support for the new MySQL 8.0 default authentication.

Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

Using the old MySQL 5.x legacy authentication method should be considered only in the following cases:

- Applications cannot be updated to use MySQL 8.0 connectors and drivers.
- Recompilation of an existing application is not feasible.
- An updated, language-specific connector or driver is not available yet.

#### **Accounts and Roles**

· Root Account Password

Assigning a root password is required and you will be asked for it when performing other MySQL Installer operations. Password strength is evaluated when you repeat the password in the box provided. For descriptive information regarding password requirements or status, move your mouse pointer over

the information icon (  $^{\triangle}$  ) when it appears.

MySQL User Accounts (Optional)

Click **Add User** or **Edit User** to create or modify MySQL user accounts with predefined roles. Next, enter the required account credentials:

- User Name: MySQL user names can be up to 32 characters long.
- Host: Select localhost for local connections only or <All Hosts (%) > when remote connections to the server are required.
- Role: Each predefined role, such as DB Admin, is configured with its own set of privileges. For example, the DB Admin role has more privileges than the DB Designer role. The Role drop-down list contains a description of each role.
- Password: Password strength assessment is performed while you type the password. Passwords must be confirmed. MySQL permits a blank or empty password (considered to be insecure).

**MySQL Installer Commercial Release Only:** MySQL Enterprise Edition for Windows, a commercial product, also supports an authentication method that performs external authentication on Windows. Accounts authenticated by the Windows operating system can access the MySQL server without providing an additional password.

To create a new MySQL account that uses Windows authentication, enter the user name and then select a value for **Host** and **Role**. Click **Windows** authentication to enable the authentication\_windows plugin. In the Windows Security Tokens area, enter a token for each Windows user (or group) who can authenticate with the MySQL user name. MySQL accounts can include security tokens for both local

Windows users and Windows users that belong to a domain. Multiple security tokens are separated by the semicolon character (;) and use the following format for local and domain accounts:

#### Local account

Enter the simple Windows user name as the security token for each local user or group; for example, finley; jeffrey; admin.

#### · Domain account

Use standard Windows syntax  $(domain \setminus domainuser)$  or MySQL syntax  $(domain \setminus domainuser)$  to enter Windows domain users and groups.

For domain accounts, you may need to use the credentials of an administrator within the domain if the account running MySQL Installer lacks the permissions to query the Active Directory. If this is the case, select **Validate Active Directory users with** to activate the domain administrator credentials.

Windows authentication permits you to test all of the security tokens each time you add or modify a token. Click **Test Security Tokens** to validate (or revalidate) each token. Invalid tokens generate a descriptive error message along with a red x icon and red token text. When all tokens resolve as valid (green text without an x icon), you can click **OK** to save the changes.

#### Windows Service

On the Windows platform, MySQL server can run as a named service managed by the operating system and be configured to start up automatically when Windows starts. Alternatively, you can configure MySQL server to run as an executable program that requires manual configuration.

Configure MySQL server as a Windows service (Selected by default.)

When the default configuration option is selected, you can also select the following:

Start the MySQL Server at System Startup

When selected (default), the service startup type is set to Automatic; otherwise, the startup type is set to Manual.

· Run Windows Service as

When Standard System Account is selected (default), the service logs on as Network Service.

The **Custom User** option must have privileges to log on to Microsoft Windows as a service. The **Next** button will be disabled until this user is configured with the required privileges.

A custom user account is configured in Windows by searching for "local security policy" in the Start menu. In the Local Security Policy window, select **Local Policies**, **User Rights Assignment**, and then **Log On As A Service** to open the property dialog. Click **Add User or Group** to add the custom user and then click **OK** in each dialog to save the changes.

• Deselect the Windows Service option.

### **Server File Permissions**

Optionally, permissions set on the folders and files located at C:\ProgramData\MySQL\MySQL Server 8.0\Data can be managed during the server configuration operation. You have the following options:

• MySQL Installer can configure the folders and files with full control granted exclusively to the user running the Windows service, if applicable, and to the Administrators group.

All other groups and users are denied access. This is the default option.

 Have MySQL Installer use a configuration option similar to the one just described, but also have MySQL Installer show which users could have full control.

You are then able to decide if a group or user should be given full control. If not, you can move the qualified members from this list to a second list that restricts all access.

• Have MySQL Installer skip making file-permission changes during the configuration operation.

If you select this option, you are responsible for securing the Data folder and its related files manually after the server configuration finishes.

## **Logging Options**

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

Advanced configuration options are related to the following MySQL log files:

- Error Log
- · General Log
- Slow Query Log
- Bin Log

#### Note

The binary log is enabled by default.

#### **Advanced Options**

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

The advanced-configuration options include:

#### Server ID

Set the unique identifier used in a replication topology. If binary logging is enabled, you must specify a server ID. The default ID value depends on the server version. For more information, see the description of the server\_id system variable.

#### • Table Names Case

You can set the following options during the initial and subsequent configuration the server. For the MySQL 8.0 release series, these options apply only to the initial configuration of the server.

#### · Lower Case

Sets the <code>lower\_case\_table\_names</code> option value to 1 (default), in which table names are stored in lowercase on disk and comparisons are not case-sensitive.

· Preserve Given Case

Sets the lower\_case\_table\_names option value to 2, in which table names are stored as given but compared in lowercase.

### **Apply Server Configuration**

All configuration settings are applied to the MySQL server when you click **Execute**. Use the **Configuration Steps** tab to follow the progress of each action; the icon for each toggles from white to green (with a check mark) on success. Otherwise, the process stops and displays an error message if an individual action times out. Click the **Log** tab to view the log.

When the installation completes successfully and you click **Finish**, MySQL Installer and the installed MySQL products are added to the Microsoft Windows Start menu under the MySQL group. Opening MySQL Installer loads the dashboard where installed MySQL products are listed and other MySQL Installer operations are available.

## 5.3.3.2 MySQL Router Configuration with MySQL Installer

During the initial setup, choose any predetermined setup type, except Server only, to install the latest GA version of the tools. Use the Custom setup type to install an individual tool or specific version. If MySQL Installer is installed on the host already, use the **Add** operation to select and install tools from the MySQL Installer dashboard.

### **MySQL Router Configuration**

MySQL Installer provides a configuration wizard that can bootstrap an installed instance of MySQL Router 8.0 to direct traffic between MySQL applications and an InnoDB Cluster. When configured, MySQL Router runs as a local Windows service.

#### Note

You are prompted to configure MySQL Router after the initial installation and when you reconfigure an installed router explicitly. In contrast, the upgrade operation does not require or prompt you to configure the upgraded product.

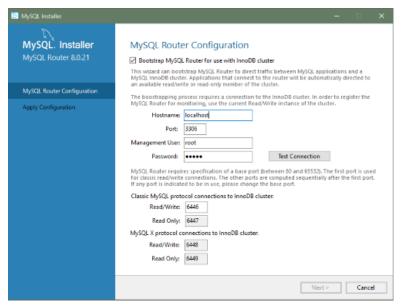
To configure MySQL Router, do the following:

- 1. Set up InnoDB Cluster.
- 2. Using MySQL Installer, download and install the MySQL Router application. After the installation finishes, the configuration wizard prompts you for information. Select the Configure MySQL Router for InnoDB Cluster check box to begin the configuration and provide the following configuration values:
  - Hostname: Host name of the primary (seed) server in the InnoDB Cluster (localhost by default).
  - Port: The port number of the primary (seed) server in the InnoDB Cluster (3306 by default).
  - Management User: An administrative user with root-level privileges.
  - **Password:** The password for the management user.
  - Classic MySQL protocol connections to InnoDB Cluster

**Read/Write:** Set the first base port number to one that is unused (between 80 and 65532) and the wizard will select the remaining ports for you.

The figure that follows shows an example of the MySQL Router configuration page, with the first base port number specified as 6446 and the remaining ports set by the wizard to 6447, 6448, and 6449.

Figure 5.4 MySQL Router Configuration



3. Click **Next** and then **Execute** to apply the configuration. Click **Finish** to close MySQL Installer or return to the MySQL Installer dashboard.

After configuring MySQL Router, the root account exists in the user table as root@localhost (local) only, instead of root@% (remote). Regardless of where the router and client are located, even if both are located on the same host as the seed server, any connection that passes through the router is viewed by server as being remote, not local. As a result, a connection made to the server using the local host (see the example that follows), does not authenticate.

\$> \c root@localhost:6446

# 5.3.4 MySQL Installer Product Catalog and Dashboard

This section describes the MySQL Installer product catalog, the dashboard, and other actions related to product selection and upgrades.

- Product Catalog
- MySQL Installer Dashboard
- · Locating Products to Install
- Upgrading MySQL Server
- Removing MySQL Server
- Upgrading MySQL Installer

## **Product Catalog**

The product catalog stores the complete list of released MySQL products for Microsoft Windows that are available to download from MySQL Downloads. By default, and when an Internet connection is present,

MySQL Installer attempts to update the catalog at startup every seven days. You can also update the catalog manually from the dashboard (described later).

An up-to-date catalog performs the following actions:

- Populates the Available Products pane of the Select Products page. This step appears when you select:
  - The Custom setup type during the initial setup.
  - The Add operation from the dashboard.
- Identifies when product updates are available for the installed products listed in the dashboard.

The catalog includes all development releases (Pre-Release), general releases (Current GA), and minor releases (Other Releases). Products in the catalog will vary somewhat, depending on the MySQL Installer release that you download.

## MySQL Installer Dashboard

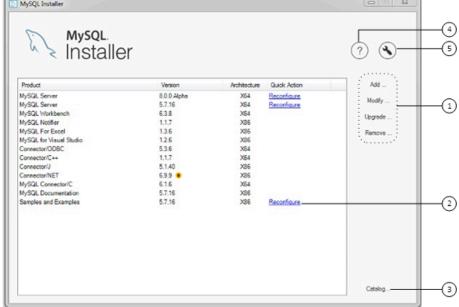
The MySQL Installer dashboard is the default view that you see when you start MySQL Installer after the initial setup finishes. If you closed MySQL Installer before the setup was finished, MySQL Installer resumes the initial setup before it displays the dashboard.

#### Note

Products covered under Oracle Lifetime Sustaining Support, if installed, may appear in the dashboard. These products, such as MySQL for Excel and MySQL Notifier, can be modified or removed only.

MySQL Installer MySQL. Installer

Figure 5.5 MySQL Installer Dashboard Elements



## **Description of MySQL Installer Dashboard Elements**

1. MySQL Installer dashboard operations provide a variety of actions that apply to installed products or products listed in the catalog. To initiate the following operations, first click the operation link and then select the product or products to manage:

• Add: This operation opens the Select Products page. From there you can adjust the filter, select one or more products to download (as needed), and begin the installation. For hints about using the filter, see Locating Products to Install.

Use the directional arrows to move each product from the **Available Products** column to the **Products To Be Installed** column. To enable the Product Features page where you can customize features, click the related check box (disabled by default).

- **Modify**: Use this operation to add or remove the features associated with installed products. Features that you can modify vary in complexity by product. When the **Program Shortcut** check box is selected, the product appears in the Start menu under the MySQL group.
- **Upgrade**: This operation loads the Select Products to Upgrade page and populates it with all the upgrade candidates. An installed product can have more than one upgrade version and the operation requires a current product catalog. MySQL Installer upgrades all of the selected products in one action. Click **Show Details** to view the actions performed by MySQL Installer.
- Remove: This operation opens the Remove Products page and populates it with the MySQL products installed on the host. Select the MySQL products you want to remove (uninstall) and then click **Execute** to begin the removal process. During the operation, an indicator shows the number of steps that are executed as a percentage of all steps.

To select products to remove, do one of the following:

- · Select the check box for one or more products.
- Select the **Product** check box to select all products.
- 2. The Reconfigure link in the Quick Action column next to each installed server loads the current configuration values for the server and then cycles through all configuration steps enabling you to change the options and values. You must provide credentials with root privileges to reconfigure these items. Click the Log tab to show the output of each configuration step performed by MySQL Installer.

On completion, MySQL Installer stops the server, applies the configuration changes, and restarts the server for you. For a description of each configuration option, see Section 5.3.3.1, "MySQL Server Configuration with MySQL Installer". Installed Samples and Examples associated with a specific MySQL server version can be also be reconfigured to apply new feature settings, if any.

3. The **Catalog** link enables you to download the latest catalog of MySQL products manually and then to integrate those product changes with MySQL Installer. The catalog-download action does not perform an upgrade of the products already installed on the host. Instead, it returns to the dashboard and adds an arrow icon to the Version column for each installed product that has a newer version. Use the **Upgrade** operation to install the newer product version.

You can also use the **Catalog** link to display the current change history of each product without downloading the new catalog. Select the **Do not update at this time** check box to view the change history only.

4. The MySQL Installer About icon (?) shows the current version of MySQL Installer and general information about MySQL. The version number is located above the **Back** button.

#### Tip

Always include this version number when reporting a problem with MySQL Installer.

In addition to the About MySQL information (), you can also select the following icons from the side panel:

License icon ( ) for MySQL Installer.

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

- Resource links icon ( ) to the latest MySQL product documentation, blogs, webinars, and more.
- 5. The MySQL Installer Options icon ( ) includes the following tabs:
  - General: Enables or disables the Offline mode option. If selected, this option configures MySQL Installer to run without depending on internet-connection capabilities. When running MySQL Installer in offline mode, you see a warning together with a **Disable** quick action on the dashboard. The warning serves to remind you that running MySQL Installer in offline mode prevents you from downloading the latest MySQL products and product catalog updates. Offline mode persists until you disable the option.

At startup, MySQL Installer determines whether an internet connection is present, and, if not, prompts you to enable offline mode to resume working without a connection.

• **Product Catalog**: Manages the automatic catalog updates. By default, MySQL Installer checks for catalog updates at startup every seven days. When new products or product versions are available,

MySQL Installer adds them to the catalog and then inserts an arrow icon ( ) next to the version number of installed products listed in the dashboard.

Use the product catalog option to enable or disable automatic updates and to reset the number of days between automatic catalog downloads. At startup, MySQL Installer uses the number of days you set to determine whether a download should be attempted. This action is repeated during next startup if MySQL Installer encounters an error downloading the catalog.

Connectivity Settings: Several operations performed by MySQL Installer require internet access.
 This option enables you to use a default value to validate the connection or to use a different URL, one selected from a list or added by you manually. With the Manual option selected, new URLs can be added and all URLs in the list can be moved or deleted. When the Automatic option is selected, MySQL Installer attempts to connect to each default URL in the list (in order) until a connection is made. If no connection can be made, it raises an error.

• **Proxy**: MySQL Installer provides multiple proxy modes that enable you to download MySQL products, updates, or even the product catalog in most network environments. The mode are:

### No proxy

Select this mode to prevent MySQL Installer from looking for system settings. This mode disables any proxy settings.

#### Automatic

Select this mode to have MySQL Installer look for system settings and to use those settings if found, or to use no proxy if nothing is found. This mode is the default.

#### Manual

Select this mode to have MySQL Installer use your authentication details to configuration proxy access to the internet. Specifically:

- A proxy-server address (http://address-to-server) and port number
- · A user name and password for authentication

### **Locating Products to Install**

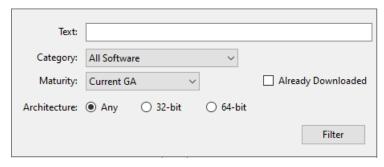
MySQL products in the catalog are listed by category: MySQL Servers, Applications, MySQL Connectors, and Documentation. Only the latest GA versions appear in the **Available Products** pane by default. If you are looking for a pre-release or older version of a product, it may not be visible in the default list.

#### Note

Keep the product catalog up-to-date. Click **Catalog** on the MySQL Installer dashboard to download the latest manifest.

To change the default product list, click **Add** in the dashboard to open the Select Products page, and then click **Edit** to open the dialog box shown in the figure that follows. Modify the settings and then click **Filter**.

Figure 5.6 Filter Available Products



Reset one or more of the following fields to modify the list of available products:

- Text: Filter by text.
- Category: All Software (default), MySQL Servers, Applications, MySQL Connectors, or Documentation (for samples and documentation).
- Maturity: Current Bundle (appears initially with the full package only), Pre-Release, Current GA, or Other Releases. If you see a warning, confirm that you have the most recent product manifest by clicking

**Catalog** on the MySQL Installer dashboard. If MySQL Installer is unable to download the manifest, the range of products you see is limited to bundled products, standalone product MSIs located in the Product Cache folder already, or both.

#### Note

The Commercial release of MySQL Installer does not display any MySQL products when you select the Pre-Release maturity filter. Products in development are available from the Community release of MySQL Installer only.

- Already Downloaded (the check box is deselected by default). Permits you to view and manage downloaded products only.
- Architecture: Any (default), 32-bit, or 64-bit.

## **Upgrading MySQL Server**

Important server upgrade conditions:

- MySQL Installer does not permit server upgrades between major release versions or minor release versions, but does permit upgrades within a release series, such as an upgrade from 8.0.36 to 8.0.37.
- Upgrades between milestone releases (or from a milestone release to a GA release) are not supported.
   Significant development changes take place in milestone releases and you may encounter compatibility issues or problems starting the server.
- For upgrades, a check box enables you to skip the upgrade check and process for system tables, while
  checking and processing data dictionary tables normally. MySQL Installer does not prompt you with
  the check box when the previous server upgrade was skipped or when the server was configured as a
  sandbox InnoDB Cluster. This behavior represents a change in how MySQL Server performs an upgrade
  (see Section 10.4, "What the MySQL Upgrade Process Upgrades") and it alters the sequence of steps
  that MySQL Installer applies to the configuration process.

If you select **Skip system tables upgrade check and process. (Not recommended)**, MySQL Installer starts the upgraded server with the <code>--upgrade=MINIMAL</code> server option, which upgrades the data dictionary only. If you stop and then restart the server without the <code>--upgrade=MINIMAL</code> option, the server upgrades the system tables automatically, if needed.

The following information appears in the **Log** tab and log file after the upgrade configuration (with system tables skipped) is complete:

```
WARNING: The system tables upgrade was skipped after upgrading MySQL Server. The server will be started now with the --upgrade=MINIMAL option, but then each time the server is started it will attempt to upgrade the system tables, unless you modify the Windows service (command line) to add --upgrade=MINIMAL to bypass the upgrade.

FOR THE BEST RESULTS: Run mysqld.exe --upgrade=FORCE on the command line to upgrade
```

the system tables manually.

To choose a new server version:

1. Click **Upgrade**. Confirm that the check box next to product name in the **Upgradeable Products** pane has a check mark. Deselect the products that you do not intend to upgrade at this time.

#### Note

For server milestone releases in the same release series, MySQL Installer deselects the server upgrade and displays a warning to indicate that the

upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform a logical upgrade manually. You can reselect server upgrade at your own risk. For instructions on how to perform a logical upgrade with a milestone release, see Logical Upgrade.

2. Click a product in the list to highlight it. This action populates the **Upgradeable Versions** pane with the details of each available version for the selected product: version number, published date, and a Changes link to open the release notes for that version.

## Removing MySQL Server

To remove a local MySQL server:

- 1. Determine whether the local data directory should be removed. If you retain the data directory, another server installation can reuse the data. This option is enabled by default (removes the data directory).
- 2. Click **Execute** to begin uninstalling the local server. Note that all products that you selected to remove are also uninstalled at this time.
- 3. (Optional) Click the **Log** tab to display the current actions performed by MySQL Installer.

## **Upgrading MySQL Installer**

MySQL Installer remains installed on your computer, and like other software, MySQL Installer can be upgraded from the previous version. In some cases, other MySQL software may require that you upgrade MySQL Installer for compatibility. This section describes how to identify the current version of MySQL Installer and how to upgrade MySQL Installer manually.

### To locate the installed version of MySQL Installer:

- 1. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
- 2. Click the MySQL Installer About icon (?). The version number is located above the **Back** button.

#### To initiate an on-demand upgrade of MySQL Installer:

- 1. Connect the computer with MySQL Installer installed to the internet.
- 2. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
- 3. Click **Catalog** on the bottom of the dashboard to open the Update Catalog window.
- 4. Click **Execute** to begin the process. If the installed version of MySQL Installer can be upgraded, you will be prompted to start the upgrade.
- 5. Click **Next** to review all changes to the catalog and then click **Finish** to return to the dashboard.
- 6. Verify the (new) installed version of MySQL Installer (see the previous procedure).

# 5.3.5 MySQL Installer Console Reference

MySQLInstallerConsole.exe provides command-line functionality that is similar to MySQL Installer. This reference includes:

- MySQL Product Names
- Command Syntax

#### • Command Actions

The console is installed when MySQL Installer is initially executed and then available within the MySQL Installer for Windows directory. By default, the directory location is C:\Program Files (x86)\MySQL\MySQL Installer for Windows. You must run the console as administrator.

To use the console:

- 1. Open a command prompt with administrative privileges by selecting **Windows System** from **Start**, then right-click **Command Prompt**, select **More**, and select **Run as administrator**.
- 2. From the command line, optionally change the directory to where the MySQLInstallerConsole.exe command is located. For example, to use the default installation location:

```
cd Program Files (x86)\MySQL\MySQL Installer for Windows
```

3. Type MySQLInstallerConsole.exe (or mysqlinstallerconsole) followed by a command action to perform a task. For example, to show the console's help:

```
MySQLInstallerConsole.exe --help
MySQL Installer is running in Community mode
Attempting to update manifest.
Initializing product requirements.
Loading product catalog.
Checking for product packages in the bundle.
Categorizing product catalog.
Finding all installed packages.
Your product catalog was last updated at 23/08/2022 12:41:05 p. m.
Your product catalog has version number 671.
========== End Initialization ===============
The following actions are available:
Configure - Configures one or more of your installed programs.
        - Provides list of available command actions.
Install
        - Installs and configures one or more available MySQL programs.
        - Lists all available MySQL products.
List.
Modify
         - Modifies the features of installed products.
         - Removes one or more products from your system.
Remove
         - Configures the general options of MySQL Installer.
Status
         - Shows the status of all installed products.
         - Updates the current product catalog.
Update
        - Upgrades one or more of your installed programs.
The basic syntax for using MySQL Installer command actions. Brackets denote optional entities.
Curly braces denote a list of possible entities.
. . .
```

## **MySQL Product Names**

Many of the MySQLInstallerConsole command actions accept one or more abbreviated phrases that can match a MySQL product (or products) in the catalog. The current set of valid short phrases for use with commands is shown in the following table.

#### Note

Starting with MySQL Installer 1.6.7 (8.0.34), the install, list, and upgrade command options no longer apply to MySQL for Visual Studio (now EOL), MySQL Connector/NET, MySQL Connector/ODBC, MySQL Connector/C++, MySQL

Connector/Python, and MySQL Connector/J. To install newer MySQL connectors, visit https://dev.mysql.com/downloads/.

Table 5.3 MySQL Product Phrases for use with the MySQLInstallerConsole.exe command

Phrase	MySQL Product	
server	MySQL Server	
workbench	MySQL Workbench	
shell	MySQL Shell	
visual	MySQL for Visual Studio	
router	MySQL Router	
backup	MySQL Enterprise Backup (requires the commercial release)	
net	MySQL Connector/NET	
odbc	MySQL Connector/ODBC	
C++	MySQL Connector/C++	
python	MySQL Connector/Python	
j	MySQL Connector/J	
documentation	MySQL Server Documentation	
samples	MySQL Samples (sakila and world databases)	

## **Command Syntax**

The MySQLInstallerConsole.exe command can be issued with or without the file extension (.exe) and the command is not case-sensitive.

mysqlinstallerconsole[.exe][[[--]action][action\_blocks\_list][options\_list]]

#### Description:

action

One of the permitted operational actions. If omitted, the default action is equivalent to the --status action. Using the -- prefix is optional for all actions.

Possible actions are: [--]configure, [--]help, [--]install, [--]list, [--]modify, [--]remove, [--]set, [--]status, [--]update, and [--]upgrade.

action blocks list

A list of blocks in which each represents a different item depending on the selected action. Blocks are separated by commas.

The --remove and --upgrade actions permit specifying an asterisk character (\*) to indicate all products. If the \* character is detected at the start of this block, it is assumed all products are to be processed and the remainder of the block is ignored.

Syntax: \* | action\_block[, action\_block][, action\_block]...

action\_block: Contains a product selector followed by an indefinite number of argument blocks that behave differently depending on the selected action (see Command Actions).

options\_list

Zero or more options with possible values separated by spaces. See Command Actions to identify the options permitted for the corresponding action.

Syntax: option\_value\_pair[ option\_value\_pair][
option\_value\_pair]...

option\_value\_pair: A single option (for example, --silent) or a tuple of a key and a corresponding value with an options prefix. The key-value pair is in the form of --key[=value].

# **Command Actions**

MySQLInstallerConsole.exe supports the following command actions:

### **Note**

Configuration block (or *arguments\_block*) values that contain a colon character (:) must be wrapped in quotation marks. For example, install\_dir="C:\MySQL\MySQL\Server 8.0".

• [--]configure [product1]:[configuration\_argument]=[value], [product2]: [configuration\_argument]=[value], [...]

Configures one or more MySQL products on your system. Multiple *configuration\_argument=value* pairs can be configured for each product.

# Options:

continue	Continues processing the next product when an error is caught while processing the action blocks containing arguments for each product. If not specified the whole operation is aborted in case of an error.
help	Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.
show-settings	Displays the available options for the selected product by passing in the product name aftershow-settings.
silent	Disables confirmation prompts.

# Examples:

```
MySQLInstallerConsole --configure --show-settings server
mysqlinstallerconsole.exe --configure server:port=3307
```

• [--]help

Displays a help message with usage examples and then exits. Pass in an additional command action to receive help specific to that action.

# Options:

--action=[action] Shows the help for a specific action. Same as using the --help option with an action.

Permitted values are: all, configure, help (default), install, list, modify, remove, status, update, upgrade, and set.

--help

Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

### Examples:

MySQLInstallerConsole help

MySQLInstallerConsole help --action=install

• [--]install [product1]:[features]:[config block]:[config block], [product2]: [config block], [...]

Installs one or more MySQL products on your system. If pre-release products are available, both GA and pre-release products are installed when the value of the --type option value is Client or Full. Use the --only\_ga\_products option to restrict the product set to GA products only when using these setup types.

# Description:

[product]

Each product can be specified by a product phrase with or without a semicolon-separated version qualifier. Passing in a product keyword alone selects the latest version of the product. If multiple architectures are available for that version of the product, the command returns the first one in the manifest list for interactive confirmation. Alternatively, you can pass in the exact version and architecture (x86 or x64) after the product keyword using the --silent option.

[features]

All features associated with a MySQL product are installed by default. The feature block is a semicolon-separated list of features or an asterisk character (\*) that selects all features. To remove a feature, use the modify command.

[config block]

One or more configuration blocks can be specified. Each configuration block is a semicolon-separated list of key-value pairs. A block can include either a config or user type key; config is the default type if one is not defined.

Configuration block values that contain a colon character (:) must be wrapped in quotation marks. For example, installdir="C:\MySQL\MySQL\Server 8.0". Only one configuration type block can be defined for each product. A user block should be defined for each user to be created during the product installation.

#### Note

The user type key is not supported when a product is being reconfigured.

## Options:

--auto-handle-preregs

If present, MySQL Installer attempts to download and install some software prerequisites, not currently present. that can be resolved

with minimal intervention. If the <code>--silent</code> option is not present, you are presented with installation pages for each prerequisite. If the <code>--auto-handle-prereqs</code> options is omitted, packages with missing prerequisites are not installed.

--continue

Continues processing the next product when an error is caught while processing the action blocks containing arguments for each product. If not specified the whole operation is aborted in case of an error.

--help

Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

--mos-password=password

Sets the My Oracle Support (MOS) user's password for commercial versions of the MySQL Installer.

--mos-user=user name

Specifies the My Oracle Support (MOS) user name for access to the commercial version of MySQL Installer. If not present, only the products in the bundle, if any, are available to be installed.

--only-ga-products

Restricts the product set to include GA products only.

--setup-type=setup\_type

Installs a predefined set of software. The setup type can be one of the following:

- Server: Installs a single MySQL server
- Client: Installs client programs and libraries (excludes MySQL connectors)
- Full: Installs everything (excludes MySQL connectors)
- Custom: Installs user-selected products. This is the default option.

#### Note

Non-custom setup types are valid only when no other MySQL products are installed.

--show-settings

Displays the available options for the selected product, by passing in the product name after -showsettings.

--silent

Disable confirmation prompts.

### Examples:

```
mysqlinstallerconsole.exe --install j;8.0.29, net;8.0.28 --silent
```

MySQLInstallerConsole install server; 8.0.30:\*:port=3307; server\_id=2:type=user; user=foo

An example that passes in additional configuration blocks, separated by ^ to fit:

```
MySQLInstallerConsole --install server;8.0.30;x64:*:type=config;open_win_firewall=true; ^
    general_log=true;bin_log=true;server_id=3306;tcp_ip=true;port=3306;root_passwd=pass; ^
    install_dir="C:\MySQL\MySQL Server 8.0":type=user;user_name=foo;password=bar;role=DBManager
```

# • [--]list

When this action is used without options, it activates an interactive list from which all of the available MySQL products can be searched. Enter MySQLInstallerConsole --list and specify a substring to search.

# Options:

all	Lists all available products. If this option is used, all other options are ignored.
arch=architecture	Lists that contain the specified architecture. Permitted values are: $x86, x64$ , and any (default). This option can be combined with thename andversion options.
help	Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.
name=package_name	Lists products that contain the specified name (see product phrase), This option can be combined with theversion andarch options.
version=version	Lists products that contain the specified version, such as 8.0 or 5.7. This option can be combined with thename andarch options.

# Examples:

MySQLInstallerConsole --list --name=net --version=8.0

• [--]modify [product1:-removelist|+addlist], [product2:-removelist|+addlist] [...]

Modifies or displays features of a previously installed MySQL product. To display the features of a product, append the product keyword to the command, for example:

MySQLInstallerConsole --modify server

# Options:

--help Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

--silent Disable confirmation prompts.

# Examples:

MySQLInstallerConsole --modify server:+documentation

MySQLInstallerConsole modify server:-debug

• [--]remove [product1], [product2] [...]

Removes one ore more products from your system. An asterisk character (\*) can be passed in to remove all MySQL products with one command.

### Options:

Continue the operation even if an error occurs. --continue Shows the options and available arguments for the corresponding --help action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. Skips the removal of the data directory when removing MySQL Server --keep-datadir products. Disable confirmation prompts.

### Examples:

--silent

mysqlinstallerconsole.exe remove \* MySQLInstallerConsole --remove server --continue

### • [--]set

Sets one or more configurable options that affect how the MySQL Installer program connects to the internet and whether the automatic products-catalog updates feature is activated.

# Options:

catalog- update=bool_value	Enables (true, default) or disables (false) the automatic products catalog update. This option requires an active connection to the internet.
catalog-update- days=int_value	Accepts an integer between 1 (default) and 365 to indicate the number of days between checks for a new catalog update when MySQL Installer is started. Ifcatalog-update is false, this option is ignored.
connection- validation=validation_typ	Sets how MySQL Installer performs the check for an internet perconnection. Permitted values are automatic (default) and manual.
connection-validation-urls=url_list	A double-quote enclosed and comma-separated string that defines the list of URLs to use for checking the internet connection whenconnection-validation is set to manual. Checks are made in

the same order provided. If the first URL fails, the next URL in the list is used and so on.

--offlinemode=bool\_value Enables MySQL Installer to run with or without internet capabilities. Valid modes are:

- True to enable offline mode (run without an internet connection).
- False (default) to disable offline mode (run with an internet connection). Set this mode before downloading the product catalog or any products to install.

--proxy-mode

Specifies the proxy mode. Valid modes are:

- Automatic to automatically identify the proxy based on the system settings.
- None to ensure that no proxy is configured.
- Manual to set the proxy details manually (--proxy-server, -proxy-port, --proxy-username, --proxy-password).

--proxy-password

The password used to authenticate to the proxy server.

--proxy-port

The port used for the proxy server.

--proxy-server

The URL that point to the proxy server.

--proxy-username

The user name used to authenticate to the proxy server.

--reset-defaults

Resets the MySQL Installer options associated with the --set action to the default values.

### Examples:

MySQLIntallerConsole.exe set --reset-defaults

mysqlintallerconsole.exe --set --catalog-update=false

MySQLIntallerConsole --set --catalog-update-days=3

mysqlintallerconsole --set --connection-validation=manual
--connection-validation-urls="https://www.bing.com,http://www.google.com"

## • [--]status

Provides a quick overview of the MySQL products that are installed on the system. Information includes product name and version, architecture, date installed, and install location.

### Options:

--help

Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

### **Examples:**

MySQLInstallerConsole status

# • [--]update

Downloads the latest MySQL product catalog to your system. On success, the catalog is applied the next time either MySQLInstaller or MySQLInstallerConsole.exe is executed.

MySQL Installer automatically checks for product catalog updates when it is started if n days have passed since the last check. Starting with MySQL Installer 1.6.4, the default value is 1 day. Previously, the default value was 7 days.

### Options:

--help

Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

## Examples:

MySQLInstallerConsole update

[--]upgrade [product1:version], [product2:version] [...]

Upgrades one or more products on your system. The following characters are permitted for this action:

\* Pass in \* to upgrade all products to the latest version, or pass in

specific products.

Pass in ! as a version number to upgrade the MySQL product to its

latest version.

### Options:

--continue Continue the operation even if an error occurs.

--help Shows the options and available arguments for the corresponding

action. If present the action is not executed, only the help is shown,

so other action-related options are ignored as well.

--mos-password=password Sets the My Oracle Support (MOS) user's password for commercial

versions of the MySQL Installer.

--mos-user=user\_name Specifies the My Oracle Support (MOS) user name for access to

the commercial version of MySQL Installer. If not present, only the

products in the bundle, if any, are available to be installed.

--silent Disable confirmation prompts.

# Examples:

MySQLInstallerConsole upgrade \*

MySQLInstallerConsole upgrade workbench: 8.0.31

MySQLInstallerConsole upgrade workbench:!

MySQLInstallerConsole --upgrade server;8.0.30:!, j;8.0.29:!

# 5.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive

Users who are installing from the noinstall package can use the instructions in this section to manually install MySQL. The process for installing MySQL from a ZIP Archive package is as follows:

- 1. Extract the main archive to the desired install directory
  - Optional: also extract the debug-test archive if you plan to execute the MySQL benchmark and test suite
- 2. Create an option file
- 3. Choose a MySQL server type
- 4. Initialize MySQL
- 5. Start the MySQL server
- 6. Secure the default user accounts

This process is described in the sections that follow.

# 5.4.1 Extracting the Install Archive

To install MySQL manually, do the following:

- 1. If you are upgrading from a previous version please refer to Section 10.11, "Upgrading MySQL on Windows", before beginning the upgrade process.
- 2. Make sure that you are logged in as a user with administrator privileges.
- 3. Choose an installation location. Traditionally, the MySQL server is installed in C:\mysql. If you do not install MySQL at C:\mysql, you must specify the path to the install directory during startup or in an option file. See Section 5.4.2, "Creating an Option File".

#### Note

The MySQL Installer installs MySQL under C:\Program Files\MySQL.

4. Extract the install archive to the chosen installation location using your preferred file-compression tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.

# 5.4.2 Creating an Option File

If you need to specify startup options when you run the server, you can indicate them on the command line or place them in an option file. For options that are used every time the server starts, you may find it most convenient to use an option file to specify your MySQL configuration. This is particularly true under the following circumstances:

- The installation or data directory locations are different from the default locations (C:\Program Files \MySQL\MySQL Server 8.0 and C:\Program Files\MySQL\MySQL Server 8.0\data).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, C:\, and the MySQL installation directory (for the full list of locations, see Using Option

Files). The Windows directory typically is named something like C:\WINDOWS. You can determine its exact location from the value of the WINDIR environment variable using the following command:

```
C:\> echo %WINDIR%
```

MySQL looks for options in each location first in the my.ini file, and then in the my.cnf file. However, to avoid confusion, it is best if you use only one file. If your PC uses a boot loader where C: is not the boot drive, your only option is to use the my.ini file. Whichever option file you use, it must be a plain text file.

### **Note**

When using the MySQL Installer to install MySQL Server, it creates the my.ini at the default location, and the user executing MySQL Installer is granted full permissions to this new my.ini file.

In other words, be sure that the MySQL Server user has permission to read the my.ini file.

You can also make use of the example option files included with your MySQL distribution; see Server Configuration Defaults.

An option file can be created and modified with any text editor, such as Notepad. For example, if MySQL is installed in E:\mysql and the data directory is in E:\mydata\data, you can create an option file containing a [mysqld] section to specify values for the basedir and datadir options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

The rules for use of backslash in option file values are given in Using Option Files.

The ZIP archive does not include a data directory. To initialize a MySQL installation by creating the data directory and populating the tables in the mysql system database, initialize MySQL using either -- initialize or --initialize-insecure. For additional information, see Section 9.1, "Initializing the Data Directory".

If you would like to use a data directory in a different location, you should copy the entire contents of the data directory to the new location. For example, if you want to use E:\mydata as the data directory instead, you must do two things:

- 1. Move the entire data directory and all of its contents from the default location (for example C: \Program Files\MySQL\MySQL Server 8.0\data) to E:\mydata.
- 2. Use a --datadir option to specify the new data directory location each time you start the server.

# 5.4.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 8.0.

Binary	Description
mysqld	Optimized binary with named-pipe support
	Like mysqld, but compiled with full debugging and automatic memory allocation checking

All of the preceding binaries are optimized for modern Intel processors, but should work on any Intel i386-class or higher processor.

