MySQL Installation Guide

Abstract

This is the MySQL Installation Guide from the MySQL 5.6 Reference Manual.

For legal information, see the Legal Notices.

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

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Licensing information—MySQL 5.6. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.6, see this document 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 5.6, see this document for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL Cluster. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL Cluster NDB 7.3 or NDB 7.4, see this document 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 Cluster NDB 7.3 or NDB 7.4, see this document for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Document generated on: 2016-10-04 (revision: 49305)

Table of Contents

Preface and Legal Notices

This is the MySQL Installation Guide from the MySQL 5.6 Reference Manual.

Legal Notices

Copyright © 1997, 2016, Oracle and/or its affiliates. All rights reserved.

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

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

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

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.

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

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

Chapter 1 Installing and Upgrading 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 Section 10.1, “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 5.6 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 http://www.mysql.com/support/supportedplatforms/database.html on the MySQL Web site.

2. **Choose which distribution to install.**

   Several versions of MySQL are available, and most are available in several distribution formats. You can choose from pre-packaged distributions containing binary (precompiled) programs or source code. When in doubt, use a binary distribution. Oracle also provides access to the MySQL source code for those who want to see recent developments and test new code. To determine which version and type of distribution you should use, see Section 2.1, “Which MySQL Version and Distribution to Install”.

3. **Download the distribution that you want to install.**

   For instructions, see Section 2.2, “How to Get MySQL”. To verify the integrity of the distribution, use the instructions in Section 2.3, “Verifying Package Integrity Using MD5 Checksums or GnuPG”.

4. **Install the distribution.**

   To install MySQL from a binary distribution, use the instructions in Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*.

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

5. **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 Accounts”. This section describes how to secure the initial MySQL user accounts, *which have no passwords* until you assign passwords. The section applies whether you install MySQL using a binary or source distribution.

6. If you want to run the MySQL benchmark scripts, Perl support for MySQL must be available. See Chapter 12, *Perl Installation Notes*.

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

- **Unix, Linux, FreeBSD**

  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*.
- Solaris and OpenSolaris, including PKG and IPS formats, see Chapter 8, *Installing MySQL on Solaris and OpenSolaris*.
- IBM AIX, see Chapter 8, *Installing MySQL on Solaris and OpenSolaris*.
- FreeBSD, see Installing MySQL on FreeBSD.

- **Microsoft Windows**

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

  For information about managing MySQL instances, see Section 5.4, “MySQL Notifier”.

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

- **OS X**

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

  For information on making use of an OS X Launch Daemon to automatically start and stop MySQL, see Section 6.3, “Installing a 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 Which MySQL Version and Distribution to Install

MySQL is available on a number of operating systems and platforms. For information about those platforms that are officially supported, see http://www.mysql.com/support/supportedplatforms/database.html on the MySQL Web site.

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

First, decide whether to install a development release or a General Availability (GA) release. Development releases have the newest features, but are not recommended for production use. GA releases, also called production or stable releases, are meant for production use. We recommend using the most recent GA release.

The naming scheme in MySQL 5.6 uses release names that consist of three numbers and an optional suffix; for example, **mysql-5.6.1-m1**. The numbers within the release name are interpreted as follows:

- The first number (**5**) is the major version number.
- The second number (**6**) 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 (**1**) is the version number within the release series. This is incremented for each new bugfix release. In most cases, the most recent version within a series is the best choice.

Release names can also include a suffix to indicate the stability level of the release. Releases within a series progress through a set of suffixes to indicate how the stability level improves. The possible suffixes are:

- **m*N*** (for example, **m1**, **m2**, **m3**, ...) indicates a milestone number. MySQL development uses a milestone model, in which each milestone introduces a small subset of thoroughly tested features. Following the releases for one milestone, development proceeds with another small number of releases that focuses on the next set of features. From one milestone to the next, feature interfaces may change or features may even be removed, based on feedback provided by community members who try these earily releases. Features within milestone releases may be considered to be of pre-production quality.

- **rc** indicates a Release Candidate (RC). Release candidates are believed to be stable, having passed all of MySQL's internal testing. New features may still be introduced in RC releases, but the focus shifts to fixing bugs to stabilize features introduced earlier within the series.
- Absence of a suffix indicates a General Availability (GA) or Production release. GA releases are stable, having successfully passed through the earlier release stages, and are believed to be reliable, free of serious bugs, and suitable for use in production systems.

Development within a series begins with milestone releases, followed by RC releases, and finally reaches GA status releases.

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 OS X. Distributions are also available in more generic formats such as Zip archives or compressed `tar` files. On Windows, you can use the MySQL Installer to install a binary distribution.

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

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

  For additional information, see Section 4.4, “MySQL Source-Configuration Options”.
- You want to configure `mysqld` without some features that are included in the standard binary distributions. For example, distributions normally are compiled with support for all character sets. If you want a smaller MySQL server, you can recompile it with support for only the character sets you need.
- 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.

2.2 How to Get MySQL

Check our downloads page at http://dev.mysql.com/downloads/ for information about the current version of MySQL and for downloading instructions. For a complete up-to-date list of MySQL download mirror sites, see http://dev.mysql.com/downloads/mirrors.html. You can also find information there about becoming a MySQL mirror site and how to report a bad or out-of-date mirror.

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 a number of Debian-based Linux platforms, such as Ubuntu, MySQL can be installed using the MySQL APT Repository. See Section 7.3, “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 7.4, “Installing MySQL on Linux Using the MySQL SLES Repository” for details.

To obtain the latest development source, see Section 4.3, “Installing MySQL Using a Development Source Tree”.

2.3 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.

2.3.1 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, and you will 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 will assume the name `md5.exe`.

Linux and Microsoft Windows examples:

```
shell> md5sum mysql-standard-5.6.35-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945  mysql-standard-5.6.35-linux-i686.tar.gz
```

```
shell> md5.exe mysql-installer-community-5.6.35.msi
aaab65abbec64d5e907dcd41b8699945  mysql-installer-community-5.6.35.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.

2.3.2 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. See http://www.gnupg.org/ for more information about GnuPG and how to obtain and install it on your system. Most Linux distributions ship with GnuPG installed by default. For more information about GnuPG, see http://www.openpgp.org/.

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. Alternatively, you can cut and paste the key directly from the following text:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.9 (SunOS)
mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q2TXlTUUwgUmVs
ZWFzZSBFbmdpbmVlcmluZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iGkE
ExECACkCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAIZAQUCUwHUZgUJGmbLywAK
CRCMcY07UHLh9V+DAKCjS1gGwgVI/eut+5L+l2v3ybl+ZgCcD7ZoA341HtoroV3U
6xRD09fUgeq0O015U1FMIFBhY2thZ2Ugc2lnbmluZyBrZXkgKHd3dy5teXNxbC5j
b20pIDxidWlsZEBteXNxbC5jb20+iG8EMBECAC8FAk53Pa0oHSBidWlsZEBteXNx
bC5jb20gd2lsbCBzdG9wIHdvcmtpbmcgc29vbgAKCRCMcY07UHLh9bU9AJ9xDK0o
xJFL9vTl9OSZC4lX0K9AzwCcCrS9cnJyz79eaRjL0s2r/CcljdyIZQQTEQIAHQUC
R6yUtAUJDTBYqAULBwoDBAMVAwIDFgIBAheAABIJEIxxjTtQcuH1B2VHUEcAAQGu
kgCffz4GUEjzXkOi71VcwgCxASTgbe0An34LPr1j9fCbrXWXO14msIADfb5piEwE
ExECAAwFAj4+o9EFgwlmALsACgkQSVDhKrJykfIk4QCfWbEeKN+3TRspe+5xKj+k
QJSammIAnjUz0xFWPlVx0f8o38qNG1bq0cU9iEwEExECAAwFAj5CggMFgwliIokA
CgkQtvXNTca6JD+WkQCgiGmnoGjMojynp5ppvMXkyUkfnykAoK79E6h8rwkSDZou
iz7nMRisH8uyiEYEEBECAAYFAj+s468ACgkQr8UjSHiDdA/2lgCg21IhIMMABTYd
p/IBiUsP/JQLiEoAnRzMywEtujQz/E9ono7H1DkebDa4iEYEEBECAAYFAj+0Q3cA
CgkQhZavqzBzTmbGwwCdFqD1frViC7WRt8GKoOS7hzNN32kAnirlbwpnT7a6NOsQ
83nk11a2dePhiEYEEBECAAYFAkNbs+oACgkQi9gubzC5S1x/dACdELKoXQKkwJN0
gZztsM7kjsIgyFMAnRRMbHQ7V39XC90OIpaPjk3a01tgiEYEExECAAYFAkTxMyYA
CgkQ9knE9GCTUwwKcQCgibak/SwhxWH1ijRhgYCo5GtM4vcAnAhtzL57wcw1Kg1X
m7nVGetUqJ7fiEwEEBECAAwFAkGBywEFgwYi2YsACgkQGFnQH2d7oexCjQCcD8sJ
NDc/mS8m8OGDUOx9VMWcnGkAnj1YWOD+Qhxo3mI/Ul9oEAhNkjcfiEwEEBECAAwF
AkGByzQFgwYi2VgACgkQgcL36+ITtpIiIwCdFVNVUB8xe8mFXoPm4d9Z54PTjpMA
niSPA/ZsfJ3oOMLKar4F0QPPrdrGiEwEEBECAAwFAkGBy2IFgwYi2SoACgkQa3Ds
2V3D9HMJqgCbBYzr5GPXOXgP88jKzmdbjweqXeEAnRss4G2G/3qD7uhTL1SPT1SH
jWUXiEwEEBECAAwFAkHQkyQFgwXUEWgACgkQfSXKCsEpp8JiVQCghvWvkPqowsw8
w7WSseTcw1tflvkAni+vLHl/DqIly0LkZYn5jzK1dpvfiEwEEBECAAwFAkIrW7oF
gwV5SNIACgkQ5hukiRXruavzEwCgkzL5QkLSypcw9LGHcFSx1ya0VL4An35nXkum
g6cCJ1NP8r2I4NcZWIrqiEwEEhECAAwFAkAqWToFgwd6S1IACgkQPKEfNJT6+GEm
XACcD+A53A5OGM7w750W11ukq4iZ9ckAnRMvndAqn3YTOxxlLPj2UPZiSgSqiEwE
EhECAAwFAkA9+roFgwdmqdIACgkQ8tdcY+OcZZyy3wCgtDcwlaq20w0cNuXFLLNe
EUaFFTwAni6RHN80moSVAdDTRkzZacJU3M5QiEwEEhECAAwFAkEOCoQFgwaWmggA
CgkQOcor9D1qil/83QCeITZ9wIo7XAMjC6y4ZWUL4m+edZsAoMOhRIRi42fmrNFu
vNZbnMGej81viEwEEhECAAwFAkKApTQFgwUj/1gACgkQBA3AhXyDn6jjJACcD1A4
UtXk84J13JQyoH9+dy24714Aniwlsso/9ndICJOkqs2j5dlHFq6oiEwEExECAAwF
Aj5NTYQFgwlXVwgACgkQLbt2v63UyTMFDACglT5G5NVKf5Mj65bFSlPzb92zk2QA
n1uc2h19/IwwrsbIyK/9POJ+JMP7iEwEExECAAwFAkHXgHYFgwXNJBYACgkQZu/b
yM2C/T4/vACfXe67xiSHB80wkmFZ2krb+oz/gBAAnjR2ucpbaonkQQgnC3GnBqmC
vNaJiEwEExECAAwFAkIYgQ4FgwWMI34ACgkQdsEDHKIxbqGg7gCfQi2HcrHn+yLF
uNlH1oSOh48ZM0oAn3hKV0uIRJphonHaUYiUP1ttWgdBiGUEExECAB0FCwcKAwQD
```

```
FQMCAxYCAQIXgAUCS3AvygUJEPPzpwASB2VHUEcAAQEJEIxxjTtQcuH1sNsAniYp
YBGqy/HhMnw3WE8kXahOOR5KAJ4xUmWPGYP4l3hKxyNK9OAUbpDVYIh7BDARAgA7
BQJCdzX1NB0AT29wcy4uLiBzaG91bGQgaGF2ZSBiZWVuIGxvY2FsISBJJ20gKnNv
KiBzdHVwaWQuLi4ACgkQOcor9D1qil/vRwCdFo08f66oKLiuEAqzlf9iDlPozEEA
n2EgvCYLCCHjfGosrkrU3WK5NFVgiI8EMBECAE8FAkVvAL9IHQBTaG91bGQgaGF2
ZSBiZWVuIGEgbG9jYWwgc2lnbmF0dXJlLCBvciBzb21ldGhpbmcgLSBXVEYgd2Fz
IEkgdGhpbmtpbmc/AAoJEDnKK/Q9aopfoPsAn3BVqKOalJeF0xPSvLR90PsRlnmG
AJ44oisY7Tl3NJbPgZal8W32fbqgbIkCIgQQAQIADAUCQYHLhQWDBiLZBwAKCRCq
4+bOZqFEaKgvEACCErnaHGyUYa0wETjj6DLEXsqeOiXad4i9aBQxnD35GUgcFofC
/nCY4XcnCMMEnmdQ9ofUuU3OBJ6BNJIbEusAabgLooebP/3KEaiCIiyhHYU5jarp
ZAh+Zopgs3Oc11mQ1tIaS69iJxrGTLodkAsAJAeEUwTPq9fHFFzC1eGBysoyFWg4
bIjz/zClI+qyTbFA5g6tRoiXTo8ko7QhY2AA5UGEg+83Hdb6akC04Z2QRErxKAqr
phHzj8XpjVOsQAdAi/qVKQeNKROlJ+iq6+YesmcWGfzeb87dGNweVFDJIGA0qY27
pTb2lExYjsRFN4Cb13NfodAbMTOxcAWZ7jAPCxAPlHUG++mHMrhQXETоZnBFE4nb
nC7vOBNgWdjUgXcpkUCkop4b17BFpR+k8ZtYLSS8p2LLz4uAeCcSm2/msJxT7rC/
FvoH8428oHincqs2ICo9zO/Ud4HmmO0O+SsZdVKIIjinGyOVWb4OOzkAlnnhEZ3o
6hAHcREIsBgPwEYVTj/9ZdC0AO44Nj9cU7awaqgtrnwwfr/o4V2gl8bLSkltZU27
/29HeuOeFGjlFe0YrDd/aRNsxbyb2O28H4sG1CVZmC5uK1iQBDiSyA7Q0bbdofCW
oQzm5twlpKWnY8Oe0ub9XP5p/sVfck4FceWFHwv+/PC9RzSl33lQ6vM2wIkCIgQT
AQIADAUCQp8KHAWDBQWacAAKCRDYwgoJWiRXzyE+D/9uc7z6fIsalfOYoLN60ajA
bQbI/uRKBFugyZ5RoaItusn9Z2rAtn61WrFhu4uCSJtFN1ny2RERg40f56pTghKr
D+YEt+Nze6+FKQ5AbGIdFsR/2bUk+ZZRSt83e14Lcb6ii/fJfzkoIox9ltkifQxq
Y7Tvk4noKu4oLSc8O1Wsfc/y0B9sYUUCmUfcnq58DEmGie9ovUslmyt5NPnveXxp
5UeaRc5Rqt9tK2B4A+7/cqENrdZJbAMSunt2+2fkYiRunAFPKPBdJBsY1sxeL/A9
aKe0viKEXQdAWqdNZKNCi8rd/oOP99/9lMbFudAbX6nL2DSb1OG2Z7NWEqgIAzjm
pwYYPCKeVz5Q8R+if9/fe5+STY/55OaI33fJ2H3v+U435VjYqbrerWe36xJItcJe
qUzW71fQtXi1CTEl3w2ch7VF5oj/QyjabLnAlHgSlkSi6p7By5C2MnbCHlCfPnIi
nPhFoRcRGPjJe9nFwGs+QblvS/Chzc2WX3s/2SWm4gEUKRX4zsAJ5ocyfa/vkxCk
SxK/erWlCPf/J1T70+i5waXDN/E3enSet/WL7h94pQKpjz8OdGL4JSBHuAVGA+a+
dknqnPF0KMKLhjrgV+L7O84FhbmAP7PXm3xmiMPriXf+el5fZZequQoIagf8rdRH
HhRJxQgI0HNknkaOqs8dtrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWs
EN/lxaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLm
RDRiRjd1DTCHqeyX7CHhcghj/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hk
AWzE7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkb
f4fmLe11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8Ilbf5vSYHb
uE5p/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+
Lwqqa8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1Z
aSafanFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGo
TbOWI39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev4
2LmuQT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkK
Ht926s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUO
etdZWhe70YGNPw1yjWJT1IhUBBgRAgAMBQJOdz3tBQkT+wG4ABIHZUdQRwABAQkQ
jHGNO1By4fUUmwCbBYr2+bBEn/L2BOcnw9Z/QFWuhRMAoKVgCFm5fadQ3Afi+UQl
AcOphrnJ
=443I
-----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:

```
shell> gpg --import mysql_pubkey.asc
gpg: key 5072E1F5: public key "MySQL Release Engineering
<mysql-build@oss.oracle.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
```

You can also download the key from the public keyserver using the public key id, `5072E1F5`:

```
shell> gpg --recv-keys 5072E1F5
gpg: requesting key 5072E1F5 from hkp server keys.gnupg.net
gpg: key 5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
1 new user ID
gpg: key 5072E1F5: "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:

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

If you experience problems or require RPM specific information, see Section 2.3.4, “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-5.6.35-linux-i686.tar.gz`
Signature file	`mysql-standard-5.6.35-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:

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

If the downloaded package is valid, you will see a "Good signature" similar to:

```
shell> gpg --verify mysql-standard-5.6.35-linux-i686.tar.gz.asc
gpg: Signature made Tue 01 Feb 2011 02:38:30 AM CST using DSA key ID 5072E1F5
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:

```
shell> gpg --verify mysql-standard-5.6.35-linux-i686.tar.gz.asc
gpg: Signature made Wed 23 Jan 2013 02:25:45 AM PST using DSA key ID 5072E1F5
gpg: checking the trustdb
gpg: no ultimately trusted keys found
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: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5
```

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.
- *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.

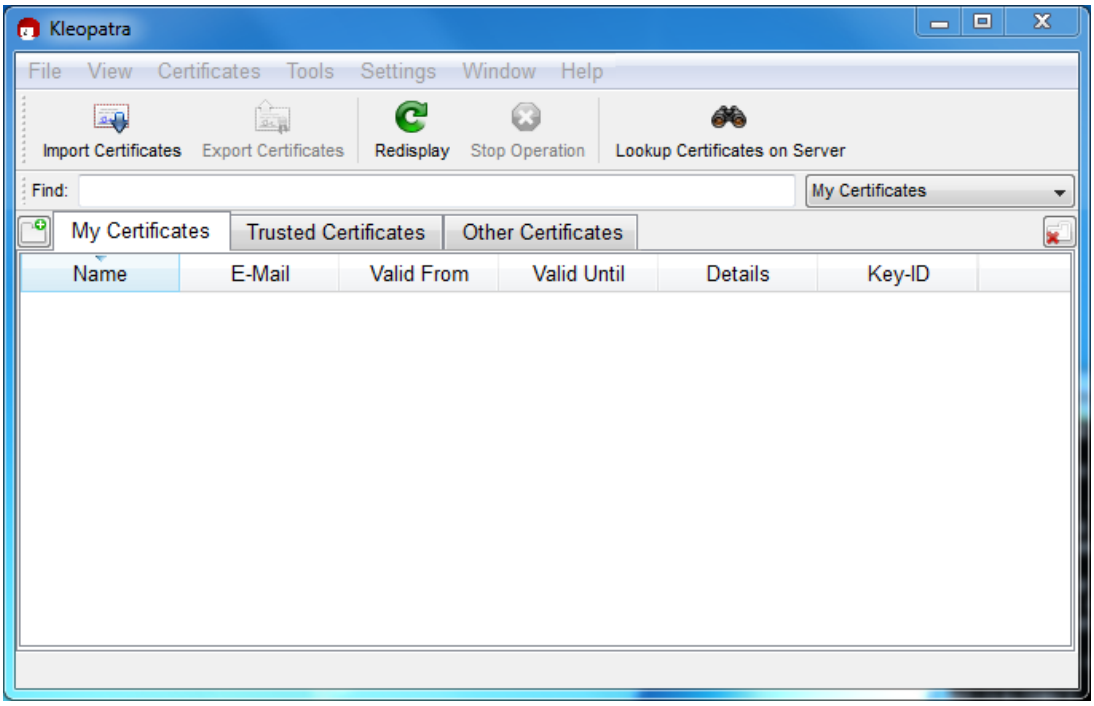
See the GPG documentation for more information on how to work with public keys.

2.3.3 Signature Checking Using Gpg4win for Windows

The Section 2.3.2, “Signature Checking Using GnuPG” section describes how to verify MySQL downloads using GPG. That guide also applies to Microsoft Windows, but another option is to use a GUI tool like Gpg4win. You may use a different tool but our examples are based on Gpg4win, and utilize its bundled `Kleopatra` GUI.

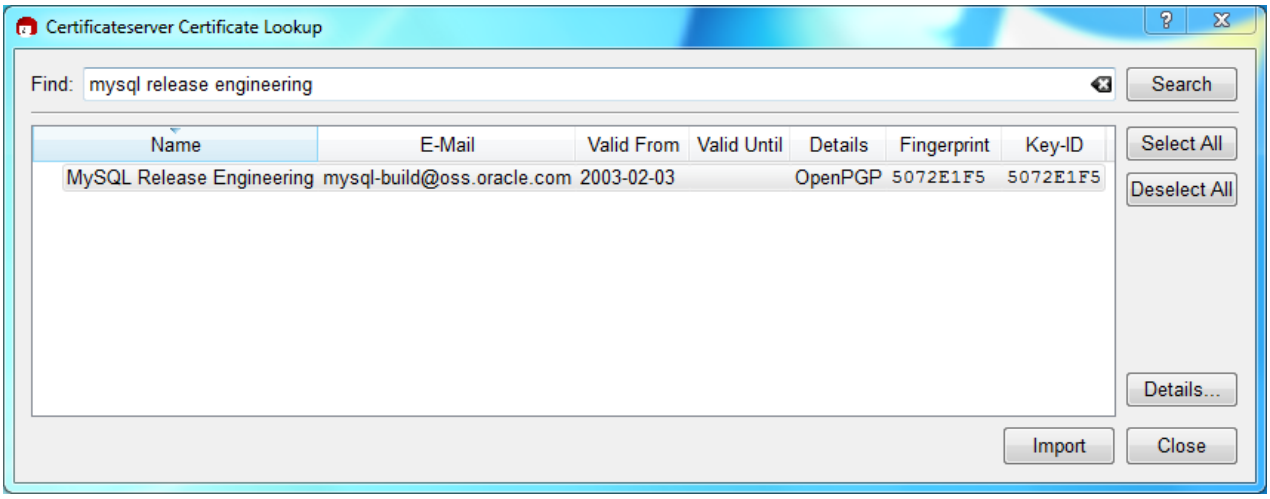
Download and install Gpg4win, and then load Kleopatra. The dialog should look similar to:

Figure 2.1 Initial screen after loading Kleopatra



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

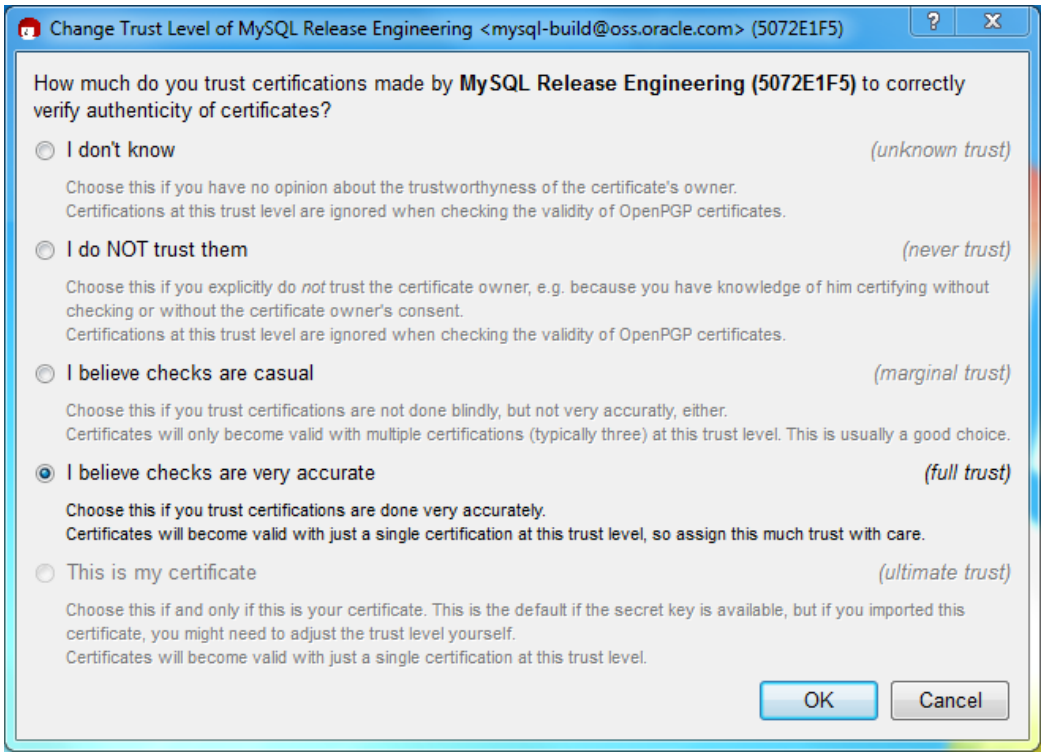
Figure 2.2 Finding the MySQL Release Engineering certificate



Select the "MySQL Release Engineering" certificate. The Fingerprint and Key-ID must be "5072E1F5", or choose **Details...** to confirm the certificate is valid. Now, import it by clicking **Import**. An import dialog will be displayed, choose **Okay**, and this certificate will now be listed under the **Imported Certificates** tab.

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

Figure 2.3 Changing the Trust level



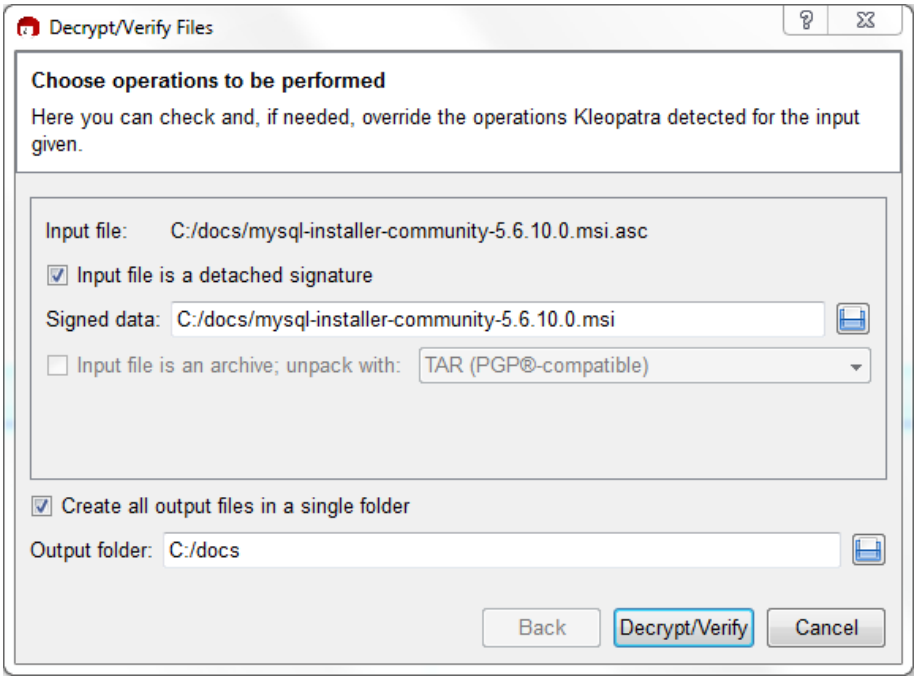
Next, verify the downloaded MySQL package file. This requires files for both the packaged file, and the signature. The signature file must have the same name as the packaged file but with an appended `.asc` extension, as shown by the example in the following table. The signature is linked to on the downloads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

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

File Type	File Name
Distribution file	`mysql-installer-community-5.6.35.msi`
Signature file	`mysql-installer-community-5.6.35.msi.asc`

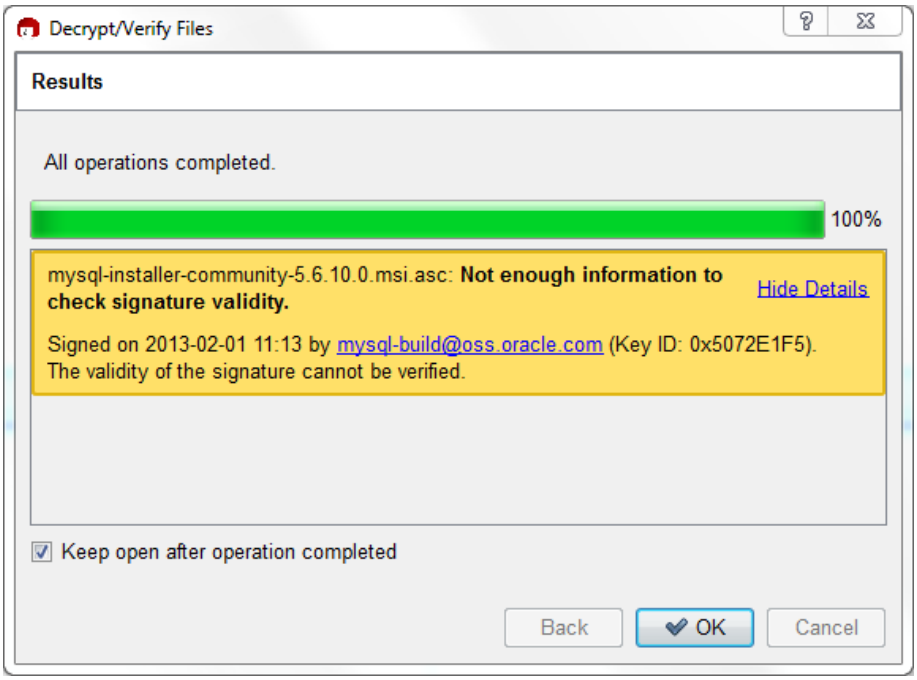
Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file. Either drag and drop the signature (`.asc`) file into Kleopatra, or load the dialog from **File**, **Decrypt/Verify Files...**, and then choose either the `.msi`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

Figure 2.4 The Decrypt/Verify Files dialog



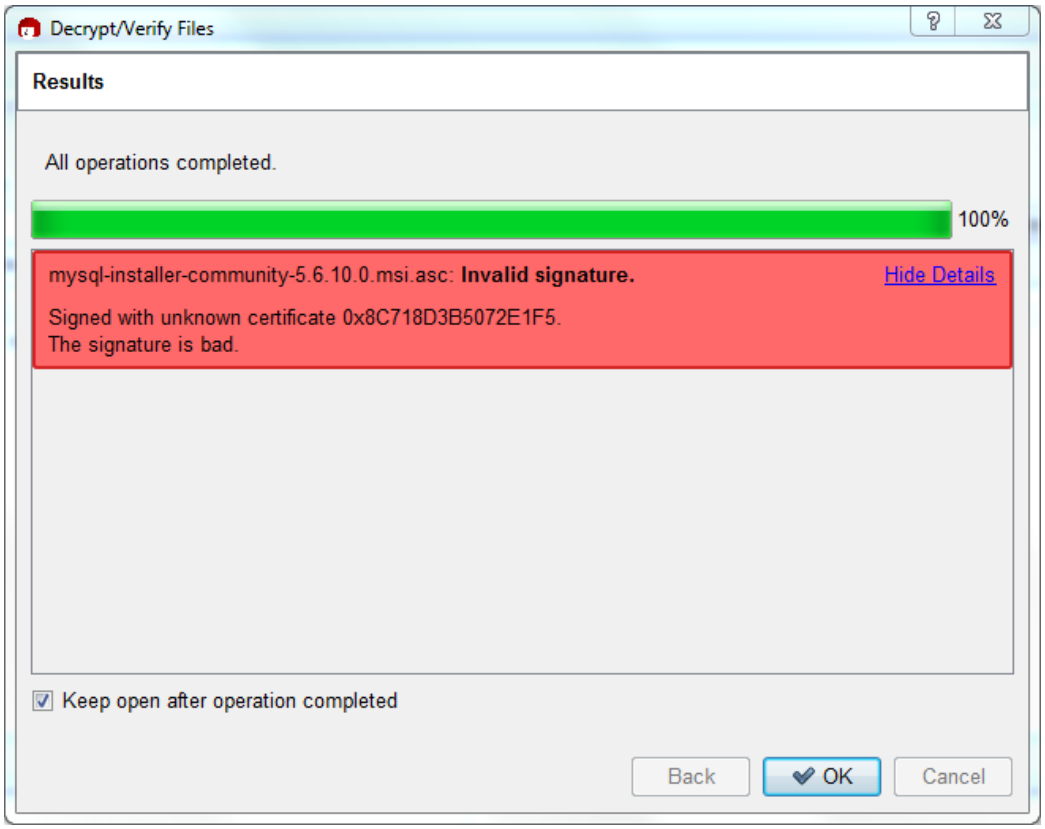
Click **Decrypt/Verify** to check the file. The two most common results will look like the following, and although the yellow warning looks problematic, the following means that the file check passed with success. You may now run this installer.

Figure 2.5 The Decrypt/Verify Results: Good



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

Figure 2.6 The Decrypt/Verify Results: Bad



The Section 2.3.2, “Signature Checking Using GnuPG” section explains why you probably don't see a green `Good signature` result.

2.3.4 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:

```
shell> rpm --checksig package_name.rpm
```

Example:

```
shell> rpm --checksig MySQL-server-5.6.35-0.linux_glibc2.5.i386.rpm
MySQL-server-5.6.35-0.linux_glibc2.5.i386.rpm: md5 gpg OK
```

> **Note**
>
> If you are using RPM 4.1 and it complains about `(GPG) NOT OK (MISSING KEYS: GPG#5072e1f5)`, 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:

```
shell> gpg --export -a 5072e1f5 > 5072e1f5.asc
shell> rpm --import 5072e1f5.asc
```

Alternatively, `rpm` also supports loading the key directly from a URL, and you can use this manual page:

```
shell> rpm --import http://dev.mysql.com/doc/refman/5.6/en/checking-gpg-signature.html
```

If you need to obtain the MySQL public key, see Section 2.3.2, “Signature Checking Using GnuPG”.

2.4 Installation Layouts

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

- Section 5.1, “MySQL Installation Layout on Microsoft Windows”
- Section 4.1, “MySQL Layout for Source Installation”
- Table 3.1, “MySQL Installation Layout for Generic Unix/Linux Binary Package”
- Table 7.1, “MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone”
- Table 6.1, “MySQL Installation Layout on OS X”

2.5 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.gz` 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. For other platform-specific package formats, see the other platform-specific sections. For example, for Windows distributions, see Chapter 5, *Installing MySQL on Microsoft Windows*.

To obtain MySQL, see Section 2.2, “How to Get MySQL”.

MySQL compressed `tar` file binary distributions have names of the form `mysql-`*`VERSION`*`-`*`OS`*`.tar.gz`, where *`VERSION`* is a number (for example, `5.6.35`), and *`OS`* indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

> **Warning**
>
> If you have previously installed MySQL using your operating system native package management system, such as `yum` or `apt-get`, 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.

> **Warning**
>
> MySQL has a dependency on the `libaio` library. Data directory initialization and subsequent server startup steps will fail if this library is not installed locally. If necessary, install it using the appropriate package manager. For example, on Yum-based systems:
>
> ```
> shell> yum search libaio  # search for info
> shell> yum install libaio # install library
> ```
>
> Or, on APT-based systems:
>
> ```
> shell> apt-cache search libaio # search for info
> shell> apt-get install libaio1 # install library
> ```

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

On Unix, 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`, `scripts`	`mysqld` server, client and utility programs

Directory	Contents of Directory
data	Log files, databases
docs	MySQL manual in Info format
man	Unix manual pages
include	Include (header) files
lib	Libraries
share	Miscellaneous support files, including error messages, sample configuration files, SQL for database installation
sql-bench	Benchmarks

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:

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> 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 (OpenSolaris) command.

> **Note**
>
> The procedure does not assign passwords to MySQL accounts. To do so, use the instructions in Section 9.4, “Securing the Initial MySQL Accounts”.

As of MySQL 5.6.8, mysql_install_db creates a default option file named my.cnf in the base installation directory. This file is created from a template included in the distribution package named my-default.cnf. For more information, see Using a Sample Default Server Configuration File.

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 one. 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, or they may have different names such as adduser and addgroup.

