MySQL Installation Guide

Abstract

This is the MySQL Installation Guide from the MySQL 8.3 Reference Manual.

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

Document generated on: 2024-04-01 (revision: 78239)

Table of Contents

Preface and Legal Notices

This is the MySQL Installation Guide from the MySQL 8.3 Reference Manual.

Licensing information—MySQL 8.1. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 8.1, see the MySQL 8.1 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 8.1, see the MySQL 8.1 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

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 8.3 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 https://www.mysql.com/support/supportedplatforms/database.html on the MySQL website.

2. **Choose which track to install.**

   MySQL offers a bugfix track (such as MySQL 8.0), and an innovation track (today it's MySQL 8.3) 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 8.0.*x* upgrading to 8.0.*y*, while innovation track releases typically only have minor releases, such as MySQL 8.3.0 upgrading to 9.0.0. However, an innovation track does have the occasional point release.

3. **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”.

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 MSI installer or Zipped binary, see Chapter 5, *Installing MySQL on Microsoft Windows*.

  For details and instructions on building MySQL from source code, 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

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 8.0, or use an innovation release like MySQL 8.3. 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 8.3 uses release names that consist of three numbers and an optional suffix (for example, **mysql-8.3.0**). The numbers within the release name are interpreted as follows:

- The first number (**8**) is the major version number.
- The second number (**3**) 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 (**0**) 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 might use an MSI 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, “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-8.3.0-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945  mysql-standard-8.3.0-linux-i686.tar.gz
```

```
$> md5.exe mysql-installer-community-8.3.0.msi
aaab65abbec64d5e907dcd41b8699945  mysql-installer-community-8.3.0.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 `GnuPG` (GNU Privacy Guard). `GnuPG` is an Open Source alternative to the well-known Pretty Good Privacy (`PGP`) by Phil Zimmermann. Most Linux distributions ship with `GnuPG` installed by default. Otherwise, see http://www.gnupg.org/ for more information about `GnuPG` 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+SqLUIZ3uWZQm6tsNhvTnahvPPZBGdl99iW
YTt2KmXp0KeN2s9pmLKkGAbacQP1RqzMFnoHawSMf0qTUVjAvhnI4+qzMDjTNSBq9fa3nHmO
YxownnrRkpiQUM/yD7/JmVENgwWb6akZeGYrXch9jd4XV3t8OD6TGzTedTki0TDNr6YZYhC7
jUm9fK9Zs299pzOXSxRRNGd+3H9gbXizrBu4L/3lUrNf//rM7OvV9Ho7u9YYyAQ3L3+OABK9
FKHNhrpi8Q0cbhvWkD4oCKJ+YZ54XrOG0YTg/YUAs5/3//FATI1sWdtLjJ5pSb0onV3LIbar
```

```
RTN8lC4Le/5kd3lcot9J8b3EMXL5p9OGW7wBfmNVRSUI74Vmwt+v9gyp0Hd0keRCUn8lo/1V
0YD9i92KsE+/IqoYTjnya/5kX41jB8vr1ebkHFuJ404+G6ETd0owwxq64jLIcsp/GBZHGU0R
KKAo9DRLH7rpQ7PVlnw8TDNlOtWt5EJlBXFcPL+NgWbqkADAyA/XSNeWlqonvPlYfmasnAHA
pMd9NhPQhC7hJTjCiAwG8UyWpV8Dj07DHFQ5xBbkTnKH2OrJtguPqSNYtTASbsWz09S8ujoT
DXFT17NbFM2dMIiq0a4VQB3SzH13H2io9Cbg/TzJrJGmwgoXgwARAQABtDZNeVNRTCBSZWxl
YXNlIEVuZ2luZWVyaW5nIDxteXNxbC1idWlsZEBvc3Mub3JhY2xlLmNvbT6JAlQEEwEIAD4W
IQS8pDQXw7SF3RKOxtS3s7eIqNN4XAUCZTas2gIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgID
AQIeAQIXgAAKCRC3s7eIqNN4XLzoD/9PlpWtfHlI8eQTHwGsGIwFA+fgipyDElapHw3MO+K9
VOEYRZCZSuBXHJe9kjGEVCGUDrfImvgTuNuqYmVUV+wyhP+w46W/cWVkqZKAW0hNp0TTvu3e
Dwap7gdk80VF24Y2Wo0bbiGkpPiPmB59oybGKaJ756JlKXIL4hTtK3/hjIPFnb64Ewe4YLZy
oJu0fQOyA8gXuBoalHhUQTbRpXI0XI3tpZiQemNbfBfJqXo6LP3/LgChAuOfHIQ8alvnhCwx
hNUSYGIRqx+BEbJw1X99Az8XvGcZ36VOQAZztkW7mEfH9NDPz7MXwoEvduc61xwlMvEsUIaS
fn6SGLFzWPClA98UMSJgF6sKb+JNoNbzKaZ8V5w13msLb/pq7hab72HH99XJbyKNliYj3+KA
3q0YLf+Hgt4Y4EhIJ8x2+g690Np7zJF4KXNFbi1BGloLGm78akY1rQlzpndKSpZq5KWw8FY/
1PEXORezg/BPD3Etp0AVKff4YdrDlOkNB7zoHRfFHAvEuuqti8aMBrbRnRSG0xunMUOEhbYS
/wOOTl0g3bF9NpAkfU1Fun57N96Us2T9gKo9AiOY5DxMe+IrBg4zaydEOovgqNi2wbU0MOBQ
b23Puhj7ZCIXcpILvcx9ygjkONr75w+XQrFDNeux4Znzay3ibXtAPqEykPMZHsZ2sbkCDQRl
NqzaARAAsdvBo8WRqZ5WVVk6lReD8b6Zx83eJUkV254YX9zn5t8KDRjYOySwS75mJIaZLsv0
YQjJk+5rt10tejyCrJIFo9CMvCmjUKtVbgmhfS5+fUDRrYCEZBBSa0Dvn68EBLiHugr+SPXF
6o1hXEUqdMCpB6oVp6X45JVQroCKIH5vsCtw2jU8S2/IjjV0V+E/zitGCiZaoZ1f6NG7ozyF
ep1CSAReZu/sssk0pCLlfCebRd9Rz3QjSrQhWYuJa+eJmiF4oahnpUGktxMD632I9aG+IMfj
tNJNtX32MbO+Se+cCtVc3cxSa/pR+89a3cb9IBA5tFF2Qoekhqo/1mmLi93Xn6uDUhl5tVxT
nB217dBT27tw+p0hjd9hXZRQbrIZUTyh3+8EMfmAjNSIeR+th86xRd9XFRr9EOqrydnALOUr
9cT7TfXWGEkFvn6ljQX7f4RvjJOTbc4jJgVFyu8K+VU6u1NnFJgDiNGsWvnYxAf7gDDbUSXE
uC2anhWvxPvpLGmsspngge4yl+3nv+UqZ9sm6LCebR/7UZ67tYz3p6xzAOVgYsYcxoIUuEZX
jHQtsYfTZZhrjUWBJ09jrMvlKUHLnS437SLbgoXVYZmcqwAWpVNOLZf+fFm4IE5aGBG5Dho2
CZ6ujngW9Zkn98T1d4N0MEwwXa2V6T1ijzcqD7GApZUAEQEAAYkCPAQYAQgAJhYhBLykNBfD
tIXdEo7G1Lezt4io03hcBQJlNqzaAhsMBQkDwmcAAAoJELezt4io03hcXqMP/01aPT3A3Sg7
oTQoHdCxj04ELkzrezNWGM+YwbSKrR2LoXR8zf2tBFzc2/Tl98V0+68f/eCvkvqCuOtq4392
Ps23j9W3r5XG+GDOwDsx0gl0E+Qkw07pwdJctA6efsmnRkjF2YVO0N9MiJA1tc8NbNXpEEHJ
Z7F8Ri5cpQrGUz/AY0eae2b7QefyP4rpUELpMZPjc8Px39Fe1DzRbT+5E19TZbrpbwlSYs1i
CzS5YGFmpCRyZcLKXo3zS6N22+82cnRBSPPipiO6WaQawcVMlQO1SX0giB+3/DryfN9VuIYd
1EWCGQa3O0MVu6o5KVHwPgl9R1P6xPZhurkDpAd0b1s4fFxin+MdxwmG7RslZA9CXRPpzo7/
fCMW8sYOH15DP+YfUckoEreBt+zezBxbIX2CGGWEV9v3UBXadRtwxYQ6sN9bqW4jm1b41vNA
17b6CVH6sVgtU3eN+5Y9an1e5jLD6kFYx+OIeqIIId/TEqwS61csY9aav4j4KLOZFCGNU0FV
ji7NQewSpepTcJwfJDOzmtiDP4vol1ApJGLRwZZZ9PB6wsOgDOoP6sr0YrDI/NNX2RyXXbgl
nQ1yJZVSH3/3eo6knG2qTthUKHCRDNKdy9Qqc1x4WWWtSRjh+zX8AvJK2q1rVLH2/3ilxe9w
cAZUlaj3id3TxquAlud4lWDz
=h5nH
-----END PGP PUBLIC KEY BLOCK-----
```


To import the build key into your personal public GPG keyring, use `gpg --import`. For example, if you have saved the key in a file named `mysql_pubkey.asc`, 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 `.asc` 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
Distribution file	`mysql-standard-8.3.0-linux-i686.tar.gz`
Signature file	`mysql-standard-8.3.0-linux-i686.tar.gz.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:

```
$> gpg --verify mysql-standard-8.3.0-linux-i686.tar.gz.asc
gpg: Signature made Fri 13 Oct 2023 01:53:29 AM PDT
gpg:                using RSA key 859BE8D7C586F538430B19C2467B942D3A79BD29
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
```

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-standard-8.3.0-linux-i686.tar.gz.asc
gpg: Signature made Fri 13 Oct 2023 01:53:29 AM PDT
gpg:                using RSA key 859BE8D7C586F538430B19C2467B942D3A79BD29
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: 859B E8D7 C586 F538 430B  19C2 467B 942D 3A79 BD29
```

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.

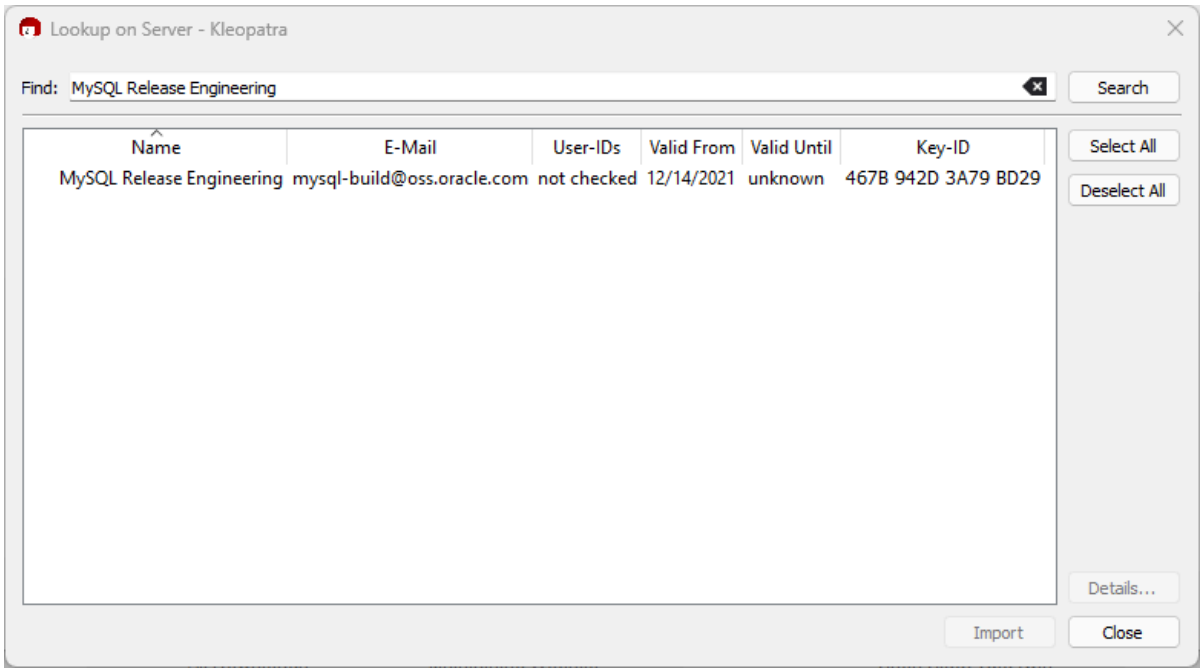
See the GPG documentation for more information on how to work with public keys.

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

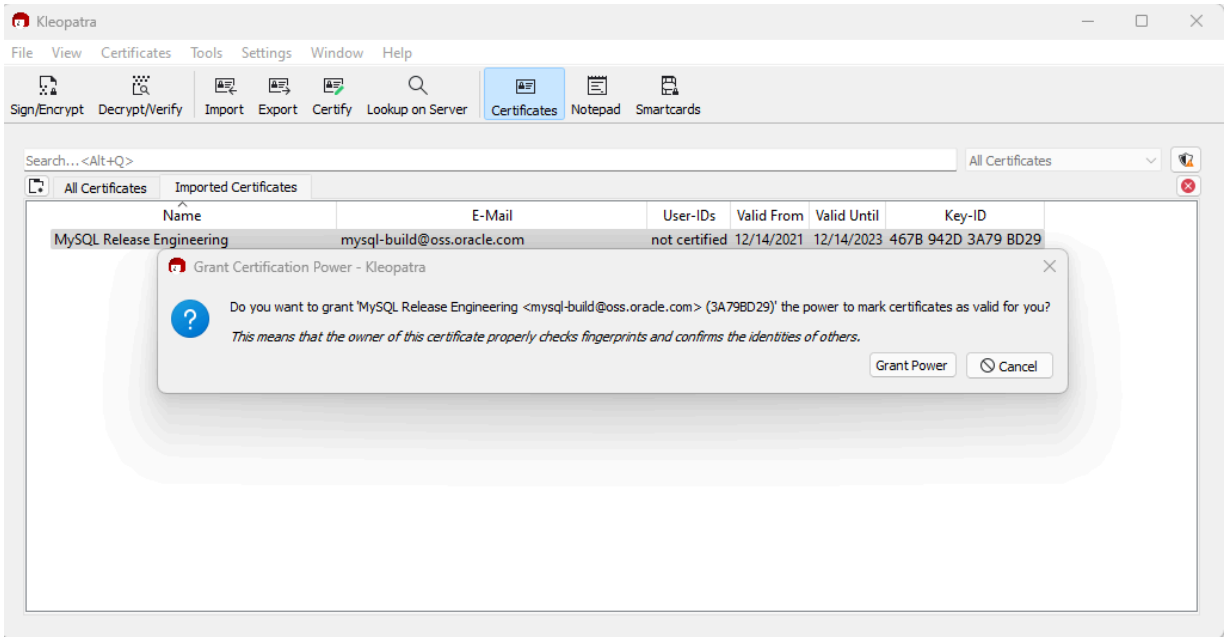
Figure 2.1 Kleopatra: Lookup on Server Wizard: Finding a Certificate



Select the "MySQL Release Engineering" certificate. The Key-ID must reference "3A79 BD29", 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, grant trust to the certificate. Select our certificate, then from the main menu select **Certificates**, **Change Certification Power**, and click **Grant Power**.

Figure 2.2 Kleopatra: Grant Certification Power 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.

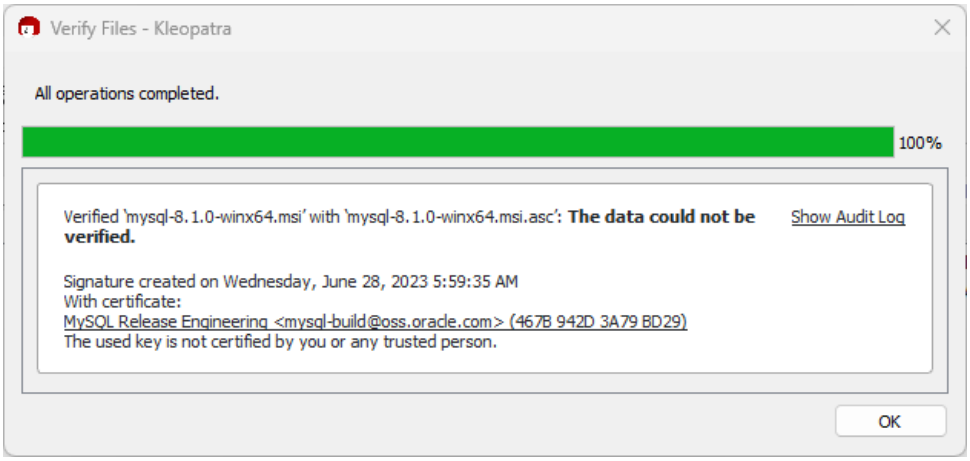
Table 2.2 MySQL Package and Signature Files for MySQL Server MSI for Microsoft Windows

File Type	File Name
Distribution file	`mysql-8.3.0-winx64.msi`
Signature file	`mysql-8.3.0-winx64.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. Load the dialog from **File**, **Decrypt/Verify Files...**, and then choose the `.asc` file.

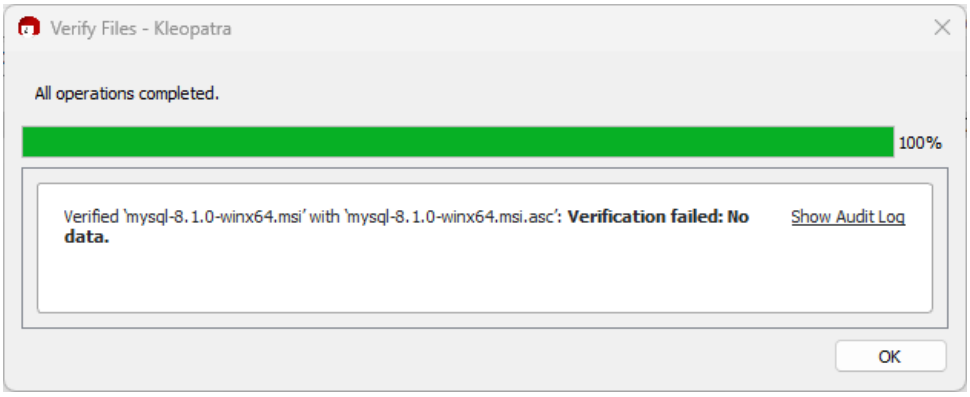
The two most common results look like the following figures; and although the "The data could not be verified." warning looks problematic, the file check passed with success. For additional information on what this warning means, click **Show Audit Log** and compare it to Section 2.4.2, “Signature Checking Using GnuPG”. You may now execute the MSI file.

Figure 2.3 Kleopatra: the Decrypt and Verify Results Dialog: Success



Seeing an error such as `Verification failed: No Data.` means the file is invalid. Do not execute the MSI file if you see this error.

Figure 2.4 Kleopatra: the Decrypt and Verify Results Dialog: Bad



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-8.3.0-1.el8.x86_64.rpm
mysql-community-server-8.3.0-1.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#a8d3785c)`, 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 a8d3785c > a8d3785c.asc
$> rpm --import a8d3785c.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
Si3Z3n3jlEg+mSdhAvW+b+hFnqxo+TY0U+RBwDi4oO0YzHefkYPSmNPdlxRPQBMv
4GPTNfxERx6XvVSPcL1+jQ4R2cQFBryNhidBFIkoCOszjWhm+WnbURsLheBp757l
qEyrpCufz77zlq2gEi+wtPHItfqsx3rzxSRqatztMGYZpNUHNBJkr13npZtGW+kd
N/xu980QLZxN+bZ88pNoOuzD6dKcpMJ0LkdUmTx5z9ewiFiFbUDzZ7PECOm2g3ve
Jrwr79CXDLE1+39Hr8rDM2kDhSr9tAlPTnHVDcaYIGgSNIBcYfLmt91133klHQHB
IdWCNVtWJjq5YcLQJ9TxG9GQzgABPrm6NDd1t9j7w1L7uwBvMB1wgpirRTPVfnUS
Cd+025PEF+wTcBhfnzLtFj5xD7mNsmDmeHkF/sDfNOfAzTE1v2wq0ndYU60xbL6/
yl/Nipyr7WiQjCG0m3WfkjjVDTfs7/DXUqHFDOu4WMF9v+oqwpJXmAeGhQTWZC/Q
hWtrjrNJAgwKpp263gDSdW70ekhRzsok1HJwX1SfxHJYCMFs2aH6ppzNsQARAQAB
tDZNeVNRTCBSZWxlYXNlIEVuZ2luZWVyaW5nIDxteXNxbC1idWlsZEBvc3Mub3Jh
Y2xlLmNvbT6JAlQEEwEIAD4WIQSFm+jXxYb1OEMLGcJGe5QtOnm9KQUCYbi6twIb
AwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRBGe5QtOnm9KUewD/99
2sS31WLGoUQ6NoL7qOB4CErkqXtMzpJAKKg2jtBGG3rKE1/0VAg1D8AwEK4LcCO4
07wohnH0hNiUbeDck5x20pgS5SplQpuXX1K9vPzHeL/WNTb98S3H2Mzj4o9obED6
Ey52tTupttMF8pC9TJ93LxbJlCHIKKwCA1cXud3GycRN72eqSqZfJGdsaeWLmFmH
f6oee27d8XLoNjbyAxna/4jdWoTqmp8oT3bgv/TBco23NzqUSVPi+7ljS1hHvcJu
oJYqaztGrAEf/lWIGdfl/kLEh8IYx8OBNUojh9mzCDlwbs83CBqoUdlzLNDdwmzu
34Aw7xK14RAVinGFCpo/7EWoX6weyB/zqevUIIE89UABTeFoGih/hx2jdQV/NQNt
hWTW0jH0hmPnajBVAJPYwAuO82rx2pnZCxDATMn0elOkTue3PCmzHBF/GT6c65aQ
C4aojj0+Veh787QllQ9FrWbwnTz+4fNzU/MBZtyLZ4JnsiWUs9eJ2V1g/A+RiIKu
357Qgy1ytLqlgYiWfzHFlYjdtbPYKjDaScnvtY8VO2Rktm7XiV4zKFKiaWp+vuVY
pR0/7Adgnlj5Jt9lQQGOr+Z2VYx8SvBcC+by3XAtYkRHtX5u4MLlVS3gcoWfDiWw
CpvqdK21EsXjQJxRr3dbSn0HaVj4FJZX0QQ7WZm6WLkCDQRhuLq3ARAA6RYjqfC0
YcLGKvHhoBnsX29vy9Wn1y2JYpEnPUIB8X0VOyz5/ALv4Hqtl4THkH+mmMuhtndo
q2BkCCk508jWBvKS1S+Bd2esB45BDDmIhuX3ozu9Xza4i1FsPnLkQ0uMZJv30ls2
pXFmskhYyzmo6aOmH2536LdtPSlXtywfNV1HEr69V/AHbrEzfoQkJ/qvPzELBOjf
jwtDPDePiVgW9LhktzVzn/BjO7XlJxw4PGcxJG6VApsXmM3t2fPN9eIHDUq8ocbH
dJ4en8/bJDXZd9ebQoILUuCg46hE3p6nTXfnPwSRnIRnsgCzeAz4rxDR4/Gv1Xpz
v5wqpL21XQi3nvZKlcv7J1IRVdphK66De9GpVQVTqC102gqJUErdjGmxmyCA1OOO
RqEPfKTrXz5YUGsWwpH+4xCuNQP0qmreRw3ghrH8potIr0iOVXFic5vJfBTgtcuE
B6E6ulAN+3jqBGTaBML0jxgj3Z5VC5HKVbpg2DbB/wMrLwFHNAbzV5hj2Os5Zmva
0ySP1YHB26pAW8dwB38GBaQvfZq3ezM4cRAo/iJ/GsVE98dZEBO+Ml+0KYj+ZG+v
yxzo20sweun7ZKT+9qZM90f6cQ3zqX6IfXZHHmQJBNv73mcZWNhDQOHs4wBoq+FG
QWNqLU9xaZxdXw80r1viDAwOy13EUtcVbTkAEQEAAYkCPAQYAQgAJhYhBIWb6NfF
hvU4QwsZwkZ7lC06eb0pBQJhuLq3AhsMBQkDwmcAAAoJEEZ7lC06eb0pSi8P/iy+
dNnxrtiENn9vkkA7AmZ8RsvPXYVeDCDSsL7UfhbS77r2L1qTa2aB3gAZUDIOXln5
```

```
1lSxMeeLtOequLMEV2Xi5km70rdtnja5SmWfc9fyExunXnsOhg6UG872At5CGEZU
0c2Nt/hlGtOR3xbt3O/Uwl+dErQPA4BUbW5K1T7OC6oPvtlKfF4bGZFloHgt2yE9
YSNWZsTPe6XJSapemHZLPOxJLnhs3VBirWE31QS0bRl5AzlO/fg7ia65vQGMOCOT
LpgChTbcZHtozeFqva4IeEgE4xN+6r8WtgSYeGGDRmeMEVjPM9dzQObf+SvGd58u
2z9f2agPK1H32c69RLoA0mHRe7Wkv4izeJUc5tumUY0e8OjdenZZjT3hjLh6tM+m
rp2oWnQIoed4LxUw1dhMOj0rYXv6laLGJ1FsW5eSke7ohBLcfBBTKnMCBohROHy2
E63Wggfsdn3UYzfqZ8cfbXetkXuLS/OM3MXbiNjg+ElYzjgWrkayu7yLakZx+mx6
sHPIJYm2hzkniMG29d5mGl7ZT9emP9b+CfqGUxoXJkjs0gnDl44bwGJ0dmIBu3aj
VAaHODXyY/zdDMGjskfEYbNXCAY2FRZSE58tgTvPKD++Kd2KGplMU2EIFT7JYfKh
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/nJKZyQT7h9bPlUWC3RODjQRey
CITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZfw2vOUgCmYv2hW0h
yDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3BqOxRznNCRCRxAuAuVztHRcE
AJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNWhxwR9pRWVArNYJdDRT+rf2RUe3vpquKN
QU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLVK2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sB
gACyP/Vb7hiPwxh6rDZ7ITnEkYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgt
obZX9qnrAXw+uNDIQJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAn
WqcyefeprV0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q2TXlTUUwgUmVs
ZWFzZSBFbmdpbmVlcmluZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iEYEEBECAAYF
AlldBJ4ACgkQvcMmpx2w8a2MYQCgga9wXfwOe/52xg0RTkhsbDQhvdAAn30njwoLBhKdDBxk
hVmwZQvzdYYNiGYEExECACYCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAUCTnc+KgUJE/sC
FQAKCRCMcY07UHLh9SbMAJ4l1+qBz2BZNSGCZwwA6YbhGPC7FwCgp8z5TzIw4YQuL5NGJ/sy
0oSazqmIZgQTEQIAJgUCTnc9dgIbIwUJEPPzpwYLCQgHAwIEFQIIAwQWAgMBAh4BAheAAAoJ
EIxxjTtQcuH1Ut4AoIKjhdf70899d+7JFq3LD7zeeyI0AJ9Z+YyE1HZSnzYi73brScilbIV6
sYhpBBMRAgApAhsjBgsJCAcDAgQVAggDBBYCAwECHgECF4ACGQEFAlGUkToFCRU3IaoACgkQ
jHGNO1By4fWLQACfV6wP8ppZqMz2Z/gPZbPP7sDHE7EAn2kDDatXTZIR9pMgcnN0cff1tsX6
iGkEExECACkCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAIZAQUCUwHUZgUJGmbLywAKCRCM
cY07UHLh9V+DAKCjS1gGwgVI/eut+5L+l2v3ybl+ZgCcD7ZoA341HtoroV3U6xRD09fUgeqI
bAQTEQIALAIbIwIeAQIXgAIZAQYLCQgHAwIGFQoJCAIDBRYCAwEABQJYpXsIBQkeKT7NAAoJ
EIxxjTtQcuH1wrMAnRGuZVbriMR077KTGAVhJF2uKJiPAJ9rCpXYFve2IdxST2i7w8nygefV
a4hsBBMRAgAsAhsjAh4BAheAAhkBBgsJCAcDAgYVCgkIAgMFFgIDAQAFAlinBSAFCR4qyRQA
CgkQjHGNO1By4fVXBQCeOqVMlXfAWdq+QqaTAtbZskN3HkYAn1T8LlbIktFREeVlKrQEA7fg
6HrQiGwEExECACwCGyMCHgECF4ACGQEGCwkIBwMCBhUKCQgCAwUWAgMBAAUCXEBY+wUJI87e
5AAKCRCMcY07UHLh9RZPAJ9uvm0zlzfCN+DHxHVaoFLFjdVYTQCfborsC9tmEZYawhhogjeB
kZkorbyJARwEEAECAAYFAlAS6+UACgkQ8aIC+GoXHivrWwf/dtLk/x+NC2VMDlg+vOeM0qgG
1IlhXZfiNsEisvvGaz4m8fSFRGe+1bvvfDoKRhxiGXU48Rusjixzvbb6KTMuY6JpOVfz9Dj3
H9spYriHa+i6rYySXZIpOhfLiMnTy7NH2OvYCyNzSS/ciIUACIfH/2NH8zNT5CNF1uPNRs7H
sHzzz7pOlTjtTWiF4cq/Ij6Z6CNrmdj+SiMvjYN9u6sdEKGtoNtpycgD5HGKR+I7Nd/7v56y
haUe4FpuvsNXig86K9tI6MUFS8CUyy7Hj3kVBZOUWVBM053knGdALSygQr50DA3jMGKVl4Zn
Hje2RVWRmFTr5YWoRTMxUSQPMLpBNIkBHAQQAQIABgUCU1B+vQAKCRAohbcD0zcc8dWwCACW
XXWDXIcAWRUw+j3ph8dr9u3SItljn3wBc7clpclKWPuLvTz7lGgzlVB0s8hH4xgkSA+zLzl6
u56mpUzskFl7f1I3Ac9GGpM40M5vmmR9hwlD1HdZtGfbD+wkjlqgitNLoRcGdRf/+U7x09Gh
SS7Bf339sunIX6sMgXSC4L32D3zDjF5icGdb0kj+3lCrRmp853dGyA3ff9yUiBkxcKNawpi7
Vz3D2ddUpOF3BP+8NKPg4P2+srKgkFbd4HidcISQCt3rY4vaTkEkLKg0nNA6U4r0YgOa7wIT
SsxFlntMMzaRg53QtK0+YkH0KuZR3GY8B7pi+tlgycyVR7mIFo7riQEcBBABAgAGBQJcSESc
AAoJENwpi/UwTWr2X/YH/0JLr/qBW7cDIx9admk5+vjPoUl6U6SGzCkIlfK24j90kU0oJxDn
FVwc9tcxGtxK8n6AEc5G0FQzjuXeYQ1SAHXquZ9CeGjidmsrRLVKXwOIcFZPBmfS9JBzdHa9
W1b99NWHOehWWnyIITVZ1KeBLbI7uoyXkvZgVp0REd37XWGgYEhT0JwAXnk4obH6djY3T/Hf
D70piuvFU7w84IRAqevUcaDppU/1QluDiOnViq6MAki85Z+uoM6ojUZtwmqXDSYIPzRHctfx
Vdv3HS423RUvcfpMUGG94r7tTOSXhHS9rcs6lzLnKl84J0xzI5bWS/Fw+5h40Gpd4HTR/kiE
Xu2JARwEEAEIAAYFAlaBV3QACgkQRm7hv+CThQqT0wf9Ge3sRxw+NIkLkKsHYBTktjYOyv49
48ja5s9awR0bzapKOMaluEgfwtKD8/NCgYeIVYyaZlYmS1FP51yAtuzdvZXAI0DAITyM4d1S
RCESjCCiZ028eIEcoeM/j+UXrwo4+I7/abFhiSakzsFZ/eQHnsMnkJOLf8kug3vMXjSoiz+n
T14++fBK2mCVtu1Sftc877X8R7xUfOKYAGibnY+RAi7E2JVTMtWfdtJaqt3l5y6ouTrLOM9d
3ZeEMdYL1PCmXrwZ4+u7oTNC26yLSbpL+weAReqH8jGsVlUmWWMXvkm+ixmrnN66WvSLqQ6K
P5jWnowV9+KEhNnWBOaT4Iu8rYkBIgQQAQIADAUCTndBLgUDABJ1AAAKCRCXELibyletfAnx
B/9t79Q72ap+hzawzKHAyk3j990FbB8uQDXYVdAM5Ay/Af0eyYSOd9SBgpexyFlGL4O4dd7U
/uXwbZpAu5uEGxB/16Mq9EVPO5YxCR0ir7oqi6XG/qh+QJy/d3XG07ZbudvnLFylUE+tF8YU
Z5sm9lrnwPKYI2DIa0BToA7Pi95q82Yjb4YgNCxjrr61gO9n4LHDN1i74cNX0easl9zp14zS
```