Each of the servers in a distribution support the same set of storage engines. The SHOW ENGINES statement displays which engines a given server supports.

All Windows MySQL 8.0 servers have support for symbolic linking of database directories.

MySQL supports TCP/IP on all Windows platforms. MySQL servers on Windows also support named pipes, if you start the server with the named\_pipe system variable enabled. It is necessary to enable this variable explicitly because some users have experienced problems with shutting down the MySQL server when named pipes were used. The default is to use TCP/IP regardless of platform because named pipes are slower than TCP/IP in many Windows configurations.

# 5.4.4 Initializing the Data Directory

If you installed MySQL using the noinstall package, no data directory is included. To initialize the data directory, use the instructions at Section 9.1, "Initializing the Data Directory".

# 5.4.5 Starting the Server for the First Time

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the noinstall version, or if you wish to configure and test MySQL manually rather than with the MySQL Installer.

The examples in these sections assume that MySQL is installed under the default location of C:\Program Files\MySQL\MySQL Server 8.0. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the shared\_memory system variable enabled. Clients can connect through shared memory by using the --protocol=MEMORY option.

For information about which server binary to run, see Section 5.4.3, "Selecting a MySQL Server Type".

Testing is best done from a command prompt in a console window (or "DOS window"). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

#### Note

The database must be initialized before MySQL can be started. For additional information about the initialization process, see Section 9.1, "Initializing the Data Directory".

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console
```

You should see messages similar to those following as it starts (the path names and sizes may differ). The ready for connections messages indicate that the server is ready to service client connections.

```
[Server] C:\mysql\bin\mysqld.exe (mysqld 8.0.30) starting as process 21236
[InnoDB] InnoDB initialization has started.
[InnoDB] InnoDB initialization has ended.
[Server] CA certificate ca.pem is self signed.
[Server] Channel mysql_main configured to support TLS.
Encrypted connections are now supported for this channel.
[Server] X Plugin ready for connections. Bind-address: '::' port: 33060
[Server] C:\mysql\bin\mysqld.exe: ready for connections.
Version: '8.0.30' socket: '' port: 3306 MySQL Community Server - GPL.
```

You can now open a new console window in which to run client programs.

If you omit the --console option, the server writes diagnostic output to the error log in the data directory (C:\Program Files\MySQL\MySQL Server 8.0\data by default). The error log is the file with the .err extension, and may be set using the --log-error option.

#### Note

The initial root account in the MySQL grant tables has no password. After starting the server, you should set up a password for it using the instructions in Section 9.4, "Securing the Initial MySQL Account".

# 5.4.6 Starting MySQL from the Windows Command Line

The MySQL server can be started manually from the command line. This can be done on any version of Windows.

To start the mysqld server from the command line, you should start a console window (or "DOS window") and enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
```

The path to mysqld may vary depending on the install location of MySQL on your system.

You can stop the MySQL server by executing this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" -u root shutdown
```

#### Note

If the MySQL root user account has a password, you need to invoke mysqladmin with the -p option and supply the password when prompted.

This command invokes the MySQL administrative utility mysqladmin to connect to the server and tell it to shut down. The command connects as the MySQL root user, which is the default administrative account in the MySQL grant system.

#### Note

Users in the MySQL grant system are wholly independent from any operating system users under Microsoft Windows.

If mysqld doesn't start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. By default, the error log is located in the C:\Program Files\MySQL\MySQL
Server 8.0\data directory. It is the file with a suffix of .err, or may be specified by passing in the --

log-error option. Alternatively, you can try to start the server with the --console option; in this case, the server may display some useful information on the screen to help solve the problem.

The last option is to start <code>mysqld</code> with the <code>--standalone</code> and <code>--debug</code> options. In this case, <code>mysqld</code> writes a log file <code>C:\mysqld.trace</code> that should contain the reason why <code>mysqld</code> doesn't start. See The <code>DBUG</code> Package.

Use mysqld --verbose --help to display all the options that mysqld supports.

# 5.4.7 Customizing the PATH for MySQL Tools

# Warning

You must exercise great care when editing your system PATH by hand; accidental deletion or modification of any portion of the existing PATH value can leave you with a malfunctioning or even unusable system.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL bin directory to your Windows system PATH environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the Advanced tab from the System Properties menu that appears, and click the Environment Variables button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL bin directory (for example, C:\Program Files\MySQL\MySQL\Server 8.0\bin)

#### Note

There must be a semicolon separating this path from any values present in this field.

Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. The new PATH value should now be available to any new command shell you open, allowing you to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the mysql client, and all MySQL command-line utilities such as mysqladmin and mysqldump.

You should not add the MySQL bin directory to your Windows PATH if you are running multiple MySQL servers on the same machine.

# 5.4.8 Starting MySQL as a Windows Service

On Windows, the recommended way to run MySQL is to install it as a Windows service, so that MySQL starts and stops automatically when Windows starts and stops. A MySQL server installed as a service can also be controlled from the command line using NET commands, or with the graphical Services utility. Generally, to install MySQL as a Windows service you should be logged in using an account that has administrator rights.

The Services utility (the Windows Service Control Manager) can be found in the Windows Control Panel. To avoid conflicts, it is advisable to close the Services utility while performing server installation or removal operations from the command line.

# Installing the service

Before installing MySQL as a Windows service, you should first stop the current server if it is running by using the following command:

#### Note

If the MySQL root user account has a password, you need to invoke mysqladmin with the -p option and supply the password when prompted.

This command invokes the MySQL administrative utility mysqladmin to connect to the server and tell it to shut down. The command connects as the MySQL root user, which is the default administrative account in the MySQL grant system.

#### Note

Users in the MySQL grant system are wholly independent from any operating system users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install
```

The service-installation command does not start the server. Instructions for that are given later in this section.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL bin directory to your Windows system PATH environment variable:

- On the Windows desktop, right-click the My Computer icon, and select Properties.
- Next select the Advanced tab from the System Properties menu that appears, and click the Environment Variables button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL bin directory (for example, C:\Program Files\MySQL\MySQL\Server 8.0\bin), and there should be a semicolon separating this path from any values present in this field. Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the mysql client, and all MySQL command-line utilities such as mysqladmin and mysqldump.

You should not add the MySQL bin directory to your Windows PATH if you are running multiple MySQL servers on the same machine.

#### Warning

You must exercise great care when editing your system PATH by hand; accidental deletion or modification of any portion of the existing PATH value can leave you with a malfunctioning or even unusable system.

The following additional arguments can be used when installing the service:

- You can specify a service name immediately following the --install option. The default service name is MySQL.
- If a service name is given, it can be followed by a single option. By convention, this should be -- defaults-file=file\_name to specify the name of an option file from which the server should read options when it starts.

The use of a single option other than --defaults-file is possible but discouraged. --defaults-file is more flexible because it enables you to specify multiple startup options for the server by placing them in the named option file.

• You can also specify a --local-service option following the service name. This causes the server to run using the LocalService Windows account that has limited system privileges. If both -- defaults-file and --local-service are given following the service name, they can be in any order.

For a MySQL server that is installed as a Windows service, the following rules determine the service name and option files that the server uses:

- If the service-installation command specifies no service name or the default service name (MySQL) following the --install option, the server uses the service name of MySQL and reads options from the [mysqld] group in the standard option files.
- If the service-installation command specifies a service name other than MySQL following the --install option, the server uses that service name. It reads options from the [mysqld] group and the group that has the same name as the service in the standard option files. This enables you to use the [mysqld] group for options that should be used by all MySQL services, and an option group with the service name for use by the server installed with that service name.
- If the service-installation command specifies a --defaults-file option after the service name, the server reads options the same way as described in the previous item, except that it reads options only from the named file and ignores the standard option files.

As a more complex example, consider the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
--install MySQL --defaults-file=C:\my-opts.cnf
```

Here, the default service name (MySQL) is given after the --install option. If no --defaults-file option had been given, this command would have the effect of causing the server to read the [mysqld] group from the standard option files. However, because the --defaults-file option is present, the server reads options from the [mysqld] option group, and only from the named file.

#### Note

On Windows, if the server is started with the --defaults-file and --install options, --install must be first. Otherwise, mysqld.exe attempts to start the MySQL server.

You can also specify options as Start parameters in the Windows Services utility before you start the MySQL service.

Finally, before trying to start the MySQL service, make sure the user variables %TEMP% and %TMP% (and also %TMPDIR%, if it has ever been set) for the operating system user who is to run the service are pointing to a folder to which the user has write access. The default user for running the MySQL service is LocalSystem, and the default value for its %TEMP% and %TMP% is C:\Windows\Temp, a directory

LocalSystem has write access to by default. However, if there are any changes to that default setup (for example, changes to the user who runs the service or to the mentioned user variables, or the --tmpdir option has been used to put the temporary directory somewhere else), the MySQL service might fail to run because write access to the temporary directory has not been granted to the proper user.

# Starting the service

After a MySQL server instance has been installed as a service, Windows starts the service automatically whenever Windows starts. The service also can be started immediately from the Services utility, or by using an sc start <code>mysqld\_service\_name</code> or <code>NET START mysqld\_service\_name</code> command. SC and <code>NET commands</code> are not case-sensitive.

When run as a service, <code>mysqld</code> has no access to a console window, so no messages can be seen there. If <code>mysqld</code> does not start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. The error log is located in the MySQL data directory (for example, <code>C:\Program Files\MySQL\MySQL Server 8.0\data</code>). It is the file with a suffix of <code>.err</code>.

When a MySQL server has been installed as a service, and the service is running, Windows stops the service automatically when Windows shuts down. The server also can be stopped manually using the Services utility, the sc stop <code>mysqld\_service\_name</code> command, the <code>NET STOP</code> <code>mysqld\_service\_name</code> command, or the <code>mysqladmin shutdown</code> command.

You also have the choice of installing the server as a manual service if you do not wish for the service to be started automatically during the boot process. To do this, use the --install-manual option rather than the --install option:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install-manual
```

# Removing the service

To remove a server that is installed as a service, first stop it if it is running by executing SC STOP mysqld\_service\_name or NET STOP mysqld\_service\_name. Then use SC DELETE mysqld\_service\_name to remove it:

```
C:\> SC DELETE mysql
```

Alternatively, use the mysqld --remove option to remove the service.

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --remove
```

If mysqld is not running as a service, you can start it from the command line. For instructions, see Section 5.4.6, "Starting MySQL from the Windows Command Line".

If you encounter difficulties during installation, see Section 5.5, "Troubleshooting a Microsoft Windows MySQL Server Installation".

For more information about stopping or removing a Windows service, see Starting Multiple MySQL Instances as Windows Services.

# 5.4.9 Testing The MySQL Installation

You can test whether the MySQL server is working by executing any of the following commands:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql" test
```

If mysqld is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start mysqld with the skip\_name\_resolve system variable enabled and

use only localhost and IP addresses in the Host column of the MySQL grant tables. (Be sure that an account exists that specifies an IP address or you may not be able to connect.)

You can force a MySQL client to use a named-pipe connection rather than TCP/IP by specifying the -pipe or --protocol=PIPE option, or by specifying . (period) as the host name. Use the --socket
option to specify the name of the pipe if you do not want to use the default pipe name.

If you have set a password for the root account, deleted the anonymous account, or created a new user account, then to connect to the MySQL server you must use the appropriate -u and -p options with the commands shown previously. See Connecting to the MySQL Server Using Command Options.

For more information about <code>mysqlshow</code>, see <code>mysqlshow</code> — Display Database, Table, and Column Information.

# 5.5 Troubleshooting a Microsoft Windows MySQL Server Installation

When installing and running MySQL for the first time, you may encounter certain errors that prevent the MySQL server from starting. This section helps you diagnose and correct some of these errors.

Your first resource when troubleshooting server issues is the error log. The MySQL server uses the error log to record information relevant to the error that prevents the server from starting. The error log is located in the data directory specified in your my.ini file. The default data directory location is C:\Program Files\MySQL\MySQL Server 8.0\data, or C:\ProgramData\Mysql on Windows 7 and Windows Server 2008. The C:\ProgramData directory is hidden by default. You need to change your folder options to see the directory and contents. For more information on the error log and understanding the content, see The Error Log.

For information regarding possible errors, also consult the console messages displayed when the MySQL service is starting. Use the SC START <code>mysqld\_service\_name</code> or NET START <code>mysqld\_service\_name</code> or NET START <code>mysqld\_service\_name</code> command from the command line after installing <code>mysqld</code> as a service to see any error messages regarding the starting of the MySQL server as a service. See Section 5.4.8, "Starting MySQL as a Windows Service".

The following examples show other common error messages you might encounter when installing MySQL and starting the server for the first time:

If the MySQL server cannot find the mysql privileges database or other critical files, it displays these
messages:

```
System error 1067 has occurred.

Fatal error: Can't open and lock privilege tables:

Table 'mysql.user' doesn't exist
```

These messages often occur when the MySQL base or data directories are installed in different locations than the default locations (C:\Program Files\MySQL\MySQL Server 8.0 and C:\Program Files\MySQL\MySQL Server 8.0\data, respectively).

This situation can occur when MySQL is upgraded and installed to a new location, but the configuration file is not updated to reflect the new location. In addition, old and new configuration files might conflict. Be sure to delete or rename any old configuration files when upgrading MySQL.

If you have installed MySQL to a directory other than C:\Program Files\MySQL\MySQL\Server 8.0, ensure that the MySQL server is aware of this through the use of a configuration (my.ini) file. Put the my.ini file in your Windows directory, typically C:\WINDOWS. To determine its exact location from the value of the WINDIR environment variable, issue the following command from the command prompt:

```
C:\> echo %WINDIR%
```

You can create or modify an option file with any text editor, such as Notepad. For example, if MySQL is installed in E:\mysql and the data directory is D:\MySQLdata, you can create the option file and set up a [mysqld] section to specify values for the basedir and datadir options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=D:/MySQLdata
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=C:\\Program Files\\MySQL\\MySQL Server 8.0
# set datadir to the location of your data directory
datadir=D:\\MySQLdata
```

The rules for use of backslash in option file values are given in Using Option Files.

If you change the datadir value in your MySQL configuration file, you must move the contents of the existing MySQL data directory before restarting the MySQL server.

See Section 5.4.2, "Creating an Option File".

 If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service and install MySQL using the MySQL Installer, you might see this error:

```
Error: Cannot create Windows service for MySql. Error: 0
```

This occurs when the Configuration Wizard tries to install the service and finds an existing service with the same name.

One solution to this problem is to choose a service name other than mysql when using the configuration wizard. This enables the new service to be installed correctly, but leaves the outdated service in place. Although this is harmless, it is best to remove old services that are no longer in use.

To permanently remove the old mysql service, execute the following command as a user with administrative privileges, on the command line:

```
C:\> SC DELETE mysql
[SC] DeleteService SUCCESS
```

If the SC utility is not available for your version of Windows, download the delsrv utility from http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp and use the delsrv mysql syntax.

# 5.6 Windows Postinstallation Procedures

GUI tools exist that perform most of the tasks described in this section, including:

- MySQL Installer: Used to install and upgrade MySQL products.
- MySQL Workbench: Manages the MySQL server and edits SQL statements.

If necessary, initialize the data directory and create the MySQL grant tables. Windows installation operations performed by MySQL Installer initialize the data directory automatically. For installation from

a ZIP Archive package, initialize the data directory as described at Section 9.1, "Initializing the Data Directory".

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned a password to the initial root account. (See Section 5.3, "MySQL Installer for Windows".) Otherwise, use the password-assignment procedure given in Section 9.4, "Securing the Initial MySQL Account".

Before assigning a password, you might want to try running some client programs to make sure that you can connect to the server and that it is operating properly. Make sure that the server is running (see Section 5.4.5, "Starting the Server for the First Time"). You can also set up a MySQL service that runs automatically when Windows starts (see Section 5.4.8, "Starting MySQL as a Windows Service").

These instructions assume that your current location is the MySQL installation directory and that it has a bin subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

If you installed MySQL using MySQL Installer (see Section 5.3, "MySQL Installer for Windows"), the default installation directory is C:\Program Files\MySQL\MySQL Server 8.0:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0"
```

A common installation location for installation from a ZIP archive is C:\mysql:

```
C:\> cd C:\mysql
```

Alternatively, add the bin directory to your PATH environment variable setting. That enables your command interpreter to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See Section 5.4.7, "Customizing the PATH for MySQL Tools".

With the server running, issue the following commands to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use mysglshow to see what databases exist:

The list of installed databases may vary, but always includes at least mysql and information\_schema.

The preceding command (and commands for other MySQL programs such as mysql) may not work if the correct MySQL account does not exist. For example, the program may fail with an error, or you may not be able to view all databases. If you install MySQL using MySQL Installer, the root user is created automatically with the password you supplied. In this case, you should use the -u root and -p options. (You must use those options if you have already secured the initial MySQL accounts.) With -p, the client program prompts for the root password. For example:

+----+

If you specify a database name, mysqlshow displays a list of the tables within the database:

```
C:\> bin\mysqlshow mysql
Database: mysql
     Tables
  columns_priv
  component
 db
 default_roles
 engine_cost
 func
 general_log
 global_grants
 gtid_executed
 help_category
 help_keyword
  help_relation
 help_topic
 innodb_index_stats
  innodb_table_stats
 ndb_binlog_index
  password_history
 plugin
 procs_priv
 proxies_priv
 role_edges
 server_cost
  servers
  slave_master_info
 slave_relay_log_info
 slave_worker_info
 slow_log
 tables_priv
  time_zone
 time_zone_leap_second
 time_zone_name
 time_zone_transition
 time_zone_transition_type
 user
```

Use the <code>mysql</code> program to select information from a table in the <code>mysql</code> database:

For more information about mysql and mysqlshow, see mysql — The MySQL Command-Line Client, and mysqlshow — Display Database, Table, and Column Information.

# 5.7 Windows Platform Restrictions

The following restrictions apply to use of MySQL on the Windows platform:

### Process memory

On Windows 32-bit platforms, it is not possible by default to use more than 2GB of RAM within a single process, including MySQL. This is because the physical address limit on Windows 32-bit is 4GB and

the default setting within Windows is to split the virtual address space between kernel (2GB) and user/applications (2GB).

Some versions of Windows have a boot time setting to enable larger applications by reducing the kernel application. Alternatively, to use more than 2GB, use a 64-bit version of Windows.

# · File system aliases

When using MyISAM tables, you cannot use aliases within Windows link to the data files on another volume and then link back to the main MySQL datadir location.

This facility is often used to move the data and index files to a RAID or other fast solution.

### Limited number of ports

Windows systems have about 4,000 ports available for client connections, and after a connection on a port closes, it takes two to four minutes before the port can be reused. In situations where clients connect to and disconnect from the server at a high rate, it is possible for all available ports to be used up before closed ports become available again. If this happens, the MySQL server appears to be unresponsive even though it is running. Ports may be used by other applications running on the machine as well, in which case the number of ports available to MySQL is lower.

For more information about this problem, see https://support.microsoft.com/kb/196271.

#### DATA DIRECTORY and INDEX DIRECTORY

The DATA DIRECTORY clause of the CREATE TABLE statement is supported on Windows for InnoDB tables only, as described in Creating Tables Externally. For MyISAM and other storage engines, the DATA DIRECTORY and INDEX DIRECTORY clauses for CREATE TABLE are ignored on Windows and any other platforms with a nonfunctional realpath() call.

### • DROP DATABASE

You cannot drop a database that is in use by another session.

#### Case-insensitive names

File names are not case-sensitive on Windows, so MySQL database and table names are also not case-sensitive on Windows. The only restriction is that database and table names must be specified using the same case throughout a given statement. See Identifier Case Sensitivity.

#### · Directory and file names

On Windows, MySQL Server supports only directory and file names that are compatible with the current ANSI code pages. For example, the following Japanese directory name does not work in the Western locale (code page 1252):

datadir="C:/私たちのプロジェクトのデータ"

The same limitation applies to directory and file names referred to in SQL statements, such as the data file path name in LOAD DATA.

# • The \ path name separator character

Path name components in Windows are separated by the \ character, which is also the escape character in MySQL. If you are using LOAD DATA or SELECT ... INTO OUTFILE, use Unix-style file names with / characters:

```
mysql> LOAD DATA INFILE 'C:/tmp/skr.txt' INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

Alternatively, you must double the \ character:

```
mysql> LOAD DATA INFILE 'C:\\tmp\\skr.txt' INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

# · Problems with pipes

Pipes do not work reliably from the Windows command-line prompt. If the pipe includes the character ^Z / CHAR(24), Windows thinks that it has encountered end-of-file and aborts the program.

This is mainly a problem when you try to apply a binary log as follows:

```
C:\> mysqlbinlog binary_log_file | mysql --user=root
```

If you have a problem applying the log and suspect that it is because of a ^Z / CHAR (24) character, you can use the following workaround:

```
C:\> mysqlbinlog binary_log_file --result-file=/tmp/bin.sql
C:\> mysql --user=root --execute "source /tmp/bin.sql"
```

The latter command also can be used to reliably read any SQL file that may contain binary data.

# Chapter 6 Installing MySQL on macOS

# **Table of Contents**

6.1 General Notes on Installing MySQL on macOS	. 125
6.2 Installing MySQL on macOS Using Native Packages	. 126
6.3 Installing and Using the MySQL Launch Daemon	. 132
6.4 Installing and Using the MySQL Preference Pane	135

For a list of macOS versions that the MySQL server supports, see https://www.mysql.com/support/supportedplatforms/database.html.

MySQL for macOS is available in a number of different forms:

- Native Package Installer, which uses the native macOS installer (DMG) to walk you through the
  installation of MySQL. For more information, see Section 6.2, "Installing MySQL on macOS Using Native
  Packages". You can use the package installer with macOS. The user you use to perform the installation
  must have administrator privileges.
- Compressed TAR archive, which uses a file packaged using the Unix tar and gzip commands. To use
  this method, you need to open a Terminal window. You do not need administrator privileges using this
  method; you can install the MySQL server anywhere using this method. For more information on using
  this method, you can use the generic instructions for using a tarball, Chapter 3, Installing MySQL on
  Unix/Linux Using Generic Binaries.

In addition to the core installation, the Package Installer also includes Section 6.3, "Installing and Using the MySQL Launch Daemon" and Section 6.4, "Installing and Using the MySQL Preference Pane" to simplify the management of your installation.

For additional information on using MySQL on macOS, see Section 6.1, "General Notes on Installing MySQL on macOS".

# 6.1 General Notes on Installing MySQL on macOS

You should keep the following issues and notes in mind:

Other MySQL installations: The installation procedure does not recognize MySQL installations by
package managers such as Homebrew. The installation and upgrade process is for MySQL packages
provided by us. If other installations are present, then consider stopping them before executing this
installer to avoid port conflicts.

**Homebrew**: For example, if you installed MySQL Server using Homebrew to its default location then the MySQL installer installs to a different location and won't upgrade the version from Homebrew. In this scenario you would end up with multiple MySQL installations that, by default, attempt to use the same ports. Stop the other MySQL Server instances before running this installer, such as executing *brew services stop mysql* to stop the Homebrew's MySQL service.

- Launchd: A launchd daemon is installed that alters MySQL configuration options. Consider editing it if needed, see the documentation below for additional information. Also, macOS 10.10 removed startup item support in favor of launchd daemons. The optional MySQL preference pane under macOS System Preferences uses the launchd daemon.
- **Users**: You may need (or want) to create a specific <code>mysql</code> user to own the MySQL directory and data. You can do this through the <code>Directory Utility</code>, and the <code>mysql</code> user should already exist. For use

in single user mode, an entry for \_mysql (note the underscore prefix) should already exist within the system /etc/passwd file.

- Data: Because the MySQL package installer installs the MySQL contents into a version and platform
  specific directory, you can use this to upgrade and migrate your database between versions. You need
  either to copy the data directory from the old version to the new version, or to specify an alternative
  datadir value to set location of the data directory. By default, the MySQL directories are installed under
  /usr/local/.
- Aliases: You might want to add aliases to your shell's resource file to make it easier to access
  commonly used programs such as mysql and mysqladmin from the command line. The syntax for
  bash is:

```
alias mysql=/usr/local/mysql/bin/mysql
alias mysqladmin=/usr/local/mysql/bin/mysqladmin
```

For tcsh, use:

```
alias mysql /usr/local/mysql/bin/mysql alias mysqladmin /usr/local/mysql/bin/mysqladmin
```

Even better, add /usr/local/mysql/bin to your PATH environment variable. You can do this by modifying the appropriate startup file for your shell. For more information, see Invoking MySQL Programs.

• **Removing**: After you have copied over the MySQL database files from the previous installation and have successfully started the new server, you should consider removing the old installation files to save disk space. Additionally, you should also remove older versions of the Package Receipt directories located in /Library/Receipts/mysql-VERSION.pkg.

# 6.2 Installing MySQL on macOS Using Native Packages

The package is located inside a disk image (.dmg) file that you first need to mount by double-clicking its icon in the Finder. It should then mount the image and display its contents.

## Note

Before proceeding with the installation, be sure to stop all running MySQL server instances by using either the MySQL Manager Application (on macOS Server), the preference pane, or mysqladmin shutdown on the command line.

To install MySQL using the package installer:

1. Download the disk image (.dmg) file (the community version is available here) that contains the MySQL package installer. Double-click the file to mount the disk image and see its contents.

Double-click the MySQL installer package from the disk. It is named according to the version of MySQL you have downloaded. For example, for MySQL server 8.0.42 it might be named  $mysq1-8.0.42-macos-10.13-x86_64.pkg$ .

2. The initial wizard introduction screen references the MySQL server version to install. Click **Continue** to begin the installation.

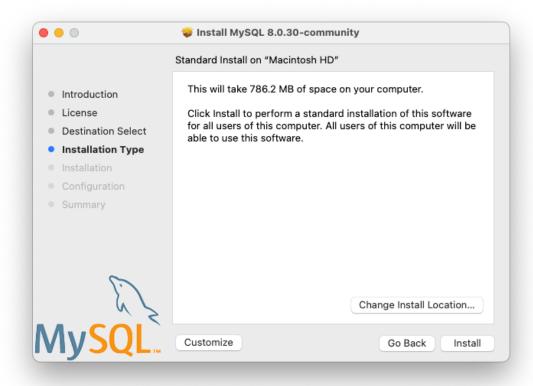
The MySQL community edition shows a copy of the relevant GNU General Public License. Click **Continue** and then **Agree** to continue.

3. From the **Installation Type** page you can either click **Install** to execute the installation wizard using all defaults, click **Customize** to alter which components to install (MySQL server, MySQL Test, Preference Pane, Launchd Support -- all but MySQL Test are enabled by default).

### Note

Although the **Change Install Location** option is visible, the installation location cannot be changed.

Figure 6.1 MySQL Package Installer Wizard: Installation Type



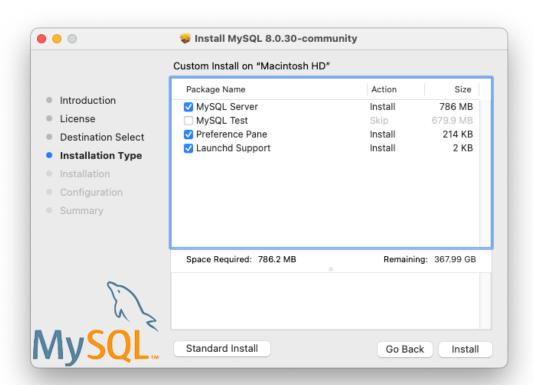


Figure 6.2 MySQL Package Installer Wizard: Customize

- 4. Click **Install** to install MySQL Server. The installation process ends here if upgrading a current MySQL Server installation, otherwise follow the wizard's additional configuration steps for your new MySQL Server installation.
- 5. After a successful new MySQL Server installation, complete the configuration steps by choosing the default encryption type for passwords, define the root password, and also enable (or disable) MySQL server at startup.

6. The default MySQL 8.0 password mechanism is caching\_sha2\_password (Strong), and this step allows you to change it to mysql\_native\_password (Legacy).

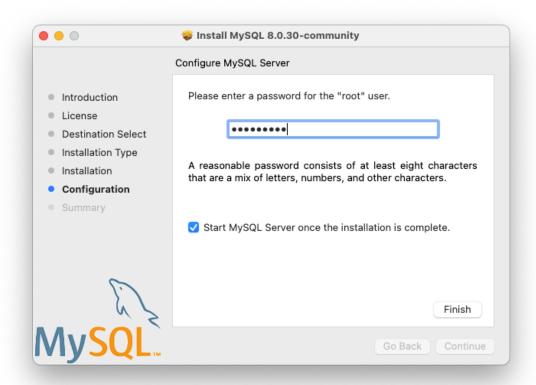
Figure 6.3 MySQL Package Installer Wizard: Choose a Password Encryption Type



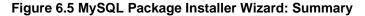
Choosing the legacy password mechanism alters the generated launchd file to set -default\_authentication\_plugin=mysql\_native\_password under ProgramArguments.
Choosing strong password encryption does not set --default\_authentication\_plugin because
the default MySQL Server value is used, which is caching\_sha2\_password.

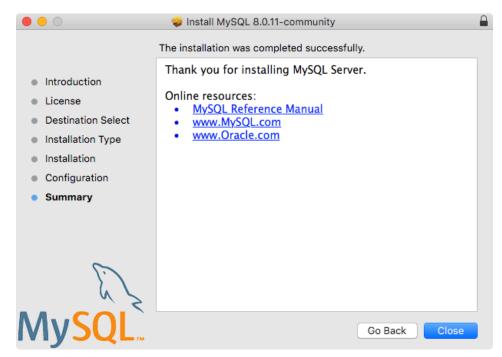
7. Define a password for the root user, and also toggle whether MySQL Server should start after the configuration step is complete.

Figure 6.4 MySQL Package Installer Wizard: Define Root Password



8. **Summary** is the final step and references a successful and complete MySQL Server installation. **Close** the wizard.





MySQL server is now installed. If you chose to not start MySQL, then use either launchctl from the command line or start MySQL by clicking "Start" using the MySQL preference pane. For additional information, see Section 6.3, "Installing and Using the MySQL Launch Daemon", and Section 6.4, "Installing and Using the MySQL Preference Pane". Use the MySQL Preference Pane or launchd to configure MySQL to automatically start at bootup.

#### Note

The macOS installation process does not create nor install a sample  ${\tt my.cnf}$  MySQL configuration file.

Table 6.1 MySQL Installation Layout on macOS

Directory	Contents of Directory
bin	mysqld server, client and utility programs
data	Log files, databases, where /usr/local/mysql/data/mysqld.local.err is the default error log
docs	Helper documents, like the Release Notes and build information
include	Include (header) files

Directory	Contents of Directory
lib	Libraries
man	Unix manual pages
mysql-test	MySQL test suite ('MySQL Test' is disabled by default during the installation process when using the installer package (DMG))
share	Miscellaneous support files, including error messages, dictionary.txt, and rewriter SQL
support-files	Support scripts, such as mysqld_multi.server, mysql.server, and mysql-log-rotate.
/tmp/mysql.sock	Location of the MySQL Unix socket

# 6.3 Installing and Using the MySQL Launch Daemon

macOS uses launch daemons to automatically start, stop, and manage processes and applications such as MySQL.

By default, the installation package (DMG) on macOS installs a launchd file named /Library/LaunchDaemons/com.oracle.oss.mysql.mysqld.plist that contains a plist definition similar to:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"</pre>
<plist version="1.0">
<dict>
   <key>Label</key>
                                <string>com.oracle.oss.mysql.mysqld</string>
   <key>ProcessType</key>
                                <string>Interactive</string>
   <key>Disabled</key>
                                 <false/>
   <key>RunAtLoad</key>
                                <true/>
   <key>KeepAlive</key>
                               <true/>
   <key>SessionCreate</key> <true/>
   <key>LaunchOnlyOnce</key>
                                <false/>
   <key>UserName</key>
                                 <string>_mysql</string>
   <key>GroupName
<key>ExitTimeOut</key>
<key>ExitTimeOut</key>
<string>/usr/local/mysql/bin/mysqld</string>
   <key>GroupName</key>
                                 <string>_mysql</string>
   <key>ProgramArguments</key>
       <array>
            <string>/usr/local/mysql/bin/mysqld</string>
           <string>--user=_mysql</string>
            <string>--basedir=/usr/local/mysql</string>
            <string>--datadir=/usr/local/mysql/data</string>
            <string>--plugin-dir=/usr/local/mysql/lib/plugin</string>
            <string>--log-error=/usr/local/mysql/data/mysqld.local.err</string>
            <string>--pid-file=/usr/local/mysql/data/mysqld.local.pid</string>
           <string>--keyring-file-data=/usr/local/mysql/keyring/keyring</string>
            <string>--early-plugin-load=keyring_file=keyring_file.so</string>
   <key>WorkingDirectory</key> <string>/usr/local/mysql</string>
</dict>
</plist>
```

#### Note

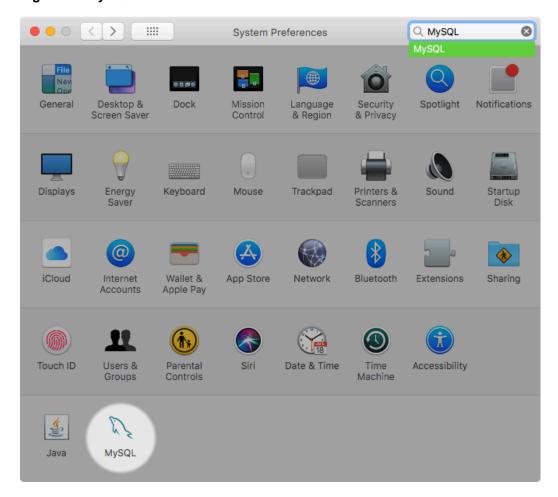
Some users report that adding a plist DOCTYPE declaration causes the launchd operation to fail, despite it passing the lint check. We suspect it's a copy-

n-paste error. The md5 checksum of a file containing the above snippet is d925f05f6d1b6ee5ce5451b596d6baed.

To enable the launchd service, you can either:

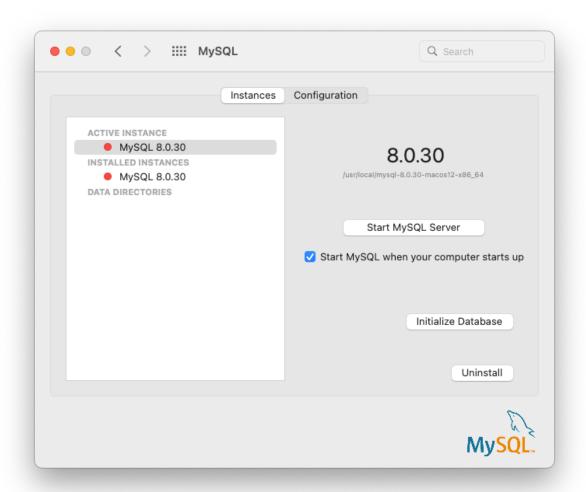
 Open macOS system preferences and select the MySQL preference panel, and then execute Start MySQL Server.

Figure 6.6 MySQL Preference Pane: Location



The **Instances** page includes an option to start or stop MySQL, and **Initialize Database** recreates the data/ directory. **Uninstall** uninstalls MySQL Server and optionally the MySQL preference panel and launchd information.

Figure 6.7 MySQL Preference Pane: Instances



• Or, manually load the launchd file.

```
$> cd /Library/LaunchDaemons
$> sudo launchctl load -F com.oracle.oss.mysql.mysqld.plist
```

• To configure MySQL to automatically start at bootup, you can:

```
$> sudo launchctl load -w com.oracle.oss.mysql.mysqld.plist
```

#### Note

When upgrading MySQL server, the launchd installation process removes the old startup items that were installed with MySQL server 5.7.7 and below.

Upgrading also replaces your existing launchd file named com.oracle.oss.mysql.mysqld.plist.

Additional launchd related information:

- The plist entries override my.cnf entries, because they are passed in as command line arguments. For additional information about passing in program options, see Specifying Program Options.
- The **ProgramArguments** section defines the command line options that are passed into the program, which is the mysqld binary in this case.
- The default plist definition is written with less sophisticated use cases in mind. For more complicated setups, you may want to remove some of the arguments and instead rely on a MySQL configuration file, such as my.cnf.
- If you edit the plist file, then uncheck the installer option when reinstalling or upgrading MySQL.
   Otherwise, your edited plist file is overwritten, and all edits are lost.

Because the default plist definition defines several **ProgramArguments**, you might remove most of these arguments and instead rely upon your my.cnf MySQL configuration file to define them. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.</pre>
<plist version="1.0">
<dict>
    <key>Label</key>
                                 <string>com.oracle.oss.mysql.mysqld</string>
   <key>ProcessType</key>
                                 <string>Interactive</string>
   <key>Disabled</key>
                                 <false/>
    <key>RunAtLoad</key>
                                 <true/>
   <key>KeepAlive</key>
                                 <true/>
   <key>SessionCreate</key>
                                 <true/>
   <key>LaunchOnlyOnce</key>
                                 <false/>
   <key>UserName</key>
<key>GroupName</key>
                                 <string>_mysql</string>
                                 <string>_mysql</string>
   <key>ExitTimeOut</key>
<key>Program</key>
                                 <integer>600</integer>
                                 <string>/usr/local/mysql/bin/mysqld</string>
    <key>ProgramArguments</key>
        <array>
            <string>/usr/local/mysql/bin/mysqld</string>
            <string>--user=_mysql</string>
            <string>--basedir=/usr/local/mysgl</string>
            <string>--datadir=/usr/local/mysql/data</string>
            <string>--plugin-dir=/usr/local/mysql/lib/plugin</string>
            <string>--log-error=/usr/local/mysql/data/mysqld.local.err</string>
            <string>--pid-file=/usr/local/mysql/data/mysqld.local.pid/string>
            <string>--keyring-file-data=/usr/local/mysql/keyring/keyring</string>
            <string>--early-plugin-load=keyring_file=keyring_file.so</string>
   <key>WorkingDirectory</key> <string>/usr/local/mysql</string>
</dict>
</plist>
```

In this case, the basedir, datadir, plugin\_dir, log\_error, pid\_file, keyring\_file\_data, and --early-plugin-load options were removed from the default plist *ProgramArguments* definition, which you might have defined in my.cnf instead.