```
shell> groupadd mysql
shell> 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`.

```
shell> cd /usr/local
```

Obtain a distribution file using the instructions in Section 2.2, “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:

```
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
```

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

If your `tar` does not have `z` option support, use `gunzip` 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:

```
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

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

```
shell> 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:

```
```

```
shell> 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 http://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*.

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 2.2, “How to Get MySQL”. For instructions on building from a standard distribution, see Section 4.2, “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 `5.6.35`. 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 4.3, “Installing MySQL Using a Development Source Tree”.

Source Installation System Requirements

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 higher. It may already be available on your system as `gmake`. GNU `make` is available from http://www.gnu.org/software/make/.
- A working ANSI C++ compiler. GCC 4.2.1 or later, Sun Studio 12 or later, Visual Studio 2010 or later, and many current vendor-supplied compilers are known to work.
- Sufficient free memory. If you encounter problems 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. On Windows, you can use a version such as ActiveState 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 http://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. The GitHub Help provides instructions for downloading and installing Git on different platforms. MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: MySQL on GitHub
- `bison` 2.1 or higher, 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 OpenSolaris and 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.

4.1 MySQL Layout for Source Installation

By default, when you install MySQL after compiling it from source, the installation step installs files under `/usr/local/mysql`. The component locations under the installation directory are the same as for binary distributions. See Table 3.1, “MySQL Installation Layout for Generic Unix/Linux Binary Package”, and Section 5.1, “MySQL Installation Layout on Microsoft Windows”. To configure installation locations different from the defaults, use the options described at Section 4.4, “MySQL Source-Configuration Options”.

4.2 Installing MySQL Using a Standard Source Distribution

To install MySQL from a standard source distribution:

1. Verify that your system satisfies the tool requirements listed at Chapter 4, *Installing MySQL from Source*.
2. Obtain a distribution file using the instructions in Section 2.2, “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*.

In MySQL 5.6, `CMake` is used 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.

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

The result is one or more binary RPM packages that you install as indicated in Section 7.5, “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
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
# Beginning of source-build specific instructions
shell> tar zxvf mysql-VERSION.tar.gz
shell> cd mysql-VERSION
shell> cmake .
shell> make
shell> make install
# End of source-build specific instructions
# Postinstallation setup
shell> cd /usr/local/mysql
shell> chown -R mysql .
```

```
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

`mysql_install_db` creates a default option file named `my.cnf` in the base installation directory. This file is created from a template included in the distribution package named `my-default.cnf`. For more information, see Using a Sample Default Server Configuration File.

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

On Unix, set up the `mysql` user and group that will be used to run and execute the MySQL server and own the database directory. For details, see Creating a `mysql` System User and Group, in Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*. 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 2.2, “How to Get MySQL”.

Unpack the distribution into the current directory:

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

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

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

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

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

  ```
  shell> 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:

```
shell> cd mysql-VERSION
```

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

```
shell> cmake .
```

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

```
shell> cmake . -G "Visual Studio 10 2010"
shell> cmake . -G "Visual Studio 10 2010 Win64"
```

On OS X, to use the Xcode IDE:

```
shell> 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 4.4, “MySQL Source-Configuration Options”.

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

```
shell> cmake . -L   # overview
shell> cmake . -LH  # overview with help text
shell> cmake . -LAH # all params with help text
shell> 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 up, 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 on Unix before re-running `CMake`:

```
shell> make clean
```

```
shell> rm CMakeCache.txt
```

Or, on Windows:

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

If you build out of the source tree (as described later), the `CMakeCache.txt` file and all built files are in the build directory, so you can remove that directory to object files and cached configuration information.

If you are going to send mail to a MySQL mailing list to ask for configuration assistance, first 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:

```
shell> make
shell> 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:

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

It is possible to build out 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 `bld` at the same level like this:

```
shell> mkdir bld
shell> cd bld
shell> cmake ../mysql-src
```

The build directory need not actually be outside the source tree. For example, to build in a directory, you can build in a directory named `bld` under the top-level source tree, do this, starting with `mysql-src` as your current working directory:

```
shell> mkdir bld
shell> cd bld
shell> 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.

If you have gotten to the compilation stage, but the distribution does not build, see Section 4.5, “Dealing with Problems Compiling MySQL”, for help. If that does not solve the problem, please enter it into our bugs database using the instructions given in How to Report Bugs or Problems. If you have installed the latest versions of the required tools, and they crash trying to process our configuration files, please report that also. However, if you get a `command not found` error or a similar problem for required tools, do

not report it. Instead, make sure that all the required tools are installed and that your `PATH` variable is set correctly so that your shell can find them.

Install the Distribution

On Unix:

```
shell> 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:

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

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

```
shell> 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:

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

You can install the resulting `.zip` archive where you like. See Section 5.5, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”.

Perform Postinstallation Setup

The remainder of the installation process involves setting up the configuration file, creating the core databases, and starting the MySQL server. For instructions, see Chapter 9, *Postinstallation Setup and Testing*.

> **Note**
>
> The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in Chapter 9, *Postinstallation Setup and Testing*.

4.3 Installing MySQL Using a Development Source Tree

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

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

MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: MySQL on GitHub

Prerequisites for Installing from Development Source

To install MySQL from a development source tree, your system must satisfy the tool requirements outlined in Chapter 4, *Installing MySQL from Source*.

Setting Up a MySQL Git Repository

To set up a MySQL Git repository on your machine, use this procedure:

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 will 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: 1035465, done.
remote: Total 1035465 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1035465/1035465), 437.48 MiB | 5.10 MiB/s, done.
Resolving deltas: 100% (855607/855607), done.
Checking connectivity... done.
Checking out files: 100% (21902/21902), 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
BUILD            COPYING             libmysqld    regex          tests
BUILD-CMAKE      dbug                libservices  scripts        unittest
client           Docs                man          sql            VERSION
cmake            extra               mysql-test   sql-bench      vio
CMakeLists.txt   include             mysys        sql-common     win
cmd-line-utils   INSTALL-SOURCE      packaging    storage        zlib
config.h.cmake   INSTALL-WIN-SOURCE  plugin       strings
configure.cmake  libmysql            README       support-files
```

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

```
~/mysql-server$ git branch -r
  origin/5.5
  origin/5.6
  origin/5.7
  origin/HEAD -> origin/5.7
  origin/cluster-7.2
  origin/cluster-7.3
  origin/cluster-7.4
```

4. To view the branches that are checked out in your local repository, issue the `git branch` command. When you cloned the MySQL Git repository, the MySQL 5.7 branch was checked out automatically. The asterisk identifies the 5.7 branch as the active branch.

```
~/mysql-server$ git branch
* 5.7
```

5. To check out a different MySQL branch, run the `git checkout` command, specifying the branch name. For example, to checkout the MySQL 5.6 branch:

```
~/mysql-server$ git checkout 5.6
Branch 5.6 set up to track remote branch 5.6 from origin.
```

```
Switched to a new branch '5.6'
```

6. Run `git branch` again to verify that the MySQL 5.6 branch is present. MySQL 5.6, which is the last branch you checked out, is marked by an asterisk indicating that it is the active branch.

   ```
   ~/mysql-server$ git branch
   * 5.6
     5.7
   ```

   The `git checkout` command is also used to switch branches. For example, to make MySQL 5.7 the active branch again, you would run `git checkout 5.7`.

7. 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 5.6
   ~/mysql-server$ git pull
   ```

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

   ```
   ~/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, send an email to the MySQL `internals` mailing list. See MySQL Mailing Lists. For information about contributing a patch, see Contributing to MySQL Server.

8. 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 4.2, “Installing MySQL Using a Standard Source Distribution”, except that you skip the part about obtaining and unpacking the distribution.

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

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

4.4 MySQL Source-Configuration Options

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

```
shell> cmake . -LH
shell> ccmake .
```

You can also affect `CMake` using certain environment variables. See Chapter 11, *Environment Variables*.

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

Table 4.1 MySQL Source-Configuration Option Reference (CMake)

Formats	Description	Default	Introduced	Removed
BUILD_CONFIG	Use same build options as official releases			
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			
CPACK_MONOLITHIC_INSTALL	Whether package build produces single file	OFF		
DEFAULT_CHARSET	The default server character set	latin1		
DEFAULT_COLLATION	The default server collation	latin1_swedish_ci		
-DWITH_SYMVER16	Whether libmysqlclient.so.18 contains both symver 16 and 18 symbols.	OFF	5.6.31	
ENABLED_LOCAL_INFILE	Whether to enable LOCAL for LOAD DATA INFILE	OFF		
ENABLED_PROFILING	Whether to enable query profiling code	ON		
ENABLE_DEBUG_SYNC	Whether to enable Debug Sync support	ON		
ENABLE_DOWNLOADS	Whether to download optional files	OFF		
ENABLE_DTRACE	Whether to include DTrace support			
ENABLE_GCOV	Whether to include gcov support		5.6.3	
ENABLE_GPROF	Enable gprof (optimized Linux builds only)	OFF	5.6.6	
IGNORE_AIO_CHECK	With -DBUILD_CONFIG=mysql_release, ignore libaio check	OFF	5.6.1	
INNODB_PAGE_ATOMIC_REF_COUNT	Enable or disable atomic page reference counting	ON	5.6.16	
INSTALL_BINDIR	User executables directory	PREFIX/bin		
INSTALL_DOCDIR	Documentation directory	PREFIX/docs		
INSTALL_DOCREADMEDIR	README file directory	PREFIX		
INSTALL_INCLUDEDIR	Header file directory	PREFIX/include		
INSTALL_INFODIR	Info file directory	PREFIX/docs		

Formats	Description	Default	Introduced	Removed
INSTALL_LAYOUT	Select predefined installation layout	STANDALONE		
INSTALL_LIBDIR	Library file directory	PREFIX/lib		
INSTALL_MANDIR	Manual page directory	PREFIX/man		
INSTALL_MYSQLSHAREDIR	Shared data directory	PREFIX/share		
INSTALL_MYSQLTESTDIR	mysql-test directory	PREFIX/mysql-test		
INSTALL_PLUGINDIR	Plugin directory	PREFIX/lib/plugin		
INSTALL_SBINDIR	Server executable directory	PREFIX/bin		
INSTALL_SCRIPTDIR	Scripts directory	PREFIX/scripts		
INSTALL_SECURE_FILE_PRIVDIR	secure_file_priv default value	platform specific	5.6.34	
INSTALL_SECURE_FILE_PRIVEMBEDDEDDIR	secure_file_priv default value for libmysqld		5.6.34	
INSTALL_SHAREDIR	aclocal/mysql.m4 installation directory	PREFIX/share		
INSTALL_SQLBENCHDIR	sql-bench directory	PREFIX		
INSTALL_SUPPORTFILESDIR	Extra support files directory	PREFIX/support-files		
MEMCACHED_HOME	Path to memcached	[none]		
MYSQL_DATADIR	Data directory			
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF		
MYSQL_PROJECT_NAME	Windows/OS X project name	3306	5.6.5	
MYSQL_TCP_PORT	TCP/IP port number	3306		
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock		
ODBC_INCLUDES	ODBC includes directory			
ODBC_LIB_DIR	ODBC library directory			
OPTIMIZER_TRACE	Whether to support optimizer tracing		5.6.3	
SUNPRO_CXX_LIBRARY	Client link library on Solaris 10+		5.6.20	
SYSCONFDIR	Option file directory			
TMPDIR	tmpdir default value		5.6.16	
WITHOUT_SERVER	Do not build the server	OFF		
WITHOUT_xxx_STORAGE_ENGINE	Exclude storage engine xxx from build			
WITH_ASAN	Enable AddressSanitizer	OFF	5.6.15	

Formats	Description	Default	Introduced	Removed
WITH_BUNDLED_LIBEVENT	Use bundled libevent when building ndbmemcache	ON		
WITH_BUNDLED_MEMCACHED	Use bundled memcached when building ndbmemcache	ON		
WITH_CLASSPATH	Classpath to use when building MySQL Cluster Connector for Java. Default is an empty string.			
WITH_DEBUG	Whether to include debugging support	OFF		
WITH_DEFAULT_COMPILER_OPTIONS	Whether to use default compiler options	ON	5.6.6	
WITH_DEFAULT_FEATURE_SET	Whether to use default feature set	ON	5.6.6	
WITH_EDITLINE	Which libedit/editline library to use	bundled	5.6.12	
WITH_EMBEDDED_SERVER	Whether to build embedded server	OFF		
WITH_EMBEDDED_SHARED_LIBRARY	Whether to build a shared embedded server library	OFF	5.6.17	
WITH_ERROR_INSERT	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF		
WITH_EXTRA_CHARSETS	Which extra character sets to include	all		
WITH_INNODB_MEMCACHED	Whether to generate memcached shared libraries.	OFF		
WITH_LIBEDIT	Use bundled libedit library	ON		5.6.12
WITH_LIBEVENT	Which libevent library to use	bundled	5.6.6	
WITH_LIBWRAP	Whether to include libwrap (TCP wrappers) support	OFF		
WITH_NDBCLUSTER	Build the NDB storage engine; alias for WITH_NDBCLUSTER_STORAGE_ENGINE	ON		
WITH_NDBCLUSTER_STORAGE_ENGINE	Build the NDB storage engine	ON		
WITH_NDBMTD	Build multi-threaded data node.	ON		
WITH_NDB_BINLOG	Enable binary logging by default by mysqld.	ON		
WITH_NDB_DEBUG	Produce a debug build for testing or troubleshooting.	OFF		

Formats	Description	Default	Introduced	Removed
WITH_NDB_JAVA	Enable building of Java and ClusterJ support. Enabled by default. Supported in MySQL Cluster only.	ON		
WITH_NDB_PORT	Default port used by a management server built with this option. If this option was not used to build it, the management server's default port is 1186.	[none]		
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_NUMA	Set NUMA memory allocation policy		5.6.27	
WITH_READLINE	Use bundled readline library	OFF		5.6.5
WITH_SSL	Type of SSL support	bundled		
WITH_UNIXODBC	Enable unixODBC support	OFF		
WITH_VALGRIND	Whether to compile in Valgrind header files	OFF		
WITH_ZLIB	Type of zlib support	bundled		
WITH_xxx_STORAGE_ENGINE	Compile storage engine xxx statically into server			

The following sections provide more information about CMake options.

- General Options
- Installation Layout Options
- Storage Engine Options
- Feature Options
- Compiler Flags
- CMake Options for Compiling MySQL Cluster

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.

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.

- `-DCMAKE_BUILD_TYPE=type`

  The type of build to produce:

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

- `-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.

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 with 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:

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

- `-DINSTALL_LIBDIR=dir_name`

  Where to install library files.

- `-DINSTALL_MANDIR=dir_name`

  Where to install manual pages.

- `-DINSTALL_MYSQLSHAREDIR=dir_name`

  Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=dir_name`

  Where to install the `mysql-test` directory. As of MySQL 5.6.12, to suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_MYSQLTESTDIR=`).

- `-DINSTALL_PLUGINDIR=dir_name`

  The location of the plugin directory.

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

- `-DINSTALL_SBINDIR=dir_name`

  Where to install the `mysqld` server.

- `-DINSTALL_SCRIPTDIR=dir_name`

  Where to install `mysql_install_db`.

- `-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.

  This option was added in MySQL 5.6.34. To set the value for the `libmysqld` embedded server, use `INSTALL_SECURE_FILE_PRIV_EMBEDDEDDIR`.

- `-DINSTALL_SECURE_FILE_PRIV_EMBEDDEDDIR=dir_name`

  The default value for the `secure_file_priv` system variable, for the `libmysqld` embedded server. This option was added in MySQL 5.6.34.

- `-DINSTALL_SHAREDIR=dir_name`

  Where to install `aclocal/mysql.m4`.

- `-DINSTALL_SQLBENCHDIR=dir_name`

Where to install the `sql-bench` directory. To suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_SQLBENCHDIR=`).

- `-DINSTALL_SUPPORTFILESDIR=`*`dir_name`*

  Where to install extra support files.

- `-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, and may be used while configuring Connector/ODBC.

- `-DODBC_LIB_DIR=`*`dir_name`*

  The location of the ODBC library directory, and 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.

- `-DTMPDIR=`*`dir_name`*

  The default location to use for the `tmpdir` system variable. If unspecified, the value defaults to `P_tmpdir` in `<stdio.h>`. This option was added in MySQL 5.6.16.

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`, `FEDERATED`, `NDB` or `NDBCLUSTER` (`NDB`), `PARTITION` (partitioning support), and `PERFSCHEMA` (Performance Schema). Examples:

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

> **Note**
>
> `WITH_NDBCLUSTER_STORAGE_ENGINE` is supported only when building MySQL Cluster using the MySQL Cluster sources. It cannot be used to enable clustering support in other MySQL source trees or distributions. In MySQL Cluster source distributions, it is enabled by default. See Building MySQL Cluster from Source on

Linux, and Compiling and Installing MySQL Cluster from Source on Windows, for more information.

To exclude a storage engine from the build, use `-DWITHOUT_engine_STORAGE_ENGINE=1`. Examples:

```
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1
-DWITHOUT_FEDERATED_STORAGE_ENGINE=1
-DWITHOUT_PARTITION_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

- `-DCOMPILATION_COMMENT=string`

  A descriptive comment about the compilation environment.

- `-DDEFAULT_CHARSET=charset_name`

  The server character set. By default, MySQL uses the `latin1` (cp1252 West European) 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`, `utf8`, `utf8mb4`, `utf16`, `utf16le`, `utf32`. The permissible character sets are listed in the `cmake/character_sets.cmake` file as the value of `CHARSETS_AVAILABLE`.

  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 `latin1_swedish_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.

- `-DENABLE_DEBUG_SYNC=bool`

  Whether to compile the Debug Sync facility into the server. This facility is used for testing and debugging. This option is enabled by default, but has no effect unless MySQL is configured with debugging enabled. If debugging is enabled and you want to disable Debug Sync, use `-DENABLE_DEBUG_SYNC=0`.

  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.

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

- `-DENABLE_DOWNLOADS=bool`

  Whether to download optional files. For example, with this option enabled, `CMake` downloads the Google Test distribution that is used by the test suite to run unit tests.

- `-DENABLE_DTRACE=bool`

  Whether to include support for DTrace probes. For information about DTrace, wee Tracing mysqld Using DTrace

- `-DENABLE_GCOV=bool`

  Whether to include gcov support (Linux only).

- `-DENABLE_GPROF=bool`

  Whether to enable `gprof` (optimized Linux builds only). This option was added in MySQL 5.6.6.

- `-DENABLED_LOCAL_INFILE=bool`

  Whether to enable `LOCAL` capability in the client library for `LOAD DATA INFILE`.

  This option controls client-side `LOCAL` capability, but the capability can be set on the server side at server startup with the `--local-infile` option. See Security Issues with LOAD DATA LOCAL.

- `-DENABLED_PROFILING=bool`

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

- `-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`. This option was added in MySQL 5.6.1.

- `-DINNODB_PAGE_ATOMIC_REF_COUNT=bool`

  Whether to enable or disable atomic page reference counting. Fetching and releasing pages from the buffer pool and tracking the page state are expensive and complex operations. Using a page mutex to track these operations does not scale well. With `INNODB_PAGE_ATOMIC_REF_COUNT=ON` (default), fetch and release is tracked using atomics where available. For platforms that do not support atomics, set `INNODB_PAGE_ATOMIC_REF_COUNT=OFF` to disable atomic page reference counting.

  When atomic page reference counting is enabled (default), "`[Note] InnoDB: Using atomics to ref count buffer pool pages`" is printed to the error log at server startup. If atomic page reference counting is disabled, "`[Note] InnoDB: Using mutexes to ref count buffer pool pages`" is printed instead.

  `INNODB_PAGE_ATOMIC_REF_COUNT` was introduced with the fix for MySQL Bug #68079. The option is removed in MySQL 5.7.5. Support for atomics is required to build MySQL as of MySQL 5.7.5, which makes the option obsolete.

- `-DMYSQL_MAINTAINER_MODE=bool`

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

- `-DMYSQL_PROJECT_NAME=name`

  For Windows or OS X, the project name to incorporate into the project file name. This option was added in MySQL 5.6.5.

- `-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. This option was added in MySQL 5.6.3.

- `-DWITH_ASAN=bool`

  Whether to enable AddressSanitizer, for compilers that support it. The default is off. This option was added in MySQL 5.6.15.

- `-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.

- `-DWITH_DEFAULT_FEATURE_SET=bool`

  Whether to use the flags from `cmake/build_configurations/feature_set.cmake`. This option was added in MySQL 5.6.6.

- `-DWITH_EDITLINE=value`

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

  `WITH_EDITLINE` was added in MySQL 5.6.12. It replaces `WITH_LIBEDIT`, which has been removed.

- `-DWITH_EMBEDDED_SERVER=bool`

  Whether to build the `libmysqld` embedded server library.

- `-DWITH_EMBEDDED_SHARED_LIBRARY=bool`

  Whether to build a shared `libmysqld` embedded server library. This option was added in MySQL 5.6.17.

- `-DWITH_EXTRA_CHARSETS=name`

  Which extra character sets to include:

  - `all`: All character sets. This is the default.
  - `complex`: Complex character sets.

  - `none`: No extra character sets.

- `-DWITH_INNODB_MEMCACHED=bool`

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

- `-DWITH_LIBEVENT=string`

  Which `libevent` library to use. Permitted values are `bundled` (default), `system`, and `yes`. If you specify `system` or `yes`, the system `libevent` library is used if present. If the system library is not found, the bundled `libevent` library is used. The `libevent` library is required by `InnoDB` memcached.

- `-DWITH_LIBEDIT=bool`

  Whether to use the `libedit` library bundled with the distribution.

  `WITH_LIBEDIT` was removed in MySQL 5.6.12. Use `WITH_EDITLINE` instead.

- `-DWITH_LIBWRAP=bool`

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

- `-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

  This option was added in MySQL 5.6.27.

- `-DWITH_READLINE=bool`

  Whether to use the `readline` library bundled with the distribution. This option was removed in MySQL 5.6.5 because `readline` is no longer bundled.

- `-DWITH_SSL={ssl_type|path_name}`

- The type of SSL support to include (if any) or the path name to the OpenSSL installation to use.

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

    - `no`: No SSL support. This is the default before MySQL 5.6.6. As of 5.6.6, this is no longer a permitted value and the default is `bundled`.
    - `yes`: Use the system SSL library if present, else the library bundled with the distribution.
    - `bundled`: Use the SSL library bundled with the distribution. This is the default as of MySQL 5.6.6.
    - `system`: Use the system SSL library.

  - *`path_name`*, permitted for MySQL 5.6.7 and after, is the path name to the OpenSSL installation to use. Using this can be preferable to using the *`ssl_type`* value of `system`, for 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 the `CMAKE_PREFIX_PATH` option to *`path_name`*.)

For information about using SSL support, see Using Secure Connections.

- `-DWITH_SYMVER16=`*`bool`*

  If enabled, this option causes the `libmysqlclient` client library to contain extra symbols to be compatible with `libmysqlclient` on RHEL/OEL 5, 6, and 7; and Fedora releases. All symbols present in `libmysqlclient.so.16` are tagged with symver 16 in `libmsqlclient.so.18`, making those symbols have both symver 16 and 18. The default is `OFF`.

  This option was added in MySQL 5.6.31.

- `-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` indicates the source of `zlib` support:

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

- `-DWITHOUT_SERVER=`*`bool`*

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

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`. This option was added in MySQL 5.6.6.

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

- `-DSUNPRO_CXX_LIBRARY="lib_name"`

  Enable linking against `libCstd` instead of `stlport4` on Solaris 10 or later. This works only for client code because the server depends on C++98. Example usage:

  ```
  cmake -DWITHOUT_SERVER=1 -DSUNPRO_CXX_LIBRARY=Cstd
  ```

  This option was added in MySQL 5.6.20.

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:

```
shell> mkdir bld
shell> cd bld
shell> 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:

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

CMake Options for Compiling MySQL Cluster

The following options are for use when building MySQL Cluster with the MySQL Cluster sources; they are not currently supported when using sources from the MySQL 5.6 Server tree.

- `-DMEMCACHED_HOME=dir_name`

  Perform the build using the memcached (version 1.6 or later) installed in the system directory indicated by `dir_name`. Files from this installation that are used in the build include the memcached binary, header files, and libraries, as well as the `memcached_utilities` library and the header file `engine_testapp.h`.

  You must leave this option unset when building `ndbmemcache` using the bundled memcached sources (`WITH_BUNDLED_MEMCACHED` option); in other words, the bundled sources are used by default).

  This option was added in MySQL Cluster NDB 7.2.2.

  While additional `CMake` options—such as for SASL authorization and for providing `dtrace` support—are available for use when compiling `memcached` from external sources, these options are currently not enabled for the `memcached` sources bundled with MySQL Cluster.

- `-DWITH_BUNDLED_LIBEVENT={ON|OFF}`

Use the `libevent` included in the MySQL Cluster sources when building MySQL Cluster with ndbmemcached support (MySQL Cluster NDB 7.2.2 and later). Enabled by default. OFF causes the system's libevent to be used instead.

- `-DWITH_BUNDLED_MEMCACHED={ON|OFF}`

  Build the memcached sources included in the MySQL Cluster source tree (MySQL Cluster NDB 7.2.3 and later), then use the resulting memcached server when building the ndbmemcache engine. In this case, `make install` places the `memcached` binary in the installation `bin` directory, and the ndbmemcache engine shared library file `ndb_engine.so` in the installation `lib` directory.

  This option is ON by default.

- `-DWITH_CLASSPATH=`*`path`*

  Sets the classpath for building MySQL Cluster Connector for Java. The default is empty. In MySQL Cluster NDB 7.2.9 and later, 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_NDBCLUSTER_STORAGE_ENGINE={ON|OFF}`

  Build and link in support for the `NDB` (`NDBCLUSTER`) storage engine in `mysqld`. The default is `ON`.

- `-DWITH_NDBCLUSTER={ON|OFF}`

  This is an alias for `WITH_NDBCLUSTER_STORAGE_ENGINE`.

- `-DWITH_NDBMTD={ON|OFF}`

  Build the multi-threaded data node executable `ndbmtd`. The default is `ON`.

- `-DWITH_NDB_BINLOG={ON|OFF}`

  Enable binary logging by default in the `mysqld` built using this option. ON by default.

- `-DWITH_NDB_DEBUG={ON|OFF}`

  Enable building the debug versions of the MySQL Cluster binaries. OFF by default.

- `-DWITH_NDB_JAVA={ON|OFF}`

  Enable building MySQL Cluster with Java support, including `ClusterJ`.

  This option was added in MySQL Cluster NDB 7.2.9, and is ON by default. If you do not wish to compile MySQL 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 MySQL 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.

4.5 Dealing with Problems Compiling MySQL

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

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts up, 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:

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

On Windows:

```
shell> devenv MySQL.sln /clean
shell> 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:

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

  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*.

4.6 MySQL Configuration and Third-Party Tools

Third-party tools that need to determine the MySQL version from the MySQL source can read the `VERSION` file in the top-level source directory. The file lists the pieces of the version separately. For example, if the version is MySQL 5.7.4-m14, the file looks like this:

```
MYSQL_VERSION_MAJOR=5
MYSQL_VERSION_MINOR=7
MYSQL_VERSION_PATCH=4
MYSQL_VERSION_EXTRA=-m14
```

If the source is not for a General Availablility (GA) release, the `MYSQL_VERSION_EXTRA` value will be nonempty. For the example, the value corresponds to Milestone 14.

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

```
MYSQL_VERSION_MAJOR*10000 + MYSQL_VERSION_MINOR*100 + MYSQL_VERSION_PATCH
```

Chapter 5 Installing MySQL on Microsoft Windows

Table of Contents

There are several different methods to install MySQL on Microsoft Windows.

Simple Installation Method

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure all of the MySQL products on your system. Here is how:

- Download MySQL Installer from http://dev.mysql.com/downloads/installer/ and execute it.

  > **Note**
  >
  > Unlike the standard MySQL Installer, the smaller "web-community" version does not bundle any MySQL applications but it will download the MySQL products you choose to install.

- Choose the appropriate **Setup Type** for your system. Typically you will choose **Developer Default** to install MySQL server and other MySQL tools related to MySQL development, helpful tools like MySQL Workbench. Or, choose the **Custom** setup type to manually select your desired MySQL products.

  > **Note**
  >
  > Multiple versions of MySQL server can exist on a single system. You can choose one or multiple versions.

- Complete the installation process by following the MySQL Installation wizard's instructions. This will install several MySQL products and start the MySQL server.

- MySQL is now installed. You probably configured MySQL as a service that will automatically start MySQL server every time you restart your system.

> **Note**
>
> You probably also installed other helpful MySQL products like MySQL Workbench and MySQL Notifier on your system. Consider loading MySQL Workbench to check your new MySQL server connection, and Section 5.4, “MySQL Notifier” to view the connection's status. By default, these two programs automatically start after installing MySQL.

This process also installs the MySQL Installer application on your system, and later you can use MySQL Installer to upgrade or reconfigure your MySQL products.

Additional Installation Information

MySQL is available for Microsoft Windows, for both 32-bit and 64-bit versions. For supported Windows platform information, see http://www.mysql.com/support/supportedplatforms/database.html.

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see Section 5.5.7, “Starting MySQL as a Windows Service”.

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`. Once 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 Windows Platform Limitations.

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

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

  > **Note**
  >
  > MySQL Installer will install and configure Connector/ODBC for you.

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

  > **Note**
  >
  > MySQL Installer will install and configure Connector/NET for you.

MySQL distributions for Windows can be downloaded from http://dev.mysql.com/downloads/. See Section 2.2, “How to Get MySQL”.

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

- A MySQL Installer distribution includes MySQL Server and additional MySQL products including MySQL Workbench, MySQL Notifier, and MySQL for Excel. MySQL Installer can also be used to upgrade these products in the future.

  For instructions on installing MySQL using MySQL Installer, see Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”.

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

  For instructions on installing MySQL using the Zip file, see Section 5.5, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”.

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

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

MySQL on Windows considerations:

- **Large Table Support**

  If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use `MAX_ROWS` and `AVG_ROW_LENGTH` when you create tables. See CREATE TABLE Syntax.

- **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 5.5.2, “Creating an Option File”.

5.1 MySQL Installation Layout on Microsoft Windows

For MySQL 5.6 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.6`. Some Windows users prefer to install in `C:\mysql`, the directory that formerly was used as the default. However, the layout of the subdirectories remains the same.

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

Table 5.1 Default MySQL Installation Layout for Microsoft Windows

Directory	Contents of Directory	Notes
`bin`, `scripts`	`mysqld` server, client and utility programs	
`%ALLUSERSPROFILE%\MySQL\MySQL Server 5.6\`	Log files, databases (Windows XP, Windows Server 2003)	The Windows system variable `%ALLUSERSPROFILE%` defaults to `C:\Documents and Settings\All Users\Application Data`
`%PROGRAMDATA%\MySQL\MySQL Server 5.6\`	Log files, databases (Vista, Windows 7, Windows Server 2008, and newer)	The Windows system variable `%PROGRAMDATA%` defaults to `C:\ProgramData`
`examples`	Example programs and scripts	
`include`	Include (header) files	
`lib`	Libraries	
`share`	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

If you install MySQL using the MySQL Installer, this package creates and sets up the data directory that the installed server will use, and also creates a pristine “template” data directory named `data` under the installation directory. After an installation has been performed using this package, the template data directory can be copied to set up additional MySQL instances. See Running Multiple MySQL Instances on One Machine.

5.2 Choosing An Installation Package

For MySQL 5.6, there are multiple installation package formats to choose from when installing MySQL on Windows.

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

- **MySQL Installer**: This package has a file name similar to `mysql-installer-community-5.6.35.0.msi` or `mysql-installer-commercial-5.6.35.0.msi`, and utilizes MSIs to automatically install MySQL server and other products. It will download and apply updates to itself, and for each of the installed products. It also configures the additional non-server products.

  The installed products are configurable, and this includes: documentation with samples and examples, connectors (such as C, C++, J, NET, and ODBC), MySQL Workbench, MySQL Notifier, MySQL for Excel, and the MySQL Server with its components.

  MySQL Installer will run on all Windows platforms that are supported by MySQL (see http://www.mysql.com/support/supportedplatforms/database.html).

> **Note**
>
> Because MySQL Installer is not a native component of Microsoft Windows and depends on .NET, it will not work on minimal installation options like the "Server Core" version of Windows Server 2008.

For instructions on installing MySQL using MySQL Installer, see Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”.

- **The Noinstall Archive**: This package has a file name similar to `mysql-5.6.35-win32.zip` or `mysql-5.6.35-winx64.zip`, and contains all the files found in the Complete install package, with the exception of the GUI. It also contains PDB files. This package does not include an automated installer, and must be manually installed and configured.

MySQL Installer is recommended for most users.

Your choice of install package affects the installation process you must follow. If you choose to use MySQL Installer, see Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”. If you choose to install a Noinstall archive, see Section 5.5, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”.

5.3 Installing MySQL on Microsoft Windows Using MySQL Installer

MySQL Installer is an application that manages MySQL products on Microsoft Windows. It installs, updates, removes, and configures MySQL products, and remains on the system as its own application. MySQL Installer is only available for Microsoft Windows, and includes both GUI and command-line interfaces.

The supported MySQL products include:

- MySQL Server (one or multiple versions on the same system)
- MySQL Workbench
- MySQL Connectors (.Net / Python / ODBC / Java / C / C++)
- MySQL Notifier
- MySQL for Excel
- MySQL for Visual Studio
- MySQL Utilities and MySQL Fabric
- MySQL Samples and Examples
- MySQL Documentation
- MySQL Installer is also installed and remains on the system as its own application, that is used to install additional MySQL products, and also to update and configure existing MySQL products
- The Enterprise edition installs the Enterprise versions of the above products, and also includes MySQL Enterprise Backup and MySQL Enterprise Firewall

Installer Package Types

- `Full:` Bundles all of the MySQL products (including the MySQL server). The file size is over 300MB, and its name has the form `mysql-installer-community-VERSION.N.msi` where `VERSION` is the MySQL Server version number such as 5.7 and `N` is the package number, which begins at 0.
- `Web:` Only contains MySQL Installer and configuration files, and it downloads the MySQL products you choose to install. The size of this file is about 2MB; the name of the file has the form `mysql-installer-community-web-VERSION.N.msi` where `VERSION` is the MySQL Server version number such as 5.7 and `N` is the package number, which begins at 0.
- `Updates:` MySQL Installer can upgrade itself, so an additional download is not requires to update MySQL Installer.

Installer Editions

- `Community edition:` Downloadable at http://dev.mysql.com/downloads/installer/. It installs the community edition of all MySQL products.
- `Commercial edition:` Downloadable at either My Oracle Support (MOS) or https://edelivery.oracle.com/. It installs the commercial version of all MySQL products, including Workbench SE/EE, MySQL Enterprise Backup, and MySQL Enterprise Firewall. It also integrates with your MOS account.

  > **Note**
  >
  > Entering your MOS credentials is optional when installing bundled MySQL products, but your credentials are required when choosing non-bundled MySQL products that MySQL Installer must download.

For notes detailing the changes in each release of MySQL Installer, see MySQL Installer Release Notes.

MySQL Installer is compatible with pre-existing installations, and adds them to its list of installed components. While the standard MySQL Installer is bundled with a specific version of MySQL server, a single MySQL Installer instance can install and manage multiple MySQL server versions. For example, a single MySQL Installer instance can install (and update) versions 5.5, 5.6, and 5.7 on the same host.

> **Note**
>
> A single host *cannot* have both community and commercial editions of MySQL server installed. For example, if you want both MySQL Server 5.6 and 5.7 installed on a single host, both must be the same edition.

MySQL Installer handles the initial configuration and set up of the applications. 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.

  > **Note**
  >
  > Some definitions are host dependent. For example, query_cache is enabled if the host has fewer than three cores.

- It can optionally import example databases.
- By default, a Windows service for the MySQL server is added.

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

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

- Checking **Show Advanced Options** allows 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.

MySQL Installer can optionally check for updated components and download them for you.