```
acGftJGOrPEk+ChNCGKFNq/qr9Hn/ank29D8fzg6BLoaOix8ZzZ25QPMI/+SF4xEp/O7IoI4
dA+0m4iPz76B+ke0RTsgNRfVKjdz2fQ92l4G9yWwNulGcI3FBZTiYGi3iQEiBBABAgAMBQJO
iLYZBQMAEnUAAAoJEJcQuJvKV618tkAH/2hGrH40L3xRAP/CXEJHK3O+L8y4+duBBQ8scRqn
XS28SLfdL8f/ENH+1wah9jhyMC+jmyRldd5ar3cC/s8AJRvOSDRfR5KvagvrDLrrF+i/vYDB
K5f6JQrryq0poupEuK0zTbLxo1FX+CAq+3tQy8aY6+znItpiWhvK8ZoULYKV+Q063YyVWdBk
KadgELA6S08aQTGK7bJkyJ9xgbFBykcpUUbn0p4XZwzZ3jFgzwcmqRIYZbfTosVVLJ5HAb7B
u22AukPlsz9PZvd8X8nfmtoJIwtl5qtFOrxrKA+X5czswzZ5H3jprDqOY6yA0EStu+8h1CPo
u50BmP7yKZxdXYqJASIEEAECAAwFAk6Z2dEFAwASdQAACgkQlxC4m8pXrXwC8ggAgQXVkn5H
LtY50oXmh5D/KdphSKDM33Z9b/3MHzK5CWeCQUkaJ1gxtyLW1HWyLOIhUkW6xHdmieoA8Yr9
JS1r1jopYuGZztzlScQeSWr8190xnZZVIjKReVy2rDSxtv7PV5wR3gby72PmKWUw7UHfqtBr
JgA+h5ctfx1jhXIUtUZpDTStZAFgVmunDXoBNZtYYk/ffY1J8KTjNmrqRcRbTurSy3dgGAAA
Z01DIR5kJrh3ikFFJfrXz0qODoYOchxqI4Xoc7o8uv19GUuvk5sKBT4b2ASF+JXAMRX0T7v8
Gralhn3CGGQGpZDN2ldM1Mzbi5oSETTUQ87nN4I7bXirqYkBIgQQAQIADAUCTqumAQUDABJ1
AAAKCRCXELibyletfMCHB/9/0733PXrdjkVlUjF7HKpdD8xy324oe5cRWdEVhsDj11AsPhLv
c37M3uCf2MV5BwGjjDypVRX3hT+1r9VsuR201ETKmU8zhdjxgTlZ931t/KDerU9sSJWOT33m
wEX7b5Oj31hgqy2Bc+qOUfSNR8TIOZ7E6P6GynxFzreS+QjHfpUFrg41FgV58YCEoMyKAvZg
CFzVSQa2QZO4uaUIbAhXqW+INkPdEl/nfvlUWdoe/t5d/BDELAT4HEbcJRGuN/GNrExOYw/I
AbauEOnmhNQS+oNg1uSjlTFg6atKO8XgXNfCp6sSVclSRTNKHSmntHEcH/WULEOzsPUXWGWA
VC40iQEiBBABAgAMBQJOvNkcBQMAEnUAAAoJEJcQuJvKV618xSkH/izTt1ERQsgGcDUPqqvd
8exAk1mpsC7IOW+AYYtbOjIQOz7UkwUWVpr4R4sijXfzoZTYNqaYMLbencgHv25CEl4PZnVN
xWDhwDrhJ8X8Idxrlyh5FKt0CK53NT9yAsa1cg/85oVqZeB0zECGWgsVtIc8JmTJvTSmFVrz
7F4hUOsrUcHJmw0hfL9JIrxTbpLY9VnajXh9a8psnUCBrw3oO5Zj8Pw/aaLdEBuK5mB/OSYo
vmJ0f/BIp+cUp1OAnOyx0JzWNkQZWTmsVhxY6skBEd4/+2ydv9TEoESw207t7c3Z7+stWcTK
RUg7TrqHPvFkr9U0FKnHeTeqPhc8rjUgfLaJASIEEAECAAwFAk7Oo7wFAwASdQAACgkQlxC4
m8pXrXza3Af/QjONcvE3jme8h8SMLvlr6L1lIuWpHyWwcvgakRJwUojRrSVPghUAhjZEob4w
CzZ4ebRR8q7AazmOW5Fn1GoqtzrWxjRdBX3/vOdj0NvXqCFfTgmOSc4qz98+Lzuu8qQH9DEl
ZLyptv96tGZb5w82NtHFMU9LkkjAVYcDXqJ4USm90CApXqd+8lVOrWuM8NycgD0Ik3ZKZQXH
1DHdJFzohNtqbWGMWdjqwKHoBSHEsjZ/WarXEf0+oTLjZSbrymtGpPInsijHWD9QMOR55RwC
DtPW+JPPu5elLdaurjPOjjI6lol8sNHekjmDZmRI0ZMyjprJITg4AG3yLU9zU+boCYkBIgQQ
AQIADAUCTvI8VgUDABJ1AAAKCRCXELibyletfNeIB/0Wtd7SWBw8z61g5YwuG/mBcmLZVQFo
vGnJFeb+QlybEicqrUYJ3fIPj8Usc27dlwLP+6SU8BtldYjQ7p7CrQtaxG2SWYmNaJ50f6Eb
JpO/3lWSWiNEgF3ycFonoz3yuWMwEdMXBa+NAVV/gUtElBmoeW+NwKSrYN30FYmkZe+v+Ckq
SYwlg0r9+19lFwKFvfk0jX1ZGk6GP27zTw49yopW9kFw/AUZXlwQHOYAL3gnslwPz5LwiTyJ
QkxAYYvdByZk4GjOi+HzqGPspNIQEeUteXzfbPz0fWEt64tudegYu/fN5QVLGS/WHfkuFkuo
gwNBFcu5TPEYcwGkuE/IZZEniQEiBBABAgAMBQJPBAkXBQMAEnUAAAoJEJcQuJvKV618AG8H
/0LLr1yM6osbLAhOzcKmD//jSOZ2FIBXGKUmK8/onlu5sUcMmVVPHnjUO/mHiFMbYFC655Di
rVUKlIZb6sdx2E/K+ZkZHPWvF1BAaHUO/QGh3Zzc8lVJg9KtFLAJkmQkc61VEF2MriaRlvlo
VPNr5Oiv2THOPgVxdV3goBL6EdAdgdwCvy23Z44vOp0QVNQt4aJKg2f49XO/N1+Gd2mEr7wX
aN9DZQq5zTU7uTRif3FlXHQ4bp8TWBK3Mu/sLlqZYtF3z7GH4w3QbwyA2CWkGgTGwQwyU8Fh
JQdrqXGl0w0y6JusjJWdwT1fxA6Eia3wrSw2f8R1u6V0k0ZhsMu3s7iJASIEEAECAAwFAk8V
1NwFAwASdQAACgkQlxC4m8pXrXzijAf7Bn+4ul7NedLGKB4fWyKDvZARcys13kNUcIl2KDdu
j4rliaY3vXT+bnP7rdcpQRal3r+SdqM5uByROHNZ+014rVJIVAY+ahhk/0RmdJTsv791JSkT
FuPzjYbkthqCsLIwa2XFHLBYSZuLvZMpL8k4rSMuI529XL48etlK7QNNVDtwmHUGY+xvPvPP
GOZwjmX7sHsrtEdkerjmcMughpvANpyPsFe8ErQCOrPhDIkZBSNcLur7zwj6m0+85eUTmcj8
1uIIk4wjp39tY3UrBisLzR9m4VrOd9AVw/JRoPDJFq6f4reQSOLbBd5yr7IyYtQSnTVMqxR4
4vnQcPqEcfTtb4kBIgQQAQIADAUCTzltCwUDABJ1AAAKCRCXELibyletfAo9CACWRtSxOvue
Sr6Fo6TSMqlodYRtEwQYysEjcXsT5EM7pX/zLgm2fTgRgNzwaBkwFqH6Y6B4g2rfLyNExhXm
NW1le/YxZgVRyMyRUEp6qGL+kYSOZR2Z23cOU+/dn58xMxGYChwj3zWJj+Cjw9U+D/6etHpw
UrbHGc5HxNpyKQkEV5J+SQ5GDW0POONi/UHlkgSSmmV6mXlqEkEGrtyliFN1jpiTRLPQnzAR
198tJo3GtG5YutGFbNlTun1sXN9v/s4dzbV0mcHvAq/lW+2AT6OJDD204pp/mFxKBFi4XqF6
74HbmBzlS7zyWjjT2ZnujFDqEMKfske/OHSuGZI34qJ3iQEiBBABAgAMBQJPSpCtBQMAEnUA
AAoJEJcQuJvKV618L1QH/ijaCAlgzQIvESk/QZTxQo6Hf7/ObUM3tB7iRjaIK0XWmUodBpOC
3kWWBEIVqJdxW/tbMbP8WebGidHWV4uX6R9GXDI8+egj8BY8LL807gKXkqeOxKax0NSk5vBn
gpix2KVlHtWIm7azB0AiCdcFTCuVElHsIrhMAqtN6idGBVKtXHxW3//z9xiPvcIuryhj8orS
IeJCtLCjji7KF2IUgCyyPJefr/YT7DTOC897E1I01E4dDymNur41NjobAogaxp6PdRNHBDum
y8pfPzLvF3OY4Cv+SEa/EHmCOTHTamKaN6Jry/rpofqtueiMkwCi81RLgQd0ee6W/iui8Lwp
/2KJASIEEAECAAwFAk9V2xoFAwASdQAACgkQlxC4m8pXrXy9UQgAsVc8HNwA7VKdBqsEvPJg
xVlm6Y+9JcqdQcA77qSMClc8n6oVF1RpI2yFnFUpj1mvJuW7iiX98tRO3QKWJIMjEPovgZcS
bhVhgKXiU87dtWwmcYhMsXBAYczbsSaNWhOIPwKHuQ+rYRevd0xGDOOl3P7pocZJR850tM9e
58O9bzdsRYZpFW5MkrD7Aity5GpD65xYmAkbBwTjN4eNlp0nHVdSbVf4Fsjve6JC6yzKOGFB
VU1TtAR2uPK6xxpn8ffzCNTA1vKXEM8Hgjyq4LWSdDTBIevuAqkz4T2eGJLXimhGpTXy7vz+
wnYxQ9edADrnfcgLbfz8s/wmCoH4GJAFNIkBIgQQAQIADAUCT2eDdwUDABJ1AAAKCRCXELib
yletfFBEB/9RmWSSkUmPWib2EhHPuBL6Xti9NopLOmj5MFzHcLtqoommKvpOUwr1xv0cZMej
ZenU3cW1AvvY287oJwmkFRFu9LJviLSGub9hxtQLhjd5qNaGRFLeJV8Y0Vtz+se2FWLPSvpj
mWFdfXppWQO/kIgVZoXcGJQrQWcetmLLgU9pxRcLASO/e5/wynFXmgSajxWzWHhMvehvJTOq
siYWsQxgT/XaWQTyJHkpYJoXx4XKXnocvc8+X3QkxAFfOHCwWhYI+7CN8znDqxYuX//PKfDG
2Un0JHP1za8rponwNG7c58Eo3WKIRw0TKeSwOc1cSufnFcrPenmlh2p70EvNRAINiQEiBBAB
AgAMBQJPeKdGBQMAEnUAAAoJEJcQuJvKV618YwoIAMn3uqSB4Ge1D61m0pIXJfOcC6BhCZvM
mV3xTp4ZJCdCQzjRV3rZRkt0DwyOVYpLzLgDgvbRwjXjOzm0ob1DvYHFA7DnGTGUsBLDX/xZ
5gRvDtkD6w8b/+r2/eQiSu7ey/riYwB6dm3GzKR7FEbIK6bEuPOUBwvV2tYkZRgTYqXq7NBL
uNv7c80GWhC/PqdvdhFn4KAvL0PjVIgr5+mdXyviKqG7uvguYBDtDUMX1qgZpi+fb7EsbJYf
EkBR63jGQw04unqT1EXWds17gj+yp4IHbkJmEJMS8d2NIZMPbIlHmN+haTA73DwNkbVD1ata
```

qSLiFIGXRyZy87fikLVIljOJASIEEAECAAwFAk+KdAUFAwASdQAACgkQlxC4m8pXrXwIUQgA
mnkFtxXv4kExFK+ShRwBYOglI/a6D3MbDkUHwn3Q8N58pYIqzlONrJ/ZO8zme2rkMT1IZpdu
WgjBrvgWhmWCqWExngC1j0Gv6jI8nlLzjjCkCZYwVzo2cQ8VodCRD5t0lilFU132XNqAk/br
U/dL5L1PZR4dV04kGBYir0xuziWdnNaydl9DguzPRo+p7jy2RTyHD6d+VvL33iojA06WT+74
j+Uls3PnMNj3WixxdNGXaNXWoGApjDAJfHIHeP1/JWlGX7tCeptNZwIgJUUv665ik/QeN2go
2qHMSC4BRBAs4H2aw9Nd9raEb7fZliDmnMjlXsYIerQo7q7kK2PdMYkBIgQQAQIADAUCT5xA
QQUDABJ1AAAKCRCXELibyletfOLsCADHzAnM10PtSWB0qasAr/9ioftqtKyxvfdd/jmxUcOl
RUDjngNd4GtmmL7MS6jTejkGEC5/fxzB9uRXqM3WYLY3QVl+nLi/tHEcotivu2vqv4NGfUvW
CJfnJvEKBjR8sDGTCxxZQoYoAFbGTP1v9t4Rdo7asy37sMFR2kA4/kU1FDxYtFYFwwZCJpNL
hhw0MCI2StI/wIwtA/7TiFCNqHHAKAGeSzKVyKrPdjn8yt7Js2dM6t2NUOwXQ563S4s6JZdR
lXUV9oYh1v+gFAuD57UHvinn6rdoXxgj3uoBmk9rWqJDNYgNfwtf1BcQXJnea+rMavGQWihx
eV40+BZPx9G6iQEiBBABAgAMBQJPrg39BQMAEnUAAAoJEJcQuJvKV618M4YIAIp9yNCVLGta
URSthhmmgE/sMT5h2Uga6a3mXq8GbGa3/k4SGqv51bC6iLILm2b0K8lu5m6nxqdZ8XNNMmY9
E+yYTjPsST7cI0xUzbAjKews63WlEUrj/lE2NEtvAjoS2gJB+ktxkn/9IHnqwrgOgUofbw6T
hymURI+egyoDdBp91IQD8Uuq9lX+I+C1PPu+NCQyCtcAhQzh+8p7eJeQATEZe2aB1cdUWgqY
evEnYNNK8zv/X3OMYl67YyEgofKoSYKTqEuPHIITmkAfn0qVsBA4/VtLbzGVGyQECmbbA34s
5lbMLrYeERF5DnSKcIa665srQ+pRCfJhz6VQXGsWlyWJASIEEAECAAwFAk+/2VUFAwASdQAA
CgkQlxC4m8pXrXwDOAf+JEUUKLiqO+iqOLV+LvI09lU4ww7YfXcqz4B9yNG0e5VprfS7nQ0P
tMf5dB7rJ6tNqkuHdoCb+w0/31pPEi7BFKXIoSgOz3f5dVKBGo8GBsX+/G/TKSiTenov0PEU
7/DlwvwmsGExmgmsSQgEWTA3y1aVxc9EVC9x0Fi/czcNNlSpj5Qec7Ee9LOyX4snRL1dx30L
lu9h9puZgm8bl5FLemPUv/LdrrLDqG9j4m2dACS3TlN14cwiBAf/NvxX3DEPOYTS6fwvKgLY
nHlOmKRCwlJ6PArpvdyjFUGWeCS7r4KoMCKY5tkvDof3FhggrQWgmzuPltBkTBQ7s4sGCNww
6okBIgQQAQIADAUCT9GlzwUDABJ1AAAKCRCXELibyletfDj1B/9N01u6faG1D5xFZquzM7Hw
EsSJb/Ho9XJRClmdX/Sq+ErOUlSMz2FA9wDQCw6OGq0I3oLLwpdsr9O8+b0P82TodbAPU+ib
OslUWTbLAYUi5NH6WW4pKnubObnKbTAmzlw+rvfUibfVFRBTyd2Muur1g5/kVUvw2qZw4BTg
Tx3rwFuZUJALkwyvT3TUUrArOdKF+nLtVg3bn8EBKPx2GfKcFhASupOg4kHoKd0mF1OVt9Hh
KKuoBhlmDdd6oaEHLK0QcTXHsUxZYViF022ycBWFgFtaoDMGzyUX0l0yFp/RVBT/jPXSBWtG
1ctH+LGsKL4/hwz985CSp3qnCpaRpe3qiQEiBBABAgAMBQJP43EgBQMAEnUAAAoJEJcQuJvK
V618UEEIALr7RNQkNw1qo7E4bUpWJjopiD00IvynA0r5Eo0r83VX5YYlAfuoMzBGg6ffKiCs
drHjEh45aIguu8crQ7p2tLUOOzKYiFFKbZdsT/yliYRu4n28eHdv8VMKGZIA7t0ONIp1YPd2
9pjyVKy4MOo91NfwXM5+tcIzbYL9g+DuhQbYDmy8TVv7KKyY/gqZU1YB6kS49lycQw8WCine
FoeD1fb6aP9u0MFivqn2QCAhjXueKC01M2O0jR0wu7jdojN50Jgeo6U0eIHTj2OQmznh8wYG
MX2o+1ybSTjjHIp3X8ldYx01Sa3AqwKEBclLdg5yIyAjHq2phROd2s/gjqrWt+uJASIEEAEC
AAwFAk/1PVUFAwASdQAACgkQlxC4m8pXrXwn3AgAjWUh31IxsQcXo8pdF7XniUSlqnmKYxT+
UZOP71lxeaV/yjY+gwyZvf8TWT4RlRp5IGg6aNLwLaDB3lcXBGuXAANGUr+kblewviHnCY3Z
+PWiuiusle+ofjbs8tFAr3LN3Abj70dME7GOhLyplP2mXIoAlnMDJ0AyrKx5EeA2jS8zCWCu
ziiOj4ZwUZAesXchpSO9V9Q86YiPtp+ikV0hmYgZpIXRNcHOpxnVyEW/95MFwi4gpG+VoN57
kWBXv6csfaco4BEIu9X/7y4OLbNuvzcinnHa0Pde5RnRlbEPQBBZyst2YZviWTFsbG8K2xok
dotdZDabvrRGMhRzBUwQEokBIgQQAQIADAUCUAZhawUDABJ1AAAKCRCXELibyletfDJUCAC+
68SXrK4aSeJY6W+4cS6xS//7YYIGDqpX4gSlW1tMIKCIWNhHkZqxKnWClnmvgGhw6VsZ2N0k
YdOnIrzEPWL7qplZRiE1GDY85dRXNw0SXaGGi7A8s6J9yZPAApTvpMS/cvlJO+IveFaBRHbI
RRndS3QgZVXq48RH2OlHep3o7c964WTB/41oZPJ7iOKgsDLdpjC1kJRfO9iY0s/3QrjL7nJq
5m14uY16rbqaIoL81C7iyc0UKU9sZGMcPV7H0oOIAy206A3hYSruytOtiC1PnfVZjh14ek2C
g+Uc+4B8LQf5Lpha4xuB9xvp1X5Gt3wiPrMzcH89yOaxhR8490+0iQEiBBABAgAMBQJQGC19
BQMAEnUAAAoJEJcQuJvKV618CbcIAJCXDbUt96B3xGYghOx+cUb+x8zcy9lyNV8QC2xjd9Mr
02LJTQHfJfQ9Td6LfuoRb7nQHOqJK1/lWE28t9tlH7I+i7ujYwA/fWardRzqCulNXrgFEiQK
ZFaDjRYyM0jWG/sA3/Rq2CMBNhBeCcTDuZ8VvRdm0xMPpyavP8D2dM9WBkPHOik4yAIILVkr
hWmr0Up0JhRoelfeyqcN/6ClUgeRMIyBYthA55fk2X5+CerommlpDfJJlFQOv64VSzS68NG8
j9yf66uuL3bB0OdzOMW6Yq/P9wskCDlMbYm/UnHfB5wAuxWpDeAvt/u+vU4xqqEjkUQGp03b
0v1xl79maSuJASIEEgEKAAwFAlWg3HIFgweGH4AACgkQSjPs1SbI/EsPUQf/Z6Htrj7wDWU8
vLYv3Fw23ZuJ8t8U/akSNwbq6UGgwqke+5MKC1fpk90ekzu5Q6N78XUII3Qg8HnfdTU0ihYg
qd3A1QmO6CG2hEz5xoxR1jJziRCbb1J7qEw8N/KzBcTkHB4+ag6bjFY9U4f9xU3TjPIu7F2V
Bk1AX+cmDo8yzPjDnP4ro0Yabbg0Q9xzvaK/7pFRz+vL/u/lxW7iE7n6vXTiaY1XnIt5xAXX
dwfLYmWeAgdc9KXFNlt4lfuqrETtNCHme+JI+B2Tz2gHmMVLHiDV59eLC0uU/uVsOXEd26ib
JC4f3KqY9kxuQm325kNzxnMxiwMPCVzsEh7lsYp+OokBMwQQAQgAHRYhBADTXowDFGilEoOK
6kPAyq+7WPawBQJasiYMAAoJEEPAyq+7WPawox0H/i96nkg1ID61ux+i20cOhVZylNJ770Vv
0zfXddWRN/67SuMVjLLiD/WfnDpw6ow6NM7vfEwbmvo1qeFF7rWWTPLm57uZfTk73un3fbaL
JiDZyrUStQKK/yhGAZmwulOQq7XBm+u8G9UcFi4XQxuoc5I/v/lUgbxXBADlxlfzpkIDwOaB
s23RDiMcWZGcosUkYHXlm8scU0tRANVLQ/PHgttlUl3x2PLzrdQm3YUDKUJ9+ynO2jN2sYwt
laSohj4UbLnq6pI4CXWZR7XWQs+NX7P3R359FDtw7OhyKoVuIkRFZljY0i3wQgwl/Sm2DAg9
3lsZDVc/avEUaOO+VuJuvJ+JATMEEAEIAB0WIQQGFx4znGT7HFjpuwT3iPLIbOWZfAUCXJ7Q
KwAKCRD3iPLIbOWZfGoXB/wN0P3m27fY/6UXTl0Ua3H+24ueUdLipsvR8ZTwEfnwkhLrbggE
0Em7ZuhZkzv7j856gv/tOekYYqWGg1CLalD3y371LAGq1tjY3k/g2RWLxLXNdzgXEyFvaNQA
oQa9aC2Q7FOyEMwVkkXrGa4MML7IBkrtMds9QPKtfipachPf6tQOFc12zHRjXMZi0eRWyQue
0sLLiJZPn7N8bBAJyZ9IJEpkhNrKS+9J5D1Refj++DwBKDh04kQXZFEZZhxcungQW5oMBQgr
uW2hULTLeiEV+C516OnwWJOz6XKJpOJp8PY0bO8pGgToGIYHkoX2x64yoROuZasFDv7sFGX6
7QxyiQEzBBABCAAdFiEEEN0MfMPATUAxIpzAoiiOmODCOrwFAlv/EJIACgkQoiiOmODCOrwg
uAf+IVXpOb2S3UQzWJLSQyWG0wQ51go4IBVpHv6hKUhDFj47YdUbYWO+cgGNBjC7FVz54PUM
PIdxImGHE1NHH+DNR8hvvAi+YpnqqdT3g+OgZ6XoYevret5B2b5fRgN1/HWUjaJ/n5g6SMsC
+3DrmdMu1FEDnKv/1HwQvOQXKt/U2rXE1ILOmVdMavRJEwkrk2SVwbdeass2EInZVsmWL+ot
9dU5hrkmLAl6iHUoK6zF6WaI1oi7UU2kgUF2DNyZG/5AumsNhxE608EAs1zEdN8wibXL48vq
Z4Ue9GvImokdlq/r/4BMUdF1qLEZHBkbaklK1zXxl7uMiW3ZIcqpg5HgwYkBMwQQAQgAHRYh