# 6.4 Installing and Using the MySQL Preference Pane

The MySQL Installation Package includes a MySQL preference pane that enables you to start, stop, and control automated startup during boot of your MySQL installation.

This preference pane is installed by default, and is listed under your system's System Preferences window.

> :::: Q MySQL 8 System Preferences General Desktop & Dock Mission Language Security Spotlight Notifications Screen Saver Control & Region & Privacy Displays Energy Keyboard Mouse Trackpad Printers & Sound Saver Scanners Disk iCloud Wallet & App Store Bluetooth Internet Extensions Sharing Network Accounts Apple Pay Touch ID Users & Parental Date & Time Time Accessibility Groups Controls Java MySQL

Figure 6.8 MySQL Preference Pane: Location

The MySQL preference pane is installed with the same DMG file that installs MySQL Server. Typically it is installed with MySQL Server but it can be installed by itself too.

To install the MySQL preference pane:

- 1. Go through the process of installing the MySQL server, as described in the documentation at Section 6.2, "Installing MySQL on macOS Using Native Packages".
- 2. Click **Customize** at the **Installation Type** step. The "Preference Pane" option is listed there and enabled by default; make sure it is not deselected. The other options, such as MySQL Server, can be selected or deselected.

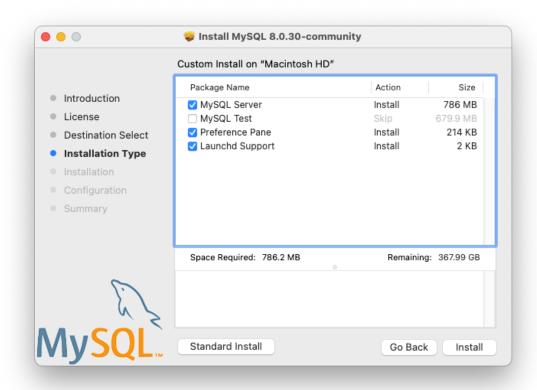


Figure 6.9 MySQL Package Installer Wizard: Customize

3. Complete the installation process.

#### Note

The MySQL preference pane only starts and stops MySQL installation installed from the MySQL package installation that have been installed in the default location.

Once the MySQL preference pane has been installed, you can control your MySQL server instance using this preference pane.

The **Instances** page includes an option to start or stop MySQL, and **Initialize Database** recreates the data/ directory. **Uninstall** uninstalls MySQL Server and optionally the MySQL preference panel and launchd information.

Figure 6.10 MySQL Preference Pane: Instances

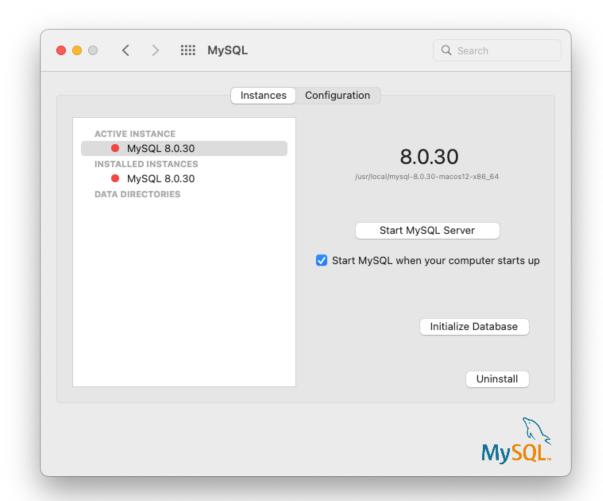
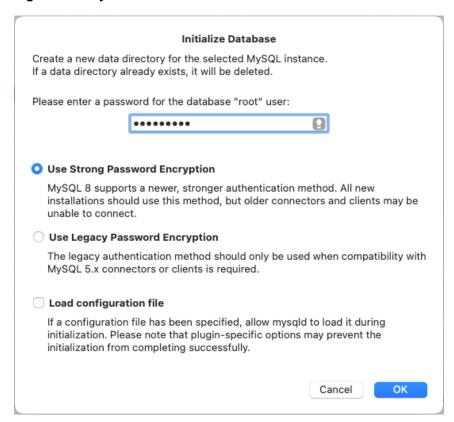
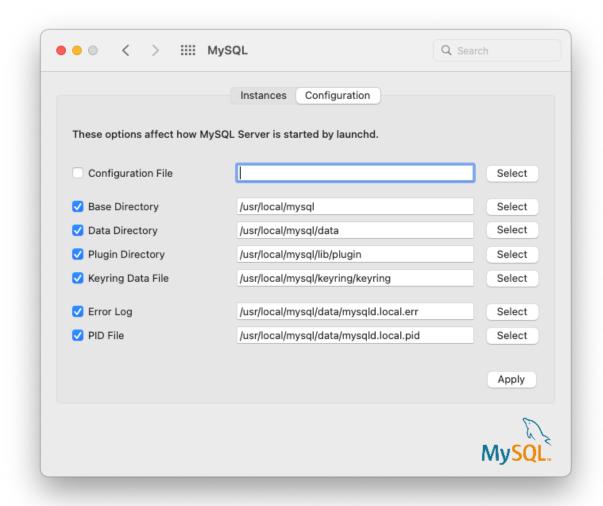


Figure 6.11 MySQL Preference Pane: Initialize Database



The **Configuration** page shows MySQL Server options including the path to the MySQL configuration file.

Figure 6.12 MySQL Preference Pane: Configuration



The MySQL Preference Pane shows the current status of the MySQL server, showing **stopped** (in red) if the server is not running and **running** (in green) if the server has already been started. The preference pane also shows the current setting for whether the MySQL server has been set to start automatically.

# Chapter 7 Installing MySQL on Linux

# **Table of Contents**

7.1 Installing MySQL on Linux Using the MySQL Yum Repository	. 142
7.2 Installing MySQL on Linux Using the MySQL APT Repository	146
7.3 Installing MySQL on Linux Using the MySQL SLES Repository	146
7.4 Installing MySQL on Linux Using RPM Packages from Oracle	. 147
7.5 Installing MySQL on Linux Using Debian Packages from Oracle	152
7.6 Deploying MySQL on Linux with Docker Containers	153
7.6.1 Basic Steps for MySQL Server Deployment with Docker	153
7.6.2 More Topics on Deploying MySQL Server with Docker	158
7.6.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker	165
7.7 Installing MySQL on Linux from the Native Software Repositories	166
7.8 Installing MySQL on Linux with Juju	168
7.9 Managing MySQL Server with systemd	168

Linux supports a number of different solutions for installing MySQL. We recommend that you use one of the distributions from Oracle, for which several methods for installation are available:

Table 7.1 Linux Installation Methods and Information

Туре	Setup Method	Additional Information
Apt	Enable the MySQL Apt repository	Documentation
Yum	Enable the MySQL Yum repository	Documentation
Zypper	Enable the MySQL SLES repository	Documentation
RPM	Download a specific package	Documentation
DEB	Download a specific package	Documentation
Generic	Download a generic package	Documentation
Source	Compile from source	Documentation
Docker	Use the Oracle Container Registry. You can also use My Oracle Support for the MySQL Enterprise Edition.	Documentation
Oracle Unbreakable Linux Network	Use ULN channels	Documentation

As an alternative, you can use the package manager on your system to automatically download and install MySQL with packages from the native software repositories of your Linux distribution. These native packages are often several versions behind the currently available release. You are also normally unable to install innovation releases, since these are not usually made available in the native repositories. For more information on using the native package installers, see Section 7.7, "Installing MySQL on Linux from the Native Software Repositories".

#### Note

For many Linux installations, you want to set up MySQL to be started automatically when your machine starts. Many of the native package installations perform this

operation for you, but for source, binary and RPM solutions you may need to set this up separately. The required script, <code>mysql.server</code>, can be found in the <code>support-files</code> directory under the MySQL installation directory or in a MySQL source tree. You can install it as <code>/etc/init.d/mysql</code> for automatic MySQL startup and shutdown. See <code>mysql.server</code> — <code>MySQL</code> Server Startup Script.

# 7.1 Installing MySQL on Linux Using the MySQL Yum Repository

The MySQL Yum repository for Oracle Linux, Red Hat Enterprise Linux, CentOS, and Fedora provides RPM packages for installing the MySQL server, client, MySQL Workbench, MySQL Utilities, MySQL Router, MySQL Shell, Connector/ODBC, Connector/Python and so on (not all packages are available for all the distributions; see Installing Additional MySQL Products and Components with Yum for details).

### **Before You Start**

As a popular, open-source software, MySQL, in its original or re-packaged form, is widely installed on many systems from various sources, including different software download sites, software repositories, and so on. The following instructions assume that MySQL is not already installed on your system using a third-party-distributed RPM package; if that is not the case, follow the instructions given in Section 10.8, "Upgrading MySQL with the MySQL Yum Repository" or Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository.

### **Important**

Repository setup RPM file names begin with mysql-84-lts-community to highlight the default active MySQL subrepository, which is MySQL 8.4 today. MySQL 8.0 must be manually enabled via your local repository configuration to install MySQL 8.0 instead of MySQL 8.4.

# Steps for a Fresh Installation of MySQL

Follow the steps below to install the latest GA version of MySQL with the MySQL Yum repository:

# Adding1the MySQL Yum Repository

First, add the MySQL Yum repository to your system's repository list. This is a one-time operation, which can be performed by installing an RPM provided by MySQL. Follow these steps:

- a. Go to the Download MySQL Yum Repository page (https://dev.mysql.com/downloads/repo/yum/) in the MySQL Developer Zone.
- b. Select and download the release package for your platform.
- c. Install the downloaded release package with the following command, replacing <code>platform-and-version-specific-package-name</code> with the name of the downloaded RPM package:

```
$> sudo yum install platform-and-version-specific-package-name.rpm
```

For an EL6-based system, the command is in the form of (note the mysql80 prefix instead of mysql84 because EL6-based systems do not support MySQL 8.4):

```
$> sudo yum install mysql80-community-release-el6-{version-number}.noarch.rpm
```

For an EL7-based system:

```
$> sudo yum install mysq184-community-release-e17-{version-number}.noarch.rpm
```

Fpr EL8 or later, change el7 to the version number of your Enterprise Linux.

For Fedora 41 and 42:

```
$> sudo dnf install mysql84-community-release-fcnn-{rpm-version-number}.noarch.rpm
```

Replace nn with the Fedora version and  $\{rpm-version-number\}$  with the rpm's version number. For example, for:

```
mysql84-community-release-fc42-1.noarch.rpm
```

The installation command adds the MySQL Yum repository to your system's repository list and downloads the GnuPG key to check the integrity of the software packages. See Section 2.4.2, "Signature Checking Using GnuPG" for details on GnuPG key checking.

You can check that the MySQL Yum repository has been successfully added by the following command (for dnf-enabled systems, replace yum in the command with dnf):

```
$> yum repolist enabled | grep "mysql.*-community.*"
```

#### Note

Once the MySQL Yum repository is enabled on your system, any system-wide update by the yum update command (or dnf upgrade for dnf-enabled systems) upgrades MySQL packages on your system and replaces any native third-party packages, if Yum finds replacements for them in the MySQL Yum repository; see Section 10.8, "Upgrading MySQL with the MySQL Yum Repository", for a discussion on some possible effects of that on your system, see Upgrading the Shared Client Libraries.

# Selecting a Release Series

When using the MySQL Yum repository, the latest LTS series (currently MySQL 8.4) is selected for installation by default. If you want to install MySQL 8.4 instead of 8.0 then skip this step.

Within the MySQL Yum repository, different release series of the MySQL Community Server are hosted in different subrepositories. The subrepository for the latest GA series (currently MySQL 8.4) is enabled by default, and the subrepositories for all other series (for example, the MySQL 8.0 series) are disabled by default. Use this command to see all the subrepositories in the MySQL Yum repository, and see which of them are enabled or disabled (for dnf-enabled systems, replace yum in the command with dnf):

```
$> yum repolist all | grep mysql
```

To install the latest release from the latest LTS series, no configuration is needed. To install the latest release from a specific series other than the latest LTS series, disable the subrepository for the latest LTS series and enable the subrepository for the specific series before running the installation command. If your platform supports yum-config-manager, you can do that by issuing these commands, which disable the subrepository for the 8.4 series and enable the one for the 8.0 series:

```
$> sudo yum-config-manager --disable mysql-8.4-lts-community
$> sudo yum-config-manager --disable mysql-tools-8.4-lts-community
$> sudo yum-config-manager --enable mysql80-community
```

```
$> sudo yum-config-manager --enable mysql-tools-community
```

For dnf-enabled platforms:

```
$> sudo dnf config-manager --disable mysql-8.4-lts-community
$> sudo dnf config-manager --disable mysql-tools-8.4-lts-community
$> sudo dnf config-manager --enable mysql80-community
$> sudo dnf config-manager --enable mysql-tools-community
```

Besides using yum-config-manager or the dnf config-manager command, you can also select a release series by editing manually the /etc/yum.repos.d/mysql-community.repo file. This is a typical entry for a MySQL 8.0 subrepository:

```
[mysq180-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/9/$basearch/
enabled=1
gpgcheck=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql-2023
```

Find the entry for the subrepository you want to configure, and edit the enabled option. Specify enabled=0 to disable a subrepository, or enabled=1 to enable a subrepository. For example, to install MySQL 8.0, make sure you have enabled=0 for the other MySQL series entries and enabled=1 for MySQL 8.0.

You should only enable subrepository for one release series at any time. When subrepositories for more than one release series are enabled, Yum uses the latest series.

Verify that the correct subrepositories have been enabled and disabled by running the following command and checking its output (for dnf-enabled systems, replace yum in the command with dnf):

```
$> yum repolist enabled | grep mysql
```

# Disabling the Default MySQL Module

(EL8 systems only) EL8-based systems such as RHEL8 and Oracle Linux 8 include a MySQL module that is enabled by default. Unless this module is disabled, it masks packages provided by MySQL repositories. To disable the included module and make the MySQL repository packages visible, use the following command (for dnf-enabled systems, replace yum in the command with dnf):

```
$> sudo yum module disable mysql
```

# Installing MySQL

Install MySQL by the following command (for dnf-enabled systems, replace yum in the command with dnf):

```
$> sudo yum install mysql-community-server
```

This installs the package for MySQL server (mysql-community-server) and also packages for the components required to run the server, including packages for the client (mysql-community-client), the common error messages and character sets for client and server (mysql-community-common), and the shared client libraries (mysql-community-libs).

# Starting the MySQL Server

Start the MySQL server with the following command:

```
$> systemctl start mysqld
```

You can check the status of the MySQL server with the following command:

```
$> systemctl status mysqld
```

If the operating system is systemd enabled, standard systemct1 (or alternatively, service with the arguments reversed) commands such as stop, start, status, and restart should be used to manage the MySQL server service. The mysqld service is enabled by default, and it starts at system reboot. See Section 7.9, "Managing MySQL Server with systemd" for additional information.

At the initial start up of the server, the following happens, given that the data directory of the server is empty:

- · The server is initialized.
- · SSL certificate and key files are generated in the data directory.
- validate\_password is installed and enabled.
- A superuser account 'root'@'localhost is created. A password for the superuser is set and stored
  in the error log file. To reveal it, use the following command:

```
$> sudo grep 'temporary password' /var/log/mysqld.log
```

Change the root password as soon as possible by logging in with the generated, temporary password and set a custom password for the superuser account:

```
$> mysql -uroot -p
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';
```

#### Note

validate\_password is installed by default. The default password policy implemented by validate\_password requires that passwords contain at least one uppercase letter, one lowercase letter, one digit, and one special character, and that the total password length is at least 8 characters.

For more information on the postinstallation procedures, see Chapter 9, Postinstallation Setup and Testing.

#### **Note**

Compatibility Information for EL7-based platforms: The following RPM packages from the native software repositories of the platforms are incompatible with the package from the MySQL Yum repository that installs the MySQL server. Once you have installed MySQL using the MySQL Yum repository, you cannot install these packages (and vice versa).

· akonadi-mysql

# Installing Additional MySQL Products and Components with Yum

You can use Yum to install and manage individual components of MySQL. Some of these components are hosted in sub-repositories of the MySQL Yum repository: for example, the MySQL Connectors are to

be found in the MySQL Connectors Community sub-repository, and the MySQL Workbench in MySQL Tools Community. You can use the following command to list the packages for all the MySQL components available for your platform from the MySQL Yum repository (for dnf-enabled systems, replace yum in the command with dnf):

```
$> sudo yum --disablerepo=\* --enablerepo='mysql*-community*' list available
```

Install any packages of your choice with the following command, replacing *package-name* with name of the package (for dnf-enabled systems, replace yum in the command with dnf):

```
$> sudo yum install package-name
```

For example, to install MySQL Workbench on Fedora:

```
$> sudo dnf install mysql-workbench-community
```

To install the shared client libraries (for dnf-enabled systems, replace yum in the command with dnf):

```
$> sudo yum install mysql-community-libs
```

# **Platform Specific Notes**

**ARM Support** 

ARM 64-bit (aarch64) is supported on Oracle Linux 7 and requires the Oracle Linux 7 Software Collections Repository (ol7\_software\_collections). For example, to install the server:

```
$> yum-config-manager --enable ol7_software_collections
$> yum install mysql-community-server
```

#### Note

ARM 64-bit (aarch64) is supported on Oracle Linux 7 as of MySQL 8.0.12.

### **Known Limitation**

The 8.0.12 release requires you to adjust the <code>libstdc++7</code> path by executing <code>ln - s /opt/oracle/oracle-armtoolset-1/root/usr/lib64 /usr/lib64/gcc7 after executing the yum install step.</code>

# **Updating MySQL with Yum**

Besides installation, you can also perform updates for MySQL products and components using the MySQL Yum repository. See Section 10.8, "Upgrading MySQL with the MySQL Yum Repository" for details.

# 7.2 Installing MySQL on Linux Using the MySQL APT Repository

The MySQL APT repository provides deb packages for installing and managing the MySQL server, client, and other components on the current Debian and Ubuntu releases.

Instructions for using the MySQL APT Repository are available in A Quick Guide to Using the MySQL APT Repository.

# 7.3 Installing MySQL on Linux Using the MySQL SLES Repository

The MySQL SLES repository provides RPM packages for installing and managing the MySQL server, client, and other components on SUSE Enterprise Linux Server.

Instructions for using the MySQL SLES repository are available in A Quick Guide to Using the MySQL SLES Repository.

# 7.4 Installing MySQL on Linux Using RPM Packages from Oracle

The recommended way to install MySQL on RPM-based Linux distributions is by using the RPM packages provided by Oracle. There are two sources for obtaining them, for the Community Edition of MySQL:

- From the MySQL software repositories:
  - The MySQL Yum repository (see Section 7.1, "Installing MySQL on Linux Using the MySQL Yum Repository" for details).
  - The MySQL SLES repository (see Section 7.3, "Installing MySQL on Linux Using the MySQL SLES Repository" for details).
- From the Download MySQL Community Server page in the MySQL Developer Zone.

### Note

RPM distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by Oracle in features, capabilities, and conventions (including communication setup), and that the installation instructions in this manual do not necessarily apply to them. The vendor's instructions should be consulted instead.

# MySQL RPM Packages

### Table 7.2 RPM Packages for MySQL Community Edition

Package Name	Summary
mysql-community-client	MySQL client applications and tools
mysql-community-client-plugins	Shared plugins for MySQL client applications
mysql-community-common	Common files for server and client libraries
mysql-community-devel	Development header files and libraries for MySQL database client applications
mysql-community-embedded-compat	MySQL server as an embedded library with compatibility for applications using version 18 of the library
mysql-community-icu-data-files	MySQL packaging of ICU data files needed by MySQL regular expressions
mysql-community-libs	Shared libraries for MySQL database client applications
mysql-community-libs-compat	Shared compatibility libraries for previous MySQL installations; only present if previous MySQL versions are supported by the platform
mysql-community-server	Database server and related tools
mysql-community-server-debug	Debug server and plugin binaries
mysql-community-test	Test suite for the MySQL server
mysql-community	The source code RPM looks similar to mysql-community-8.0.42-1.el7.src.rpm, depending on selected OS
Additional *debuginfo* RPMs	There are several debuginfo packages: mysql-community-client-debuginfo, mysql-community-

Package Name	Summary
	libs-debuginfo mysql-community-server-debug- debuginfo mysql-community-server-debuginfo, and mysql-community-test-debuginfo.

Table 7.3 RPM Packages for the MySQL Enterprise Edition

Package Name	Summary
mysql-commercial-backup	MySQL Enterprise Backup (added in 8.0.11)
mysql-commercial-client	MySQL client applications and tools
mysql-commercial-client-plugins	Shared plugins for MySQL client applications
mysql-commercial-common	Common files for server and client libraries
mysql-commercial-devel	Development header files and libraries for MySQL database client applications
mysql-commercial-embedded-compat	MySQL server as an embedded library with compatibility for applications using version 18 of the library
mysql-commercial-icu-data-files	MySQL packaging of ICU data files needed by MySQL regular expressions
mysql-commercial-libs	Shared libraries for MySQL database client applications
mysql-commercial-libs-compat	Shared compatibility libraries for previous MySQL installations; only present if previous MySQL versions are supported by the platform. The version of the libraries matches the version of the libraries installed by default by the distribution you are using.
mysql-commercial-server	Database server and related tools
mysql-commercial-test	Test suite for the MySQL server
Additional *debuginfo* RPMs	There are several debuginfo packages: mysql-commercial-client-debuginfo, mysql-commercial-libs-debuginfo mysql-commercial-server-debug-debuginfo mysql-commercial-server-debuginfo, and mysql-commercial-test-debuginfo.

The full names for the RPMs have the following syntax:

packagename-version-distribution-arch.rpm

The *distribution* and *arch* values indicate the Linux distribution and the processor type for which the package was built. See the table below for lists of the distribution identifiers:

Table 7.4 MySQL Linux RPM Package Distribution Identifiers

Distribution Value	Intended Use
el{version} where {version} is the major Enterprise Linux version, such as el8	EL6 (8.0), EL7, EL8, EL9, and EL10-based platforms (for example, the corresponding versions of Oracle Linux, Red Hat Enterprise Linux, and CentOS)
fc{version} where {version} is the major Fedora version, such as fc37	Fedora 41 and 42

Distribution Value	Intended Use
sles12	SUSE Linux Enterprise Server 12

To see all files in an RPM package (for example, mysql-community-server), use the following command:

```
$> rpm -qpl mysql-community-server-version-distribution-arch.rpm
```

The discussion in the rest of this section applies only to an installation process using the RPM packages directly downloaded from Oracle, instead of through a MySQL repository.

Dependency relationships exist among some of the packages. If you plan to install many of the packages, you may wish to download the RPM bundle tar file instead, which contains all the RPM packages listed above, so that you need not download them separately.

In most cases, you need to install the <code>mysql-community-server</code>, <code>mysql-community-client</code>, <code>mysql-community-client-plugins</code>, <code>mysql-community-libs</code>, <code>mysql-community-icu-data-files</code>, <code>mysql-community-community-community-community-libs-compat packages to get a functional, standard <code>MySQL</code> installation. To perform such a standard, basic installation, go to the folder that contains all those packages (and, preferably, no other RPM packages with similar names), and issue the following command:</code>

```
$> sudo yum install mysql-community-{server,client,client-plugins,icu-data-files,common,libs}-*
```

Replace yum with zypper for SLES, and with dnf for Fedora.

While it is much preferable to use a high-level package management tool like yum to install the packages, users who prefer direct rpm commands can replace the yum install command with the rpm -Uvh command; however, using rpm -Uvh instead makes the installation process more prone to failure, due to potential dependency issues the installation process might run into.

To install only the client programs, you can skip mysql-community-server in your list of packages to install; issue the following command:

```
$> sudo yum install mysql-community-{client,client-plugins,common,libs}-*
```

Replace yum with zypper for SLES, and with dnf for Fedora.

A standard installation of MySQL using the RPM packages result in files and resources created under the system directories, shown in the following table.

Table 7.5 MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone

Files or Resources	Location
Client programs and scripts	/usr/bin
mysqld server	/usr/sbin
Configuration file	/etc/my.cnf
Data directory	/var/lib/mysql
Error log file	For RHEL, Oracle Linux, CentOS or Fedora platforms: /var/log/mysqld.log
	For SLES: /var/log/mysql/mysqld.log
Value of secure_file_priv	/var/lib/mysql-files

Files or Resources	Location
System V init script	For RHEL, Oracle Linux, CentOS or Fedora platforms: /etc/init.d/mysqld
	For SLES: /etc/init.d/mysql
Systemd service	For RHEL, Oracle Linux, CentOS or Fedora platforms: mysqld
	For SLES: mysql
Pid file	/var/run/mysql/mysqld.pid
Socket	/var/lib/mysql/mysql.sock
Keyring directory	/var/lib/mysql-keyring
Unix manual pages	/usr/share/man
Include (header) files	/usr/include/mysql
Libraries	/usr/lib/mysql
Miscellaneous support files (for example, error messages, and character set files)	/usr/share/mysql

The installation also creates a user named mysql and a group named mysql on the system.

#### Notes

• The mysql user is created using the -r and -s /bin/false options of the useradd command, so that it does not have login permissions to your server host (see Creating the mysql User and Group for details). To switch to the mysql user on your OS, use the --shell=/bin/bash option for the su command:

```
su - mysql --shell=/bin/bash
```

• Installation of previous versions of MySQL using older packages might have created a configuration file named /usr/my.cnf. It is highly recommended that you examine the contents of the file and migrate the desired settings inside to the file /etc/my.cnf file, then remove /usr/my.cnf.

MySQL is NOT automatically started at the end of the installation process. For Red Hat Enterprise Linux, Oracle Linux, CentOS, and Fedora systems, use the following command to start MySQL:

```
$> systemctl start mysqld
```

For SLES systems, the command is the same, but the service name is different:

```
$> systemctl start mysql
```

If the operating system is systemd enabled, standard <code>systemctl</code> (or alternatively, <code>service</code> with the arguments reversed) commands such as <code>stop</code>, <code>start</code>, <code>status</code>, and <code>restart</code> should be used to manage the MySQL server service. The <code>mysqld</code> service is enabled by default, and it starts at system reboot. Notice that certain things might work differently on systemd platforms: for example, changing the location of the data directory might cause issues. See Section 7.9, "Managing MySQL Server with systemd" for additional information.

During an upgrade installation using RPM and DEB packages, if the MySQL server is running when the upgrade occurs then the MySQL server is stopped, the upgrade occurs, and the MySQL server is restarted. One exception: if the edition also changes during an upgrade (such as community to commercial, or vice-versa), then MySQL server is not restarted.

At the initial start up of the server, the following happens, given that the data directory of the server is empty:

- · The server is initialized.
- An SSL certificate and key files are generated in the data directory.
- validate\_password is installed and enabled.
- A superuser account 'root'@'localhost' is created. A password for the superuser is set and stored
  in the error log file. To reveal it, use the following command for RHEL, Oracle Linux, CentOS, and
  Fedora systems:

```
$> sudo grep 'temporary password' /var/log/mysqld.log
```

Use the following command for SLES systems:

```
$> sudo grep 'temporary password' /var/log/mysql/mysqld.log
```

The next step is to log in with the generated, temporary password and set a custom password for the superuser account:

```
$> mysql -uroot -p
```

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';

#### Note

validate\_password is installed by default. The default password policy implemented by validate\_password requires that passwords contain at least one uppercase letter, one lowercase letter, one digit, and one special character, and that the total password length is at least 8 characters.

If something goes wrong during installation, you might find debug information in the error log file /var/log/mysqld.log.

For some Linux distributions, it might be necessary to increase the limit on number of file descriptors available to mysqld. See File Not Found and Similar Errors

Installing Client Libraries from Multiple MySQL Versions. It is possible to install multiple client library versions, such as for the case that you want to maintain compatibility with older applications linked against previous libraries. To install an older client library, use the --oldpackage option with rpm. For example, to install mysql-community-libs-5.5 on an EL6 system that has libmysqlclient.21 from MySQL 8.0. use a command like this:

```
$> rpm --oldpackage -ivh mysql-community-libs-5.5.50-2.el6.x86_64.rpm
```

**Debug Package.** A special variant of MySQL Server compiled with the debug package has been included in the server RPM packages. It performs debugging and memory allocation checks and produces a trace file when the server is running. To use that debug version, start MySQL with /usr/sbin/mysqld-debug, instead of starting it as a service or with /usr/sbin/mysqld. See The DBUG Package for the debug options you can use.

#### Note

The default plugin directory for debug builds changed from /usr/lib64/mysql/plugin to /usr/lib64/mysql/plugin/debug in MySQL 8.0.4. Previously, it was necessary to change plugin\_dir to /usr/lib64/mysql/plugin/debug for debug builds.

**Rebuilding RPMs from source SRPMs.** Source code SRPM packages for MySQL are available for download. They can be used as-is to rebuild the MySQL RPMs with the standard rpmbuild tool chain.

# 7.5 Installing MySQL on Linux Using Debian Packages from Oracle

Oracle provides Debian packages for installing MySQL on Debian or Debian-like Linux systems. The packages are available through two different channels:

- The MySQL APT Repository. This is the preferred method for installing MySQL on Debian-like systems, as it provides a simple and convenient way to install and update MySQL products. For details, see Section 7.2, "Installing MySQL on Linux Using the MySQL APT Repository".
- The MySQL Developer Zone's Download Area. For details, see Section 2.3, "How to Get MySQL". The following are some information on the Debian packages available there and the instructions for installing them:
  - Various Debian packages are provided in the MySQL Developer Zone for installing different
    components of MySQL on the current Debian and Ubuntu platforms. The preferred method is to use
    the tarball bundle, which contains the packages needed for a basic setup of MySQL. The tarball
    bundles have names in the format of mysql-server\_MVER-DVER\_CPU.deb-bundle.tar.MVER is
    the MySQL version and DVER is the Linux distribution version. The CPU value indicates the processor
    type or family for which the package is built, as shown in the following table:

Table 7.6 MySQL Debian and Ubuntu Installation Packages CPU Identifiers

CPU Value	Intended Processor Type or Family
i386	Pentium processor or better, 32 bit
amd64	64-bit x86 processor

• After downloading the tarball, unpack it with the following command:

\$> tar -xvf mysql-server\_MVER-DVER\_CPU.deb-bundle.tar

You may need to install the libaio library if it is not already present on your system:

\$> sudo apt-get install libaio1

• Preconfigure the MySQL server package with the following command:

\$> sudo dpkg-preconfigure mysql-community-server\_\*.deb

You are asked to provide a password for the root user for your MySQL installation. You might also be asked other questions regarding the installation.

### **Important**

Make sure you remember the root password you set. Users who want to set a password later can leave the **password** field blank in the dialogue box and just press **OK**; in that case, root access to the server is authenticated using the MySQL Socket Peer-Credential Authentication Plugin for connections using a Unix socket file. You can set the root password later using mysql\_secure\_installation.

For a basic installation of the MySQL server, install the database common files package, the client
package, the client metapackage, the server package, and the server metapackage (in that order); you
can do that with a single command:

\$> sudo dpkg -i mysql-{common,community-client-plugins,community-client-core,community-client,client,c

There are also packages with server-core and client-core in the package names. These contain binaries only and are installed automatically by the standard packages. Installing them by themselves does not result in a functioning MySQL setup.

If you are being warned of unmet dependencies by dpkg (such as libmecab2), you can fix them using apt-get:

sudo apt-get -f install

Here are where the files are installed on the system:

- All configuration files (like my.cnf) are under /etc/mysql
- All binaries, libraries, headers, etc., are under /usr/bin and /usr/sbin
- The data directory is under /var/lib/mysql

#### Note

Debian distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by Oracle in features, capabilities, and conventions (including communication setup), and that the instructions in this manual do not necessarily apply to installing them. The vendor's instructions should be consulted instead.

# 7.6 Deploying MySQL on Linux with Docker Containers

This section explains how to deploy MySQL Server using Docker containers.

While the docker client is used in the following instructions for demonstration purposes, in general, the MySQL container images provided by Oracle work with any container tools that are compliant with the OCI 1.0 specification.

### Warning

Before deploying MySQL with Docker containers, make sure you understand the security risks of running containers and mitigate them properly.

# 7.6.1 Basic Steps for MySQL Server Deployment with Docker

### Warning

The MySQL Docker images maintained by the MySQL team are built specifically for Linux platforms. Other platforms are not supported, and users using these MySQL Docker images on them are doing so at their own risk. See the discussion here for some known limitations for running these containers on non-Linux operating systems.

- Downloading a MySQL Server Docker Image
- Starting a MySQL Server Instance
- Connecting to MySQL Server from within the Container
- Container Shell Access

- · Stopping and Deleting a MySQL Container
- Upgrading a MySQL Server Container
- More Topics on Deploying MySQL Server with Docker

### **Downloading a MySQL Server Docker Image**

### **Important**

For users of MySQL Enterprise Edition: A subscription is required to use the Docker images for MySQL Enterprise Edition. Subscriptions work by a Bring Your Own License model; see How to Buy MySQL Products and Services for details.

Downloading the server image in a separate step is not strictly necessary; however, performing this step before you create your Docker container ensures your local image is up to date. To download the MySQL Community Edition image from the Oracle Container Registry (OCR), run this command:

```
docker pull container-registry.oracle.com/mysql/community-server: tag
```

The *tag* is the label for the image version you want to pull (for example, 5.7, 8.0, or latest). If *tag* is omitted, the latest label is used, and the image for the latest GA version of MySQL Community Server is downloaded.

To download the MySQL Enterprise Edition image from the OCR, you need to first accept the license agreement on the OCR and log in to the container repository with your Docker client. Follow these steps:

- Visit the OCR at https://container-registry.oracle.com/ and choose MySQL.
- Under the list of MySQL repositories, choose enterprise-server.
- If you have not signed in to the OCR yet, click the **Sign in** button on the right of the page, and then enter your Oracle account credentials when prompted to.
- Follow the instructions on the right of the page to accept the license agreement.
- Log in to the OCR with your container client using, for example, the docker login command:

```
# docker login container-registry.oracle.com
Username: Oracle-Account-ID
Password: password
Login successful.
```

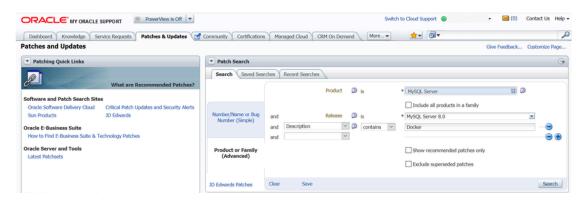
Download the Docker image for MySQL Enterprise Edition from the OCR with this command:

```
docker pull container-registry.oracle.com/mysql/enterprise-server:tag
```

To download the MySQL Enterprise Edition image from My Oracle Support website, go onto the website, sign in to your Oracle account, and perform these steps once you are on the landing page:

- Select the Patches and Updates tab.
- Go to the **Patch Search** region and, on the **Search** tab, switch to the **Product or Family (Advanced)** subtab.
- Enter "MySQL Server" for the **Product** field, and the desired version number in the **Release** field.
- Use the dropdowns for additional filters to select **Description—contains**, and enter "Docker" in the text field.

The following figure shows the search settings for the MySQL Enterprise Edition image for MySQL Server 8.0:



- Click the Search button and, from the result list, select the version you want, and click the Download button.
- In the File Download dialogue box that appears, click and download the .zip file for the Docker image.

Unzip the downloaded .zip archive to obtain the tarball inside (mysql-enterprise-server-version.tar), and then load the image by running this command:

```
docker load -i mysql-enterprise-server-version.tar
```

You can list downloaded Docker images with this command:

```
$> docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

container-registry.oracle.com/mysql/community-server latest 1d9c2219ff69 2 months ago 496MB
```

## Starting a MySQL Server Instance

To start a new Docker container for a MySQL Server, use the following command:

```
docker run --name=container_name --restart on-failure -d image_name:tag
```

image\_name is the name of the image to be used to start the container; see Downloading a MySQL Server Docker Image for more information.

The --name option, for supplying a custom name for your server container, is optional; if no container name is supplied, a random one is generated.

The --restart option is for configuring the restart policy for your container; it should be set to the value on-failure, to enable support for server restart within a client session (which happens, for example, when the RESTART statement is executed by a client or during the configuration of an InnoDB cluster instance). With the support for restart enabled, issuing a restart within a client session causes the server and the container to stop and then restart. Support for server restart is available for MySQL 8.0.21 and later.

For example, to start a new Docker container for the MySQL Community Server, use this command:

```
docker run --name=mysql1 --restart on-failure -d container-registry.oracle.com/mysql/community-server:late
```

To start a new Docker container for the MySQL Enterprise Server with a Docker image downloaded from the OCR, use this command:

```
docker run --name=mysql1 --restart on-failure -d container-registry.oracle.com/mysql/enterprise-server:lat
```

To start a new Docker container for the MySQL Enterprise Server with a Docker image downloaded from My Oracle Support, use this command:

```
docker run --name=mysql1 --restart on-failure -d mysql/enterprise-server:latest
```

If the Docker image of the specified name and tag has not been downloaded by an earlier docker pull or docker run command, the image is now downloaded. Initialization for the container begins, and the container appears in the list of running containers when you run the docker ps command. For example:

CREATED 8 second

The container initialization might take some time. When the server is ready for use, the STATUS of the container in the output of the docker ps command changes from (health: starting) to (healthy).