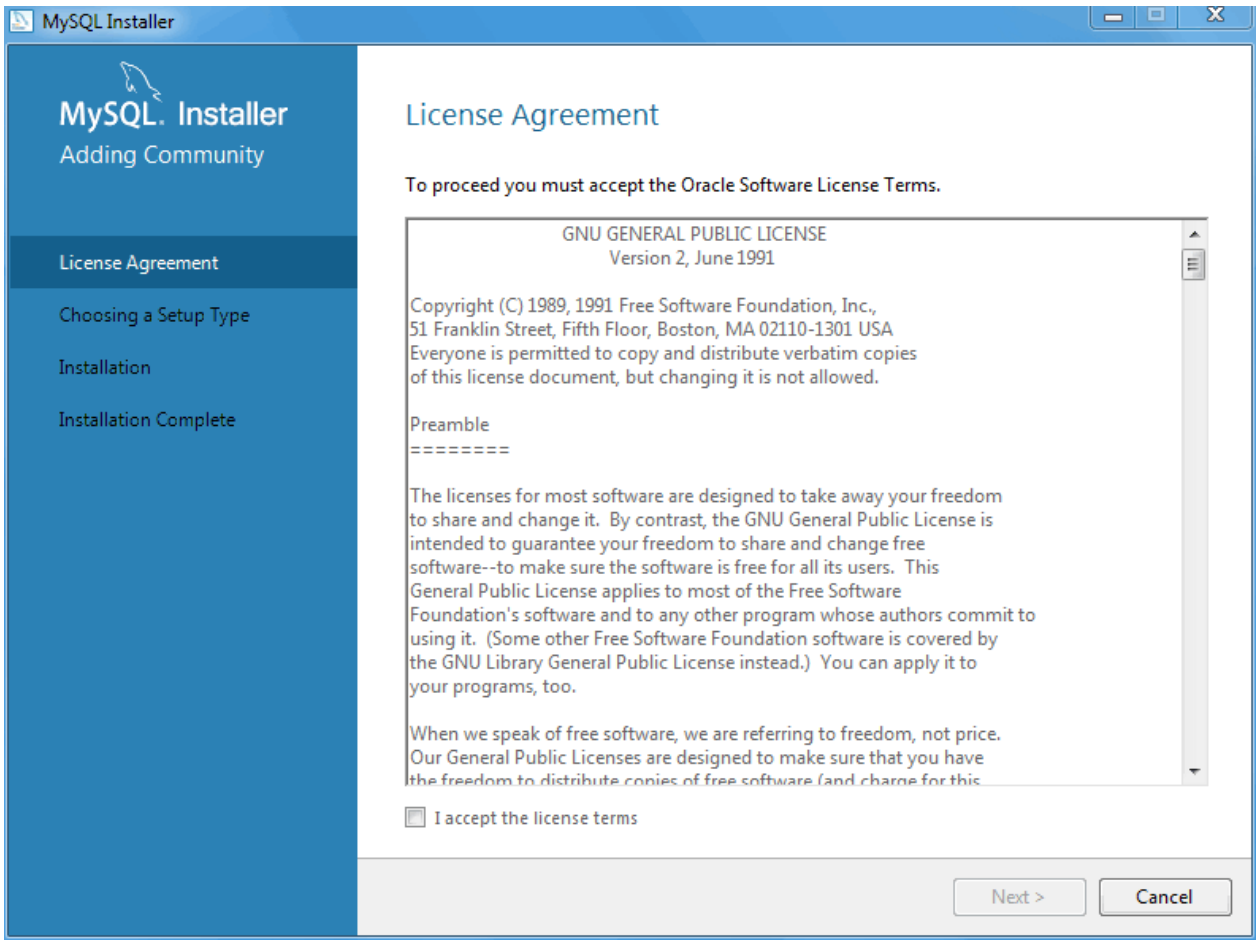
5.3.1 MySQL Installer GUI

Installing MySQL Installer adds a link to the Start menu under the **MySQL** group. Click **Start**, **All Programs**, **MySQL**, **MySQL Installer** to reload the MySQL Installer GUI.

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

MySQL Installer requires you to accept the license agreement before it will install MySQL products.

Figure 5.1 MySQL Installer - License Agreement



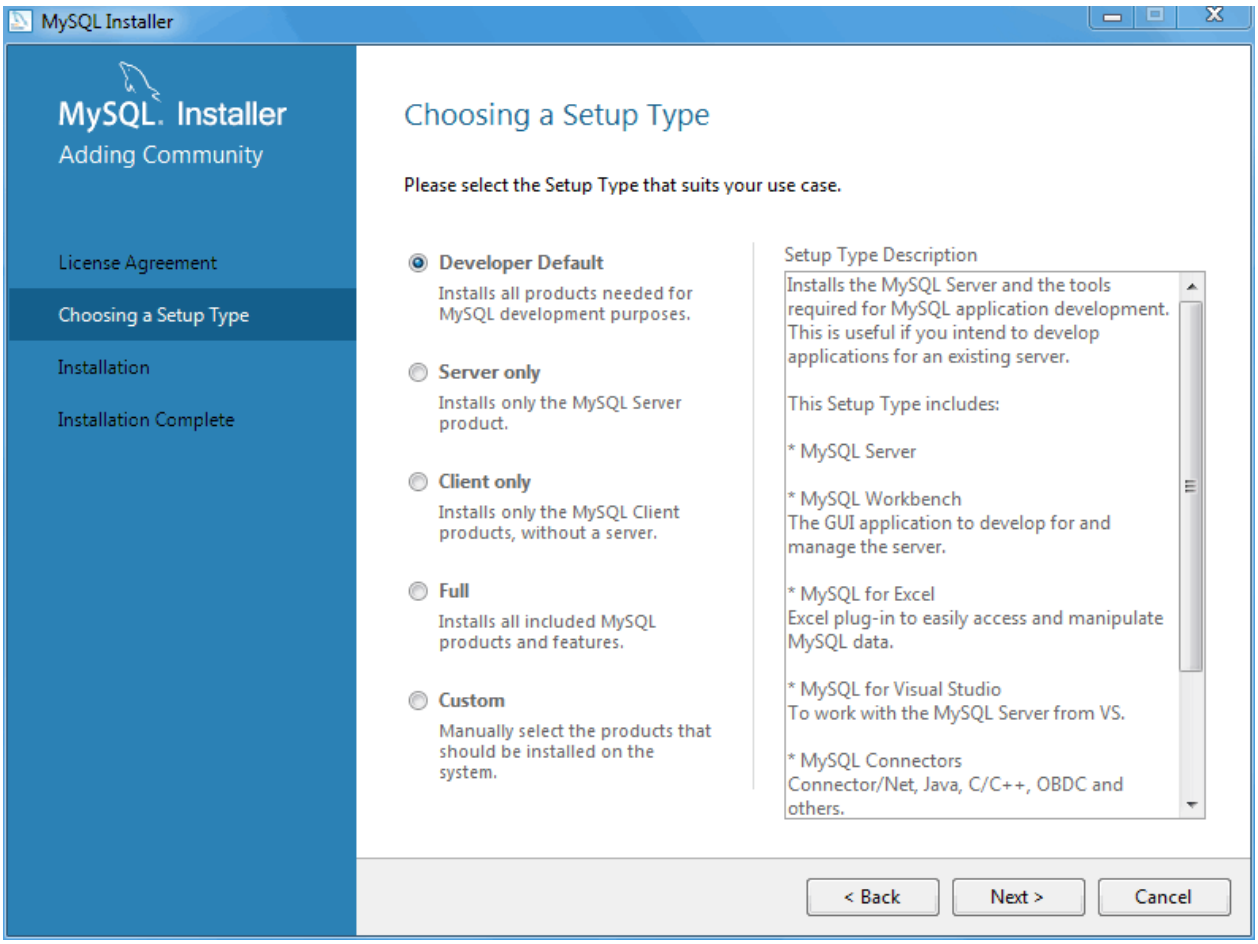
Installing New Packages

Choose the appropriate **Setup Type** for your system. This type determines which MySQL products are initially installed on your system, or select **Custom** to manually choose the products.

- **Developer**: Install all products needed to develop applications with MySQL. This is the default option.
- **Server only**: Only install the MySQL server.
- **Client only**: Only install the MySQL client products, such as MySQL Workbench. This does not include the MySQL server.
- **Full**: Install all available MySQL products.
- **Custom**: Manually select the MySQL products to install, and optionally configure custom MySQL data and installation paths.

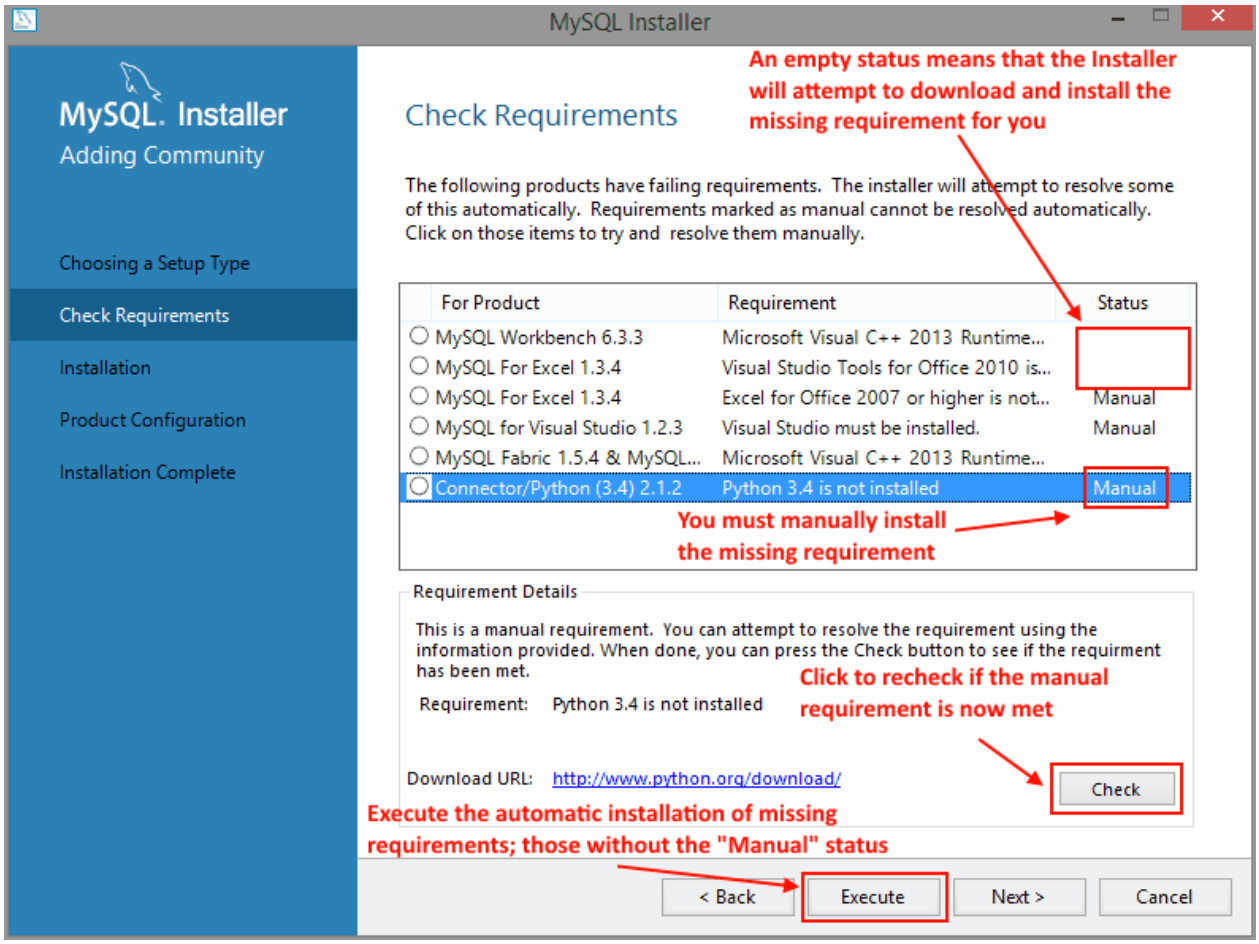
> **Note**
>
> After the initial installation, you may use MySQL Installer to manually select MySQL products to install or remove. In other words, MySQL Installer becomes a MySQL product management system.

Figure 5.2 MySQL Installer - Choosing a Setup Type



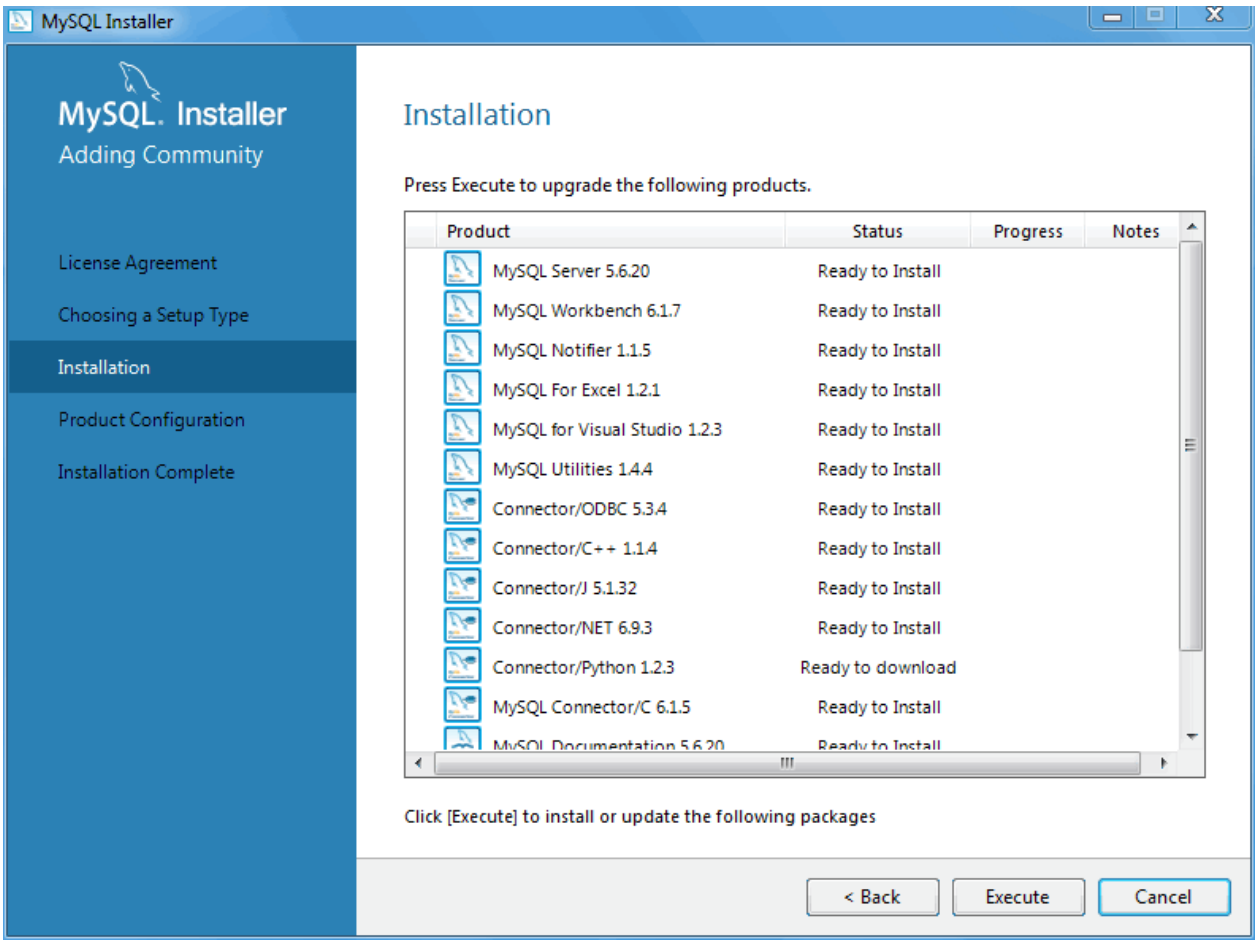
MySQL Installer checks your system for the external requirements (prerequisites) required to install the selected MySQL products. MySQL Installer can download and install some prerequisites, but others require manual intervention. Download and install all prerequisites that have **Status** set to "Manual". Click **Check** to recheck if a manual prerequisite was installed. After manually installing those requirements, click **Execute** to download and install the other prerequisites. Once finished, click **Next** to continue.

Figure 5.3 MySQL Installer - Check Requirements



The next window lists the MySQL products that are scheduled for installation:

Figure 5.4 MySQL Installer - Installation Progress



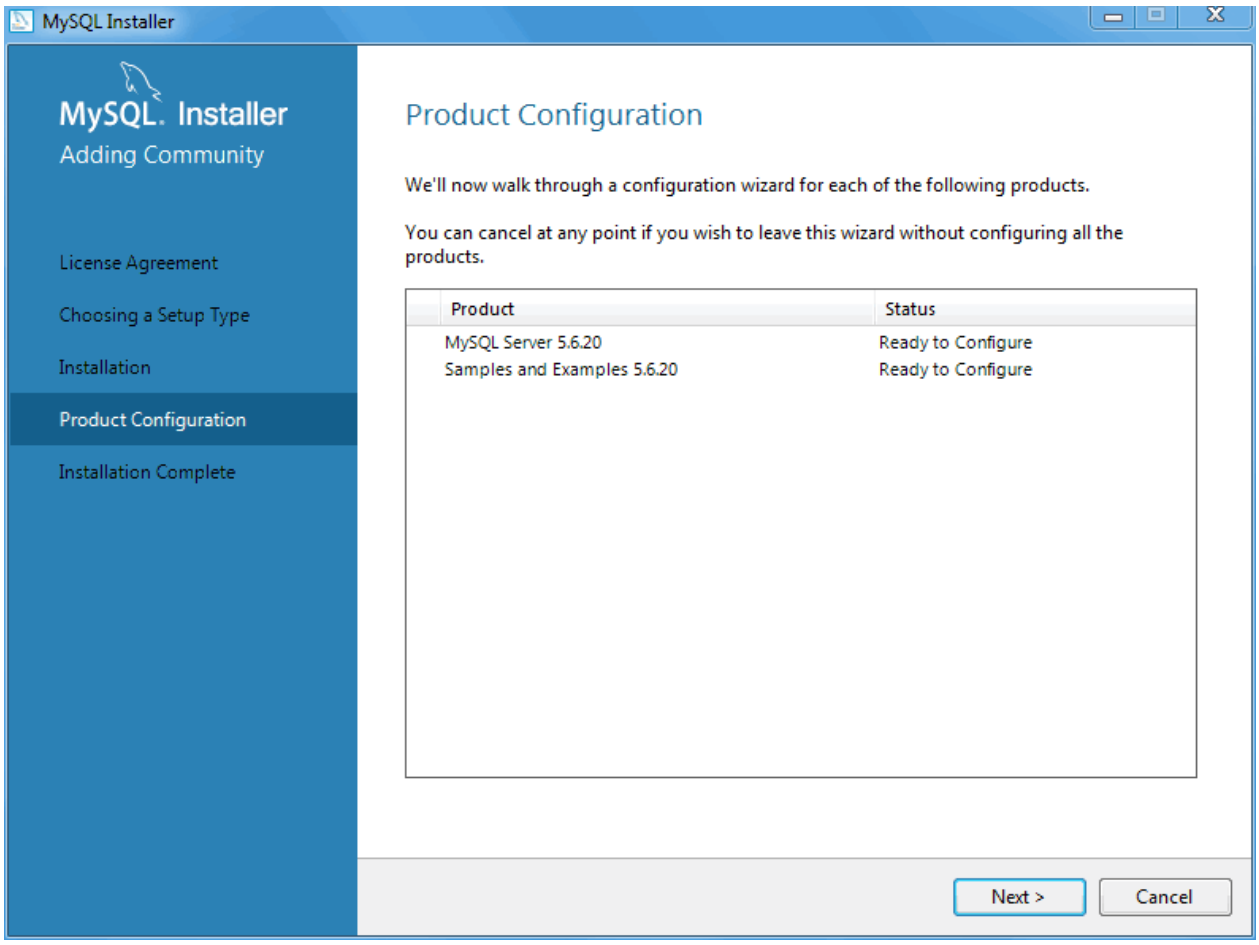
As components are installed, their **Status** changes from a progress percentage to "Complete".

After all components are installed, the next step configures some of the recently installed MySQL products. The `Configuration Overview` window displays the progress and then loads a configuration window, if required. Our example configures MySQL Server 5.6.x.

Configuring MySQL Server

Configuring the MySQL server begins with defining several **Type and Networking** options.

Figure 5.5 MySQL Installer - Configuration Overview



Server Configuration Type

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

- **Developer**: A machine that will host many other applications, and typically this is your personal workstation. This option configures MySQL to use the least amount of memory.
- **Server**: Several other applications will be running on this machine, such as a web server. This option configures MySQL to use a medium amount of memory.
- **Dedicated**: A machine that is dedicated to running the MySQL server. Because no other major applications will run on this server, such as a web server, this option configures MySQL to use the majority of available memory.

Connectivity

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

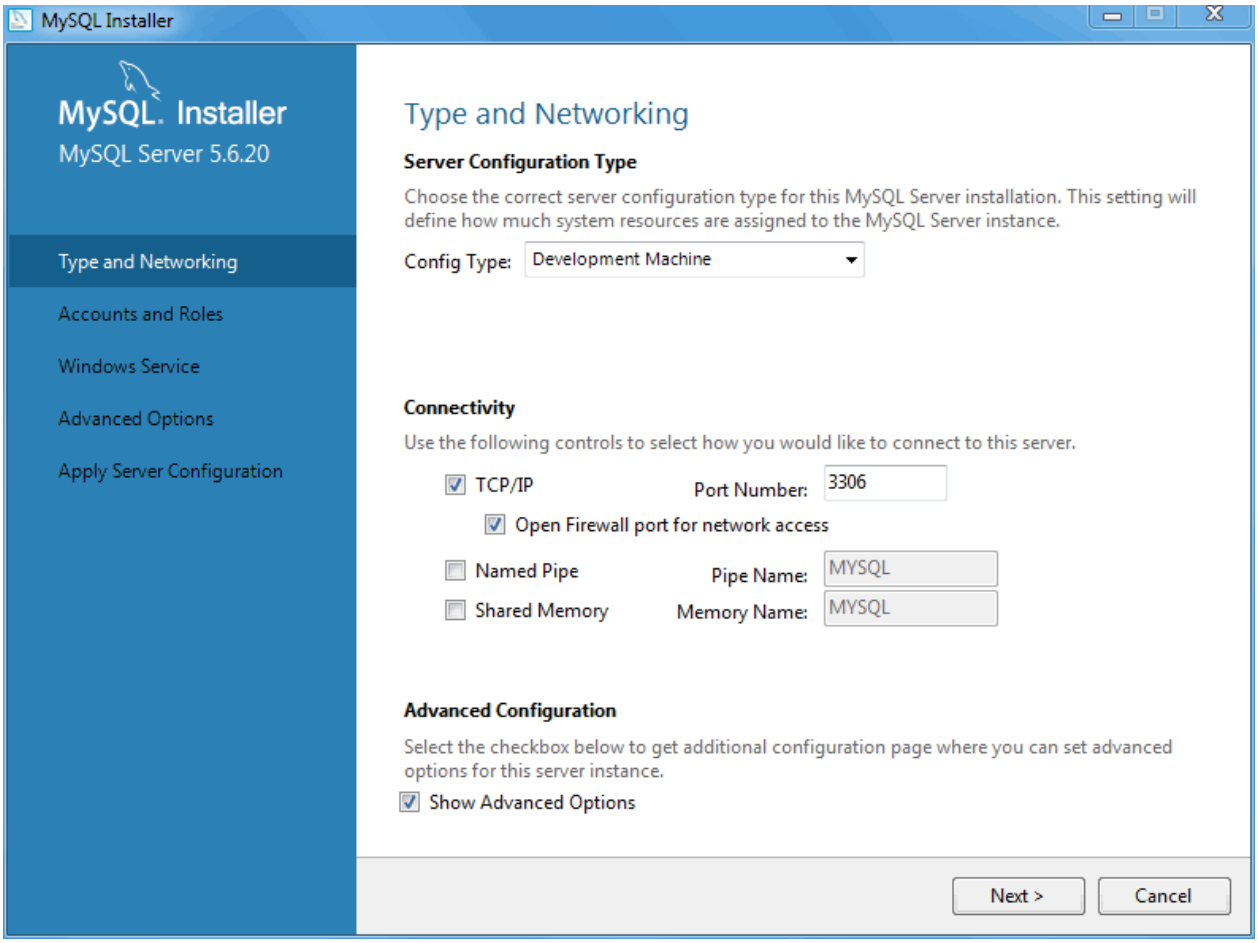
- **TCP/IP**: You may enable TCP/IP Networking here as otherwise only localhost connections are allowed. Also define the **Port Number** and whether to open the firewall port for network access.
- **Named Pipe**: Enable and define the pipe name, similar to using the `--enable-named-pipe` option.

- **Shared Memory**: Enable and then define the memory name, similar to using the `--shared-memory` option.

Advanced Configuration

Check **Show Advanced Options** to set additional **Logging Options**. 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.

Figure 5.6 MySQL Installer - MySQL Server Configuration: Type and Networking



Accounts and Roles

Next, define your MySQL account information. Assigning a root password is required.

Optionally, you can add additional MySQL user accounts with predefined user roles. Each predefined role, such as "DB Admin", are configured with their own set of privileges. For example, the "DB Admin" role has more privileges than the "DB Designer" role. Click the **Role** dropdown for a list of role descriptions.

> **Note**
>
> If the MySQL Server is already installed, then you must also enter the `Current Root Password`.

Figure 5.7 MySQL Installer - MySQL Server Configuration: User Accounts and Roles

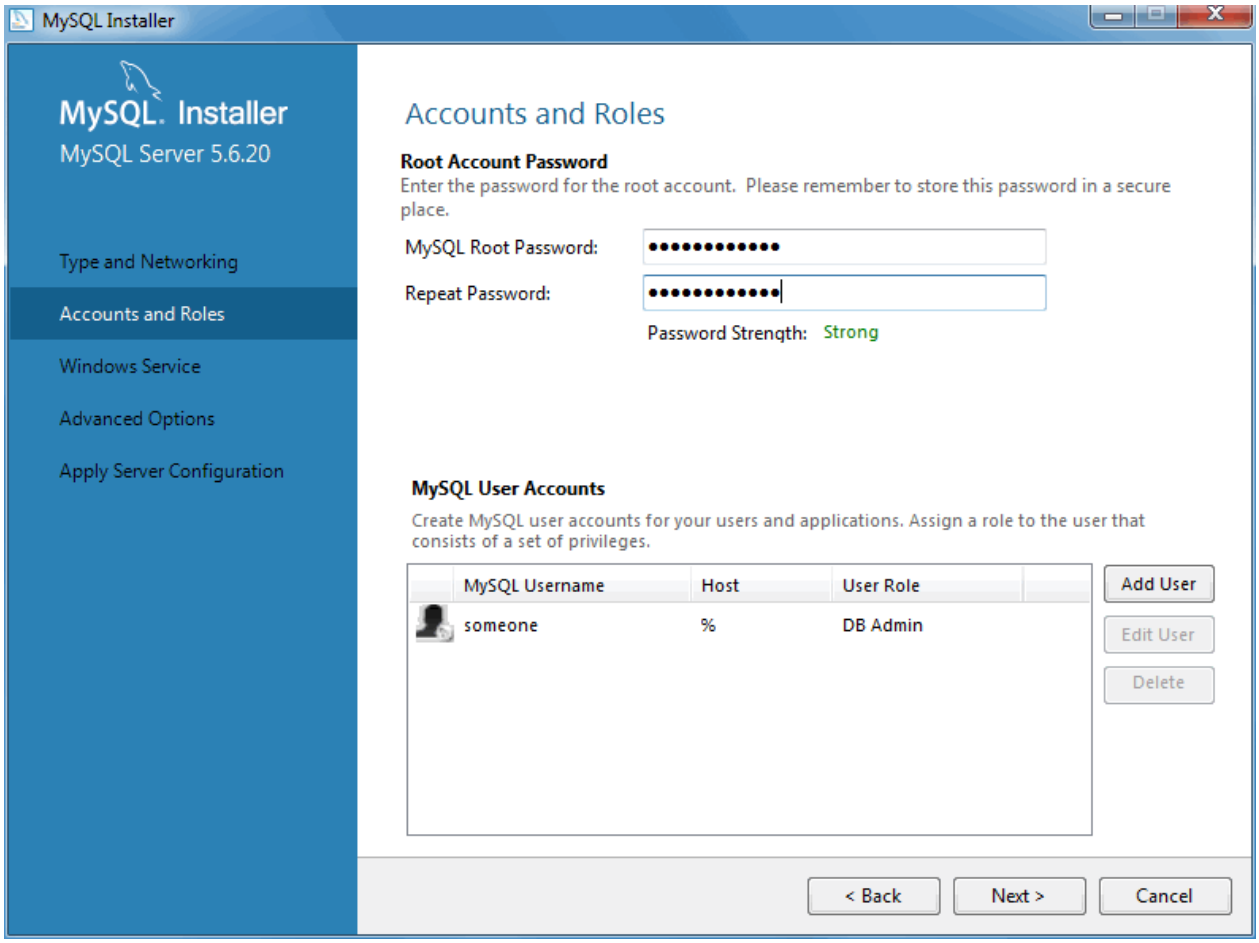
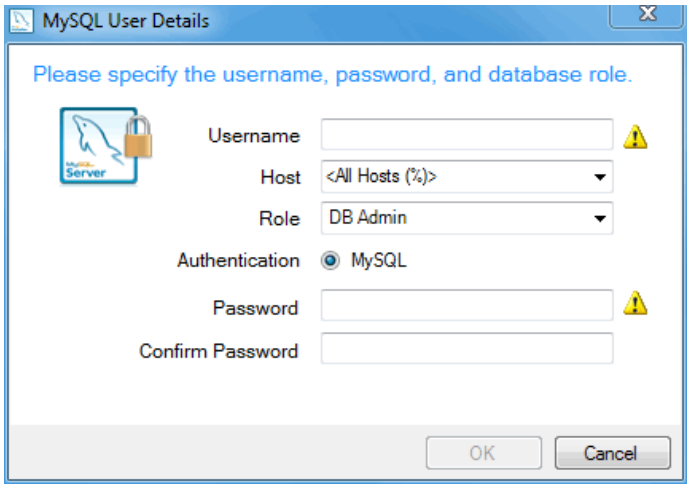


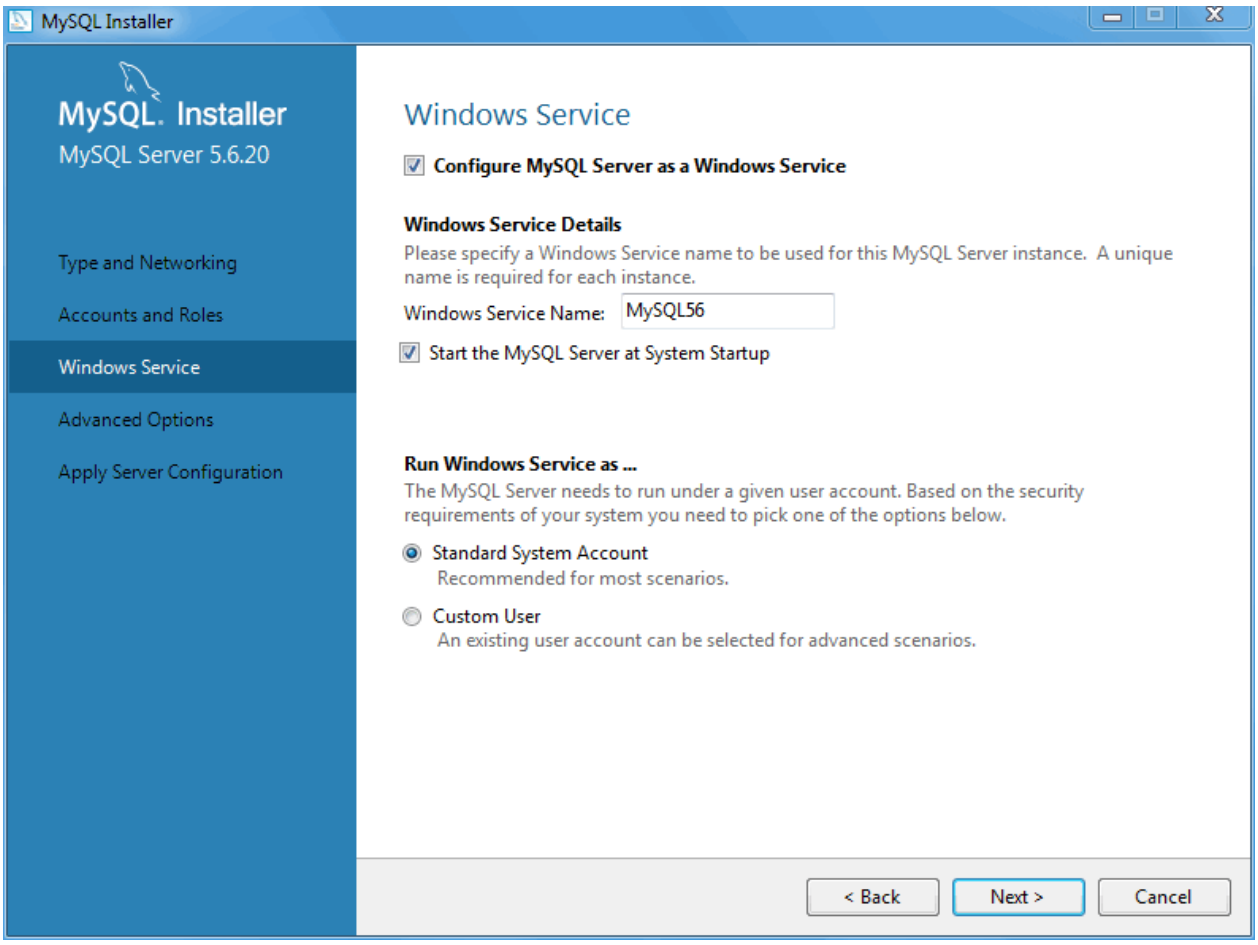
Figure 5.8 MySQL Installer - MySQL Server Configuration: User Accounts and Roles: Adding a User



Windows Service

Next, configure the **Windows Service** details. This includes the service name, whether the MySQL server should be loaded at startup, and how the MySQL server Windows service is executed.

Figure 5.9 MySQL Installer - MySQL Server Configuration: Windows Service

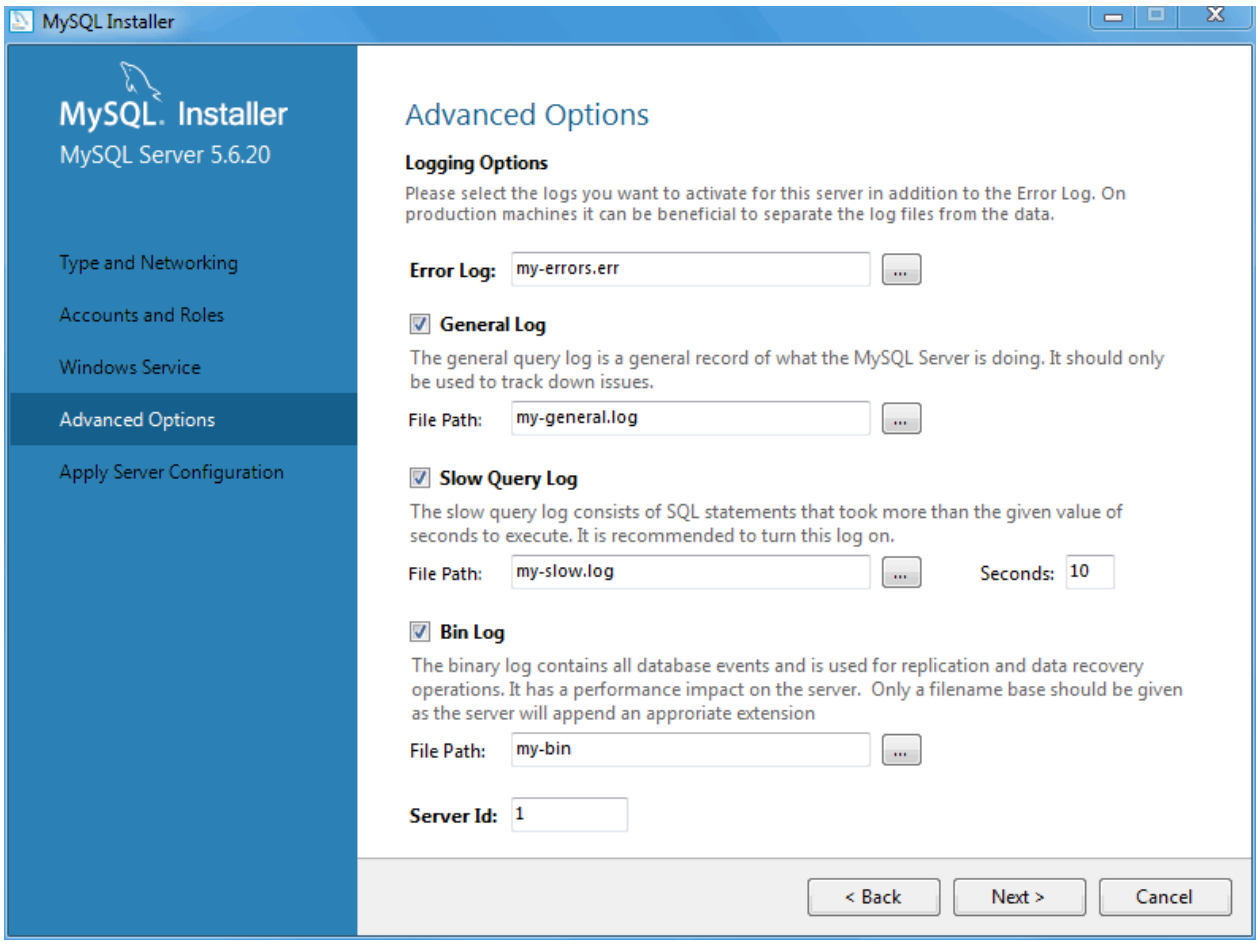


> **Note**
>
> When configuring **Run Windows Services as ...** using a **Custom User**, the custom user must have privileges to log on to Microsoft Windows as a service. The **Next** button will be disabled until this user is configured with the required privileges.
>
> On Microsoft Windows 7, this is configured by loading the `Start Menu`, `Control Panel`, `Administrative Tools`, `Local Security Policy`, `Local Policies`, `User Rights Assignment`, then `Log On As A Service`. Choose `Add User or Group` here to add the custom user, and then **OK**, **OK** to save.