```
BBTHGHD/tHbAjAF4NhhrZPEl5/iCBQJZ+o/oAAoJEBhrZPEl5/iCyfMH/3YP3ND8jFqIWkmG
JaITHP9GhAQda73g7BFIrBHeL033tcLtUbEHXvnIZzulo7jiu9oQBjQvgGgIl5AqH1m7lHaD
iAL3VmuUFZ4wys7SODHvSZUW1aPLEdOoLKeiG9J6elu0d/xWZmj86IaHMHrUEm1itMoo0m+U
MwVNLFNZrAjCn82DiS6sS0A52tOlpq/jR4v9AYfMZSnd1MLm/CZaZpzWq6aqm7ef7CDfsUvU
w7VsL3p1s+Jgo6+8RwQ1W2Lgt5ORthvpjPKE1z0qgDpoXTkPOi8M20taD5UZbpByzMZPJXXr
+LBrRbs48IcPVHx8sxHMh1HsQCiXHDGiTNSaJ1qJATMEEAEIAB0WIQQazDqcUxAL9VrKN9zD
LyvJ+reoRgUCW4YZiAAKCRDDLyvJ+reoRptWCACoIgFrvhbr3c1WVq16LJ8UmQLk/6uFFZPN
CiR6ZbvzOd+a3gk1G8AhDEW2zoNhFg9+I7yqUBGqn+B1nDZ6psyu8d5EoRUFTm3PghqEccy5
KixqoPxBTquzkKGbN8PDLUY5KvpTOLLlYZxlHzSHw4roPsU4rxZtxyu98sSW0cm47VPr069p
91p9rCoHY8Fng7r3w28tVfvLuZ1SK4jtykIvw+M/pVBk9rQVCAJ0JjkAHkTOpkHqsVBYhtu7
mzsXfkQZkeuxdNx6X1fMrbJofzH0GYTT8Knn75Ljhr3hozrsL4Kz4J9gsLHCjkD5XKzLwCFK
R6UhhZZr7uhufbqZIyTLiQEzBBABCAAdFiEELLeCvUfxyJI8qMqHHSPVZ6Jn8NcFAltZjFMA
CgkQHSPVZ6Jn8NfKSggApk065wFrxq2uqkZKfJGw2mdsGeDVjGq9tMKUWeYVxTNxjiYly8Dc
/jrOS3AU6q7X7tAAcmvaXoBfW3xEIXMSH73GeinVG7wnlab6GKPDRKJzXfJ88rF07pX8R1pc
ZH+eikiFsN9bcnEycH82bonS7dzyoo6yg2zBqNtsmWYLDg2hcoTw4UHAPwdX6+n99m3VzOqO
8ThQI9hqpUYGvP5qyYahFf+39HSViof+Kq5KKhvSoiS9NzFzYZ0ZszYt+2jozUpAM6XqtEGu
TMzXHkE+/V4yI3hIsvHNkXKgDrqjwA+UmT1R4/gBoiRhZ8r4mn1gYI08darQmkppf9MEbcDz
U4kBMwQQAQgAHRYhBC1hIxvZohEBMIEUf5vAD7YffmHCBQJcns2XAAoJEJvAD7YffmHCC0UH
/R8c5xY96ntPI2u6hwn5i0BGD/2IdO+VdnBUnyE4k9t2fXKDRtq6LAR2PAD0OehSe4qiR6hw
ldaC8yiyg+zgpZusbCLGxbsBdYEqMwTIeFsa8DyPMANpJ0XLkGGf8oC7+6RuAJvlm6DRlurr
U93/QIG6M2SNsmnPgSZWYV4Y5/G7Xxyj0Fc3gNjjjGGP61CBR01W6rgNPn35sZ9GYCZcGlQA
GGrT8mSVoUhPgPCXKz2dZDzsmDHn7rULB6bXcsHiC/nW/wFBpoVOIFIxND0rb1SYyJzPdPtO
K6S+o+ancZct8ed/4fUJPBGqrBsuFS1SKzvJfPXjHGtZBitqOE7h57SJATMEEAEIAB0WIQQt
9h/1MHY0zPQ0K+NHN096zf0O3AUCXK2H5QAKCRBHN096zf0O3OJtB/wKbQN4IjVNkmWxSaBc
JABRu/WSbNjoTo/auJV6IRUBpwR130izMw239w5suuWx1phjPq3PdglBaKKeQNdeRoiudUjd
hydON1cq2wh9O073wU2GHeZLi48MopUNksrhHfd/XWV//0LcSpERsqIBVIUi+8DHwFvpCzCz
zIRg9lOcQmEtJAFFUtkF9FEeZgO2NPO3fEwkjKDeJYUiB+mD9BliyxhU8apUx/c2zaFGQOCr
MllN/gHztAWDcIadK/tujqRWR4wnJ0+ny/HP+bWd18+YjhcWzUQ8FytG+DA3oylQ1d0w0emt
qfn0zqiFkJQdG0M4qtItJYEYHlYpG2yoQHcCiQEzBBABCAAdFiEERVx3frY8YaOOhcAGjZrN
vi2vIgUFAlnScGAACgkQjZrNvi2vIgW5IQf8DKjeoHF9ChDcb4T01uJJiAUu6lxewSRD7iwD
6MjCsaxgMifTD7Bzvdem4finoOul2YAPtlLfIfVtVRtGG97R/Wvs3yjI9NSzxkDGuuE7/IIi
4dKlcKkvijg7G6A8+MGXaQTw8iOePI/44IyG5yogKjno7L4h0f3WguGzmCRUJcgYm23IsaTh
Pvdq39ARyHAlrk0hXZ+OqsYBrlW7KLyPrbPA3N+/2RkMz6m+T8ZksOrEdF/90nC9Rky4Wbg4
SJqWQNNSMfgT0rQL2Qvne598FKmltrTJuwBtIrSeuL/dbKt+hkLgnRjnmtA5yPaf0gXvMtfU
P9goQMWD+A2BU/bXJokBMwQQAQgAHRYhBFBgHh7ZZZpG0pg7f1ToXvZveJ/LBQJblegpAAoJ
EFToXvZveJ/LS0YH/jpcVprmEGnqlC0mYG2MlRqeK4T8Y6UnHE2zBPc125P4QcQfhgUJ98m4
0B5UkzljreFr9Zebk3pE8r4NBsamlJvi8sGbZONTsX4D3oW9ks0eicKOcTZJgtX5RmSNFh63
+EHbqTneK/NTQIuqRSCOufqCOH6QY1PVsICBlFZUPMfuxRlO7EwHKNIHPVBZNlM7AXxdjCMU
kXvda8V14kActb1w7NWxWxo5q4hkQ2K3FsmbWXvz+YBhJ8FnRjdzWNUoWveggOD6u4H7GuOg
kCyXn1fVnbCyJWsXQT9polJRnIAJMAtykcYVLNS/IS65U+K1cMshcF+Gil9BuGyckbRuNaSJ
ATMEEAEIAB0WIQRh2+o6RdTFb7cSlWG3d+zE2Q5m7gUCWdJutAAKCRC3d+zE2Q5m7rgJB/9k
c+prmrnjsq/Lt6d90LqYoavvIeFkAoDhhWgQeEOAD1wgyHIpS6qoMKgvBlvda2r0bmk1kUL2
xQaiDj36wB5yJHauOnFX+3ZJ6QCYUaeoWtqO2ROHvTiuyUdVKC5NtKaHpM1/lP/jl/1ZRWay
idggH7EnwDMt+9O0xD02n5J29Vp9uPO1GtMVsVSiJCGcOxwNBgNiXX1BpZbN4bRm5F8DAGiN
v4ZI69QZFWbpj8wFVJ/rV4ouvCFPlutVEAuIlKpAj35joXDFJhMvPpnPj84iocGqYPZHKR6j
a90+o8dZw3hXObFowjcxsJuQUTVkPuhzqr6kEu1ampaQ8OGpXCZHiQEzBBABCAAdFiEEZ/mR
TQQxCZjglXUwgzhtKKq2evsFAltbmWkACgkQgzhtKKq2evsdrAgAubfuG1vWX3TTG/VYYrfM
1aS1Roc034ePoJHK5rLT0O/TnnnObw38kJM1juyu4Ebfou+ZAlspiWgHad62R1B29Kys/6uC
qG2Jvbf716da4oLXeLYd9eb+IKVEiSb2yfbsLtLLB0c/kBdcHUp6A1zz0HV8l1HWj1Wx8cFU
MV7aAQoOfnNBbnNWLzNXXLYGHh47/QmjifE5V8r6UJZGsyv/1hP4JHsQ2nqcM8Vfj+K+HEuu
nnxzgWAcQXP/0IhIllVwoWhsJlHW+4kwW02DDopdBfLTzCtzcdOkfBcCg8hsmC4Jpxww5eHm
saY6sIB32keCpikVOGwdGDbRH7+da8knzokBMwQQAQgAHRYhBG4VA/IlW5kLV/VchhLcHkBr
mersBQJaX4N4AAoJEBLcHkBrmersksUH/3M0cypXBnyGIl/yE576MDa0G1xJvciup0ELeyhj
48Y7IAr7XiqDtiPt8tlIiPFF8iaw56vJw5H6UKraOcjZHOH1SwDr5gAWJgMqnqlFX/DxVKif
USt81KX0tHN6t6oMESgm2jRKvcWjh6PvEZlIArxZG4IjrErqWIJjUJR86xzkLyhRVTkUL/Yk
uNl1i013AlaD/0CGuAnjrluUUXypadtNr7/qsBx8dG6B/VMLWToEDEon76b8BzL/Cqr0eRyg
Qz6KWi3hmsK+mE4+2VoDGwuHquM90R0uS9Z+7LUws24mX5QE7fz+AT9F5pthJQzN9BTVgvGc
kpI2sz3PNvzBL5WJATMEEAEIAB0WIQR00X0/mB27LBoNhwQL60sMns+mzQUCWoyYfgAKCRAL
60sMns+mzYgnB/9y+G1B/9tGDC+9pitnVtCL2yCHGpGAg+TKhQsabXzzQfyykTgzCHhvqRQc
XHz5NSgR0Io+kbGMUUqCaen6OlcORVxYIuivZekJOAG+9kiqWRbyTv4aR6zvh8O5wCyEhhyi
ifi65PM7y9lD6i22qTt/JoDnFkP5Ri6Af/fZ9iaIaluQKJCU5xY1Lt/BorGlrGvX5KiZD8xc
AjhJRATZ0CJ21gbxISSxELAfH42KzGAvJw/0hARrMkl/eK0HVDpD47mcmC5h/O/HlwPYi0hn
xB+6/nuwwtRgMDBufNV0StU43njxCYmGI9/I1z5Vs+zhz8ypw/xCr1U7aAPZQdSSsfEViQEz
BBABCAAdFiEEelR8OpStCJs7bhrK1TniJxBsvzsFAlv+8d0ACgkQ1TniJxBsvzsiFwf/a3lt
OuSrFs4M03YVp6LoCM6CwZfvcFl+6B0TAurOiCja9lsNmbusSx0ad7bZy6/kHDXH/eqomXeu
O4hkxxBvGK3gZt7iQsr9vsUSbbJnc1zMyOZKlhdxAOLOskttqtPs6hiJ9kUHFGZe47V3c77G
GMgi/akIU5PkxhK7+/bbAsW0iK60aXCZ5nAbWlzTQLgJnYrlk4b920rzGe8nDTGzGmSjIGnb
YvuD9ZI40DZRWVf1tXqCY643AXFYoOhRxj54uHnMLYhc0I65u2ZGwRiTI0g/en5E8i7WoejA
/sR0+cYs7l1IJwlNRwfqmnJWRGREEHcJ3N52k3X7ayq3qmr3K4kBMwQQAQgAHRYhBJSRYHFB
cqf4Tl2vzE+YN4Ly8sn+BQJae/KHAAoJEE+YN4Ly8sn+5ckH/juc2h7bC4OGmRHcZBLAG2vW
WEMTc8dAr9ZyJYXzR25W1/Cz/JXgJgMjSrE6m9ptycpvWc6IRlrQM/IqG+ywYFPwNp3PYsc0
1N33yC15W7DPRDTtJE+9yUbSY9FeYraV4ghxiBxD1cDwtd7DFNGNRvBDH7yQHmXBW0K8x6yX
```

```
Mwl1gj2/MvdFUKmz8Lku94OmrbDOi83cnAjUNbN15Wle7hWAIRALt3P1VusjV/XyzxvcSffb
mt3CgBCyK9CNyEr27CVkhZ8pcabITx9afMd1UTEii90+qzgcJwcR46bJPZBdavMt56kVCeC0
kG44O3OOk+OahKXzw4YspZMO046gYRKJATMEEAEIAB0WIQSm5fcyEkLUw6FcN0ZJlMJhNZ28
bgUCXTJMCQAKCRBJlMJhNZ28bsgCB/96PlBUdsKgnh/RpmPB+piFQf6Og+97L4fxHuQbzKOe
UNCSWNF7saVa5VaPxbV/9jDCTPZI5vBtnJebXtkmLoWFSZaXCYb49SijfvRsRAeX5QSqIRd4
3KMuO7nAvbPVYtMChCO/g1T3riF2icC6pgvmNZWm5Nu4pkLzRmQv8U33BAkL7EYIjZZaC/9h
o4Sh4l/gLNItOxMdsD34sJwBLvEi1pQOa1xNJ4kfQSRD/8ufakE5wfSie/s04w/2Cp7RD9H0
VlD+7FwPO1HQ3XJjONvOzj6uVdwCC5fcmbXbb2bbJ/xe4YVL3xmwWz5m2w+kBSpaZ6VHNocB
8S2OmIIPpr7OiQEzBBABCAAdFiEEp6WxZJrn5Z0o967I/htVRVZtQSYFAlqnkGEACgkQ/htV
RVZtQSYV2Af9E7FLIUi8lqOyYyZuX6skkNf5rNSew+7i5NsiNpQzZMdscJh9eJzyLrePLp7q
9HUOhMF/Fc0SgbDtKSWbfSidXkeaQ2twPj4rP1xxYBc0OY0OX4fNVA5O/pTI9nxIVQCDTljl
/WIY+fnj88lCkaKWoRJITaotjFmYt+gbJMBn3MMYf0VODeIRozV7//NdkzFXKmJ3fsCDGXXF
CVWM1Fn3M91o1fh3FSgKd+0sexUDn5afwWCqjGgiXDsE7fEdwsbnz1rDzWvuqCoZyIh1RXQf
QVbiakpzfvtDytC3Vo6F2KzpZ9d69Adhfn2ydAYxL/Xuvk9pWdEBNF4T+HfS9Z30BokBMwQQ
AQgAHRYhBPJCF6TG7RrucA13q1lkfneVsjZHBQJawgLrAAoJEFlkfneVsjZHgNsIAIaSJ3gF
tBtf0WLxYIo5zhNclXOnfgUUNjGrXHm5NxoI4Eulpx9dQYCJ++whMFbxpZQTgFAUq8q342EZ
raLCWwALZEZmkZjv+FX6bk8sgqZESpUOLJAIqpobKpaawOQ7LS+XWO0SchH1oLFAgDyBeIDZ
N/LiTlIdkJe1xpDQDtgUHawksqMCbIaBe60B5xvm1Nkhnrmn M1p+e3LUd4j+XxACdcY5LSqV
zVT4OyD1WkKzk8EAASUI8xysNBEeX9/8/EXaAciECQb3MkYxTQZ4WqCLU0GCGl6Sx2fY5zI6
4Y1j/Sfn3JHikJots8eR1D/UxrXOuG5n9VUY/4tTa0UGPuCJAU4EEAEIADgWIQRLXddYAQl0
69GnwU+qS4a3H5yDGgUCX6xjgBoUgAAAAAANAARyZW1AZ251cGcub3JnYW5uaQAKCRCqS4a3
H5yDGkRfB/9z/5MuAWLwoRLJtnJQzEOW7jsfzYpepL3ocT9tdGcs8jJTH3vh2x4Kp2d0Zaxx
Zs7R8ehZO5XJQ/DWdhH+7cifoeXmAEqDnlKSXZQZY/bG054tM6zes3tFTH3dCrn7LF59fQOG
OaZHgbFRQJO6F++90Mj9WAgeqGxyEhAlFIxFw4Cuul8OZAUIfq7YISnpkg2Tm/Q0SRRDJE4i
/7WJE/HVMB0Rf9KJXuk2BJlRIpQz8Cf+GVZ5aGIlXdM58QknprnollxoTKhrE74rAGHW7nRD
xIxOoP8odiXbLzn//g2m123usqncCKWZONDdVupax3RQ7xsIuFc9Kx4OtjwPQftziQFOBBAB
CAA4FiEE6hBKAqPbygqOC7fUwpbDMFwG9MsFAl8u+m8aFIAAAAAADQAEcmVtQGdudXBnLm9y
Z2FubmkACgkQwpbDMFwG9MsIvggAhRfd2Z5WLR6hGxOHu+A+ysjX6xKjcqshCYr8jRuOflFN
vxugQQoFM5pQr15TyhokaU78aDUoIbLnKcxxmH1l4hXxcRtg/9Y22TidOVN4jjNbc69KvCC4
uANYuAJaI3o5fb1jv8Lx82OiRDMhtRqyTdSGdU5//8X5FXCt+HhhzpSNoNtpxyhsKP0PAWao
zuETqvxy7t0uy0f1OTbZLI5nb52DxjBdZlThnJ2L9RwR2nSGhxjhTFg8LrZWgWNtY5HG+vk9
qbCwaC6ovNJ0G98i0DMrlbyGCbxa4Rv332n1xPfl/EPYWmNPlMu0V3bSCqxVa5u3etA5fw3r
qIm333vgFIkBswQQAQoAHRYhBJTatFFgHAZYHkTw9GcRGDP/RljgBQJa7LubAAoJEGcRGDP/
RljgNu8L/jN8j4HSggpnzJ0+3dFjVg7FUHJF6BZ84tv9huhmyrByaIrEfFf9ARn8OizKgdpC
/wJT1+KXarvsxdnEDlYSat3HS/sEw3BmZjAeTwPi0ShloiSjYgYRbg3irDskqUHML4hhvMx0
x9nZIag2XoSSH7kPEd5jOb8cd7jJeoGg6Z9Z9lMHuyqTGi0T/EbnhjQfVTxWkSkcDvdxbSuW
D96mvZrbRnrMebXKkISb0uVUn3/o11iUo9jXs+Q/03Tb9i0H3eOliP1kcB/kggu9xblIPM+J
VaK5Z+zAVLPKTQJi+sP/ayEux0xZzfbZ96WERnzT4E7Wwv8MvaLbybtID28Oy9YoBBYv7CrC
tyfrHh1t4v2AedRSZcTPKAaQ5NtLAvIdex0kOvvofaGi+7nmgV00vCZFBSXetvBMZkCapW09
vF7wcahaXpF+0Spl9vE2JiesST7uQobCUm1EjxJP0vMDcO1vIfJHlbIhB/f3PE3rXZIzYTdL
s3Kb4OONaUfNy9jYtYkCHAQQAQIABgUCVJqcUgAKCRB3MepTnaVyot2+D/9wAQ+p03lVMpYS
gMWMNLgjq3z7QrN0NYNpxUXAonxECjUzZKSUPGci+fPKxl3ZUenk+ruLgtgJmjmUOR6u1Dov
BpDFzhfqbIpjgtMDrnY5sWqxJ+CH2Rb5okEEDJ5qE9DwIMP5iXbf4xjnBOyPiq3sp983PLvy
8ttidWe9FDf8JuhWLHRJHODQjc6LufcHSWKG9fLmCjL2KSPNl696MwR+N95EKCivLL2PlG8c
f08Xd8lW1S0cJLh/6TEuZtAnVeo0NUOGUXOPPyhTPP/xhfLeKbkxjtm6rg/jBaIjuuQgUyNN
hKnP96/GRWWRHvio6eBPalhUcvImSrCHnqLRpdyMxmK67ZzKZS3YsH0ixozJYE0mNevZ2hEY
wB+O5HllqK22YwvJnCLH2ZZWTu2TCUjGZP8hbo2nSoyENlxZio9Gl/v4ypjdlgwrjnnZvxoM
yOFeuc47AuzP5QjhtlrWv12C4hYi3YLZvkLVFD0CxAE/CDuHk/4eFG4UC4Mor6+BXwVG7NEl
4qQWrAHjLQ2/sHMpsUqY/5X7+StG/78PLP0HP+PIBCDDTa7W0+6kf0EaGVHKW43IIkVNI2Ps
b44tTT+Xhc2mHk44LuzL4Axlywv+CxP9NcKLNFwK4Ck1M8Np6cAKlu+Dw6gjOY1aGHgtdsBQ
cIqZj/+ETD0+9NkDXEoeDIkCHAQSAQIABgUCUliwpAAKCRCiKuTrQynFRXZdD/9vb+69OGSR
t456C6wMLgBl+Ocv9XeaCTiJjLgAL2G6bRH2g2VcNHnU/VMTD2YLVu0eP7ubsirVrmR7nAgL
sQ1mKKWvTI+p5aAvn4sL3x3P8vzmGoDAigZ458yGuVpVsBkSPjJBMAkMDfm9kdWxCanzuKXS
b59lfTg4EtcHPDzoSgABntASgfioVxP2TVPfre282cibeYS+RDlaMTVH25yElrWDuF2U1CVW
SMWY9mskr1+XjPnoO2jz0+jhKB7jyMMfSmJqzgcBNgezFbzX2fPmNnMZzEucVFFHmIhNVmL2
rOwc/s1tSHerG5YIdL3HOJek5xJljzjzFfDrdjmMMl+nO6nO78oePoLNdglQQSqn0yW6gZv8
EIIQ/N1nSi/LEW60z8FFxzoO8TqxMMX9QRLbVE6p+7C0nqolhZf6UEiDIIm+PihF1vPFSV54
+7OoLObCshe2g4pbRGWPhIJ4X3ILBQwFMZbn+cIuY3h3B/UpbZE/YSDgRFu5TLtCfBE/lQKX
7QhJknJhQhJ+Dx+Y8h1Cx61Qr0KP5DmOkHYZfAQtdacgrqEr/qNen4QYRdKp0gTne8AV7svB
8eI/8PkzvUPaHrax0g6ZSbeWbvEw6czm0qUGJX7iMlJSauIJPrbOjvXT7qIsaqZRRiUSWXo+
m+jzK5qdeRhEIUmlJI/tU/RsGokCMwQQAQgAHRYhBEW+vuyVCr0Fzw71w1CgTQw7ZRfyBQJd
hy3eAAoJEFCgTQw7ZRfyRf4P/3Igs5dYm0fhposI5iwBGtN5SsxYTZGte2cZ+dXVcnLwLIZc
Ry1nDu/SFXPUS0lQBj7/Bc2kl8934+pUtte+B5KZI2s/28Gn98C2IjxxU+YZ1X1LbUkx0cPA
jFWjUh/JSfu6Hif2J0NAG3meySnlmpxl6oZeTojeWo1t39PF4N/ay7S2TqIjGSBfxvD1peIU
bnziKsyM5ULbkMdgHssQvyZvrVzQxacRzPK424jXtKR6B2oA0wqMcP4c69UmVKEKIzJNYrn4
Kjs+An8vZvJYAVbiWEyEseTTo3XJePdBNs1xxK2vWLA5PeLkE8bmzHr8iQ3hA0NaY7jSJp3e
GrhWIdXV+nfclrFUPghYr5z+ljCSK5sow+aRiED39qd1Y+0iUAy94cqY3MQ4ayGgnB/+YuSx
B5jNjCBYJetFWWSJXnkbiYRLjU88dflXCrTbhkSuCu3agOjsBJYUyg/c1Z4eCQgpTWB2cjYQ
0ucKOsWt8U6qsl12qwYLr0RfcP2aCwTTnWIxqIN9F6iMafOsG+za8JY+B8PDJxxwWWz8vCvX
ChTYrfiFei8oUqoHYTbw07cxaxkDd2CgXsQMmOcZSoXZZPAe8AhsUibDl+BZs/vLZT7HrXtt
/ggz8LzVCcyQqwmCHurvgjauwjk6IcyZ5CzHFUTYWUjvFqYfAoN15xUZbvPYiQIzBBABCAAd
FiEERsRGITzmkUU5TZu635zONxKwpCkFAlxFLcAACgkQ35zONxKwpClKVw/+PfrtIVHFsOdl
```

```
2crWBSo5Hifvx9Vn2nPiNKErygB+tPWDS4UwzVUnpZfXCM7bKJFFPeKbitYxN3BlDmVhZMkc
1DZMAtIPSstO2oX7Tv/C0WOZPlAWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm
Wfl1LMvgwnFsQ74xobnCpssLgmogXfoLFQNF/VUfRveJ2Ci8raWyAdXFBdAIrejawAx5MMhO
/lEfQ3W3f9bqtJZ5DzLbxQ3Xtqs+RY1ihv1y12lr9vLpgKKGmZ92KDvjv2UXHd7XZ90aPMj7
Rx0MQ1d+5d/tNQ8rLJGuj1I7NqHmLHMz67TvRtPl4aNP7Mss8OHiEKLYq23kGqXN+6cjG3UM
i290uJZaAnTno65Cgsyn7JFKyXDdTOmp3TSoyVsPFq92qgd/jFBf3dJj8c+mZEVXkUFeeUEK
31EMGFCH+oE8un7nu+XWqFyFSw5wn+PGYDXkSd6z/NyIN5DXa326KV+qpUmIWOlcymm7cmZ4
KJQt7zgWCxh2DuWQzRlTjeQd8Iw62V8tIOBokWP9Thes18Qk2GOUeCnvczLdevT4lqr8IzvV
nSwX/LQyxmmz2/dmPhzJ6kA6KQKGOSF6WnV/WuD4kESFKwtABFi6mYQi1F6CynpVw/nu535C
4fFG4d+A5G6sKJx//hjOCgmJAjMEEAEIAB0WIQRGxEYhPOaRRTlNm7rfnM43ErCkKQUCXa6e
YgAKCRDfnM43ErCkKfNXD/0cTEjvQlgyy3UI3xfhYtRng8fsRXcACjMajnrvYCoRceWwF6D+
Ekvh5hNQqrZsxrD6nozY+iJhkkaQitIj4qw7i4KY03fo613FjeLFXWqf4sfLTANSsRNxawEo
/JxP1JeOToOgYTkikWOkgZWSs/mqvHAxJZrVq/Zhz06OugfOYVGmGZonU7zP12toiwParIZ9
hcZ/byxfNoXEtsQyUHO1Tu8Fdypmk0zYUgZK2kGwXslfOGj5m0M5nfUuVWq5C5mWtOI6ZngT
LPJ32tRW526KIXXZMTc0PzrQqQvTFHEWRLdc3MAOI1gumHzSE9fgIBjvzBUvs665ChAVE7p2
BU6nx1tC4DojuwXWECVMlqLOHKjC5xvmil12QhseV7Da341I0k5TcLRcomkbkv8IhcCI5gO8
1gUq1YwZAMflienJt4zRPVSPyYKa4sfPuIzlPYxXB01lGEpuE5UKJ94ld+BJu04alQJ6jKz2
DUdH/Vg/1L7YJNALV2cHKsis2z9JBaRg/AsFGN139XqoOatJ8yDs+FtSy1t12u1waT33TqJ0
nHZ8nuAfyUmpdG74RC0twbv94EvCebmqVg2lJIxcxaRdU0ZiSDZJNbXjcgVA4gvIRCYbadl9
OTHPTKUYrOZ2hN1LUKVoLmWkpsO4J2D1T5wXgcSH5DfdToMd88RGhkhH7YkCMwQQAQgAHRYh
BH+P4y2Z05oUXOVHZQXCWLGt3v4UBQJhrDYPAAoJEAXCWLGt3v4Uh2oQAMS3sK0MEnTPE+gu
7lLi9rMbD/3O5nlAxBJLX4MzLi2xP1648YV5nq9WMMt6qyp+OVwDXefneYNMgfU2/uu/Wi/o
XTHBJuU36lmFzhRWPj2h/vtfgDIYG2wio0DNJyaUQwLEi6gqPm0AHhKS4td69R+7qyQsbUIa
BFgoytxFzxDb5o2hicEOXa573m4myfAdCx5ucYfq+jlXJW9Wgw7ERnF1v9xQDXiuryXWFRdv
UOOWzVPu9T0gPkcG8NABwqxs28Oc7n9Al9HM2FtDAkD0LIcm/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//WSUO22Csa60I6VN8yJQmf0wCo9sieWDXCdHZ+CB0+gu0I3EMYR2a
gL8lqCd6M79fpP8DiLKOJvn9mhXCsjYjTJQUsuNi5kQ/O9gwarRsr7EjJ7R8u8lpSh9YPlMS
yN6XXfOa4Qy5HOw9idJdb3owKAXSjuRdi/hUExjA8TWliyWrfwiVDQi/aCoLZ4b9p6SfGR3Y
gE8UIZLZtdWgsPJHkvdvntTPi4fwMsadBfa2f+m4Wq2CAU5KSfYsVpKAwSQ1OsdUZUK7g+Ui
jy//ad7eZ+BAc75blHs7ua2iiF8Sc7MC55ZM5ldkv+0lqJ7td5vOCT1LKJg5PKKUC7YTTh9U
PHlERJ/SWcHNES1YhwLvUO2VROlPN9H1QkPnEMBOObpmYkNQyLBfFwioJ3ilptYY0IUX5qBM
5UkwgyqMsdyrL+2ozIYc+/A8KUnZXozOAG9LP8gBE5jBJSIkbqsi9Fumf7Q63++g4ojcYpOZ
F92X6kQMGqBvkvs8UajR5f/n6QH0je4XFPj4l4lVM/PPfZSShNGdOOi4l+KwozICnQ1+fhwh
N0VG4eALSJ6XQEEfJ18PrBRS3sdC7OVEMLevEC8ojSQeZE1lCLe1qAUoEcmgmXjsODaJn2tt
qNYYUxcFOycFnzgWL679C9FVp+DAg9jzDMKsqWo/Lt3IDNF19ZUc93WJAjMEEAEKAB0WIQSC
piWCWP+fBOH/9bx9bbut3FAu7gUCW8ygHQAKCRB9bbut3FAu7mOaD/9QJ1MiyKvw9rYqTvkU
OSDSLu88g6NP5R9ozgGZegInZ/NzT8u5emYccflnLlfvRQZPnT7YIH4+h25CCGQ5HzXUGENx
ndeuG4dm3B10A8hxv+abEM9VYDGqSIvF6z1xObvENOpMgmlmFdDi9O9d6jFFy4Hd6/BWejbU
4M3kfuD39RxaT1OEWfqvTVf4GKiLqM71glNB8WrTqxt2t/Mo2h6UPCF7/wPF/idMAbKEn0ye
b1WDCaZVXxAQETfNo129hPb2qxPGoCWGw24ySpGrM5We4Nd3bbdGItSZ0mATNM1+m9FY9j30
vpePFzzYGZ+23EcpxWU+7jWbjZ42ssCW6kx2/ERLVma7FuneEAqUc3gZr/3ZdZOVMvseg8c0
n66D/NRLgMcpOQK62qJfSrxQj6sJCGRY4dxAfdTZWrcxu8UvvcINezGITOQ0y+Mc5LM1vMOd
srXcaVnuJTfWorOeqnFecnClcOwKNAKBXjE8bSANUBKlrw0RIpye/IilrKGEMaYkP2nnnNZE
GPmumGkejDstWGmnHi5IogN8ibzyywsbNsO+qDdlUFA2bmVhh2uK7M95kyuMH3GnWbz4IiMx
RyUVEyK8yKnEmgOmLG4WiJjksP1jIPf3ztTEVVDJxy1gT3R36lsxd+OabnPOgiz1oFewKaur
aWX1e0E6eBWJ95ufookCMwQQAQoAHRYhBM8z5mfkMwAXdpGlbLdWs0L0i1qEBQJcBMl7AAoJ
ELdWs0L0i1qEmxwP/jDweTwTh1s+7Pp39L6aLB7nuQzdMleTksPGgmtguRBZipbOYOryEozK
9hI3Hq/ymV/loINv6GZhieDoZvxrv9eEKgO2eUE0IletSy7znlhV6MB7PBOc29dbCMf5L4qo
xUG/f+XfHkRZEkjZRWMlitlERlDU5gHAQ3skLuT9bu3aZkGdBgw0U5qjVvGzYxp2LFpNHXlf
TrlN3RZoDbRI+E9BPILqZFIZczp/fxRRNkXyogkrGD+0PANFsjySQKd/rr8/Z4isl3AM8CZ7
s4tMWM4EVJ2OygnrcMuIEJdXVsR0Ln1gJLuQ9HpWehve0d7/cIZkN7a0fqgE7bMvSPyxWL3m
yTA4FwdbrebBr2y7ixlXZ6WtX/rqTvo2HTDFLle0ZwMbbfAtoFX0M0lPtXTLmJAl5w1G8Nj8
bthWdN4KVFyOpqPt7OXc/G1YNLzcyYQXX5e8Uskmg40OH5cQV5OFEG8qpxTg53wANDdxXGzs
NUQe84Qkoyk75nwzVfsi00/OhTZmfIC48esXcs0kTrkSPrFcHktSMoYPmHfV3dTF17ifjz5a
C2SL22R+RokWuzGxxpvEaQAWIyCt6izf1a+CjnXPD2Jw3yDC/Oeg68XYiSrbeFdCRzQbS9YP
ipUFIlHuCiNZeGg3rFL2N2JodXg2LGORJz1RKazT7uAfRr5z7W1FtDtNeVNRTCBQYWNrYWdl
IHNpZ25pbmcga2V5ICh3d3cubXlzcWwuY29tKSA8YnVpbGRAbXlzcWwuY29tPohGBBARAgAG
BQI/rOOvAAoJEK/FI0h4g3QP9pYAoNtSISDDAAU2HafyAYlLD/yUC4hKAJ0czMsBLbo0M/xP
aJ6Ox9Q5Hmw2uIhGBBARAgAGBQI/tEN3AAoJEIWWr6swc05mxsMAnRag9X61Ygu1kbfBiqDk
u4czTd9pAJ4q5W8KZ0+2ujTrEPN55NdWtnXj4YhGBBARAgAGBQJDW7PqAAoJEIvYLm8wuUtc
f3QAnRCyqF0CpMCTdIGc7bDO5I7CIMhTAJ0UTGx0O1d/VwvdDiKWj45N2tNbYIhGBBARAgAG
BQJEgG8nAAoJEAssGHlMQ+b1g3AAn0LFZP1xoiExchVUNyEf91re86gTAKDYbKP3F/FVH7Ng
c8T77xkt8vuUPYhGBBARAgAGBQJFMJ7XAAoJEDiOJeizQZWJMhYAmwXMOYCIotEUwybHTYri
Q3LvzT6hAJ4kqvYk2i44BR2W2os1FPGq7FQgeYhGBBARAgAGBQJFoaNrAAoJELvbtoQbsCq+
m48An2u2Sujvl5k9PEsrIOAxKGZyuC/VAKC1oB7mIN+cG2WMfmVE4ffHYhlP5ohGBBMRAgAG
BQJE8TMmAAoJEPZJxPRgk1MMCnEAoIm2pP0sIcVh9Yo0YYGAqORrTOL3AJwIbcy+e8HMNSoN
```