The -d option used in the docker run command above makes the container run in the background. Use this command to monitor the output from the container:

```
docker logs mysql1
```

Once initialization is finished, the command's output is going to contain the random password generated for the root user; check the password with, for example, this command:

```
$> docker logs mysql1 2>&1 | grep GENERATED
GENERATED ROOT PASSWORD: Axegh3kAJyDLaRuBemecis&EShOs
```

### Connecting to MySQL Server from within the Container

Once the server is ready, you can run the <code>mysql</code> client within the MySQL Server container you just started, and connect it to the MySQL Server. Use the <code>docker exec -it</code> command to start a <code>mysql</code> client inside the Docker container you have started, like the following:

```
docker exec -it mysql1 mysql -uroot -p
```

When asked, enter the generated root password (see the last step in Starting a MySQL Server Instance above on how to find the password). Because the MYSQL\_ONETIME\_PASSWORD option is true by default, after you have connected a mysql client to the server, you must reset the server root password by issuing this statement:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';
```

Substitute *password* with the password of your choice. Once the password is reset, the server is ready for use.

### **Container Shell Access**

To have shell access to your MySQL Server container, use the docker exec -it command to start a bash shell inside the container:

```
$> docker exec -it mysql1 bash
bash-4.2#
```

You can then run Linux commands inside the container. For example, to view contents in the server's data directory inside the container, use this command:

```
bash-4.2# ls /var/lib/mysql
auto.cnf ca.pem client-key.pem ib_logfile0 ibdatal mysql mysql.sock.lock private_key.pem
ca-key.pem client-cert.pem ib_buffer_pool ib_logfile1 ibtmpl mysql.sock performance_schema public_key.
```

## **Stopping and Deleting a MySQL Container**

To stop the MySQL Server container we have created, use this command:

```
docker stop mysql1
```

docker stop sends a SIGTERM signal to the mysqld process, so that the server is shut down gracefully.

Also notice that when the main process of a container (mysqld in the case of a MySQL Server container) is stopped, the Docker container stops automatically.

To start the MySQL Server container again:

```
docker start mysql1
```

To stop and start again the MySQL Server container with a single command:

```
docker restart mysql1
```

To delete the MySQL container, stop it first, and then use the docker rm command:

```
docker stop mysql1
docker rm mysql1
```

If you want the Docker volume for the server's data directory to be deleted at the same time, add the -v option to the docker rm command.

### **Upgrading a MySQL Server Container**

### **Important**

- Before performing any upgrade to MySQL, follow carefully the instructions in Chapter 10, *Upgrading MySQL*. Among other instructions discussed there, it is especially important to back up your database before the upgrade.
- The instructions in this section require that the server's data and configuration have been persisted on the host. See Persisting Data and Configuration Changes for details.

Follow these steps to upgrade a Docker installation of MySQL 5.7 to 8.0:

• Stop the MySQL 5.7 server (container name is mysql57 in this example):

```
docker stop mysq157
```

- Download the MySQL 8.0 Server Docker image. See instructions in Downloading a MySQL Server Docker Image. Make sure you use the right tag for MySQL 8.0.
- Start a new MySQL 8.0 Docker container (named mysql80 in this example) with the old server data and configuration (with proper modifications if needed—see Chapter 10, *Upgrading MySQL*) that have been persisted on the host (by bind-mounting in this example). For the MySQL Community Server, run this command:

```
docker run --name=mysq180 \
   --mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
   --mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
   -d container-registry.oracle.com/mysql/community-server:8.0
```

If needed, adjust container-registry.oracle.com/mysql/community-server to the correct image name—for example, replace it with container-registry.oracle.com/mysql/enterprise-server for MySQL Enterprise Edition images downloaded from the OCR, or mysql/enterprise-server for MySQL Enterprise Edition images downloaded from My Oracle Support.

 Wait for the server to finish startup. You can check the status of the server using the docker ps command (see Starting a MySQL Server Instance for how to do that). Follow the same steps for upgrading within the 8.0 series (that is, from release 8.0.x to 8.0.y): stop the original container, and start a new one with a newer image on the old server data and configuration. If you used the 8.0 or the latest tag when starting your original container and there is now a new MySQL 8.0 release you want to upgrade to it, you must first pull the image for the new release with the command:

```
docker pull container-registry.oracle.com/mysql/community-server:8.0
```

You can then upgrade by starting a *new* container with the same tag on the old data and configuration (adjust the image name if you are using the MySQL Enterprise Edition; see Downloading a MySQL Server Docker Image):

```
docker run --name=mysq180new \
    --mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
    --mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
    -d container-registry.oracle.com/mysql/community-server:8.0
```

#### Note

For MySQL 8.0.15 and earlier: You need to complete the upgrade process by running the mysql\_upgrade utility in the MySQL 8.0 Server container (the step is not required for MySQL 8.0.16 and later):

```
• docker exec -it mysql80 mysql_upgrade -uroot -p

When prompted, enter the root password for your old server.
```

Finish the upgrade by restarting the new container:

```
docker restart mysq180
```

## More Topics on Deploying MySQL Server with Docker

For more topics on deploying MySQL Server with Docker like server configuration, persisting data and configuration, server error log, and container environment variables, see Section 7.6.2, "More Topics on Deploying MySQL Server with Docker".

# 7.6.2 More Topics on Deploying MySQL Server with Docker

#### Note

Most of the following sample commands have container-registry.oracle.com/mysql/community-server as the Docker image being used (like with the docker pull and docker run commands); change that if your image is from another repository—for example, replace it with container-registry.oracle.com/mysql/enterprise-server for MySQL Enterprise Edition images downloaded from the Oracle Container Registry (OCR), or mysql/enterprise-server for MySQL Enterprise Edition images downloaded from My Oracle Support.

- The Optimized MySQL Installation for Docker
- Configuring the MySQL Server
- Persisting Data and Configuration Changes
- Running Additional Initialization Scripts
- Connect to MySQL from an Application in Another Docker Container
- Server Error Log

- · Using MySQL Enterprise Backup with Docker
- · Using mysqldump with Docker
- Known Issues
- Docker Environment Variables

### The Optimized MySQL Installation for Docker

Docker images for MySQL are optimized for code size, which means they only include crucial components that are expected to be relevant for the majority of users who run MySQL instances in Docker containers. A MySQL Docker installation is different from a common, non-Docker installation in the following aspects:

- Only a limited number of binaries are included.
- All binaries are stripped; they contain no debug information.

### Warning

Any software updates or installations users perform to the Docker container (including those for MySQL components) may conflict with the optimized MySQL installation created by the Docker image. Oracle does not provide support for MySQL products running in such an altered container, or a container created from an altered Docker image.

### Configuring the MySQL Server

When you start the MySQL Docker container, you can pass configuration options to the server through the docker run command. For example:

```
docker run --name mysql1 -d container-registry.oracle.com/mysql/community-server:tag --character-set-serve
```

The command starts the MySQL Server with utf8mb4 as the default character set and utf8mb4\_col as the default collation for databases.

Another way to configure the MySQL Server is to prepare a configuration file and mount it at the location of the server configuration file inside the container. See Persisting Data and Configuration Changes for details.

# **Persisting Data and Configuration Changes**

Docker containers are in principle ephemeral, and any data or configuration are expected to be lost if the container is deleted or corrupted (see discussions here). Docker volumes provides a mechanism to persist data created inside a Docker container. At its initialization, the MySQL Server container creates a Docker volume for the server data directory. The JSON output from the docker inspect command on the container includes a Mount key, whose value provides information on the data directory volume:

```
],
...
```

The output shows that the source directory /var/lib/docker/volumes/4f2d463cfc4bdd4baebcb098c97d7da3337195ed2c6572bc0b89f7e845d27652/\_data, in which data is persisted on the host, has been mounted at /var/lib/mysql, the server data directory inside the container.

Another way to preserve data is to bind-mount a host directory using the --mount option when creating the container. The same technique can be used to persist the configuration of the server. The following command creates a MySQL Server container and bind-mounts both the data directory and the server configuration file:

```
docker run --name=mysq11 \
--mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
--mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
-d container-registry.oracle.com/mysql/community-server:tag
```

The command mounts <code>path-on-host-machine/my.cnf</code> at <code>/etc/my.cnf</code> (the server configuration file inside the container), and <code>path-on-host-machine/datadir</code> at <code>/var/lib/mysql</code> (the data directory inside the container). The following conditions must be met for the bind-mounting to work:

• The configuration file path-on-host-machine/my.cnf must already exist, and it must contain the specification for starting the server by the user mysql:

```
[mysqld]
user=mysql
```

You can also include other server configuration options in the file.

• The data directory path-on-host-machine/datadir must already exist. For server initialization
to happen, the directory must be empty. You can also mount a directory prepopulated with data and
start the server with it; however, you must make sure you start the Docker container with the same
configuration as the server that created the data, and any host files or directories required are mounted
when starting the container.

# **Running Additional Initialization Scripts**

If there are any .sh or .sql scripts you want to run on the database immediately after it has been created, you can put them into a host directory and then mount the directory at /docker-entrypoint-initdb.d/ inside the container. For example:

```
docker run --name=mysql1 \
--mount type=bind,src=/path-on-host-machine/scripts/,dst=/docker-entrypoint-initdb.d/ \
-d container-registry.oracle.com/mysql/community-server:tag
```

# Connect to MySQL from an Application in Another Docker Container

By setting up a Docker network, you can allow multiple Docker containers to communicate with each other, so that a client application in another Docker container can access the MySQL Server in the server container. First, create a Docker network:

```
docker network create my-custom-net
```

Then, when you are creating and starting the server and the client containers, use the --network option to put them on network you created. For example:

```
docker run --name=mysql1 --network=my-custom-net -d container-registry.oracle.com/mysql/community-server docker run --name=myapp1 --network=my-custom-net -d myapp
```

The myapp1 container can then connect to the mysql1 container with the mysql1 hostname and vice versa, as Docker automatically sets up a DNS for the given container names. In the following example, we run the mysql client from inside the myapp1 container to connect to host mysql1 in its own container:

```
docker exec -it myapp1 mysql --host=mysqll --user=myuser --password
```

For other networking techniques for containers, see the Docker container networking section in the Docker Documentation.

# **Server Error Log**

When the MySQL Server is first started with your server container, a server error log is NOT generated if either of the following conditions is true:

- A server configuration file from the host has been mounted, but the file does not contain the system
  variable log\_error (see Persisting Data and Configuration Changes on bind-mounting a server
  configuration file).
- A server configuration file from the host has not been mounted, but the Docker environment variable MYSQL\_LOG\_CONSOLE is true (which is the variable's default state for MySQL 8.0 server containers). The MySQL Server's error log is then redirected to stderr, so that the error log goes into the Docker container's log and is viewable using the docker logs mysqld-container command.

To make MySQL Server generate an error log when either of the two conditions is true, use the --log-error option to configure the server to generate the error log at a specific location inside the container. To persist the error log, mount a host file at the location of the error log inside the container as explained in Persisting Data and Configuration Changes. However, you must make sure your MySQL Server inside its container has write access to the mounted host file.

# **Using MySQL Enterprise Backup with Docker**

MySQL Enterprise Backup is a commercially-licensed backup utility for MySQL Server, available with MySQL Enterprise Edition. MySQL Enterprise Backup is included in the Docker installation of MySQL Enterprise Edition.

In the following example, we assume that you already have a MySQL Server running in a Docker container (see Section 7.6.1, "Basic Steps for MySQL Server Deployment with Docker" on how to start a MySQL Server instance with Docker). For MySQL Enterprise Backup to back up the MySQL Server, it must have access to the server's data directory. This can be achieved by, for example, bind-mounting a host directory on the data directory of the MySQL Server when you start the server:

```
docker run --name=mysqlserver \
--mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
-d mysql/enterprise-server:8.0
```

With this command, the MySQL Server is started with a Docker image of the MySQL Enterprise Edition, and the host directory /path-on-host-machine/datadir/ has been mounted onto the server's data directory (/var/lib/mysql) inside the server container. We also assume that, after the server has been started, the required privileges have also been set up for MySQL Enterprise Backup to access the server (see Grant MySQL Privileges to Backup Administrator, for details). Use the following steps to back up and restore a MySQL Server instance.

To back up a MySQL Server instance running in a Docker container using MySQL Enterprise Backup with Docker, follow the steps listed here:

 On the same host where the MySQL Server container is running, start another container with an image of MySQL Enterprise Edition to perform a back up with the MySQL Enterprise Backup command backup-to-image. Provide access to the server's data directory using the bind mount we created in the last step. Also, mount a host directory (/path-on-host-machine/backups/ in this example) onto the storage folder for backups in the container (/data/backups in the example) to persist the backups we are creating. Here is a sample command for this step, in which MySQL Enterprise Backup is started with a Docker image downloaded from My Oracle Support:

```
$> docker run \
--mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
--mount type=bind,src=/path-on-host-machine/backups/,dst=/data/backups \
--rm mysql/enterprise-server:8.0 \
mysqlbackup -umysqlbackup -ppassword --backup-dir=/tmp/backup-tmp --with-timestamp \
--backup-image=/data/backups/db.mbi backup-to-image
[Entrypoint] MySQL Docker Image 8.0.11-1.1.5
MySQL Enterprise Backup version 8.0.11 Linux-4.1.12-61.1.16.el7uek.x86_64-x86_64 [2018-04-08 07:06:45]
Copyright (c) 2003, 2018, Oracle and/or its affiliates. All Rights Reserved.
180921 17:27:25 MAIN INFO: A thread created with Id '140594390935680'
180921 17:27:25 MAIN INFO: Starting with following command line ...
  Parameters Summary
                   : 29615616
  Start LSN
  End LSN
                            : 29651854
mysqlbackup completed OK!
```

It is important to check the end of the output by mysqlbackup to make sure the backup has been completed successfully.

2. The container exits once the backup job is finished and, with the --rm option used to start it, it is removed after it exits. An image backup has been created, and can be found in the host directory mounted in the last step for storing backups, as shown here:

```
$> ls /tmp/backups
db.mbi
```

To restore a MySQL Server instance in a Docker container using MySQL Enterprise Backup with Docker, follow the steps listed here:

1. Stop the MySQL Server container, which also stops the MySQL Server running inside:

```
docker stop mysqlserver
```

2. On the host, delete all contents in the bind mount for the MySQL Server data directory:

```
rm -rf /path-on-host-machine/datadir/*
```

3. Start a container with an image of MySQL Enterprise Edition to perform the restore with the MySQL Enterprise Backup command copy-back-and-apply-log. Bind-mount the server's data directory and the storage folder for the backups, like what we did when we backed up the server:

```
lsn 29680612.

180921 22:06:52 PCR1 INFO: Last MySQL binlog file position 0 155, file name binlog.000003

180921 22:06:52 PCR1 INFO: The first data file is '/var/lib/mysql/ibdatal' and the new created log files are at '/var/lib/mysql'

180921 22:06:52 MAIN INFO: No Keyring file to process.

180921 22:06:52 MAIN INFO: Apply-log operation completed successfully.

180921 22:06:52 MAIN INFO: Full Backup has been restored successfully.

180921 22:06:52 MAIN INFO: Full Backup has been restored successfully.
```

The container exits once the backup job is finished and, with the --rm option used when starting it, it is removed after it exits.

4. Restart the server container, which also restarts the restored server, using the following command:

```
docker restart mysqlserver
```

Or, start a new MySQL Server on the restored data directory, as shown here:

```
docker run --name=mysqlserver2 \
--mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
-d mysql/enterprise-server:8.0
```

Log on to the server to check that the server is running with the restored data.

### Using mysqldump with Docker

Besides using MySQL Enterprise Backup to back up a MySQL Server running in a Docker container, you can perform a logical backup of your server by using the mysqldump utility, run inside a Docker container.

The following instructions assume that you already have a MySQL Server running in a Docker container and, when the container was first started, a host directory /path-on-host-machine/datadir/ has been mounted onto the server's data directory /var/lib/mysql (see bind-mounting a host directory on the data directory of the MySQL Server for details), which contains the Unix socket file by which mysqldump and mysql can connect to the server. We also assume that, after the server has been started, a user with the proper privileges (admin in this example) has been created, with which mysqldump can access the server. Use the following steps to back up and restore MySQL Server data:

Backing up MySQL Server data using mysqldump with Docker.

1. On the same host where the MySQL Server container is running, start another container with an image of MySQL Server to perform a backup with the mysqldump utility (see documentation of the utility for its functionality, options, and limitations). Provide access to the server's data directory by bind mounting /path-on-host-machine/datadir/. Also, mount a host directory (/path-on-host-machine/backups/ in this example) onto a storage folder for backups inside the container (/data/backups is used in this example) to persist the backups you are creating. Here is a sample command for backing up all databases on the server using this setup:

```
$> docker run --entrypoint "/bin/sh" \
--mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
--mount type=bind,src=/path-on-host-machine/backups/,dst=/data/backups \
--rm container-registry.oracle.com/mysql/community-server:8.0 \
-c "mysqldump -uadmin --password='password' --all-databases > /data/backups/all-databases.sql"
```

In the command, the --entrypoint option is used so that the system shell is invoked after the container is started, and the -c option is used to specify the mysqldump command to be run in the shell, whose output is redirected to the file all-databases.sql in the backup directory.

2. The container exits once the backup job is finished and, with the --rm option used to start it, it is removed after it exits. A logical backup been created, and can be found in the host directory mounted for storing the backup, as shown here:

```
$> ls /path-on-host-machine/backups/
all-databases.sql
```

Restoring MySQL Server data using mysqldump with Docker.

- Make sure you have a MySQL Server running in a container, onto which you want your backed-up data to be restored.
- 2. Start a container with an image of MySQL Server to perform the restore with a mysql client. Bindmount the server's data directory, as well as the storage folder that contains your backup:

```
$> docker run \
--mount type=bind,src=/path-on-host-machine/datadir/,dst=/var/lib/mysql \
--mount type=bind,src=/path-on-host-machine/backups/,dst=/data/backups \
--rm container-registry.oracle.com/mysql/community-server:8.0 \
mysql -uadmin --password='password' -e "source /data/backups/all-databases.sql"
```

The container exits once the backup job is finished and, with the --rm option used when starting it, it is removed after it exits.

3. Log on to the server to check that the restored data is now on the server.

### **Known Issues**

• When using the server system variable audit\_log\_file to configure the audit log file name, use the loose option modifier with it; otherwise, Docker cannot start the server.

### **Docker Environment Variables**

When you create a MySQL Server container, you can configure the MySQL instance by using the -- env option (short form -e) and specifying one or more environment variables. No server initialization is performed if the mounted data directory is not empty, in which case setting any of these variables has no effect (see Persisting Data and Configuration Changes), and no existing contents of the directory, including server settings, are modified during container startup.

Environment variables which can be used to configure a MySQL instance are listed here:

- The boolean variables including MYSQL\_RANDOM\_ROOT\_PASSWORD, MYSQL\_ONETIME\_PASSWORD, MYSQL\_ALLOW\_EMPTY\_PASSWORD, and MYSQL\_LOG\_CONSOLE are made true by setting them with any strings of nonzero lengths. Therefore, setting them to, for example, "0", "false", or "no" does not make them false, but actually makes them true. This is a known issue.
- MYSQL\_RANDOM\_ROOT\_PASSWORD: When this variable is true (which is its default state, unless MYSQL\_ROOT\_PASSWORD is set or MYSQL\_ALLOW\_EMPTY\_PASSWORD is set to true), a random password for the server's root user is generated when the Docker container is started. The password is printed to stdout of the container and can be found by looking at the container's log (see Starting a MySQL Server Instance).
- MYSQL\_ONETIME\_PASSWORD: When the variable is true (which is its default state, unless MYSQL\_ROOT\_PASSWORD is set or MYSQL\_ALLOW\_EMPTY\_PASSWORD is set to true), the root user's password is set as expired and must be changed before MySQL can be used normally.
- MYSQL\_DATABASE: This variable allows you to specify the name of a database to be created on image startup. If a user name and a password are supplied with MYSQL\_USER and MYSQL\_PASSWORD, the user is created and granted superuser access to this database (corresponding to GRANT ALL). The specified database is created by a CREATE DATABASE IF NOT EXIST statement, so that the variable has no effect if the database already exists.

MYSQL\_USER, MYSQL\_PASSWORD: These variables are used in conjunction to create a user and set
that user's password, and the user is granted superuser permissions for the database specified by
the MYSQL\_DATABASE variable. Both MYSQL\_USER and MYSQL\_PASSWORD are required for a user
to be created—if any of the two variables is not set, the other is ignored. If both variables are set but
MYSQL\_DATABASE is not, the user is created without any privileges.

#### Note

There is no need to use this mechanism to create the root superuser, which is created by default with the password set by either one of the mechanisms discussed in the descriptions for MYSQL\_ROOT\_PASSWORD and MYSQL\_RANDOM\_ROOT\_PASSWORD, unless MYSQL\_ALLOW\_EMPTY\_PASSWORD is true.

- MYSQL\_ROOT\_HOST: By default, MySQL creates the 'root'@'localhost' account. This account can only be connected to from inside the container as described in Connecting to MySQL Server from within the Container. To allow root connections from other hosts, set this environment variable. For example, the value 172.17.0.1, which is the default Docker gateway IP, allows connections from the host machine that runs the container. The option accepts only one entry, but wildcards are allowed (for example, MYSQL\_ROOT\_HOST=172.\*.\*.\* or MYSQL\_ROOT\_HOST=%).
- MYSQL\_LOG\_CONSOLE: When the variable is true (which is its default state for MySQL 8.0 server containers), the MySQL Server's error log is redirected to stderr, so that the error log goes into the Docker container's log and is viewable using the docker logs mysqld-container command.

#### Note

The variable has no effect if a server configuration file from the host has been mounted (see Persisting Data and Configuration Changes on bind-mounting a configuration file).

MYSOL ROOT PASSWORD: This variable specifies a password that is set for the MySQL root account.

### Warning

Setting the MySQL root user password on the command line is insecure. As an alternative to specifying the password explicitly, you can set the variable with a container file path for a password file, and then mount a file from your host that contains the password at the container file path. This is still not very secure, as the location of the password file is still exposed. It is preferable to use the default settings of MYSQL\_RANDOM\_ROOT\_PASSWORD and MYSQL\_ONETIME\_PASSWORD both being true.

• MYSQL\_ALLOW\_EMPTY\_PASSWORD. Set it to true to allow the container to be started with a blank password for the root user.

#### Warning

Setting this variable to true is insecure, because it is going to leave your MySQL instance completely unprotected, allowing anyone to gain complete superuser access. It is preferable to use the default settings of MYSQL\_RANDOM\_ROOT\_PASSWORD and MYSQL\_ONETIME\_PASSWORD both being true.

# 7.6.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker

### Warning

The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk. This section discusses some known issues for the images when used on non-Linux platforms.

Known Issues for using the MySQL Server Docker images from Oracle on Windows include:

If you are bind-mounting on the container's MySQL data directory (see Persisting Data and Configuration
 Changes for details), you have to set the location of the server socket file with the --socket option to
 somewhere outside of the MySQL data directory; otherwise, the server fails to start. This is because the
 way Docker for Windows handles file mounting does not allow a host file from being bind-mounted on
 the socket file.

# 7.7 Installing MySQL on Linux from the Native Software Repositories

Many Linux distributions include a version of the MySQL server, client tools, and development components in their native software repositories and can be installed with the platforms' standard package management systems. This section provides basic instructions for installing MySQL using those package management systems.

#### **Important**

Native packages are often several versions behind the currently available release. You are also normally unable to install development milestone releases (DMRs), since these are not usually made available in the native repositories. Before proceeding, we recommend that you check out the other installation options described in Chapter 7, *Installing MySQL on Linux*.

Distribution specific instructions are shown below:

• Red Hat Linux, Fedora, CentOS

#### Note

For a number of Linux distributions, you can install MySQL using the MySQL Yum repository instead of the platform's native software repository. See Section 7.1, "Installing MySQL on Linux Using the MySQL Yum Repository" for details.

For Red Hat and similar distributions, the MySQL distribution is divided into a number of separate packages, <code>mysql</code> for the client tools, <code>mysql-server</code> for the server and associated tools, and <code>mysql-libs</code> for the libraries. The libraries are required if you want to provide connectivity from different languages and environments such as Perl, Python and others.

To install, use the yum command to specify the packages that you want to install. For example:

```
#> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
---> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
---> Running transaction check
---> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
```

```
--> Finished Dependency Resolution
Dependencies Resolved
______
                   Arch Version
                                                   Repository Size
______
Installing:

        mysql
        x86_64
        5.1.48-2.fc13
        updates

        mysql-libs
        x86_64
        5.1.48-2.fc13
        updates

        mysql-server
        x86_64
        5.1.48-2.fc13
        updates

                                                                889 k
                                                                 1.2 M
                                                                  8.1 M
Installing for dependencies:
perl-DBD-MySQL x86_64 4.017-1.fc13 updates
                                                                136 k
Transaction Summary
______
Install 4 Package(s)
Upgrade 0 Package(s)
Total download size: 10 M
Installed size: 30 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
Processing delta metadata
Package(s) data still to download: 10 M
(1/4): mysql-5.1.48-2.fc13.x86_64.rpm
                                                   | 889 kB 00:04
(2/4): mysql-libs-5.1.48-2.fc13.x86_64.rpm
                                                   | 1.2 MB 00:06
(3/4): mysql-server-5.1.48-2.fc13.x86_64.rpm
                                                    8.1 MB
                                                               00:40
                                                  | 136 kB
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm
                                                               00:00
                 -----
                                          201 kB/s | 10 MB 00:52
Total
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
 Installing : mysql-libs-5.1.48-2.fc13.x86_64
                                                                    1/4
 Installing : mysql-5.1.48-2.fc13.x86_64
                                                                    2/4
                                                                    3/4
 Installing : perl-DBD-MySQL-4.017-1.fc13.x86_64
 Installing : mysql-server-5.1.48-2.fc13.x86_64
                                                                    4/4
Installed:
 mysql.x86_64 0:5.1.48-2.fc13
                                     mysql-libs.x86_64 0:5.1.48-2.fc13
 mysql-server.x86_64 0:5.1.48-2.fc13
Dependency Installed:
 perl-DBD-MySQL.x86_64 0:4.017-1.fc13
Complete!
```

MySQL and the MySQL server should now be installed. A sample configuration file is installed into / etc/my.cnf. To start the MySQL server use systematl:

```
$> systemctl start mysqld
```

The database tables are automatically created for you, if they do not already exist. You should, however, run mysql\_secure\_installation to set the root passwords on your server.

### Debian, Ubuntu, Kubuntu

### Note

For supported Debian and Ubuntu versions, MySQL can be installed using the MySQL APT Repository instead of the platform's native software repository. See Section 7.2, "Installing MySQL on Linux Using the MySQL APT Repository" for details.

On Debian and related distributions, there are two packages for MySQL in their software repositories, mysql-client and mysql-server, for the client and server components respectively. You should

specify an explicit version, for example mysql-client-5.1, to ensure that you install the version of MySQL that you want.

To download and install, including any dependencies, use the apt-get command, specifying the packages that you want to install.

#### Note

Before installing, make sure that you update your apt-get index files to ensure you are downloading the latest available version.

#### Note

The apt-get command installs a number of packages, including the MySQL server, in order to provide the typical tools and application environment. This can mean that you install a large number of packages in addition to the main MySQL package.

During installation, the initial database is created, and you are prompted for the MySQL root password (and confirmation). A configuration file is created in /etc/mysql/my.cnf. An init script is created in /etc/init.d/mysql.

The server should already be started. You can manually start and stop the server using:

```
#> service mysql [start|stop]
```

The service is automatically added to the 2, 3 and 4 run levels, with stop scripts in the single, shutdown and restart levels.

# 7.8 Installing MySQL on Linux with Juju

The Juju deployment framework supports easy installation and configuration of MySQL servers. For instructions, see https://jujucharms.com/mysql/.

# 7.9 Managing MySQL Server with systemd

If you install MySQL using an RPM or Debian package on the following Linux platforms, server startup and shutdown is managed by systemd:

- · RPM package platforms:
  - · Enterprise Linux variants version 7 and higher
  - SUSE Linux Enterprise Server 12 and higher
  - Fedora 29 and higher
- · Debian family platforms:
  - · Debian platforms
  - · Ubuntu platforms

If you install MySQL from a generic binary distribution on a platform that uses systemd, you can manually configure systemd support for MySQL following the instructions provided in the post-installation setup section of the MySQL Secure Deployment Guide.

If you install MySQL from a source distribution on a platform that uses systemd, obtain systemd support for MySQL by configuring the distribution using the <code>-DWITH\_SYSTEMD=1 CMake</code> option. See Section 4.7, "MySQL Source-Configuration Options".

The following discussion covers these topics:

- Overview of systemd
- Configuring systemd for MySQL
- Configuring Multiple MySQL Instances Using systemd
- Migrating from mysqld\_safe to systemd

### **Note**

On platforms for which systemd support for MySQL is installed, scripts such as  ${\tt mysqld\_safe}$  and the System V initialization script are unnecessary and are not installed. For example,  ${\tt mysqld\_safe}$  can handle server restarts, but systemd provides the same capability, and does so in a manner consistent with management of other services rather than by using an application-specific program.

One implication of the non-use of <code>mysqld\_safe</code> on platforms that use systemd for server management is that use of <code>[mysqld\_safe]</code> or <code>[safe\_mysqld]</code> sections in option files is not supported and might lead to unexpected behavior.

Because systemd has the capability of managing multiple MySQL instances on platforms for which systemd support for MySQL is installed, mysqld\_multi and mysqld\_multi.server are unnecessary and are not installed.

# Overview of systemd

systemd provides automatic MySQL server startup and shutdown. It also enables manual server management using the systematl command. For example:

```
$> systemctl {start|stop|restart|status} mysqld
```

Alternatively, use the service command (with the arguments reversed), which is compatible with System V systems:

\$> service mysqld {start|stop|restart|status}

#### Note

For the systemctl command (and the alternative service command), if the MySQL service name is not mysqld then use the appropriate name. For example, use mysql rather than mysqld on Debian-based and SLES systems.

Support for systemd includes these files:

- mysqld.service (RPM platforms), mysql.service (Debian platforms): systemd service unit configuration file, with details about the MySQL service.
- mysqld@.service (RPM platforms), mysql@.service (Debian platforms): Like mysqld.service or mysql.service, but used for managing multiple MySQL instances.
- mysqld.tmpfiles.d: File containing information to support the tmpfiles feature. This file is installed under the name mysql.conf.

• mysqld\_pre\_systemd (RPM platforms), mysql-system-start (Debian platforms): Support script for the unit file. This script assists in creating the error log file only if the log location matches a pattern (/ var/log/mysql\*.log for RPM platforms, /var/log/mysql/\*.log for Debian platforms). In other cases, the error log directory must be writable or the error log must be present and writable for the user running the mysqld process.

# Configuring systemd for MySQL

To add or change systemd options for MySQL, these methods are available:

- Use a localized systemd configuration file.
- Arrange for systemd to set environment variables for the MySQL server process.
- Set the MYSQLD\_OPTS systemd variable.

To use a localized systemd configuration file, create the /etc/systemd/system/mysqld.service.d directory if it does not exist. In that directory, create a file that contains a [Service] section listing the desired settings. For example:

```
[Service]
LimitNOFILE=max_open_files
Nice=nice_level
LimitCore=core_file_limit
Environment="LD_PRELOAD=/path/to/malloc/library"
Environment="TZ=time_zone_setting"
```

The discussion here uses override.conf as the name of this file. Newer versions of systemd support the following command, which opens an editor and permits you to edit the file:

```
systemctl edit mysqld  # RPM platforms
systemctl edit mysql  # Debian platforms
```

Whenever you create or change override.conf, reload the systemd configuration, then tell systemd to restart the MySQL service:

```
systemctl daemon-reload
systemctl restart mysqld # RPM platforms
systemctl restart mysql # Debian platforms
```

With systemd, the override.conf configuration method must be used for certain parameters, rather than settings in a [mysqld], [mysqld safe], or [safe mysqld] group in a MySQL option file:

- For some parameters, override.conf must be used because systemd itself must know their values and it cannot read MySQL option files to get them.
- Parameters that specify values otherwise settable only using options known to mysqld\_safe must be specified using systemd because there is no corresponding mysqld parameter.

For additional information about using systemd rather than mysqld\_safe, see Migrating from mysqld\_safe to systemd.

You can set the following parameters in override.conf:

- To set the number of file descriptors available to the MySQL server, use LimitNOFILE in override.conf rather than the open\_files\_limit system variable for mysqld or --open-files-limit option for mysqld\_safe.
- To set the maximum core file size, use LimitCore in override.conf rather than the --core-file-size option for mysqld\_safe.

• To set the scheduling priority for the MySQL server, use Nice in override.conf rather than the -- nice option for mysqld\_safe.

Some MySQL parameters are configured using environment variables:

- LD\_PRELOAD: Set this variable if the MySQL server should use a specific memory-allocation library.
- NOTIFY\_SOCKET: This environment variable specifies the socket that mysqld uses to communicate notification of startup completion and service status change with systemd. It is set by systemd when the mysqld service is started. The mysqld service reads the variable setting and writes to the defined location.

In MySQL 8.0, mysqld uses the Type=notify process startup type. (Type=forking was used in MySQL 5.7.) With Type=notify, systemd automatically configures a socket file and exports the path to the NOTIFY\_SOCKET environment variable.

TZ: Set this variable to specify the default time zone for the server.

There are multiple ways to specify environment variable values for use by the MySQL server process managed by systemd:

- Use Environment lines in the override.conf file. For the syntax, see the example in the preceding discussion that describes how to use this file.
- Specify the values in the /etc/sysconfig/mysql file (create the file if it does not exist). Assign values using the following syntax:

```
LD_PRELOAD=/path/to/malloc/library
TZ=time_zone_setting
```

After modifying /etc/sysconfig/mysql, restart the server to make the changes effective:

```
systemctl restart mysqld # RPM platforms
systemctl restart mysql # Debian platforms
```

To specify options for mysqld without modifying systemd configuration files directly, set or unset the MYSQLD\_OPTS systemd variable. For example:

```
systemctl set-environment MYSQLD_OPTS="--general_log=1"
systemctl unset-environment MYSQLD_OPTS
```

MYSQLD\_OPTS can also be set in the /etc/sysconfig/mysql file.

After modifying the systemd environment, restart the server to make the changes effective:

```
systemctl restart mysqld # RPM platforms
systemctl restart mysql # Debian platforms
```

For platforms that use systemd, the data directory is initialized if empty at server startup. This might be a problem if the data directory is a remote mount that has temporarily disappeared: The mount point would appear to be an empty data directory, which then would be initialized as a new data directory. To suppress this automatic initialization behavior, specify the following line in the /etc/sysconfig/mysql file (create the file if it does not exist):

```
NO INIT=true
```

# Configuring Multiple MySQL Instances Using systemd

This section describes how to configure systemd for multiple instances of MySQL.

#### Note

Because systemd has the capability of managing multiple MySQL instances on platforms for which systemd support is installed, mysqld\_multi and mysqld\_multi.server are unnecessary and are not installed.

To use multiple-instance capability, modify the my.cnf option file to include configuration of key options for each instance. These file locations are typical:

- /etc/my.cnf or /etc/mysql/my.cnf (RPM platforms)
- /etc/mysql/mysql.conf.d/mysqld.cnf (Debian platforms)

For example, to manage two instances named replica01 and replica02, add something like this to the option file:

### RPM platforms:

```
[mysqld@replica01]
datadir=/var/lib/mysql-replica01
socket=/var/lib/mysql-replica01/mysql.sock
port=3307
log-error=/var/log/mysqld-replica01.log
[mysqld@replica02]
datadir=/var/lib/mysql-replica02
socket=/var/lib/mysql-replica02/mysql.sock
port=3308
log-error=/var/log/mysqld-replica02.log
```

### Debian platforms:

```
[mysqld@replica01]
datadir=/var/lib/mysql-replica01
socket=/var/lib/mysql-replica01/mysql.sock
port=3307
log-error=/var/log/mysql/replica01.log
[mysqld@replica02]
datadir=/var/lib/mysql-replica02
socket=/var/lib/mysql-replica02/mysql.sock
port=3308
log-error=/var/log/mysql/replica02.log
```

The replica names shown here use @ as the delimiter because that is the only delimiter supported by systemd.

Instances then are managed by normal systemd commands, such as:

```
systemctl start mysqld@replica01 systemctl start mysqld@replica02
```

To enable instances to run at boot time, do this:

```
systemctl enable mysqld@replica01
systemctl enable mysqld@replica02
```

Use of wildcards is also supported. For example, this command displays the status of all replica instances:

```
systemctl status 'mysqld@replica*'
```

For management of multiple MySQL instances on the same machine, systemd automatically uses a different unit file:

mysqld@.service rather than mysqld.service (RPM platforms)

• mysql@.service rather than mysql.service (Debian platforms)

In the unit file, %I and %i reference the parameter passed in after the @ marker and are used to manage the specific instance. For a command such as this:

```
systemctl start mysqld@replica01
```

systemd starts the server using a command such as this:

```
mysqld --defaults-group-suffix=@%I ...
```

The result is that the [server], [mysqld], and [mysqld@replica01] option groups are read and used for that instance of the service.

### Note

On Debian platforms, AppArmor prevents the server from reading or writing / var/lib/mysql-replica\*, or anything other than the default locations. To address this, you must customize or disable the profile in /etc/apparmor.d/usr.sbin.mysqld.

#### Note

On Debian platforms, the packaging scripts for MySQL uninstallation cannot currently handle mysqld@ instances. Before removing or upgrading the package, you must stop any extra instances manually first.

# Migrating from mysqld\_safe to systemd

Because mysqld\_safe is not installed on platforms that use systemd to manage MySQL, options previously specified for that program (for example, in an [mysqld\_safe] or [safe\_mysqld] option group) must be specified another way:

• Some mysqld\_safe options are also understood by mysqld and can be moved from the [mysqld\_safe] or [safe\_mysqld] option group to the [mysqld] group. This does *not* include --pid-file, --open-files-limit, or --nice. To specify those options, use the override.conf systemd file, described previously.

#### Note

On systemd platforms, use of [mysqld\_safe] and [safe\_mysqld] option groups is not supported and may lead to unexpected behavior.

- For some mysqld\_safe options, there are alternative mysqld procedures. For example, the mysqld\_safe option for enabling syslog logging is --syslog, which is deprecated. To write error log output to the system log, use the instructions at Error Logging to the System Log.
- mysqld\_safe options not understood by mysqld can be specified in override.conf or environment variables. For example, with mysqld\_safe, if the server should use a specific memory allocation library, this is specified using the --malloc-lib option. For installations that manage the server with systemd, arrange to set the LD\_PRELOAD environment variable instead, as described previously.

# Chapter 8 Installing MySQL on Solaris

### **Table of Contents**

#### Note

MySQL 8.0 supports Solaris 11.4 and higher

MySQL on Solaris is available in a number of different formats.

- For information on installing using the native Solaris PKG format, see Section 8.1, "Installing MySQL on Solaris Using a Solaris PKG".
- To use a standard tar binary installation, use the notes provided in Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*. Check the notes and hints at the end of this section for Solaris specific notes that you may need before or after installation.

#### Note

MySQL 5.7 has a dependency on the Oracle Developer Studio Runtime Libraries; but this does not apply to MySQL 8.0.

To obtain a binary MySQL distribution for Solaris in tarball or PKG format, https://dev.mysql.com/downloads/mysql/8.0.html.

Additional notes to be aware of when installing and using MySQL on Solaris:

• If you want to use MySQL with the mysql user and group, use the groupadd and useradd commands:

```
groupadd mysql
useradd -g mysql -s /bin/false mysql
```

If you install MySQL using a binary tarball distribution on Solaris, because the Solaris tar cannot handle
long file names, use GNU tar (gtar) to unpack the distribution. If you do not have GNU tar on your
system, install it with the following command:

```
pkg install archiver/gnu-tar
```

- You should mount any file systems on which you intend to store InnoDB files with the forcedirectio option. (By default mounting is done without this option.) Failing to do so causes a significant drop in performance when using the InnoDB storage engine on this platform.
- If you would like MySQL to start automatically, you can copy support-files/mysql.server to / etc/init.d and create a symbolic link to it named /etc/rc3.d/S99mysql.server.
- If too many processes try to connect very rapidly to mysqld, you should see this error in the MySQL log:

```
Error in accept: Protocol error
```

You might try starting the server with the --back log=50 option as a workaround for this.

• To configure the generation of core files on Solaris you should use the coreadm command. Because of the security implications of generating a core on a setuid() application, by default, Solaris does not

support core files on setuid() programs. However, you can modify this behavior using coreadm. If
you enable setuid() core files for the current user, they are generated using mode 600 and are owned
by the superuser.

## 8.1 Installing MySQL on Solaris Using a Solaris PKG

You can install MySQL on Solaris using a binary package of the native Solaris PKG format instead of the binary tarball distribution.

#### Note

MySQL 5.7 has a dependency on the Oracle Developer Studio Runtime Libraries; but this does not apply to MySQL 8.0.

To use this package, download the corresponding <code>mysql-VERSION-solaris11-PLATFORM.pkg.gz</code> file, then uncompress it. For example:

```
$> gunzip mysql-8.0.42-solaris11-x86_64.pkg.gz
```

To install a new package, use pkgadd and follow the onscreen prompts. You must have root privileges to perform this operation:

The PKG installer installs all of the files and tools needed, and then initializes your database if one does not exist. To complete the installation, you should set the root password for MySQL as provided in the instructions at the end of the installation. Alternatively, you can run the mysql\_secure\_installation script that comes with the installation.

By default, the PKG package installs MySQL under the root path /opt/mysql. You can change only the installation root path when using pkgadd, which can be used to install MySQL in a different Solaris zone. If you need to install in a specific directory, use a binary tar file distribution.

The pkg installer copies a suitable startup script for MySQL into /etc/init.d/mysql. To enable MySQL to startup and shutdown automatically, you should create a link between this file and the init script directories. For example, to ensure safe startup and shutdown of MySQL you could use the following commands to add the right links:

```
$> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
$> ln /etc/init.d/mysql /etc/rc0.d/K02mysql
```

To remove MySQL, the installed package name is mysql. You can use this in combination with the pkgrm command to remove the installation.

To upgrade when using the Solaris package file format, you must remove the existing installation before installing the updated package. Removal of the package does not delete the existing database information, only the server, binaries and support files. The typical upgrade sequence is therefore:

```
$> mysqladmin shutdown
$> pkgrm mysql
$> pkgadd -d mysql-8.0.42-solaris11-x86_64.pkg
$> mysqld_safe &
$> mysql_upgrade  # prior to MySQL 8.0.16 only
```

You should check the notes in Chapter 10, *Upgrading MySQL* before performing any upgrade.

# Chapter 9 Postinstallation Setup and Testing

### **Table of Contents**

9.1 Initializing the Data Directory	179
9.2 Starting the Server	
9.2.1 Troubleshooting Problems Starting the MySQL Server	
9.3 Testing the Server	188
9.4 Securing the Initial MySQL Account	190
9.5 Starting and Stopping MySQL Automatically	

This section discusses tasks that you should perform after installing MySQL:

- If necessary, initialize the data directory and create the MySQL grant tables. For some MySQL installation methods, data directory initialization may be done for you automatically:
  - Windows installation operations performed by MySQL Installer.
  - Installation on Linux using a server RPM or Debian distribution from Oracle.
  - Installation using the native packaging system on many platforms, including Debian Linux, Ubuntu Linux, Gentoo Linux, and others.
  - Installation on macOS using a DMG distribution.

For other platforms and installation types, you must initialize the data directory manually. These include installation from generic binary and source distributions on Unix and Unix-like system, and installation from a ZIP Archive package on Windows. For instructions, see Section 9.1, "Initializing the Data Directory".

- Start the server and make sure that it can be accessed. For instructions, see Section 9.2, "Starting the Server", and Section 9.3, "Testing the Server".
- Assign passwords to the initial root account in the grant tables, if that was not already done during data directory initialization. Passwords prevent unauthorized access to the MySQL server. For instructions, see Section 9.4, "Securing the Initial MySQL Account".
- Optionally, arrange for the server to start and stop automatically when your system starts and stops. For instructions, see Section 9.5, "Starting and Stopping MySQL Automatically".
- Optionally, populate time zone tables to enable recognition of named time zones. For instructions, see MySQL Server Time Zone Support.

When you are ready to create additional user accounts, you can find information on the MySQL access control system and account management in Access Control and Account Management.

## 9.1 Initializing the Data Directory

After MySQL is installed, the data directory must be initialized, including the tables in the mysql system schema:

• For some MySQL installation methods, data directory initialization is automatic, as described in Chapter 9, *Postinstallation Setup and Testing*.

• For other installation methods, you must initialize the data directory manually. These include installation from generic binary and source distributions on Unix and Unix-like systems, and installation from a ZIP Archive package on Windows.

This section describes how to initialize the data directory manually for MySQL installation methods for which data directory initialization is not automatic. For some suggested commands that enable testing whether the server is accessible and working properly, see Section 9.3, "Testing the Server".

#### **Note**

In MySQL 8.0, the default authentication plugin has changed from mysql\_native\_password to caching\_sha2\_password, and the 'root'@'localhost' administrative account uses caching\_sha2\_password by default. If you prefer that the root account use the previous default authentication plugin (mysql\_native\_password), see caching\_sha2\_password and the root Administrative Account.

The mysql\_native\_password plugin is deprecated as of MySQL 8.0.34, disabled by default as of MySQL 8.4.0, and removed as of MySQL 9.0.0.

- · Data Directory Initialization Overview
- · Data Directory Initialization Procedure
- Server Actions During Data Directory Initialization
- · Post-Initialization root Password Assignment

### **Data Directory Initialization Overview**

In the examples shown here, the server is intended to run under the user ID of the mysql login account. Either create the account if it does not exist (see Create a mysql User and Group), or substitute the name of a different existing login account that you plan to use for running the server.

1. Change location to the top-level directory of your MySQL installation, which is typically /usr/local/mysql (adjust the path name for your system as necessary):

```
cd /usr/local/mysql
```

Within this directory you can find several files and subdirectories, including the bin subdirectory that contains the server, as well as client and utility programs.

2. The secure\_file\_priv system variable limits import and export operations to a specific directory. Create a directory whose location can be specified as the value of that variable:

```
mkdir mysql-files
```

Grant directory user and group ownership to the mysql user and mysql group, and set the directory permissions appropriately:

```
chown mysql:mysql mysql-files
chmod 750 mysql-files
```

3. Use the server to initialize the data directory, including the mysql schema containing the initial MySQL grant tables that determine how users are permitted to connect to the server. For example:

```
bin/mysqld --initialize --user=mysql
```

For important information about the command, especially regarding command options you might use, see Data Directory Initialization Procedure. For details about how the server performs initialization, see Server Actions During Data Directory Initialization.

Typically, data directory initialization need be done only after you first install MySQL. (For upgrades to an existing installation, perform the upgrade procedure instead; see Chapter 10, *Upgrading MySQL*.) However, the command that initializes the data directory does not overwrite any existing mysql schema tables, so it is safe to run in any circumstances.

4. If you want to deploy the server with automatic support for secure connections, use the mysql\_ssl\_rsa\_setup utility to create default SSL and RSA files:

bin/mysql\_ssl\_rsa\_setup

For more information, see mysql\_ssl\_rsa\_setup — Create SSL/RSA Files.

#### Note

The mysql\_ssl\_rsa\_setup utility is deprecated as of MySQL 8.0.34.

- 5. In the absence of any option files, the server starts with its default settings. (See Server Configuration Defaults.) To explicitly specify options that the MySQL server should use at startup, put them in an option file such as /etc/my.cnf or /etc/mysql/my.cnf. (See Using Option Files.) For example, you can use an option file to set the secure file priv system variable.
- 6. To arrange for MySQL to start without manual intervention at system boot time, see Section 9.5, "Starting and Stopping MySQL Automatically".
- 7. Data directory initialization creates time zone tables in the mysql schema but does not populate them. To do so, use the instructions in MySQL Server Time Zone Support.

## **Data Directory Initialization Procedure**

Change location to the top-level directory of your MySQL installation, which is typically /usr/local/mysql (adjust the path name for your system as necessary):

```
cd /usr/local/mysql
```

To initialize the data directory, invoke <code>mysqld</code> with the <code>--initialize</code> or <code>--initialize-insecure</code> option, depending on whether you want the server to generate a random initial password for the <code>'root'@'localhost'</code> account, or to create that account with no password:

- Use --initialize for "secure by default" installation (that is, including generation of a random initial root password). In this case, the password is marked as expired and you must choose a new one.
- With --initialize-insecure, no root password is generated. This is insecure; it is assumed that you intend to assign a password to the account in a timely fashion before putting the server into production use.

For instructions on assigning a new 'root'@'localhost' password, see Post-Initialization root Password Assignment.

#### Note

The server writes any messages (including any initial password) to its standard error output. This may be redirected to the error log, so look there if you do not see

the messages on your screen. For information about the error log, including where it is located, see The Error Log.

On Windows, use the --console option to direct messages to the console.

On Unix and Unix-like systems, it is important for the database directories and files to be owned by the mysql login account so that the server has read and write access to them when you run it later. To ensure this, start mysqld from the system root account and include the --user option as shown here:

```
bin/mysqld --initialize --user=mysql
bin/mysqld --initialize-insecure --user=mysql
```

Alternatively, execute <code>mysqld</code> while logged in as <code>mysql</code>, in which case you can omit the <code>--user</code> option from the command.

On Windows, use one of these commands:

```
bin\mysqld --initialize --console
bin\mysqld --initialize-insecure --console
```

#### Note

Data directory initialization might fail if required system libraries are missing. For example, you might see an error like this:

```
bin/mysqld: error while loading shared libraries:
libnuma.so.1: cannot open shared object file:
No such file or directory
```

If this happens, you must install the missing libraries manually or with your system's package manager. Then retry the data directory initialization command.

It might be necessary to specify other options such as --basedir or --datadir if mysqld cannot identify the correct locations for the installation directory or data directory. For example (enter the command on a single line):

```
bin/mysqld --initialize --user=mysql
   --basedir=/opt/mysql/mysql
   --datadir=/opt/mysql/mysql/data
```

Alternatively, put the relevant option settings in an option file and pass the name of that file to mysqld. For Unix and Unix-like systems, suppose that the option file name is opt/mysql/mysql/etc/my.cnf. Put these lines in the file:

```
[mysqld]
basedir=/opt/mysql/mysql
datadir=/opt/mysql/mysql/data
```

Then invoke mysqld as follows (enter the command on a single line, with the --defaults-file option first):

```
bin/mysqld --defaults-file=/opt/mysql/mysql/etc/my.cnf
   --initialize --user=mysql
```

On Windows, suppose that C:\my.ini contains these lines:

```
[mysqld]
basedir=C:\\Program Files\\MySQL\\MySQL Server 8.0
datadir=D:\\MySQLdata
```

Then invoke mysqld as follows (again, you should enter the command on a single line, with the --defaults-file option first):

```
bin\mysqld --defaults-file=C:\my.ini
    --initialize --console
```

### **Important**

When initializing the data directory, you should not specify any options other than those used for setting directory locations such as --basedir or --datadir, and the --user option if needed. Options to be employed by the MySQL server during normal use can be set when restarting it following initialization. See the description of the --initialize option for further information.

# **Server Actions During Data Directory Initialization**

#### Note

The data directory initialization sequence performed by the server does not substitute for the actions performed by mysql\_secure\_installation and mysql\_ssl\_rsa\_setup. See mysql\_secure\_installation — Improve MySQL Installation Security, and mysql\_ssl\_rsa\_setup — Create SSL/RSA Files.

When invoked with the --initialize or --initialize-insecure option, mysqld performs the following actions during the data directory initialization sequence:

- 1. The server checks for the existence of the data directory as follows:
  - If no data directory exists, the server creates it.
  - If the data directory exists but is not empty (that is, it contains files or subdirectories), the server exits after producing an error message:

```
[ERROR] --initialize specified but the data directory exists. Aborting.
```

In this case, remove or rename the data directory and try again.

An existing data directory is permitted to be nonempty if every entry has a name that begins with a period (.).

- 2. Within the data directory, the server creates the <code>mysql</code> system schema and its tables, including the data dictionary tables, grant tables, time zone tables, and server-side help tables. See The mysql System Schema.
- The server initializes the system tablespace and related data structures needed to manage InnoDB tables.

#### Note

After mysqld sets up the InnoDB system tablespace, certain changes to tablespace characteristics require setting up a whole new instance. Qualifying changes include the file name of the first file in the system tablespace and the number of undo logs. If you do not want to use the default values, make sure that the settings for the innodb\_data\_file\_path and innodb\_log\_file\_size configuration parameters are in place in the MySQL configuration file before running mysqld. Also make sure to specify as necessary other parameters that affect the creation and location of InnoDB files, such as innodb\_data\_home\_dir\_and\_innodb\_log\_group\_home\_dir.

If those options are in your configuration file but that file is not in a location that MySQL reads by default, specify the file location using the --defaults-extra-file option when you run mysqld.

4. The server creates a 'root'@'localhost' superuser account and other reserved accounts (see Reserved Accounts). Some reserved accounts are locked and cannot be used by clients, but 'root'@'localhost' is intended for administrative use and you should assign it a password.

Server actions with respect to a password for the 'root'@'localhost' account depend on how you invoke it:

• With --initialize but not --initialize-insecure, the server generates a random password, marks it as expired, and writes a message displaying the password:

```
[Warning] A temporary password is generated for root@localhost:iTag*AfrH5ej
```

With --initialize-insecure, (either with or without --initialize because --initialize-insecure implies --initialize), the server does not generate a password or mark it expired, and writes a warning message:

```
[Warning] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
```

For instructions on assigning a new 'root'@'localhost' password, see Post-Initialization root Password Assignment.

- The server populates the server-side help tables used for the HELP statement (see HELP Statement).
   The server does not populate the time zone tables. To do so manually, see MySQL Server Time Zone Support.
- 6. If the init\_file system variable was given to name a file of SQL statements, the server executes the statements in the file. This option enables you to perform custom bootstrapping sequences.

When the server operates in bootstrap mode, some functionality is unavailable that limits the statements permitted in the file. These include statements that relate to account management (such as CREATE USER or GRANT), replication, and global transaction identifiers.

7. The server exits.

## **Post-Initialization root Password Assignment**

After you initialize the data directory by starting the server with --initialize or --initialize-insecure, start the server normally (that is, without either of those options) and assign the 'root'@'localhost' account a new password:

- 1. Start the server. For instructions, see Section 9.2, "Starting the Server".
- 2. Connect to the server:
  - If you used --initialize but not --initialize-insecure to initialize the data directory, connect to the server as root:

```
mysql -u root -p
```

Then, at the password prompt, enter the random password that the server generated during the initialization sequence:

```
Enter password: (enter the random root password here)
```

Look in the server error log if you do not know this password.

 If you used --initialize-insecure to initialize the data directory, connect to the server as root without a password:

```
mysql -u root --skip-password
```

After connecting, use an ALTER USER statement to assign a new root password:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'root-password';
```

See also Section 9.4, "Securing the Initial MySQL Account".

#### Note

Attempts to connect to the host 127.0.0.1 normally resolve to the localhost account. However, this fails if the server is run with skip\_name\_resolve enabled. If you plan to do that, make sure that an account exists that can accept a connection. For example, to be able to connect as root using --host=127.0.0.1 or --host=::1, create these accounts:

```
CREATE USER 'root'@'127.0.0.1' IDENTIFIED BY 'root-password';
CREATE USER 'root'@'::1' IDENTIFIED BY 'root-password';
```

It is possible to put those statements in a file to be executed using the <code>init\_file</code> system variable, as discussed in Server Actions During Data Directory Initialization.

## 9.2 Starting the Server

This section describes how start the server on Unix and Unix-like systems. (For Windows, see Section 5.4.5, "Starting the Server for the First Time".) For some suggested commands that you can use to test whether the server is accessible and working properly, see Section 9.3, "Testing the Server".

Start the MySQL server like this if your installation includes mysqld\_safe:

```
$> bin/mysqld_safe --user=mysql &
```

#### Note

For Linux systems on which MySQL is installed using RPM packages, server startup and shutdown is managed using systemd rather than  $mysqld\_safe$ , and  $mysqld\_safe$  is not installed. See Section 7.9, "Managing MySQL Server with systemd".

Start the server like this if your installation includes systemd support:

```
$> systemctl start mysqld
```

Substitute the appropriate service name if it differs from mysqld (for example, mysql on SLES systems).

It is important that the MySQL server be run using an unprivileged (non-root) login account. To ensure this, run mysqld\_safe as root and include the --user option as shown. Otherwise, you should execute the program while logged in as mysql, in which case you can omit the --user option from the command.

For further instructions for running MySQL as an unprivileged user, see How to Run MySQL as a Normal User.

If the command fails immediately and prints <code>mysqld ended</code>, look for information in the error log (which by default is the <code>host name.err</code> file in the data directory).

If the server is unable to access the data directory it starts or read the grant tables in the mysql schema, it writes a message to its error log. Such problems can occur if you neglected to create the grant tables by initializing the data directory before proceeding to this step, or if you ran the command that initializes the data directory without the --user option. Remove the data directory and run the command with the --user option.

If you have other problems starting the server, see Section 9.2.1, "Troubleshooting Problems Starting the MySQL Server". For more information about mysqld\_safe, see mysqld\_safe — MySQL Server Startup Script. For more information about systemd support, see Section 7.9, "Managing MySQL Server with systemd".

### 9.2.1 Troubleshooting Problems Starting the MySQL Server

This section provides troubleshooting suggestions for problems starting the server. For additional suggestions for Windows systems, see Section 5.5, "Troubleshooting a Microsoft Windows MySQL Server Installation".

If you have problems starting the server, here are some things to try:

• Check the error log to see why the server does not start. Log files are located in the data directory (typically C:\Program Files\MySQL\MySQL Server 8.0\data on Windows, /usr/local/mysql/data for a Unix/Linux binary distribution, and /usr/local/var for a Unix/Linux source distribution). Look in the data directory for files with names of the form host\_name.err and host\_name.log, where host\_name is the name of your server host. Then examine the last few lines of these files. Use tail to display them:

```
$> tail host_name.err
$> tail host_name.log
```

• Specify any special options needed by the storage engines you are using. You can create a my.cnf file and specify startup options for the engines that you plan to use. If you are going to use storage engines that support transactional tables (Innode, Nde), be sure that you have them configured the way you want before starting the server. If you are using Innode tables, see Innode Configuration for guidelines and Innode Startup Options and System Variables for option syntax.

Although storage engines use default values for options that you omit, Oracle recommends that you review the available options and specify explicit values for any options whose defaults are not appropriate for your installation.

• Make sure that the server knows where to find the data directory. The mysqld server uses this directory as its current directory. This is where it expects to find databases and where it expects to write log files. The server also writes the pid (process ID) file in the data directory.

The default data directory location is hardcoded when the server is compiled. To determine what the default path settings are, invoke mysqld with the --verbose and --help options. If the data directory is located somewhere else on your system, specify that location with the --datadir option to mysqld or mysqld\_safe, on the command line or in an option file. Otherwise, the server does not work properly. As an alternative to the --datadir option, you can specify mysqld the location of the base directory under which MySQL is installed with the --basedir, and mysqld looks for the data directory there.

To check the effect of specifying path options, invoke mysqld with those options followed by the --verbose and --help options. For example, if you change location to the directory where mysqld

is installed and then run the following command, it shows the effect of starting the server with a base directory of /usr/local:

```
$> ./mysqld --basedir=/usr/local --verbose --help
```

You can specify other options such as --datadir as well, but --verbose and --help must be the last options.

Once you determine the path settings you want, start the server without --verbose and --help.

If mysqld is currently running, you can find out what path settings it is using by executing this command:

```
$> mysqladmin variables
```

Or:

```
$> mysqladmin -h host_name variables
```

host name is the name of the MySQL server host.

 Make sure that the server can access the data directory. The ownership and permissions of the data directory and its contents must allow the server to read and modify them.

If you get Errcode 13 (which means Permission denied) when starting mysqld, this means that the privileges of the data directory or its contents do not permit server access. In this case, you change the permissions for the involved files and directories so that the server has the right to use them. You can also start the server as root, but this raises security issues and should be avoided.

Change location to the data directory and check the ownership of the data directory and its contents to make sure the server has access. For example, if the data directory is /usr/local/mysql/var, use this command:

```
$> ls -la /usr/local/mysql/var
```

If the data directory or its files or subdirectories are not owned by the login account that you use for running the server, change their ownership to that account. If the account is named <code>mysql</code>, use these commands:

```
$> chown -R mysql /usr/local/mysql/var
$> chgrp -R mysql /usr/local/mysql/var
```

Even with correct ownership, MySQL might fail to start up if there is other security software running on your system that manages application access to various parts of the file system. In this case, reconfigure that software to enable mysqld to access the directories it uses during normal operation.

Verify that the network interfaces the server wants to use are available.

If either of the following errors occur, it means that some other program (perhaps another mysqld server) is using the TCP/IP port or Unix socket file that mysqld is trying to use:

```
Can't start server: Bind on TCP/IP port: Address already in use
Can't start server: Bind on unix socket...
```

Use ps to determine whether you have another mysqld server running. If so, shut down the server before starting mysqld again. (If another server is running, and you really want to run multiple servers, you can find information about how to do so in Running Multiple MySQL Instances on One Machine.)

If no other server is running, execute the command telnet your\_host\_name
tcp\_ip\_port\_number. (The default MySQL port number is 3306.) Then press Enter a couple of

times. If you do not get an error message like telnet: Unable to connect to remote host: Connection refused, some other program is using the TCP/IP port that mysqld is trying to use. Track down what program this is and disable it, or tell mysqld to listen to a different port with the --port option. In this case, specify the same non-default port number for client programs when connecting to the server using TCP/IP.

Another reason the port might be inaccessible is that you have a firewall running that blocks connections to it. If so, modify the firewall settings to permit access to the port.

If the server starts but you cannot connect to it, make sure that you have an entry in /etc/hosts that looks like this:

```
127.0.0.1 localhost
```

If you cannot get mysqld to start, try to make a trace file to find the problem by using the --debug option. See The DBUG Package.

## 9.3 Testing the Server

After the data directory is initialized and you have started the server, perform some simple tests to make sure that it works satisfactorily. This section assumes that your current location is the MySQL installation directory and that it has a bin subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

Alternatively, add the bin directory to your PATH environment variable setting. That enables your shell (command interpreter) to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See Setting Environment Variables.

Use mysqladmin to verify that the server is running. The following commands provide simple tests to check whether the server is up and responding to connections:

```
$> bin/mysqladmin version
$> bin/mysqladmin variables
```

If you cannot connect to the server, specify a -u root option to connect as root. If you have assigned a password for the root account already, you'll also need to specify -p on the command line and enter the password when prompted. For example:

```
$> bin/mysqladmin -u root -p version
Enter password: (enter root password here)
```

The output from mysqladmin version varies slightly depending on your platform and version of MySQL, but should be similar to that shown here:

To see what else you can do with mysgladmin, invoke it with the --help option.

Verify that you can shut down the server (include a -p option if the root account has a password already):

```
$> bin/mysqladmin -u root shutdown
```

Verify that you can start the server again. Do this by using mysqld\_safe or by invoking mysqld directly. For example:

```
$> bin/mysqld_safe --user=mysql &
```

If mysqld\_safe fails, see Section 9.2.1, "Troubleshooting Problems Starting the MySQL Server".

Run some simple tests to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use mysqlshow to see what databases exist:

The list of installed databases may vary, but always includes at least mysgl and information\_schema.

If you specify a database name, mysqlshow displays a list of the tables within the database:

```
$> bin/mysqlshow mysql
Database: mysql
         Tables
  columns_priv
  component
 db
 default_roles
  engine_cost
 func
  general_log
  global_grants
 gtid_executed
 help_category
 help_keyword
  help_relation
 help_topic
  innodb_index_stats
  innodb_table_stats
  ndb_binlog_index
  password_history
  plugin
 procs_priv
 proxies_priv
 role_edges
  server_cost
  servers
  slave_master_info
 slave_relay_log_info
  slave_worker_info
  slow_log
  tables_priv
  time_zone
  time_zone_leap_second
  time_zone_name
 time_zone_transition
```

```
| time_zone_transition_type |
| user
+-----+
```

Use the mysql program to select information from a table in the mysql schema:

At this point, your server is running and you can access it. To tighten security if you have not yet assigned a password to the initial account, follow the instructions in Section 9.4, "Securing the Initial MySQL Account".

For more information about <code>mysql, mysqladmin</code>, and <code>mysqlshow</code>, see <code>mysql</code> — The MySQL Command-Line Client, <code>mysqladmin</code> — A MySQL Server Administration Program, and <code>mysqlshow</code> — Display Database, Table, and Column Information.

# 9.4 Securing the Initial MySQL Account

The MySQL installation process involves initializing the data directory, including the grant tables in the mysql system schema that define MySQL accounts. For details, see Section 9.1, "Initializing the Data Directory".

This section describes how to assign a password to the initial root account created during the MySQL installation procedure, if you have not already done so.

#### Note

Alternative means for performing the process described in this section:

- On Windows, you can perform the process during installation with MySQL Installer (see Section 5.3, "MySQL Installer for Windows").
- On all platforms, the MySQL distribution includes mysql\_secure\_installation, a command-line utility that automates much of the process of securing a MySQL installation.
- On all platforms, MySQL Workbench is available and offers the ability to manage user accounts (see MySQL Workbench).

A password may already be assigned to the initial account under these circumstances:

- On Windows, installations performed using MySQL Installer give you the option of assigning a password.
- Installation using the macOS installer generates an initial random password, which the installer displays to the user in a dialog box.
- Installation using RPM packages generates an initial random password, which is written to the server error log.
- Installations using Debian packages give you the option of assigning a password.
- For data directory initialization performed manually using mysqld --initialize, mysqld generates an initial random password, marks it expired, and writes it to the server error log. See Section 9.1, "Initializing the Data Directory".

The mysql.user grant table defines the initial MySQL user account and its access privileges. Installation of MySQL creates only a 'root'@'localhost' superuser account that has all privileges and can do anything. If the root account has an empty password, your MySQL installation is unprotected: Anyone can connect to the MySQL server as root without a password and be granted all privileges.

The 'root'@'localhost' account also has a row in the mysql.proxies\_priv table that enables granting the PROXY privilege for ''@'', that is, for all users and all hosts. This enables root to set up proxy users, as well as to delegate to other accounts the authority to set up proxy users. See Proxy Users.

To assign a password for the initial MySQL root account, use the following procedure. Replace root-password in the examples with the password that you want to use.

Start the server if it is not running. For instructions, see Section 9.2, "Starting the Server".

The initial root account may or may not have a password. Choose whichever of the following procedures applies:

- If the root account exists with an initial random password that has been expired, connect to the server as root using that password, then choose a new password. This is the case if the data directory was initialized using mysqld --initialize, either manually or using an installer that does not give you the option of specifying a password during the install operation. Because the password exists, you must use it to connect to the server. But because the password is expired, you cannot use the account for any purpose other than to choose a new password, until you do choose one.
  - 1. If you do not know the initial random password, look in the server error log.
  - 2. Connect to the server as root using the password:

```
$> mysql -u root -p
Enter password: (enter the random root password here)
```

3. Choose a new password to replace the random password:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'root-password';
```

- If the root account exists but has no password, connect to the server as root using no password, then assign a password. This is the case if you initialized the data directory using mysqld --initialize-insecure.
  - 1. Connect to the server as root using no password:

```
$> mysql -u root --skip-password
```

2. Assign a password:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'root-password';
```

After assigning the root account a password, you must supply that password whenever you connect to the server using the account. For example, to connect to the server using the mysql client, use this command:

```
$> mysql -u root -p
Enter password: (enter root password here)
```