Advanced Options

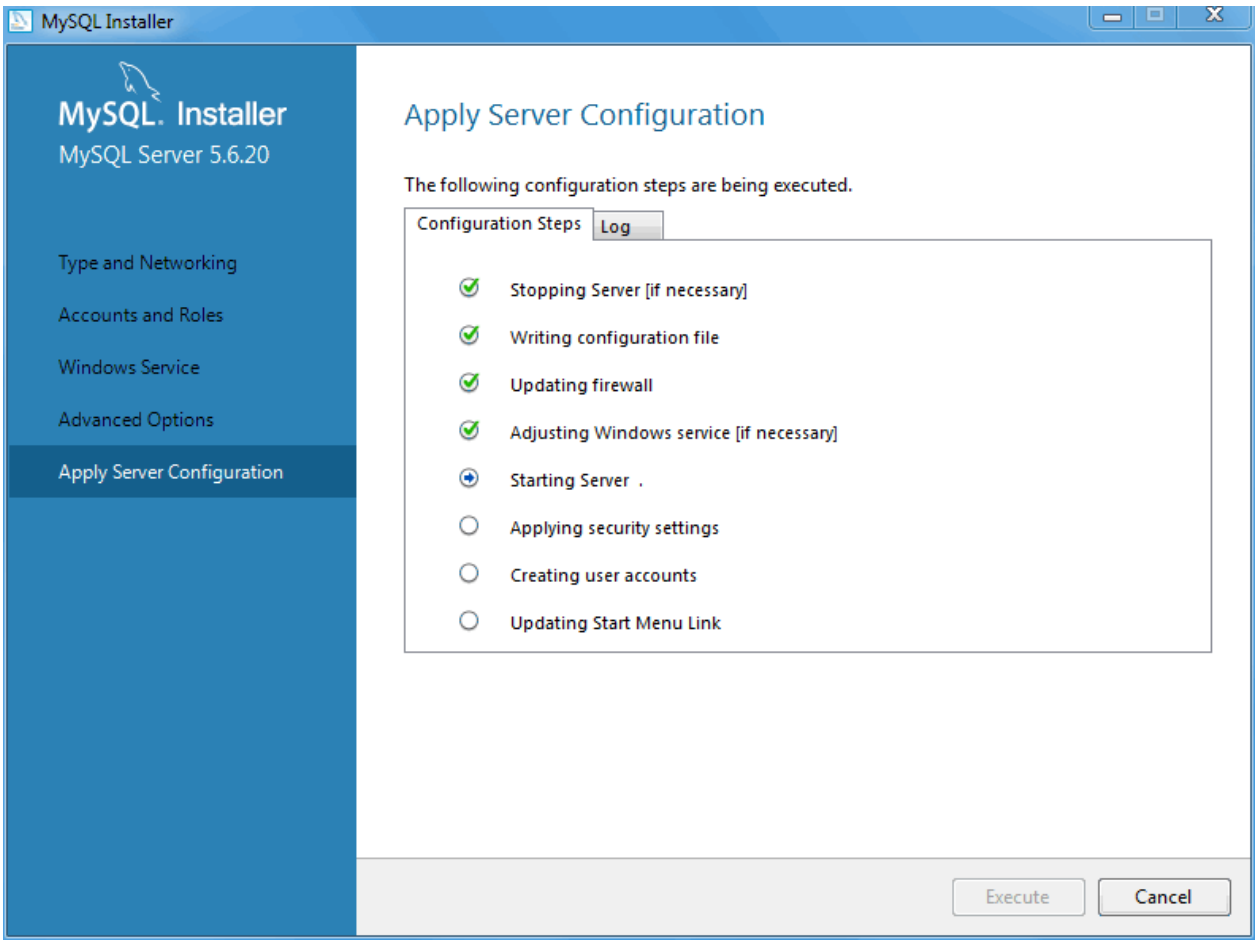
The next configuration step is available if the **Advanced Configuration** option was checked. This section includes options that are related to the MySQL log files:

Figure 5.10 MySQL Installer - MySQL Server Configuration: Logging Options



Click **Next** to continue on to the final page before all of the requested changes are applied. This **Apply Server Configuration** page details the configuration steps that will be performed.

Figure 5.11 MySQL Installer - MySQL Server Configuration: Apply Server Configuration

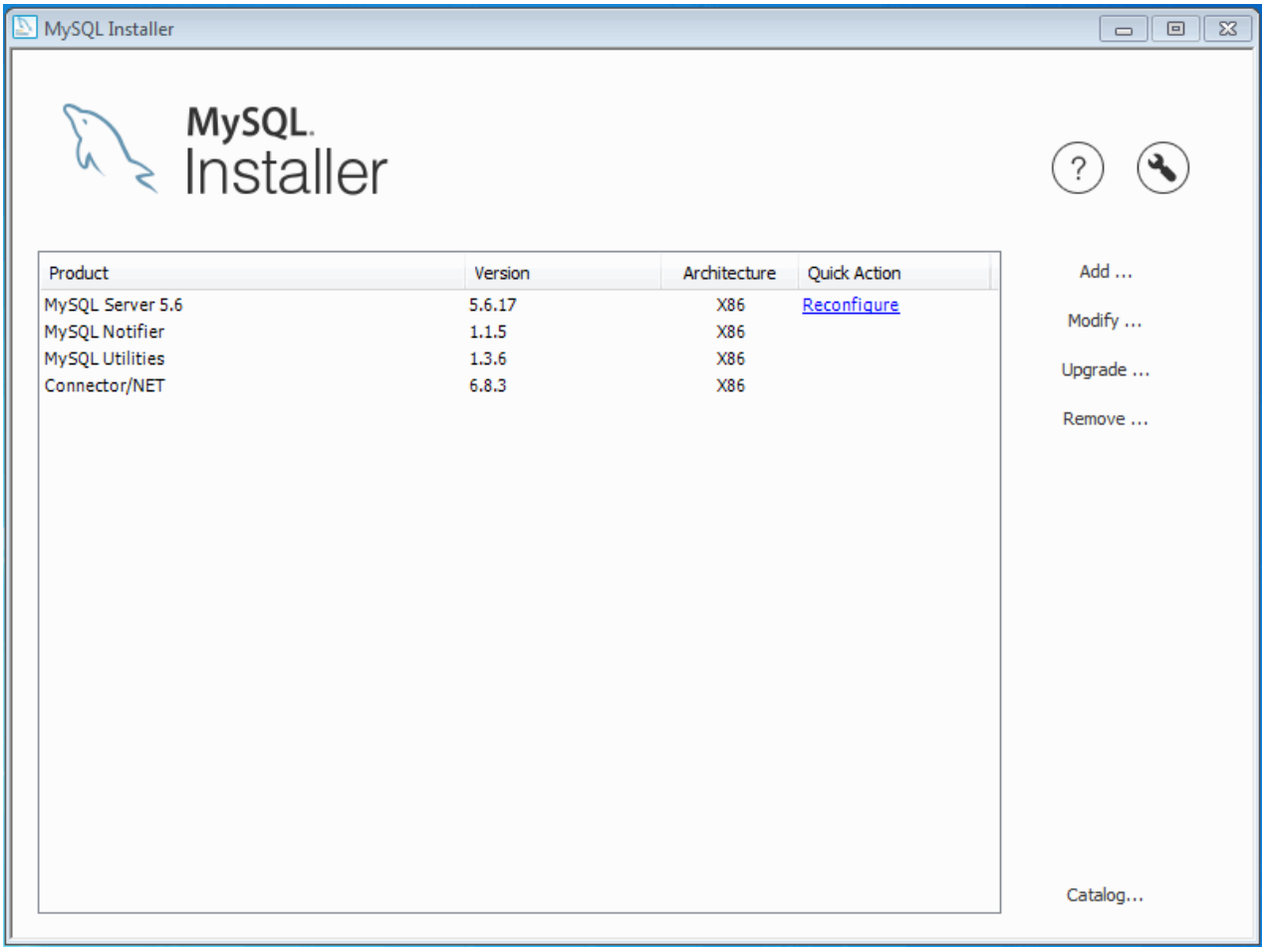


Click **Execute** to execute the configuration steps. The icon for each step toggles from white to green on success, or the process stops on failure. Click the **Log** tab to view the log.

After the MySQL Installer configuration process is finished, MySQL Installer reloads the opening page where you can execute other installation and configuration related actions.

MySQL Installer is added to the Microsoft Windows Start menu under the `MySQL` group. Opening MySQL Installer loads its dashboard where installed MySQL products are listed, and other MySQL Installer actions are available:

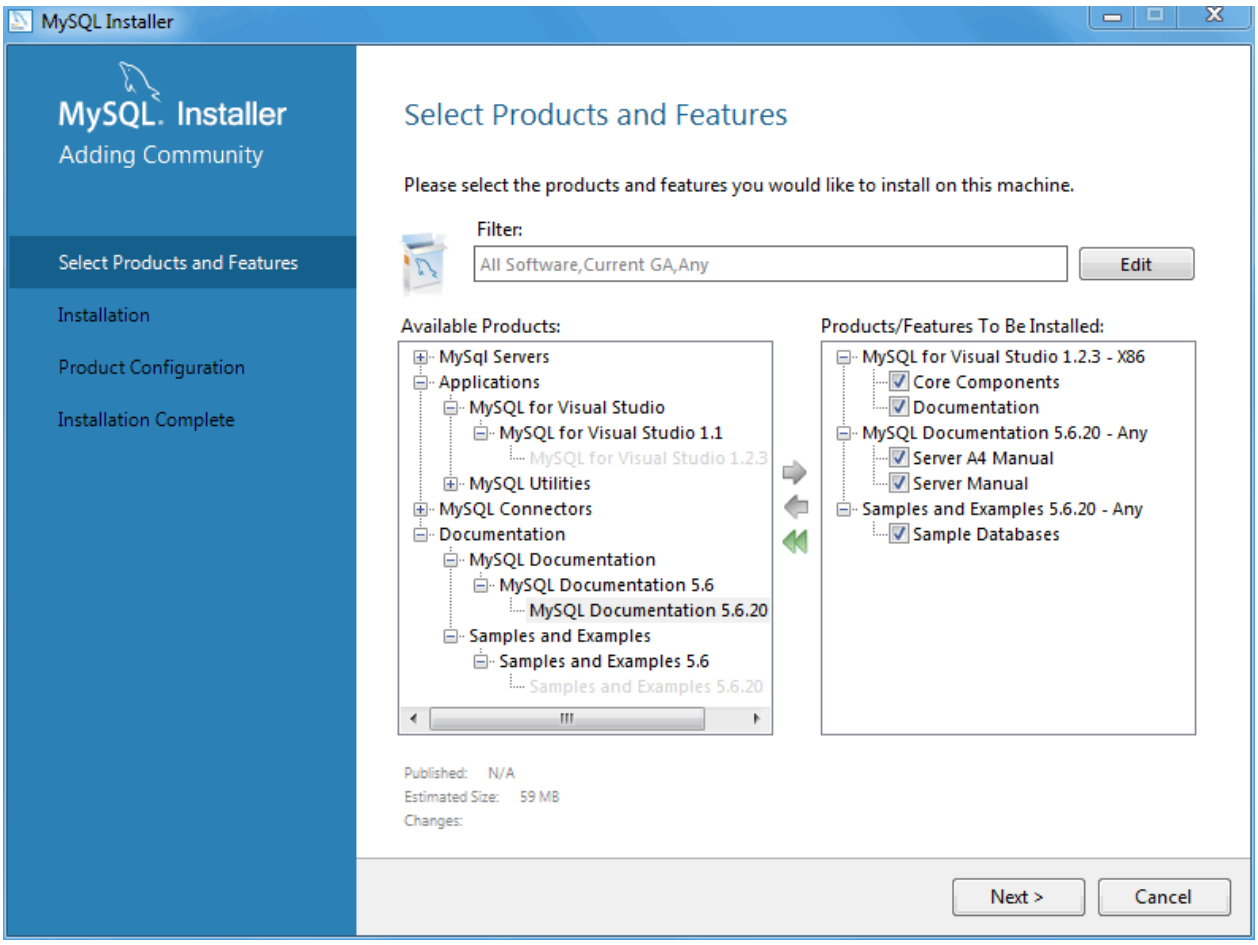
Figure 5.12 MySQL Installer - Main Dashboard



Adding MySQL Products

Click **Add** to add new products. This loads the **Select Products and Features** page:

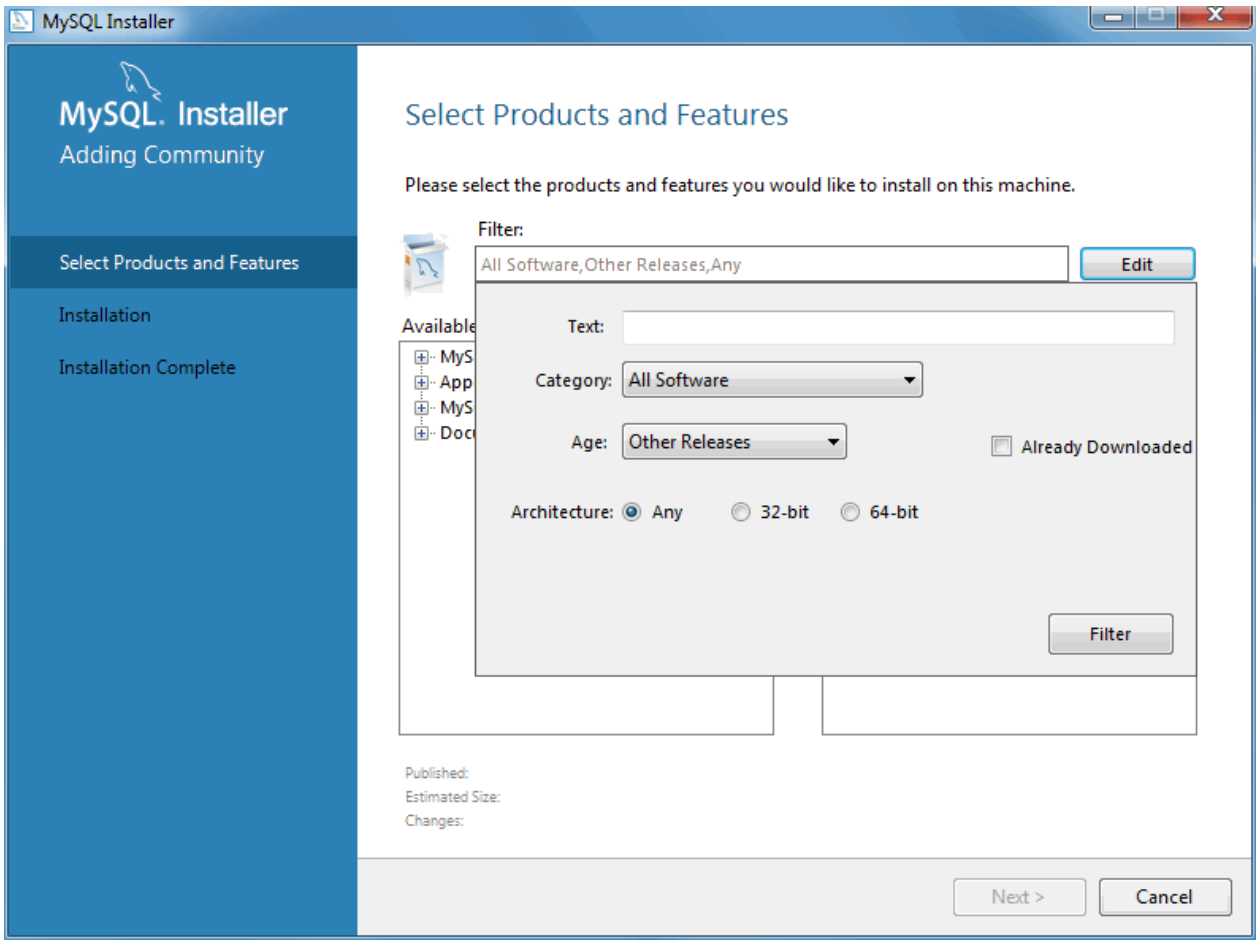
Figure 5.13 MySQL Installer - Select Products and Features



From here, choose the MySQL products you want to install from the left **Available Products** pane, and then click the green right arrow to queue products for installation.

Optionally, click **Edit** to open the product and features search filter:

Figure 5.14 MySQL Installer - Select Products and Features Filter



For example, you might choose to include Pre-Release products in your selections, such as a Beta product that has not yet reached General Availability (GA) status.

Select all of the MySQL products you want to install, then click **Next** to continue using the defaults, or highlight a selected product and click **Advanced Options** to optionally alter options such as the MySQL server data and installation paths. Click **Execute** to execute the installation process to install all of the selected products.

5.3.1.1 MySQL Product Catalog

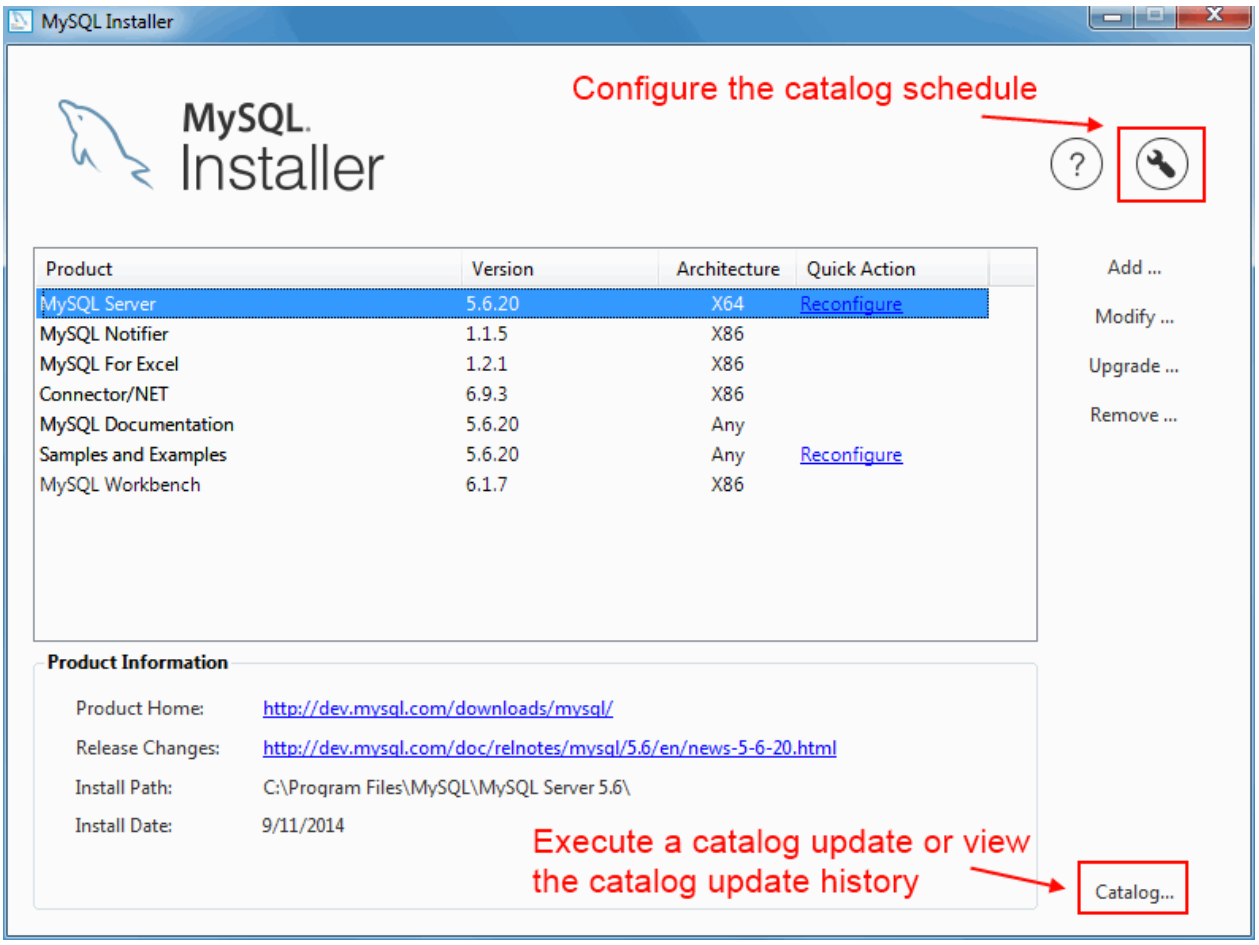
MySQL Installer stores a MySQL product catalog. The catalog can be updated either manually or automatically, and the catalog change history is also available. The automatic update is enabled by default.

> **Note**
>
> The product catalog update also checks for a newer version of MySQL Installer, and prompts for an update if one is present.

Manual updates

You can update the MySQL product catalog at any time by clicking **Catalog** on the Installer dashboard.

Figure 5.15 MySQL Installer - Open the MySQL Product Catalog



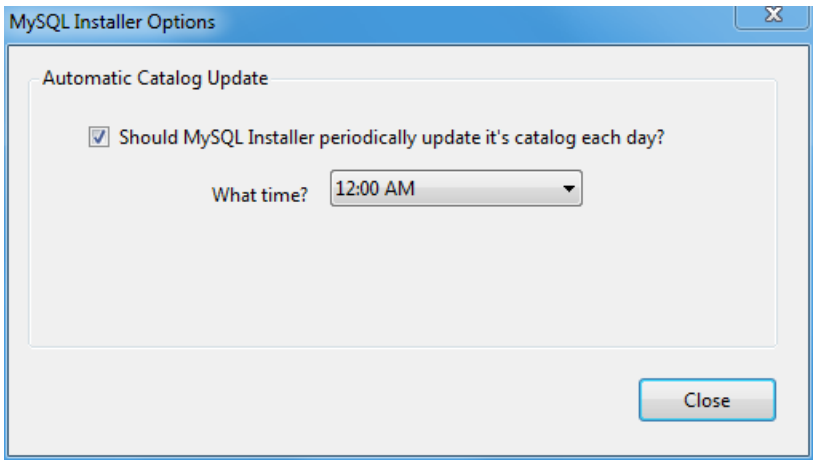
From there, click **Execute** to update the product catalog.

Automatic updates

MySQL Installer can automatically update the MySQL product catalog. By default, this feature is enabled to execute each day at 12:00 AM. To configure this feature, click the wrench icon on the Installer dashboard.

The next window configures the **Automatic Catalog Update**. Enable or disable this feature, and also set the hour.

Figure 5.16 MySQL Installer - Configure the Catalog Scheduler

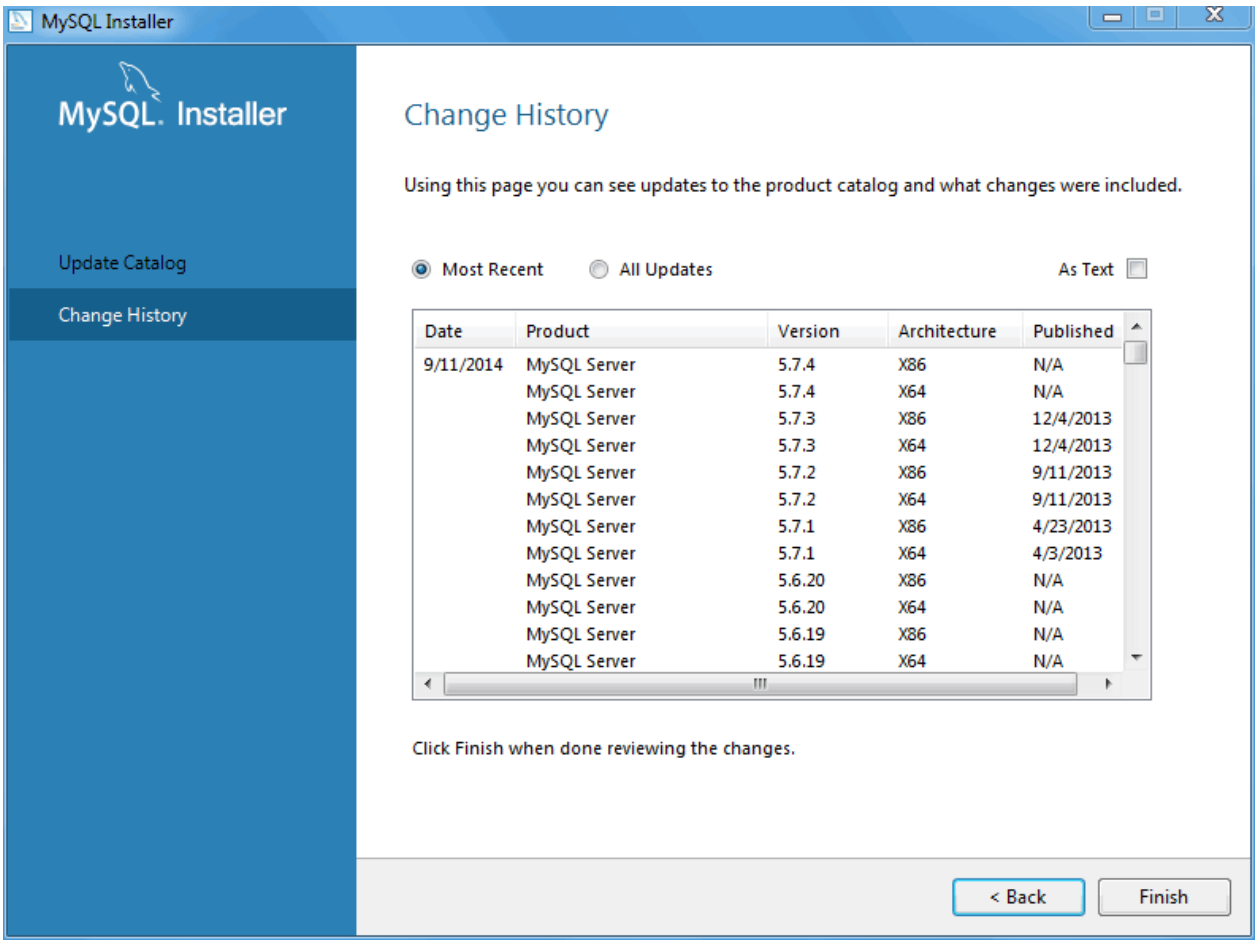


This option uses the Windows Task Scheduler to schedule a task named "ManifestUpdate".

Change History

MySQL Installer tracks the change history for all of the MySQL products. Click **Catalog** from the dashboard, optionally update the catalog (or, toggle the **Do not update at this time** checkbox), click **Next**/**Execute**, and then view the change history.

Figure 5.17 MySQL Installer - Catalog Change History



5.3.1.2 Remove MySQL Products

MySQL Installer can also remove MySQL products from your system. To remove a MySQL product, click **Remove** from the Installer dashboard. This opens a window with a list of installed MySQL products. Select the MySQL products you want to remove (uninstall), and then click **Execute** to begin the removal process.

> **Note**
>
> To select all MySQL products, click the **[ ]** checkbox to the left of the **Product** label.

Figure 5.18 MySQL Installer - Removing Products: Select

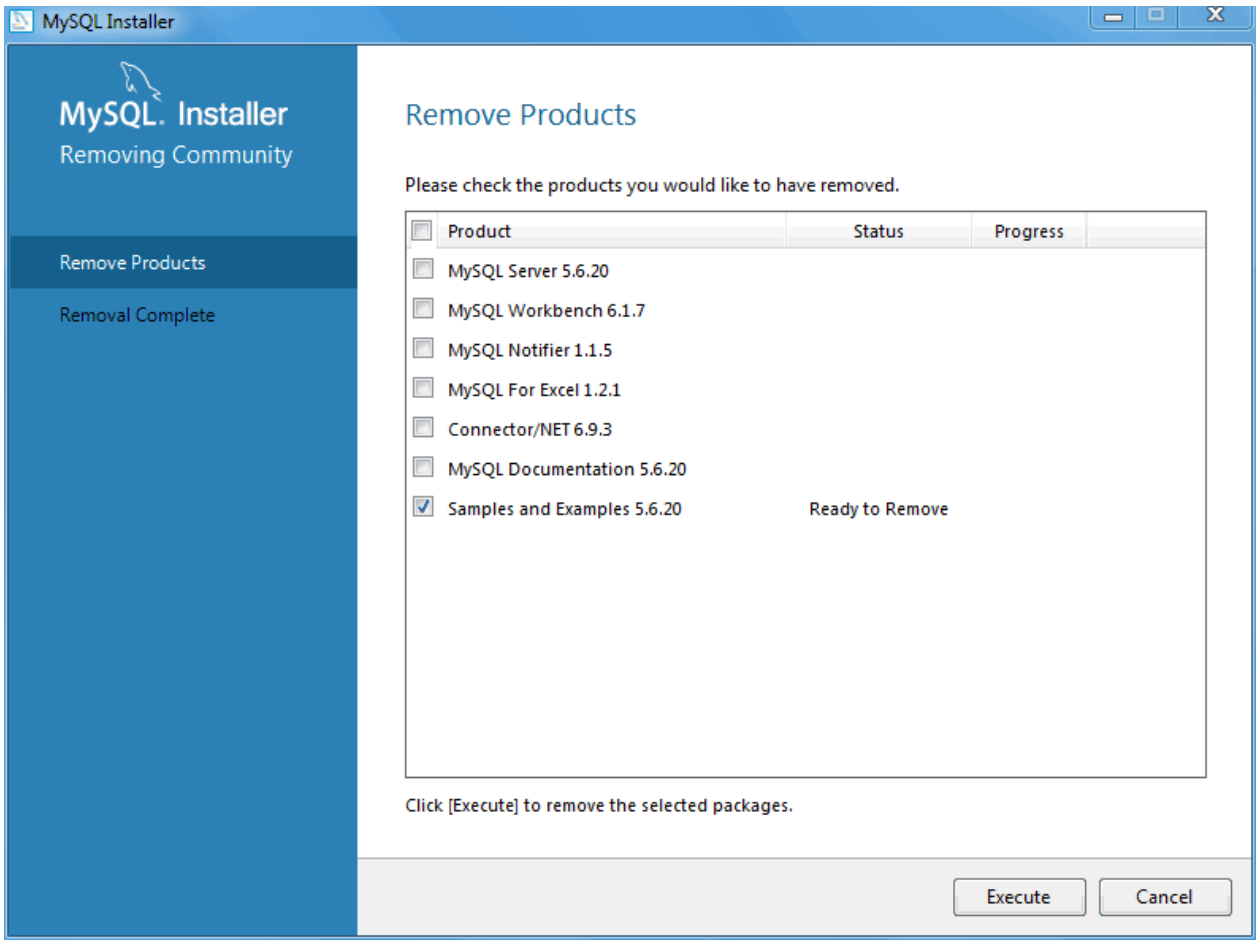
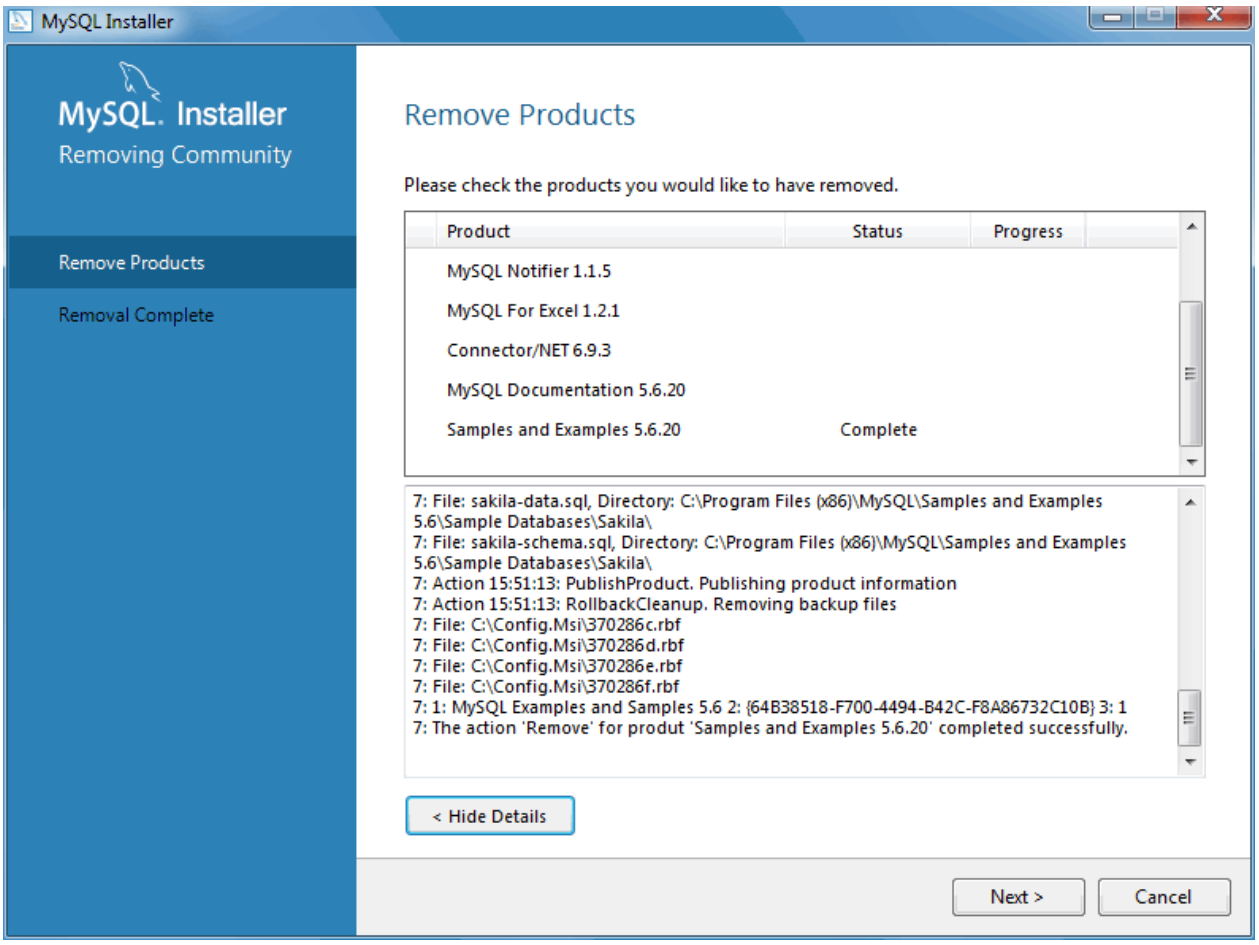


Figure 5.19 MySQL Installer - Removing Products: Executed



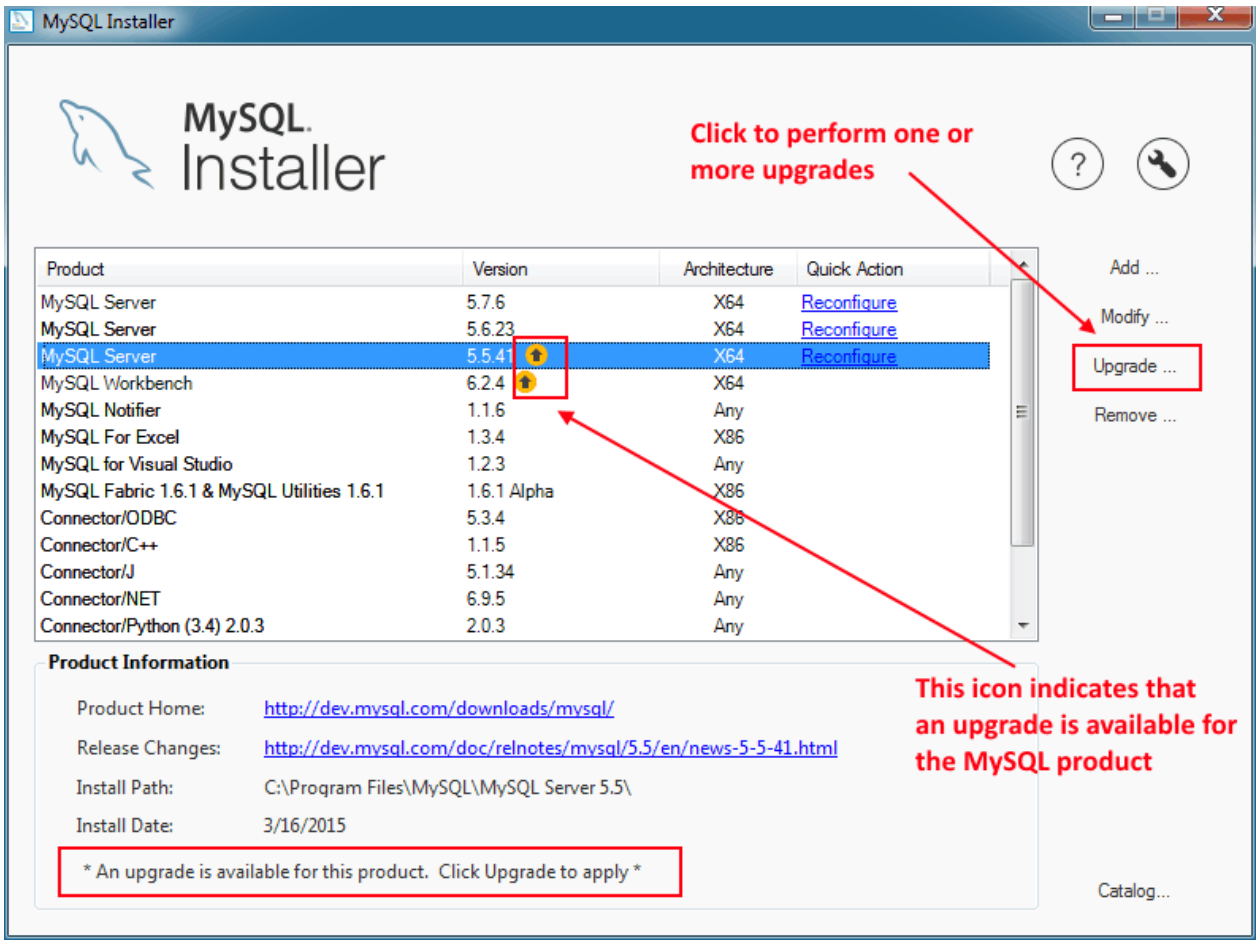
5.3.1.3 Alter MySQL Products

Use MySQL Installer to modify, configure, or upgrade your MySQL product installations.