```
V5u51RnrVKie34hMBBARAgAMBQJBgcsBBYMGItmLAAoJEBhZ0B9ne6HsQo0AnA/LCTQ3P5kv
JvDhg1DsfVTFnJxpAJ49WFjg/kIcaN5iP1JfaBAITZI3H4hMBBARAgAMBQJBgcs0BYMGItlY
AAoJEIHC9+viE7aSIiMAnRVTVVAfMXvJhV6D5uHfWeeD046TAJ4kjwP2bHyd6DjCymq+BdED
z63axohMBBARAgAMBQJBgctiBYMGItkqAAoJEGtw7Nldw/RzCaoAmwWM6+Rj1zl4D/PIys5n
W48Hql3hAJ0bLOBthv96g+7oUy9Uj09Uh41lF4hMBBARAgAMBQJB0JMkBYMF1BFoAAoJEH0l
ygrBKafCYlUAoIb1r5D6qMLMPMO1krHk3MNbX5b5AJ4vryx5fw6iJctC5GWJ+Y8ytXab34hM
BBARAgAMBQJCK1u6BYMFeUjSAAoJEOYbpIkV67mr8xMAoJMy+UJC0sqXMPSxh3BUsdcmtFS+
AJ9+Z15LpoOnAidTT/K9iODXGViK6ohMBBIRAgAMBQJAKlk6BYMHektSAAoJEDyhHzSU+vhh
JlwAnA/gOdwOThjO8O+dFtdbpKuImfXJAJ0TL53QKp92EzscZSz49lD2YkoEqohMBBIRAgAM
BQJAPfq6BYMHZqnSAAoJEPLXXGPjnGWcst8AoLQ3MJWqttMNHDblxSyzXhFGhRU8AJ4ukRzf
NJqElQHQ00ZM2WnCVNzOUIhMBBIRAgAMBQJBDgqEBYMGlpoIAAoJEDnKK/Q9aopf/N0AniE2
fcCKO1wDIwusuGVlC+JvnnWbAKDDoUSEYuNn5qzRbrzWW5zBno/Nb4hMBBIRAgAMBQJCgKU0
BYMFI/9YAAoJEAQNwIV8g5+o4yQAnA9QOFLV5POCddyUMqB/fnctuO9eAJ4sJbLKP/Z3SAiT
pKrNo+XZRxauqIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00b
KXvucSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJYiKJ
AAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8JEg2aLos+5zEY
rB/LsohMBBMRAgAMBQI+TU2EBYMJV1cIAAoJEC27dr+t1MkzBQwAoJU+RuTVSn+TI+uWxUpT
82/ds5NkAJ9bnNodffyMMK7GyMiv/TzifiTD+4hMBBMRAgAMBQJB14B2BYMFzSQWAAoJEGbv
28jNgv0+P7wAn13uu8YkhwfNMJJhWdpK2/qM/4AQAJ40drnKW2qJ5EEIJwtxpwapgrzWiYhM
BBMRAgAMBQJCGIEOBYMFjCN+AAoJEHbBAxyiMW6hoO4An0Ith3Kx5/sixbjZR9aEjoePGTNK
AJ94SldLiESaYaJx2lGIlD9bbVoHQYhdBBMRAgAdBQI+PqMMBQkJZgGABQsHCgMEAxUDAgMW
AgECF4AACgkQjHGNO1By4fVxjgCeKVTBNefwxq1A6IbRr9s/Gu8r+AIAniiKdI1lFhOduUKH
AVprO3s8XerMiF0EExECAB0FAkeslLQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAAKCRCMcY07
UHLh9a6SAJ9/PgZQSPNeQ6LvVVzCALEBJOBt7QCffgs+vWP18JutdZc7XiawgAN9vmmIXQQT
EQIAHQUCR6yUzwUJDTBYqAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQcuH1dCoAoLC6RtsD
9K3N7NOxcp3PYOzH2oqzAKCFHn0jSqxk7E8by3sh+Ay8yVv0BYhdBBMRAgAdBQsHCgMEAxUD
AgMWAgECF4AFAkequSEFCQ0ufRUACgkQjHGNO1By4fUdtwCfRNcueXikBMy7tE2BbfwEyTLB
TFAAnifQGbkmcARVS7nqauGhe1ED/vdgiF0EExECAB0FCwcKAwQDFQMCAxYCAQIXgAUCS3Au
ZQUJEPPyWQAKCRCMcY07UHLh9aA+AKCHDkOBKBrGb8tOg9BIub3LFhMvHQCeIOOot1hHHUls
TIXAUrD8+ubIeZaIZQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAABIJEIxx
jTtQcuH1B2VHUEcAAQFxjgCeKVTBNefwxq1A6IbRr9s/Gu8r+AIAniiKdI1lFhOduUKHAVpr
O3s8XerMiGUEExECAB0FAkeslLQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAASCRCMcY07UHLh
9QdlR1BHAAEBrpIAn38+BlBI815Dou9VXMIAsQEk4G3tAJ9+Cz69Y/Xwm611lzteJrCAA32+
aYhlBBMRAgAdBQsHCgMEAxUDAgMWAgECF4AFAktwL8oFCRDz86cAEgdlR1BHAAEBCRCMcY07
UHLh9bDbAJ4mKWARqsvx4TJ8N1hPJF2oTjkeSgCeMVJljxmD+Jd4SscjSvTgFG6Q1WCIbwQw
EQIALwUCTnc9rSgdIGJ1aWxkQG15c3FsLmNvbSB3aWxsIHN0b3Agd29ya2luZyBzb29uAAoJ
EIxxjTtQcuH1tT0An3EMrSjEkUv29OX05JkLiVfQr0DPAJwKtL1ycnLPv15pGMvSzav8JyWN
3Ih7BDARAgA7BQJCdzX1NB0AT29wcy4uLiBzaG91bGQgaGF2ZSBiZWVuIGxvY2FsISBJJ20g
KnNvKiBzdHVwaWQuLi4ACgkQOcor9D1qil/vRwCdFo08f66oKLiuEAqzlf9iDlPozEEAn2Eg
vCYLCCHjfGosrkrU3WK5NFVgiI8EMBECAE8FAkVvAL9IHQBTaG91bGQgaGF2ZSBiZWVuIGEg
bG9jYWwgc2lnbmF0dXJlLCBvciBzb21ldGhpbmcgLSBXVEYgd2FzIEkgdGhpbmtpbmc/AAoJ
EDnKK/Q9aopfoPsAn3BVqKOalJeF0xPSvLR90PsRlnmGAJ44oisY7Tl3NJbPgZal8W32fbqg
bIkBHAQSAQIABgUCS8IiAwAKCRDc9Osew28OLx5CB/91LHRH0qWjPPyIrv3DTQ06x2gljQ1r
Q1MWZNuoeDfRcmgbrZxdiBzf5Mmd36liFiLmDIGLEX8vyT+Q9U/Nf1bRh/AKFkOx9PDSINWY
bE6zCI2PNKjSWFarzr+cQvfQqGX0CEILVcU1HDxZlir1nWpRccccnasMBFp52+koc6PNFjQ13
HpHbM3IcPHaaV8JD3ANyFYS4l0C/S4etDQdX37GruVb9Dcv9XkC5TS2KjDIBsEs89isHrH2+
3ZlxdLsE7LxJ9DWLxbZAND9OiiuThjAGK/pYJb+hyLLuloCg85ZX81/ZLqEOKyl55xuTvCql
tSPmSUObCuWAH+OagBdYSduxiQEiBBABAgAMBQJJKmigBQMAEnUAAAoJEJcQuJvKV618U4wI
AKk/45VnuUf9w1j7fvdzgWdIjT9Lk9dLQAGB13gEVZEVYqtYF5cEZzyxl8c7NUTCTNX3qLId
ul114A4CQQDg5U9bUwwUKaUfGLaz380mtKtM9V9A4fl9H2Gfsdumr8RPDQihfUUqju+d0ycd
mcUScj48Nctx0xhCCWNjOFPERHi9hjRQq7x6RKyFTLjM5ftdInHCo9S+mzyqz9O+iMgX68Mm
+AVgdWSC9L6yGnw6H97GD28oRMGWBTzsmCyqf9I3YutH8mGXRot3QbSJD7/AeZVh1BQwVoJn
CT8Eo1pc/OYZkRRndE1thrX0yjuFwTeOzvqeHlgzEW/FtOCBW7iR0WSJASIEEAECAAwFAkoz
TogFAwASdQAACgkQlxC4m8pXrXwXiAf+Ked6Mgd98YyTyNiLHhllPulboCnKgj430jLzkfgv
7ytVCu1xMfKrRWRw3fA9LC19mzNQX/So/o/ywsk0nUG2sfEs5FiMk+aC957Ic/MDagmXqKap
ZROJbzbZ/KNj9rPCG9kXPGa9sUn6vk39nnv4hri30tNKpM0fMxRhpcoNoCrNl4rs/QTpdRpp
7KBuNaMEtDU7R7OjMDL4qT+BcCmYMIYW4dIV7tmaC0VxtcszZcVCkxSigRMPZHwxSx37GdCx
9/+TqlA4vGL6NQSxZKv+Kqa+WTqBngOl6YGO6FxdiXEliNRpf1mafmz6h8XgYXFGpehjuX1n
60Iz0BffuWbpL4kBIgQQAQIADAUCSkRyCgUDABJ1AAAKCRCXELibyletfPaaB/9FCSmYwz7m
vzOfHZOlEAYeLnCS290XGW89o4FYTbw0PBOulygyqj2TMCK68RCNU2KFs/bXBHeS+dDzitMA
fSaULYi7LJuCCmrDM5SX5aLSj6+TxkDQDR1K1ZE3y6qd4Kx3VeeoN7Wu+oLj/3Jjbbe0uYCQ
+/PniRra9f0Z0neTExZ7CGtVBIsKS1CnKBTR26MZMOom2eTRZwGFUX1PzuW/dbZ4Z0+J6XMd
Tm2td7OYYWPbV3noblkUrxyjtGtO3ip3Oe3zSCWHUFMaaEuXOMw8tN51wy6ybcPVAH0hOiBw
b3iCFJ/20QqaZEno6edYzkqf0pwvrcTmiPb+Vj0fnjBJiQEiBBABAgAMBQJKVj5HBQMAEnUA
AAoJEJcQuJvKV61845AH/R3IkGIGOB/7x3fI0gOkOS0uFljDxysiM8FV06BfXbFpRgFMZxAh
NFUdKCDN98MDkFBd5S5aGkvhAHS7PVwQ8/BIyJaJeUG3AXmrpFV/c9kYn1+YW5OQ9E7tKu5l
5UOj1Y/weNtC04u6Rh/nrp6CvMBhH2nvhSBZ+2kO2auqtFOhuK6+wUHGixt5EK8RAKs3Sf6n
kP2EJUHzy1Q8ec5YDiaV24AVkPFBZMCkpD3Z+seIGrL4zUkV7PPY4zd9g34Oqj8JvtnA4AD/
Z1vBLujLixcQdt9aieOySA9DAVgHbe2wVS4zi5nBURsmD5u96CUOwNK1sOV+ACtdIv/T5qSU
VweJASIEEAECAAwFAkpoCoQFAwASdQAACgkQlxC4m8pXrXysfQf+IJyIPhTphk0kGPQY3v9e
3znW30VahyZxoL6q25eeQWGmVeTFlU4JThUEyzgYGip8i9qBsFPJ9XgOL5bxTGv7/WOK7eX8
e+gXHB3A2QYbrM0GFZKN3BCkbA++HmvJXU58tf+aBCB0ObG+rPn6QUNSPibu4tp65TaPVPSV
```

```
HjNTTICxu3sneHB+okJcc5z1ubme8nAytKb6x0JM/keNSXAev2ZN7zG5m+Pqw7/DQ/gCogzG
ML1bulP2rSh8bYpJPC3vAVuHTmxsbhRBg4l7j5KiHf4qMBrVzRy+YiHhwpf2p8JbCGF141+H
UD1VMeGeXnNO/9SO+dC2OGUf8WrV4FIpxIkBIgQQAQIADAUCSnkuCgUDABJ1AAAKCRCXELib
yletfBjrCACDd/zvoveoNlNiUUBazelcGXwaxSvUMSROUQNkxkoMzfA+aFpYFHWEwDfLqndp
oJTIkgkESd5fODJT26oLFekLvx3mpzfGz8l39KzDM1i6+7Mtg7DnA3kvfVIuZBNDwqoTS6hH
KcGa0MJDgzZQqJ9Ke/7T7eY+HzktUBLjzUY2kv5VV8Ji0p6xY27jT73xiDov00ZbBFN+xBtx
2iRmjjgnPtjt/zU5sLiv9fUOA+Pb53gBT+mXMNx2tsg07Kmuz7vfjR5ydoY7guyB3X1vUK9y
AmCW1Gq67eRG934SujZFikO/oZUrwRrQu2jj5v8B7xwtcCFCdpZAIRabD4BTglvPiQEiBBAB
AgAMBQJKjl+9BQMAEnUAAAoJEJcQuJvKV618DTwH/3DzIl1zwr6TTtTfTBH9FSDdhvaUEPKC
bLT3WZWzIHREaLEENcQ85cGoYoBeJXVBIwBczZUpGy4pqFjYcWQ9vKFm2Nt1Nrs+v9tKc+9G
ECH0Y1a+9GDYqnepcN2O/3HLASCEpXFwQhVe01G+lupGgqYfMgTG9RByTkMzVXB9ER5gijGC
zjTflYAOFUx2eBBLYa3w/ZZpT+nwRmEUaDpfwq06UPrzMZuhol7SGPZUNz4lz4p2NF8Td9bk
hOiJ3+gORRohbq0HdaRdvSDoP/aGsQltfeF5p0KEcpIHx5B05H1twIkOGFTxyx3nTWqauEJy
2a+Wl5ZBl0hB2TqwAE9Z54KJASIEEAECAAwFAkqgEkcFAwASdQAACgkQlxC4m8pXrXwyXwf/
UPzz+D+n19JWivha7laUxuDzMQCKTcEjFCu4QVZ1rqcBFPoz0Tt74/X75QdmxZizqX1E6lbF
EsbVjL2Mt5zZjedS1vbSbrmn4hV4pHZr08dbflZkNX105g8ZlpsqQ7VyUt5YtWCn0tGNn4B5
Eb6WMeqxQteujV3B7AtMH+CD0ja+A2/p0rHIpqScz8aupksBMCrYqhoT+7/qXNEVkjNmcu2N
mHxfv6dL5Xy/0iJjie2umStu8WTfRTpYmnv2gEhbCdb/zhFvG61GgTBJqv9MvBVGRxnJFd4l
NqlucsadD+UM7WjV3v5VuN2r9KD9wocd/s22ELCRA2wKccvR/nWBkIkBIgQQAQIADAUCSqgQ
AAUDABJ1AAAKCRCXELibyletfAT8B/9cPhH8DlHoiv+cK8rAJMomZqVqOyy4BwsRrakycVlg
7/yvMs74anynSoUf0LgsXADQ29Hmrpf+zC5E5/jPGWNK81x2VBVoB8nZkMSAnkZfOw+mWu9I
Aj2NLcsvt9JYNmAq5R7RrirHsDQ2DIYxRgaE/5CVEVry9YQEj18A13/SYyoB4FWpDI4fRfUW
JbUJrYmfg0p+4zL0YS9F11UhsHUu+g1W1c83N54ozI1v0l3HUwVayzII4E/YNrIkpOaO+o8R
z9g6M6jCg3mwn+OfiZVJO++VOiguJF5KzoZIICMxXE3t5hL87Kroi7UkNwm+YHw3ZaLEBm0B
WAXw4DsJZcpViQEiBBABAgAMBQJKuceJBQMAEnUAAAoJEJcQuJvKV6188KEH/24QK2LV1l42
4Wx3T9G4bJFRWWuuEkTpYJw6ss72lqus9t7BsoGaNLMHQzKAlca9wLTqY826q4nv9anEqwWZ
+Di8kE+UAMUq2BFTL0EvOMJ6i1ZyE8cUFVb1+09tpBWJJS7t3z00uMMMznGuHzSm4MgCnGhA
sOgiuHdPWSlnHnqNJa/SB6UVQxtcDOaqQlLIvhd2HVqrOBRtER3td/YgLO6HSxXpXtz8DBa2
NYQYSwAdlqJAPLBnBsLXwbCswuIDMZZv8BJwUNBEJkokOMv5CXxhPrP5kxWvyBvsIhTk8ph2
GIh/ZRVNDAsChbuU1EJBACpwaMrcgwjPtI7/KTgeZVSJASIEEAECAAwFAkreCMYFAwASdQAA
CgkQlxC4m8pXrXyOQQf7BvRm/3PvFCCksyjBW4EVBW7z/Ps/kBK6bIE9Q7f7QlXFIcGGUIpA
rufXWbV+G4a3Z8LFeFJTovNePfquwpFjneUZn1CG+oVS1AfddvYhAsgkLhQqMbaNJIJ1y4D/
H3xvCna/s7Teufud0JLXoLBedFXeB5Cg2KlEoxINqMo+lm/VGJmbykwqoRvxZLDfnbFag5zG
59+OWw4TC8nzlIQYIBn22YiWRk5zsCJA40O+KL1vwBiFDrREhALQc/YBJKYrRX3ZV4U/EeYD
KB0NCBk1W1tXGCee3uhM0S5VFc1j7Pg58ECuntH5xOy+KMNFljiQwvWfbaFTJvCjFQS+OplX
b4kBIgQQAQIADAUCSu86VAUDABJ1AAAKCRCXELibyletfGs8CACteI2BmKs24GF80JeWTOQI
cvHnCdV7hKZOltbNPBbDv6qTt3iX2GVa10iYhI5Eg3Ojt/hKFJTMlfYZyI1peFodGjv7Lk5l
u7zaNBvT1pBCP+eJspi6rGpSuhtMSb4O5jPclRBmbY+w9wctLyZf1zG+slSdw8adcRXQNFqr
vVIZYOmu2S8FunqLfxpjewiFiDPzAzmbWzMoO2PLCYFhwV6Eh2jO33OGbvBmyHNFZBfX5F/+
kiyeT47MEhrfhytJ6ZOdpxtX8HvbvzPZcDLOI80W6rPTG76KW06ZiZrJ81YCa6a7D01y7BYy
W2HoxzYcuumjRkGF4nqK4Mw+wefCp0H/iQEiBBABAgAMBQJLAF3aBQMAEnUAAAoJEJcQuJvK
V618/q0H/ibXDQG2WQmC1LoT4H+ezXjPgDg8aiuz6f4xibTmrO+L4ScMX+zK0KZVwp6Kau28
Nx+gO0oAUW8mNxhd+cl0ZaY+7RIkxEvkooKKsArBmZT+xrE6CgHlAs3D4Mc+14nfD0aZaUbE
iobWvXlYLl27MELLcWyeMlgbeNoucc473JddvmHSRRM5F9Qp28CvWDEXYqhq1laoaho8+cei
pvzyuO3OTwjuAOqhefOHzAvFrRli99ML8xzF1ZOvBct+36SuYxDXyIhkSd7aG9Us0lW6W5Si
JYt4cDyI0JDhbhZN0tzWYKcKMZMxf8w3jW4sfQL0prhHrARqqPiU8OTUH/VNX5CJASIEEAEC
AAwFAksRgasFAwASdQAACgkQlxC4m8pXrXydogf/a31ofmYFMoE3p9SqGt/v28iyO0j9A1Lm
qKwEhJkxff/X/Qa7pafGQ9J90JQkxYKMxydWPspTbDFMccZWkBK132vZp9Q3FHKpnDPDLK2S
25miTReeAAQNgMMFLeyy7ZHi5YsKwLbKxcSo7/m0jlitNYlmt94imFNpg/mHGsy6O+rLeQTA
opuIzP3VwN6ItL5gIFxqWPmf/V0xh/vxTwLqJ66vECD8vyHrHblUzgiXHgyYbZPxAa2SRRd3
4V38phaZ/QsTkss+Sd/QeHChWyU9d6KengWwcr/nDO+K/hhmnO5Oqz02Upwyxrgi6484HQUN
/Smf44VBsSD1DBjaAKjMr4kBIgQQAQIADAUCSyNN1AUDABJ1AAAKCRCXELibyletfCWiB/9c
EZtdFVcsxpE3hJzM6PBPf+1QKuJORve/7MqNEb3TMWFgBxyOfvD7uMpCJyOrqq5AbUQfZfj9
K7qmzWUMuoYceGIlbdmHFBJwtmaF0BiyHaobgY/9RbdCNcbtzrW34feiW9aDZyvCoLHEVkCC
QACSv3FwdYVkkRB5eihvpwJk5tpScdIA12YLqzmVTFdhrZuYvtDdQHjgoLMO8B9s9kok7D2T
SpveVzXXPH68Z3JkVubhHT7cs+n+9PRvcaVJtsX2VTUY5eFVqmGuAUVrvp2aN8cKQ+mVcCQr
VVIhT9o8YB5925MUx2VJml0y0nkBQuMZyzMEOVGkuU/G+pVrRmmAiQEiBBABAgAMBQJLJyaS
BQMAEnUAAAoJEJcQuJvKV618eU0IAKnVh6ymId9C3ZqVyxwTnOB8RMQceJzwCLqk2RT0dPhN
5ZwUcQN7lCp9hymMutC8FdKRK/ESK21vJF2/576Pln4fIeOIbycBAEvqrL14epATj53uBizo
NOTuwb1kximFERuW3MP4XiFUJB0tPws2vR5UU3t6GoQJJwNoIbz9DK2L6X/Qz3Tb9if6bPSK
U6JR1Yn3Hos9ogg21vWCxgMTKUuPAYhmYjSvkqH3BihXi+c17MVvE7W5GJbQHuJo+MgSxu04
4qnvDHZpf4Mzc30XcG1ohjxefNyeiY2bzdI2yCaCtmWOlCW1Sc2oiE0zwO6lD4hY5XmC2Xql
MLsKB5VNXJGJASIEEAECAAwFAks4Ze4FAwASdQAACgkQlxC4m8pXrXyWXggAon2abiNvRzx9
7364Mjx4IlFvM1tVebzNbOkDwZS1ABqTDGgq/ffZA/VZrU+h2eL97cQyGxJEQ5kkm/v1iobE
ZEFMT0pv9WMzfidqzhdKdcpbbxdaErIjD5fBACKdjazAUeH7zce2v+bBN0l9LZoRiXbNugG9
38lkJ2E4ZTYYfvftL/e4RzOgqR9VD/A5MzxfXFbCVharHbeT8OwZy4Oz2UDaDszHsNKoG1WN
pOSf2HTMBPNcsOSY/hIBRWNxnzdYOkWt7laeLNmN1eUEwzk4J7GnlambPIctOdoEUriMSaey
TkLZGejKnwi/PqARyDW1FsReKNHD753ZMViUnAsq2IkBIgQQAQIADAUCS0oyJwUDABJ1AAAK
CRCXELibyletfGodCAC5hjmxwquHSb8ZL0RifIL3j3iU6U7qLK1TQKkTqgELfUzeF9f8NuNR
txLmzNk1T7YI9iji6NAtnuy43v61OMbqlkV8x69qNP36Owv408wXxEt0s5ViZuVOZJAY075c
YRhopgfmhkh4hbkAoKCLajOR0WUEEsDHsqqj8XLJuGRREURy8TJWaB/cotXsgiJf99gt+gIw
In8tyb3+WVIUHWfw2+Drpd3nfcMqgeO54PePJo0BWWjaar+wgC/76Se286IHcYMrml/Adnvx
```