To shut down the server with mysqladmin, use this command:

```
$> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

#### Note

For additional information about setting passwords, see Assigning Account Passwords. If you forget your root password after setting it, see How to Reset the Root Password.

To set up additional accounts, see Adding Accounts, Assigning Privileges, and Dropping Accounts.

# 9.5 Starting and Stopping MySQL Automatically

This section discusses methods for starting and stopping the MySQL server.

Generally, you start the mysqld server in one of these ways:

- Invoke mysqld directly. This works on any platform.
- On Windows, you can set up a MySQL service that runs automatically when Windows starts. See Section 5.4.8, "Starting MySQL as a Windows Service".
- On Unix and Unix-like systems, you can invoke <code>mysqld\_safe</code>, which tries to determine the proper options for <code>mysqld</code> and then runs it with those options. See <code>mysqld\_safe</code> <code>MySQL</code> Server Startup Script.
- On Linux systems that support systemd, you can use it to control the server. See Section 7.9, "Managing MySQL Server with systemd".
- On systems that use System V-style run directories (that is, /etc/init.d and run-level specific directories), invoke mysql.server. This script is used primarily at system startup and shutdown. It usually is installed under the name mysql. The mysql.server script starts the server by invoking mysqld\_safe. See mysql.server MySQL Server Startup Script.
- On macOS, install a launchd daemon to enable automatic MySQL startup at system startup. The daemon starts the server by invoking mysqld\_safe. For details, see Section 6.3, "Installing and Using the MySQL Launch Daemon". A MySQL Preference Pane also provides control for starting and stopping MySQL through the System Preferences. See Section 6.4, "Installing and Using the MySQL Preference Pane".
- On Solaris, use the service management framework (SMF) system to initiate and control MySQL startup.

systemd, the mysqld\_safe and mysql.server scripts, Solaris SMF, and the macOS Startup Item (or MySQL Preference Pane) can be used to start the server manually, or automatically at system startup time. systemd, mysql.server, and the Startup Item also can be used to stop the server.

The following table shows which option groups the server and startup scripts read from option files.

**Table 9.1 MySQL Startup Scripts and Supported Server Option Groups** 

Script	Option Groups
mysqld	[mysqld], [server], [mysqld-major_version]
mysqld_safe	[mysqld],[server],[mysqld_safe]
mysql.server	[mysqld],[mysql.server],[server]

[mysqld-major\_version] means that groups with names like [mysqld-5.7] and [mysqld-8.0] are read by servers having versions 5.7.x, 8.0.x, and so forth. This feature can be used to specify options that can be read only by servers within a given release series.

For backward compatibility, <code>mysql.server</code> also reads the <code>[mysql\_server]</code> group and <code>mysqld\_safe</code> also reads the <code>[safe\_mysqld]</code> group. To be current, you should update your option files to use the <code>[mysql.server]</code> and <code>[mysqld\_safe]</code> groups instead.

For more information on MySQL configuration files and their structure and contents, see Using Option Files.

# Chapter 10 Upgrading MySQL

### **Table of Contents**

10.1 Before You Begin	195
10.2 Upgrade Paths	. 196
10.3 Upgrade Best Practices	
10.4 What the MySQL Upgrade Process Upgrades	. 199
10.5 Changes in MySQL 8.0	. 203
10.6 Preparing Your Installation for Upgrade	222
10.7 Upgrading MySQL Binary or Package-based Installations on Unix/Linux	225
10.8 Upgrading MySQL with the MySQL Yum Repository	. 230
10.9 Upgrading MySQL with the MySQL APT Repository	. 232
10.10 Upgrading MySQL with the MySQL SLES Repository	232
10.11 Upgrading MySQL on Windows	. 232
10.12 Upgrading a Docker Installation of MySQL	. 234
10.13 Upgrade Troubleshooting	
10.14 Rebuilding or Repairing Tables or Indexes	
10.15 Copying MySQL Databases to Another Machine	

This chapter describes the steps to upgrade a MySQL installation.

Upgrading is a common procedure, as you pick up bug fixes within the same MySQL release series or significant features between major MySQL releases. You perform this procedure first on some test systems to make sure everything works smoothly, and then on the production systems.

#### Note

In the following discussion, MySQL commands that must be run using a MySQL account with administrative privileges include -u root on the command line to specify the MySQL root user. Commands that require a password for root also include a -p option. Because -p is followed by no option value, such commands prompt for the password. Type the password when prompted and press Enter.

SQL statements can be executed using the mysql command-line client (connect as root to ensure that you have the necessary privileges).

# 10.1 Before You Begin

Review the information in this section before upgrading. Perform any recommended actions.

- Understand what may occur during an upgrade. See Section 10.4, "What the MySQL Upgrade Process Upgrades".
- Protect your data by creating a backup. The backup should include the mysql system database, which contains the MySQL data dictionary tables and system tables. See Database Backup Methods.

#### **Important**

Downgrade from MySQL 8.0 to MySQL 5.7, or from a MySQL 8.0 release to a previous MySQL 8.0 release, is not supported. The only supported alternative is to restore a backup taken *before* upgrading. It is therefore imperative that you back up your data before starting the upgrade process.

- Review Section 10.2, "Upgrade Paths" to ensure that your intended upgrade path is supported.
- Review Section 10.5, "Changes in MySQL 8.0" for changes that you should be aware of before upgrading. Some changes may require action.
- Review What Is New in MySQL 8.0 for deprecated and removed features. An upgrade may require changes with respect to those features if you use any of them.
- Review Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.0. If you use deprecated or removed variables, an upgrade may require configuration changes.
- Review the Release Notes for information about fixes, changes, and new features.
- If you use replication, review Upgrading a Replication Topology.
- Review Section 10.3, "Upgrade Best Practices" and plan accordingly.
- Upgrade procedures vary by platform and how the initial installation was performed. Use the procedure that applies to your current MySQL installation:
  - For binary and package-based installations on non-Windows platforms, refer to Section 10.7, "Upgrading MySQL Binary or Package-based Installations on Unix/Linux".

#### Note

For supported Linux distributions, the preferred method for upgrading package-based installations is to use the MySQL software repositories (MySQL Yum Repository, MySQL APT Repository, and MySQL SLES Repository).

- For installations on an Enterprise Linux platform or Fedora using the MySQL Yum Repository, refer to Section 10.8, "Upgrading MySQL with the MySQL Yum Repository".
- For installations on Ubuntu using the MySQL APT repository, refer to Section 10.9, "Upgrading MySQL with the MySQL APT Repository".
- For installations on SLES using the MySQL SLES repository, refer to Section 10.10, "Upgrading MySQL with the MySQL SLES Repository".
- For installations performed using Docker, refer to Section 10.12, "Upgrading a Docker Installation of MySQL".
- For installations on Windows, refer to Section 10.11, "Upgrading MySQL on Windows".
- If your MySQL installation contains a large amount of data that might take a long time to convert after an in-place upgrade, it may be useful to create a test instance for assessing the conversions that are required and the work involved to perform them. To create a test instance, make a copy of your MySQL instance that contains the mysql database and other databases without the data. Run the upgrade procedure on the test instance to assess the work involved to perform the actual data conversion.
- Rebuilding and reinstalling MySQL language interfaces is recommended when you install or upgrade to a new release of MySQL. This applies to MySQL interfaces such as PHP mysql extensions and the Perl DBD::mysql module.

# 10.2 Upgrade Paths

 Upgrade from MySQL 5.7 to 8.0 is supported. However, upgrade is only supported between General Availability (GA) releases. For MySQL 8.0, it is required that you upgrade from a MySQL 5.7 GA release (5.7.9 or higher). Upgrades from non-GA releases of MySQL 5.7 are not supported.

- Upgrading to the latest release is recommended before upgrading to the next version. For example, upgrade to the latest MySQL 5.7 release before upgrading to MySQL 8.0.
- Upgrade that skips versions is not supported. For example, upgrading directly from MySQL 5.6 to 8.0 is not supported.
- Once a release series reaches General Availability (GA) status, upgrade within the release series (from one GA version to another GA version) is supported. For example, upgrading from MySQL 8.0.x to 8.0.y is supported. (Upgrade involving development-status non-GA releases is not supported.) Skipping a release is also supported. For example, upgrading from MySQL 8.0.x to 8.0.z is supported. MySQL 8.0.11 is the first GA status release within the MySQL 8.0 release series.

# 10.3 Upgrade Best Practices

MySQL supports upgrading between minor versions (within an LTS series) and to the next major version (across an LTS series). Upgrading provides the latest features, performance, and security fixes.

To prepare and help ensure that your upgrade to the latest MySQL release is successful, we recommend the following best practices:

- · Decide on Major or Minor Version for Upgrade
- Decide on Upgrade Type
- Review Supported Platforms
- Understand MySQL Server Changes
- · Run Upgrade Checker and Fix Incompatibilities
- Run Applications in a Test Environment
- Benchmark Applications and Workload Performance
- · Run Both MySQL Versions in Parallel
- · Run Final Test Upgrade
- Check MySQL Backup
- Upgrade Production Server
- Enterprise Support

## **Decide on Major or Minor Version for Upgrade**

The MySQL Release Model makes a distinction between LTS (Long Term Support) and Innovation Releases. LTS releases have 8+ years of support and are meant for production use. Innovation Releases provide users with the latest features and capabilities. Learn more about the MySQL Release Model.

Performing a minor version upgrade is straightforward while major version upgrades require strategic planning and additional testing before the upgrade. This guide is especially useful for major version upgrades.

## **Decide on Upgrade Type**

There are three main ways to upgrade MySQL, read the associated documentation to determine which type of upgrade is best suited for your situation.

- An in-place upgrade: Replacing the MySQL Server packages.
- A logical upgrade: exporting SQL from the old MySQL instance to the new.
- A replication topology upgrade: account for each server's topology role.

### **Review Supported Platforms**

If your current operating system is not supported by the new version of MySQL, then plan to upgrade the operating system as otherwise an in-place upgrade is not supported.

For a current list of supported platforms, see: https://www.mysql.com/support/supportedplatforms/database.html

### **Understand MySQL Server Changes**

Each major version comes with new features, changes in behavior, deprecations, and removals. It is important to understand the impact of each of these to existing applications.

See: Section 10.5, "Changes in MySQL 8.0".

### Run Upgrade Checker and Fix Incompatibilities

MySQL Shell's Upgrade Checker Utility detects incompatibilities between database versions that must be addressed before performing the upgrade. The util.checkForServerUpgrade() function verifies that MySQL server instances are ready to upgrade. Connect to the existing MySQL server and select the MySQL Server version you plan to upgrade to for the utility to report issues to address prior to an upgrade. These include incompatibilities in data types, storage engines, and so on.

You are ready to upgrade when the upgrade checking utility no longer reports any issues.

## **Run Applications in a Test Environment**

After completing the upgrade checker's requirements, next test your applications on the new target MySQL server. Check for errors and warnings in the MySQL error log and application logs.

## **Benchmark Applications and Workload Performance**

We recommend benchmarking your own applications and workloads by comparing how they perform using the previous and new versions of MySQL. Usually, newer MySQL versions add features and improve performance but there are cases where an upgrade might run slower for specific queries. Possible issues resulting in performance regressions:

- Prior server configuration is not optimal for newer version
- Changes to data types
- Additional storage required by Multi-byte character set support
- Storage engines changes
- · Dropped or changed indexes
- Stronger encryption

- Stronger authentication
- SQL optimizer changes
- · Newer version of MySQL require additional memory
- Physical or Virtual Hardware is slower compute or storage

For related information and potential mitigation techniques, see Valid Performance Regressions.

### **Run Both MySQL Versions in Parallel**

To minimize risk, it is best keep the current system running while running the upgraded system in parallel.

### **Run Final Test Upgrade**

Practice and do a run though prior to upgrading your production server. Thoroughly test the upgrade procedures before upgrading a production system.

## **Check MySQL Backup**

Check that the full backup exists and is viable before performing the upgrade.

### **Upgrade Production Server**

You are ready to complete the upgrade.

## **Enterprise Support**

If you're a MySQL Enterprise Edition customer, you can also contact the MySQL Support Team experts with any questions you may have.

# 10.4 What the MySQL Upgrade Process Upgrades

Installing a new version of MySQL may require upgrading these parts of the existing installation:

- The mysql system schema, which contains tables that store information required by the MySQL server
  as it runs (see The mysql System Schema). mysql schema tables fall into two broad categories:
  - Data dictionary tables, which store database object metadata.
  - System tables (that is, the remaining non-data dictionary tables), which are used for other operational purposes.
- Other schemas, some of which are built in and may be considered "owned" by the server, and others which are not:
  - The performance\_schema, INFORMATION\_SCHEMA, ndbinfo, and sys schemas.
  - · User schemas.

Two distinct version numbers are associated with parts of the installation that may require upgrading:

The data dictionary version. This applies to the data dictionary tables.

 The server version, also known as the MySQL version. This applies to the system tables and objects in other schemas.

In both cases, the actual version applicable to the existing MySQL installation is stored in the data dictionary, and the current expected version is compiled into the new version of MySQL. When an actual version is lower than the current expected version, those parts of the installation associated with that version must be upgraded to the current version. If both versions indicate an upgrade is needed, the data dictionary upgrade must occur first.

As a reflection of the two distinct versions just mentioned, the upgrade occurs in two steps:

Step 1: Data dictionary upgrade.

This step upgrades:

- The data dictionary tables in the mysql schema. If the actual data dictionary version is lower than the current expected version, the server creates data dictionary tables with updated definitions, copies persisted metadata to the new tables, atomically replaces the old tables with the new ones, and reinitializes the data dictionary.
- The Performance Schema, INFORMATION\_SCHEMA, and ndbinfo.
- Step 2: Server upgrade.

This step comprises all other upgrade tasks. If the server version of the existing MySQL installation is lower than that of the new installed MySQL version, everything else must be upgraded:

- The system tables in the mysql schema (the remaining non-data dictionary tables).
- The sys schema.
- · User schemas.

The data dictionary upgrade (step 1) is the responsibility of the server, which performs this task as necessary at startup unless invoked with an option that prevents it from doing so. The option is --upgrade=NONE as of MySQL 8.0.16, --no-dd-upgrade prior to MySQL 8.0.16.

If the data dictionary is out of date but the server is prevented from upgrading it, the server does not run, and exits with an error instead. For example:

```
[ERROR] [MY-013381] [Server] Server shutting down because upgrade is required, yet prohibited by the command line option '--upgrade=NONE'.
[ERROR] [MY-010334] [Server] Failed to initialize DD Storage Engine
[ERROR] [MY-010020] [Server] Data Dictionary initialization failed.
```

Some changes to the responsibility for step 2 occurred in MySQL 8.0.16:

- Prior to MySQL 8.0.16, mysql\_upgrade upgrades the Performance Schema, the INFORMATION\_SCHEMA, and the objects described in step 2. The DBA is expected to invoke mysql\_upgrade manually after starting the server.
- As of MySQL 8.0.16, the server performs all tasks previously handled by mysql\_upgrade. Although upgrading remains a two-step operation, the server performs them both, resulting in a simpler process.

Depending on the version of MySQL to which you are upgrading, the instructions in In-Place Upgrade and Logical Upgrade indicate whether the server performs all upgrade tasks or whether you must also invoke <code>mysql\_upgrade</code> after server startup.

#### Note

Because the server upgrades the Performance Schema, INFORMATION\_SCHEMA, and the objects described in step 2 as of MySQL 8.0.16, mysql\_upgrade is unneeded and is deprecated as of that version; expect it to be removed in a future version of MySQL.

Most aspects of what occurs during step 2 are the same prior to and as of MySQL 8.0.16, although different command options may be needed to achieve a particular effect.

As of MySQL 8.0.16, the --upgrade server option controls whether and how the server performs an automatic upgrade at startup:

- With no option or with --upgrade=AUTO, the server upgrades anything it determines to be out of date (steps 1 and 2).
- With --upgrade=NONE, the server upgrades nothing (skips steps 1 and 2), but also exits with an error if the data dictionary must be upgraded. It is not possible to run the server with an out-of-date data dictionary; the server insists on either upgrading it or exiting.
- With --upgrade=MINIMAL, the server upgrades the data dictionary, the Performance Schema, and the INFORMATION\_SCHEMA, if necessary (step 1). Note that following an upgrade with this option, Group Replication cannot be started, because system tables on which the replication internals depend are not updated, and reduced functionality might also be apparent in other areas.
- With --upgrade=FORCE, the server upgrades the data dictionary, the Performance Schema, and the INFORMATION\_SCHEMA, if necessary (step 1), and forces an upgrade of everything else (step 2). Expect server startup to take longer with this option because the server checks all objects in all schemas.

FORCE is useful to force step 2 actions to be performed if the server thinks they are not necessary. One way that FORCE differs from AUTO is that with FORCE, the server re-creates system tables such as help tables or time zone tables if they are missing.

The following list shows upgrade commands prior to MySQL 8.0.16 and the equivalent commands for MySQL 8.0.16 and higher:

- Perform a normal upgrade (steps 1 and 2 as necessary):
  - Prior to MySQL 8.0.16: mysqld followed by mysql\_upgrade
  - As of MySQL 8.0.16: mysqld
- · Perform only step 1 as necessary:
  - Prior to MySQL 8.0.16: It is not possible to perform all upgrade tasks described in step 1 while
    excluding those described in step 2. However, you can avoid upgrading user schemas and the sys
    schema using mysqld followed by mysql\_upgrade with the --upgrade-system-tables and -skip-sys-schema options.
  - As of MySQL 8.0.16: mysqld --upgrade=MINIMAL
- Perform step 1 as necessary, and force step 2:
  - Prior to MySQL 8.0.16: mysqld followed by mysql\_upgrade --force
  - As of MySQL 8.0.16: mysqld --upgrade=FORCE

Prior to MySQL 8.0.16, certain mysql\_upgrade options affect the actions it performs. The following table shows which server --upgrade option values to use as of MySQL 8.0.16 to achieve similar effects.

(These are not necessarily exact equivalents because a given --upgrade option value may have additional effects.)

mysql_upgrade Option	Server Option
skip-sys-schema	upgrade=NONE <b>or</b> upgrade=MINIMAL
upgrade-system-tables	upgrade=NONE <b>or</b> upgrade=MINIMAL
force	upgrade=FORCE

Additional notes about what occurs during upgrade step 2:

• Step 2 installs the sys schema if it is not installed, and upgrades it to the current version otherwise. An error occurs if a sys schema exists but has no version view, on the assumption that its absence indicates a user-created schema:

```
A sys schema exists with no sys.version view. If you have a user created sys schema, this must be renamed for the upgrade to succeed.
```

To upgrade in this case, remove or rename the existing sys schema first. Then perform the upgrade procedure again. (It may be necessary to force step 2.)

To prevent the sys schema check:

- As of MySQL 8.0.16: Start the server with the --upgrade=NONE or --upgrade=MINIMAL option.
- Prior to MySQL 8.0.16: Invoke mysql\_upgrade with the --skip-sys-schema option.
- Step 2 upgrades the system tables to ensure that they have the current structure. This is true whether the server or mysql\_upgrade performs the step. With respect to the content of the help tables and time zone tables, mysql\_upgrade does not load either type of table, whereas the server loads the help tables, but not the time zone tables. (That is, prior to MySQL 8.0.16, the server loads the help tables only at data directory initialization time. As of MySQL 8.0.16, it loads the help tables at initialization and upgrade time.) The procedure for loading time zone tables is platform dependent and requires decision making by the DBA, so it cannot be done automatically.
- From MySQL 8.0.30, when Step 2 is upgrading the system tables in the mysql schema, the column order in the primary key of the mysql.db, mysql.tables\_priv, mysql.columns\_priv and mysql.procs\_priv tables is changed to place the host name and user name columns together. Placing the host name and user name together means that index lookup can be used, which improves performance for CREATE USER, DROP USER, and RENAME USER statements, and for ACL checks for multiple users with multiple privileges. Dropping and re-creating the index is necessary and might take some time if the system has a large number of users and privileges.
- Step 2 processes all tables in all user schemas as necessary. Table checking might take a long time to
  complete. Each table is locked and therefore unavailable to other sessions while it is being processed.
  Check and repair operations can be time-consuming, particularly for large tables. Table checking uses
  the FOR UPGRADE option of the CHECK TABLE statement. For details about what this option entails, see
  CHECK TABLE Statement.

To prevent table checking:

- As of MySQL 8.0.16: Start the server with the --upgrade=NONE or --upgrade=MINIMAL option.
- Prior to MySQL 8.0.16: Invoke mysql\_upgrade with the --upgrade-system-tables option.

To force table checking:

- As of MySQL 8.0.16: Start the server with the --upgrade=FORCE option.
- Prior to MySQL 8.0.16: Invoke mysql\_upgrade with the --force option.
- Step 2 saves the MySQL version number in a file named mysql\_upgrade\_info in the data directory.

To ignore the mysql\_upgrade\_info file and perform the check regardless:

- As of MySQL 8.0.16: Start the server with the --upgrade=FORCE option.
- Prior to MySQL 8.0.16: Invoke mysql\_upgrade with the --force option.

#### Note

The mysql\_upgrade\_info file is deprecated; expect it to be removed in a future version of MySQL.

Step 2 marks all checked and repaired tables with the current MySQL version number. This ensures that
the next time upgrade checking occurs with the same version of the server, it can be determined whether
there is any need to check or repair a given table again.

## 10.5 Changes in MySQL 8.0

Before upgrading to MySQL 8.0, review the changes described in this section to identify those that apply to your current MySQL installation and applications. Perform any recommended actions.

Changes marked as **Incompatible change** are incompatibilities with earlier versions of MySQL, and may require your attention *before upgrading*. Our aim is to avoid these changes, but occasionally they are necessary to correct problems that would be worse than an incompatibility between releases. If an upgrade issue applicable to your installation involves an incompatibility, follow the instructions given in the description.

- Data Dictionary Changes
- · caching\_sha2\_password as the Preferred Authentication Plugin
- · Configuration Changes
- Server Changes
- InnoDB Changes
- SQL Changes
- Changed Server Defaults
- · Valid Performance Regressions

## **Data Dictionary Changes**

MySQL Server 8.0 incorporates a global data dictionary containing information about database objects in transactional tables. In previous MySQL series, dictionary data was stored in metadata files and nontransactional system tables. As a result, the upgrade procedure requires that you verify the upgrade readiness of your installation by checking specific prerequisites. For more information, see Section 10.6, "Preparing Your Installation for Upgrade". A data dictionary-enabled server entails some general operational differences; see Data Dictionary Usage Differences.

### caching\_sha2\_password as the Preferred Authentication Plugin

The caching\_sha2\_password and sha256\_password authentication plugins provide more secure password encryption than the mysql\_native\_password plugin, and caching\_sha2\_password provides better performance than sha256\_password. Due to these superior security and performance characteristics of caching\_sha2\_password, it is as of MySQL 8.0 the preferred authentication plugin, and is also the default authentication plugin rather than mysql\_native\_password. This change affects both the server and the libmysqlclient client library:

• For the server, the default value of the default\_authentication\_plugin system variable changes from mysgl\_native\_password to caching\_sha2\_password.

This change applies only to new accounts created after installing or upgrading to MySQL 8.0 or higher. For accounts already existing in an upgraded installation, their authentication plugin remains unchanged. Existing users who wish to switch to <a href="mailto:caching\_sha2\_password">caching\_sha2\_password</a> can do so using the ALTER USER statement:

```
ALTER USER user

IDENTIFIED WITH caching_sha2_password

BY 'password';
```

• The libmysqlclient library treats caching\_sha2\_password as the default authentication plugin rather than mysql\_native\_password.

The following sections discuss the implications of the more prominent role of caching sha2 password:

- caching\_sha2\_password Compatibility Issues and Solutions
- caching\_sha2\_password-Compatible Clients and Connectors
- caching\_sha2\_password and the root Administrative Account
- caching\_sha2\_password and Replication

### caching\_sha2\_password Compatibility Issues and Solutions

#### **Important**

If your MySQL installation must serve pre-8.0 clients and you encounter compatibility issues after upgrading to MySQL 8.0 or higher, the simplest way to address those issues and restore pre-8.0 compatibility is to reconfigure the server to revert to the previous default authentication plugin (mysql\_native\_password). For example, use these lines in the server option file:

```
[mysqld]
default_authentication_plugin=mysql_native_password
```

That setting enables pre-8.0 clients to connect to 8.0 servers until such time as the clients and connectors in use at your installation are upgraded to know about caching\_sha2\_password. However, the setting should be viewed as temporary, not as a long term or permanent solution, because it causes new accounts created with the setting in effect to forego the improved authentication security provided by caching sha2 password.

The use of caching\_sha2\_password offers more secure password hashing than mysql\_native\_password (and consequent improved client connection authentication). However, it also has compatibility implications that may affect existing MySQL installations:

• Clients and connectors that have not been updated to know about <a href="caching\_sha2\_password">caching\_sha2\_password</a> may have trouble connecting to a MySQL 8.0 server configured with <a href="caching\_sha2\_password">caching\_sha2\_password</a> as the default authentication plugin, even to use accounts that do not authenticate with <a href="caching\_sha2\_password">caching\_sha2\_password</a>. This issue occurs because the server specifies the name of its default authentication plugin to clients. If a client or connector is based on a client/server protocol implementation that does not gracefully handle an unrecognized default authentication plugin, it may fail with an error such as one of these:

```
Authentication plugin 'caching_sha2_password' is not supported

Authentication plugin 'caching_sha2_password' cannot be loaded:
dlopen(/usr/local/mysql/lib/plugin/caching_sha2_password.so, 2):
image not found

Warning: mysqli_connect(): The server requested authentication
method unknown to the client [caching_sha2_password]
```

For information about writing connectors to gracefully handle requests from the server for unknown default authentication plugins, see Authentication Plugin Connector-Writing Considerations.

- Clients that use an account that authenticates with <aching\_sha2\_password must use either a secure connection (made using TCP using TLS/SSL credentials, a Unix socket file, or shared memory), or an unencrypted connection that supports password exchange using an RSA key pair. This security requirement does not apply to <a href="maysq1\_native\_passsword">maysq1\_native\_passsword</a>, so the switch to <a href="maysqa2\_password">caching\_sha2\_password</a> may require additional configuration (see Caching SHA-2 Pluggable Authentication). However, client connections in MySQL 8.0 prefer use of TLS/SSL by default, so clients that already conform to that preference may need no additional configuration.
- Clients and connectors that have not been updated to know about caching\_sha2\_password cannot connect to accounts that authenticate with caching\_sha2\_password because they do not recognize this plugin as valid. (This is a particular instance of how client/server authentication plugin compatibility requirements apply, as discussed at Authentication Plugin Client/Server Compatibility.) To work around this issue, relink clients against libmysqlclient from MySQL 8.0 or higher, or obtain an updated connector that recognizes caching\_sha2\_password.
- Because caching\_sha2\_password is also now the default authentication plugin in the libmysqlclient client library, authentication requires an extra round trip in the client/server protocol for connections from MySQL 8.0 clients to accounts that use mysql\_native\_password (the previous default authentication plugin), unless the client program is invoked with a --default-auth=mysql\_native\_password option.

The libmysqlclient client library for pre-8.0 MySQL versions is able to connect to MySQL 8.0 servers (except for accounts that authenticate with caching\_sha2\_password). That means pre-8.0 clients based on libmysqlclient should also be able to connect. Examples:

- Standard MySQL clients such as mysql and mysqladmin are libmysqlclient-based.
- The DBD::mysql driver for Perl DBI is libmysqlclient-based.
- MySQL Connector/Python has a C Extension module that is libmysqlclient-based. To use it, include the use\_pure=False option at connect time.

When an existing MySQL 8.0 installation is upgraded to MySQL 8.0.4 or higher, some older <code>libmysqlclient</code>-based clients may "automatically" upgrade if they are dynamically linked, because they use the new client library installed by the upgrade. For example, if the DBD::mysql driver for Perl DBI uses dynamic linking, it can use the <code>libmysqlclient</code> in place after an upgrade to MySQL 8.0.4 or higher, with this result:

- Prior to the upgrade, DBI scripts that use DBD::mysql can connect to a MySQL 8.0 server, except for accounts that authenticate with caching sha2 password.
- After the upgrade, the same scripts become able to use caching\_sha2\_password accounts as well.

However, the preceding results occur because libmysqlclient instances from MySQL 8.0 installations prior to 8.0.4 are binary compatible: They both use a shared library major version number of 21. For clients linked to libmysqlclient from MySQL 5.7 or older, they link to a shared library with a different version number that is not binary compatible. In this case, the client must be recompiled against libmysqlclient from 8.0.4 or higher for full compatibility with MySQL 8.0 servers and caching\_sha2\_password accounts.

MySQL Connector/J 5.1 through 8.0.8 is able to connect to MySQL 8.0 servers, except for accounts that authenticate with caching\_sha2\_password. (Connector/J 8.0.9 or higher is required to connect to caching\_sha2\_password accounts.)

Clients that use an implementation of the client/server protocol other than libmysqlclient may need to be upgraded to a newer version that understands the new authentication plugin. For example, in PHP, MySQL connectivity usually is based on mysqlnd, which currently does not know about caching\_sha2\_password. Until an updated version of mysqlnd is available, the way to enable PHP clients to connect to MySQL 8.0 is to reconfigure the server to revert to mysql\_native\_password as the default authentication plugin, as previously discussed.

If a client or connector supports an option to explicitly specify a default authentication plugin, use it to name a plugin other than caching\_sha2\_password. Examples:

- Some MySQL clients support a --default-auth option. (Standard MySQL clients such as mysql and mysqladmin support this option but can successfully connect to 8.0 servers without it. However, other clients may support a similar option. If so, it is worth trying it.)
- Programs that use the libmysqlclient C API can call the mysql\_options() function with the MYSQL\_DEFAULT\_AUTH option.
- MySQL Connector/Python scripts that use the native Python implementation of the client/server protocol can specify the auth\_plugin connection option. (Alternatively, use the Connector/Python C Extension, which is able to connect to MySQL 8.0 servers without the need for auth plugin.)

### caching sha2 password-Compatible Clients and Connectors

If a client or connector is available that has been updated to know about caching\_sha2\_password, using it is the best way to ensure compatibility when connecting to a MySQL 8.0 server configured with caching sha2 password as the default authentication plugin.

These clients and connectors have been upgraded to support caching sha2 password:

- The libmysqlclient client library in MySQL 8.0 (8.0.4 or higher). Standard MySQL clients such as mysql and mysqladmin are libmysqlclient-based, so they are compatible as well.
- The libmysqlclient client library in MySQL 5.7 (5.7.23 or higher). Standard MySQL clients such as mysql and mysqladmin are libmysqlclient-based, so they are compatible as well.
- MySQL Connector/C++ 1.1.11 or higher or 8.0.7 or higher.
- MySQL Connector/J 8.0.9 or higher.
- MySQL Connector/NET 8.0.10 or higher (through the classic MySQL protocol).
- MySQL Connector/Node.js 8.0.9 or higher.

• PHP: the X DevAPI PHP extension (mysql\_xdevapi) supports caching\_sha2\_password.

PHP: the PDO\_MySQL and ext/mysqli extensions do not support caching\_sha2\_password. In addition, when used with PHP versions before 7.1.16 and PHP 7.2 before 7.2.4, they fail to connect with default\_authentication\_plugin=caching\_sha2\_password even if caching\_sha2\_password is not used.

### caching\_sha2\_password and the root Administrative Account

For upgrades to MySQL 8.0, the authentication plugin existing accounts remains unchanged, including the plugin for the 'root'@'localhost' administrative account.

For new MySQL 8.0 installations, when you initialize the data directory (using the instructions at Section 9.1, "Initializing the Data Directory"), the 'root'@'localhost' account is created, and that account uses caching\_sha2\_password by default. To connect to the server following data directory initialization, you must therefore use a client or connector that supports caching\_sha2\_password. If you can do this but prefer that the root account use mysql\_native\_password after installation, install MySQL and initialize the data directory as you normally would. Then connect to the server as root and use ALTER USER as follows to change the account authentication plugin and password:

```
ALTER USER 'root'@'localhost'

IDENTIFIED WITH mysql_native_password

BY 'password';
```

If the client or connector that you use does not yet support caching\_sha2\_password, you can use a modified data directory-initialization procedure that associates the root account with mysql native password as soon as the account is created. To do so, use either of these techniques:

- Supply a --default-authentication-plugin=mysql\_native\_password option along with -initialize or --initialize-insecure.
- Set default\_authentication\_plugin to mysql\_native\_password in an option file, and name that option file using a --defaults-file option along with --initialize or --initialize-insecure. (In this case, if you continue to use that option file for subsequent server startups, new accounts are created with mysql\_native\_password rather than caching\_sha2\_password unless you remove the default\_authentication\_plugin setting from the option file.)

### caching sha2 password and Replication

In replication scenarios for which all servers have been upgraded to MySQL 8.0.4 or higher, replica connections to source servers can use accounts that authenticate with <aching\_sha2\_password. For such connections, the same requirement applies as for other clients that use accounts that authenticate with <aching\_sha2\_password: Use a secure connection or RSA-based password exchange.

To connect to a caching sha2 password account for source/replica replication:

Use any of the following CHANGE MASTER TO options:

```
MASTER_SSL = 1
GET_MASTER_PUBLIC_KEY = 1
MASTER_PUBLIC_KEY_PATH='path to RSA public key file'
```

 Alternatively, you can use the RSA public key-related options if the required keys are supplied at server startup.

To connect to a caching\_sha2\_password account for Group Replication:

For MySQL built using OpenSSL, set any of the following system variables:

```
SET GLOBAL group_replication_recovery_use_ssl = ON;
SET GLOBAL group_replication_recovery_get_public_key = 1;
SET GLOBAL group_replication_recovery_public_key_path = 'path to RSA public key file';
```

 Alternatively, you can use the RSA public key-related options if the required keys are supplied at server startup.

### **Configuration Changes**

• **Incompatible change**: A MySQL storage engine is now responsible for providing its own partitioning handler, and the MySQL server no longer provides generic partitioning support. **InnoDB** and **NDB** are the only storage engines that provide a native partitioning handler that is supported in MySQL 8.0. A partitioned table using any other storage engine must be altered—either to convert it to **InnoDB** or **NDB**, or to remove its partitioning—before upgrading the server, else it cannot be used afterwards.

For information about converting MyISAM tables to InnoDB, see Converting Tables from MyISAM to InnoDB.

A table creation statement that would result in a partitioned table using a storage engine without such support fails with an error (ER\_CHECK\_NOT\_IMPLEMENTED) in MySQL 8.0. If you import databases from a dump file created in MySQL 5.7 (or earlier) using mysqldump into a MySQL 8.0 server, you must make sure that any statements creating partitioned tables do not also specify an unsupported storage engine, either by removing any references to partitioning, or by specifying the storage engine as InnoDB or allowing it to be set as InnoDB by default.

#### Note

The procedure given at Section 10.6, "Preparing Your Installation for Upgrade", describes how to identify partitioned tables that must be altered before upgrading to MySQL 8.0.

See Partitioning Limitations Relating to Storage Engines, for further information.

- **Incompatible change**: Several server error codes are not used and have been removed (for a list, see Features Removed in MySQL 8.0). Applications that test specifically for any of them should be updated.
- **Important change**: The default character set has changed from latin1 to utf8mb4. These system variables are affected:
  - The default value of the character\_set\_server and character\_set\_database system variables has changed from latin1 to utf8mb4.
  - The default value of the collation\_server and collation\_database system variables has changed from latin1\_swedish\_ci to utf8mb4\_0900\_ai\_ci.

As a result, the default character set and collation for new objects differ from previously unless an explicit character set and collation are specified. This includes databases and objects within them, such as tables, views, and stored programs. Assuming that the previous defaults were used, one way to preserve them is to start the server with these lines in the my.cnf file:

```
[mysqld]
character_set_server=latin1
collation_server=latin1_swedish_ci
```

In a replicated setting, when upgrading from MySQL 5.7 to 8.0, it is advisable to change the default character set back to the character set used in MySQL 5.7 before upgrading. After the upgrade is completed, the default character set can be changed to utf8mb4.

In addition, you should be aware that MySQL 8.0 enforces checks on permitted characters in a given character set which MySQL 5.7 does not; this is a known issue. This means that, prior to attempting to upgrade, you should ensure that no comments contain characters that are not defined for the character set in use. You can fix this in either of two ways:

- Change the character set to one that includes the character or characters in question.
- Remove the offending character or characters.

The preceding applies to table, file, and index comments.

• Incompatible change: As of MySQL 8.0.11, it is prohibited to start the server with a lower\_case\_table\_names setting that is different from the setting used when the server was initialized. The restriction is necessary because collations used by various data dictionary table fields are based on the lower\_case\_table\_names setting that was defined when the server was initialized, and restarting the server with a different setting would introduce inconsistencies with respect to how identifiers are ordered and compared.

### **Server Changes**

 In MySQL 8.0.11, several deprecated features related to account management have been removed, such as use of the GRANT statement to modify nonprivilege characteristics of user accounts, the NO\_AUTO\_CREATE\_USER SQL mode, the PASSWORD() function, and the old\_passwords system variable.

Replication from MySQL 5.7 to 8.0 of statements that refer to these removed features can cause replication failure. Applications that use any of the removed features should be revised to avoid them and use alternatives when possible, as described in Features Removed in MySQL 8.0.

To avoid a startup failure on MySQL 8.0, remove any instance of NO\_AUTO\_CREATE\_USER from sql mode system variable settings in MySQL option files.

Loading a dump file that includes the NO\_AUTO\_CREATE\_USER SQL mode in stored program definitions into a MySQL 8.0 server causes a failure. As of MySQL 5.7.24 and MySQL 8.0.13, mysqldump removes NO\_AUTO\_CREATE\_USER from stored program definitions. Dump files created with an earlier version of mysqldump must be modified manually to remove instances of NO\_AUTO\_CREATE\_USER.

In MySQL 8.0.11, these deprecated compatibility SQL modes were removed: DB2, MAXDB,
 MSSQL, MYSQL323, MYSQL40, ORACLE, POSTGRESQL, NO\_FIELD\_OPTIONS, NO\_KEY\_OPTIONS,
 NO\_TABLE\_OPTIONS. They can no longer be assigned to the sql\_mode system variable or used as
 permitted values for the mysqldump --compatible option.

Removal of MAXDB means that the TIMESTAMP data type for CREATE TABLE or ALTER TABLE is no longer treated as DATETIME.

Replication from MySQL 5.7 to 8.0 of statements that refer to the removed SQL modes can cause replication failure. This includes replication of CREATE statements for stored programs (stored procedures and functions, triggers, and events) that are executed while the current sql\_mode value includes any of the removed modes. Applications that use any of the removed modes should be revised to avoid them.

• The text of many MySQL 8.0 error messages has been revised and improved to provide more and better information than in MySQL 5.7. If your application depends on specific content or formatting of error messages, you should test these and be prepared to update the application accordingly prior to performing an upgrade.

• As of MySQL 8.0.3, spatial data types permit an SRID attribute, to explicitly indicate the spatial reference system (SRS) for values stored in the column. See Spatial Data Types.

A spatial column with an explicit SRID attribute is SRID-restricted: The column takes only values with that ID, and SPATIAL indexes on the column become subject to use by the optimizer. The optimizer ignores SPATIAL indexes on spatial columns with no SRID attribute. See SPATIAL Index Optimization. If you want the optimizer to consider SPATIAL indexes on spatial columns that are not SRID-restricted, each such column should be modified:

• Verify that all values within the column have the same SRID. To determine the SRIDs contained in a geometry column col\_name, use the following query:

```
SELECT DISTINCT ST_SRID(col_name) FROM tbl_name;
```

If the query returns more than one row, the column contains a mix of SRIDs. In that case, modify its contents so all values have the same SRID.

- Redefine the column to have an explicit SRID attribute.
- Recreate the SPATIAL index.
- Several spatial functions were removed in MySQL 8.0.0 due to a spatial function namespace change
  that implemented an ST\_ prefix for functions that perform an exact operation, or an MBR prefix for
  functions that perform an operation based on minimum bounding rectangles. The use of removed
  spatial functions in generated column definitions could cause an upgrade failure. Before upgrading, run
  mysqlcheck --check-upgrade for removed spatial functions and replace any that you find with their
  ST\_ or MBR named replacements. For a list of removed spatial functions, refer to Features Removed in
  MySQL 8.0.
- The BACKUP\_ADMIN privilege is automatically granted to users with the RELOAD privilege when performing an in-place upgrade to MySQL 8.0.3 or higher.
- From MySQL 8.0.13, because of differences between row-based or mixed replication mode and statement-based replication mode in the way that temporary tables are handled, there are new restrictions on switching the binary logging format at runtime.
  - SET @@SESSION.binlog format cannot be used if the session has any open temporary tables.
  - SET @@global.binlog\_format and SET @@persist.binlog\_format cannot be used if any replication channel has any open temporary tables. SET @@persist\_only.binlog\_format is allowed if replication channels have open temporary tables, because unlike PERSIST, PERSIST\_ONLY does not modify the runtime global system variable value.
  - SET @@global.binlog\_format and SET @@persist.binlog\_format cannot be used if any replication channel applier is running. This is because the change only takes effect on a replication channel when its applier is restarted, at which time the replication channel might have open temporary tables. This behavior is more restrictive than before. SET @@persist\_only.binlog\_format is allowed if any replication channel applier is running.
  - From MySQL 8.0.27, configuring a session setting for internal\_tmp\_mem\_storage\_engine requires the SESSION\_VARIABLES\_ADMIN or SYSTEM\_VARIABLES\_ADMIN privilege.
  - As of MySQL 8.0.27, the clone plugin permits concurrent DDL operations on the donor MySQL Server instance while a cloning operation is in progress. Previously, a backup lock was held during the cloning operation, preventing concurrent DDL on the donor. To revert to the previous behavior of blocking concurrent DDL on the donor during a clone operation, enable the clone\_block\_ddl variable. See Cloning and Concurrent DDL.

• From MySQL 8.0.30, error log components listed in the log\_error\_services value at startup are loaded implicitly early in the MySQL Server startup sequence. If you have previously installed loadable error log components using INSTALL COMPONENT and you list those components in a log\_error\_services setting that is read at startup (from an option file, for example), your configuration should be updated to avoid startup warnings. For more information, see Error Log Configuration Methods.

### **InnoDB Changes**

• INFORMATION\_SCHEMA views based on InnoDB system tables were replaced by internal system views on data dictionary tables. Affected InnoDB INFORMATION\_SCHEMA views were renamed:

**Table 10.1 Renamed InnoDB Information Schema Views** 

Old Name	New Name
INNODB_SYS_COLUMNS	INNODB_COLUMNS
INNODB_SYS_DATAFILES	INNODB_DATAFILES
INNODB_SYS_FIELDS	INNODB_FIELDS
INNODB_SYS_FOREIGN	INNODB_FOREIGN
INNODB_SYS_FOREIGN_COLS	INNODB_FOREIGN_COLS
INNODB_SYS_INDEXES	INNODB_INDEXES
INNODB_SYS_TABLES	INNODB_TABLES
INNODB_SYS_TABLESPACES	INNODB_TABLESPACES
INNODB_SYS_TABLESTATS	INNODB_TABLESTATS
INNODB_SYS_VIRTUAL	INNODB_VIRTUAL

After upgrading to MySQL 8.0.3 or higher, update any scripts that reference previous InnoDB INFORMATION SCHEMA view names.

The zlib library version bundled with MySQL was raised from version 1.2.3 to version 1.2.11.

The zlib compressBound() function in zlib 1.2.11 returns a slightly higher estimate of the buffer size required to compress a given length of bytes than it did in zlib version 1.2.3. The compressBound() function is called by InnoDB functions that determine the maximum row size permitted when creating compressed InnoDB tables or inserting and updating rows in compressed InnoDB tables. As a result, CREATE TABLE ... ROW\_FORMAT=COMPRESSED, INSERT, and UPDATE operations with row sizes very close to the maximum row size that were successful in earlier releases could now fail. To avoid this issue, test CREATE TABLE statements for compressed InnoDB tables with large rows on a MySQL 8.0 test instance prior to upgrading.

• With the introduction of the --innodb-directories feature, the location of file-per-table and general tablespace files created with an absolute path or in a location outside of the data directory should be added to the innodb\_directories argument value. Otherwise, InnoDB is not able to locate these files during recovery. To view tablespace file locations, query the Information Schema FILES table:

SELECT TABLESPACE\_NAME, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES \G

 Undo logs can no longer reside in the system tablespace. In MySQL 8.0, undo logs reside in two undo tablespaces by default. For more information, see Undo Tablespaces.

When upgrading from MySQL 5.7 to MySQL 8.0, any undo tablespaces that exist in the MySQL 5.7 instance are removed and replaced by two new default undo tablespaces. Default undo

tablespaces are created in the location defined by the <code>innodb\_undo\_directory</code> variable. If the <code>innodb\_undo\_directory</code> variable is undefined, undo tablespaces are created in the data directory. Upgrade from MySQL 5.7 to MySQL 8.0 requires a slow shutdown which ensures that undo tablespaces in the MySQL 5.7 instance are empty, permitting them to be removed safely.

When upgrading to MySQL 8.0.14 or later from an earlier MySQL 8.0 release, undo tablespaces that exist in the pre-upgrade instance as a result of an <code>innodb\_undo\_tablespaces</code> setting greater than 2 are treated as user-defined undo tablespaces, which can be deactivated and dropped using ALTER UNDO TABLESPACE and DROP UNDO TABLESPACE syntax, respectively, after upgrading. Upgrade within the MySQL 8.0 release series may not always require a slow shutdown which means that existing undo tablespaces could contain undo logs. Therefore, existing undo tablespaces are not removed by the upgrade process.

• Incompatible change: As of MySQL 8.0.17, the CREATE TABLESPACE ... ADD DATAFILE clause does not permit circular directory references. For example, the circular directory reference (/../) in the following statement is not permitted:

```
CREATE TABLESPACE tsl ADD DATAFILE tsl.ibd 'any_directory/../tsl.ibd';
```

An exception to the restriction exists on Linux, where a circular directory reference is permitted if the preceding directory is a symbolic link. For example, the data file path in the example above is permitted if any\_directory is a symbolic link. (It is still permitted for data file paths to begin with '../'.)

To avoid upgrade issues, remove any circular directory references from tablespace data file paths before upgrading to MySQL 8.0.17 or higher. To inspect tablespace paths, query the Information Schema INNOB DATAFILES table.

Due to a regression introduced in MySQL 8.0.14, in-place upgrade on a case-sensitive file system from MySQL 5.7 or a MySQL 8.0 release prior to MySQL 8.0.14 to MySQL 8.0.16 failed for instances with partitioned tables and lower\_case\_table\_names=1. The failure was caused by a case mismatch issue related to partitioned table file names. The fix that introduced the regression was reverted, which permits upgrades to MySQL 8.0.17 from MySQL 5.7 or MySQL 8.0 releases prior to MySQL 8.0.14 to function as normal. However, the regression is still present in the MySQL 8.0.14, 8.0.15, and 8.0.16 releases.

In-place upgrade on a case-sensitive file system from MySQL 8.0.14, 8.0.15, or 8.0.16 to MySQL 8.0.17 fails with the following error when starting the server after upgrading binaries or packages to MySQL 8.0.17 if partitioned tables are present and lower case table names=1:

```
Upgrading from server version <a href="version_number">version_number</a> with partitioned tables and lower_case_table_names == 1 on a case sensitive file system may cause issues, and is therefore prohibited. To upgrade anyway, restart the new server version with the command line option 'upgrade=FORCE'. When upgrade is completed, please execute 'RENAME TABLE <a href="part_table_name">part_table_name</a> TO <a href="new_table_name">new_table_name</a>; RENAME TABLE <a href="new_table_name">new_table_name</a>; for each of the partitioned tables. Please see the documentation for further information.
```

If you encounter this error when upgrading to MySQL 8.0.17, perform the following workaround:

- 1. Restart the server with --upgrade=force to force the upgrade operation to proceed.
- Identify partitioned table file names with lowercase partition name delimiters (#p# or #sp#):

mysql> SELECT FILE\_NAME FROM INFORMATION\_SCHEMA.FILES WHERE FILE\_NAME LIKE '%#p#%' OR FILE\_NAME LIKE '%#s

3. For each file identified, rename the associated table using a temporary name, then rename the table back to its original name.