Upgrade

Upgradable MySQL products are listed on the main dashboard with an arrow icon ( ) next to their version number.

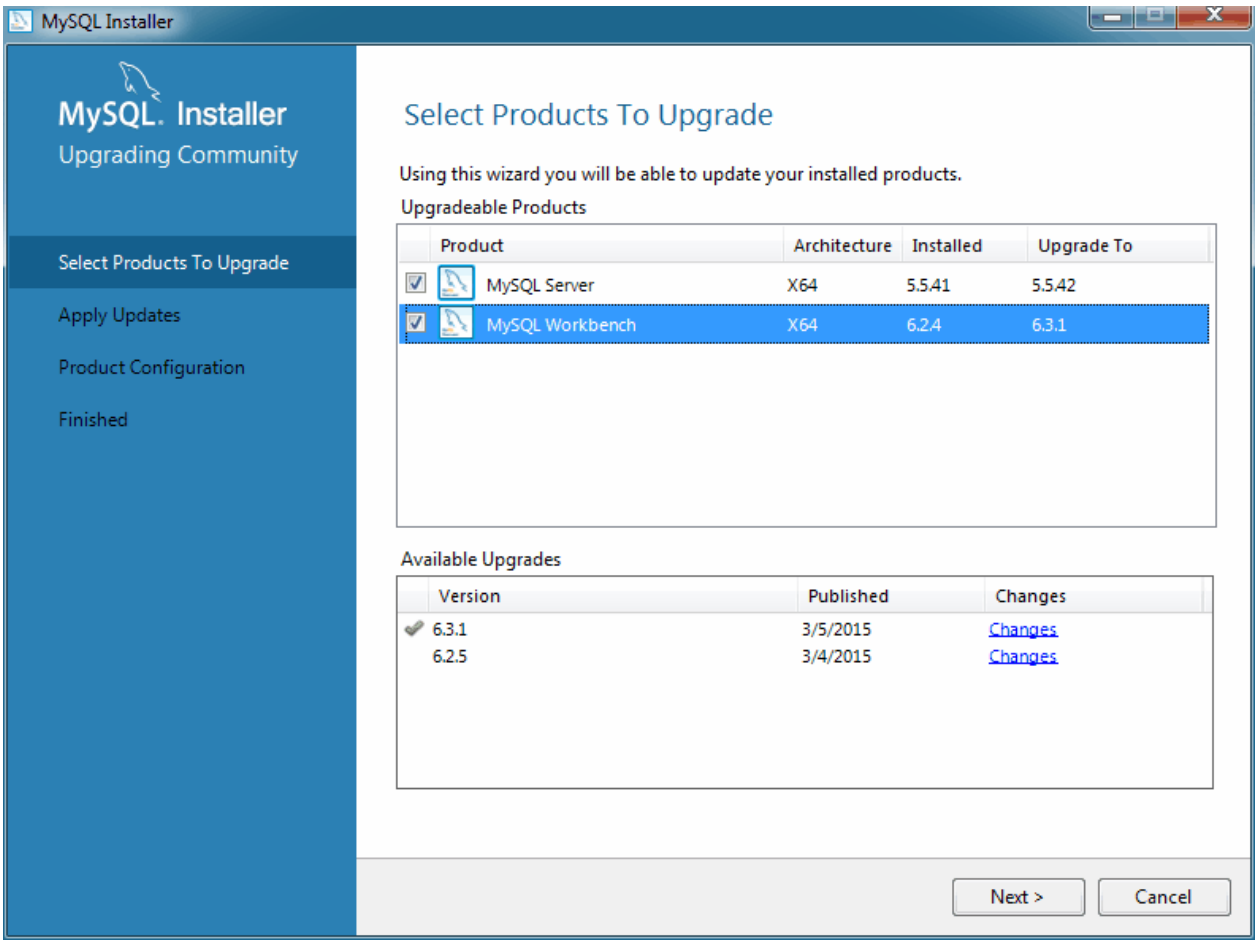
Figure 5.20 MySQL Installer - Upgrade a MySQL Product



> **Note**
>
> The "upgrade" functionality requires a current product catalog. This catalog is updated either manually or automatically (daily) by enabling the **Automatic Catalog Update** feature. For additional information, see Section 5.3.1.1, “MySQL Product Catalog”.

Click **Upgrade** to upgrade the available products. Our example indicates that MySQL Workbench 6.2.4 can be upgraded version 6.3.1 or 6.2.5, and MySQL server from 5.5.41 to 5.5.42.

Figure 5.21 MySQL Installer - Select Products To Upgrade

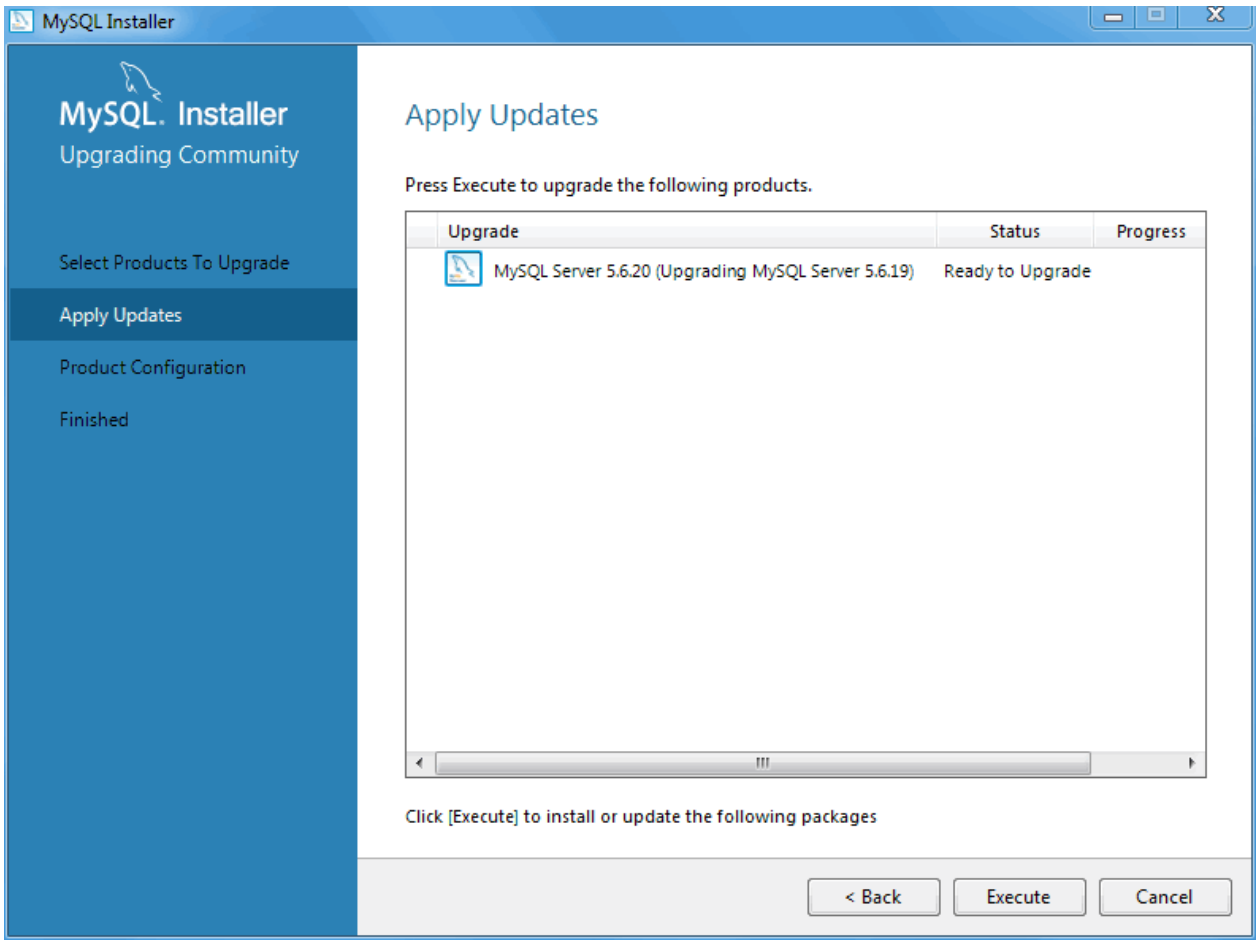


If multiple upgrade versions are available (such as our MySQL Workbench example above), select the desired version for the upgrade in the **Available Upgrades** area.

> **Note**
>
> Optionally, click the **Changes** link to view the version's release notes.

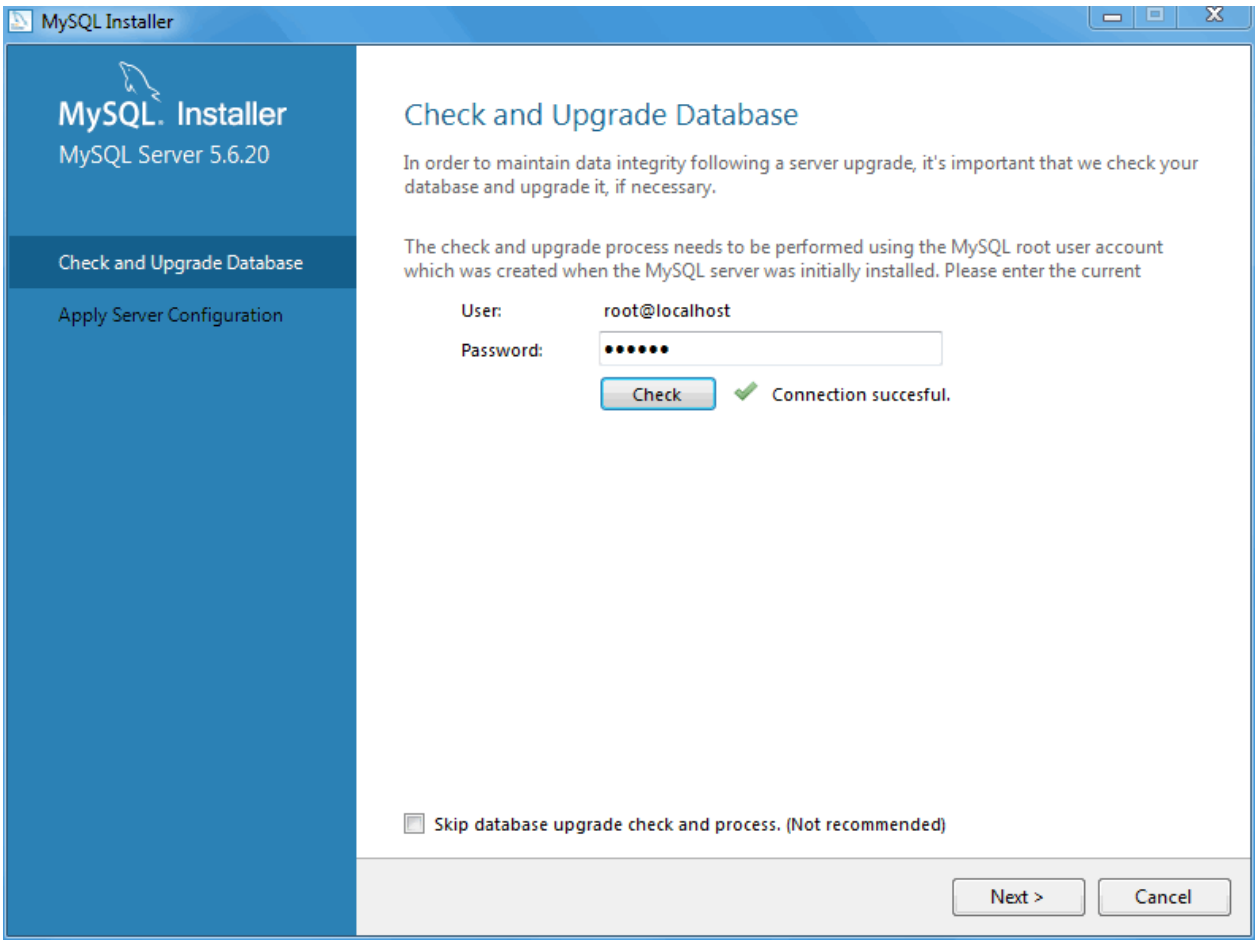
After selecting (checking) the products and versions to upgrade, click **Next** to begin the upgrade process.

Figure 5.22 MySQL Installer - Apply Updates



A MySQL server upgrade will also check and upgrade the server's database. Although optional, this step is recommended.

Figure 5.23 MySQL Installer - Check and Upgrade Database



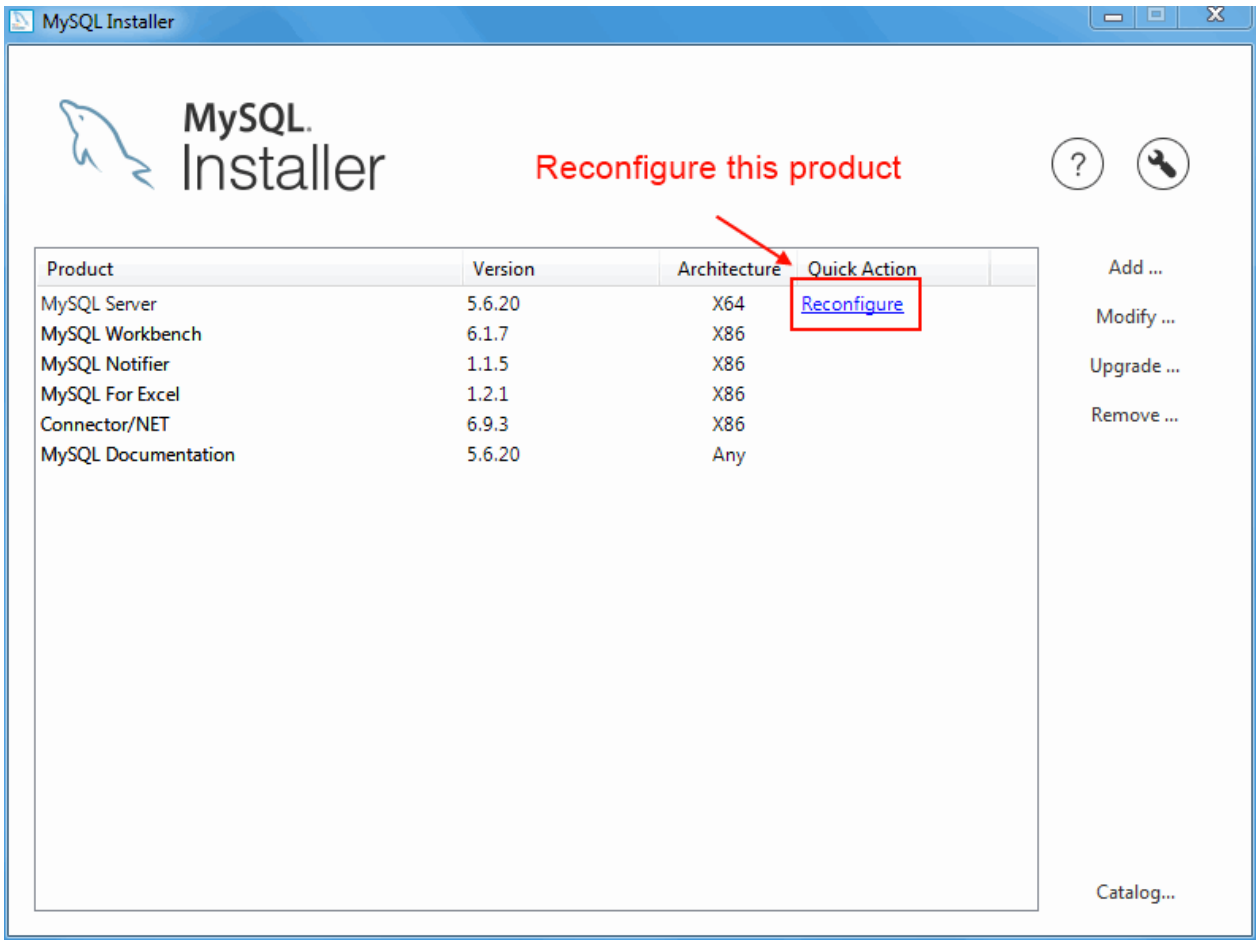
Upon completion, your upgraded products will be upgraded and available to use. A MySQL server upgrade also restarts the MySQL server.

Reconfigure

Some MySQL products, such as the MySQL server, include a **Reconfigure** option. It opens the same configuration options that were set when the MySQL product was installed, and is pre-populated with the current values.

To execute, click the `Reconfigure` link under the **Quick Action** column on the main dashboard for the MySQL product that you want to reconfigure.

Figure 5.24 MySQL Installer - Reconfigure a MySQL Product



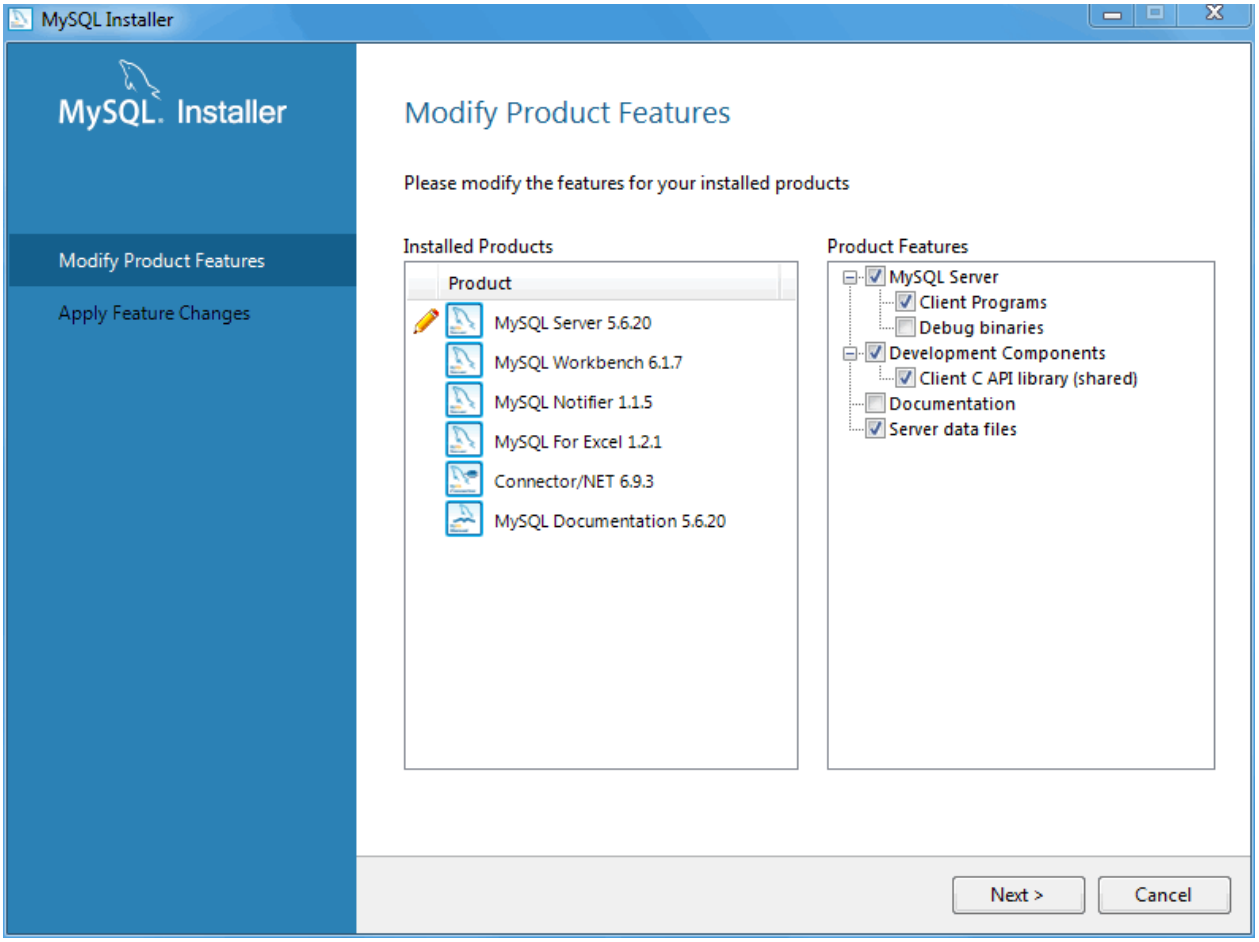
In the case of the MySQL server, this opens a configuration wizard that relates to the selected product. For example, for MySQL Server this includes setting the type, ports, log paths, and so on.

Modify

Many MySQL products contain feature components that can be added or removed. For example, `Debug binaries` and `Client Programs` are subcomponents of the MySQL server.

The modify the features of a product, click **Modify** on the main dashboard.

Figure 5.25 MySQL Installer - Modify Product Features



Click **Execute** to execute the modification request.

5.3.2 MySQL Installer Console

`MySQLInstallerConsole` provides functionality similar to the GUI version of MySQL Installer, but from the command line. It is installed when MySQL Installer is initially executed, and then available within the `MySQL Installer` directory. Typically that is in `C:\Program Files (x86)\MySQL\MySQL Installer\`, and the console must be executed with administrative privileges.

To use, invoke the Command Prompt with administrative privileges by choosing **Start**, **Accessories**, then right-click on **Command Prompt** and choose `Run as administrator`. And from the command-line, optionally change the directory to where `MySQLInstallerConsole.exe` is located:

```
C:\> cd "C:\Program Files (x86)\MySQL\MySQL Installer for Windows"
C:\> MySQLInstallerConsole.exe help

C:\Program Files (x86)\MySQL\MySQL Installer for Windows>MySQLInstallerConsole.exe help

The following commands are available:

Configure - Configures one or more of your installed programs.
Help      - Provides list of available commands.
Install   - Install and configure one or more available MySQL programs.
List      - Provides an interactive way to list all products available.
Modify    - Modifies the features of installed products.
```

```
Remove    - Removes one or more products from your system.
Status    - Shows the status of all installed products.
Update    - Update the current product catalog.
Upgrade   - Upgrades one or more of your installed programs.
```

`MySQLInstallerConsole.exe` supports the following options, which are specified on the command line:

> **Note**
>
> Configuration block values that contain a colon (":") must be wrapped in double quotes. For example, installdir="C:\MySQL\MySQL Server 5.6".

- `configure` *`[product1]:[setting]=[value]; [product2]:[setting]=[value]; [...]`*

  Configure one or more MySQL products on your system. Multiple setting=value pairs can be configured for each product.

  Switches include:

  - `-showsettings` : Displays the available options for the selected product, by passing in the product name after `-showsettings`.
  - `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole configure -showsettings server
C:\> MySQLInstallerConsole configure server:port=3307
```

- `help` *`[command]`*

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

```
C:\> MySQLInstallerConsole help
C:\> MySQLInstallerConsole help install
```

- `install` *`[product]:[features]:[config block]:[config block]:[config block]; [...]`*

  Install one or more MySQL products on your system.

  Switches and syntax options include:

  - `-type=[SetupType]` : Installs a predefined set of software. The "SetupType" can be one of the following:

    > **Note**
    >
    > Non-custom setup types can only be chosen if no other MySQL products are installed.

    - **Developer**: Installs a complete development environment.
    - **Server**: Installs a single MySQL server
    - **Client**: Installs client programs and libraries
    - **Full**: Installs everything

    - **Custom**: Installs user selected products. This is the default option.

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

  - `-silent` : Disable confirmation prompts.

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

    Configuration block values that contain a colon (":") must be wrapped in double quotes. For example, `installdir="C:\MySQL\MySQL Server 5.6"`.

    Only one "config" type block can be defined per product. A "user" block should be defined for each user that should be created during the product's installation.

    > **Note**
    >
    > Adding users is not supported when a product is being reconfigured.

  - `[feature]`: The feature block is a semicolon separated list of features, or '*' to select all features.

```
C:\> MySQLInstallerConsole install server;5.6.25:*:port=3307;serverid=2:type=user;username=foo;password=
C:\> MySQLInstallerConsole install server;5.6.25;x64 -silent
```

An example that passes in additional configuration blocks, broken up by ^ to fit this screen:

```
C:\> MySQLInstallerConsole install server;5.6.25;x64:*:type=config;openfirewall=true; ^
         generallog=true;binlog=true;serverid=3306;enable_tcpip=true;port=3306;rootpasswd=pass; ^
         installdir="C:\MySQL\MySQL Server 5.6":type=user;datadir="C:\MySQL\data";username=foo;password
```

- `list`

  Lists an interactive console where all of the available MySQL products can be searched. Execute `MySQLInstallerConsole list` to launch the console, and enter in a substring to search.

```
C:\> MySQLInstallerConsole list
```

- `modify` *`[product1:-removelist|+addlist] [product2:-removelist|+addlist] [...]`*

  Modifies or displays features of a previously installed MySQL product.

  - `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole modify server
C:\> MySQLInstallerConsole modify server:+documentation
C:\> MySQLInstallerConsole modify server:-debug
```

- `remove` *`[product1] [product2] [...]`*

  Removes one ore more products from your system.

  - `*` : Pass in `*` to remove all of the MySQL products.

- `-continue` : Continue the operation even if an error occurs.
- `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole remove *
C:\> MySQLInstallerConsole remove server
```

- `status`

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

```
C:\> MySQLInstallerConsole status
```

- `update`

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

```
C:\> MySQLInstallerConsole update
```

> **Note**
>
> The **Automatic Catalog Update** GUI option executes this command from the Windows Task Scheduler.

- `upgrade` *`[product1:version] [product2:version] [...]`*

  Upgrades one or more products on your system. Syntax options include:

  - `*` : Pass in `*` to upgrade all products to the latest version, or pass in specific products.
  - `!` : Pass in `!` as a version number to upgrade the MySQL product to its latest version.
  - `-silent` : Disable confirmation prompts.