```
ZaIKmxZmkTmDMCfMnVjRYSKBGjQ9Uu7dws7SMsbbd34f8Jt9nyuRqMcl4INAXthWY/S3Sdil
iQEiBBABAgAMBQJLW/5mBQMAEnUAAAoJEJcQuJvKV6181L8IAKq3ZOQHzqaOoz5wnvj51YG8
nZoW5RG7HOb3mL1D9b+FTTzaIxsLf7STagPwKtM57rU/7ehHIuO/9QQNQ3Mudw17ZiwD0l5X
7iG8/AflWnc6bXfTz18IplRuqyVc0qQeJZhT7MBpklcS4ZGZHPQdtAh4Aw5YXihrbbq6jV7j
CzUmFz4XcT8CkJHIUGoFR0vTmFqlAt2K1imwGMh2IEamPOJ0wsTbBfZbhmkB03RToEjIipGZ
M+NtKS/NL2RJYWZ+FCCcEMoRgmlVmATWw3natgLWwN4Z6K4rGXONWi/0wyFgxZpmjdHmjcXa
Igz8EroVsLbnaV/8yG7cgK5e6M0Fk1iJASIEEAECAAwFAkttIfgFAwASdQAACgkQlxC4m8pX
rXyR3QgAksvAMfqC+ACUEWSVAlepDFR1xI45UwBa2UeBY7KjOOCiZlkGREvx20IOv1gExyPl
zNxDeqmYsl2mleEoH6QlXaJRd8MxIVfAnjAt8izwU2dfDwflTTWgGQYf8q7qeAv1XC34yNge
0JaTD1C55QpmcO51f2ojMsAi36bBJO4Dr59jhVYiDjQADS/d7FpAznlhH9SGUq6ekYb2jxCS
rvt0wRtMyk6YGgts4xEHcN0wC9VTobaXo9xvsqhtUK44Gdvptq1cBFX8byzD6fN8nXp+v8qh
tlPYDqb4muqTh2UXXiWMtvPXo7kkZQ8CvI3YbZ10F1IDLt20VJWFZaJYL2fzyokCIgQQAQIA
DAUCQYHLhQWDBiLZBwAKCRCq4+bOZqFEaKgvEACCErnaHGyUYa0wETjj6DLEXsqeOiXad4i9
aBQxnD35GUgcFofC/nCY4XcnCMMEnmdQ9ofUuU3OBJ6BNJIbEusAabgLooebP/3KEaiCIiyh
HYU5jarpZAh+Zopgs3Oc11mQ1tIaS69iJxrGTLodkAsAJAeEUwTPq9fHFFzC1eGBysoyFWg4
bIjz/zClI+qyTbFA5g6tRoiXTo8ko7QhY2AA5UGEg+83Hdb6akC04Z2QRErxKAqrphHzj8Xp
jVOsQAdAi/qVKQeNKROlJ+iq6+YesmcWGfzeb87dGNweVFDJIGA0qY27pTb2lExYjsRFN4Cb
13NfodAbMTOxcAWZ7jAPCxAPlHUG++mHMrhQXEToZnBFE4nbnC7vOBNgWdjUgXcpkUCkop4b
17BFpR+k8ZtYLSS8p2LLz4uAeCcSm2/msJxT7rC/FvoH8428oHincqs2ICo9zO/Ud4HmmO0O
+SsZdVKIIjinGyOVWb4OOzkAlnnhEZ3o6hAHcREIsBgPwEYVTj/9ZdC0AO44Nj9cU7awaqgt
rnwwfr/o4V2gl8bLSkltZU27/29HeuOeFGjlFe0YrDd/aRNsxbyb2O28H4sG1CVZmC5uK1iQ
BDiSyA7Q0bbdofCWoQzm5twlpKWnY8Oe0ub9XP5p/sVfck4FceWFHwv+/PC9RzSl33lQ6vM2
wIkCIgQTAQIADAUCQp8KHAWDBQWacAAKCRDYwgoJWiRXzyE+D/9uc7z6fIsalfOYoLN60ajA
bQbI/uRKBFugyZ5RoaItusn9Z2rAtn61WrFhu4uCSJtFN1ny2RERg40f56pTghKrD+YEt+Nz
e6+FKQ5AbGIdFsR/2bUk+ZZRSt83e14Lcb6ii/fJfzkoIox9ltkifQxqY7Tvk4noKu4oLSc8
O1Wsfc/y0B9sYUUCmUfcnq58DEmGie9ovUslmyt5NPnveXxp5UeaRc5Rqt9tK2B4A+7/cqEN
rdZJbAMSunt2+2fkYiRunAFPKPBdJBsY1sxeL/A9aKe0viKEXQdAWqdNZKNCi8rd/oOP99/9
lMbFudAbX6nL2DSb1OG2Z7NWEqgIAzjmpwYYPCKeVz5Q8R+if9/fe5+STY/55OaI33fJ2H3v
+U435VjYqbrerWe36xJItcJeqUzW71fQtXi1CTEl3w2ch7VF5oj/QyjabLnAlHgSlkSi6p7B
y5C2MnbCHlCfPnIinPhFoRcRGPjJe9nFwGs+QblvS/Chzc2WX3s/2SWm4gEUKRX4zsAJ5ocy
fa/vkxCkSxK/erWlCPf/J1T70+i5waXDN/E3enSet/WL7h94pQKpjz8OdGL4JSBHuAVGA+a+
dknqnPF0KMKLhjrgV+L7O84FhbmAP7PXm3xmiMPriXf+el5fZZequQoIagf8rdRHHhRJxQgI
0HNknkaOqs8dtrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/lxaZoJYc3a6M0
2WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRiRjd1DTCHqeyX7CHhcghj
/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE7zaD5cH9J7yv/6xuZVw411x0h4Uq
sTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fmLe11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRs
KTfozBu74F47D8Ilbf5vSYHbuE5p/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnl
LzKUb/F5GwADBQf+Lwqqa8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPD
sDD9MZ1ZaSafanFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGo
TbOWI39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42LmuQT5N
dKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt926s/ymfdf5Hkd
Q1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZWhe70YGNPw1yjWJT1IhM
BBgRAgAMBQI+PqMdBQkJZgGAAAoJEIxxjTtQcuH17p4An3r1QpVC9yhnW2cSAjq+kr72GX0e
AJ4295kl6NxYEuFApmr1+0uUq/SlsYhMBBgRAgAMBQJHrJT8BQkNMFjfAAoJEIxxjTtQcuH1
pc4An0I965H3JY2GTrizp+dCezxbhexaAJ48FhocFYvfhZtgeUWb6aPvgQZHT4hUBBgRAgAM
BQI+PqMdBQkJZgGAABIJEIxxjTtQcuH1B2VHUEcAAQHungCfevVClUL3KGdbZxICOr6SvvYZ
fR4Anjb3mSXo3FgS4UCmavX7S5Sr9KWxiFQEGBECAAwFAk53Pe0FCRP7AbgAEgdlR1BHAAEB
CRCMcY07UHLh9RSbAJsFivb5sESf8vYE5yfD1n9AVa6FEwCgpWAIWbl9p1DcB+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.

- 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, `8.3.0`), 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/my.cnf` or the `/etc/mysql` 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
  ```

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
$> 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
```

> **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 `mysqld`, you may need to create them. The following commands add the `mysql` group and the `mysql` user. You might want to call the user and group something else instead of `mysql`. If so, substitute the appropriate name in the following instructions. The syntax for `useradd` and `groupadd` may differ slightly on different versions of Unix/Linux, or they may have different names such as `adduser` and `addgroup`.

```
$> 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.

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 http://www.gnu.org/software/tar/.

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
```

The `ln` command makes a symbolic link to the installation directory. This enables you to refer more easily to it as `/usr/local/mysql`. To avoid having to type the path name of client programs always when you are working with MySQL, you can add the `/usr/local/mysql/bin` directory to your `PATH` variable:

```
$> export PATH=$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

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

Before you proceed with an installation from source, check whether Oracle produces a precompiled binary distribution for your platform and whether it works for you. We put a great deal of effort into ensuring that our binaries are built with the best possible options for optimal performance. Instructions for installing binary distributions are available in 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 `8.3.0`. 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
  ```

- MySQL 8.3 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 12
  - macOS: XCode 10
  - Solaris: GCC 10
  - Windows: Visual Studio 2019
- 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). In MySQL 8.3 and later, these libraries are always bundled with the MySQL source.
- 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 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 -
  ```

  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 5.3, "Configuration: Manually".

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.
```

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

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

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

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

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

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

5. To check out an earlier MySQL branch, run the `git checkout` command, specifying the branch name. For example, to check out the MySQL 8.0 branch:

```
~/mysql-server$ git checkout 8.0
Checking out files: 100% (9600/9600), done.
Branch 8.0 set up to track remote branch 8.0 from origin.
Switched to a new branch '8.0'
```

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 trunk
~/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 by using `WITH_SSL=openssl11` on EL7 or `WITH_SSL=openssl3` 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:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
```

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
ADD_GDB_INDEX	Whether to enable generation of .gdb_index section in binaries	
BUILD_CONFIG	Use same build options as official releases	
BUNDLE_RUNTIME_LIBRARIES	Bundle runtime libraries with server MSI and Zip packages for Windows	OFF
CMAKE_BUILD_TYPE	Type of build to produce	RelWithDebInfo
CMAKE_CXX_FLAGS	Flags for C++ Compiler	
CMAKE_C_FLAGS	Flags for C Compiler	
CMAKE_INSTALL_PREFIX	Installation base directory	/usr/local/mysql
COMPILATION_COMMENT	Comment about compilation environment	
COMPILATION_COMMENT_SERVER	Comment about compilation environment for use by mysqld	
COMPRESS_DEBUG_SECTIONS	Compress debug sections of binary executables	OFF
CPACK_MONOLITHIC_INSTALL	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_COND	Exclude Performance Schema condition instrumentation	OFF
DISABLE_PSI_DATA_LOCK	Exclude the performance schema data lock instrumentation	OFF
DISABLE_PSI_ERROR	Exclude the performance schema server error instrumentation	OFF
DISABLE_PSI_FILE	Exclude Performance Schema file instrumentation	OFF
DISABLE_PSI_IDLE	Exclude Performance Schema idle instrumentation	OFF
DISABLE_PSI_MEMORY	Exclude Performance Schema memory instrumentation	OFF
DISABLE_PSI_METADATA	Exclude Performance Schema metadata instrumentation	OFF
DISABLE_PSI_MUTEX	Exclude Performance Schema mutex instrumentation	OFF
DISABLE_PSI_PS	Exclude the performance schema prepared statements	OFF
DISABLE_PSI_RWLOCK	Exclude Performance Schema rwlock instrumentation	OFF
DISABLE_PSI_SOCKET	Exclude Performance Schema socket instrumentation	OFF

Formats	Description	Default
DISABLE_PSI_SP	Exclude Performance Schema stored program instrumentation	OFF
DISABLE_PSI_STAGE	Exclude Performance Schema stage instrumentation	OFF
DISABLE_PSI_STATEMENT	Exclude Performance Schema statement instrumentation	OFF
DISABLE_PSI_STATEMENT_DIGEST	Exclude Performance Schema statements_digest instrumentation	OFF
DISABLE_PSI_TABLE	Exclude Performance Schema table instrumentation	OFF
DISABLE_PSI_THREAD	Exclude the performance schema thread instrumentation	OFF
DISABLE_PSI_TRANSACTION	Exclude the performance schema transaction instrumentation	OFF
ENABLED_LOCAL_INFILE	Whether to enable LOCAL for LOAD DATA	OFF
ENABLED_PROFILING	Whether to enable query profiling code	ON
ENABLE_EXPERIMENTAL_SYSVARS	Whether to enabled experimental InnoDB system variables	OFF
ENABLE_GCOV	Whether to include gcov support	
ENABLE_GPROF	Enable gprof (optimized Linux builds only)	OFF
FORCE_COLORED_OUTPUT	Whether to colorize compiler output	OFF
FORCE_INSOURCE_BUILD	Whether to force an in-source build	OFF
FORCE_UNSUPPORTED_COMPILER	Whether to permit unsupported compilers	OFF
FPROFILE_GENERATE	Whether to generate profile guided optimization data	OFF
FPROFILE_USE	Whether to use profile guided optimization data	OFF
HAVE_PSI_MEMORY_INTERFACE	Enable performance schema memory tracing module for memory allocation functions used in dynamic storage of over-aligned types	OFF
IGNORE_AIO_CHECK	With -DBUILD_CONFIG=mysql_release, ignore libaio check	OFF
INSTALL_BINDIR	User executables directory	PREFIX/bin
INSTALL_DOCDIR	Documentation directory	PREFIX/docs
INSTALL_DOCREADMEDIR	README file directory	PREFIX
INSTALL_INCLUDEDIR	Header file directory	PREFIX/include

Formats	Description	Default
INSTALL_INFODIR	Info file directory	PREFIX/docs
INSTALL_LAYOUT	Select predefined installation layout	STANDALONE
INSTALL_LIBDIR	Library file directory	PREFIX/lib
INSTALL_MANDIR	Manual page directory	PREFIX/man
INSTALL_MYSQLKEYRINGDIR	Directory for keyring_file plugin data file	platform specific
INSTALL_MYSQLSHAREDIR	Shared data directory	PREFIX/share
INSTALL_MYSQLTESTDIR	mysql-test directory	PREFIX/mysql-test
INSTALL_PKGCONFIGDIR	Directory for mysqlclient.pc pkg-config file	INSTALL_LIBDIR/pkgconfig
INSTALL_PLUGINDIR	Plugin directory	PREFIX/lib/plugin
INSTALL_PRIV_LIBDIR	Installation private library directory	
INSTALL_SBINDIR	Server executable directory	PREFIX/bin
INSTALL_SECURE_FILE_PRIVDIR	secure_file_priv default value	platform specific
INSTALL_SHAREDIR	aclocal/mysql.m4 installation directory	PREFIX/share
INSTALL_STATIC_LIBRARIES	Whether to install static libraries	ON
INSTALL_SUPPORTFILESDIR	Extra support files directory	PREFIX/support-files
LINK_RANDOMIZE	Whether to randomize order of symbols in mysqld binary	OFF
LINK_RANDOMIZE_SEED	Seed value for LINK_RANDOMIZE option	mysql
MAX_INDEXES	Maximum indexes per table	64
MSVC_CPPCHECK	Enable MSVC code analysis.	ON
MUTEX_TYPE	InnoDB mutex type	event
MYSQLX_TCP_PORT	TCP/IP port number used by X Plugin	33060
MYSQLX_UNIX_ADDR	Unix socket file used by X Plugin	/tmp/mysqlx.sock
MYSQL_DATADIR	Data directory	
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF
MYSQL_PROJECT_NAME	Windows/macOS project name	MySQL
MYSQL_TCP_PORT	TCP/IP port number	3306
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock
NDB_UTILS_LINK_DYNAMIC	Cause NDB tools to be dynamically linked to ndbclient	
ODBC_INCLUDES	ODBC includes directory	
ODBC_LIB_DIR	ODBC library directory	
OPTIMIZER_TRACE	Whether to support optimizer tracing	

Formats	Description	Default
OPTIMIZE_SANITIZER_BUILDS	Whether to optimize sanitizer builds	ON
REPRODUCIBLE_BUILD	Take extra care to create a build result independent of build location and time	
SHOW_SUPPRESSED_COMPILER_WARNINGS	Whether to show suppressed compiler warnings and not fail with -Werror.	OFF
SYSCONFDIR	Option file directory	
SYSTEMD_PID_DIR	Directory for PID file under systemd	/var/run/mysqld
SYSTEMD_SERVICE_NAME	Name of MySQL service under systemd	mysqld
TMPDIR	tmpdir default value	
WIN_DEBUG_NO_INLINE	Whether to disable function inlining	OFF
WITHOUT_SERVER	Do not build the server	OFF
WITHOUT_xxx_STORAGE_ENGINE	Exclude storage engine xxx from build	
WITH_ANT	Path to Ant for building GCS Java wrapper	
WITH_ASAN	Enable AddressSanitizer	OFF
WITH_ASAN_SCOPE	Enable AddressSanitizer -fsanitize-address-use-after-scope Clang flag	OFF
WITH_AUTHENTICATION_CLIENT_PLUGINS	Enabled automatically if any corresponding server authentication plugins are built	
WITH_AUTHENTICATION_LDAP	Whether to report error if LDAP authentication plugins cannot be built	OFF
WITH_AUTHENTICATION_PAM	Build PAM authentication plugin	OFF
WITH_AWS_SDK	Location of Amazon Web Services software development kit	
WITH_BUILD_ID	On Linux systems, generate a unique build ID	ON
WITH_CLASSPATH	Classpath to use when building MySQL Cluster Connector for Java. Default is an empty string.	
WITH_CLIENT_PROTOCOL_TRACING	Build client-side protocol tracing framework	ON
WITH_CURL	Location of curl library	
WITH_DEBUG	Whether to include debugging support	OFF
WITH_DEFAULT_COMPILER_OPTIONS	Whether to use default compiler options	ON

Formats	Description	Default
WITH_DEVELOPER_ENTITLEMENTS	Whether to add the 'get-task-allow' entitlement to all executables on macOS to generate a core dump in the event of an unexpected server halt	OFF
WITH_EDITLINE	Which libedit/editline library to use	bundled
WITH_ERROR_INSERT	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF
WITH_FIDO	Type of FIDO library support	bundled
WITH_ICU	Type of ICU support	bundled
WITH_INNODB_EXTRA_DEBUG	Whether to include extra debugging support for InnoDB.	OFF
WITH_JEMALLOC	Whether to link with -ljemalloc	OFF
WITH_KEYRING_TEST	Build the keyring test program	OFF
WITH_LD	Whether to use the LLVM lld or mold linker	
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
WITH_LSAN	Whether to run LeakSanitizer, without AddressSanitizer	OFF
WITH_LTO	Enable link-time optimizer	OFF
WITH_LZ4	Type of LZ4 library support	bundled
WITH_MECAB	Compiles MeCab	
WITH_MSAN	Enable MemorySanitizer	OFF
WITH_MSCRT_DEBUG	Enable Visual Studio CRT memory leak tracing	OFF
WITH_MYSQLX	Whether to disable X Protocol	ON
WITH_NDB	Build MySQL NDB Cluster	OFF
WITH_NDBAPI_EXAMPLES	Build API example programs	OFF
WITH_NDBCLUSTER	Build the NDB storage engine	OFF
WITH_NDBCLUSTER_STORAGE_ENGINE	For internal use; may not work as expected in all circumstances; users should employ WITH_NDBCLUSTER instead	ON
WITH_NDBMTD	Build multithreaded data node.	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	ON

Formats	Description	Default
	default. Supported in MySQL Cluster only.	
WITH_NDB_PORT	Default port used by a management server built with this option. If this option was not used to build it, the management server's default port is 1186.	[none]
WITH_NDB_TEST	Include NDB API test programs.	OFF
WITH_NDB_TLS_SEARCH_PATH	Default path used by NDB programs to search for TLS certificate and key files.	$HOME/ndb-tls
WITH_NUMA	Set NUMA memory allocation policy	
WITH_PACKAGE_FLAGS	For flags typically used for RPM/DEB packages, whether to add them to standalone builds on those platforms	
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
WITH_ROUTER	Whether to build MySQL Router	ON
WITH_SHOW_PARSE_TREE	Support for SHOW PARSE_TREE debugging statement	
WITH_SSL	Type of SSL support	system
WITH_SYSTEMD	Enable installation of systemd support files	OFF
WITH_SYSTEMD_DEBUG	Enable additional systemd debug information	OFF
WITH_SYSTEM_LIBS	Set system value of library options not set explicitly	OFF
WITH_TCMALLOC	Whether to link with -ltcmalloc	OFF
WITH_TEST_TRACE_PLUGIN	Build test protocol trace plugin	OFF
WITH_TSAN	Enable ThreadSanitizer	OFF
WITH_UBSAN	Enable Undefined Behavior Sanitizer	OFF
WITH_UNIT_TESTS	Compile MySQL with unit tests	ON
WITH_UNIXODBC	Enable unixODBC support	OFF
WITH_VALGRIND	Whether to compile in Valgrind header files	OFF
WITH_WIN_JEMALLOC	Path to directory containing jemalloc.dll	
WITH_ZLIB	Type of zlib support	bundled
WITH_ZSTD	Type of zstd support	bundled

Formats	Description	Default
`WITH_xxx_STORAGE_ENGINE`	Compile storage engine xxx statically into server	

General Options

- `-DBUILD_CONFIG=mysql_release`

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

- `-DWITH_BUILD_ID=bool`

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

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

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

- `-DCPACK_MONOLITHIC_INSTALL=bool`

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

- `-DFORCE_INSOURCE_BUILD=bool`

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

- `-DFORCE_COLORED_OUTPUT=bool`

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

Installation Layout Options

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

- `-DCMAKE_INSTALL_PREFIX=dir_name`

The installation base directory.

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

- `-DINSTALL_BINDIR=dir_name`

  Where to install user programs.

- `-DINSTALL_DOCDIR=dir_name`

  Where to install documentation.

- `-DINSTALL_DOCREADMEDIR=dir_name`

  Where to install `README` files.

- `-DINSTALL_INCLUDEDIR=dir_name`

  Where to install header files.

- `-DINSTALL_INFODIR=dir_name`

  Where to install Info files.

- `-DINSTALL_LAYOUT=name`

  Select a predefined installation layout:

  - `STANDALONE`: Same layout as used for `.tar.gz` and `.zip` packages. This is the default.
  - `RPM`: Layout similar to RPM packages.
  - `SVR4`: Solaris package layout.
  - `DEB`: DEB package layout (experimental).

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

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

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

- `-DINSTALL_LIBDIR=dir_name`

  Where to install library files.

- `-DINSTALL_MANDIR=dir_name`

  Where to install manual pages.

- `-DINSTALL_MYSQLKEYRINGDIR=dir_path`

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

- `-DINSTALL_MYSQLSHAREDIR=dir_name`

  Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=dir_name`

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

- `-DINSTALL_PKGCONFIGDIR=dir_name`

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

- `-DINSTALL_PLUGINDIR=dir_name`

  The location of the plugin directory.

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

- `-DINSTALL_PRIV_LIBDIR=dir_name`

  The location of the dynamic library directory.

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

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

- `-DINSTALL_SBINDIR=dir_name`

  Where to install the `mysqld` server.

- `-DINSTALL_SECURE_FILE_PRIVDIR=dir_name`

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

- `-DINSTALL_SHAREDIR=dir_name`

  Where to install `aclocal/mysql.m4`.

- `-DINSTALL_STATIC_LIBRARIES=bool`

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

- `-DINSTALL_SUPPORTFILESDIR=dir_name`

  Where to install extra support files.

- `-DLINK_RANDOMIZE=bool`

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

- `-DLINK_RANDOMIZE_SEED=val`

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

- `-DMYSQL_DATADIR=dir_name`

  The location of the MySQL data directory.

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

- `-DODBC_INCLUDES=dir_name`

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

- `-DODBC_LIB_DIR=dir_name`

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

- `-DSYSCONFDIR=dir_name`

  The default `my.cnf` option file directory.

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

- `-DSYSTEMD_PID_DIR=dir_name`

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

  This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DSYSTEMD_SERVICE_NAME=name`

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

  This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DTMPDIR=dir_name`

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

Storage Engine Options

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

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

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

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

To build MySQL with support for NDB Cluster, use the `WITH_NDB` option.

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

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

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

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

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

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

Feature Options

- `-DADD_GDB_INDEX=bool`

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

- `-DCOMPILATION_COMMENT=string`

  A descriptive comment about the compilation environment. While `mysqld` uses `COMPILATION_COMMENT_SERVER`, other programs 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.

- `-DCOMPILATION_COMMENT_SERVER=string`

  A descriptive comment about the compilation environment for use by `mysqld` (for example, to set the `version_comment` system variable). Programs other than the server use `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_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).

- `-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”.

- `-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.

- `-DENABLE_GCOV=bool`

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

- `-DENABLE_GPROF=bool`

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

- `-DENABLED_LOCAL_INFILE=bool`

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

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

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

- `-DENABLED_PROFILING=bool`

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

- `-DFORCE_UNSUPPORTED_COMPILER=bool`

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

- `-DSHOW_SUPPRESSED_COMPILER_WARNINGS=bool`

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

- `-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.

- `-DFPROFILE_USE=bool`

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

  Enabling `FPROFILE_USE` also enables `WITH_LTO`.

- `-DHAVE_PSI_MEMORY_INTERFACE=bool`

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

- `-DIGNORE_AIO_CHECK=bool`

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

- `-DMAX_INDEXES=num`

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

- `-DMYSQL_MAINTAINER_MODE=bool`

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

- `-DWITH_DEVELOPER_ENTITLEMENTS=bool`

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

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

- `-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 MySQL Internals: 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 defaults to `ON` for `RelWithDebInfo` builds.

- `-DWITH_LD=string`

  `CMake` uses the standard linker by default. Optionally pass in `lld` or `mold` to specify an alternative linker. `mold` must be version 2 or newer.

  This option can be used on Linux-based systems other than Enterprise Linux, which always uses the `ld` linker.

  > **Note**
  >
  > Previously, the option `USE_LD_LLD` could be used to enable (the default) or disable explicitly the LLVM `lld` linker for Clang. In MySQL 8.3, `USE_LD_LLD` has been removed.

- `-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.

- `-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.

- `-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_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), a path name to the `curl` library, `no`|`off`|`none` to disable curl support, or `bundled` to use the bundled curl distribution in `extra/curl/`.

- `-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_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.

    MySQL includes `fido2` version 1.8.0.

  - `system`: Use the system FIDO library.

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

- `-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_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.

- -DWITH_WIN_JEMALLOC=*string*

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

  This option is enabled for official MySQL binaries for Windows.

- -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. 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.

- `-DWITH_LSAN=bool`

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

- `-DWITH_LTO=bool`

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

- `-DWITH_LZ4=lz4_type`

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

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

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

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

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

- `-DWITH_MSAN=bool`

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

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

- `-DWITH_MSCRT_DEBUG=bool`

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

- `-DMSVC_CPPCHECK=bool`

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

- `-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.

- `-DWITH_PROTOBUF=protobuf_type`

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

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

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

- `-DWITH_RAPID=bool`

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

- `-DWITH_RAPIDJSON=rapidjson_type`

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

  - `bundled`: Use the RapidJSON library bundled with the distribution. This is the default.
  - `system`: Use the system RapidJSON library. Version 1.1.0 or later is required.

- `-DWITH_ROUTER=bool`

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

- `-DWITH_SSL={ssl_type|path_name}`

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

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

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

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

%ProgramFiles%/OpenSSL-Win32, %ProgramFiles%/OpenSSL-Win64, C:/OpenSSL, C:/OpenSSL-Win32, and C:/OpenSSL-Win64.

- yes: This is a synonym for system.
- openssl*version*: Use an alternate OpenSSL system package such as openssl11 on EL7, or openssl3 (or openssl3-fips) 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 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_SHOW_PARSE_TREE=*bool*

  Enables support for SHOW PARSE_TREE in the server, used in development and debugging only. Not used for release builds or supported in production.

- -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.

  When the server was built using this option, MySQL includes all systemd messages in the server's error log (see The Error Log).

  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_LZ4, WITH_LZMA, WITH_PROTOBUF, WITH_RE2, WITH_SSL, WITH_ZLIB, WITH_ZSTD.

- -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.

- -DWITH_TCMALLOC=*bool*

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

  WITH_TCMALLOC and WITH_JEMALLOC are mutually exclusive.

- -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_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:

  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.

Compiler Flags

- `-DCMAKE_C_FLAGS="flags"`

  Flags for the C compiler.

- `-DCMAKE_CXX_FLAGS="flags"`

  Flags for the C++ compiler.

- `-DWITH_DEFAULT_COMPILER_OPTIONS=bool`

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

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

- `-DOPTIMIZE_SANITIZER_BUILDS=bool`

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

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

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

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

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

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

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

CMake Options for Compiling NDB Cluster

The following options are for use when building the MySQL sources with NDB Cluster support.

- `-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.

- `-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.

- `-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}`

  Controls (only) whether the `ndbcluster` plugin 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}`

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

  This option is deprecated and subject to eventual removal; use `WITH_NDB` instead.

- `-DWITH_NDBMTD={ON|OFF}`

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

- `-DWITH_NDB_DEBUG={ON|OFF}`

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

- `-DWITH_NDB_JAVA={ON|OFF}`

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

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

- `-DWITH_NDB_PORT=`*`port`*

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

- `-DWITH_NDB_TEST={ON|OFF}`

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

- `-DWITH_NDB_TLS_SEARCH_PATH=`*`path`*

  Set the default path searched by `ndb_sign_keys` and other `NDB` programs for TLS certificate and key files.

  The default for Windows platforms is `$HOMEDIR/ndb-tls`; for other platforms, such as Linux, it is `$HOME/ndb-tls`.

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 re-create it, you might encounter this error:

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

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

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

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

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

For information about acquiring or updating tools, see the system requirements in Chapter 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 8.3.0, the file looks like this:

```
MYSQL_VERSION_MAJOR=8
MYSQL_VERSION_MINOR=3
MYSQL_VERSION_PATCH=0
MYSQL_VERSION_EXTRA="INNOVATION"
```

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 https://dev.mysql.com/doc/index-other.html. 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

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: the MSI, the standard binary distribution (packaged as a compressed file) containing all of the necessary files that you unpack, and source files to compile MySQL yourself. For related information, see Section 5.1, “Choosing an Installation Package”.

> **Note**
>
> MySQL 8.3 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.

Recommended MSI Installation Method

The simplest and recommended method is to download the MSI and let it install MySQL Server, and then use the MySQL Configurator it installs to configure MySQL:

1. Download the MSI from https://dev.mysql.com/downloads/ and execute it. This installs the MySQL server, an associated MySQL Configurator application, and it adds related MySQL items to the Microsoft Windows Start menu under the `MySQL` group.

2. Upon completion, the installation wizard prompts to execute MySQL Configurator. Execute it now (recommended) or later, or instead choose to manually configure MySQL.

   > **Note**
   >
   > The MySQL server won't start until it's configured; it's recommended to execute the bundled MySQL Configurator immediately after the MSI.

MySQL is now installed. If you used MySQL Configurator to configure MySQL as a Windows service, then Windows automatically starts the MySQL server every time you restart the system. Also, the MSI installs the MySQL Configurator application on the local host, which you can use later to reconfigure MySQL server. It and other MySQL start up menu items were added by the MSI.

MySQL Installation Layout on Microsoft Windows

For MySQL 8.3 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.3` for installations using the MSI, although the MSI **Custom** setup type allows using a different location. If you use the ZIP archive method to install MySQL, install it there are elsewhere, such as `C:\mysql`. Regardless, 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 8.3\`	Log files, databases	The Windows system variable `%PROGRAMDATA%` defaults to `C:\ProgramData`.
`docs`	Release documentation	With the MSI, use the `Custom` type to include this optional component.
`include`	Include (header) files	
`lib`	Libraries	
`share`	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

Additional Installation Information

By default, MySQL Configurator sets up the MySQL server 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 related information about manually setting up the Windows service, see Section 5.3.8, “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 5.6, “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.
- 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.

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 the MSI to install MySQL server and MySQL Configurator to configure it. The MSI is simpler to use than the compressed file, and you need no additional tools to get MySQL up and running. MySQL Configurator automatically configures MySQL Server, creates an options file, starts the server, enables you to create default user accounts, and more. For more information on choosing a package, see Section 5.1, “Choosing an Installation Package”.

> **Note**
>
> Before MySQL 8.1, an application named MySQL Installer included functionality now present in MySQL Configurator. The MySQL Installer both installed and configured MySQL products but it does not exist as of MySQL 8.1.

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 5.3.2, “Creating an Option File”.

5.1 Choosing an Installation Package

For MySQL 8.3, there are multiple installation package formats to choose from when installing MySQL on Windows. The package formats described in this section are:

- MySQL Installation MSI
- MySQL noinstall ZIP Archives
- MySQL Docker Images

MySQL Installation MSI

This package has a file name similar to `mysql-community-8.3.0.msi` or `mysql-commercial-8.3.0.msi`, and installs MySQL server along with MySQL Configurator. The MSI includes a MySQL Configurator application that is recommended for most users to set up, configure, and reconfigure the MySQL server.

The MSI and MySQL Configurator operate on all MySQL supported versions of Windows (see https://www.mysql.com/support/supportedplatforms/database.html). For instructions on how to configure MySQL using MySQL Configurator, see Section 5.2, “Configuration: Using MySQL Configurator”.

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, but does include MySQL Configurator to configure the MySQL server.

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-debug-test.zip`.

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.

To install MySQL by extracting the Zip archive rather than use the MSI, consider 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 5.3.2, “Creating an Option File”.

   > **Note**
   >
   > The MSI installs MySQL under `C:\Program Files\MySQL\MySQL Server 8.3\`.

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. Configure the MySQL server using either MySQL Configurator (recommended) or Section 5.3, “Configuration: Manually”.

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.

5.2 Configuration: Using MySQL Configurator

MySQL Configurator is a standalone application designed to ease the complexity of configuring a MySQL server to run MySQL on Microsoft Windows. It is bundled with the MySQL server, in both the MSI and standalone Zip archive.

Methods to Start MySQL Configurator

MySQL Configurator can both configure and reconfigure MySQL server; and the methods to start MySQL Configurator are:

- The MySQL server MSI prompts to execute MySQL Configurator immediately after it installs the MySQL server.
- From the Start Menu: the MSI creates a MySQL Configurator start menu item.
- From the command line: the `mysql-configurator.exe` executable is located in the same directory as `mysqld.exe` and other MySQL binaries installed with the MySQL server.

  Typically this location is in `C:\Program Files\MySQL\MySQL Server X.Y\bin` if installed via the MSI, or a custom directory for the Zip archive.

5.2.1 MySQL Server Configuration with MySQL Configurator

MySQL Configurator performs the initial configuration, a reconfiguration, and also functions as part of the uninstallation process.

> **Note**
>
> Full permissions are granted to the user executing MySQL Configurator 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 `%ProgramData%` that is owned by `SYSTEM`.

MySQL Configurator performs the 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.
- It can optionally create MySQL server user accounts with configurable permissions based on general roles, such as DB Administrator, DB Designer, and Backup Admin.
- 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.

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.

5.2.1.1 MySQL Server Installations

MySQL Configurator adds an upgrade option if it finds an existing MySQL Server installation is discovered. It offers two options:

> **Note**
>
> This upgrade functionality was added in MySQL 8.3.0.

In-Place Upgrade of an Existing MySQL Server Installation

This replaces the existing MySQL server installation as part of the upgrade process which may also upgrade the data schema. Upon success, the existing MySQL server installation is removed from the system.

> **Note**
>
> The existing MySQL server instance must be running for the in-place upgrade option to function.

While MySQL Configurator may attempt (and succeed) to perform an in-place upgrade for other scenarios, the following table lists the scenarios officially supported by the configurator:

Table 5.2 Supported Upgrade Paths

A supported upgrade scenario	Description
8.0.35+ to 8.1	From 8.0.35 or higher to the first MySQL 8 Innovation release.
8.0.35+ to 8.4	From 8.0.35 or higher to the MySQL next LTS release.
8.X to 8.Y where Y = X + 1	From an Innovation release to the next consecutive Innovation release.
8.3 to 8.4	From the last MySQL 8 Innovation release to the next MySQL 8 LTS release.
8.4.X to 8.4.Y where Y > X	From within the same LTS release.
8.4.X to 9.0.0	From an LTS release to the first consecutive Innovation release.
8.4 to 9.7	From an LTS release to the next consecutive LTS release.

This dialogue prompts for the protocol (default: TCP/IP), port (default: 3306), and root password for the existing installation. Execute connect and then review and confirm the MySQL instance's information (such as version, paths, and configuration file) before proceeding with the upgrade.

This upgrade may replace the file paths. For example, "MySQL Server 8.2\Data\" changes to "MySQL Server 8.3\Data\" when upgrading 8.2 to 8.3.

This upgrade functionality also provides these additional options: "Backup Data" allows running `mysqldump` before performing the upgrade, and "Server File Permissions" to optionally customize file permissions.

Add a Separate MySQL Server Installation

Configure a standard side-by-side installation with the new MySQL server installation. This means having multiple MySQL server installations installed and running on the system.

5.2.1.2 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 Configurator 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 Configurator 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** defaults to `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 error 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 `MySQL`.

- 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).

5.2.1.3 Accounts and Roles

- Root Account Password

  Assigning a root password is required and you will be asked for it when reconfiguring with MySQL Configurator in the future. 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 ( ) 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 Configurator 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\domainuser`*) or MySQL syntax (*`domain\\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 Configurator 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.

5.2.1.4 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:

  - **Windows Service Name**

    Defaults to MySQL*XY* where XY is 81 for MySQL 8.1.

  - **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.

5.2.1.5 Server File Permissions

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

- MySQL Configurator 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 Configurator use a configuration option similar to the one just described, but also have MySQL Configurator 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 Configurator 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.

5.2.1.6 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
- Binary Log

5.2.1.7 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**

  These options only apply to the initial configuration of the MySQL server.

  - Lower Case

    Sets the `lower_case_table_names` 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.

5.2.1.8 Sample Databases

Optionally install sample databases that include test data to help develop applications with MySQL. The options include the sakila and world databases.

5.2.1.9 Apply 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.

5.3 Configuration: Manually

These instructions apply to those *not* using the MySQL Configurator application to configure and set up the MySQL server; or for manually making additional changes. Using MySQL Configurator is recommended.

5.3.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version then 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 5.3.2, “Creating an Option File”.

   > **Note**
   >
   > The MSI installs MySQL under `C:\Program Files\MySQL\MySQL Server 8.3`.

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.

5.3.2 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 8.3` and `C:\Program Files\MySQL\MySQL Server 8.3\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 MySQL Configurator to configure MySQL Server, it creates the `my.ini` at the default location, and the user executing MySQL Configurator 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 8.3\data`) to `E:\mydata`.
2. Use a `--datadir` option to specify the new data directory location each time you start the server.

5.3.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 8.3.

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

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 8.3 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.

5.3.4 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”.

5.3.5 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 using MySQL Configurator.

The examples in these sections assume that MySQL is installed under the default location of `C:\Program Files\MySQL\MySQL Server 8.3`. 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 5.3.3, “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 8.3\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 8.3\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”.

5.3.6 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 8.3\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 8.3\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 8.3\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 `mysqld` with the `--standalone` and `--debug` options. In this case, `mysqld` writes a log file `C:\mysqld.trace` that should contain the reason why `mysqld` doesn't start. See The DBUG Package.

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

5.3.7 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 8.3\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.

5.3.8 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:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.3\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 Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.3\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 8.3\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 8.3\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 *mysqld_service_name* or NET START *mysqld_service_name* command. SC and NET commands are not case-sensitive.

When run as a service, mysqld has no access to a console window, so no messages can be seen there. If mysqld 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, C:\Program Files\MySQL\MySQL Server 8.3\data). It is the file with a suffix of .err.

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 *mysqld_service_name* command, the NET STOP *mysqld_service_name* command, or the mysqladmin shutdown 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 8.3\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 8.3\bin\mysqld" --remove
```

If mysqld is not running as a service, you can start it from the command line. For instructions, see Section 5.3.6, “Starting MySQL from the Windows Command Line”.

If you encounter difficulties during installation, see Section 5.4, “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.

5.3.9 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 8.3\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 8.3\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 8.3\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 8.3\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 `mysqlshow`, see mysqlshow — Display Database, Table, and Column Information.

5.4 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 8.3\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` *`mysqld_service_name`* or `NET START` *`mysqld_service_name`* command from the command line after installing `mysqld` as a service to see any error messages regarding the starting of the MySQL server as a service. See Section 5.3.8, “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 8.3` and `C:\Program Files\MySQL\MySQL Server 8.3\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 8.3`, 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 8.3
# 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 5.3.2, “Creating an Option File”.

- If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service, and then configure MySQL using MySQL Configurator, 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.

5.5 Windows Postinstallation Procedures

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

- MySQL Configurator: Used to configure the MySQL server.
- 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 Configurator can 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 configured MySQL using the MySQL Configurator, you may have already assigned a password to the initial `root` account. (See Section 5.2, “Configuration: Using MySQL Configurator”.) 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 5.3.5, “Starting the Server for the First Time”). You can also set up a MySQL service that runs automatically when Windows starts (see Section 5.3.8, “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 the MSI, the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.3`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.3"
```

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 5.3.7, “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 `mysqlshow` to see what databases exist:

```
C:\> bin\mysqlshow
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

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 configured MySQL using MySQL Configurator, 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:

```
C:\> bin\mysqlshow -u root -p
Enter password: (enter root password here)
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

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 `mysql` program to select information from a table in the `mysql` database:

```
C:\> bin\mysql -e "SELECT User, Host, plugin FROM mysql.user" mysql
+------+-----------+-----------------------+
| User | Host      | plugin                |
+------+-----------+-----------------------+
| root | localhost | caching_sha2_password |
+------+-----------+-----------------------+
```

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

5.6 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

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 `mysql` user to own the MySQL directory and data. You can do this through the `Directory Utility`, and the `mysql` 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 8.3.0 it might be named `mysql-8.3.0-macos-10.13-x86_64.pkg`.

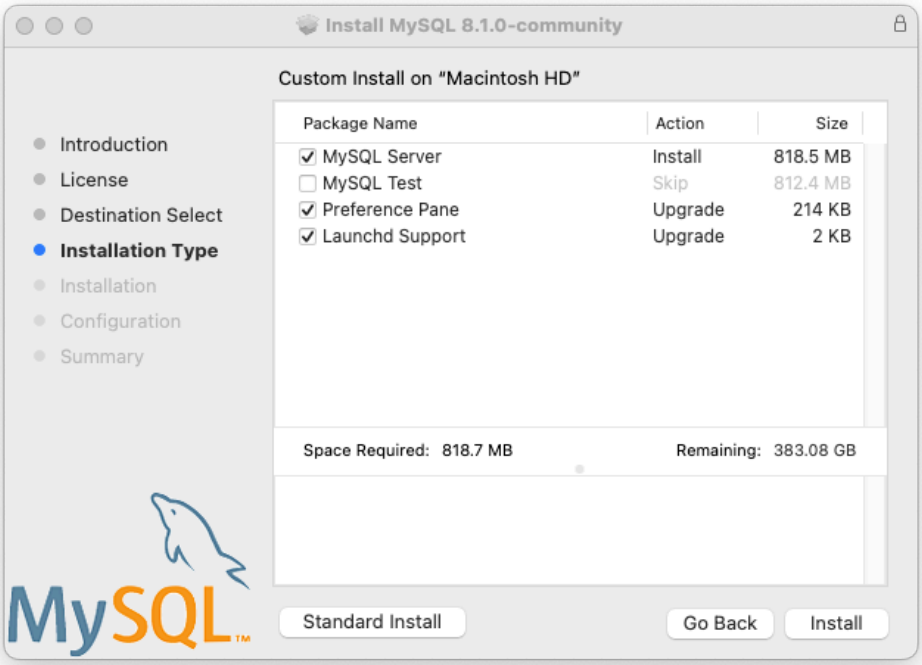
2. The initial wizard introduction screen references the MySQL server version to install. Click **Continue** to begin the installation.

   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: 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 8.3 password mechanism is `caching_sha2_password` (Strong), and this step allows you to change it to `mysql_native_password` (Legacy).

   Figure 6.2 MySQL Package Installer Wizard: Choose a Password Encryption Type

   Choosing the legacy password mechanism, which is deprecated, 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.

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.

When installing using the package installer, the files are installed into a directory within `/usr/local` matching the name of the installation version and platform. For example, the installer file `mysql-8.3.0-macos10.15-x86_64.dmg` installs MySQL into `/usr/local/mysql-8.3.0-macos10.15-x86_64/` with a symlink to `/usr/local/mysql`. The following table shows the layout of this MySQL installation directory.

> **Note**
>
> The macOS installation process does not create nor install a sample `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
`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
<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>         <string>_mysql</string>
    <key>ExitTimeOut</key>       <integer>600</integer>
    <key>Program</key>           <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/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>
        </array>
    <key>WorkingDirectory</key>  <string>/usr/local/mysql</string>
```

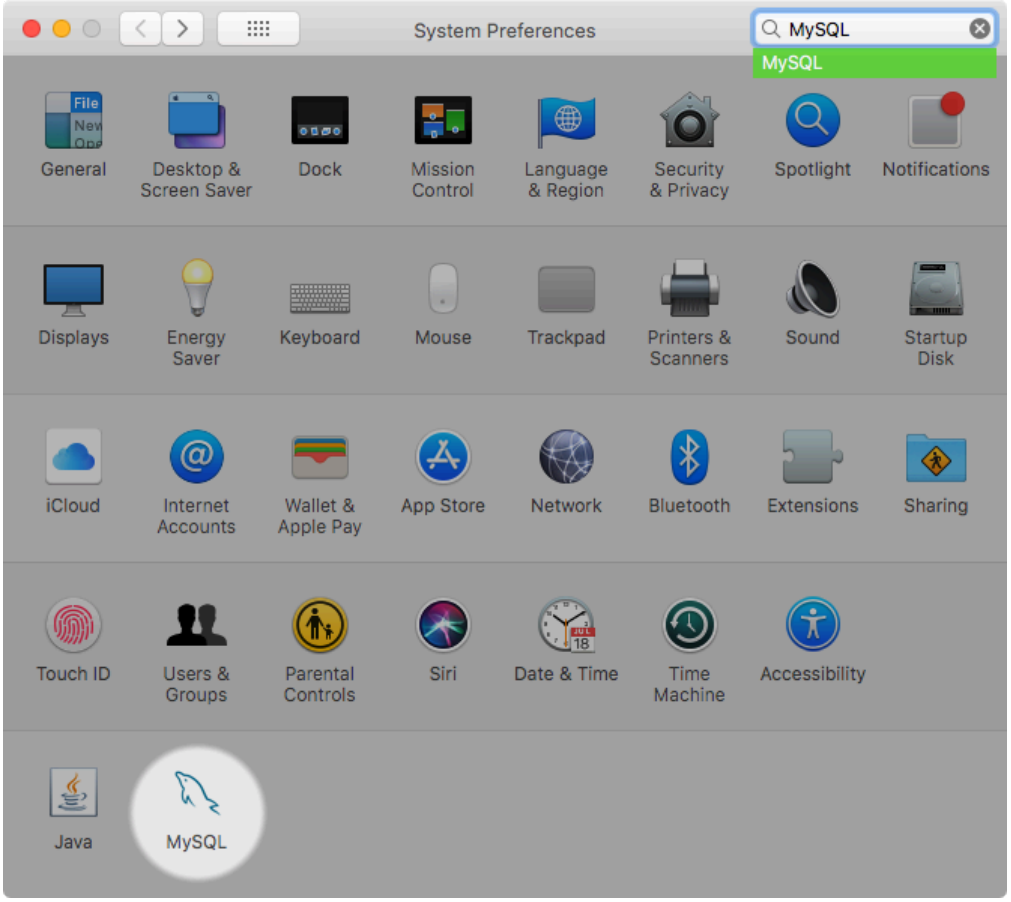
```
</dict>
</plist>
```

> **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:

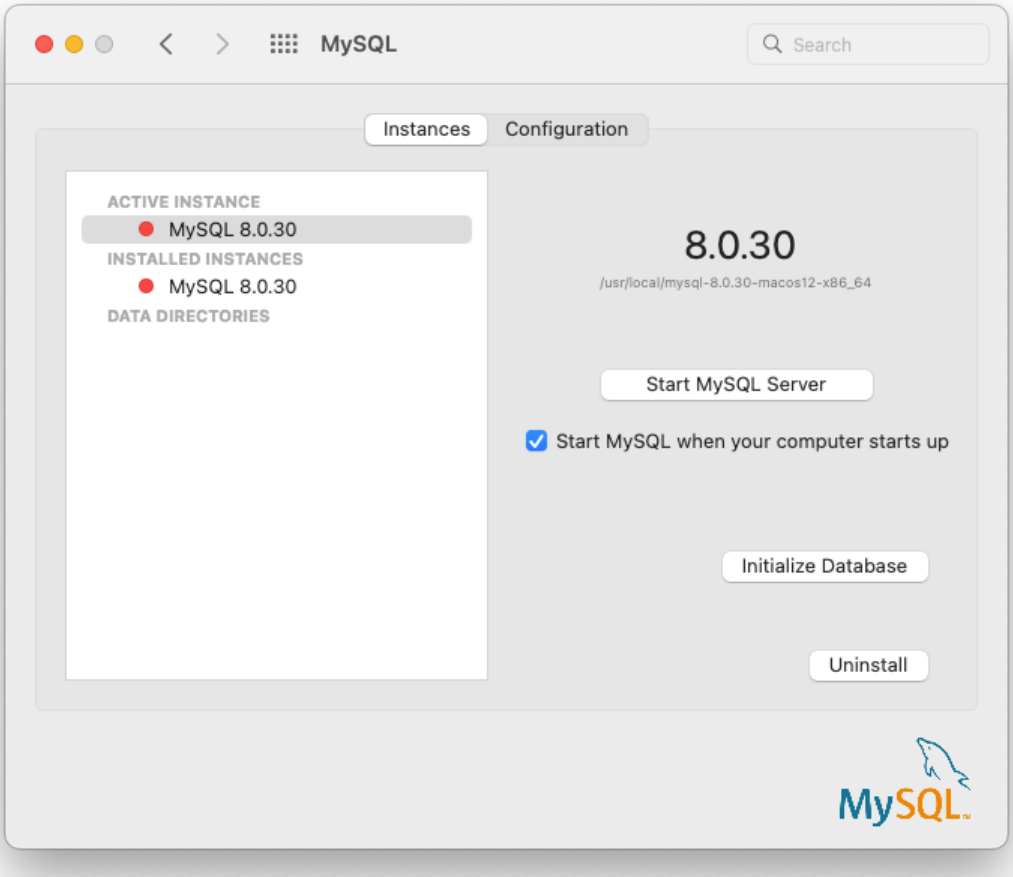
- Open macOS system preferences and select the MySQL preference panel, and then execute **Start MySQL Server**.

Figure 6.3 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 6.4 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.d
<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>         <string>_mysql</string>
    <key>ExitTimeOut</key>       <integer>600</integer>
    <key>Program</key>           <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/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>
        </array>
    <key>WorkingDirectory</key>  <string>/usr/local/mysql</string>
</dict>
</plist>
```

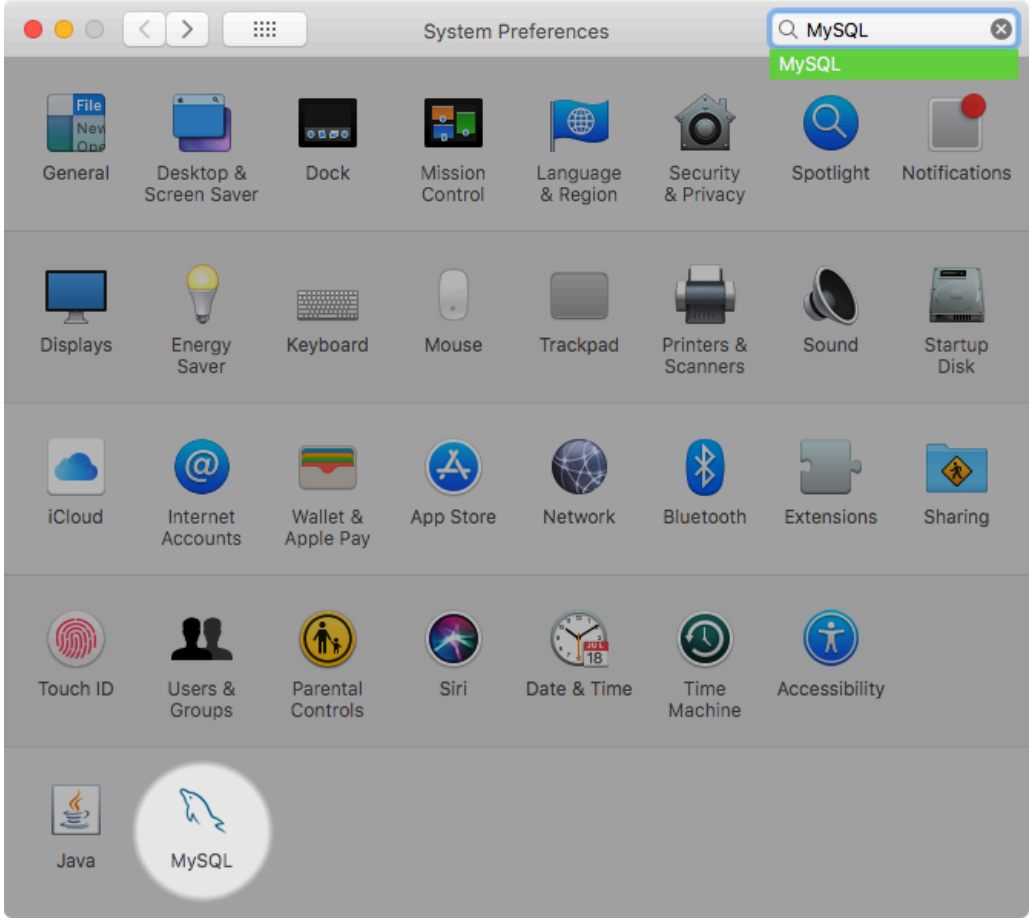
In this case, the `basedir`, `datadir`, `plugin_dir`, `log_error`, `pid_file`, `keyring_file_data`, and `--early-plugin-load` options were removed from the default plist *ProgramArguments* definition, which you might have defined in `my.cnf` instead.

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.

Figure 6.5 MySQL Preference Pane: Location



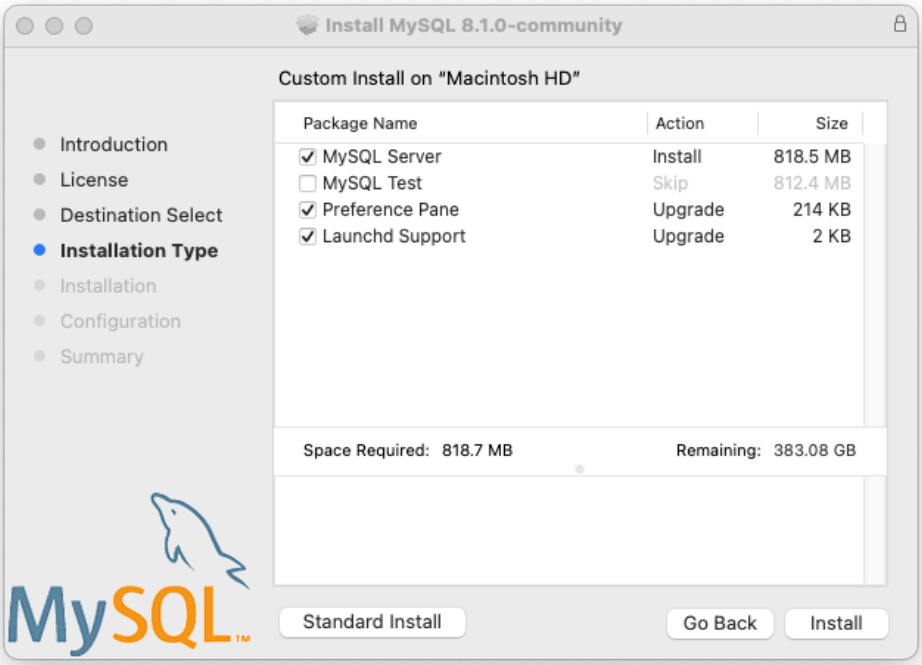
The MySQL preference pane is installed with the same DMG file that installs MySQL Server. Typically it is installed with MySQL Server but it can be installed by itself too.

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 6.6 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 6.7 MySQL Preference Pane: Instances

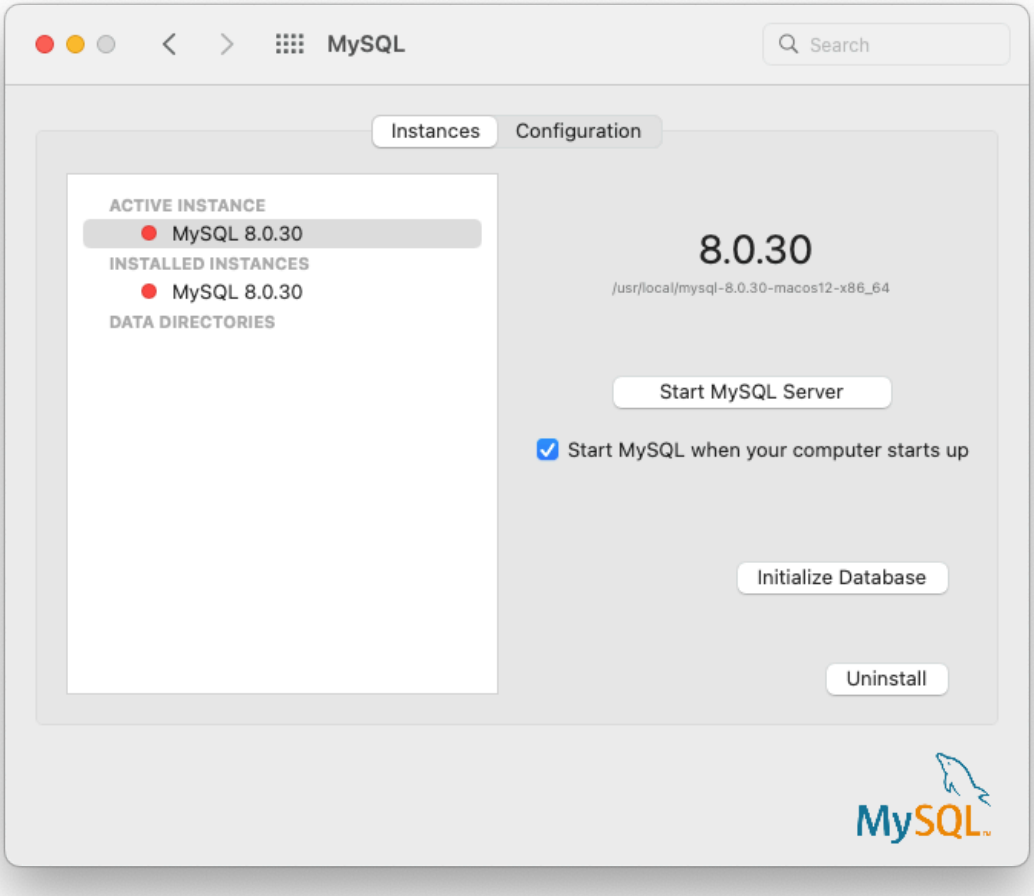
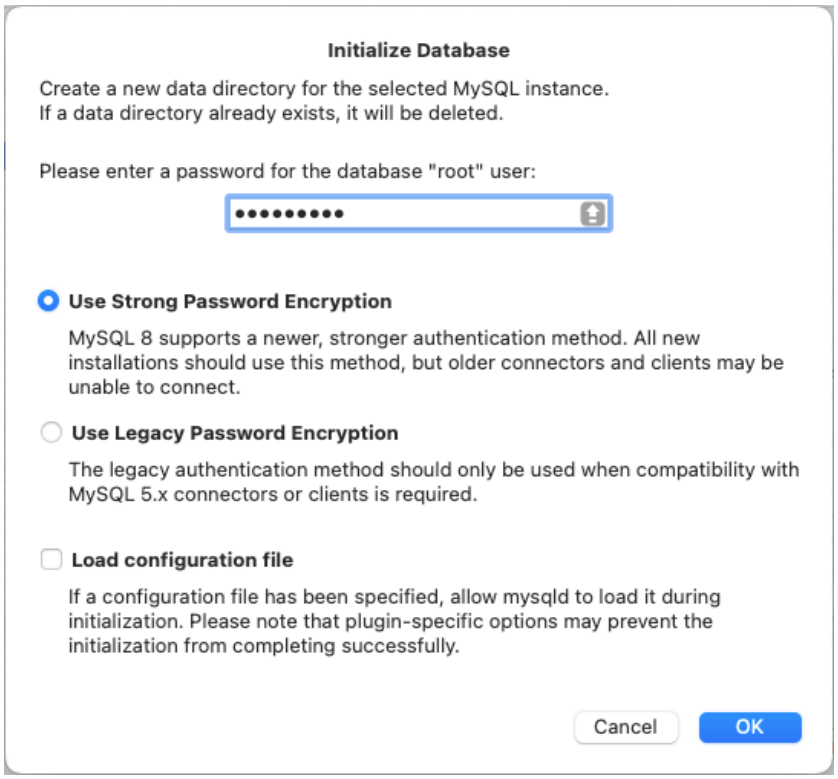
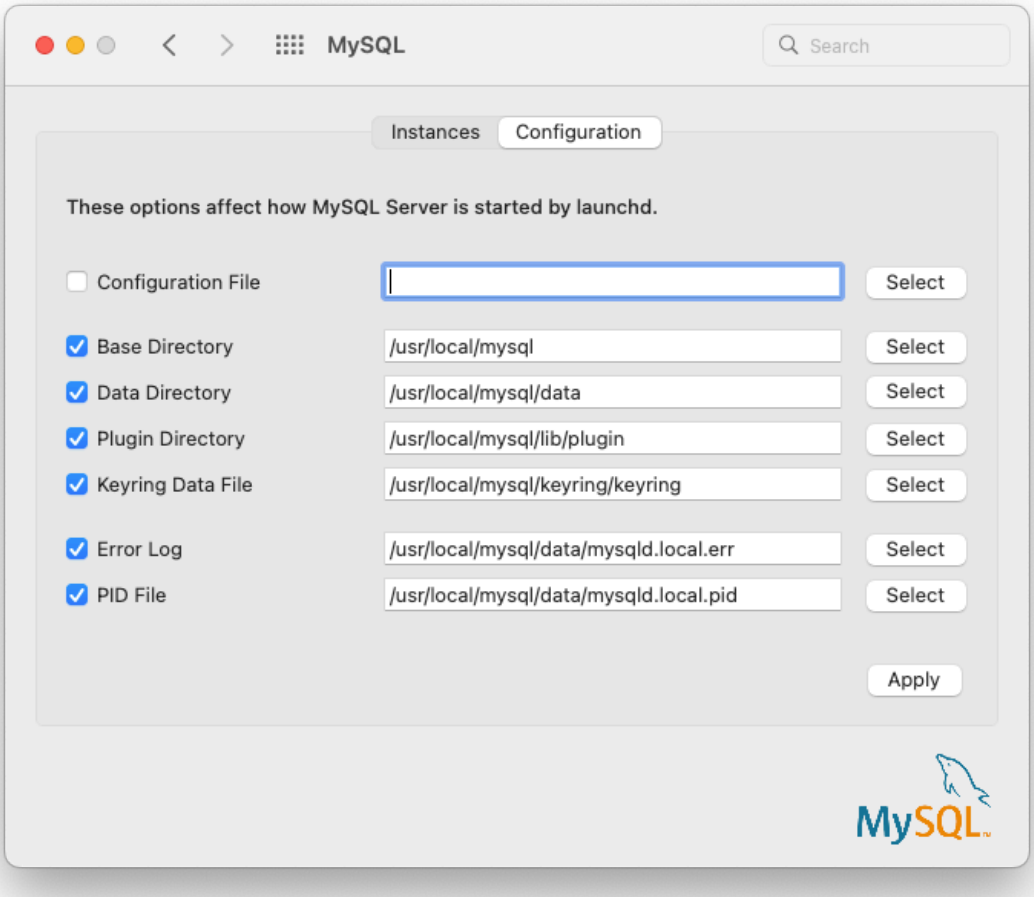


Figure 6.8 MySQL Preference Pane: Initialize Database



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

Figure 6.9 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

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

Type	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, `mysql.server`, can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. You can install it

> as `/etc/init.d/mysql` for automatic MySQL startup and shutdown. See mysql.server — MySQL 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.

> **Note**
>
> Repository setup RPM file names begin with `mysql80-community` to highlight the default active MySQL subrepository, which is MySQL 8.0 today. They also install subrepositories for the MySQL innovation track, which allows installing and upgrading the MySQL 8.3 innovation release.

Steps for a Fresh Installation of MySQL

Follow these steps to choose and install the latest MySQL products:

Adding[1] the MySQL Yum Repository

Add the MySQL Yum repository to your system's repository list. This is typically a one-time operation that's performed by installing the RPM provided by MySQL. Follow these steps:

a. Download it from the 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. The package file format is:

```
mysql80-community-release-{platform}-{version-number}.noarch.rpm
```

- *`mysql80`*: Indicates the MySQL version that's enabled by default. In this case, MySQL 8.0 is enabled by default and MySQL 8.3 (the innovation track) is available but disabled by default.
- *`{platform}`*: The platform code, such as el7, el8, el9, fc37, fc38, or fc39. The 'el' represents Enterprise Linux, 'fc' for Fedora, and it ends with the platform's base version number.
- *`{version-number}`*: Version of the MySQL repository configuration RPM as they do receive occasional updates.

Install the RPM you downloaded for your system, for example:

```
$> sudo yum localinstall mysql80-community-release-{platform}-{version-number}.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 and enabled by the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
$> yum repolist enabled | grep mysql.*-community
```

Example output:

```
mysql-connectors-community       MySQL Connectors Community
mysql-tools-community            MySQL Tools Community
mysql80-community                MySQL 8.0 Community Server
```

This also demonstrates that the latest bugfix or LTS MySQL version is enabled by default. Methods to choose a different release series, such as the innovation track (which today is 8.3), are described below.

> **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 bugfix series (currently MySQL 8.0) is selected for installation by default. If this is what you want, you can skip to the next step, Installing MySQL.

Within the MySQL Yum repository, each MySQL Community Server release series is hosted in a different subrepository. The subrepository for the latest bugfix series (currently MySQL 8.0) is enabled by default, and the subrepositories for all other series' (for example, the MySQL Innovation series) are disabled by default. Use this command to see all available MySQL-related subrepositories (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
$> yum repolist all | grep mysql
```

Example output:

```
mysql-connectors-community                 MySQL Connectors Community   enabled
mysql-connectors-community-debuginfo       MySQL Connectors Community - disabled
mysql-connectors-community-source          MySQL Connectors Community - disabled
mysql-innovation-community                 MySQL Innovation Release Com disabled
mysql-innovation-community-debuginfo       MySQL Innovation Release Com disabled
mysql-innovation-community-source          MySQL Innovation Release Com disabled
mysql-tools-community                      MySQL Tools Community        enabled
mysql-tools-community-debuginfo            MySQL Tools Community - Debu disabled
mysql-tools-community-source               MySQL Tools Community - Sour disabled
mysql-tools-innovation-community           MySQL Tools Innovation Commu disabled
mysql-tools-innovation-community-debuginfo MySQL Tools Innovation Commu disabled
mysql-tools-innovation-community-source    MySQL Tools Innovation Commu disabled
mysql80-community                          MySQL 8.0 Community Server   enabled
mysql80-community-debuginfo                MySQL 8.0 Community Server - disabled
mysql80-community-source                   MySQL 8.0 Community Server - disabled
```

To install the latest release from a specific series other than the latest bugfix series, disable the bug subrepository for the latest bugfix series and enable the subrepository for the specific series before running the installation command. If your platform supports the `yum-config-manager` or `dnf config-manager` command, you can do that by issuing the following commands to disable the subrepository for the 8.0 series and enable the one for the innovation track:

```
$> sudo yum-config-manager --disable mysql80-community
```

```
$> sudo yum-config-manager --enable mysql-innovation-community
```

For dnf-enabled platforms:

```
$> sudo dnf config-manager --disable mysql80-community
$> sudo dnf config-manager --enable mysql-innovation-community
```

Instead of using the config-manager commands you can manually edit the `/etc/yum.repos.d/mysql-community.repo` file by toggling the `enabled` option. For example, a typical default entry for EL8:

```
[mysql80-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/8/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql-2023
       file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql-2022
```

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 from the MySQL innovation track, make sure you have `enabled=0` for the MySQL 8.0 subrepository entries and have `enabled=1` for the innovation entries:

```
[mysql-innovation-community]
name=MySQL Innovation Release Community Server
baseurl=http://repo.mysql.com/yum/mysql-innovation-community/el/8/$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql-2023
```

You should only enable subrepository for one release series at any time.

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
```

3 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
```

4 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`).

5. 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 `systemctl` (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
```

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.

Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository

To replace third-party distributions of MySQL that were installed from the supported Linux platforms' native software repositories with the latest bug release (from the MySQL 8.0 series currently) from the MySQL Yum repository, follow these steps:

Replacing a Native Third-Party Distribution of MySQL

If you have installed a third-party distribution of MySQL from a native software repository (that is, a software repository provided by your own Linux distribution), follow these steps:

1. Backing Up Your Database

To avoid loss of data, always back up your database before trying to replace your MySQL installation using the MySQL Yum repository. See Backup and Recovery, on how to back up your database.

2. Adding the MySQL Yum Repository

Add the MySQL Yum repository to your system's repository list by following the instructions given in Adding the MySQL Yum Repository.

3. Replacing the Native Third-Party Distribution by a Yum Update or a DNF Upgrade

By design, the MySQL Yum repository replaces your native, third-party MySQL with the latest bugfix release from the MySQL Yum repository when you perform a `yum update` command (or

`dnf upgrade` for dnf-enabled systems) on the system, or a `yum update mysql-server` (or `dnf upgrade mysql-server` for dnf-enabled systems).

After updating MySQL using the Yum repository, applications compiled with older versions of the shared client libraries should continue to work. However, *if you want to recompile applications and dynamically link them with the updated libraries*, see Upgrading the Shared Client Libraries, for some special considerations.

> **Note**
>
> *For EL7-based platforms:* See Compatibility Information for EL7-based platforms [105].

7.2 Installing MySQL on Linux Using the MySQL APT Repository

Steps for a Fresh Installation of MySQL

> **Note**
>
> The following instructions assume that no versions of MySQL (whether distributed by Oracle or other parties) have already been installed on your system; if that is not the case, follow the instructions given in Replacing a Native Distribution of MySQL Using the MySQL APT Repository or Replacing a MySQL Server Installed by a Direct deb Package Download instead.

Adding the MySQL Yum Repository. First, add the MySQL APT repository to your system's software repository list. Follow these steps:

1. Go to the download page for the MySQL APT repository at https://dev.mysql.com/downloads/repo/apt/.

2. Select and download the release package for your Linux distribution.

   Although this is not required for each update, it does update MySQL repository information to include the current information. For example, `mysql-apt-config_0.8.26-1_all.deb` is the first APT repository configuration file that adds the innovation release track that begins with MySQL 8.1.

3. Install the downloaded release package with the following command, replacing *`version-specific-package-name`* with the name of the downloaded package (preceded by its path, if you are not running the command inside the folder where the package is):

   ```
   $> sudo dpkg -i /PATH/version-specific-package-name.deb
   ```

   For example, for version *`w.x.y-z`* of the package, the command is:

   ```
   $> sudo dpkg -i mysql-apt-config_w.x.y-z_all.deb
   ```

   Note that the same package works on all supported Debian and Ubuntu platforms.

4. During the installation of the package, you will be asked to choose the versions of the MySQL server and other components (for example, the MySQL Workbench) that you want to install. If you are not sure which version to choose, do not change the default options selected for you. You can also choose **none** if you do not want a particular component to be installed. After making the choices for all components, choose **Ok** to finish the configuration and installation of the release package.

   > **Note**
   >
   > The innovation track, which begins with MySQL 8.1, includes "-innovation-" in the component name.

You can always change your choices for the versions later; see Selecting a Major Release Version for instructions.

5. Update package information from the MySQL APT repository with the following command (*this step is mandatory*):

```
$> sudo apt-get update
```

Instead of using the release package, you can also add and configure the MySQL APT repository manually; see Appendix A: Adding and Configuring the MySQL APT Repository Manually for details.

> **Note**
>
> Once the MySQL APT repository is enabled on your system, you will no longer be able to install any MySQL packages from your platform's native software repositories until the MySQL APT repository is disabled.

> **Note**
>
> Once the MySQL APT repository is enabled on your system, any system-wide upgrade by the `apt-get upgrade` command will automatically upgrade the MySQL packages on your system and also replace any native MySQL packages you installed from your Linux distribution's software repository, if APT finds replacements for them from within the MySQL APT repository.

Selecting a Major Release Version

By default, all installations and upgrades for your MySQL server and the other required components come from the release series of the major version you have selected during the installation of the configuration package (see Adding the MySQL Yum Repository). However, you can switch to another supported major release series at any time by reconfiguring the configuration package you have installed. Use the following command:

```
$> sudo dpkg-reconfigure mysql-apt-config
```

A dialogue box then asks you to choose the major release version you want. Make your selection and choose **Ok**. After returning to the command prompt, update package information from the MySQL APT repository with this command:

```
$> sudo apt-get update
```

The latest version in the selected series will then be installed when you use the `apt-get install` command next time.

You can use the same method to change the version for any other MySQL component you want to install with the MySQL APT repository.

Installing MySQL with APT

Install MySQL by the following command:

```
$> sudo apt-get install mysql-server
```

This installs the package for the MySQL server, as well as the packages for the client and for the database common files.

During the installation, you are asked to supply a password for the root user for your MySQL 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 will be authenticated by Socket Peer-Credential Pluggable Authentication for connections using a Unix socket file. You can set the root password later using the program `mysql_secure_installation`.

Starting and Stopping the MySQL Server

The MySQL server is started automatically after installation. You can check the status of the MySQL server with the following command:

```
$> systemctl status mysql
```

If the operating system is systemd enabled, standard `systemctl` (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 `mysql` service is enabled by default, and it starts at system reboot. See Section 7.9, “Managing MySQL Server with systemd” for additional information.

> **Note**
>
> A few third-party native repository packages that have dependencies on the native MySQL packages may not work with the MySQL APT repository packages and should not be used together with them; these include akonadi-backend-mysql, handlersocket-mysql-5.5, and zoneminder.

Installing Additional MySQL Products and Components with APT

You can use APT to install individual components of MySQL from the MySQL APT repository. Assuming you already have the MySQL APT repository on your system's repository list (see Adding the MySQL Yum Repository for instructions), first, use the following command to get the latest package information from the MySQL APT repository:

```
$> sudo apt-get update
```

Install any packages of your choice with the following command, replacing *`package-name`* with name of the package to install:

```
$> sudo apt-get install package-name
```

For example, to install the MySQL Workbench:

```
$> sudo apt-get install mysql-workbench-community
```

To install the shared client libraries:

```
$> sudo apt-get install libmysqlclient21
```

Installing MySQL from Source with the MySQL APT Repository

> **Note**
>
> This feature is only supported on 64-bit systems.

You can download the source code for MySQL and build it using the MySQL APT Repository:

1. Add the MySQL APT repository to your system's repository list and choose the major release series you want (see Adding the MySQL Yum Repository for instructions).
2. Update package information from the MySQL APT repository with the following command (*this step is mandatory*):

```
$> sudo apt-get update
```

3. Install packages that the build process depends on:

```
$> sudo apt-get build-dep mysql-server
```

4. Download the source code for the major components of MySQL and then build them (run this command in the folder in which you want the downloaded files and the builds to be located):

```
$> apt-get source -b mysql-server
```

`deb` packages for installing the various MySQL components are created.

5. Pick the `deb` packages for the MySQL components you need and install them with the command:

```
$> sudo dpkg -i package-name.deb
```

Notice that dependency relationships exist among the MySQL packages. 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) with the following steps:

- Preconfigure the MySQL server package with the following command:

```
$> sudo dpkg-preconfigure mysql-community-server_version-and-platform-specific-part.deb
```

  You will be asked to provide a password for the root user for your MySQL installation; see important information on root password given in Installing MySQL with APT above. You might also be asked other questions regarding the installation.

- Install the required packages with a single command:

```
$> sudo dpkg -i mysql-{common,community-client,client,community-server,server}_*.deb
```

- If you are being warned of unmet dependencies by `dpkg`, 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`

See also information given in Starting and Stopping the MySQL Server.

Upgrading MySQL with the MySQL APT Repository

Notes

- 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 following instructions assume that MySQL has been installed on your system using the MySQL APT repository; if that is not the case, follow the instructions given in Replacing a Native Distribution of MySQL Using the MySQL APT Repository or Replacing a MySQL Server Installed by a Direct deb Package Download instead. Also notice that you cannot use the MySQL APT repository to upgrade a distribution of MySQL that you have

> installed from a nonnative software repository (for example, from MariaDB or Percona).

Use the MySQL APT repository to perform an in-place upgrade for your MySQL installation (that is, replacing the old version and then running the new version using the old data files) by following these steps:

1. Make sure you already have the MySQL APT repository on your system's repository list (see Adding the MySQL Yum Repository for instructions).

2. Make sure you have the most up-to-date package information on the MySQL APT repository by running:

   ```
   $> sudo apt-get update
   ```

3. Note that, by default, the MySQL APT repository will update MySQL to the release series you have selected when you were adding the MySQL APT repository to your system. If you want to upgrade to another release series, select it by following the steps given in Selecting a Major Release Version.

   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 a newer series, upgrade to MySQL 5.7 first before upgrading to 8.0.

   > **Important**
   >
   > - For important information about upgrading from MySQL 5.6 to 5.7, see Upgrading from MySQL 5.6 to 5.7.
   > - For important information about upgrading from MySQL 5.7 to 8.0, see Upgrading from MySQL 5.7 to 8.0.
   > - In-place downgrading of MySQL is not supported by the MySQL APT repository. Follow the instructions in Chapter 11, *Downgrading MySQL*.

4. Upgrade MySQL by the following command:

   ```
   $> sudo apt-get install mysql-server
   ```

   The MySQL server, client, and the database common files are upgraded if newer versions are available. To upgrade any other MySQL package, use the same `apt-get install` command and supply the name for the package you want to upgrade:

   ```
   $> sudo apt-get install package-name
   ```

   To see the names of the packages you have installed from the MySQL APT repository, use the following command:

   ```
   $> dpkg -l | grep mysql | grep ii
   ```

   > **Note**
   >
   > If you perform a system-wide upgrade using `apt-get upgrade`, only the MySQL library and development packages are upgraded with newer versions (if available). To upgrade other components including the server, client, test suite, etc., use the `apt-get install` command.

5. The MySQL server always restarts after an update by APT. 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; see mysql_upgrade — Deprecated; Performs No Tasks and Exits with Status 0 for details. As of MySQL 8.0.16, this step is not required, as the server performs all tasks previously handled by `mysql_upgrade`.

Replacing a Native Distribution of MySQL Using the MySQL APT Repository

Variants and forks of MySQL are distributed by different parties through their own software repositories or download sites. You can replace a native distribution of MySQL installed from your Linux platform's software repository with a distribution from the MySQL APT repository in a few steps.

> **Note**
>
> The MySQL APT repository can only replace distributions of MySQL maintained and distributed by Debian or Ubuntu. It cannot replace any MySQL forks found either inside or outside of the distributions' native repositories. To replace such MySQL forks, you have to uninstall them first before you install MySQL using the MySQL APT repository. Follow the instructions for uninstallation from the forks' distributors and, before you proceed, make sure you back up your data and you know how to restore them to a new server.

> **Warning**
>
> A few third-party native repository packages that have dependencies on the native MySQL packages may not work with the MySQL APT repository packages and should not be used together with them; these include akonadi-backend-mysql, handlersocket-mysql-5.5, and zoneminder.

1 Backing Up Your Database

To avoid loss of data, always back up your database before trying to replace your MySQL installation using the MySQL APT repository. See Backup and Recovery for instructions.

2 Adding the MySQL APT Repository and Selecting a Release Series

Add the MySQL APT repository to your system's repository list and select the release series you want by following the instructions given in Adding the MySQL Yum Repository.

3 Replacing the Native Distribution by an APT Update

By design, the MySQL APT repository replaces your native distribution of MySQL when you perform upgrades on the MySQL packages. To perform the upgrades, follow the same instructions given in Step 4 in Upgrading MySQL with the MySQL APT Repository.

> **Warning**
>
> Once the native distribution of MySQL has been replaced using the MySQL APT repository, purging the old MySQL packages from the native repository using the `apt-get purge`, `apt-get remove --purge`, or `dpkg -P` command might impact the newly installed MySQL server in various ways. Therefore, *do not purge the old MySQL packages from the native repository packages*.

Replacing a MySQL Server Installed by a Direct deb Package Download

`deb` packages from MySQL for installing the MySQL server and its components can either be downloaded from the MySQL Developer Zone's MySQL Download page or from the MySQL APT repository. The `deb` packages from the two sources are different, and they install and configure MySQL in different ways.

If you have installed MySQL with the MySQL Developer Zone's `deb` packages and now want to replace the installation using the ones from the MySQL APT repository, follow these steps:

1. Back up your database. See Backup and Recovery for instructions.
2. Follow the steps given previously for adding the MySQL APT repository.

3. Remove the old installation of MySQL by running:

```
$> sudo dpkg -P mysql
```

4. Install MySQL from the MySQL APT repository:

```
$> sudo apt-get install mysql-server
```

5. If needed, restore the data on the new MySQL installation. See Backup and Recovery for instructions.

Removing MySQL with APT

To uninstall the MySQL server and the related components that have been installed using the MySQL APT repository, first, remove the MySQL server using the following command:

```
$> sudo apt-get remove mysql-server
```

Then, remove any other software that was installed automatically with the MySQL server:

```
$> sudo apt-get autoremove
```

To uninstall other components, use the following command, replacing **`package-name`** with the name of the package of the component you want to remove:

```
$> sudo apt-get remove package-name
```

To see a list of packages you have installed from the MySQL APT repository, use the following command:

```
$> dpkg -l | grep mysql | grep ii
```

Special Notes on Upgrading the Shared Client Libraries

You can install the shared client libraries from MySQL APT repository by the following command (see Installing Additional MySQL Products and Components with APT for more details):

```
$> sudo apt-get install libmysqlclient21
```

If you already have the shared client libraries installed from you Linux platform's software repository, it can be updated by the MySQL APT repository with its own package by using the same command (see Replacing the Native Distribution by an APT Update for more details).

After updating MySQL using the APT 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, any applications compiled using the updated, newer shared libraries might require those updated libraries on systems where the applications are deployed. If those libraries are not in place, the applications requiring the shared libraries might fail. Therefore, it is recommended that the packages for the shared libraries from MySQL be deployed on those systems. You can do this by adding the MySQL APT repository to the systems (see Adding the MySQL Yum Repository) and installing the latest shared client libraries using the command given at the beginning of this section.

Installing MySQL NDB Cluster Using the APT Repository

Notes

- The MySQL APT repository supports installation of MySQL NDB Cluster on Debian and Ubuntu systems. For methods to install NDB Cluster on other Debian-based systems, see Installing NDB Cluster Using .deb Files.
- If you already have the MySQL server or MySQL NDB Cluster installed on your system, make sure it is stopped and you have your data and configuration files backed up before proceeding.

1. Adding the MySQL APT Repository for MySQL NDB Cluster

Follow the steps in Adding the MySQL Yum Repository to add the MySQL APT repository to your system's repository list. During the installation process of the configuration package, when you are asked which MySQL product you want to configure, choose “MySQL Server & Cluster”; when asked which version you wish to receive, choose “mysql-cluster-*x*.*y*.” After returning to the command prompt, go to Step 2 below.

If you already have the configuration package installed on your system, make sure it is up-to-date by running the following command:

```
$> sudo apt-get install mysql-apt-config
```

Then, use the same method described in Selecting a Major Release Version to select MySQL NDB Cluster for installation. When you are asked which MySQL product you want to configure, choose “MySQL Server & Cluster”; when asked which version you wish to receive, choose “mysql-cluster-*x*.*y*.” After returning to the command prompt, update package information from the MySQL APT repository with this command:

```
$> sudo apt-get update
```

2. Installing MySQL NDB Cluster

For a minimal installation of MySQL NDB Cluster, follow these steps:

- Install the components for SQL nodes:

  ```
  $> sudo apt-get install mysql-cluster-community-server
  ```

  You will be asked to provide a password for the root user for your SQL node; see important information on the root password given in Installing MySQL with APT above. You might also be asked other questions regarding the installation.

- Install the executables for management nodes:

  ```
  $> sudo apt-get install mysql-cluster-community-management-server
  ```

- Install the executables for data nodes:

  ```
  $> sudo apt-get install mysql-cluster-community-data-node
  ```

3. Configuring and Starting MySQL NDB Cluster

See Initial Configuration of NDB Cluster on how to configure MySQL NDB Cluster and Initial Startup of NDB Cluster on how to start it for the first time. When following those instructions, adjust them according to the following details regarding the SQL nodes of your NDB Cluster installation:

- 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 `/var/lib/mysql`

Installing Additional MySQL NDB Cluster Products and Components

You can use APT to install individual components and additional products of MySQL NDB Cluster from the MySQL APT repository. To do that, assuming you already have the MySQL APT repository on your system's repository list (see Adding the MySQL Yum Repository for MySQL NDB Cluster), follow the same steps given in Installing Additional MySQL Products and Components with APT.

> **Note**
>
> *Known issue:* Currently, not all components required for running the MySQL NDB Cluster test suite are installed automatically when you install the test suite package (`mysql-cluster-community-test`). Install the following packages with `apt-get install` before you run the test suite:
>
> - `mysql-cluster-community-auto-installer`
> - `mysql-cluster-community-management-server`
> - `mysql-cluster-community-data-node`
> - `mysql-cluster-community-memcached`
> - `mysql-cluster-community-java`
> - `ndbclient-dev`

Appendix A: Adding and Configuring the MySQL APT Repository Manually

Here are the steps for adding manually the MySQL APT repository to your system's software repository list and configuring it, without using the release packages provided by MySQL:

- Download the MySQL GPG Public key (see Section 2.4.2, “Signature Checking Using GnuPG” on how to do that) and save it to a file, without adding any spaces or special characters. Then, add the key to your system's GPG keyring with the following command:

```
$> sudo apt-key add path/to/signature-file
```

- Alternatively, you can download the GPG key to your APT keyring directly using the apt-key utility:

```
$> sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29
```

> **Note**
>
> The KeyID for MySQL 8.0.28 release packages and higher is `3A79BD29`, as shown above. For earlier MySQL releases, the keyID is `5072E1F5`. Using an incorrect key can cause a key verification error.

- Create a file named `/etc/apt/sources.list.d/mysql.list`, and put into it repository entries in the following format (this is not a command to execute):

```
deb http://repo.mysql.com/apt/{debian|ubuntu}/ {buster|bionic} {mysql-5.7|mysql-8.0|workbench-8.0|con
```

  Pick the relevant options for your repository set up:

  - Choose “debian” or “ubuntu” according to your platform.
  - Choose the appropriate version name for the version of your system; examples include “bullseye” (for Debian 11) and “jammy” (for Ubuntu 22.04).
  - For installing MySQL server, client, and database common files, choose “mysql-5.7” or “mysql-8.0” according to the MySQL version you want. To switch to another release series later, come back and adjust the entry with your new choice.

> **Note**
>
> If you already have a version of MySQL installed on your system, do not choose a lower version at this step, or it might result in an unsupported downgrade operation.

- For installing components like MySQL Workbench or MySQL Connector/Python, create a single entry for each of them, specifying respectively “workbench-8.0” or “connector-python-8.0” at the end of each entry.

For example, on the Ubuntu 18.04 platform use these lines in your `mysql.list` files to install MySQL 8.0 and MySQL Connector/Python from the MySQL APT repository:

```
deb http://repo.mysql.com/apt/ubuntu/ bionic mysql-8.0
deb http://repo.mysql.com/apt/ubuntu/ bionic connector-python-8.0
```

- Use the following command to get the most up-to-date package information from the MySQL APT repository:

```
$> sudo apt-get update
```

You have configured your system to use the MySQL APT repository and are now ready to continue with Installing MySQL with APT or Installing Additional MySQL Products and Components with APT.

7.3 Using the MySQL SLES Repository

Adding 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.

Add or update the official MySQL SLES repository for your system's repository list:

> **Note**
>
> The "`mysql80`" in the repository file name indicates that the MySQL 8.0 series is enabled by default, but it also includes the optional "Innovation" track that starts with MySQL 8.1.

New MySQL Repository Installation

If the MySQL repository is not yet present on the system then:

1. Go to the download page for MySQL SLES repository at https://dev.mysql.com/downloads/repo/suse/.
2. Select and download the release package for your SLES version.
3. Install the downloaded release package with the following command, replacing *`package-name`* with the name of the downloaded package:

   ```
   $> sudo rpm -Uvh package-name.rpm
   ```

   For example, to install the SLES 15 package where # indicates the release number within a version such as `15-1`:

   ```
   $> sudo rpm -Uvh mysql80-community-release-sl15-#.noarch.rpm
   ```

Update an Existing MySQL Repository Installation

If an older version is already present then update it:

- ```
  $> sudo zypper update mysql80-community-release
  ```

- Although this is not required for each MySQL release, it does update MySQL repository information to include the current information. For example, `mysql80-community-release-sl15-7.noarch.rpm` is the first SUSE 15 repository configuration file that adds the innovation release track that begins with MySQL 8.1. series.
  ```

Selecting a Release Series

Within the MySQL SLES repository, different release series of the MySQL Community Server are hosted in different subrepositories. The subrepository for the latest bugfix series (currently MySQL 8.0) is enabled by default, and the subrepositories for all other series are disabled. Use this command to see all of the subrepositories in the MySQL SLES repository, and to see which of them are enabled or disabled:

```
$> zypper repos | grep mysql.*community
```

The innovation track is available for SLES 15 as of MySQL 8.1, with entries such as `mysql-innovation-community`.

To install the latest release from a specific series, before running the installation command, make sure that the subrepository for the series you want is enabled and the subrepositories for other series are disabled. For example, on SLES 15, to disable the subrepositories for MySQL 8.0 server and tools, which are enabled by default, use the following:

```
$> sudo zypper modifyrepo -d mysql80-community
$> sudo zypper modifyrepo -d mysql-tools-community
```

Then, enable the subrepositories for the release series you want. For example, to enable the Innovation track on SLES 15:

```
$> sudo zypper modifyrepo -e mysql-innovation-community
$> sudo zypper modifyrepo -e mysql-tools-innovation-community
```

You should only enable a subrepository for one release series at any time.