```
mysql> RENAME TABLE table_name TO temporary_table_name;
mysql> RENAME TABLE temporary_table_name TO table_name;
```

4. Verify that there are no partitioned table file names lowercase partition name delimiters (an empty result set should be returned).

```
mysql> select file_name from information_schema.files where file_name like '%#p#%' OR file_name like Empty set (0.00 sec)
```

5. Run ANALYZE TABLE on each renamed table to update the optimizer statistics in the mysql.innodb\_index\_stats and mysql.innodb\_table\_stats tables.

Because of the regression still present in the MySQL 8.0.14, 8.0.15, and 8.0.16 releases, importing partitioned tables from MySQL 8.0.14, 8.0.15, or 8.0.16 to MySQL 8.0.17 is not supported on case-sensitive file systems where <code>lower\_case\_table\_names=1</code>. Attempting to do so results in a "Tablespace is missing for table" error.

MySQL uses delimiter strings when constructing tablespace names and file names for table partitions. A
 "#p#" delimiter string precedes partition names, and an "#sp#" delimiter string precedes subpartition
 names, as shown:

```
schema_name.table_name#p#partition_name#sp#subpartition_name
table_name#p#partition_name#sp#subpartition_name.ibd
```

Historically, delimiter strings have been uppercase (#P# and #SP#) on case-sensitive file systems such as Linux, and lowercase (#p# and #sp#) on case-insensitive file systems such as Windows. As of MySQL 8.0.19, delimiter strings are lowercase on all file systems. This change prevents issues when migrating data directories between case-sensitive and case-insensitive file systems. Uppercase delimiter strings are no longer used.

Additionally, partition tablespace names and file names generated based on user-specified partition or subpartition names, which can be specified in uppercase or lowercase, are now generated (and stored internally) in lowercase regardless of the lower\_case\_table\_names setting to ensure case-insensitivity. For example, if a table partition is created with the name PART\_1, the tablespace name and file name are generated in lowercase:

```
schema_name.table_name#p#part_1
table_name#p#part_1.ibd
```

During upgrade, MySQL checks and modifies if necessary:

- Partition file names on disk and in the data dictionary to ensure lowercase delimiters and partition names.
- Partition metadata in the data dictionary for related issues introduced by previous bug fixes.
- InnoDB statistics data for related issues introduced by previous bug fixes.

During tablespace import operations, partition tablespace file names on disk are checked and modified if necessary to ensure lowercase delimiters and partition names.

 As of MySQL 8.0.21, a warning is written to the error log at startup or when upgrading from MySQL 5.7 if tablespace data files are found to reside in unknown directories. Known directories are those defined by the datadir, innodb\_data\_home\_dir, and innodb\_directories variables. To make a directory known, add it to the innodb\_directories setting. Making directories known ensures that data files can be found during recovery. For more information, see Tablespace Discovery During Crash Recovery.

• Important change: From MySQL 8.0.30, the innodb\_redo\_log\_capacity variable controls the amount of disk space occupied by redo log files. With this change, the default number of redo log files and their location has also changed. From MySQL 8.0.30, InnoDB maintains 32 redo log files in the #innodb\_redo directory in the data directory. Previously, InnoDB created two redo log files in the data directory by default, and the number and size of redo log files were controlled by the innodb\_log\_files\_in\_group and innodb\_log\_file\_size variables. These two variables are now deprecated.

When the innodb\_redo\_log\_capacity setting is defined, innodb\_log\_files\_in\_group and innodb\_log\_file\_size settings are ignored; otherwise, those settings are used to compute the innodb\_redo\_log\_capacity setting (innodb\_log\_files\_in\_group \* innodb\_log\_file\_size = innodb\_redo\_log\_capacity). If none of those variables are set, redo log capacity is set to the innodb\_redo\_log\_capacity default value, which is 104857600 bytes (100MB).

As is generally required for any upgrade, this change requires a clean shutdown before upgrading.

For more information about this feature, see Redo Log.

 Before MySQL 5.7.35, there was no size limitation for indexes in tables with redundant or compact row format. As of MySQL 5.7.35, the limit is 767 bytes. An upgrade from a MySQL version before 5.7.35 to MySQL 8.0 can produce inaccessible tables. If a table with redundant or compact row format has an index larger than 767 bytes, drop the index and re-create it before an upgrade to MySQL 8.0. The error message is:

mysql> ERROR 1709 (HY000): Index column size too large. The maximum column size is 767 bytes.

# **SQL Changes**

• Incompatible change: As of MySQL 8.0.13, the deprecated ASC or DESC qualifiers for GROUP BY clauses have been removed. Queries that previously relied on GROUP BY sorting may produce results that differ from previous MySQL versions. To produce a given sort order, provide an ORDER BY clause.

Queries and stored program definitions from MySQL 8.0.12 or lower that use ASC or DESC qualifiers for GROUP BY clauses should be amended. Otherwise, upgrading to MySQL 8.0.13 or higher may fail, as may replicating to MySQL 8.0.13 or higher replica servers.

- Some keywords may be reserved in MySQL 8.0 that were not reserved in MySQL 5.7. See Keywords and Reserved Words. This can cause words previously used as identifiers to become illegal. To fix affected statements, use identifier quoting. See Schema Object Names.
- After upgrading, it is recommended that you test optimizer hints specified in application code to ensure
  that the hints are still required to achieve the desired optimization strategy. Optimizer enhancements can
  sometimes render certain optimizer hints unnecessary. In some cases, an unnecessary optimizer hint
  may even be counterproductive.
- Incompatible change: In MySQL 5.7, specifying a FOREIGN KEY definition for an InnoDB table without a CONSTRAINT symbol clause, or specifying the CONSTRAINT keyword without a symbol, causes InnoDB to use a generated constraint name. That behavior changed in MySQL 8.0, with InnoDB using the FOREIGN KEY index\_name value instead of a generated name. Because constraint names must be unique per schema (database), the change caused errors due to foreign key index names that were not unique per schema. To avoid such errors, the new constraint naming behavior has been reverted in MySQL 8.0.16, and InnoDB once again uses a generated constraint name.

For consistency with InnoDB, NDB releases based on MySQL 8.0.16 or higher use a generated constraint name if the CONSTRAINT symbol clause is not specified, or the CONSTRAINT keyword is specified without a symbol. NDB releases based on MySQL 5.7 and earlier MySQL 8.0 releases used the FOREIGN KEY index name value.

The changes described above may introduce incompatibilities for applications that depend on the previous foreign key constraint naming behavior.

- The handling of system variable values by MySQL flow control functions such as IFNULL() and CASE() changed in MySQL 8.0.22; system variable values are now handled as column values of the same character and collation, rather than as constants. Some queries using these functions with system variables that were previously successful may subsequently be rejected with Illegal mix of collations. In such cases, cast the system variable to the correct character set and collation.
- Incompatible change: MySQL 8.0.28 fixes an issue in previous MySQL 8.0 releases whereby the CONVERT() function sometimes allowed invalid casts of BINARY values to nonbinary character sets.

   Applications which may have relied on this behavior should be checked and if necessary modified prior to upgrade.

In particular, where CONVERT() was used as part of an expression for an indexed generated column, the change in the function's behavior may result in index corruption following an upgrade to MySQL 8.0.28. You can prevent this from happening by following these steps:

- 1. Prior to performing the upgrade, correct any invalid input data.
- 2. Drop and then re-create the index.

You can also force a table rebuild using ALTER TABLE table FORCE, instead.

3. Upgrade the MySQL software.

If you cannot validate the input data beforehand, you should not re-create the index or rebuild the table until after you perform the upgrade to MySQL 8.0.28.

# **Changed Server Defaults**

MySQL 8.0 comes with improved defaults, aiming at the best out of the box experience possible. These changes are driven by the fact that technology is advancing (machines have more CPUS, use SSDs and so on), more data is being stored, MySQL is evolving (InnoDB, Group Replication, AdminAPI), and so on. The following table summarizes the defaults which have been changed to provide the best MySQL experience for the majority of users.

Option/Parameter	Old Default	New Default
Server changes		
character_set_server	latin1	utf8mb4
collation_server	latin1_swedish_ci	utf8mb4_0900_ai_ci
explicit_defaults_for_time	<b>.0Fal</b> inp	ON
optimizer_trace_max_mem_si	₫ <b>6</b> KB	1MB
validate_password_check_us	<b>OF</b> Fname	ON
back_log	-1 (autosize) changed from: back_log = 50 + (max_connections / 5)	-1 (autosize) changed to : back_log = max_connections

Option/Parameter	Old Default	New Default
max_allowed_packet	4194304 (4MB)	67108864 (64MB)
max_error_count	64	1024
event_scheduler	OFF	ON
table_open_cache	2000	4000
log_error_verbosity	3 (Notes)	2 (Warning)
local_infile	ON (5.7)	OFF
InnoDB changes		
innodb_undo_tablespaces	0	2
innodb_undo_log_truncate	OFF	ON
innodb_flush_method	NULL	fsync (Unix), unbuffered (Windows)
innodb_autoinc_lock_mode	1 (consecutive)	2 (interleaved)
innodb_flush_neighbors	1 (enable)	0 (disable)
innodb_max_dirty_pages_pct	01(%)	10 (%)
innodb_max_dirty_pages_pct	75 (%)	90 (%)
Performance Schema changes		
<pre>performance-schema- instrument='wait/lock/ metadata/sql/%=ON'</pre>	OFF	ON
<pre>performance-schema- instrument='memory/ %=COUNTED'</pre>	OFF	COUNTED
performance-schema- consumer-events- transactions-current=ON	OFF	ON
performance-schema- consumer-events- transactions-history=ON	OFF	ON
<pre>performance-schema- instrument='transaction %=ON'</pre>	OFF	ON
Replication changes		
log_bin	OFF	ON
server_id	0	1
log-slave-updates	OFF	ON
expire_logs_days	0	30
master-info-repository	FILE	TABLE
relay-log-info-repository	FILE	TABLE
transaction-write-set- extraction	OFF	XXHASH64
slave_rows_search_algorith	HNDEX_SCAN, TABLE_SCAN	INDEX_SCAN, HASH_SCAN

Option/Parameter	Old Default	New Default
slave_pending_jobs_size_ma	⊿16M	128M
gtid_executed_compression_	<b>1000</b> od	0
Group Replication changes		
group_replication_autorejo	<b>O</b> n_tries	3
group_replication_exit_sta	tABQRT <u>i</u> SERVER	READ_ONLY
group_replication_member_e	<b>£</b> pel_timeout	5

For more information about options or variables which have been added, see Option and Variable Changes for MySQL 8.0, in the MySQL Server Version Reference.

The following sections explain the changes to defaults and any impact they might have on your deployment.

#### Server Defaults

- The default value of the <a href="mailto:character\_set\_server">character\_set\_server</a> character-set-server changed from <a href="mailto:latin1">latin1</a> to <a href="mailto:utf8mb4">utf8mb4</a>. This is the server's default character set. At this time, UTF8MB4 is the dominant character encoding for the web, and this change makes life easier for the vast majority of MySQL users. The upgrade from 5.7 to 8.0 does not change the character set for any existing database objects, but, unless you set <a href="mailto:character\_set\_server">character\_set\_server</a> explicitly (either back to the previous value, or to a new one), a new schema uses <a href="mailto:utf8mb4">utf8mb4</a> by default. We recommend that you move to <a href="mailto:utf8mb4">utf8mb4</a> whenever possible.
- The default value of the collation\_server system variable and command line argument -collation-server changed from latin1\_swedish\_ci to utf8mb4\_0900\_ai\_ci. This is the
  server's default collation, the ordering of characters in a character set. There is a link between collations
  and character sets as each character set comes with a list of possible collations. The upgrade from 5.7
  to 8.0 does not change any collation for any existing database objects, but takes effect for new objects.
- The default value of the explicit\_defaults\_for\_timestamp system variable changed from OFF (MySQL legacy behavior) to ON (SQL standard behavior). This option was originally introduced in 5.6 and was OFF in 5.6 and 5.7.
- The default value of the <code>optimizer\_trace\_max\_mem\_size</code> system variable changed from <code>16KB</code> to <code>1MB</code>. The old default caused the optimizer trace to be truncated for any non-trivial query. This change ensures useful optimizer traces for most queries.
- The default value of the validate\_password\_check\_user\_name system variable changed from OFF to ON. This means that when the validate\_password plugin is enabled, by default it now rejects passwords that match the current session user name.
- The autosize algorithm for the back\_log system variable has changed. The value for autosize
   (-1) is now set to the value of max\_connections, which is bigger than the calculated by 50
   + (max\_connections / 5). The back\_log queues up incoming IP connect requests in situations where the server is not able to keep up with incoming requests. In the worst case, with max\_connections number of clients trying to reconnect at the same time, for example after a network failure, they can all be buffered and reject-retry loops are avoided.
- The default value of the max\_allowed\_packet system variable changed from 4194304 (4M) to 67108864 (64M). The main advantage with this larger default is less chance of receiving errors about an insert or query being larger than max\_allowed\_packet. It should be as big as the largest The BLOB and TEXT Types you want to use. To revert to the previous behavior, set max\_allowed\_packet=4194304.

- The default value of the max\_error\_count system variable changed from 64 to 1024. This ensures that MySQL handles a larger number of warnings, such as an UPDATE statement that touches 1000s of rows and many of them give conversion warnings. It is common for many tools to batch updates, to help reduce replication lag. External tools such as pt-online-schema-change defaults to 1000, and ghost defaults to 100. MySQL 8.0 covers full error history for these two use cases. There are no static allocations, so this change only affects memory consumption for statements that generate lots of warnings.
- The default value of the event\_scheduler system variable changed from OFF to ON. In other words, the event scheduler is enabled by default. This is an enabler for new features in SYS, for example "kill idle transactions".
- The default value of the table\_open\_cache system variable changed from 2000 to 4000. This is a minor change which increases session concurrency on table access.
- The default value of the log\_error\_verbosity system variable changed from 3 (Notes) to 2 (Warning). The purpose is to make the MySQL 8.0 error log less verbose by default.

#### **InnoDB Defaults**

- Incompatible change The default value of the innodb\_undo\_tablespaces system variable changed from 0 to 2. The configures the number of undo tablespaces used by InnoDB. In MySQL 8.0 the minimum value for innodb\_undo\_tablespaces is 2 and rollback segments cannot be created in the system tablespace anymore. Thus, this is a case where you cannot revert back to 5.7 behavior. The purpose of this change is to be able to auto-truncate Undo logs (see next item), reclaiming disk space used by (occasional) long transactions such as a mysqldump.
- The default value of the <a href="innodb\_undo\_log\_truncate">innodb\_undo\_log\_truncate</a> system variable changed from OFF to ON. When enabled, undo tablespaces that exceed the threshold value defined by <a href="innodb\_max\_undo\_log\_size">innodb\_max\_undo\_log\_size</a> are marked for truncation. Only undo tablespaces can be truncated. Truncating undo logs that reside in the system tablespace is not supported. An upgrade from 5.7 to 8.0 automatically converts your system to use undo tablespaces, using the system tablespace is not an option in 8.0.
- The default value of the innodb\_flush\_method system variable changed from NULL to fsync on Unix-like systems and from NULL to unbuffered on Windows systems. This is more of a terminology and option cleanup without any tangible impact. For Unix this is just a documentation change as the default was fsync also in 5.7 (the default NULL meant fsync). Similarly on Windows, innodb\_flush\_method default NULL meant async\_unbuffered in 5.7, and is replaced by default unbuffered in 8.0, which in combination with the existing default innodb\_use\_native\_aio=ON has the same effect.
- Incompatible change The default value of the innodb\_autoinc\_lock\_mode system variable changed from 1 (consecutive) to 2 (interleaved). The change to interleaved lock mode as the default setting reflects the change from statement-based to row-based replication as the default replication type, which occurred in MySQL 5.7. Statement-based replication requires the consecutive autoincrement lock mode to ensure that auto-increment values are assigned in a predictable and repeatable order for a given sequence of SQL statements, whereas row-based replication is not sensitive to the execution order of SQL statements. Thus, this change is known to be incompatible with statement based replication, and may break some applications or user-generated test suites that depend on sequential auto increment. The previous default can be restored by setting innodb\_autoinc\_lock\_mode=1;
- The default value of the <code>innodb\_flush\_neighbors</code> system variable changes from 1 (enable) to 0 (disable). This is done because fast IO (SSDs) is now the default for deployment. We expect that for the majority of users, this results in a small performance gain. Users who are using slower hard drives may see a performance loss, and are encouraged to revert to the previous defaults by setting <code>innodb\_flush\_neighbors=1</code>.

- The default value of the innodb\_max\_dirty\_pages\_pct\_lwm system variable changed from 0 (%) to 10 (%). With innodb\_max\_dirty\_pages\_pct\_lwm=10, InnoDB increases its flushing activity when >10% of the buffer pool contains modified ('dirty') pages. The purpose of this change is to trade off peak throughput slightly, in exchange for more consistent performance.
- The default value of the innodb\_max\_dirty\_pages\_pct system variable changed from 75 (%) to 90 (%). This change combines with the change to innodb\_max\_dirty\_pages\_pct\_lwm and together they ensure a smooth InnoDB flushing behavior, avoiding flushing bursts. To revert to the previous behavior, set innodb\_max\_dirty\_pages\_pct=75 and innodb\_max\_dirty\_pages\_pct\_lwm=0.

#### **Performance Schema Defaults**

- Performance Schema Meta Data Locking (MDL) instrumentation is turned on by default. The compiled default for performance-schema-instrument='wait/lock/metadata/sql/%=ON' changed from OFF to ON. This is an enabler for adding MDL oriented views in SYS.
- Performance Schema Memory instrumentation is turned on by default. The compiled default for performance-schema-instrument='memory/%=COUNTED' changed from OFF to COUNTED. This is important because the accounting is incorrect if instrumentation is enabled after server start, and you could get a negative balance from missing an allocation, but catching a free.
- Performance Schema Transaction instrumentation is turned on by default. The compiled default for
   performance-schema-consumer-events-transactions-current=ON, performance schema-consumer-events-transactions-history=ON, and performance-schema instrument='transaction%=ON' changed from OFF to ON.

#### **Replication Defaults**

- The default value of the log\_bin system variable changed from OFF to ON. In other words, binary logging is enabled by default. Nearly all production installations have the binary log enabled as it is used for replication and point-in-time recovery. Thus, by enabling binary log by default we eliminate one configuration step, enabling it later requires a mysqld restart. Enabling it by default also provides better test coverage and it becomes easier to spot performance regressions. Remember to also set server\_id (see following change). The 8.0 default behavior is as if you issued ./mysqld --log-bin --server-id=1. If you are on 8.0 and want 5.7 behavior you can issue ./mysqld --skip-log-bin --server-id=0.
- The default value of the server\_id system variable changed from 0 to 1 (combines with the change to log\_bin=ON). The server can be started with this default ID, but in practice you must set the server-id according to the replication infrastructure being deployed, to avoid having duplicate server ids.
- The default value of the log-slave-updates system variable changed from OFF to ON. This causes a replica to log replicated events into its own binary log. This option is required for Group Replication, and also ensures correct behavior in various replication chain setups, which have become the norm today.
- The default value of the expire\_logs\_days system variable changed from 0 to 30. The new default 30 causes mysqld to periodically purge unused binary logs that are older than 30 days. This change helps prevent excessive amounts of disk space being wasted on binary logs that are no longer needed for replication or recovery purposes. The old value of 0 disables any automatic binary log purges.
- The default value of the master\_info\_repository and relay\_log\_info\_repository system variables change from FILE to TABLE. Thus in 8.0, replication metadata is stored in InnoDB by default. This increases reliability to try and achieve crash safe replication by default.
- The default value of the transaction-write-set-extraction system variable changed from OFF to XXHASH64. This change enables transaction write sets by default. By using Transaction Write Sets, the source has to do slightly more work to generate the write sets, but the result is helpful in conflict

detection. This is a requirement for Group Replication and the new default makes it easy to enable binary log writeset parallelization on the source to speed up replication.

- The default value of the slave\_rows\_search\_algorithms system variable changed from INDEX\_SCAN, TABLE\_SCAN to INDEX\_SCAN, HASH\_SCAN. This change speeds up row-based replication by reducing the number of table scans the replica applier has to do to apply the changes to a table without a primary key.
- The default value of the slave\_pending\_jobs\_size\_max system variable changed from 16M to 128M. This change increases the amount of memory available to multithreaded replicas.
- The default value of the <code>gtid\_executed\_compression\_period</code> system variable changed from 1000 to 0. This change ensures that compression of the <code>mysql.gtid\_executed</code> table only occurs implicitly as required.

#### **Group Replication Defaults**

- The default value of group\_replication\_autorejoin\_tries changed from 0 to 3, which means
  that automatic rejoin is enabled by default. This system variable specifies the number of tries that a
  member makes to automatically rejoin the group if it is expelled, or if it is unable to contact a majority of
  the group before the group\_replication\_unreachable\_majority\_timeout setting is reached.
- The default value of group\_replication\_exit\_state\_action changed from ABORT\_SERVER to READ\_ONLY. This means that when a member exits the group, for example after a network failure, the instance becomes read-only, rather than being shut down.
- The default value of group\_replication\_member\_expel\_timeout changed from 0 to 5, meaning that a member suspected of having lost contact with the group is liable for expulsion 5 seconds after the 5-second detection period.

Most of these defaults are reasonably good for both development and production environments. An exception to this is the <code>--innodb-dedicated-server</code> option, whose default value remains <code>OFF</code>, although we recommend <code>ON</code> for production environments. The reason for defaulting to <code>OFF</code> is that it causes shared environments such as developer laptops to become unusable, because it takes <code>all</code> the memory it can find.

For production environments we recommend using <code>--innodb-dedicated-server</code>, which determines values for the following InnoDB variables (if not specified explicitly), based on available memory: <code>innodb\_buffer\_pool\_size</code>, <code>innodb\_log\_file\_size</code>, and <code>innodb\_flush\_method</code>. See Enabling Automatic InnoDB Configuration for a Dedicated MySQL Server.

Although the new defaults are the best configuration choices for most use cases, there are special cases, as well as legacy reasons for using existing 5.7 configuration choices. For example, some people prefer to upgrade to 8.0 with as few changes to their applications or operational environment as possible. We recommend to evaluate all the new defaults and use as many as you can. Most new defaults can be tested in 5.7, so you can validate the new defaults in 5.7 production before upgrading to 8.0. For the few defaults where you need your old 5.7 value, set the corresponding configuration variable or startup option in your operational environment.

MySQL 8.0 has the Performance Schema variables\_info table, which shows for each system variable
the source from which it was most recently set, as well as its range of values. This provides SQL access to
all there is to know about a configuration variable and its values.

# **Valid Performance Regressions**

Performance regressions are expected between MySQL versions 5.7 and 8.0. MySQL 8.0 has more features, changes default values, is more robust, and adds security functionality and additional diagnostic

information. Listed here are valid reasons for regressions between these versions which includes potential mediation options. This is not an exhaustive list.

Changes related to default values changing between MySQL versions 5.7 and 8.0:

• Binary logs are disabled by default in 5.7, and enabled by default in 8.0.

*Mediation*: Disable binary logging by specifying the --skip-log-bin or --disable-log-bin option at startup.

• The default character set changed from latin1 to utf8mb4 in 8.0. While utf8mb4 performs significantly better in 8.0 than it did in 5.7, latin1 is faster than utf8mb4.

Mediation: Use latin1 in 8.0 if utf8mb4 is not needed.

Transactional Data Dictionary (atomic DDL) was introduced in 8.0.

 This increases robustness/reliability at the expense of DDL performance (CREATE / DROP intensive loads), but it should not impact the DML load (SELECT / INSERT / UPDATE / DELETE).

Mediation: None

The more modern TLS ciphers/algorithms used as of 5.7.28 has an effect when TLS (SSL) is enabled (the default):

 Before MySQL 5.7.28, MySQL uses the yaSSL library for the community edition and OpenSSL for the enterprise edition.

As of MySQL 5.7.28, MySQL only uses OpenSSL with its stronger TLS ciphers, which are more costly in terms of performance.

Upgrading to MySQL 8.0 from MySQL 5.7.28 or earlier can cause a TLS performance regression.

Mediation: None (if TLS is required for security reasons)

Performance Schema (PFS) instrumentation is much wider in 8.0 than in 5.7:

 PFS cannot be compiled out in MySQL 8.0 but it can be turned off. Some performance schema instrumentation will still exist even when turned off, but overhead will be smaller.

*Mediation*: Set performance\_schema = OFF in 8.0, or turn off performance schema instrumentation at finer granularity if some but not all PFS functionality is needed.

Truncating undo tablespaces is enabled by default in 8.0 which can significantly impact performance:

Historically InnoDB stored undo logs in the system tablespace but there was no way to reclaim space
used by undo log. The system tablespace would only grow and not shrink, and this inspired feature
requests to remedy this.

MySQL 8.0 moved the undo log to separate tablespaces which allows both manual and automatic undo log truncation.

However, auto-truncation has a permanent performance overhead and it can potentially cause stalls.

*Mediation*: Set innodb\_undo\_log\_truncate = OFF in 8.0, and manually truncate undo logs as needed. For related information, see Truncating Undo Tablespaces.

The character classes [[:alpha:]] or [[:digit:]] do not perform as well with regular expression functions such as REGEXP() and RLIKE() in MySQL 8.0 as they did in MySQL 5.7. This is due to the

replacement in MySQL 8.0 of the Spencer regular expression library with the ICU library, which uses UTF-16 internally.

*Mediation*: In place of [[:alpha:]], use [a-zA-Z]; in place of [[:digit:]], use [0-9].

# 10.6 Preparing Your Installation for Upgrade

Before upgrading to the latest MySQL 8.0 release, ensure the upgrade readiness of your current MySQL 5.7 or MySQL 8.0 server instance by performing the preliminary checks described below. The upgrade process may fail otherwise.

#### Tip

Consider using the MySQL Shell upgrade checker utility that enables you to verify whether MySQL server instances are ready for upgrade. You can select a target MySQL Server release to which you plan to upgrade, ranging from the MySQL Server 8.0.11 up to the MySQL Server release number that matches the current MySQL Shell release number. The upgrade checker utility carries out the automated checks that are relevant for the specified target release, and advises you of further relevant checks that you should make manually. The upgrade checker works for all Bugfix, Innovation, and LTS releases of MySQL. Installation instructions for MySQL Shell can be found here.

#### Preliminary checks:

- 1. The following issues must not be present:
  - There must be no tables that use obsolete data types or functions.

In-place upgrade to MySQL 8.0 is not supported if tables contain old temporal columns in pre-5.6.4 format (TIME, DATETIME, and TIMESTAMP columns without support for fractional seconds precision). If your tables still use the old temporal column format, upgrade them using REPAIR TABLE before attempting an in-place upgrade to MySQL 8.0. For more information, see Server Changes, in MySQL 5.7 Reference Manual.

- There must be no orphan .frm files.
- Triggers must not have a missing or empty definer or an invalid creation context (indicated by the character\_set\_client, collation\_connection, Database Collation attributes displayed by SHOW TRIGGERS or the INFORMATION\_SCHEMA TRIGGERS table). Any such triggers must be dumped and restored to fix the issue.

To check for these issues, execute this command:

```
mysqlcheck -u root -p --all-databases --check-upgrade
```

If mysqlcheck reports any errors, correct the issues.

2. There must be no partitioned tables that use a storage engine that does not have native partitioning support. To identify such tables, execute this query:

```
SELECT TABLE_SCHEMA, TABLE_NAME

FROM INFORMATION_SCHEMA.TABLES

WHERE ENGINE NOT IN ('innodb', 'ndbcluster')

AND CREATE_OPTIONS LIKE '%partitioned%';
```

Any table reported by the query must be altered to use InnoDB or be made nonpartitioned. To change a table storage engine to InnoDB, execute this statement:

```
ALTER TABLE table_name ENGINE = INNODB;
```

For information about converting MyISAM tables to InnoDB, see Converting Tables from MyISAM to InnoDB.

To make a partitioned table nonpartitioned, execute this statement:

```
ALTER TABLE table_name REMOVE PARTITIONING;
```

- Some keywords may be reserved in MySQL 8.0 that were not reserved previously. See Keywords and Reserved Words. This can cause words previously used as identifiers to become illegal. To fix affected statements, use identifier quoting. See Schema Object Names.
- 4. There must be no tables in the MySQL 5.7 mysql system database that have the same name as a table used by the MySQL 8.0 data dictionary. To identify tables with those names, execute this query:

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE LOWER(TABLE_SCHEMA) = 'mysql'
and LOWER(TABLE NAME) IN
'catalogs',
'character_sets',
'check_constraints',
'collations',
'column_statistics',
'column_type_elements',
'columns',
'dd_properties',
'events',
'foreign_key_column_usage',
'foreign_keys',
'index_column_usage',
'index_partitions',
'index_stats',
'indexes',
'parameter_type_elements',
'parameters',
'resource groups',
'routines',
'schemata',
'st_spatial_reference_systems',
'table_partition_values',
'table_partitions',
'table_stats',
'tables',
'tablespace_files',
'tablespaces',
'triggers',
'view_routine_usage',
'view_table_usage'
);
```

Any tables reported by the query must be dropped or renamed (use RENAME TABLE). This may also entail changes to applications that use the affected tables.

5. There must be no tables that have foreign key constraint names longer than 64 characters. Use this query to identify tables with constraint names that are too long:

```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME IN
(SELECT LEFT(SUBSTR(ID, INSTR(ID, '/')+1),
```

```
INSTR(SUBSTR(ID,INSTR(ID,'/')+1),'_ibfk_')-1)
FROM INFORMATION_SCHEMA.INNODB_SYS_FOREIGN
WHERE CHAR_LENGTH(SUBSTR(ID,INSTR(ID,'/')+1))>64);
```

For a table with a constraint name that exceeds 64 characters, drop the constraint and add it back with constraint name that does not exceed 64 characters (use ALTER TABLE).

- There must be no obsolete SQL modes defined by sql\_mode system variable. Attempting to use an
  obsolete SQL mode prevents MySQL 8.0 from starting. Applications that use obsolete SQL modes
  should be revised to avoid them. For information about SQL modes removed in MySQL 8.0, see Server
  Changes.
- 7. Only upgrade a MySQL server instance that was properly shut down. If the instance unexpectedly shutdown, then restart the instance and shut it down with innodb\_fast\_shutdown=0 before upgrade.
- 8. There must be no views with explicitly defined columns names that exceed 64 characters (views with column names up to 255 characters were permitted in MySQL 5.7). To avoid upgrade errors, such views should be altered before upgrading. Currently, the only method of identify views with column names that exceed 64 characters is to inspect the view definition using SHOW CREATE VIEW. You can also inspect view definitions by querying the Information Schema VIEWS table.
- 9. There must be no tables or stored procedures with individual ENUM or SET column elements that exceed 255 characters or 1020 bytes in length. Prior to MySQL 8.0, the maximum combined length of ENUM or SET column elements was 64K. In MySQL 8.0, the maximum character length of an individual ENUM or SET column element is 255 characters, and the maximum byte length is 1020 bytes. (The 1020 byte limit supports multibyte character sets). Before upgrading to MySQL 8.0, modify any ENUM or SET column elements that exceed the new limits. Failing to do so causes the upgrade to fail with an error.
- 10. Before upgrading to MySQL 8.0.13 or higher, there must be no table partitions that reside in shared InnoDB tablespaces, which include the system tablespace and general tablespaces. Identify table partitions in shared tablespaces by querying INFORMATION\_SCHEMA:

If upgrading from MySQL 5.7, run this query:

```
SELECT DISTINCT NAME, SPACE, SPACE_TYPE FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES
WHERE NAME LIKE '%#P#%' AND SPACE_TYPE NOT LIKE 'Single';
```

If upgrading from an earlier MySQL 8.0 release, run this query:

```
SELECT DISTINCT NAME, SPACE, SPACE_TYPE FROM INFORMATION_SCHEMA.INNODB_TABLES
WHERE NAME LIKE '%#P#%' AND SPACE_TYPE NOT LIKE 'Single';
```

Move table partitions from shared tablespaces to file-per-table tablespaces using ALTER TABLE ... REORGANIZE PARTITION:

```
ALTER TABLE table_name REORGANIZE PARTITION partition_name
INTO (partition_definition TABLESPACE=innodb_file_per_table);
```

11. There must be no queries and stored program definitions from MySQL 8.0.12 or lower that use ASC or DESC qualifiers for GROUP BY clauses. Otherwise, upgrading to MySQL 8.0.13 or higher may fail, as may replicating to MySQL 8.0.13 or higher replica servers. For additional details, see SQL Changes.

12. Your MySQL 5.7 installation must not use features that are not supported by MySQL 8.0. Any changes here are necessarily installation specific, but the following example illustrates the kind of thing to look for:

Some server startup options and system variables have been removed in MySQL 8.0. See Features Removed in MySQL 8.0, and Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.0. If you use any of these, an upgrade requires configuration changes.

Example: Because the data dictionary provides information about database objects, the server no longer checks directory names in the data directory to find databases. Consequently, the <code>--ignore-db-dir</code> option is extraneous and has been removed. To handle this, remove any instances of <code>--ignore-db-dir</code> from your startup configuration. In addition, remove or move the named data directory subdirectories before upgrading to MySQL 8.0. (Alternatively, let the 8.0 server add those directories to the data dictionary as databases, then remove each of those databases using <code>DROPDATABASE</code>.)

13. If you intend to change the lower\_case\_table\_names setting to 1 at upgrade time, ensure that schema and table names are lowercase before upgrading. Otherwise, a failure could occur due to a schema or table name lettercase mismatch. You can use the following queries to check for schema and table names containing uppercase characters:

```
mysql> select TABLE_NAME, if(sha(TABLE_NAME) !=sha(lower(TABLE_NAME)),'Yes','No') as UpperCase from inf
```

As of MySQL 8.0.19, if <code>lower\_case\_table\_names=1</code>, table and schema names are checked by the upgrade process to ensure that all characters are lowercase. If table or schema names are found to contain uppercase characters, the upgrade process fails with an error.