```
C:\> MySQLInstallerConsole upgrade *
C:\> MySQLInstallerConsole upgrade workbench:6.3.5
C:\> MySQLInstallerConsole upgrade workbench:!
C:\> MySQLInstallerConsole upgrade workbench:6.3.5 excel:1.3.2
```

5.4 MySQL Notifier

MySQL Notifier is a tool that enables you to monitor and adjust the status of your local and remote MySQL server instances through an indicator that resides in the system tray. MySQL Notifier also gives quick access to MySQL Workbench through its context menu.

The MySQL Notifier is installed by MySQL Installer, and (by default) will start-up when Microsoft Windows is started.

To install, download and execute the MySQL Installer, be sure the MySQL Notifier product is selected, then proceed with the installation. See the MySQL Installer manual for additional details.

For notes detailing the changes in each release of MySQL Notifier, see the MySQL Notifier Release Notes.

Visit the MySQL Notifier forum for additional MySQL Notifier help and support.

Features include:

- Start, Stop, and Restart instances of the MySQL Server.
- Automatically detects (and adds) new MySQL Server services. These are listed under **Manage Monitored Items**, and may also be configured.
- The Tray icon changes, depending on the status. It's green if all monitored MySQL Server instances are running, or red if at least one service is stopped. The **Update MySQL Notifier tray icon based on service status** option, which dictates this behavior, is enabled by default for each service.
- Links to other applications like MySQL Workbench, MySQL Installer, and the MySQL Utilities. For example, choosing **Manage Instance** will load the MySQL Workbench Server Administration window for that particular instance.
- If MySQL Workbench is also installed, then the **Manage Instance** and **SQL Editor** options are available for local (but not remote) MySQL instances.
- Monitors both local and remote MySQL instances.

5.4.1 MySQL Notifier Usage

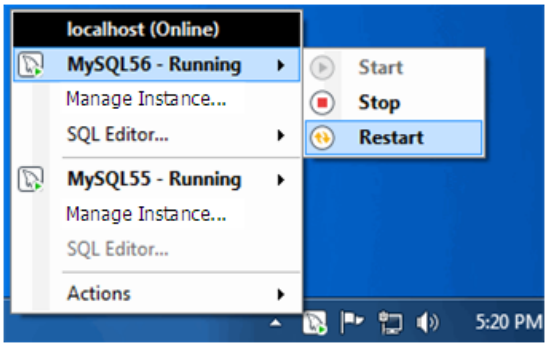
MySQL Notifier resides in the system tray and provides visual status information for your MySQL server instances. A green icon is displayed at the top left corner of the tray icon if the current MySQL server is running, or a red icon if the service is stopped.

MySQL Notifier automatically adds discovered MySQL services on the local machine, and each service is saved and configurable. By default, the **Automatically add new services whose name contains** option is enabled and set to `mysql`. Related **Notifications Options** include being notified when new services are either discovered or experience status changes, and are also enabled by default. And uninstalling a service will also remove the service from MySQL Notifier.

Clicking the system tray icon will reveal several options, as the follow figures show:

The Service Instance menu is the main MySQL Notifier window, and enables you to Stop, Start, and Restart the MySQL server.

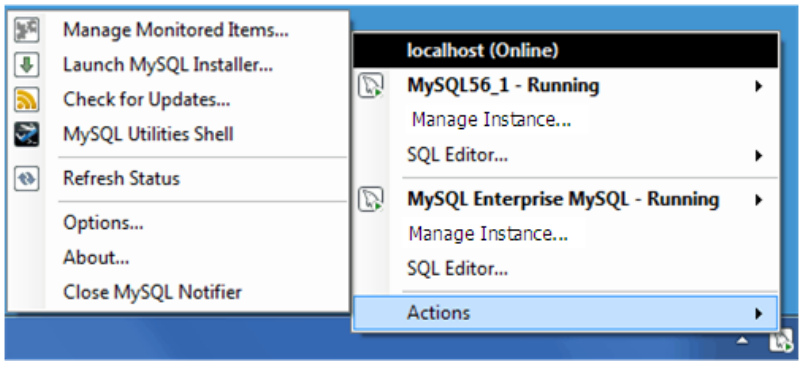
Figure 5.26 MySQL Notifier Service Instance menu



The **Actions** menu includes several links to external applications (if they are installed), and a **Refresh Status** option to manually refresh the status of all monitored services (in both local and remote computers) and MySQL instances.

> **Note**
>
> The main menu will not show the **Actions** menu when there are no services being monitored by MySQL Notifier.

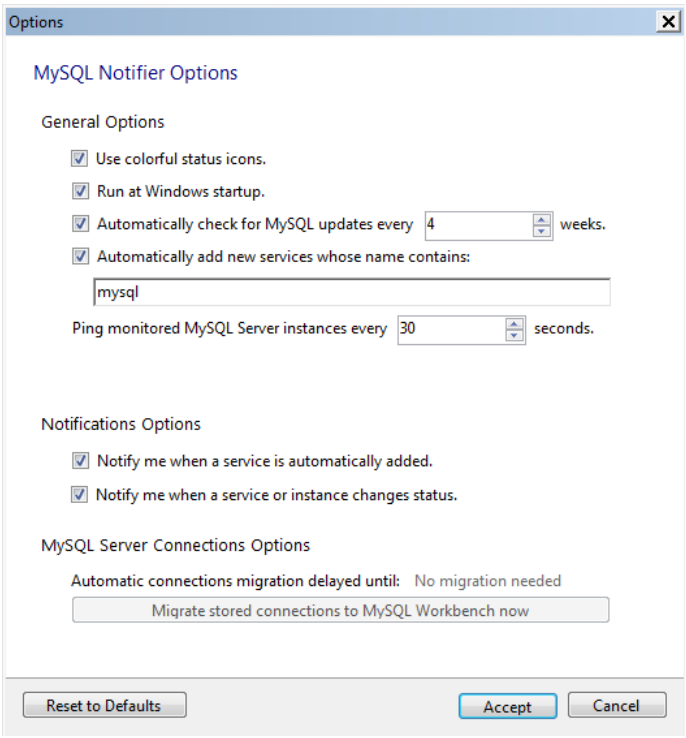
Figure 5.27 MySQL Notifier Actions menu



The **Actions**, **Options** menu configures MySQL Notifier and includes options to:

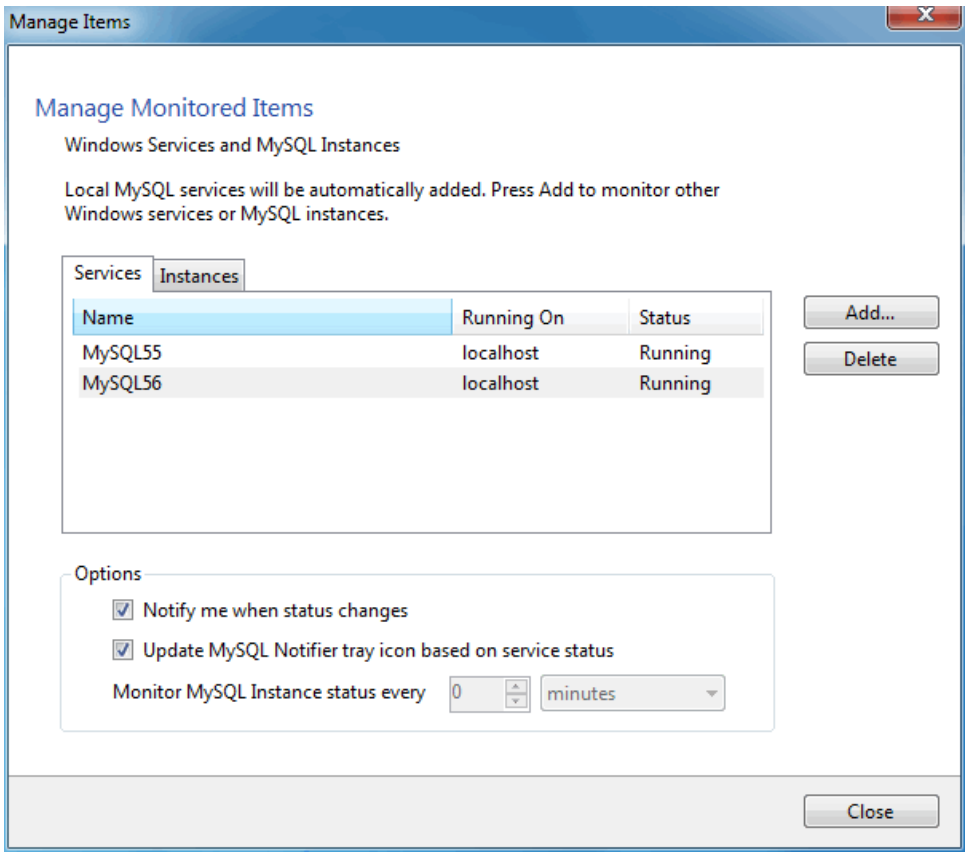
- **Use colorful status icons**: Enables a colorful style of icons for the tray of MySQL Notifier.
- **Run at Windows Startup**: Allows the application to be loaded when Microsoft Windows starts.
- **Automatically Check For Updates Every** # **Weeks**: Checks for a new version of MySQL Notifier, and runs this check every # weeks.
- **Automatically add new services whose name contains:** The text used to filter services and add them automatically to the monitored list of the local computer running MySQL Notifier, and on remote computers already monitoring Windows services.
- **Ping monitored MySQL Server instances every** # **seconds**: The interval (in seconds) to ping monitored MySQL Server instances for status changes. Longer intervals might be necessary if the list of monitored remote instances is large.
- **Notify me when a service is automatically added**: Will display a balloon notification from the taskbar when a newly discovered service is added to the monitored services list.
- **Notify me when a service changes status**: Will display a balloon notification from the taskbar when a monitored service changes its status.
- **Automatic connections migration delayed until:** When there are connections to migrate, postpone the migration by one hour, one day, one week, one month, or indefinitely.

Figure 5.28 MySQL Notifier Options menu



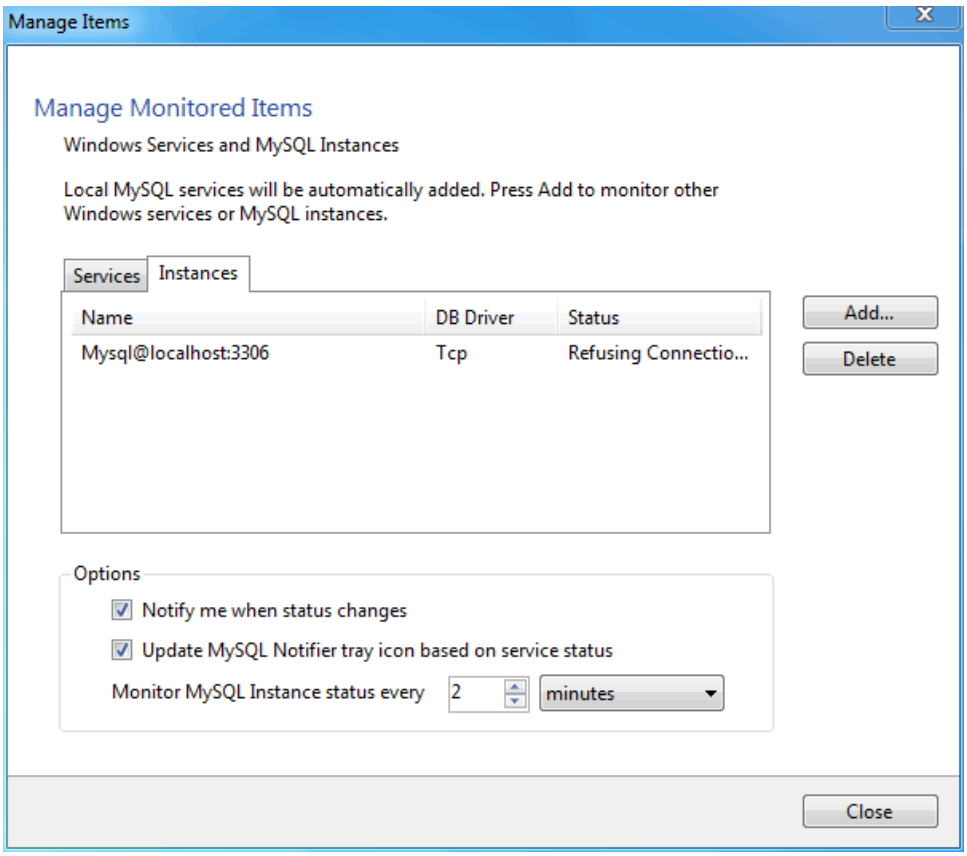
The **Actions**, **Manage Monitored Items** menu enables you to configure the monitored services and MySQL instances. First, with the **Services** tab open:

Figure 5.29 MySQL Notifier Manage Services menu



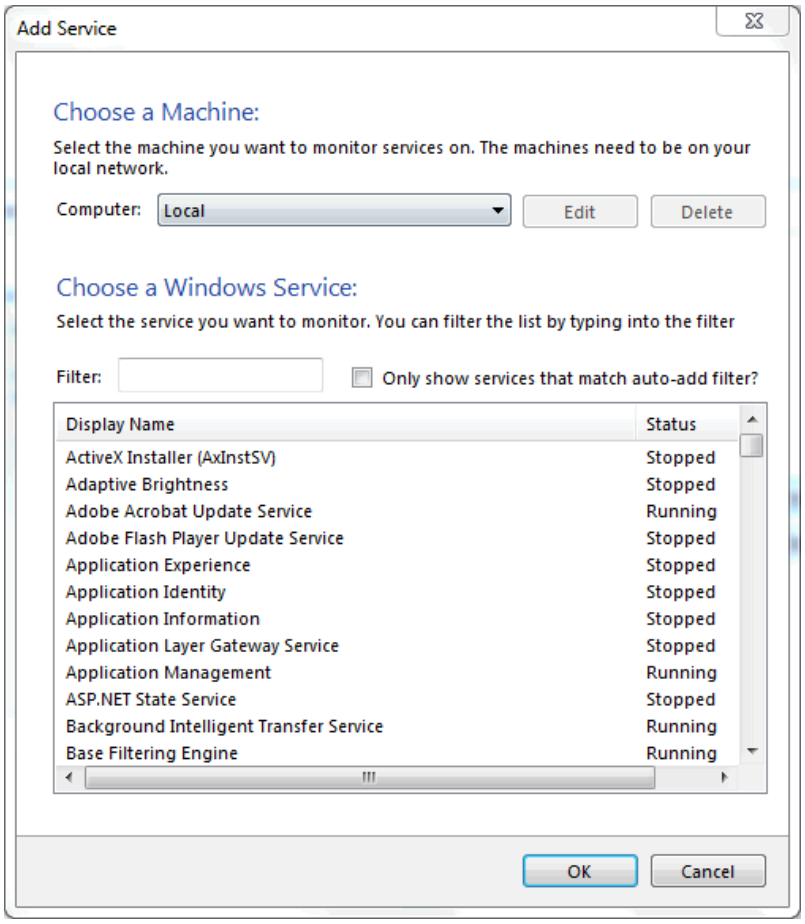
The **Instances** tab is similar:

Figure 5.30 MySQL Notifier Manage Instances menu



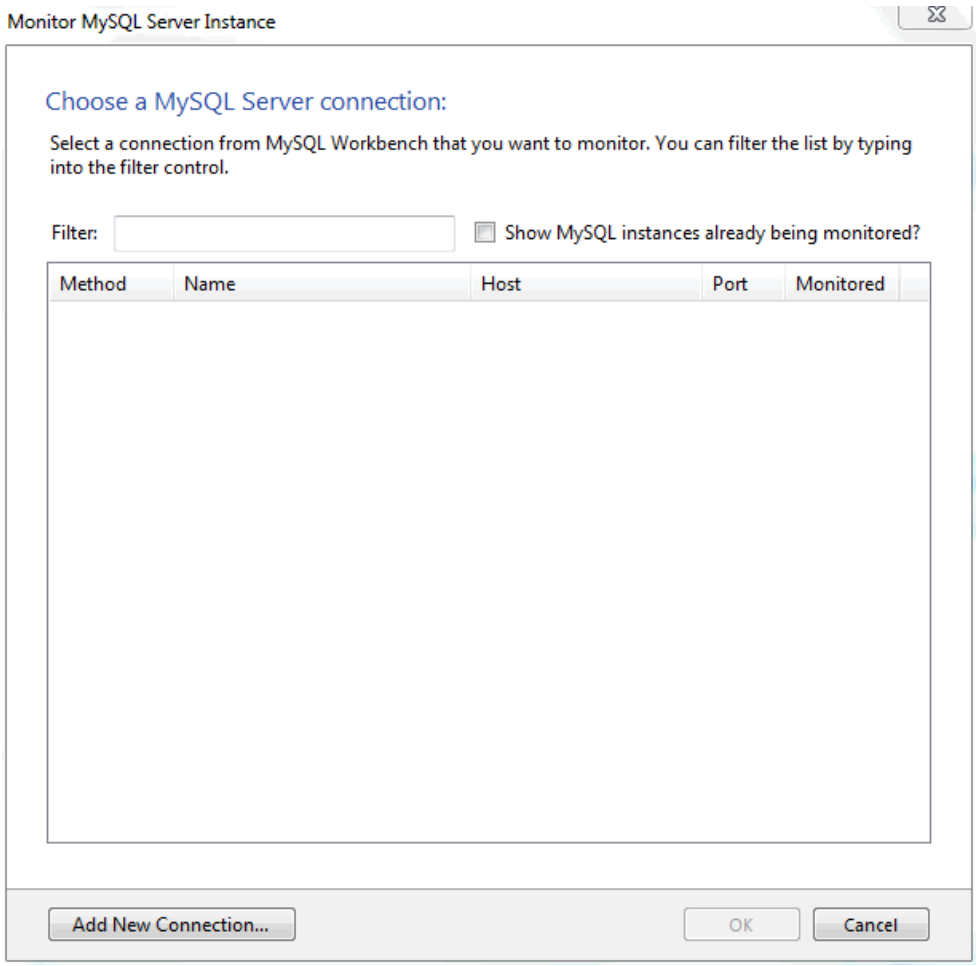
Adding a service or instance (after clicking **Add** in the **Manage Monitored Items** window) enables you to select a running Microsoft Windows service or instance connection, and configure MySQL Notifier to monitor it. Add a new service or instance by clicking service name from the list, then **OK** to accept. Multiple services and instances may be selected.

Figure 5.31 MySQL Notifier Adding new services



Add instances:

Figure 5.32 MySQL Notifier Adding new instances



Troubleshooting

For issues that are not documented here, visit the MySQL Notifier Support Forum for MySQL Notifier help and support.

- *Problem*: attempting to start/stop/restart a MySQL service might generate an error similar to "The Service **MySQL*VERSION*** failed the most recent status change request with the message "The service **mysql*VERSION*** was not found in the Windows Services".

  Explanation: this is a case-sensitivity issue, in that the service name is **MySQL*VERSION*** compared to having **mysql*VERSION*** in the configuration file.

  Solution: either update your MySQL Notifier configuration file with the correct information, or stop MySQL Notifier and delete this configuration file. The MySQL Notifier configuration file is located at `%APPDATA%\Oracle\MySQL Notifier\settings.config` where `%APPDATA%` is a variable and depends on your system. A typical location is "C:\Users*YourUsername*\AppData\Running\Oracle\MySQL Notifier\settings.config" where *YourUsername* is your system's user name. In this file, and within the ServerList section, change the ServerName values from lowercase to the actual service names. For example, change **mysql*VERSION*** to **MySQL*VERSION***, save, and then restart MySQL Notifier. Alternatively, stop MySQL Notifier, delete this file, then restart MySQL Notifier.

- *Problem*: when connecting to a remote computer for the purpose of monitoring a remote Windows service, the **Add Service** dialog does not always show all the services shown in the Windows Services console.

  Explanation: this behavior is governed by the operating system and the outcome is expected when working with nondomain user accounts. For a complete description of the behavior, see the User Account Control and WMI article from Microsoft.

  Solution: when the remote computer is in a compatible domain, it is recommended that domain user accounts are used to connect through WMI to a remote computer. For detailed setup instructions using WMI, see Section 5.4.2, “Setting Up Remote Monitoring in MySQL Notifier”.

  Alternatively, when domain user accounts are not available, Microsoft provides a less secure workaround that should only be implemented with caution. For more information, see the Description of User Account Control and remote restrictions in Windows Vista KB article from Microsoft.

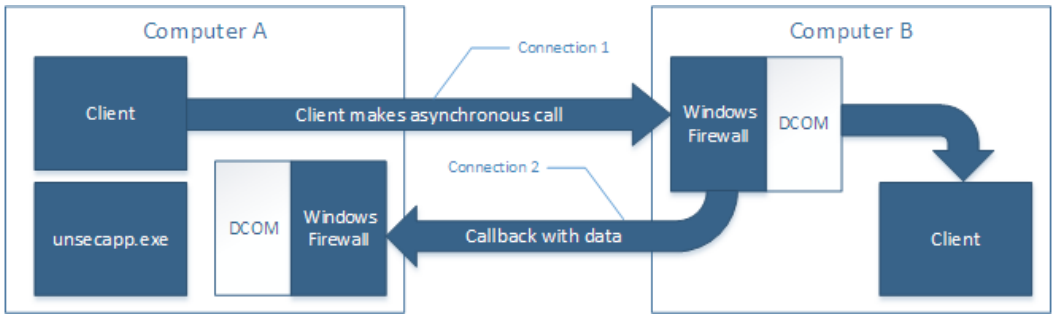
5.4.2 Setting Up Remote Monitoring in MySQL Notifier

MySQL Notifier uses Windows Management Instrumentation (WMI) to manage and monitor services on remote computers. This section explains how it works and how to set up your system to monitor remote MySQL instances.

In order to configure WMI, it is important to understand that the underlying Distributed Component Object Model (DCOM) architecture is doing the WMI work. Specifically, MySQL Notifier is using asynchronous notification queries on remote Microsoft Windows hosts as .NET events. These events send an asynchronous callback to the computer running MySQL Notifier so it knows when a service status has changed on the remote computer. Asynchronous notifications offer the best performance compared to semisynchronous notifications or synchronous notifications that use timers.

Asynchronous notification requires the remote computer to send a callback to the client computer (thus opening a reverse connection), so the Windows Firewall and DCOM settings must be properly configured for the communication to function properly.

Figure 5.33 MySQL Notifier Distributed Component Object Model (DCOM)



Most of the common errors thrown by asynchronous WMI notifications are related to Windows Firewall blocking the communication, or to DCOM / WMI settings not being set up properly. For a list of common errors with solutions, see Common Errors.

The following steps are required to make WMI function. These steps are divided between two machines. A single host computer that runs MySQL Notifier (Computer A), and multiple remote machines that are being monitored (Computer B).

Computer running MySQL Notifier (Computer A)

1. Enable remote administration by either editing the **Group Policy Editor**, or using `NETSH`:

Using the **Group Policy Editor**:

a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.

b. Under the **Local Computer Policy** heading, expand **Computer Configuration**.

c. Expand **Administrative Templates**, then **Network**, **Network Connections**, and then **Windows Firewall**.

d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.

e. Double-click **Windows Firewall: Allow inbound remote administration exception** to open a configuration window.

f. Check the **Enabled** option button and then click **OK**.

Using the `NETSH` command:

> **Note**
>
> The "netsh firewall" command is deprecated as of Microsoft Server 2008 and Vista, and replaced with "netsh advfirewall firewall".

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).

b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

2. Open the DCOM port TCP 135:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).

b. Execute the following command:

```
NETSH advfirewall firewall add rule name=DCOM_TCP135 protocol=TCP localport=135 dir=in action=allow
```

3. Add the client application that contains the sink for the callback (`MySqlNotifier.exe`) to the Windows Firewall Exceptions List (use either the Windows Firewall configuration or `NETSH`):

Using the Windows Firewall configuration:

a. In the Control Panel, double-click **Windows Firewall**.

b. In the Windows Firewall window's left panel, click **Allow a program or feature through Windows Firewall**.

c. In the Allowed Programs window, click **Change Settings** and do one of the following:

- If `MySqlNotifier.exe` is in the Allowed programs and features list, make sure it is checked for the type of networks the computer connects to (Private, Public or both).
- If `MySqlNotifier.exe` is not in the list, click **Allow another program...**.

i. In the **Add a Program** window, select the `MySqlNotifier.exe` if it exists in the Programs list, otherwise click Browse... and go to the directory where `MySqlNotifier.exe` was installed to select it, then click **Add**.

ii. Make sure `MySqlNotifier.exe` is checked for the type of networks the computer connects to (Private, Public or both).

Using the `NETSH` command:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click **Run as Administrator**).

b. Execute the following command, where you change "*`[YOUR_INSTALL_DIRECTORY]`*":

```
NETSH advfirewall firewall add rule name=MySqlNotifier program=[YOUR_INSTALL_DIRECTORY]\MySqlNotifier.ex
```

4. If Computer B is either a member of `WORKGROUP` or is in a different domain that is untrusted by Computer A, then the callback connection (Connection 2) is created as an Anonymous connection. To grant Anonymous connections DCOM Remote Access permissions:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Access Permissions, click **Edit Limits**.

e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

Monitored Remote Computer (Computer B)

If the user account that is logged on to the computer running the MySQL Notifier (Computer A) is a local administrator on the remote computer (Computer B), such that the same account is an administrator on Computer B, you can skip to the "Allow for remote administration" step.

Setting DCOM security to allow a non-administrator user to access a computer remotely:

1. Grant "DCOM remote launch" and activation permissions for a user or group:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Launch and Activation Permission, click **Edit Limits**.

e. In the **Launch and Activation Permission** dialog box, follow these steps if your name or your group does not appear in the Groups or user names list:

i. In the **Launch and Activation Permission** dialog box, click **Add**.

ii. In the Select Users or Groups dialog box, add your name and the group in the **Enter the object names to select** box, and then click **OK**.

f. In the **Launch and Activation Permission** dialog box, select your user and group in the Group or user names box. In the Allow column under Permissions for User, select **Remote Launch**, select **Remote Activation**, and then click **OK**.

Grant DCOM remote access permissions:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Access Permissions, click **Edit Limits**.

e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

2. Allowing non-administrator users access to a specific WMI namespace:

a. In the Control Panel, double-click **Administrative Tools**.

b. In the Administrative Tools window, double-click **Computer Management**.

c. In the Computer Management window, expand the **Services and Applications** tree.

d. Right-click the WMI Control icon and select **Properties**.

e. In the WMI Control Properties window, click the **Security** tab.

f. In the Security tab, select the namespace and click **Security**. Root/CIMV2 is a commonly used namespace.

g. Locate the appropriate account and check **Remote Enable** in the Permissions list.

3. Allow for remote administration by either editing the **Group Policy Editor** or using `NETSH`:

Using the **Group Policy Editor**:

a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.

b. Under the Local Computer Policy heading, double-click **Computer Configuration**.

c. Double-click **Administrative Templates**, then **Network**, **Network Connections**, and then **Windows Firewall**.

d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.

e. Click **Windows Firewall: Allow inbound remote administration exception**.

f. On the Action menu either select **Edit**, or double-click the selection from the previous step.

g. Check the **Enabled** radio button, and then click **OK**.

Using the `NETSH` command:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click Run as Administrator).

b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

4. Confirm that the user account you are logging in with uses the `Name` value and not the `Full Name` value:

   a. In the **Control Panel**, double-click **Administrative Tools**.

   b. In the **Administrative Tools** window, double-click **Computer Management**.

   c. In the **Computer Management** window, expand the **System Tools** then **Local Users and Groups**.

   d. Click the **Users** node, and on the right side panel locate your user and make sure it uses the **Name** value to connect, and not the **Full Name** value.

Common Errors

- `0x80070005`
  - DCOM Security was not configured properly (see Computer B, the `Setting DCOM security...` step).
  - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the `Grant Anonymous connections DCOM Remote Access permissions` step).
- `0x8007000E`
  - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the `Grant Anonymous connections DCOM Remote Access permissions` step).
- `0x80041003`
  - Access to the remote WMI namespace was not configured properly (see Computer B, the `Allowing non-administrator users access to a specific WMI namespace` step).
- `0x800706BA`
  - The DCOM port is not open on the client computers (Computer A) firewall. See the `Open the DCOM port TCP 135` step for Computer A.
  - The remote computer (Computer B) is inaccessible because its network location is set to Public. Make sure you can access it through the Windows Explorer.

5.5 Installing MySQL on Microsoft Windows Using a noinstall Zip Archive

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

1. Extract the archive to the desired install directory
2. Create an option file
3. Choose a MySQL server type
4. Start the MySQL server
5. Secure the default user accounts

This process is described in the sections that follow.

5.5.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to Section 5.8, “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`. The MySQL Installation Wizard installs MySQL under `C:\Program Files\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 5.5.2, “Creating an Option File”.

   > **Note**
   >
   > The MySQL Installer installs MySQL under `C:\Program Files\MySQL`.

4. Extract the install archive to the chosen installation location using your preferred Zip archive 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.

5.5.2 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 5.6` and `C:\Program Files\MySQL\MySQL Server 5.6\data`).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, `C:\`, and the MySQL installation directory (for the full list of locations, see Using Option Files). The Windows directory typically is named something like `C:\WINDOWS`. You can determine its exact location from the value of the `WINDIR` environment variable using the following command:

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

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

> **Note**
>
> When using the MySQL Installer to install MySQL Server, it will create the `my.ini` at the default location. And as of MySQL Server 5.5.27, the user running MySQL Installer is granted full permissions to this new `my.ini`.
>
> 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 data directory is located within the `AppData` directory for the user running MySQL.

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 5.6\data`) to `E:\mydata`.
2. Use a `--datadir` option to specify the new data directory location each time you start the server.

5.5.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 5.6.

Binary	Description
`mysqld`	Optimized binary with named-pipe support

Binary	Description
`mysqld-debug`	Like `mysqld`, but compiled with full debugging and automatic memory allocation checking

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

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

All Windows MySQL 5.6 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 `--enable-named-pipe` option. It is necessary to use this option 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.

5.5.4 Starting the Server for the First Time

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

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

> **Note**
>
> The MySQL server will automatically start after using the MySQL Installer, and the MySQL Notifier GUI can be used to start/stop/restart at any time.

The examples in these sections assume that MySQL is installed under the default location of `C:\Program Files\MySQL\MySQL Server 5.6`. 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` option. Clients can connect through shared memory by using the `--protocol=MEMORY` option.

For information about which server binary to run, see Section 5.5.3, “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.

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --console
```

For a server that includes `InnoDB` support, you should see the messages similar to those following as it starts (the path names and sizes may differ):

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
```

```
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

When the server finishes its startup sequence, you should see something like this, which indicates that the server is ready to service client connections:

```
mysqld: ready for connections
Version: '5.6.35'  socket: ''  port: 3306
```

The server continues to write to the console any further diagnostic output it produces. You can 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 5.6\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

> **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 Section 9.4, “Securing the Initial MySQL Accounts”.

5.5.5 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.

> **Note**
>
> The MySQL Notifier GUI can also be used to start/stop/restart the MySQL server.

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 5.6\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 5.6\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 login 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 5.6\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 that will 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.

5.5.6 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 5.6\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.

5.5.7 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.

> **Note**
>
> The MySQL Notifier GUI can also be used to monitor the status of the MySQL service.

The `Services` utility (the Windows `Service Control Manager`) can be found in the Windows Control Panel (under **Administrative Tools** on Windows 2000, XP, Vista, and Server 2003). 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 5.6\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 login users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\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 5.6\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. This account is available only for Windows XP or newer. 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 a 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 5.6\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` will attempt to start the MySQL server.

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

Starting the service

Once a MySQL server 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 a `NET START MySQL` command. The `NET` command is 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 5.6\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 by using the `Services` utility, the `NET STOP MySQL` 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 5.6\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 `NET STOP MySQL`. Then use the `--remove` option to remove it:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld" --remove
```

If `mysqld` is not running as a service, you can start it from the command line. For instructions, see Section 5.5.5, “Starting MySQL from the Windows Command Line”.

If you encounter difficulties during installation, see Section 5.6, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

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

5.5.8 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 5.6\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 5.6\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` option 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.

For more information about `mysqlshow`, see `mysqlshow` — Display Database, Table, and Column Information.

5.6 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 5.6\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 `NET START MySQL` 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 5.5.7, “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 5.6` and `C:\Program Files\MySQL\MySQL Server 5.6\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 5.6`, 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 5.6
# 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 5.5.2, “Creating an Option File”.

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

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

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

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

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

```
C:\> sc delete mysql
[SC] DeleteService SUCCESS
```

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

5.7 Windows Postinstallation Procedures

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

- MySQL Installer: Used to install and upgrade MySQL products.
- MySQL Workbench: Manages the MySQL server and edits SQL statements.
- MySQL Notifier: Starts, stops, or restarts the MySQL server, and monitors its status.
- MySQL for Excel: Edits MySQL data with Microsoft Excel.

On Windows, you need not create the data directory and the grant tables. MySQL Windows distributions include the grant tables with a set of preinitialized accounts in the `mysql` database under the data directory.

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned passwords to the accounts. (See Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”.) Otherwise, use the password-assignment procedure given in Section 9.4, “Securing the Initial MySQL Accounts”.

Before assigning passwords, 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 5.5.4, “Starting the Server for the First Time”). You can also set up a MySQL service that runs automatically when Windows starts (see Section 5.5.7, “Starting MySQL as a Windows Service”).

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

If you installed MySQL using MySQL Installer (see Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”), the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.6`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 5.6"
```

A common installation location for installation from a Zip package 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 5.5.6, “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 |
| test               |
```

```
+--------------------+
```

The list of installed databases may vary, but will always include the minimum of `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 installed MySQL using MySQL Installer, the `root` user will have been 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 |
| test               |
+--------------------+
```