Verify that the correct subrepositories have been enabled by running the following command and checking its output:

```
$> zypper repos -E | grep mysql.*community

 7 | mysql-connectors-community       | MySQL Connectors Community                  | Yes     | (r ) Ye
10 | mysql-innovation-community       | MySQL Innovation Release Community Server   | Yes     | (r ) Ye
16 | mysql-tools-innovation-community | MySQL Tools Innovation Community            | Yes     | ( p) Ye
```

After that, use the following command to refresh the repository information for the enabled subrepository:

```
$> sudo zypper refresh
```

Installing MySQL with Zypper

With the official MySQL repository enabled, install MySQL Server:

```
$> sudo zypper install mysql-community-server
```

This installs the package for the MySQL server, as well as other required packages.

Starting the MySQL Server

Start the MySQL server with the following command:

```
$> systemctl start mysql
```

You can check the status of the MySQL server with the following command:

```
$> systemctl status mysql
```

If the operating system is systemd enabled, standard `systemctl` (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 `mysql` service is enabled by default, and it starts at system reboot. See Section 7.9, “Managing MySQL Server with systemd” for additional information.

MySQL Server Initialization: When the server is started for the first time, the server is initialized, and the following happens (if the data directory of the server is empty when the initialization process begins):

- The SSL certificate and key files are generated in the data directory.
- The validate_password plugin 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/mysql/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**
  >
  > MySQL's validate_password plugin is installed by default. This will require 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.

You can stop the MySQL Server with the following command:

```
$> sudo systemctl stop mysql
```

Installing Additional MySQL Products and Components

You can install more components of MySQL. List subrepositories in the MySQL SLES repository with the following command:

```
$> zypper repos | grep mysql.*community
```

Use the following command to list the packages for the MySQL components available for a certain subrepository, changing *`subrepo-name`* to the name of the subrepository you are interested in :

```
$> zypper packages subrepo-name
```

Install any packages of your choice with the following command, replacing *`package-name`* with name of the package (you might need to enable first the subrepository for the package, using the same method for selecting a subrepository for a specific release series outlined in Selecting a Release Series):

```
$> sudo zypper install package-name
```

For example, to install the MySQL benchmark suite from the subrepository for the release series you have already enabled:

```
$> sudo zypper install mysql-community-bench
```

Upgrading MySQL with the MySQL SLES Repository

> **Note**
>
> - 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.

Use the MySQL SLES repository to perform an in-place update (that is, replacing the old version of the server and then running the new version using the old data files) for your MySQL installation by following these steps (they assume you have installed MySQL with the MySQL SLES repository; if that is not the case, following the instructions in Replacing MySQL Installed by an RPM from Other Sources instead):

1. Selecting a Target Series

During an update operation, by default, the MySQL SLES 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 bugfix series installation, such as 8.0, will *not* update to an innovation series, such as 8.3. To update to another release series, you need to 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, follow the general instructions given in Selecting a Release Series.

As a general rule, to upgrade from one release series to another, go to the next series rather than skipping a series.

> **Important**
>
> - For important information about upgrading from MySQL 5.7 to 8.0, see Upgrading from MySQL 5.7 to 8.0.
> - For important information about upgrading from MySQL 8.0 to an innovation series, see Upgrading from MySQL 8.0 to 8.3.
> - In-place downgrading of MySQL is not supported by the MySQL SLES repository. Follow the instructions in Chapter 11, *Downgrading MySQL*.

2. Upgrading MySQL

Upgrade MySQL and its components by the following command:

```
$> sudo zypper update mysql-community-server
```

Alternatively, you can update MySQL by telling Zypper to update everything on your system (this might take considerably more time):

```
$> sudo zypper update
```

You can also update a specific component only. Use the following command to list all the installed packages from the MySQL SLES repository:

```
$> zypper packages -i | grep mysql-.*community
```

After identifying the package name of the component of your choice, update the package with the following command, replacing *`package-name`* with the name of the package:

```
$> sudo zypper update package-name
```

Replacing MySQL Installed by an RPM from Other Sources

RPMs for installing the MySQL Community Server and its components can be downloaded from MySQL either from the MySQL Developer Zone, from the native software repository of SLES, or from the MySQL SLES repository. The RPMs from the those sources might be different, and they might install and configure MySQL in different ways.

If you have installed MySQL with RPMs from the MySQL Developer Zone or the native software repository of SLES and want to replace the installation using the RPM from the MySQL SLES repository, follow these steps:

1. Back up your database to avoid data loss. See Backup and Recovery on how to do that.

2. Stop your MySQL Server, if it is running. If the server is running as a service, you can stop it with the following command:

```
$> systemctl stop mysql
```

3. Follow the steps given for Adding the MySQL SLES Repository.

4. Follow the steps given for Selecting a Release Series.

5. Follow the steps given for Installing MySQL with Zypper. You will be asked if you want to replace the old packages with the new ones; for example:

```
Problem: mysql-community-server-5.6.22-2.sles11.x86_64 requires mysql-community-client = 5.6.22-2.sles1
  but this requirement cannot be provided uninstallable providers:
  mysql-community-client-5.6.22-2.sles11.x86_64[mysql56-community]
 Solution 1: replacement of mysql-client-5.5.31-0.7.10.x86_64 with mysql-community-client-5.6.22-2.sles
 Solution 2: do not install mysql-community-server-5.6.22-2.sles11.x86_64
 Solution 3: break mysql-community-server-5.6.22-2.sles11.x86_64 by ignoring some of its dependencies

Choose from above solutions by number or cancel [1/2/3/c] (c)
```

Choose the “replacement” option (“Solution 1” in the example) to finish your installation from the MySQL SLES repository.

Installing MySQL NDB Cluster Using the SLES Repository

- The following instructions assume that neither the MySQL Server nor MySQL NDB Cluster has already been installed on your system; if that is not the case, remove the MySQL Server or MySQL NDB Cluster, including all its executables, libraries, configuration files, log files, and data directories, before you continue. However there is no need to remove the release package you might have used to enable the MySQL SLES repository on your system.

- The NDB Cluster Auto-Installer package has a dependency on the `python2-crypto` and `python-paramiko` packages. Zypper can take care of this dependency if the Python repository has been enabled on your system.

Selecting the MySQL NDB Cluster Subrepository

Within the MySQL SLES repository, the MySQL Community Server and MySQL NDB Cluster are hosted in different subrepositories. By default, the subrepository for the latest bugfix series of the MySQL Server is enabled and the subrepository for MySQL NDB Cluster is disabled. To install NDB Cluster, disable the subrepository for the MySQL Server and enable the subrepository for NDB Cluster. For example, disable the subrepository for MySQL 8.0, which is enabled by default, with the following command:

```
$> sudo zypper modifyrepo -d mysql80-community
```

Then, enable the subrepository for MySQL NDB Cluster:

```
$> sudo zypper modifyrepo -e mysql-cluster-8.0-community
```

Verify that the correct subrepositories have been enabled by running the following command and checking its output:

```
$> zypper repos -E | grep mysql.*community
10 | mysql-cluster-8.0-community | MySQL Cluster 8.0 Community | Yes     | No
```

After that, use the following command to refresh the repository information for the enabled subrepository:

```
$> sudo zypper refresh
```

Installing MySQL NDB Cluster

For a minimal installation of MySQL NDB Cluster, follow these steps:

- Install the components for SQL nodes:

```
$> sudo zypper install mysql-cluster-community-server
```

  After the installation is completed, start and initialize the SQL node by following the steps given in Starting the MySQL Server.

  If you choose to initialize the data directory manually using the `mysqld --initialize` command (see Section 9.1, “Initializing the Data Directory” for details), a `root` password is going to be generated and stored in the SQL node's error log; see Starting the MySQL Server for how to find the password, and for a few things you need to know about it.

- Install the executables for management nodes:

```
$> sudo zypper install mysql-cluster-community-management-server
```

- Install the executables for data nodes:

```
$> sudo zypper install mysql-cluster-community-data-node
```

To install more NDB Cluster components, see Installing Additional MySQL Products and Components.

See Initial Configuration of NDB Cluster on how to configure MySQL NDB Cluster and Initial Startup of NDB Cluster on how to start it for the first time.

Installing Additional MySQL NDB Cluster Products and Components

You can use Zypper to install individual components and additional products of MySQL NDB Cluster from the MySQL SLES repository. To do that, assuming you already have the MySQL SLES repository on your system's repository list (if not, follow Step 1 and 2 of Installing MySQL NDB Cluster Using the SLES Repository), follow the same steps given in Installing Additional MySQL NDB Cluster Products and Components.

> **Note**
>
> *Known issue:* Currently, not all components required for running the MySQL NDB Cluster test suite are installed automatically when you install the test suite package (`mysql-cluster-community-test`). Install the following packages with `zypper install` before you run the test suite:
>
> - `mysql-cluster-community-auto-installer`
> - `mysql-cluster-community-management-server`
> - `mysql-cluster-community-data-node`
> - `mysql-cluster-community-memcached`
> - `mysql-cluster-community-java`
> - `mysql-cluster-community-ndbclient-devel`

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, “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-8.3.0-1.el7.src.rpm, depending on selected OS
Additional *debuginfo* RPMs	There are several `debuginfo` packages: mysql-community-client-debuginfo, mysql-community-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
`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

Package Name	Summary
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, and EL9-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 38 and 39
sl5	SUSE Linux Enterprise Server 15

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 mysql-community-server, mysql-community-client, mysql-community-client-plugins, mysql-community-libs, mysql-community-icu-data-files, mysql-community-common, and mysql-community-libs-compat packages to get a functional, standard MySQL 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:

```
$> 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`
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 `systemctl` (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. 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 is `/usr/lib64/mysql/plugin/debug` and is configurable with `plugin_dir`.

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,clier
```

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, `8.0`, or `8.3`, or `latest`). If ***:tag*** is omitted, the `latest` label is used, and the image for the latest GA release (which is the latest innovation release) 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.
  ```

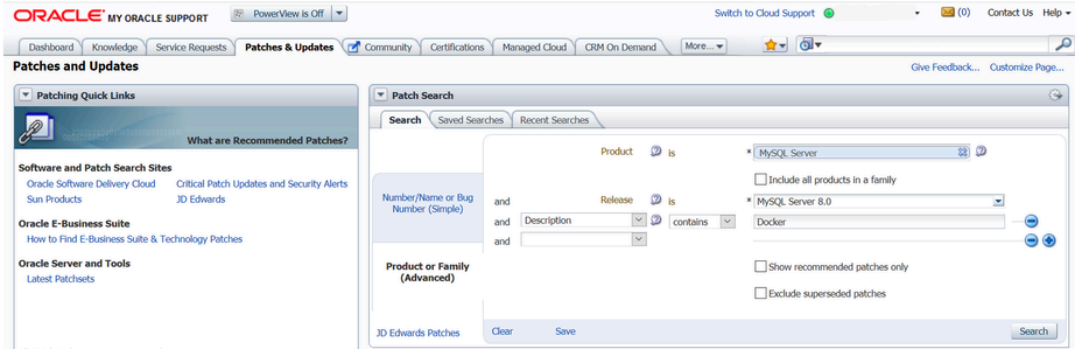
Download the Docker image for MySQL Enterprise Edition from the OCR with this command:

```
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.

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:lates
```

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:late
```

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:

```
$> docker ps
CONTAINER ID   IMAGE                                                    COMMAND                  CREAT
4cd4129b3211   container-registry.oracle.com/mysql/community-server:latest   "/entrypoint.sh mysq…"   8 sec
```

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 `mysql` client within the MySQL Server container you just started, and connect it to the MySQL Server. Use the `docker exec -it` command to start a `mysql` 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  ibdata1  mysql       mysql.sock.lock    private_key.pe
```

```
ca-key.pem  client-cert.pem  ib_buffer_pool  ib_logfile1  ibtmp1   mysql.sock  performance_schema  publ
```

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 8.0 to 8.3:

- Stop the MySQL 8.0 server (container name is `mysql80` in this example):

  ```
  docker stop mysql80
  ```

- Download the MySQL 8.3 Server Docker image. See instructions in Downloading a MySQL Server Docker Image. Make sure you use the right tag for MySQL 8.3.

- Start a new MySQL 8.3 Docker container (named `mysql83` 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=mysql83 \
     --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:8.3
  ```

  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 8.3 series (that is, from release 8.3.*x* to 8.3.*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 8.3 or the `latest` tag when starting your original container and there is now a new MySQL 8.3 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:8.3
```

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=mysql83new \
   --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:8.3
```

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-se
```

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:

```
$> docker inspect mysql1
...
 "Mounts": [
            {
                "Type": "volume",
                "Name": "4f2d463cfc4bdd4baebcb098c97d7da3337195ed2c6572bc0b89f7e845d27652",
                "Source": "/var/lib/docker/volumes/4f2d463cfc4bdd4baebcb098c97d7da3337195ed2c6572bc0b89
                "Destination": "/var/lib/mysql",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            }
        ],
...
```

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=mysql1 \
--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 *path-on-host-machine/my.cnf* at `/etc/my.cnf` (the server configuration file inside the container), and *path-on-host-machine/datadir* at `/var/lib/mysql` (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=mysql1 --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 8.3 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:8.3
```

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:

1. 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:8.3 \
   mysqlbackup -umysqlbackup -ppassword --backup-dir=/tmp/backup-tmp --with-timestamp \
   --backup-image=/data/backups/db.mbi backup-to-image
   ```

   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:

```
$> 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:8.3 \
mysqlbackup --backup-dir=/tmp/backup-tmp --with-timestamp \
--datadir=/var/lib/mysql --backup-image=/data/backups/db.mbi copy-back-and-apply-log
mysqlbackup completed OK! with 3 warnings
```

   The container exits with the message " `mysqlbackup completed OK!`" 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:8.3
```

   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:8.3 \
-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:

1. 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. Bind-mount 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:8.3 \
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 8.3 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).

- `MYSQL_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, `mysql` for the client tools, `mysql-server` for the server and associated tools, and `mysql-libs` 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
```

```
================================================================================
 Package                Arch          Version                Repository      Size
================================================================================
Installing:
 mysql                  x86_64        5.1.48-2.fc13          updates        889 k
 mysql-libs             x86_64        5.1.48-2.fc13          updates        1.2 M
 mysql-server           x86_64        5.1.48-2.fc13          updates        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
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm             | 136 kB     00:00
--------------------------------------------------------------------------------
Total                                            201 kB/s |  10 MB     00:52
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
  Installing     : perl-DBD-MySQL-4.017-1.fc13.x86_64                       3/4
  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 `systemctl`:

```
$> 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 8.0 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 `-DWITH_SYSTEMD=1` `CMake` option. See Section 4.7, “MySQL Source-Configuration Options”.

The following discussion covers these topics:

> **Note**
>
> On platforms for which systemd support for MySQL is installed, scripts such as `mysqld_safe` and the System V initialization script are unnecessary and are not installed. For example, `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 `mysqld_safe` on platforms that use systemd for server management is that use of `[mysqld_safe]` or `[safe_mysqld]` 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 `systemctl` 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 8.3, `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 8.3 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.

To obtain a binary MySQL distribution for Solaris in tarball or PKG format, https://dev.mysql.com/downloads/mysql/8.3.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.

To use this package, download the corresponding `mysql-VERSION-solaris11-PLATFORM.pkg.gz` file, then uncompress it. For example:

```
$> gunzip mysql-8.3.0-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:

```
$> pkgadd -d mysql-8.3.0-solaris11-x86_64.pkg
The following packages are available:
  1  mysql     MySQL Community Server (GPL)
               (i86pc) 8.3.0
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

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-8.3.0-solaris11-x86_64.pkg
$> mysqld_safe &
```

You should check the notes in Chapter 10, *Upgrading MySQL* before performing any upgrade.

Chapter 9 Postinstallation Setup and Testing

Table of Contents

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 the MSI installer and MySQL Configurator.
  - 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**
>
> The default authentication plugin is `caching_sha2_password`, and the `'root'@'localhost'` administrative account uses `caching_sha2_password` by default. The default authentication plugin before MySQL 8.0 was `mysql_native_password`, which is deprecated.

- 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.

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 `mysqld` with the `--initialize` or `--initialize-insecure` option, depending on whether you want the server to generate a random initial password for the `'root'@'localhost'` 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 `mysqld` while logged in as `mysql`, in which case you can omit the `--user` 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 8.3
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 `mysql` 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.

3. 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.

5. 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
     ```

3. 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 `init_file` 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 5.3.5, “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 `mysqld ended`, look for information in the error log (which by default is the *`host_name`*`.err` 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 5.4, “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 8.3\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 (`InnoDB`, `NDB`), be sure that you have them configured the way you want before starting the server. If you are using `InnoDB` tables, see InnoDB Configuration for guidelines and InnoDB 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 `mysql`, 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:

```
$> bin/mysqladmin version
mysqladmin  Ver 14.12 Distrib 8.3.0, for pc-linux-gnu on i686
...
Server version          8.3.0
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 14 days 5 hours 5 min 21 sec
Threads: 1  Questions: 366  Slow queries: 0
Opens: 0  Flush tables: 1  Open tables: 19
Queries per second avg: 0.000
```

To see what else you can do with `mysqladmin`, 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:

```
$> bin/mysqlshow
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

The list of installed databases may vary, but always includes at least `mysql` 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:

```
$> bin/mysql -e "SELECT User, Host, plugin FROM mysql.user" mysql
+------+-----------+-----------------------+
| User | Host      | plugin                |
+------+-----------+-----------------------+
| root | localhost | caching_sha2_password |
+------+-----------+-----------------------+
```

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 `mysql`, `mysqladmin`, and `mysqlshow`, see mysql — The MySQL Command-Line Client, mysqladmin — A MySQL Server Administration Program, and mysqlshow — 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 Configurator (see Section 5.2, “Configuration: Using MySQL Configurator”).

- 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 the MSI installer and MySQL Configurator 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 5.3.8, “Starting MySQL as a Windows Service”.
- On Unix and Unix-like systems, you can invoke `mysqld_safe`, which tries to determine the proper options for `mysqld` and then runs it with those options. See mysqld_safe — MySQL 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-8.2]` and `[mysqld-8.3]` are read by servers having versions 8.2.x, 8.3.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, `mysql.server` also reads the `[mysql_server]` group and `mysqld_safe` also reads the `[safe_mysqld]` group. To be current, you should update your option files to use the `[mysql.server]` and `[mysqld_safe]` 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

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 8.3 to MySQL 8.2, or from a MySQL 8.3 release to a previous MySQL 8.3 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 8.3” for changes that you should be aware of before upgrading. Some changes may require action.
- Review What Is New in MySQL 8.3 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 8.3. 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

> **Notes**
>
> - Make sure you understand the release model for MySQL innovation and long-term support (LTS) versions before you proceed with an upgrade.

- Monthly Rapid Updates (MRUs) and hotfixes also count as "releases" in the following discussions.
- It is necessary to upgrade to the latest innovation releases (for example, from MySQL 8.2 to 8.3) regularly to keep up with the latest bug fixes and security patches.
- We use "8.4.x LTS" as an example for an LTS release *only for illustrative purposes*.

The supported upgrade paths for MySQL Server are listed in Table 10.1, “Upgrade Paths for MySQL Server” below:

Table 10.1 Upgrade Paths for MySQL Server

Upgrade Path	Path Examples	Upgrade Methods	Notes
Within an LTS or bugfix release series	8.0.34 to 8.0.36 8.4.x LTS to 8.4.x+1 LTS	in-place upgrade, logical upgrade, asynchronous replication, MySQL Clone	
From an LTS or bugfix release to the next LTS or bugfix release	8.0.34 to 8.4.x LTS	in-place upgrade, logical upgrade, asynchronous replication, MySQL Clone	
From an LTS or bugfix release to an innovation release *before the next LTS series*	8.0.34 to 8.3.0	in-place upgrade, logical upgrade, asynchronous replication	
From an innovation release series to another one *before the next LTS series*	8.1.0 to 8.3.0	in-place upgrade, logical upgrade, asynchronous replication	
From MySQL 5.7 to an LTS or Innovation release	MySQL 5.7 to 8.x	Upgrade first to a MySQL 8.0 bugfix release (8.0.34 or later), then to an innovation release	See upgrade paths for MySQL 8.0 for upgrade instructions for MySQL 5.7.

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 8.3”.

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`.

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.
```

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.

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, start the server with the `--upgrade=NONE` or `--upgrade=MINIMAL` option.

- Step 2 upgrades the system tables to ensure that they have the current structure, and this includes the help tables but not the time zone tables. The procedure for loading time zone tables is platform dependent and requires decision making by the DBA, so it cannot be done automatically.
- 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, start the server with the `--upgrade=NONE` or `--upgrade=MINIMAL` option.

  To force table checking, start the server with the `--upgrade=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, start the server with the `--upgrade=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 8.3

Before upgrading to MySQL 8.3, review the changes linked in this section to identify those that apply to your current MySQL installation and applications.

- Overview of changes in MySQL 8.3.0
- Overview of changes in MySQL 8.2.0
- Overview of changes in MySQL 8.1.0
- MySQL 8.3.0 Release Notes
- MySQL 8.2.0 Release Notes
- MySQL 8.1.0 Release Notes

10.6 Preparing Your Installation for Upgrade

Before upgrading to the latest MySQL 8.3 release, ensure the upgrade readiness of your current MySQL 8.2 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 GA releases of MySQL 5.7, 8.0, and 8.3. 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.
   - 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;
```

3. Some keywords may be reserved in MySQL 8.3 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 8.2 `mysql` system database that have the same name as a table used by the MySQL 8.3 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 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`).

6. There must be no obsolete SQL modes defined by `sql_mode` system variable. Attempting to use an obsolete SQL mode prevents MySQL 8.3 from starting. Applications that use obsolete SQL modes should be revised to avoid them. For information about SQL modes removed in MySQL 8.3, see Server Changes.

7. 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.

8. 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 8.3, the maximum combined length of `ENUM` or `SET` column elements was 64K. In MySQL 8.3, 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.

9. Your MySQL 8.2 installation must not use features that are not supported by MySQL 8.3. 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 8.3. See Features Removed in MySQL 8.3, and Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.3. If you use any of these, an upgrade requires configuration changes.

10. 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
```

If `lower_case_table_names=1`, 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 `lower_case_table_names` setting at upgrade time is not recommended.

If upgrade to MySQL 8.3 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 8.2 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, `mysqld_safe` 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 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.

5. Shut down the old MySQL server. For example:

   ```
   mysqladmin -u root -p shutdown
   ```

6. 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.

7. Start the MySQL 8.3 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 8.3 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 8.2 to MySQL 8.3, 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 8.2 server on the same data directory, and fix the cause of any errors. Then perform another slow shutdown of the 8.2 server and start the MySQL 8.3 server to try again.

8. In the previous step, the server upgrades the data dictionary as necessary, making any changes required in the `mysql` system database between MySQL 8.2 and MySQL 8.3, 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 8.3, and examines all user databases for incompatibilities with the current version of MySQL.

   > **Note**
   >
   > The upgrade process does not upgrade the contents of the time zone tables. For upgrade instructions, see MySQL Server Time Zone 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, `mysqld_safe` 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 8.3 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 8.3. 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/8.3-datadir
```

   Copy the temporary `'root'@'localhost'` password displayed to your screen or written to your error log for later use.

6. Start the MySQL 8.3 server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/8.3-datadir &
```

7. Reset the `root` password:

```
$> mysql -u root -p
Enter password: ****  <- enter temporary root password
```

```
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:

   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/8.3-datadir --upgrade=FORCE &
   ```

   Upon restart with `--upgrade=FORCE`, the server makes any changes required in the `mysql` system schema between MySQL 8.2 and MySQL 8.3, so that you can take advantage of new privileges or capabilities. It also brings the Performance Schema, `INFORMATION_SCHEMA`, and `sys` schema up to date for MySQL 8.3, and examines all user schemas for incompatibilities with the current version of MySQL.

   > **Note**
   >
   > The upgrade process does not upgrade the contents of the time zone tables. For upgrade instructions, see MySQL Server Time Zone 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.

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.
3. Upgrade API nodes one at a time (including MySQL servers).

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.

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 by restarting 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

- An innovation series, such as MySQL 8.3, is in a separate track than a bugfix series, such as MySQL 8.0. The newest bugfix series is active by default.
- 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 track 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 editing the subrepository entries in the `/etc/yum.repos.d/mysql-community.repo` file.

As a general rule, to upgrade from one bugfix series to another, go to the next bugfix series rather than skipping a bugfix series. For example, if you are currently running MySQL 5.6 and wish to upgrade to MySQL 8.0, upgrade to MySQL 5.7 first before upgrading to MySQL 8.0. For additional details, see Section 10.5, "Changes in MySQL 8.3".

- For important information about upgrading from MySQL 5.6 to 5.7, see Upgrading from MySQL 5.6 to 5.7.
- For important information about upgrading from MySQL 5.7 to 8.0, see Upgrading from MySQL 5.7 to 8.0.
- In-place downgrading of MySQL is not supported by the MySQL Yum repository. Follow the instructions in Chapter 11, *Downgrading MySQL*.

Upgrading MySQL

Upgrade MySQL components using standard yum (or dnf) commands, such as MySQL Server:

```
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
```

> **Note**
>
> The MySQL server always restarts after an update by Yum.

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 *`package-name`* 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 8.3 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

To upgrade MySQL on Windows, either download and execute the latest MySQL Server MSI or use the Windows ZIP archive distribution.

> **Note**
>
> Unlike MySQL 8.3, MySQL 8.0 uses MySQL Installer to install and upgrade MySQL Server along with most other MySQL products; but MySQL Installer is not available with MySQL 8.1 and higher. However, the configuration functionality used in MySQL Installer is available as of MySQL 8.1 using Section 5.2, “Configuration: Using MySQL Configurator” that's bundled with both the MSI and Zip archive.

Considering 8.3 is an innovation release, there likely won't be a 8.3 point release to upgrade to. Upgrading beyond MySQL 8.3 involves a new series number, such as 8.1 to 8.2, so it's especially important to always back up your current MySQL installation before performing an upgrade. See Database Backup Methods.

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.

Upgrading MySQL with MSI

Download and execute the latest MSI. Although upgrading between release series is not directly supported, the "Custom Setup" option allows defining an installation location as otherwise the MSI installs to the standard location, such as `C:\Program Files\MySQL\MySQL Server 8.3\`.

Execute MySQL Configurator to configure your installation.

Upgrading MySQL Using the Windows ZIP Distribution

To perform an upgrade using the Windows ZIP archive distribution:

1. 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 mysqld_service_name` .

   If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 8.2 to 8.3, use `mysqladmin` from MySQL 8.2 as follows:

   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 8.2\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 mysqld_service_name` or `NET START mysqld_service_name` command if you run MySQL as a service, or invoke `mysqld` directly otherwise.

5. If you encounter errors, see Section 5.4, “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 8.2 instance between the `.frm` file of a table and the `InnoDB` data dictionary can cause an upgrade to MySQL 8.3 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 re-create 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 8.3 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 8.2 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` or `mysqlcheck`.

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
```

To rebuild all tables in all databases, use the `--all-databases` option:

```
mysqldump --all-databases > dump.sql
mysql < dump.sql
```

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 `mysqldump` to create a file containing SQL statements. You can then transfer the file to the other machine and feed it as input to the `mysql` 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, *`DUMPDIR`* 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:

```
mysqladmin create db_name           # create database
```

```
cat DUMPDIR/*.sql | mysql db_name   # create tables in database
mysqlimport db_name
   DUMPDIR/*.txt   # load data into tables
```

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 `mysql` database on the new machine, execute `mysqladmin flush-privileges` so that the server reloads the grant table information.

Chapter 11 Downgrading MySQL

Notes

- Make sure you understand the release model for MySQL innovation and long-term support (LTS) versions before you proceed with a downgrade.
- Monthly Rapid Updates (MRUs) and hotfixes also count as "releases" in the following discussions.
- We use "8.4.x LTS" as an example for an LTS release and "8.3.0" as an example for a bugfix release *only for illustrative purposes*.

The supported downgrade paths for MySQL Server are listed in Table 11.1, “Downgrade Paths for MySQL Server” below:

Table 11.1 Downgrade Paths for MySQL Server

Downgrade Path	Path Examples	Downgrade M
Within an LTS or bugfix release series	8.0.36 to 8.0.35 8.4.x+1 LTS to 8.4.x LTS	in-place downg MySQL Clone
From an LTS or bugfix release to the last LTS or bugfix release series	8.4.x LTS to 8.0.y	logical downgra
From an LTS or bugfix release to an innovation release *after the last LTS series*	8.4.x LTS to 8.3.0	logical downgra
From an innovation release series to another one *after the last LTS series*	8.3 to 8.2	logical downgra

Downgrading to MySQL 5.7 or earlier is not supported.

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` if `--syslog` 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.
`MYSQL_OPENSSL_UDF_DSA_BITS_THRESHOLD`	Maximum DSA key length for `create_asymmetric_priv_key()`. See MySQL Enterprise Encryption Usage and Examples.

Variable	Description
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 8.3; 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

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 http://search.cpan.org 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 8.3.

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-I-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 `libmysqlclient.so` is located to the `LD_RUN_PATH` environment variable. Some systems use `LD_LIBRARY_PATH` 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`.