#### Note

Changing the <code>lower\_case\_table\_names</code> setting at upgrade time is not recommended.

If upgrade to MySQL 8.0 fails due to any of the issues outlined above, the server reverts all changes to the data directory. In this case, remove all redo log files and restart the MySQL 5.7 server on the existing data directory to address the errors. The redo log files (ib\_logfile\*) reside in the MySQL data directory by default. After the errors are fixed, perform a slow shutdown (by setting innodb\_fast\_shutdown=0) before attempting the upgrade again.

# 10.7 Upgrading MySQL Binary or Package-based Installations on Unix/Linux

This section describes how to upgrade MySQL binary and package-based installations on Unix/Linux. In-place and logical upgrade methods are described.

- In-Place Upgrade
- Logical Upgrade
- MySQL Cluster Upgrade

# In-Place Upgrade

An in-place upgrade involves shutting down the old MySQL server, replacing the old MySQL binaries or packages with the new ones, restarting MySQL on the existing data directory, and upgrading any

remaining parts of the existing installation that require upgrading. For details about what may need upgrading, see Section 10.4, "What the MySQL Upgrade Process Upgrades".

#### Note

If you are upgrading an installation originally produced by installing multiple RPM packages, upgrade all the packages, not just some. For example, if you previously installed the server and client RPMs, do not upgrade just the server RPM.

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, <code>mysqld\_safe</code> is not installed. In such cases, use systemd for server startup and shutdown instead of the methods used in the following instructions. See Section 7.9, "Managing MySQL Server with systemd".

For upgrades to MySQL Cluster installations, see also MySQL Cluster Upgrade.

To perform an in-place upgrade:

- 1. Review the information in Section 10.1, "Before You Begin".
- 2. Ensure the upgrade readiness of your installation by completing the preliminary checks in Section 10.6, "Preparing Your Installation for Upgrade".
- 3. If you use XA transactions with InnoDB, run XA RECOVER before upgrading to check for uncommitted XA transactions. If results are returned, either commit or rollback the XA transactions by issuing an XA COMMIT or XA ROLLBACK statement.
- 4. If you are upgrading from MySQL 5.7.11 or earlier to MySQL 8.0, and there are encrypted InnoDB tablespaces, rotate the keyring master key by executing this statement:

```
ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

5. If you normally run your MySQL server configured with innodb\_fast\_shutdown set to 2 (cold shutdown), configure it to perform a fast or slow shutdown by executing either of these statements:

```
SET GLOBAL innodb_fast_shutdown = 1; -- fast shutdown
SET GLOBAL innodb_fast_shutdown = 0; -- slow shutdown
```

With a fast or slow shutdown, InnoDB leaves its undo logs and data files in a state that can be dealt with in case of file format differences between releases.

6. Shut down the old MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

- 7. Upgrade the MySQL binaries or packages. If upgrading a binary installation, unpack the new MySQL binary distribution package. See Obtain and Unpack the Distribution. For package-based installations, install the new packages.
- 8. Start the MySQL 8.0 server, using the existing data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir &
```

If there are encrypted InnoDB tablespaces, use the --early-plugin-load option to load the keyring plugin.

When you start the MySQL 8.0 server, it automatically detects whether data dictionary tables are present. If not, the server creates them in the data directory, populates them with metadata, and then proceeds with its normal startup sequence. During this process, the server upgrades metadata for

all database objects, including databases, tablespaces, system and user tables, views, and stored programs (stored procedures and functions, triggers, and Event Scheduler events). The server also removes files that previously were used for metadata storage. For example, after upgrading from MySQL 5.7 to MySQL 8.0, you may notice that tables no longer have .frm files.

If this step fails, the server reverts all changes to the data directory. In this case, you should remove all redo log files, start your MySQL 5.7 server on the same data directory, and fix the cause of any errors. Then perform another slow shutdown of the 5.7 server and start the MySQL 8.0 server to try again.

- 9. In the previous step, the server upgrades the data dictionary as necessary. Now it is necessary to perform any remaining upgrade operations:
  - As of MySQL 8.0.16, the server does so as part of the previous step, making any changes required in the mysql system database between MySQL 5.7 and MySQL 8.0, so that you can take advantage of new privileges or capabilities. It also brings the Performance Schema, INFORMATION\_SCHEMA, and sys databases up to date for MySQL 8.0, and examines all user databases for incompatibilities with the current version of MySQL.
  - Prior to MySQL 8.0.16, the server upgrades only the data dictionary in the previous step. After the MySQL 8.0 server starts successfully, execute mysql\_upgrade to perform the remaining upgrade tasks:

```
mysql_upgrade -u root -p
```

Then shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir &
```

The first time you start the MySQL 8.0 server (in an earlier step), you may notice messages in the error log regarding nonupgraded tables. If mysql\_upgrade has been run successfully, there should be no such messages the second time you start the server.

#### Note

The upgrade process does not upgrade the contents of the time zone tables. For upgrade instructions, see MySQL Server Time Zone Support.

If the upgrade process uses mysql\_upgrade (that is, prior to MySQL 8.0.16), the process does not upgrade the contents of the help tables, either. For upgrade instructions in that case, see Server-Side Help Support.

# **Logical Upgrade**

A logical upgrade involves exporting SQL from the old MySQL instance using a backup or export utility such as mysqldump or mysqlpump, installing the new MySQL server, and applying the SQL to your new MySQL instance. For details about what may need upgrading, see Section 10.4, "What the MySQL Upgrade Process Upgrades".

#### Note

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, <code>mysqld\_safe</code> is not installed. In such cases, use systemd for server startup and shutdown instead of the methods used in the following instructions. See Section 7.9, "Managing MySQL Server with systemd".

#### Warning

Applying SQL extracted from a previous MySQL release to a new MySQL release may result in errors due to incompatibilities introduced by new, changed, deprecated, or removed features and capabilities. Consequently, SQL extracted from a previous MySQL release may require modification to enable a logical upgrade.

To identify incompatibilities before upgrading to the latest MySQL 8.0 release, perform the steps described in Section 10.6, "Preparing Your Installation for Upgrade".

To perform a logical upgrade:

- 1. Review the information in Section 10.1, "Before You Begin".
- 2. Export your existing data from the previous MySQL installation:

```
mysqldump -u root -p
--add-drop-table --routines --events
--all-databases --force > data-for-upgrade.sql
```

#### Note

Use the --routines and --events options with mysqldump (as shown above) if your databases include stored programs. The --all-databases option includes all databases in the dump, including the mysql database that holds the system tables.

#### **Important**

If you have tables that contain generated columns, use the mysqldump utility provided with MySQL 5.7.9 or higher to create your dump files. The mysqldump utility provided in earlier releases uses incorrect syntax for generated column definitions (Bug #20769542). You can use the Information Schema COLUMNS table to identify tables with generated columns.

3. Shut down the old MySQL server. For example:

```
mysqladmin -u root -p shutdown
```

- 4. Install MySQL 8.0. For installation instructions, see Chapter 1, *Installing MySQL*.
- 5. Initialize a new data directory, as described in Section 9.1, "Initializing the Data Directory". For example:

```
mysqld --initialize --datadir=/path/to/8.0-datadir
```

Copy the temporary 'root'@'localhost' password displayed to your screen or written to your error log for later use.

6. Start the MySQL 8.0 server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/8.0-datadir &
```

7. Reset the root password:

```
$> mysql -u root -p
Enter password: **** <- enter temporary root password</pre>
```

```
mysql> ALTER USER USER() IDENTIFIED BY 'your new password';
```

8. Load the previously created dump file into the new MySQL server. For example:

```
mysql -u root -p --force < data-for-upgrade.sql
```

#### Note

It is not recommended to load a dump file when GTIDs are enabled on the server (gtid\_mode=ON), if your dump file includes system tables. mysqldump issues DML instructions for the system tables which use the non-transactional MyISAM storage engine, and this combination is not permitted when GTIDs are enabled. Also be aware that loading a dump file from a server with GTIDs enabled, into another server with GTIDs enabled, causes different transaction identifiers to be generated.

- 9. Perform any remaining upgrade operations:
  - In MySQL 8.0.16 and higher, shut down the server, then restart it with the --upgrade=FORCE option to perform the remaining upgrade tasks:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/8.0-datadir --upgrade=FORCE &
```

Upon restart with <code>--upgrade=FORCE</code>, the server makes any changes required in the <code>mysql</code> system schema between MySQL 5.7 and MySQL 8.0, so that you can take advantage of new privileges or capabilities. It also brings the Performance Schema, <code>INFORMATION\_SCHEMA</code>, and <code>sys</code> schema up to date for MySQL 8.0, and examines all user schemas for incompatibilities with the current version of MySQL.

Prior to MySQL 8.0.16, execute mysql\_upgrade to perform the remaining upgrade tasks:

```
mysql_upgrade -u root -p
```

Then shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/8.0-datadir &
```

#### Note

The upgrade process does not upgrade the contents of the time zone tables. For upgrade instructions, see MySQL Server Time Zone Support.

If the upgrade process uses  $mysql\_upgrade$  (that is, prior to MySQL 8.0.16), the process does not upgrade the contents of the help tables, either. For upgrade instructions in that case, see Server-Side Help Support.

#### **Note**

Loading a dump file that contains a MySQL 5.7 mysql schema re-creates two tables that are no longer used: event and proc. (The corresponding MySQL 8.0 tables are events and routines, both of which are data dictionary tables and are protected.) After you are satisfied that the upgrade was successful, you can remove the event and proc tables by executing these SQL statements:

```
DROP TABLE mysql.event;
DROP TABLE mysql.proc;
```

# MySQL Cluster Upgrade

The information in this section is an adjunct to the in-place upgrade procedure described in In-Place Upgrade, for use if you are upgrading MySQL Cluster.

As of MySQL 8.0.16, a MySQL Cluster upgrade can be performed as a regular rolling upgrade, following the usual three ordered steps:

- 1. Upgrade MGM nodes.
- 2. Upgrade data nodes one at a time.
- Upgrade API nodes one at a time (including MySQL servers).

The way to upgrade each of the nodes remains almost the same as prior to MySQL 8.0.16 because there is a separation between upgrading the data dictionary and upgrading the system tables. There are two steps to upgrading each individual mysqld:

1. Import the data dictionary.

Start the new server with the --upgrade=MINIMAL option to upgrade the data dictionary but not the system tables. This is essentially the same as the pre-MySQL 8.0.16 action of starting the server and not invoking mysql\_upgrade.

The MySQL server must be connected to NDB for this phase to complete. If any NDB or NDBINFO tables exist, and the server cannot connect to the cluster, it exits with an error message:

```
Failed to Populate DD tables.
```

2. Upgrade the system tables.

Prior to MySQL 8.0.16, the DBA invokes the mysql\_upgrade client to upgrade the system tables. As of MySQL 8.0.16, the server performs this action: To upgrade the system tables, restart each individual mysqld without the --upgrade=MINIMAL option.

# 10.8 Upgrading MySQL with the MySQL Yum Repository

For supported Yum-based platforms (see Section 7.1, "Installing MySQL on Linux Using the MySQL Yum Repository", for a list), you can perform an in-place upgrade for MySQL (that is, replacing the old version and then running the new version using the old data files) with the MySQL Yum repository.

#### **Notes**

- Before performing any update to MySQL, follow carefully the instructions in Chapter 10, *Upgrading MySQL*. Among other instructions discussed there, it is especially important to back up your database before the update.
- The following instructions assume you have installed MySQL with the MySQL Yum repository or with an RPM package directly downloaded from MySQL Developer Zone's MySQL Download page; if that is not the case, following the instructions in Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository.

### Selecting a Target Series

By default, the MySQL Yum repository updates MySQL to the latest version in the release series you have chosen during installation (see Selecting a Release Series for details), which means, for example, a 5.7.x installation is *not* updated to a 8.0.x release automatically. To update to another release series, you must first disable the subrepository for the series that has been selected (by default, or by yourself) and enable the subrepository for your target series. To do that, see the general instructions given in Selecting a Release Series. For upgrading from MySQL 5.7 to 8.0, perform the *reverse* of the steps illustrated in Selecting a Release Series, disabling the subrepository for the MySQL 5.7 series and enabling that for the MySQL 8.0 series.

As a general rule, to upgrade from one release series to another, go to the next series rather than skipping a series. For example, if you are currently running MySQL 5.6 and wish to upgrade to 8.0, upgrade to MySQL 5.7 first before upgrading to 8.0.

#### **Important**

For important information about upgrading from MySQL 5.7 to 8.0, see Upgrading from MySQL 5.7 to 8.0.

# Upgrading MySQL

Upgrade MySQL and its components by the following command, for platforms that are not dnf-enabled:

sudo yum update mysql-server

For platforms that are dnf-enabled:

sudo dnf upgrade mysql-server

Alternatively, you can update MySQL by telling Yum to update everything on your system, which might take considerably more time. For platforms that are not dnf-enabled:

sudo yum update

For platforms that are dnf-enabled:

sudo dnf upgrade

# Restarting MySQL

The MySQL server always restarts after an update by Yum. Prior to MySQL 8.0.16, run mysql\_upgrade after the server restarts to check and possibly resolve any incompatibilities between the old data and the upgraded software. mysql\_upgrade also performs other functions; for details, see mysql\_upgrade — Check and Upgrade MySQL Tables. As of MySQL 8.0.16, this step is not required, as the server performs all tasks previously handled by mysql\_upgrade.

You can also update only a specific component. Use the following command to list all the installed packages for the MySQL components (for dnf-enabled systems, replace yum in the command with dnf):

```
sudo yum list installed | grep "^mysql"
```

After identifying the package name of the component of your choice, update the package with the following command, replacing <code>package-name</code> with the name of the package. For platforms that are not dnf-enabled:

sudo yum update package-name

#### For dnf-enabled platforms:

sudo dnf upgrade package-name

### **Upgrading the Shared Client Libraries**

After updating MySQL using the Yum repository, applications compiled with older versions of the shared client libraries should continue to work.

If you recompile applications and dynamically link them with the updated libraries: As typical with new versions of shared libraries where there are differences or additions in symbol versioning between the newer and older libraries (for example, between the newer, standard 8.0 shared client libraries and some older—prior or variant—versions of the shared libraries shipped natively by the Linux distributions' software repositories, or from some other sources), any applications compiled using the updated, newer shared libraries require those updated libraries on systems where the applications are deployed. As expected, if those libraries are not in place, the applications requiring the shared libraries fail. For this reason, be sure to deploy the packages for the shared libraries from MySQL on those systems. To do this, add the MySQL Yum repository to the systems (see Adding the MySQL Yum Repository) and install the latest shared libraries using the instructions given in Installing Additional MySQL Products and Components with Yum.

# 10.9 Upgrading MySQL with the MySQL APT Repository

On Debian and Ubuntu platforms, to perform an in-place upgrade of MySQL and its components, use the MySQL APT repository. See Upgrading MySQL with the MySQL APT Repository in A Quick Guide to Using the MySQL APT Repository.

# 10.10 Upgrading MySQL with the MySQL SLES Repository

On the SUSE Linux Enterprise Server (SLES) platform, to perform an in-place upgrade of MySQL and its components, use the MySQL SLES repository. See Upgrading MySQL with the MySQL SLES Repository in A Quick Guide to Using the MySQL SLES Repository.

# 10.11 Upgrading MySQL on Windows

There are two approaches for upgrading MySQL on Windows:

- Using MySQL Installer
- Using the Windows ZIP archive distribution

The approach you select depends on how the existing installation was performed. Before proceeding, review Chapter 10, *Upgrading MySQL* for additional information on upgrading MySQL that is not specific to Windows.

#### Note

Whichever approach you choose, always back up your current MySQL installation before performing an upgrade. See Database Backup Methods.

Upgrades between non-GA releases (or from a non-GA release to a GA release) are not supported. Significant development changes take place in non-GA releases and you may encounter compatibility issues or problems starting the server.

#### **Note**

MySQL Installer does not support upgrades between *Community* releases and *Commercial* releases. If you require this type of upgrade, perform it using the ZIP archive approach.

### **Upgrading MySQL with MySQL Installer**

Performing an upgrade with MySQL Installer is the best approach when the current server installation was performed with it and the upgrade is within the current release series. MySQL Installer does not support upgrades between release series, such as from 5.7 to 8.0, and it does not provide an upgrade indicator to prompt you to upgrade. For instructions on upgrading between release series, see Upgrading MySQL Using the Windows ZIP Distribution.

To perform an upgrade using MySQL Installer:

- 1. Start MySQL Installer.
- 2. From the dashboard, click **Catalog** to download the latest changes to the catalog. The installed server can be upgraded only if the dashboard displays an arrow next to the version number of the server.
- 3. Click **Upgrade**. All products that have a newer version now appear in a list.

#### Note

MySQL Installer deselects the server upgrade option for milestone releases (Pre-Release) in the same release series. In addition, it displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform an upgrade manually. You can reselect server upgrade and proceed at your own risk.

- 4. Deselect all but the MySQL server product, unless you intend to upgrade other products at this time, and click **Next**.
- 5. Click **Execute** to start the download. When the download finishes, click **Next** to begin the upgrade operation.

Upgrades to MySQL 8.0.16 and higher may show an option to skip the upgrade check and process for system tables. For more information about this option, see Important server upgrade conditions.

6. Configure the server.

# **Upgrading MySQL Using the Windows ZIP Distribution**

To perform an upgrade using the Windows ZIP archive distribution:

- Download the latest Windows ZIP Archive distribution of MySQL from https://dev.mysql.com/downloads/.
- 2. If the server is running, stop it. If the server is installed as a service, stop the service with the following command from the command prompt:

```
C:\> SC STOP mysqld_service_name
```

Alternatively, use NET STOP mysgld service name .

If you are not running the MySQL server as a service, use mysqladmin to stop it. For example, before upgrading from MySQL 5.7 to 8.0, use mysqladmin from MySQL 5.7 as follows:

C:\> "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqladmin" -u root shutdown

#### Note

If the MySQL root user account has a password, invoke mysqladmin with the -p option and enter the password when prompted.

- 3. Extract the ZIP archive. You may either overwrite your existing MySQL installation (usually located at C:\mysql), or install it into a different directory, such as C:\mysql8. Overwriting the existing installation is recommended.
- 4. Restart the server. For example, use the SC START <code>mysqld\_service\_name</code> or NET START <code>mysqld\_service\_name</code> command if you run MySQL as a service, or invoke <code>mysqld</code> directly otherwise.
- 5. Prior to MySQL 8.0.16, run mysql\_upgrade as Administrator to check your tables, attempt to repair them if necessary, and update your grant tables if they have changed so that you can take advantage of any new capabilities. See mysql\_upgrade Check and Upgrade MySQL Tables. As of MySQL 8.0.16, this step is not required, as the server performs all tasks previously handled by mysql\_upgrade.
- 6. If you encounter errors, see Section 5.5, "Troubleshooting a Microsoft Windows MySQL Server Installation".

# 10.12 Upgrading a Docker Installation of MySQL

To upgrade a Docker installation of MySQL, refer to Upgrading a MySQL Server Container.

# 10.13 Upgrade Troubleshooting

- A schema mismatch in a MySQL 5.7 instance between the .frm file of a table and the InnoDB data dictionary can cause an upgrade to MySQL 8.0 to fail. Such mismatches may be due to .frm file corruption. To address this issue, dump and restore affected tables before attempting the upgrade again.
- If problems occur, such as that the new mysqld server does not start, verify that you do not have an old my.cnf file from your previous installation. You can check this with the --print-defaults option (for example, mysqld --print-defaults). If this command displays anything other than the program name, you have an active my.cnf file that affects server or client operation.
- If, after an upgrade, you experience problems with compiled client programs, such as Commands out of sync or unexpected core dumps, you probably have used old header or library files when compiling your programs. In this case, check the date for your mysql.h file and libmysqlclient.a library to verify that they are from the new MySQL distribution. If not, recompile your programs with the new headers and libraries. Recompilation might also be necessary for programs compiled against the shared client library if the library major version number has changed (for example, from libmysqlclient.so.20 to libmysqlclient.so.21).
- If you have created a loadable function with a given name and upgrade MySQL to a version that implements a new built-in function with the same name, the loadable function becomes inaccessible. To correct this, use DROP FUNCTION to drop the loadable function, and then use CREATE FUNCTION to recreate the loadable function with a different nonconflicting name. The same is true if the new version of

MySQL implements a built-in function with the same name as an existing stored function. See Function Name Parsing and Resolution, for the rules describing how the server interprets references to different kinds of functions.

• If upgrade to MySQL 8.0 fails due to any of the issues outlined in Section 10.6, "Preparing Your Installation for Upgrade", the server reverts all changes to the data directory. In this case, remove all redo log files and restart the MySQL 5.7 server on the existing data directory to address the errors. The redo log files (ib\_logfile\*) reside in the MySQL data directory by default. After the errors are fixed, perform a slow shutdown (by setting innodb\_fast\_shutdown=0) before attempting the upgrade again.

# 10.14 Rebuilding or Repairing Tables or Indexes

This section describes how to rebuild or repair tables or indexes, which may be necessitated by:

- Changes to how MySQL handles data types or character sets. For example, an error in a collation might
  have been corrected, necessitating a table rebuild to update the indexes for character columns that use
  the collation.
- Required table repairs or upgrades reported by CHECK TABLE, mysqlcheck, or mysql\_upgrade.

Methods for rebuilding a table include:

- · Dump and Reload Method
- ALTER TABLE Method
- REPAIR TABLE Method

# **Dump and Reload Method**

If you are rebuilding tables because a different version of MySQL cannot handle them after a binary (in-place) upgrade or downgrade, you must use the dump-and-reload method. Dump the tables *before* upgrading or downgrading using your original version of MySQL. Then reload the tables *after* upgrading or downgrading.

If you use the dump-and-reload method of rebuilding tables only for the purpose of rebuilding indexes, you can perform the dump either before or after upgrading or downgrading. Reloading still must be done afterward.

If you need to rebuild an InnoDB table because a CHECK TABLE operation indicates that a table upgrade is required, use mysqldump to create a dump file and mysql to reload the file. If the CHECK TABLE operation indicates that there is a corruption or causes InnoDB to fail, refer to Forcing InnoDB Recovery for information about using the innodb\_force\_recovery option to restart InnoDB. To understand the type of problem that CHECK TABLE may be encountering, refer to the InnoDB notes in CHECK TABLE Statement.

To rebuild a table by dumping and reloading it, use mysqldump to create a dump file and mysql to reload the file:

```
mysqldump db_name t1 > dump.sql
mysql db_name < dump.sql
```

To rebuild all the tables in a single database, specify the database name without any following table name:

```
mysqldump db_name > dump.sql
mysql db_name < dump.sql</pre>
```

To rebuild all tables in all databases, use the --all-databases option:

```
mysqldump --all-databases > dump.sql
mysql < dump.sql</pre>
```

#### **ALTER TABLE Method**

To rebuild a table with ALTER TABLE, use a "null" alteration; that is, an ALTER TABLE statement that "changes" the table to use the storage engine that it already has. For example, if t1 is an InnoDB table, use this statement:

```
ALTER TABLE t1 ENGINE = InnoDB;
```

If you are not sure which storage engine to specify in the ALTER TABLE statement, use SHOW CREATE TABLE to display the table definition.

#### **REPAIR TABLE Method**

The REPAIR TABLE method is only applicable to MyISAM, ARCHIVE, and CSV tables.

You can use REPAIR TABLE if the table checking operation indicates that there is a corruption or that an upgrade is required. For example, to repair a MyISAM table, use this statement:

```
REPAIR TABLE t1;
```

mysqlcheck --repair provides command-line access to the REPAIR TABLE statement. This can be a more convenient means of repairing tables because you can use the --databases or --all-databases option to repair all tables in specific databases or all databases, respectively:

```
mysqlcheck --repair --databases db_name ...
mysqlcheck --repair --all-databases
```

# 10.15 Copying MySQL Databases to Another Machine

In cases where you need to transfer databases between different architectures, you can use <code>mysqldump</code> to create a file containing SQL statements. You can then transfer the file to the other machine and feed it as input to the <code>mysql</code> client.

Use mysqldump --help to see what options are available.

#### Note

If GTIDs are in use on the server where you create the dump (gtid\_mode=ON), by default, mysqldump includes the contents of the gtid\_executed set in the dump to transfer these to the new machine. The results of this can vary depending on the MySQL Server versions involved. Check the description for mysqldump's --set-gtid-purged option to find what happens with the versions you are using, and how to change the behavior if the outcome of the default behavior is not suitable for your situation.

The easiest (although not the fastest) way to move a database between two machines is to run the following commands on the machine on which the database is located:

```
mysqladmin -h 'other_hostname' create db_name
mysqldump db_name | mysql -h 'other_hostname' db_name
```

If you want to copy a database from a remote machine over a slow network, you can use these commands:

```
mysqladmin create db_name
mysqldump -h 'other_hostname' --compress db_name | mysql db_name
```

You can also store the dump in a file, transfer the file to the target machine, and then load the file into the database there. For example, you can dump a database to a compressed file on the source machine like this:

```
mysqldump --quick db_name | gzip > db_name.gz
```

Transfer the file containing the database contents to the target machine and run these commands there:

```
mysqladmin create db_name
gunzip < db_name.gz | mysql db_name
```

You can also use mysqldump and mysqlimport to transfer the database. For large tables, this is much faster than simply using mysqldump. In the following commands, <code>DUMPDIR</code> represents the full path name of the directory you use to store the output from mysqldump.

First, create the directory for the output files and dump the database:

```
mkdir DUMPDIR
mysqldump --tab=DUMPDIR
db_name
```

Then transfer the files in the *DUMPDIR* directory to some corresponding directory on the target machine and load the files into MySQL there:

Do not forget to copy the mysql database because that is where the grant tables are stored. You might have to run commands as the MySQL root user on the new machine until you have the mysql database in place.

After you import the <code>mysql</code> database on the new machine, execute <code>mysqladmin flush-privileges</code> so that the server reloads the grant table information.

# Chapter 11 Downgrading MySQL

Downgrading from MySQL 8.0 to MySQL 5.7 is not supported.

In-place downgrades are supported from within the MySQL 8.0 series as of MySQL 8.0.35. In-place means starting and running a new MySQL server binary on an existing MySQL data directory that was created by a different MySQL server version.

Attempting to downgrade below MySQL 8.0.35 yields an error similar to:

```
[ERROR] [MY-013171] [InnoDB] Cannot boot server version 80034 on data directory built by version 80035. Do
```

Here's a successful log message from an in-place MySQL 8.0.36 to 8.0.35 downgrade:

```
[System] [MY-014064] [Server] Server downgrade from '80036' to '80035' started.
[System] [MY-014064] [Server] Server downgrade from '80036' to '80035' completed.
```

An alternative is to restore a backup taken before an upgrade.

# Chapter 12 Environment Variables

This section lists environment variables that are used directly or indirectly by MySQL. Most of these can also be found in other places in this manual.

Options on the command line take precedence over values specified in option files and environment variables, and values in option files take precedence over values in environment variables. In many cases, it is preferable to use an option file instead of environment variables to modify the behavior of MySQL. See Using Option Files.

Variable	Description
AUTHENTICATION_KERBEROS_CLIENT_LOG	Kerberos authentication logging level.
AUTHENTICATION_LDAP_CLIENT_LOG	Client-side LDAP authentication logging level.
AUTHENTICATION_PAM_LOG	PAM authentication plugin debug logging settings.
CC	The name of your C compiler (for running CMake).
CXX	The name of your C++ compiler (for running CMake).
cc	The name of your C compiler (for running CMake).
DBI_USER	The default user name for Perl DBI.
DBI_TRACE	Trace options for Perl DBI.
HOME	The default path for the mysql history file is \$HOME/.mysql_history.
LD_RUN_PATH	Used to specify the location of libmysqlclient.so.
LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN	Enable mysql_clear_password authentication plugin; see Client-Side Cleartext Pluggable Authentication.
LIBMYSQL_PLUGIN_DIR	Directory in which to look for client plugins.
LIBMYSQL_PLUGINS	Client plugins to preload.
MYSQL_DEBUG	Debug trace options when debugging.
MYSQL_GROUP_SUFFIX	Option group suffix value (like specifying defaults-group-suffix).
MYSQL_HISTFILE	The path to the mysql history file. If this variable is set, its value overrides the default for \$HOME/.mysql_history.
MYSQL_HISTIGNORE	Patterns specifying statements that mysql should not log to \$HOME/.mysql_history, or syslog ifsyslog is given.
MYSQL_HOME	The path to the directory in which the server-specific my.cnf file resides.
MYSQL_HOST	The default host name used by the mysql command-line client.
MYSQL_OPENSSL_UDF_DH_BITS_THRESHOLD	Maximum key length for create_dh_parameters(). See MySQL Enterprise Encryption Usage and Examples.

Variable	Description
MYSQL_OPENSSL_UDF_DSA_BITS_THRESHOLD	Maximum DSA key length for create_asymmetric_priv_key(). See MySQL Enterprise Encryption Usage and Examples.
MYSQL_OPENSSL_UDF_RSA_BITS_THRESHOLD	Maximum RSA key length for create_asymmetric_priv_key(). See MySQL Enterprise Encryption Usage and Examples.
MYSQL_PS1	The command prompt to use in the mysql command-line client.
MYSQL_PWD	The default password when connecting to mysqld. Using this is insecure. See note following table.
MYSQL_TCP_PORT	The default TCP/IP port number.
MYSQL_TEST_LOGIN_FILE	The name of the .mylogin.cnf login path file.
MYSQL_TEST_TRACE_CRASH	Whether the test protocol trace plugin crashes clients. See note following table.
MYSQL_TEST_TRACE_DEBUG	Whether the test protocol trace plugin produces output. See note following table.
MYSQL_UNIX_PORT	The default Unix socket file name; used for connections to localhost.
MYSQLX_TCP_PORT	The X Plugin default TCP/IP port number.
MYSQLX_UNIX_PORT	The X Plugin default Unix socket file name; used for connections to localhost.
NOTIFY_SOCKET	Socket used by mysqld to communicate with systemd.
PATH	Used by the shell to find MySQL programs.
PKG_CONFIG_PATH	Location of mysqlclient.pc pkg-config file. See note following table.
TMPDIR	The directory in which temporary files are created.
TZ	This should be set to your local time zone. See Time Zone Problems.
UMASK	The user-file creation mode when creating files. See note following table.
UMASK_DIR	The user-directory creation mode when creating directories. See note following table.
USER	The default user name on Windows when connecting to mysqld.

For information about the mysql history file, see mysql Client Logging.

Use of MYSQL\_PWD to specify a MySQL password must be considered *extremely insecure* and should not be used. Some versions of ps include an option to display the environment of running processes. On some systems, if you set MYSQL\_PWD, your password is exposed to any other user who runs ps. Even on systems without such a version of ps, it is unwise to assume that there are no other methods by which users can examine process environments.

MYSQL\_PWD is deprecated as of MySQL 8.0; expect it to be removed in a future version of MySQL.

MYSQL\_TEST\_LOGIN\_FILE is the path name of the login path file (the file created by mysql\_config\_editor). If not set, the default value is %APPDATA%\MySQL\.mylogin.cnf directory on Windows and \$HOME/.mylogin.cnf on non-Windows systems. See mysql\_config\_editor — MySQL Configuration Utility.

The MYSQL\_TEST\_TRACE\_DEBUG and MYSQL\_TEST\_TRACE\_CRASH variables control the test protocol trace client plugin, if MySQL is built with that plugin enabled. For more information, see Using the Test Protocol Trace Plugin.

The default UMASK and UMASK\_DIR values are 0640 and 0750, respectively. MySQL assumes that the value for UMASK or UMASK\_DIR is in octal if it starts with a zero. For example, setting UMASK=0600 is equivalent to UMASK=384 because 0600 octal is 384 decimal.

The UMASK and UMASK\_DIR variables, despite their names, are used as modes, not masks:

- If UMASK is set, mysqld uses (\$UMASK | 0600) as the mode for file creation, so that newly created files have a mode in the range from 0600 to 0666 (all values octal).
- If UMASK\_DIR is set, mysqld uses (\$UMASK\_DIR | 0700) as the base mode for directory creation, which then is AND-ed with ~(~\$UMASK & 0666), so that newly created directories have a mode in the range from 0700 to 0777 (all values octal). The AND operation may remove read and write permissions from the directory mode, but not execute permissions.

See also Problems with File Permissions.

It may be necessary to set PKG\_CONFIG\_PATH if you use pkg-config for building MySQL programs. See Building C API Client Programs Using pkg-config.

# Chapter 13 Perl Installation Notes

### **Table of Contents**

13.1 Installing Perl on Unix	245
13.2 Installing ActiveState Perl on Windows	246
13.3 Problems Using the Perl DBI/DBD Interface	246

The Perl DBI module provides a generic interface for database access. You can write a DBI script that works with many different database engines without change. To use DBI, you must install the DBI module, as well as a DataBase Driver (DBD) module for each type of database server you want to access. For MySQL, this driver is the DBD::mysql module.

#### Note

Perl support is not included with MySQL distributions. You can obtain the necessary modules from <a href="http://search.cpan.org">http://search.cpan.org</a> for Unix, or by using the ActiveState ppm program on Windows. The following sections describe how to do this.

The DBI/DBD interface requires Perl 5.6.0, and 5.6.1 or later is preferred. DBI *does not work* if you have an older version of Perl. You should use DBD::mysql 4.009 or higher. Although earlier versions are available, they do not support the full functionality of MySQL 8.0.

# 13.1 Installing Perl on Unix

MySQL Perl support requires that you have installed MySQL client programming support (libraries and header files). Most installation methods install the necessary files. If you install MySQL from RPM files on Linux, be sure to install the developer RPM as well. The client programs are in the client RPM, but client programming support is in the developer RPM.

The files you need for Perl support can be obtained from the CPAN (Comprehensive Perl Archive Network) at http://search.cpan.org.

The easiest way to install Perl modules on Unix is to use the CPAN module. For example:

```
$> perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

The DBD::mysql installation runs a number of tests. These tests attempt to connect to the local MySQL server using the default user name and password. (The default user name is your login name on Unix, and ODBC on Windows. The default password is "no password.") If you cannot connect to the server with those values (for example, if your account has a password), the tests fail. You can use force install DBD::mysql to ignore the failed tests.

DBI requires the Data::Dumper module. It may be installed; if not, you should install it before installing DBI.

It is also possible to download the module distributions in the form of compressed tar archives and build the modules manually. For example, to unpack and build a DBI distribution, use a procedure such as this:

1. Unpack the distribution into the current directory:

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

This command creates a directory named DBI-VERSION.

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

```
$> cd DBI-VERSION
```

3. Build the distribution and compile everything:

```
$> perl Makefile.PL
$> make
$> make test
$> make install
```

The make test command is important because it verifies that the module is working. Note that when you run that command during the DBD::mysql installation to exercise the interface code, the MySQL server must be running or the test fails.

It is a good idea to rebuild and reinstall the DBD::mysql distribution whenever you install a new release of MySQL. This ensures that the latest versions of the MySQL client libraries are installed correctly.

If you do not have access rights to install Perl modules in the system directory or if you want to install local Perl modules, the following reference may be useful: http://learn.perl.org/faq/perlfaq8.html#How-do-l-keep-my-own-module-library-directory-

# 13.2 Installing ActiveState Perl on Windows

On Windows, you should do the following to install the MySQL DBD module with ActiveState Perl:

- 1. Get ActiveState Perl from http://www.activestate.com/Products/ActivePerl/ and install it.
- 2. Open a console window.
- 3. If necessary, set the HTTP\_proxy variable. For example, you might try a setting like this:

```
C:\> set HTTP_proxy=my.proxy.com:3128
```

4. Start the PPM program:

```
C:\> C:\perl\bin\ppm.pl
```

5. If you have not previously done so, install DBI:

```
ppm> install DBI
```

6. If this succeeds, run the following command:

```
ppm> install DBD-mysql
```

This procedure should work with ActiveState Perl 5.6 or higher.

If you cannot get the procedure to work, you should install the ODBC driver instead and connect to the MySQL server through ODBC:

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn",$user,$password) ||
die "Got error $DBI::errstr when connecting to $dsn\n";
```

# 13.3 Problems Using the Perl DBI/DBD Interface

If Perl reports that it cannot find the ../mysql/mysql.so module, the problem is probably that Perl cannot locate the libmysqlclient.so shared library. You should be able to fix this problem by one of the following methods:

- Copy libmysqlclient.so to the directory where your other shared libraries are located (probably / usr/lib or /lib).
- Modify the -L options used to compile DBD::mysql to reflect the actual location of libmysqlclient.so.
- On Linux, you can add the path name of the directory where libmysqlclient.so is located to the / etc/ld.so.conf file.
- Add the path name of the directory where <code>libmysqlclient.so</code> is located to the <code>LD\_RUN\_PATH</code> environment variable. Some systems use <code>LD\_LIBRARY\_PATH</code> instead.

Note that you may also need to modify the -L options if there are other libraries that the linker fails to find. For example, if the linker cannot find libc because it is in /lib and the link command specifies -L/usr/lib, change the -L option to -L/lib or add -L/lib to the existing link command.

If you get the following errors from DBD::mysql, you are probably using gcc (or using an old binary compiled with gcc):

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Add -L/usr/lib/gcc-lib/... -lgcc to the link command when the mysql.so library gets built (check the output from make for mysql.so when you compile the Perl client). The -L option should specify the path name of the directory where libgcc.a is located on your system.

Another cause of this problem may be that Perl and MySQL are not both compiled with gcc. In this case, you can solve the mismatch by compiling both with gcc.