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              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| innodb_index_stats        |
| innodb_table_stats        |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| proxies_priv              |
| 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 | mysql_native_password |
+------+-----------+-----------------------+
```

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

5.8 Upgrading MySQL on Windows

To upgrade MySQL on Windows, follow these steps:

1. Review Section 10.1, “Upgrading MySQL”, for additional information on upgrading MySQL that is not specific to Windows.

2. Always back up your current MySQL installation before performing an upgrade. See Database Backup Methods.

3. Download the latest Windows distribution of MySQL from http://dev.mysql.com/downloads/.

4. Before upgrading MySQL, stop the server. If the server is installed as a service, stop the service with the following command from the command prompt:

   ```
   C:\> NET STOP MySQL
   ```

   If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 5.5 to 5.6, use `mysqladmin` from MySQL 5.5 as follows:

   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 5.5\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.

5. Before upgrading to MySQL 5.6 from a version previous to 4.1.5, or from a version of MySQL installed from a Zip archive to a version of MySQL installed with the MySQL Installation Wizard, you must first manually remove the previous installation and MySQL service (if the server is installed as a service).

   To remove the MySQL service, use the following command:

   ```
   C:\> C:\mysql\bin\mysqld --remove
   ```

   If you do not remove the existing service, the MySQL Installation Wizard may fail to properly install the new MySQL service.

6. If you are using the MySQL Installer, start it as described in Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”.

7. If you are upgrading MySQL from a Zip archive, extract the archive. You may either overwrite your existing MySQL installation (usually located at `C:\mysql`), or install it into a different directory, such as `C:\mysql5`. Overwriting the existing installation is recommended. However, for upgrades (as opposed to installing for the first time), you must remove the data directory from your existing MySQL installation to avoid replacing your current data files. To do so, follow these steps:

a. Unzip the Zip archive in some location other than your current MySQL installation

b. Remove the data directory

c. Rezip the Zip archive

d. Unzip the modified Zip archive on top of your existing installation

Alternatively:

a. Unzip the Zip archive in some location other than your current MySQL installation

b. Remove the data directory

c. Move the data directory from the current MySQL installation to the location of the just-removed data directory

d. Remove the current MySQL installation

e. Move the unzipped installation to the location of the just-removed installation

8. If you were running MySQL as a Windows service and you had to remove the service earlier in this procedure, reinstall the service. (See Section 5.5.7, “Starting MySQL as a Windows Service”.)

9. Restart the server. For example, use `NET START MySQL` if you run MySQL as a service, or invoke `mysqld` directly otherwise.

10. As Administrator, run `mysql_upgrade` to check your tables, attempt to repair them if necessary, and update your grant tables if they have changed so that you can take advantage of any new capabilities. See `mysql_upgrade` — Check and Upgrade MySQL Tables.

11. If you encounter errors, see Section 5.6, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

Chapter 6 Installing MySQL on OS X

Table of Contents

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

MySQL for OS X is available in a number of different forms:

- Native Package Installer, which uses the native OS X installer (DMG) to walk you through the installation of MySQL. For more information, see Section 6.2, “Installing MySQL on OS X Using Native Packages”. You can use the package installer with OS X. 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 will need to open a `Terminal` window. You do not need administrator privileges using this method, as 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 a MySQL Launch Daemon” and Section 6.4, “Installing and Using the MySQL Preference Pane”, both of which simplify the management of your installation.

For additional information on using MySQL on OS X, see Section 6.1, “General Notes on Installing MySQL on OS X”.

6.1 General Notes on Installing MySQL on OS X

You should keep the following issues and notes in mind:

- As of MySQL server 5.6.26, the DMG bundles a launchd daemon instead of the deprecated startup item. Startup items do not function as of OS X 10.10 (Yosemite), so using launchd is preferred. The available MySQL preference pane under OS X **System Preferences** was also updated to use launchd.
- 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.
- 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 will need to either copy the `data` directory from the old version to the new version, or alternatively specify an alternative `datadir` value to set location of the data directory. By default, the MySQL directories are installed under `/usr/local/`.
- 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.

- 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`.
- Prior to OS X 10.7, MySQL server was bundled with OS X Server.

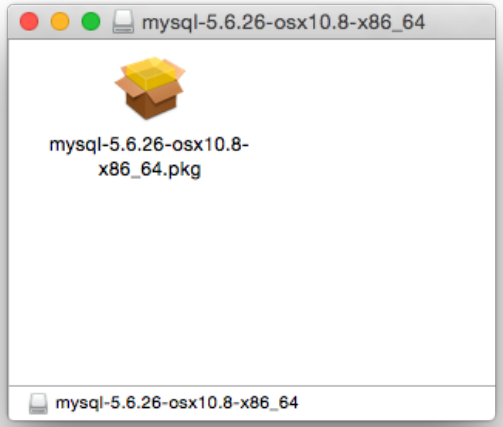
6.2 Installing MySQL on OS X Using Native Packages

> **Note**
>
> Before proceeding with the installation, be sure to stop all running MySQL server instances by using either the MySQL Manager Application (on OS X 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.

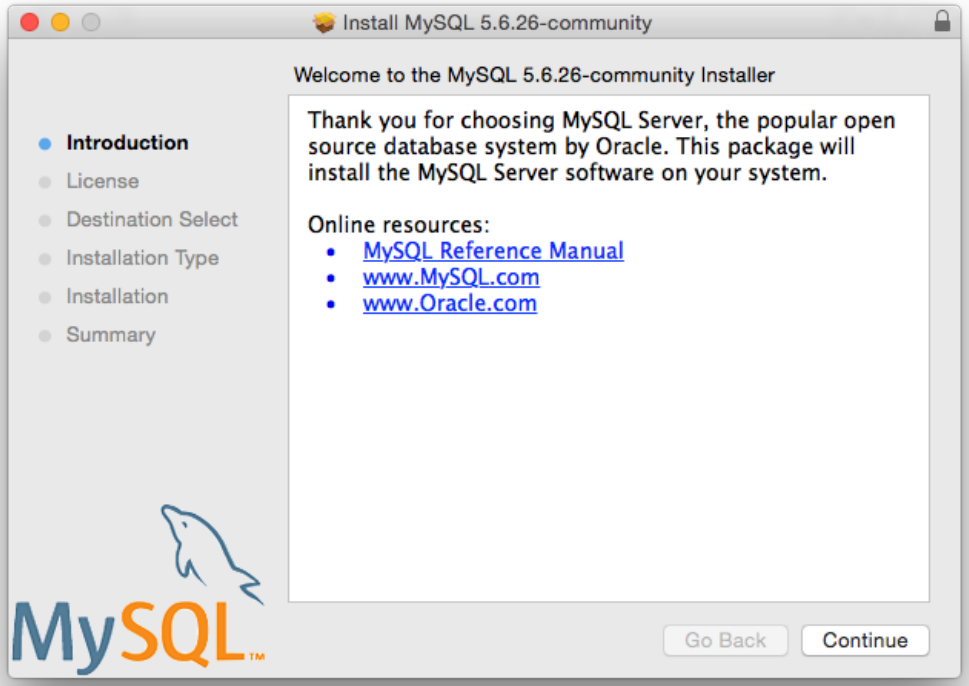
   Figure 6.1 MySQL Package Installer: DMG Contents

2. Double-click the MySQL installer package. It will be named according to the MySQL version and the OS X version you have chosen. For example, if you have downloaded the package for MySQL 5.6.35 and OS X 10.8, double-click `mysql-5.6.35-osx-10.8-x86_64.pkg`.

3. You will be presented with the opening installer dialog. Click **Continue** to begin installation.

Figure 6.2 MySQL Package Installer: Introduction



4. If you have downloaded the community version of MySQL, you will be shown a copy of the relevant GNU General Public License. Click **Continue** and then **Agree** to continue.

5. 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, Preference Pane, Launchd Support -- all enabled by default), or click **Change Installation Location** to change the type of installation, if available.

Figure 6.3 MySQL Package Installer: Installation Type

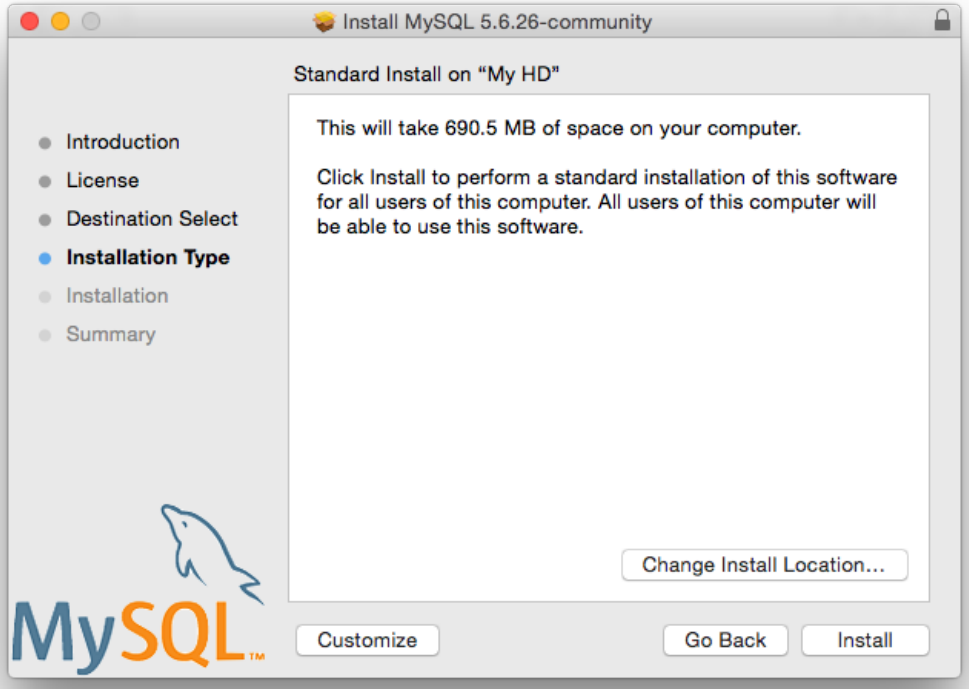


Figure 6.4 MySQL Package Installer: Destination Select (Change Installation Location)

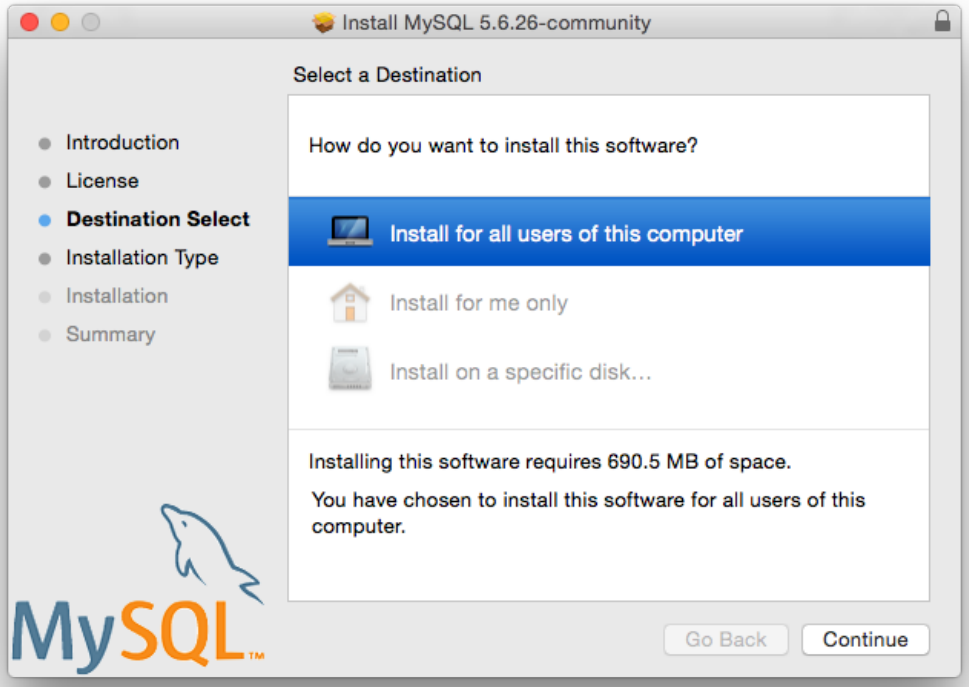
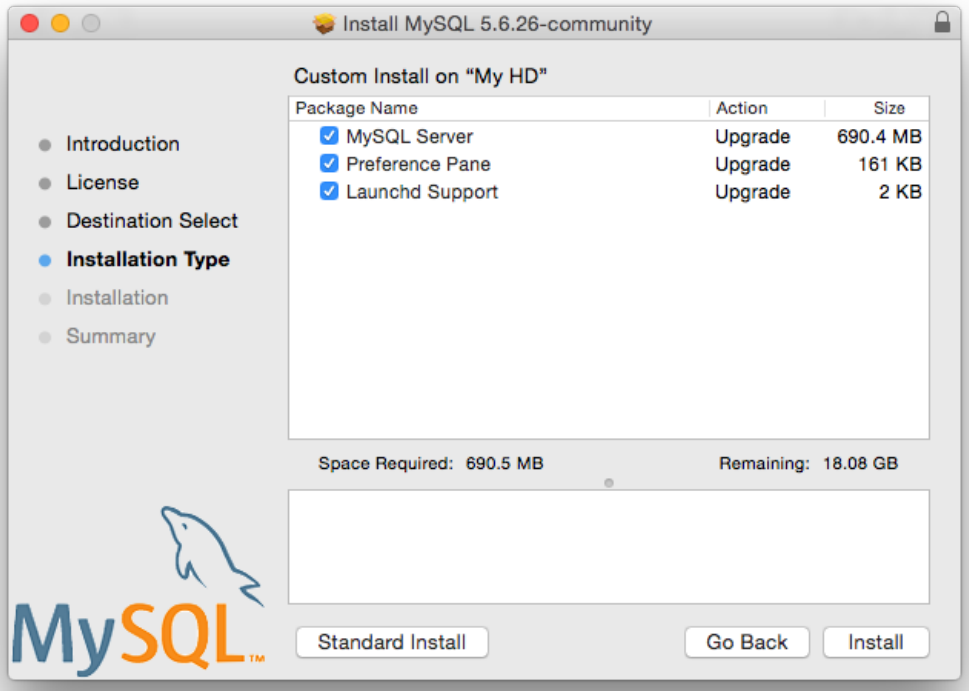


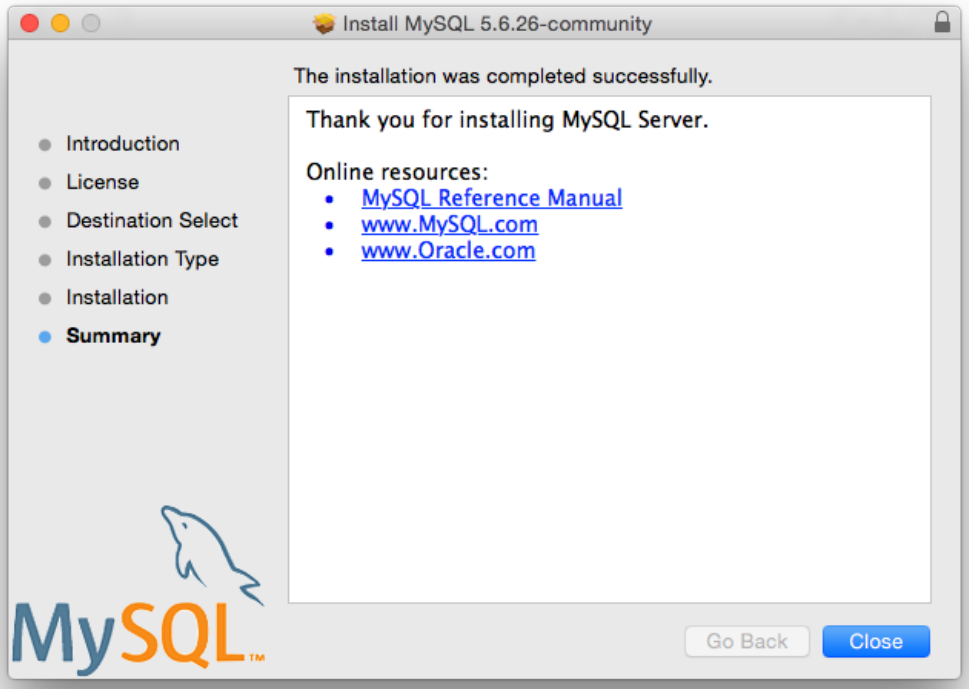
Figure 6.5 MySQL Package Installer: Customize



6. Click **Install** to begin the installation process.

7. Once the installation has been completed successfully, you will be shown an **Install Succeeded** message with a short summary. Now, **Close** the wizard and begin using the MySQL server.

Figure 6.6 MySQL Package Installer: Summary



MySQL server is now installed, but it is not loaded (or started) by default. 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 a 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-5.6.35-osx10.8-x86_64.dmg` installs MySQL into `/usr/local/mysql-5.6.35-osx10.8-x86_64/` . The following table shows the layout of the installation directory.

Table 6.1 MySQL Installation Layout on OS X

Directory	Contents of Directory
`bin`, `scripts`	`mysqld` server, client and utility programs
`data`	Log files, databases
`docs`	Helper documents, like the Release Notes and build information
`include`	Include (header) files
`lib`	Libraries
`man`	Unix manual pages

Directory	Contents of Directory
`mysql-test`	MySQL test suite
`share`	Miscellaneous support files, including error messages, sample configuration files, SQL for database installation
`sql-bench`	Benchmarks
`support-files`	Scripts and sample configuration files
`/tmp/mysql.sock`	Location of the MySQL Unix socket

During the package installer process, a symbolic link from `/usr/local/mysql` to the version/platform specific directory created during installation will be created automatically.

6.3 Installing a MySQL Launch Daemon

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

> **Note**
>
> Before MySQL 5.6.26, the OS X builds installed startup items instead of launchd daemons. However, startup items do not function as of OS X 10.10 (Yosemite). The OS X builds now install launchd daemons.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"
<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>
        </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 60d7963a0bb2994b69b8b9c123db09df.

To enable the launchd service, you can either:

- Click **Start MySQL Server** from the MySQL preference pane.

Figure 6.7 MySQL Preference Pane: Location

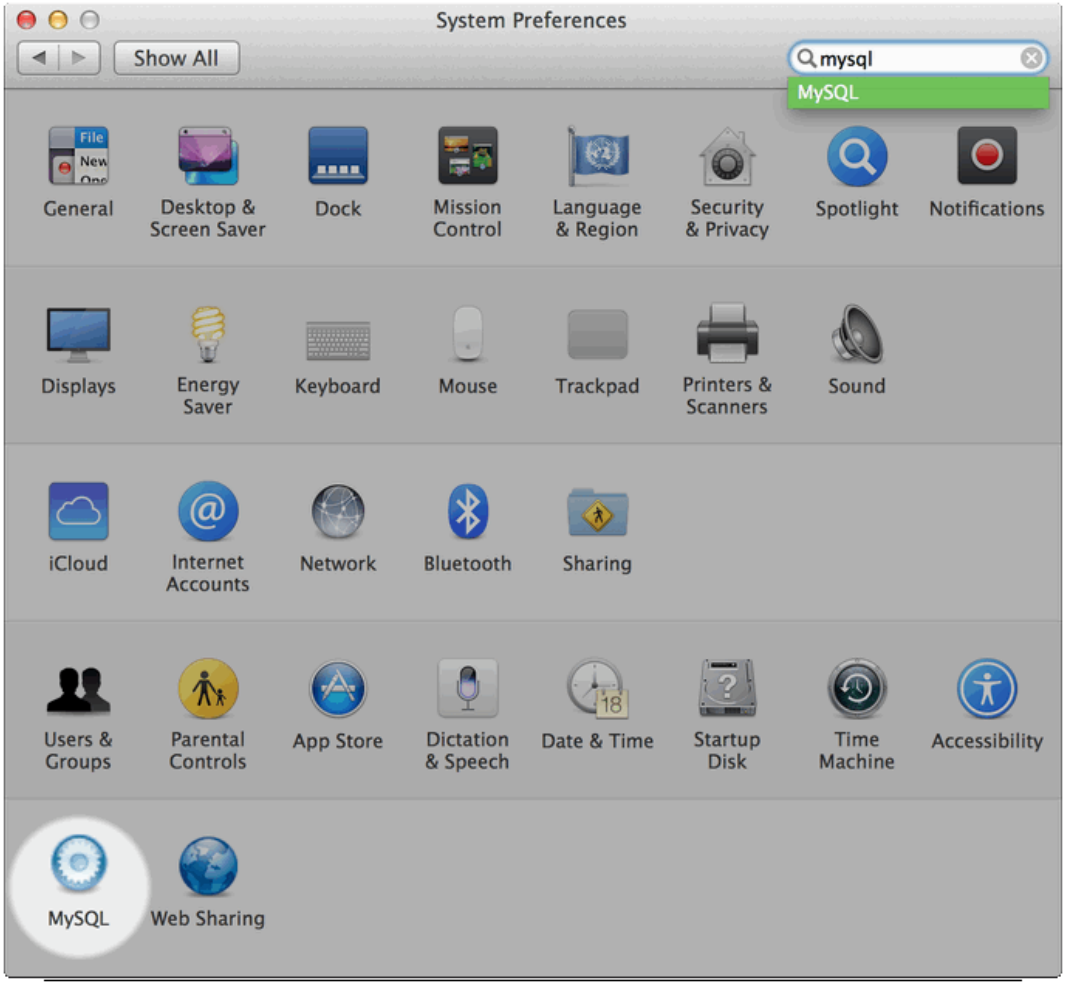
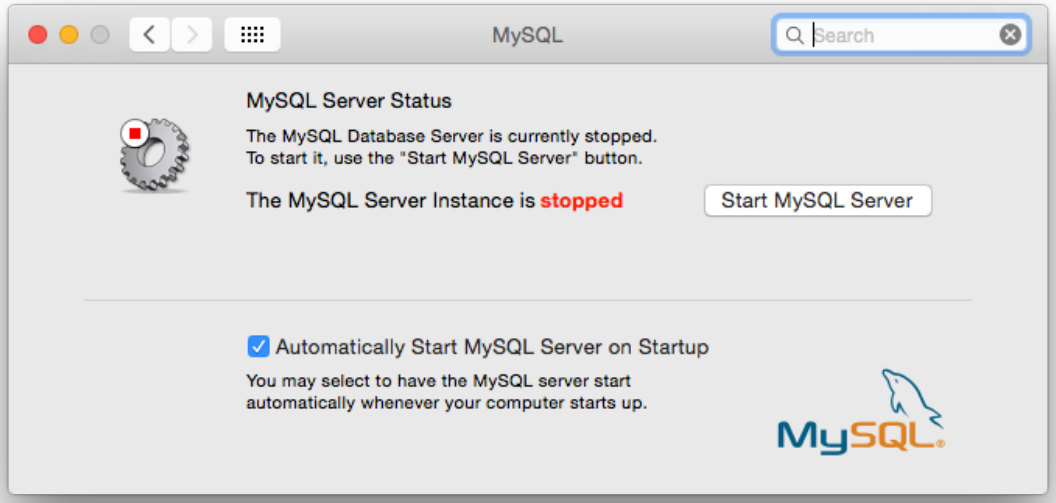


Figure 6.8 MySQL Preference Pane: Usage



- Or, manually load the launchd file.

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

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

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

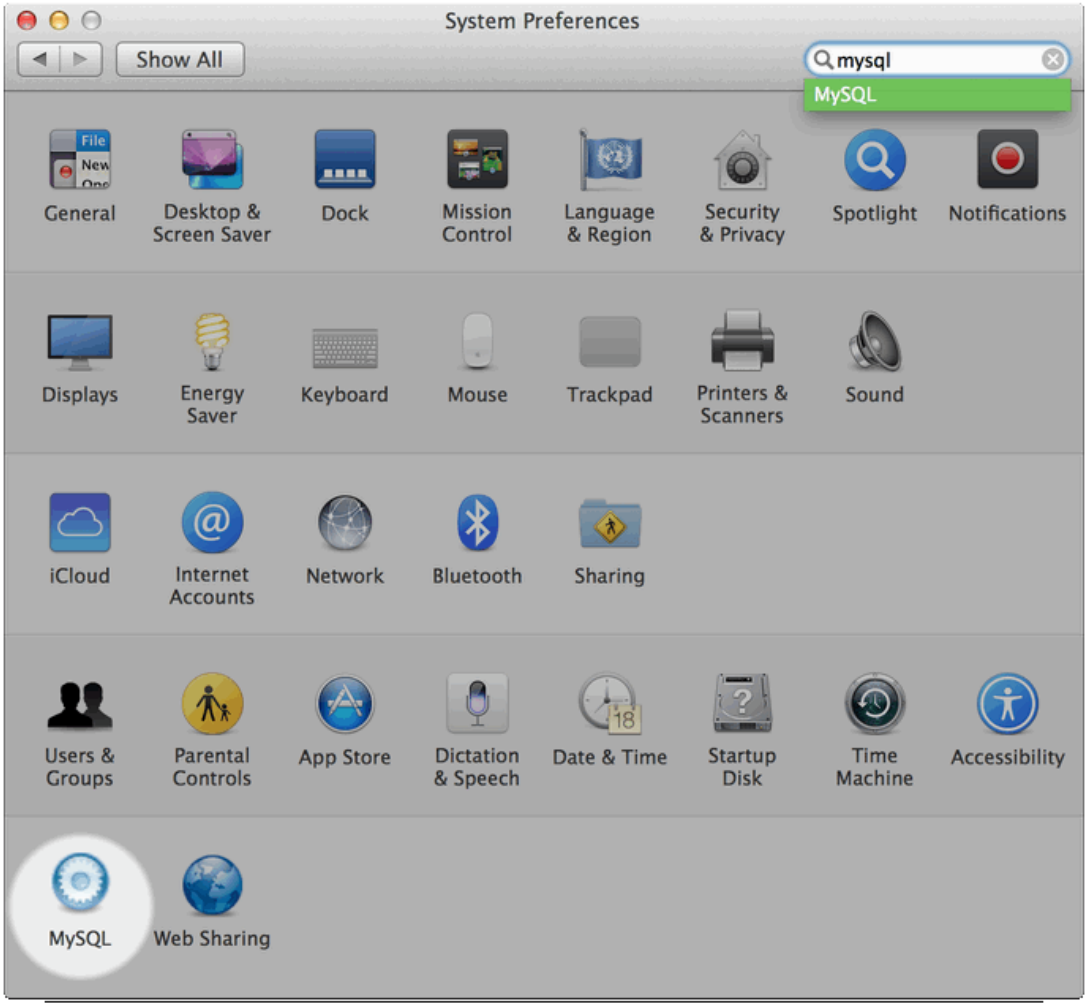
> **Note**
>
> When upgrading MySQL server, the launchd installation process will remove the old startup items that were installed with MySQL server 5.6.25 and below.

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.

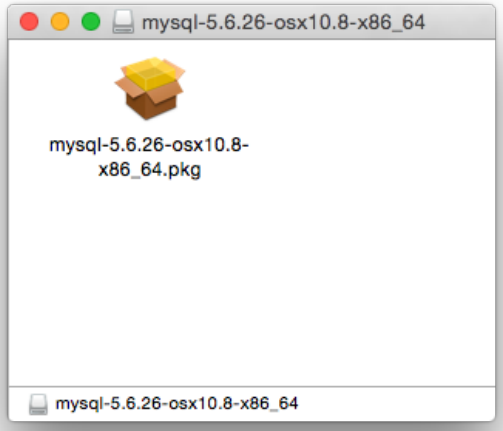
Figure 6.9 MySQL Preference Pane: Location



To install the MySQL Preference Pane:

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.

Figure 6.10 MySQL Package Installer: DMG Contents

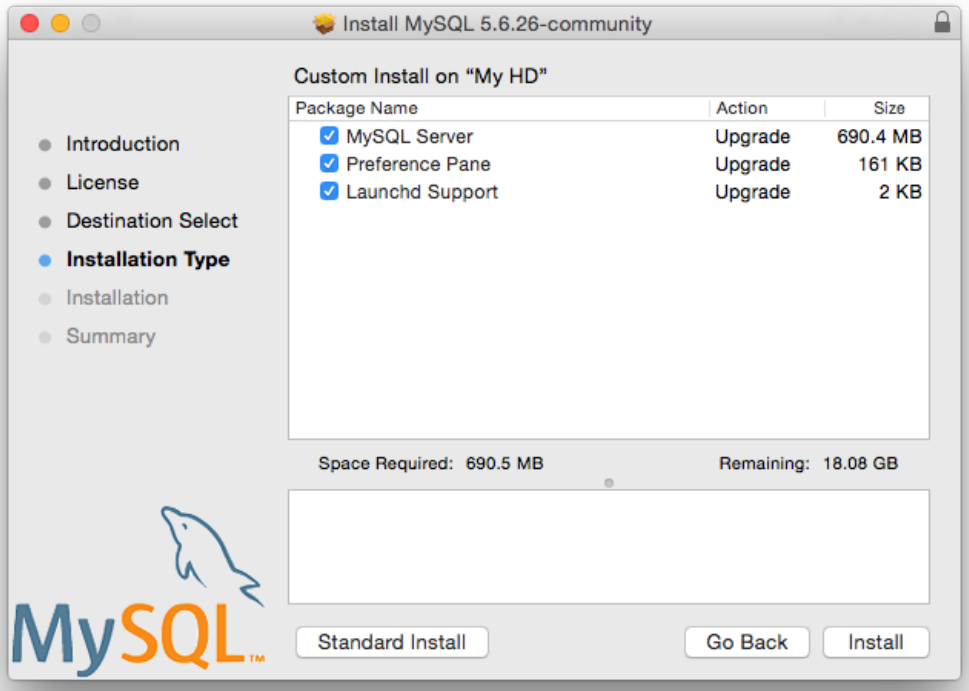


> **Note**
>
> Before MySQL 5.6.26, OS X packages included the deprecated startup items instead of launchd daemons, and the preference pane managed that instead of launchd.

2. Go through the process of installing the MySQL server, as described in the documentation at Section 6.2, “Installing MySQL on OS X Using Native Packages”.

3. Click **Customize** at the **Installation Type** step. The "Preference Pane" option is listed there and enabled by default; make sure it is not deselected.

Figure 6.11 MySQL Installer on OS X: Customize



4. Complete the MySQL server 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 the preference pane. To use the preference pane, open the **System Preferences...** from the Apple menu. Select the MySQL preference pane by clicking the MySQL logo within the **bottom** section of the preference panes list.

Figure 6.12 MySQL Preference Pane: Location

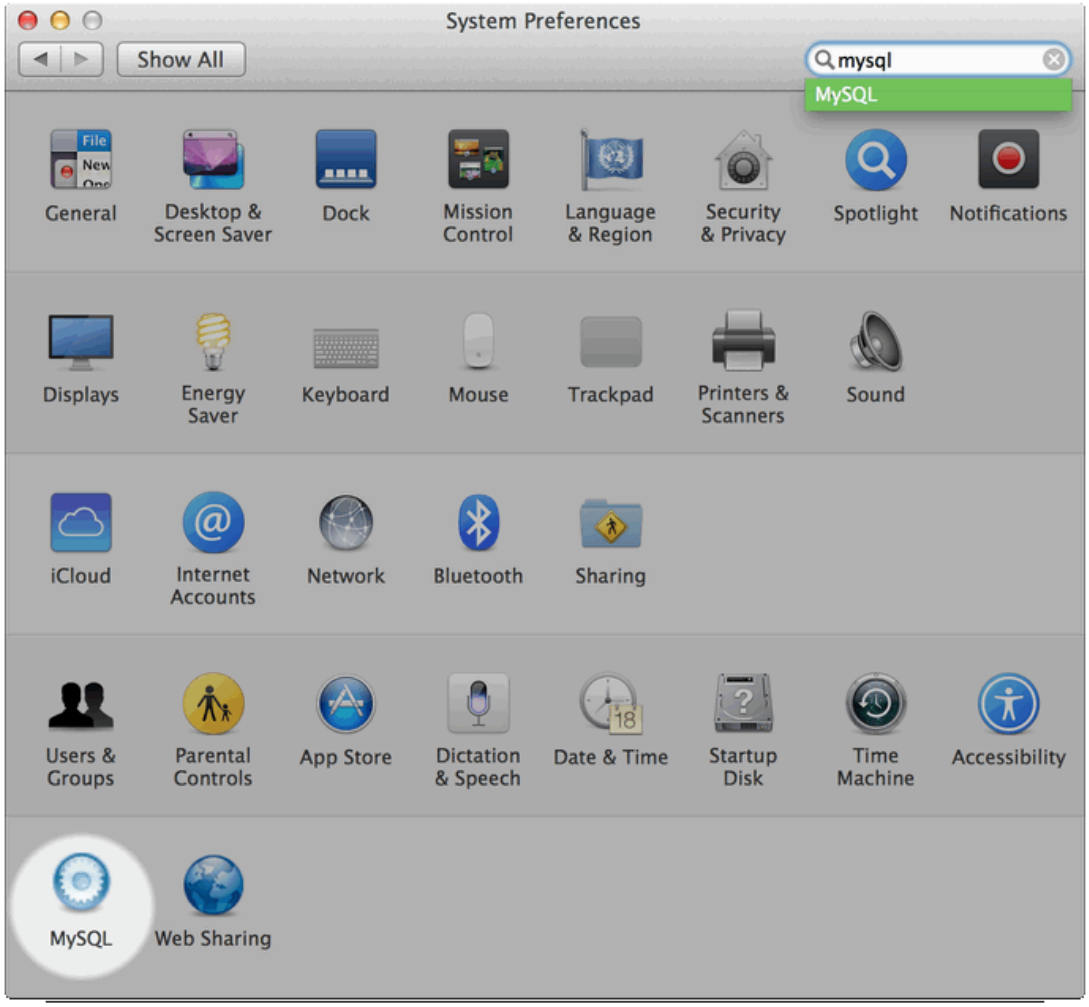
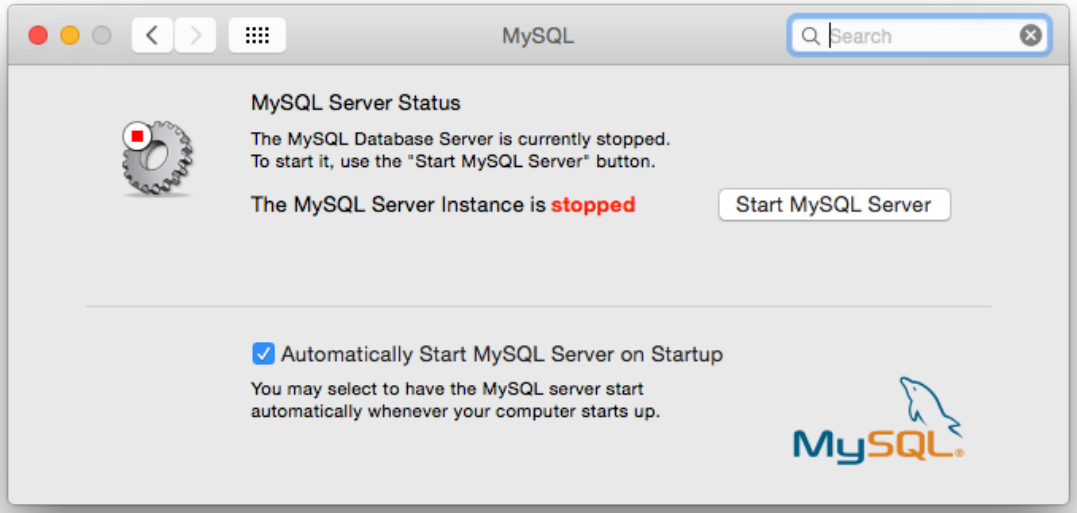


Figure 6.13 MySQL Preference Pane: Usage



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.

- **To start the MySQL server using the preference pane:**

  Click **Start MySQL Server**. You may be prompted for the username and password of a user with administrator privileges to start the MySQL server.

- **To stop the MySQL server using the preference pane:**

  Click **Stop MySQL Server**. You may be prompted for the username and password of a user with administrator privileges to stop the MySQL server.

- **To automatically start the MySQL server when the system boots:**

  Check the check box next to **Automatically Start MySQL Server on Startup**.

- **To disable automatic MySQL server startup when the system boots:**

  Uncheck the check box next to **Automatically Start MySQL Server on Startup**.

You can close the `System Preferences...` window once you have completed your settings.

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:

- Installing with Yum using the MySQL Yum repository. For details, see Section 7.1, “Installing MySQL on Linux Using the MySQL Yum Repository”.
- Installing with APT using the MySQL APT Repository. For details, see Section 7.3, “Installing MySQL on Linux Using the MySQL APT Repository”.
- Installing with Zypper using the MySQL SLES Repository. For details, see Section 7.4, “Installing MySQL on Linux Using the MySQL SLES Repository”.
- Installing using a precompiled RPM package. For more information, see Section 7.5, “Installing MySQL on Linux Using RPM Packages from Oracle”.
- Installing using a precompiled Debian package. For more information, see Section 7.6, “Installing MySQL on Linux Using Debian Packages from Oracle”.
- Installing from a generic binary package in `.tar.gz` format. See Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries* for more information.
- Installing using Oracle's Unbreakable Linux Network (ULN). For more information, see Installing MySQL Using Unbreakable Linux Network (ULN).
- Extracting and compiling MySQL from a source distribution. For detailed instructions, see Chapter 4, *Installing MySQL from Source*.

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 will also normally be unable to install development milestone releases (DMRs), as 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 will 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

MySQL provides a Yum-style software repository for the following Linux platforms:

- EL5, EL6, and EL7-based platforms (for example, the corresponding versions of Red Hat Enterprise Linux, Oracle Linux, and CentOS)
- Fedora 23 and 24

Currently, the MySQL Yum repository for the above-mentioned platforms provides RPM packages for installing the MySQL server, client, MySQL Workbench, MySQL Utilities, Connector/ODBC, and Connector/Python (not all packages are available for all the platforms; 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 10.1.2, “Upgrading MySQL with the MySQL Yum Repository” or Section 7.2, “Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository”.

Steps for a Fresh Installation of MySQL

Follow the steps below to install the latest GA release of MySQL (from the MySQL 5.7 series currently) with the MySQL Yum repository:

Adding[1] the MySQL Yum Repository

First, add the MySQL Yum repository to your system's repository list. This is a one-time operation, which can be performed by installing an RPM provided by MySQL. Follow these steps:

a. Go to the Download MySQL Yum Repository page (http://dev.mysql.com/downloads/repo/yum/) in the MySQL Developer Zone.

b. Select and download the release package for your platform.

c. Install the downloaded release package with the following command (except for EL5-based systems), replacing *`platform-and-version-specific-package-name`* with the name of the downloaded RPM package:

```
shell> sudo yum localinstall platform-and-version-specific-package-name.rpm
```

For an EL6-based system, the command is in the form of:

```
shell> sudo yum localinstall mysql57-community-release-el6-{version-number}.noarch.rpm
```

For an EL7-based system:

```
shell> sudo yum localinstall mysql57-community-release-el7-{version-number}.noarch.rpm
```

For Fedora 23:

```
shell> sudo dnf install mysql57-community-release-fc23-{version-number}.noarch.rpm
```

For Fedora 24:

```
shell> sudo dnf install mysql57-community-release-fc24-{version-number}.noarch.rpm
```

For an EL5-based system, use the following command instead:

```
shell> sudo rpm -Uvh mysql57-community-release-el5-{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 2.3.2, “Signature Checking Using GnuPG” for details on GnuPG key checking.

You can check that the MySQL Yum repository has been successfully added by the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

> **Note**
>
> Once the MySQL Yum repository is enabled on your system, any system-wide update by the `yum update` command (or `dnf upgrade` for dnf-enabled systems) will upgrade MySQL packages on your system and also replace any native third-party packages, if Yum finds replacements for them in the MySQL Yum repository; see Section 10.1.2, “Upgrading MySQL with the MySQL Yum Repository” and, 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 GA series (currently MySQL 5.7) 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, different release series of the MySQL Community Server are hosted in different subrepositories. The subrepository for the latest GA series (currently MySQL 5.7) is enabled by default, and the subrepositories for all other series (for example, the MySQL 5.6 series) are disabled by default. Use this command to see all the subrepositories in the MySQL Yum repository, and see which of them are enabled or disabled (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

To install the latest release from the latest GA series, no configuration is needed. To install the latest release from a specific series other than the latest GA series, disable the subrepository for the latest GA series and enable the subrepository for the specific series before running the installation command. If your platform supports `yum-config-manager`, you can do that by issuing these commands, which disable the subrepository for the 5.7 series and enable the one for the 5.6 series:

```
shell> sudo yum-config-manager --disable mysql57-community
```

```
shell> sudo yum-config-manager --enable mysql56-community
```

For dnf-enabled platforms:

```
shell> sudo dnf config-manager --disable mysql57-community
shell> sudo dnf config-manager --enable mysql56-community
```

Besides using `yum-config-manager` or the `dnf config-manager` command, you can also select a release series by editing manually the `/etc/yum.repos.d/mysql-community.repo` file. This is a typical entry for a release series' subrepository in the file:

```
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/6/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

Find the entry for the subrepository you want to configure, and edit the `enabled` option. Specify `enabled=0` to disable a subrepository, or `enabled=1` to enable a subrepository. For example, to install MySQL 5.6, make sure you have `enabled=0` for the above subrepository entry for MySQL 5.7, and have `enabled=1` for the entry for the 5.6 series:

```
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/6/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

You should only enable subrepository for one release series at any time. When subrepositories for more than one release series are enabled, the latest series will be used by Yum.

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`):

```
shell> yum repolist enabled | grep mysql
```

3. Installing MySQL

Install MySQL by the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> 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`).

4. Starting the MySQL Server

Start the MySQL server with the following command:

```
shell> sudo service mysqld start
```

This is a sample output of the above command:

```
Starting mysqld:[ OK ]
```

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

```
shell> sudo service mysqld status
```

This is a sample output of the above command:

```
mysqld (pid 3066) is running.
```

Securing the MySQL Installation

The program `mysql_secure_installation` allows you to perform important operations like setting the root password, removing anonymous users, and so on. Always run it to secure your MySQL installation:

```
shell> mysql_secure_installation
```

It is important to remember the root password you set. See `mysql_secure_installation` — Improve MySQL Installation Security for details.

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 will not be able to 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`):

```
shell> 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`):

```
shell> sudo yum install package-name
```

For example, to install MySQL Workbench on Fedora 24:

```
shell> sudo dnf install mysql-workbench-community
```

To install the shared client libraries (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> sudo yum install mysql-community-libs
```

Updating MySQL with Yum

Besides installation, you can also perform updates for MySQL products and components using the MySQL Yum repository. See Section 10.1.2, “Upgrading MySQL with the MySQL Yum Repository” for details.

7.2 Replacing a Third-Party Distribution of MySQL Using 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 replace a third-party distribution of MySQL with the latest GA release (from the MySQL 5.7 series currently) from the MySQL Yum repository. According to how your third-party distribution of MySQL was installed, there are different steps to follow:

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 will replace your native, third-party MySQL with the latest GA release (from the MySQL 5.7 series currently) 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.

Replacing a Nonnative Third-Party Distribution of MySQL

If you have installed a third-party distribution of MySQL from a nonnative software repository (that is, a software repository not 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.

2.4 Stopping Yum from Receiving MySQL Packages from Third-Party, Nonnative Repositories

Before you can use the MySQL Yum repository for installing MySQL, you must stop your system from receiving MySQL packages from any third-party, nonnative Yum repositories.

For example, if you have installed MariaDB using their own software repository, get a list of the installed MariaDB packages using the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> yum list installed mariadb\*
```

This is a sample output for the command:

```
MariaDB-common.i686                     10.0.4-1                        @mariadb
MariaDB-compat.i686                     10.0.4-1                        @mariadb
MariaDB-server.i686                     10.0.4-1                        @mariadb
```

From the command output, we can identify the installed packages (`MariaDB-common`, `MariaDB-compat`, and `MariaDB-server`) and the source of them (a nonnative software repository named `mariadb`).

As another example, if you have installed Percona using their own software repository, get a list of the installed Percona packages using the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> yum list installed Percona\*
```

This is a sample output for the command:

```
Percona-Server-client-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
Percona-Server-server-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
Percona-Server-shared-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
percona-release.noarch            0.1-3                       @/percona-release-0.1-3.noarch
```

From the command output, we can identify the installed packages (`Percona-Server-client`, `Percona-Server-server`, `Percona-Server-shared`, and `percona-release.noarch`) and the source of them (a nonnative software repository named `percona-release`).

If you are not sure which third-party MySQL fork you have installed, this command should reveal it and list the RPM packages installed for it, as well as the third-party repository that supplies the packages (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> yum --disablerepo=\* provides mysql\*
```

The next step is to stop Yum from receiving packages from the nonnative repository. If the `yum-config-manager` utility is supported on your platform, you can, for example, use this command for stopping delivery from MariaDB (on dnf-enabled systems, use the `dnf config-manager` command instead of `yum-config-manager`):

```
shell> sudo yum-config-manager --disable mariadb
```

And use this command for stopping delivery from Percona (on dnf-enabled systems, use the `dnf config-manager` command instead of `yum-config-manager`):

```
shell> sudo yum-config-manager --disable percona-release
```

You can perform the same task by removing the entry for the software repository existing in one of the repository files under the `/etc/yum.repos.d/` directory. This is how the entry typically looks like for MariaDB:

```
[mariadb] name = MariaDB
 baseurl = [base URL for repository]
 gpgkey = [URL for GPG key]
 gpgcheck =1
```

The entry is usually found in the file `/etc/yum.repos.d/MariaDB.repo` for MariaDB—delete the file, or remove entry from it (or from the file in which you find the entry).

> **Note**
>
> This step is not necessary for an installation that was configured with a Yum repository release package (like Percona) if you are going to remove the release package (`percona-release.noarch` for Percona), as shown in the uninstall command for Percona in Step 3 below.

3 Uninstalling the Nonnative Third-Party MySQL Distribution of MySQL

The nonnative third-party MySQL distribution must first be uninstalled before you can use the MySQL Yum repository to install MySQL. For the MariaDB packages found in Step 2 above, uninstall them with the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> sudo yum remove MariaDB-common MariaDB-compat MariaDB-server
```

For the Percona packages we found in Step 2 above (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
shell> sudo yum remove Percona-Server-client-55 Percona-Server-server-55 \
  Percona-Server-shared-55.i686 percona-release
```

4 Installing MySQL with the MySQL Yum Repository

Then, install MySQL with the MySQL Yum repository by following the instructions given in Section 7.1, “Installing MySQL on Linux Using the MySQL Yum Repository”: .

> **Important**
>
> - If you have chosen to replace your third-party MySQL distribution with a newer version of MySQL from the MySQL Yum repository, remember to run `mysql_upgrade` after the server starts, to check and possibly resolve any incompatibilities between the old data and the upgraded software. `mysql_upgrade` also performs other functions; see `mysql_upgrade` — Check and Upgrade MySQL Tables for details.
> - *For EL7-based platforms:* See Compatibility Information for EL7-based platforms [125].

7.3 Installing MySQL on Linux Using the MySQL APT Repository

The MySQL APT repository provides `deb` packages for installing and managing the MySQL server, client, and other components on the following Linux platforms: :

- Debian 7.x (“wheezy”)
- Debian 8.x (“jessie”)
- Ubuntu 12.04 LTS (“Precise Pangolin”)
- Ubuntu 14.04 LTS (“Trusty Tahr”)
- Ubuntu 15.10 (“Wily Werewolf”)

Instructions for using the MySQL APT Repository are available in A Quick Guide to Using the MySQL APT Repository.

7.4 Installing MySQL on Linux Using the MySQL SLES Repository

The MySQL SLES repository provides RPM packages for installing and managing the MySQL server, client, and other components on SUSE Enterprise Linux Server.

Instructions for using the MySQL SLES repository are available in A Quick Guide to Using the MySQL SLES Repository.

> **Note**
>
> The MySQL SLES repository is now in development release. We encourage you to try it and provide us with feedback. Please report any bugs or inconsistencies you observe to our Bugs Database.

7.5 Installing MySQL on Linux Using RPM Packages from Oracle

> **Note**
>
> To install or upgrade to MySQL 5.6.11 or later, be sure to read the special instructions at the end of this section.

The recommended way to install MySQL on RPM-based Linux distributions that use `glibc` is by using the RPM packages provided by Oracle. There are two sources for obtaining the Community versions of the RPM packages:

- From the MySQL software repositories, for the following platforms:
  - For EL5, EL6, or EL7-based platforms and Fedora 23 or 24, use the MySQL Yum repository (see Section 7.1, “Installing MySQL on Linux Using the MySQL Yum Repository” for details).
  - For SUSE Enterprise Linux Server, use the MySQL SLES repository (see Section 7.4, “Installing MySQL on Linux Using the MySQL SLES Repository” for details).
- From the MySQL Downloads page in the MySQL Developer Zone, which provides RPM packages that work for different platforms.

The discussion in this section applies only to the RPM packages downloaded from the MySQL Developer Zone. Installations created with these packages result in files under the system directories shown in the following table.

Table 7.1 MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone

Directory	Contents of Directory
`/usr/bin`	Client programs and scripts

Directory	Contents of Directory
`/usr/sbin`	The `mysqld` server
`/var/lib/mysql`	Log files, databases
`/var/lib/mysql-files`	Value of `secure_file_priv`
`/usr/share/info`	MySQL manual in Info format
`/usr/share/man`	Unix manual pages
`/usr/include/mysql`	Include (header) files
`/usr/lib/mysql`	Libraries
`/usr/share/mysql`	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation
`/usr/share/sql-bench`	Benchmarks

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 instructions in this manual do not necessarily apply to installing them. The vendor's instructions should be consulted instead. Because of these differences, RPM packages built by Oracle check whether such RPMs built by other vendors are installed. If so, the RPM does not install and produces a message explaining this.

Conflicts can arise when an RPM from another vendor is already installed, such as when a vendor's conventions about which files belong with the server and which belong with the client library differ from the breakdown used for Oracle packages. In such cases, attempts to install an Oracle RPM with `rpm -i` may result in messages that files in the RPM to be installed conflict with files from an installed package (denoted `mysql-libs` in the following paragraphs).

Each MySQL release provides a `MySQL-shared-compat` package that is meant to replace `mysql-libs` and provides a replacement-compatible client library for older MySQL series. `MySQL-shared-compat` is set up to make `mysql-libs` obsolete, but `rpm` explicitly refuses to replace obsoleted packages when invoked with `-i` (unlike `-U`), which is why installation with `rpm -i` produces a conflict.

`MySQL-shared-compat` can safely be installed alongside `mysql-libs` because libraries are installed to different locations. Therefore, it is possible to install `MySQL-shared-compat` first, then manually remove `mysql-libs` before continuing with the installation. After `mysql-libs` is removed, the dynamic linker stops looking for the client library in the location where `mysql-libs` puts it, and the library provided by the `MySQL-shared-compat` package takes over.

Another alternative is to install packages using `yum`. In a directory containing all RPM packages for a MySQL release, `yum install MySQL*rpm` installs them in the correct order and removes `mysql-libs` in one step without conflicts.

In most cases, you need install only the `MySQL-server` and `MySQL-client` packages to get a functional standard MySQL installation. The other packages are not required for a standard installation.

As of MySQL 5.6.8, new RPM install operations (not upgrades) invoke `mysql_install_db` with the `--random-passwords` option that provides for more secure MySQL installation. Invoking

`mysql_install_db` with `--random-passwords` causes it to assign a random password to the MySQL `root` accounts, set the “password expired” flag for those accounts, and not create anonymous-user MySQL accounts. It will be necessary after installation to start the server, connect as `root` using the initial random password, and assign a new `root` password. Until this is done, `root` cannot do anything else. This must be done for each `root` account you intend to use. To change the password, you can use the `SET PASSWORD` statement (for example, with the `mysql` client). You can also use `mysqladmin` or `mysql_secure_installation`. For additional details, see `mysql_install_db` — Initialize MySQL Data Directory. (Install operations using RPMs for Unbreakable Linux Network are unaffected because they do not use `mysql_install_db`.)

> **Important**
>
> **RPMs for MySQL Cluster.** Standard MySQL server RPMs built by MySQL do not provide support for the `NDBCLUSTER` storage engine. For more information about installing MySQL Cluster from RPMs, see MySQL Cluster Installation.
>
> When upgrading a MySQL Cluster RPM installation, you must upgrade *all* installed RPMs, including the `Server` and `Client` RPMs.

For upgrades, if your installation was originally produced by installing multiple RPM packages, it is best to upgrade all the installed packages, not just some. For example, if you previously installed the server and client RPMs, do not upgrade just the server RPM.

If the data directory exists at RPM installation time, the installation process does not modify existing data. This has the effect, for example, that accounts in the grant tables are not initialized to the default set of accounts.

If you get a dependency failure when trying to install MySQL packages (for example, `error: removing these packages would break dependencies: libmysqlclient.so.10 is needed by ...`), you should also install the `MySQL-shared-compat` package, which includes the shared libraries for older releases for backward compatibility.

The following list shows the available RPM packages. The names shown here use a suffix of `.linux_glibc2.5.i386.rpm`, but particular packages can have different suffixes, described later. If you plan to install multiple RPM packages, you may wish to download the RPM Bundle `tar` file instead, which contains multiple RPM packages so that you need not download them separately.

- `MySQL-server-`*`VERSION`*`.linux_glibc2.5.i386.rpm`

  The MySQL server. You need this unless you only want to connect to a MySQL server running on another machine.

- `MySQL-client-`*`VERSION`*`.linux_glibc2.5.i386.rpm`

  The standard MySQL client programs. You probably always want to install this package.

- `MySQL-devel-`*`VERSION`*`.linux_glibc2.5.i386.rpm`

  The libraries and include files needed to compile other MySQL clients, such as the Perl MySQL module. Install this RPM if you intend to compile C API applications.

- `MySQL-shared-`*`VERSION`*`.linux_glibc2.5.i386.rpm`

  The shared libraries (`libmysqlclient.so*`) that certain languages and applications need to dynamically load and use MySQL. It contains single-threaded and thread-safe libraries. Install this RPM if you intend to compile or run C API applications that depend on the shared client library.

- `MySQL-shared-compat-VERSION.linux_glibc2.5.i386.rpm`

  The shared libraries for older releases, but not the libraries for the current release. It contains single-threaded and thread-safe libraries. Install this package if you have applications installed that are dynamically linked against older versions of MySQL but you want to upgrade to the current version without breaking the library dependencies.

  As of MySQL 5.6.5, the `MySQL-shared-compat` RPM package enables users of Red Hat-provided `mysql-*-5.1` RPM packages to migrate to Oracle-provided `MySQL-*-5.5` packages. `MySQL-shared-compat` replaces the Red Hat `mysql-libs` package by replacing `libmysqlclient.so` files of the latter package, thus satisfying dependencies of other packages on `mysql-libs`. This change affects only users of Red Hat (or Red Hat-compatible) RPM packages. Nothing is different for users of Oracle RPM packages.

- `MySQL-embedded-VERSION.linux_glibc2.5.i386.rpm`

  The embedded MySQL server library.

- `MySQL-test-VERSION.linux_glibc2.5.i386.rpm`

  The MySQL test suite.

- `MySQL-VERSION.src.rpm`

  The source code for all of the previous packages. It can also be used to rebuild the RPMs on other architectures (for example, SPARC).

In RPM package names, the suffix (following the *VERSION* value) has the following syntax:

```
.PLATFORM.CPU.rpm
```

The *PLATFORM* and *CPU* values indicate the type of system for which the package is built. *PLATFORM* indicates the platform and *CPU* indicates the processor type or family.

All packages are dynamically linked against `glibc` 2.5. The *PLATFORM* value indicates whether the package is platform independent or intended for a specific platform, as shown in the following table.

Table 7.2 MySQL Linux RPM Package Platforms

***PLATFORM* Value**	**Intended Use**
`linux_glibc25`	Platform independent, should run on any Linux distribution that supports `glibc` 2.5
`rhel5`	Red Hat Enterprise Linux 5
`el6`, `el7`	Enterprise Linux 6 or 7
`sles11`, `sles12`	SUSE Linux Enterprise Server 11 or 12

In MySQL 5.6, only `linux_glibc2.5` packages are available currently.

The *CPU* value indicates the processor type or family for which the package is built, as shown in the following table.

Table 7.3 MySQL Linux RPM Package CPU Identifiers

***CPU* Value**	**Intended Processor Type or Family**
`i386`, `i686`	Pentium processor or better, 32 bit
`x86_64`	64-bit x86 processor

To see all files in an RPM package (for example, a `MySQL-server` RPM), run a command like this (modify the platform and CPU identifiers appropriately for your system):

```
shell> rpm -qpl MySQL-server-VERSION.linux_glibc2.5.i386.rpm
```

To perform a standard minimal installation, install the server and client RPMs:

```
shell> rpm -i MySQL-server-VERSION.linux_glibc2.5.i386.rpm
shell> rpm -i MySQL-client-VERSION.linux_glibc2.5.i386.rpm
```

To install only the client programs, install just the client RPM:

```
shell> rpm -i MySQL-client-VERSION.linux_glibc2.5.i386.rpm
```

RPM provides a feature to verify the integrity and authenticity of packages before installing them. To learn more about this feature, see Section 2.3, “Verifying Package Integrity Using MD5 Checksums or GnuPG”.

The server RPM places data under the `/var/lib/mysql` directory. The RPM also creates a login account for a user named `mysql` (if one does not exist) to use for running the MySQL server, and creates the appropriate entries in `/etc/init.d/` to start the server automatically at boot time. (This means that if you have performed a previous installation and have made changes to its startup script, you may want to make a copy of the script so that you can reinstall it after you install a newer RPM.) See Section 9.5, “Starting and Stopping MySQL Automatically”, for more information on how MySQL can be started automatically at system startup.

In MySQL 5.6, for a new installation using RPM packages, the server boot scripts are installed, but the MySQL server is not started at the end of the installation, since the status of the server during an unattended installation is not known.

In MySQL 5.6, for an upgrade installation using RPM packages, if the MySQL server is running when the upgrade occurs, the MySQL server is stopped, the upgrade occurs, and the MySQL server is restarted. If the MySQL server is not already running when the RPM upgrade occurs, the MySQL server is not started at the end of the installation.

> **Note**
>
> Upgrading from a community version to a commercial version of MySQL requires that you first uninstall the community version and then install the commercial version. In this case, you must restart the server manually after the upgrade.

If something goes wrong, you can find more information in the binary installation section. See Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*.

> **Note**
>
> The accounts created in the MySQL grant tables for an RPM installation initially have no passwords. After starting the server, you should assign passwords to them using the instructions in Chapter 9, *Postinstallation Setup and Testing*.

An RPM installation creates a user named `mysql` and a group named `mysql` on the system using the `useradd`, `groupadd`, and `usermod` commands. Those commands require appropriate administrative privileges, which is required for locally managed users and groups (as listed in the `/etc/passwd` and `/etc/group` files) by the RPM installation process being run by `root`.

If you log in as the `mysql` user, you may find that MySQL displays “Invalid (old?) table or database name” errors that mention `.mysqlgui`, `lost+found`, `.mysqlgui`, `.bash_history`, `.fonts.cache-1`, `.lesshst`, `.mysql_history`, `.profile`, `.viminfo`, and similar files created by MySQL or operating

system utilities. You can safely ignore these error messages or remove the files or directories that cause them if you do not need them.

For nonlocal user management (LDAP, NIS, and so forth), the administrative tools may require additional authentication (such as a password), and will fail if the installing user does not provide this authentication. Even if they fail, the RPM installation will not abort but succeed, and this is intentional. If they failed, some of the intended transfer of ownership may be missing, and it is recommended that the system administrator then manually ensures some appropriate user and group exists and manually transfers ownership following the actions in the RPM spec file.

In MySQL 5.6.11, the RPM spec file has been updated, which has the following consequences:

- For a non-upgrade installation (no existing MySQL version installed), it is possible to install MySQL using `yum`.
- For upgrades, it is necessary to clean up any earlier MySQL installations. In effect, the update is performed by removing the old installations and installing the new one.

Additional details follow.

For a non-upgrade installation of MySQL 5.6.11 or later, it is possible to install using `yum`:

```
shell> yum install MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
```

For upgrades to MySQL 5.6.11 or later, perform the upgrade by removing the old installation and installing the new one:

1. Remove the existing 5.6.*X* installation. *`OLDVERSION`* is the version to remove.

   ```
   shell> rpm -e MySQL-server-OLDVERSION.linux_glibc2.5.i386.rpm
   ```

   Repeat this step for all installed MySQL RPMs.

2. Install the new version. *`NEWVERSION`* is the version to install.

   ```
   shell> rpm -ivh MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
   ```

Alternatively, the removal and installation can be done using `yum`:

```
shell> yum remove MySQL-server-OLDVERSION.linux_glibc2.5.i386.rpm
shell> yum install MySQL-server-NEWVERSION.linux_glibc2.5.i386.rpm
```

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

7.6 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, supporting the Debian 7 and 8, and Ubuntu 12, 14, and 15 platforms. For details, see Section 7.3, “Installing MySQL on Linux Using the MySQL APT Repository”.
- The MySQL Developer Zone's Download Area. For details, see Section 2.2, “How to Get MySQL”. The following are some information on the Debian packages available there and the instructions for installing them:

- You may also need to install the `libaio` library if it is not already present on your system:

```
shell> sudo apt-get install libaio1
```

- *For Debian 7 and 8, and Ubuntu 12, 14, and 15*:

  - Various Debian packages are provided in the MySQL Developer Zone for installing different components of MySQL. 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 7.4 MySQL Debian 7 and 8, and Ubuntu 12, 14, and 15 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:

```
shell> tar -xvf mysql-server_MVER-DVER_CPU.deb-bundle.tar
```

  - In general, install the `deb` packages unpacked from the tarball with the command (see explanations below for the extra steps required for installing the server package):

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

    There are four packages to install:

    - The database common files (install this package before the other ones):

```
shell> sudo dpkg -i mysql-common_MVER-DVER_CPU.deb
```

    - The MySQL server:

      Install first the package for the database common files (see the last bullet), and then pre-configure your server installation by the following command:

```
shell> sudo dpkg-preconfigure mysql-community-server_MVER-DVER_CPU.deb
```

      You will be 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**. However, it is very important that you set the password soon using the program mysql_secure_installation, as people can gain anonymous access to your MySQL server until you have secured the database's root account with a password.

      Next, install the server package with the following command:

```
shell> sudo dpkg -i mysql-community-server_MVER-DVER_CPU.deb
```

- The MySQL client:

```
shell> sudo dpkg -i mysql-community-client_MVER-DVER_CPU.deb
```

- The MySQL shared client library:

```
shell> sudo dpkg -i libmysqlclient18_MVER-DVER_CPU.deb
```

Here are where the files are installed on the system:

- All configuration files (like `my.cnf`) are under `/etc`
- All binaries, libraries, headers, etc., are under `/usr`
- The data directory is under `/var`

- *For Debian 6:*

  - Debian package files directly downloaded from the MySQL Developer Zone have names in the `mysql-MVER-DVER-CPU.deb` format. *MVER* is the MySQL version and *DVER* is the Debian version. The *CPU* value indicates the processor type or family for which the package is built, as shown in the following table:

    Table 7.5 MySQL Debian 6 Installation Package CPU Identifiers

CPU Value	Intended Processor Type or Family
`i686`	Pentium processor or better, 32 bit
`x86_64`	64-bit x86 processor

  - After downloading a Debian package, use the following command to install it;

```
shell> dpkg -i mysql-MVER-DVER-CPU.deb
```

    The Debian package installs files under the `/opt/mysql/server-5.6` directory.

> **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.

7.7 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 will also normally be unable to install development milestone releases

(DMRs), as 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 EL5, EL6, or EL7-based Linux platforms and Fedora 23 or 24, 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:

```
root-shell> 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`. An init script, to start and stop the server, will have been installed into `/etc/init.d/mysqld`. To start the MySQL server use `service`:

```
root-shell> service mysqld start
```

To enable the server to be started and stopped automatically during boot, use `chkconfig`:

```
root-shell> chkconfig --levels 235 mysqld on
```

Which enables the MySQL server to be started (and stopped) automatically at the specified the run levels.

The database tables will have been 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 Debian 7 and 8, and Ubuntu 12, 14, and 15, MySQL can be installed using the MySQL APT Repository instead of the platform's native software repository. See Section 7.3, “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.

  A sample installation of the MySQL packages might look like this (some sections trimmed for clarity):

```
root-shell> apt-get install mysql-client-5.1 mysql-server-5.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.28-11 linux-headers-2.6.28-11-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-common postfix
Suggested packages:
  dbishell libipc-sharedcache-perl tinyca procmail postfix-mysql postfix-pgsql
  postfix-ldap postfix-pcre sasl2-bin resolvconf postfix-cdb
The following NEW packages will be installed
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-client-5.1 mysql-common mysql-server-5.1 postfix
0 upgraded, 13 newly installed, 0 to remove and 182 not upgraded.
Need to get 1907kB/25.3MB of archives.
After this operation, 59.5MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get: 1 http://gb.archive.ubuntu.com jaunty-updates/main mysql-common 5.1.30really5.0.75-0ubuntu10.5 [63.
Get: 2 http://gb.archive.ubuntu.com jaunty-updates/main libmysqlclient15off 5.1.30really5.0.75-0ubuntu10
Fetched 1907kB in 9s (205kB/s)
Preconfiguring packages ...
Selecting previously deselected package mysql-common.
(Reading database ... 121260 files and directories currently installed.)
...
Processing 1 added doc-base file(s)...
Registering documents with scrollkeeper...
Setting up libnet-daemon-perl (0.43-1) ...
Setting up libplrpc-perl (0.2020-1) ...
Setting up libdbi-perl (1.607-1) ...
Setting up libmysqlclient15off (5.1.30really5.0.75-0ubuntu10.5) ...
Setting up libdbd-mysql-perl (4.008-1) ...
Setting up libmysqlclient16 (5.1.31-1ubuntu2) ...
Setting up mysql-client-5.1 (5.1.31-1ubuntu2) ...
Setting up mysql-server-5.1 (5.1.31-1ubuntu2) ...
 * Stopping MySQL database server mysqld
   ...done.
100825 11:46:15  InnoDB: Started; log sequence number 0 46409
100825 11:46:15  InnoDB: Starting shutdown...
100825 11:46:17  InnoDB: Shutdown completed; log sequence number 0 46409
100825 11:46:17 [Warning] Forcing shutdown of 1 plugins
 * Starting MySQL database server mysqld
   ...done.
 * Checking for corrupt, not cleanly closed and upgrade needing tables.
...
Processing triggers for libc6 ...
ldconfig deferred processing now taking place
```

> **Note**
>
> The `apt-get` command will install 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 will be created, and you will be prompted for the MySQL root password (and confirmation). A configuration file will have been created in `/etc/mysql/my.cnf`. An init script will have been created in `/etc/init.d/mysql`.

The server will already be started. You can manually start and stop the server using:

```
root-shell> service mysql [start|stop]
```

The service will automatically be added to the 2, 3 and 4 run levels, with stop scripts in the single, shutdown and restart levels.

- **Gentoo Linux**

  As a source-based distribution, installing MySQL on Gentoo involves downloading the source, patching the Gentoo specifics, and then compiling the MySQL server and installing it. This process is handled automatically by the `emerge` command.

  The MySQL server and client tools are provided within a single package, `dev-db/mysql`. You can obtain a list of the versions available to install by looking at the portage directory for the package:

```
root-shell> ls /usr/portage/dev-db/mysql/mysql-5.6*
mysql-5.6.27.ebuild
mysql-5.6.27-r1.ebuild
mysql-5.6.28.ebuild
```

  To install a specific MySQL version, you must specify the entire atom. For example:

```
root-shell> emerge =dev-db/mysql-5.6.27-r1
```

  After installation, you should initialize the data directory and set the password for the MySQL *`root`* user (see Section 9.1, “Initializing the Data Directory”). Alternatively, use the configuration interface to perform those tasks:

```
root-shell> emerge --config =dev-db/mysql-5.6.27-r1
```

  During installation, a sample configuration file is created for you in `/etc/mysql/my.cnf`, and an init script is created in `/etc/init.d/mysql`.

  To enable MySQL to start automatically at the normal (default) run levels, use this command:

```
root-shell> rc-update add mysql default
```

7.8 Installing MySQL on Linux with docker

The docker deployment framework supports easy installation and configuration of MySQL servers. For instructions, see https://hub.docker.com/r/mysql/mysql-server/. This page also provides extensive documentation about using MySQL under docker.

7.9 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/.

Chapter 8 Installing MySQL on Solaris and OpenSolaris

Table of Contents

MySQL on Solaris and OpenSolaris 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”.
- On OpenSolaris, the standard package repositories include MySQL packages specially built for OpenSolaris that include entries for the Service Management Framework (SMF) to enable control of the installation using the SMF administration commands. For more information, see Section 8.2, “Installing MySQL on OpenSolaris Using IPS”.
- 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, http://dev.mysql.com/downloads/mysql/5.6.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, you may run into trouble even before you get the MySQL distribution unpacked, as the Solaris `tar` cannot handle long file names. This means that you may see errors when you try to unpack MySQL.

  If this occurs, you must use GNU `tar` (`gtar`) to unpack the distribution. In Solaris 10 and OpenSolaris `gtar` is normally located in `/usr/sfw/bin/gtar`, but may not be included in the default path definition.

- When using Solaris 10 for x86_64, 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 will cause 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 will be generated using the mode 600 and owned by the superuser.

8.1 Installing MySQL on Solaris Using a Solaris PKG

You can install MySQL on Solaris and OpenSolaris using a binary package using the native Solaris PKG format instead of the binary tarball distribution.

To use this package, download the corresponding `mysql-VERSION-solaris10-PLATFORM.pkg.gz` file, then uncompress it. For example:

```
shell> gunzip mysql-5.6.35-solaris10-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:

```
shell> pkgadd -d mysql-5.6.35-solaris10-x86_64.pkg
The following packages are available:
  1  mysql     MySQL Community Server (GPL)
               (i86pc) 5.6.35
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:

```
shell> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
shell> 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:

```
shell> mysqladmin shutdown
shell> pkgrm mysql
shell> pkgadd -d mysql-5.6.35-solaris10-x86_64.pkg
shell> mysqld_safe &
shell> mysql_upgrade
```

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

8.2 Installing MySQL on OpenSolaris Using IPS

OpenSolaris includes standard packages for MySQL in the core repository. The MySQL packages are based on a specific release of MySQL and updated periodically. For the latest release you must use either the native Solaris PKG, `tar`, or source installations. The native OpenSolaris packages include SMF files so that you can easily control your MySQL installation, including automatic startup and recovery, using the native service management tools.

To install MySQL on OpenSolaris, use the `pkg` command. You will need to be logged in as root, or use the `pfexec` tool, as shown in the example below:

```
shell> pfexec pkg install SUNWmysql56
```

The package set installs three individual packages, `SUNWmysql56lib`, which contains the MySQL client libraries; `SUNWmysql56r` which contains the root components, including SMF and configuration files; and `SUNWmysql56u` which contains the scripts, binary tools and other files. You can install these packages individually if you only need the corresponding components.

The MySQL files are installed into `/usr/mysql` which symbolic links for the sub directories (`bin`, `lib`, etc.) to a version specific directory. For MySQL 5.6, the full installation is located in `/usr/mysql/5.6`. The default data directory is `/var/mysql/5.6/data`. The configuration file is installed in `/etc/mysql/5.6/my.cnf`. This layout permits multiple versions of MySQL to be installed, without overwriting the data and binaries from other versions.

Once installed, you must initialize the data directory (see Section 9.1, “Initializing the Data Directory”), and use the `mysql_secure_installation` to secure your installation.

Using SMF to manage your MySQL installation

Once installed, you can start and stop your MySQL server using the installed SMF configuration. The service name is `mysql`, or if you have multiple versions installed, you should use the full version name, for example `mysql:version_56`. To start and enable MySQL to be started at boot time:

```
shell> svcadm enable mysql
```

To view the SMF logs, use this command:

```
shell> svcadm enable svc:/application/database/mysql
```

To check whether the MySQL service is running:

```
shell> svcs -xv svc:/application/database/mysql
```

To disable MySQL from starting during boot time, and shut the MySQL server down if it is running:

```
shell> svcadm disable mysql
```

To restart MySQL, for example after a configuration file changes, use the `restart` option:

```
shell> svcadm restart mysql
```

You can also use SMF to configure the data directory and enable full 64-bit mode. For example, to set the data directory used by MySQL:

```
shell> svccfg
svc:> select mysql:version_56
svc:/application/database/mysql:version_56> setprop mysql/data=/data0/mysql
```

By default, the 32-bit binaries are used. To enable the 64-bit server on 64-bit platforms, set the `enable_64bit` parameter. For example:

```
svc:/application/database/mysql:version_56> setprop mysql/enable_64bit=1
```

You must refresh the SMF after setting these options:

```
shell> svcadm refresh mysql
```

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:
  - Installation on Windows
  - 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 OS X using a DMG distribution.

  For other platforms and installation types, including installation from generic binary and source distributions, you must initialize the data directory yourself. For instructions, see Section 9.1, “Initializing the Data Directory”.
- For instructions, see Section 9.2, “Starting the Server”, and Section 9.3, “Testing the Server”.
- Assign passwords to any initial accounts in the grant tables, if that was not already done during data directory initialization. Passwords prevent unauthorized access to the MySQL server. You may also wish to restrict access to test databases. For instructions, see Section 9.4, “Securing the Initial MySQL Accounts”.
- 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 The MySQL Access Privilege System, and MySQL User Account Management.

9.1 Initializing the Data Directory

After installing MySQL, you must initialize the data directory, including the tables in the `mysql` system database. For some MySQL installation methods, data directory initialization may be done automatically,

as described in Chapter 9, *Postinstallation Setup and Testing*. For other installation methods, including installation from generic binary and source distributions, you must initialize the data directory yourself.

This section describes how to initialize the data directory on Unix and Unix-like systems. (For Windows, see Section 5.7, “Windows Postinstallation Procedures”.) 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”.

In the examples shown here, the server runs under the user ID of the `mysql` login account. This assumes that such an account exists. Either create the account if it does not exist, or substitute the name of a different existing login account that you plan to use for running the server. For information about creating the account, see Creating a `mysql` System User and Group, in Chapter 3, *Installing MySQL on Unix/Linux Using Generic Binaries*.

1. Change location into the top-level directory of your MySQL installation, represented here by *`BASEDIR`*:

   ```
   shell> cd BASEDIR
   ```

   `BASEDIR` is likely to be something like `/usr/local/mysql`, `/usr/local`, or `/usr/bin` (for installation wtih MySQL Yum repository, or other means). The following steps assume that you have changed location to this directory.

   You will find several files and subdirectories in the *`BASEDIR`* directory. The most important for installation purposes are the `bin` and `scripts` subdirectories, which contain the server as well as client and utility programs.

2. If necessary, ensure that the distribution contents are accessible to `mysql`. If you installed the distribution as `mysql`, no further action is required. If you installed the distribution as `root`, its contents will be owned by `root`. Change its ownership to `mysql` by executing the following commands as `root` in the installation directory. The first command changes the owner attribute of the files to the `mysql` user. The second changes the group attribute to the `mysql` group.

   ```
   shell> chown -R mysql .
   shell> chgrp -R mysql .
   ```

3. If necessary, initialize the data directory, including the `mysql` database containing the initial MySQL grant tables that determine how users are permitted to connect to the server.

   Typically, data directory initialization need be done only the first time you install MySQL. If you are upgrading an existing installation, you should run `mysql_upgrade` instead (see `mysql_upgrade` — Check and Upgrade MySQL Tables). However, the command that initializes the data directory does not overwrite any existing privilege tables, so it should be safe to run in any circumstances.

   ```
   shell> scripts/mysql_install_db --user=mysql
   ```

   It is important to make sure that the database directories and files are 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 if you run `mysql_install_db` as `root`, 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.

   The `mysql_install_db` command creates the server's data directory. Under the data directory, it creates directories for the `mysql` database that holds the grant tables and the `test` database that you can use to test MySQL. The program also creates privilege table entries for the initial account or accounts. `test_`. For a complete listing and description of the grant tables, see The MySQL Access Privilege System.

It might be necessary to specify other options such as `--basedir` or `--datadir` if `mysql_install_db` does not identify the correct locations for the installation directory or data directory. For example:

```
shell> scripts/mysql_install_db --user=mysql \
         --basedir=/opt/mysql/mysql \
         --datadir=/opt/mysql/mysql/data
```

For a more secure installation, invoke `mysql_install_db` with the `--random-passwords` option. This causes it to assign a random password to the MySQL `root` accounts, set the “password expired” flag for those accounts, and remove the anonymous-user MySQL accounts. For additional details, see `mysql_install_db` — Initialize MySQL Data Directory. (Install operations using RPMs for Unbreakable Linux Network are unaffected because they do not use `mysql_install_db`.)

If you do not want to have the `test` database, you can remove it after starting the server, using the instructions in Section 9.4, “Securing the Initial MySQL Accounts”.

If you have trouble with `mysql_install_db` at this point, see Section 9.1.1, “Problems Running mysql_install_db”.

4. After initializing the data directory, you can establish the final installation ownership settings. To leave the installation owned by `mysql`, no action is required here. Otherwise, most of the MySQL installation can be owned by `root` if you like. The exception is that the data directory must be owned by `mysql`. To accomplish this, run the following commands as `root` in the installation directory. For some distribution types, the data directory might be named `var` rather than `data`; adjust the second command accordingly.

   ```
   shell> chown -R root .
   shell> chown -R mysql data
   ```

   If the plugin directory (the directory named by the `plugin_dir` system variable) is writable by the server, it may be possible for a user to write executable code to a file in the directory using `SELECT ... INTO DUMPFILE`. This can be prevented by making the plugin directory read only to the server or by setting the `secure_file_priv` system variable at server startup to a directory where `SELECT` writes can be performed safely.

5. To specify options that the MySQL server should use at startup, put them in a `/etc/my.cnf` or `/etc/mysql/my.cnf` file. See Server Configuration Defaults. If you do not do this, the server starts with its default settings.

6. If you want MySQL to start automatically when you boot your machine, see Section 9.5, “Starting and Stopping MySQL Automatically”.

Data directory initialization creates time zone tables in the `mysql` database but does not populate them. To do so, use the instructions in MySQL Server Time Zone Support.

9.1.1 Problems Running mysql_install_db

The purpose of the `mysql_install_db` program is to initialize the data directory, including the tables in the `mysql` system database. It does not overwrite existing MySQL privilege tables, and it does not affect any other data.

To re-create your privilege tables, first stop the `mysqld` server if it is running. Then rename the `mysql` directory under the data directory to save it, and run `mysql_install_db`. Suppose that your current directory is the MySQL installation directory and that `mysql_install_db` is located in the `bin` directory

and the data directory is named `data`. To rename the `mysql` database and re-run `mysql_install_db`, use these commands.

```
shell> mv data/mysql data/mysql.old
shell> scripts/mysql_install_db --user=mysql
```

When you run `mysql_install_db`, you might encounter the following problems:

- **`mysql_install_db` fails to install the grant tables**

  You may find that `mysql_install_db` fails to install the grant tables and terminates after displaying the following messages:

  ```
  Starting mysqld daemon with databases from XXXXXX
  mysqld ended
  ```

  In this case, you should examine the error log file very carefully. The log should be located in the directory `XXXXXX` named by the error message and should indicate why `mysqld` did not start. If you do not understand what happened, include the log when you post a bug report. See How to Report Bugs or Problems.

- **There is a `mysqld` process running**

  This indicates that the server is running, in which case the grant tables have probably been created already. If so, there is no need to run `mysql_install_db` at all because it needs to be run only once, when you first install MySQL.

- **Installing a second `mysqld` server does not work when one server is running**

  This can happen when you have an existing MySQL installation, but want to put a new installation in a different location. For example, you might have a production installation, but you want to create a second installation for testing purposes. Generally the problem that occurs when you try to run a second server is that it tries to use a network interface that is in use by the first server. In this case, you should see one of the following error messages:

  ```
  Can't start server: Bind on TCP/IP port:
  Address already in use
  Can't start server: Bind on unix socket...
  ```

  For instructions on setting up multiple servers, see Running Multiple MySQL Instances on One Machine.

- **You do not have write access to the `/tmp` directory**

  If you do not have write access to create temporary files or a Unix socket file in the default location (the `/tmp` directory) or the `TMPDIR` environment variable, if it has been set, an error occurs when you run `mysql_install_db` or the `mysqld` server.

  You can specify different locations for the temporary directory and Unix socket file by executing these commands prior to starting `mysql_install_db` or `mysqld`, where *`some_tmp_dir`* is the full path name to some directory for which you have write permission:

  ```
  shell> TMPDIR=/some_tmp_dir/
  shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
  shell> export TMPDIR MYSQL_UNIX_PORT
  ```

  Then you should be able to run `mysql_install_db` and start the server with these commands:

```
shell> scripts/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
```

If `mysql_install_db` is located in the `scripts` directory, modify the first command to `scripts/mysql_install_db`.

See How to Protect or Change the MySQL Unix Socket File, and Chapter 11, *Environment Variables*.

There are some alternatives to running the `mysql_install_db` program provided in the MySQL distribution:

- If you want the initial privileges to be different from the standard defaults, use account-management statements such as `CREATE USER`, `GRANT`, and `REVOKE` to change the privileges *after* the grant tables have been set up. In other words, run `mysql_install_db`, and then use `mysql -u root mysql` to connect to the server as the MySQL `root` user so that you can issue the necessary statements. (See Account Management Statements.)

  To install MySQL on several machines with the same privileges, put the `CREATE USER`, `GRANT`, and `REVOKE` statements in a file and execute the file as a script using `mysql` after running `mysql_install_db`. For example:

  ```
  shell> scripts/mysql_install_db --user=mysql
  shell> bin/mysql -u root < your_script_file
  ```

  This enables you to avoid issuing the statements manually on each machine.

- It is possible to re-create the grant tables completely after they have previously been created. You might want to do this if you are just learning how to use `CREATE USER`, `GRANT`, and `REVOKE` and have made so many modifications after running `mysql_install_db` that you want to wipe out the tables and start over.

  To re-create the grant tables, stop the server if it is running and remove the `mysql` database directory. Then run `mysql_install_db` again.

9.2 Starting the Server

This section describes how start the server on Unix and Unix-like systems. (For Windows, see Section 5.5.4, “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:

```
shell> bin/mysqld_safe --user=mysql &
```

It is important that the MySQL server be run using an unprivileged (non-`root`) login account. To ensure this if you run `mysqld_safe` as `root`, include the `--user` option as shown. Otherwise, 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` database, 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.

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 5.6, “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 5.6\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:

  ```
  shell> tail host_name.err
  shell> 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 will 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 into 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`:

  ```
  shell> ./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:

```
shell> mysqladmin variables
```

Or:

```
shell> 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 into 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:

  ```
  shell> 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:

  ```
  shell> chown -R mysql /usr/local/mysql/var
  shell> 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:

```
shell> bin/mysqladmin version
shell> 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:

```
shell> 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:

```
shell> bin/mysqladmin version
mysqladmin  Ver 14.12 Distrib 5.6.35, for pc-linux-gnu on i686
...
Server version          5.6.35
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):

```
shell> 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:

```
shell> 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:

```
shell> bin/mysqlshow
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

The list of installed databases may vary, but will always include the minimum of `mysql` and `information_schema`.

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

```
shell> bin/mysqlshow mysql
Database: mysql
+---------------------------+
|          Tables           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| innodb_index_stats        |
| innodb_table_stats        |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| proxies_priv              |
| 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:

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

At this point, your server is running and you can access it. To tighten security if you have not yet assigned passwords to the initial account or accounts, follow the instructions in Section 9.4, “Securing the Initial MySQL Accounts”.

For more information about `mysql`, `mysqladmin`, and `mysqlshow`, see `mysql` — The MySQL Command-Line Tool, `mysqladmin` — Client for Administering a MySQL Server, and `mysqlshow` — Display Database, Table, and Column Information.

9.4 Securing the Initial MySQL Accounts

The MySQL installation process involves initializing the data directory, including the `mysql` database containing the grant tables that define MySQL accounts. For details, see Chapter 9, *Postinstallation Setup and Testing*.

This section describes how to assign passwords to the initial accounts created during the MySQL installation procedure, if you have not already done so.

The `mysql.user` grant table defines the initial MySQL user accounts and their access privileges:

- Some accounts have the user name `root`. These are superuser accounts that have all privileges and can do anything. If these `root` accounts have empty passwords, anyone can connect to the MySQL server as `root` *without a password* and be granted all privileges.

  - On Windows, `root` accounts are created that permit connections from the local host only. Connections can be made by specifying the host name `localhost`, the IP address `127.0.0.1`, or the IPv6 address `::1`. If the user selects the **Enable root access from remote machines** option during installation, the Windows installer creates another `root` account that permits connections from any host.

  - On Unix, each `root` account permits connections from the local host. Connections can be made by specifying the host name `localhost`, the IP address `127.0.0.1`, the IPv6 address `::1`, or the actual host name or IP address.

  An attempt to connect to the host `127.0.0.1` normally resolves to the `localhost` account. However, this fails if the server is run with the `--skip-name-resolve` option, so the `127.0.0.1` account is useful in that case. The `::1` account is used for IPv6 connections.

- If accounts for anonymous users were created, these have an empty user name. The anonymous accounts have no password, so anyone can use them to connect to the MySQL server.

  - On Windows, there is one anonymous account that permits connections from the local host. Connections can be made by specifying a host name of `localhost`.

  - On Unix, each anonymous account permits connections from the local host. Connections can be made by specifying a host name of `localhost` for one of the accounts, or the actual host name or IP address for the other.

  - 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 display which accounts exist in the `mysql.user` table and check whether their passwords are empty, use the following statement:

```
mysql> SELECT User, Host, Password FROM mysql.user;
+------+--------------------+----------+
| User | Host               | Password |
+------+--------------------+----------+
| root | localhost          |          |
| root | myhost.example.com |          |
| root | 127.0.0.1          |          |
| root | ::1                |          |
|      | localhost          |          |
|      | myhost.example.com |          |
+------+--------------------+----------+
```

This output indicates that there are several `root` and anonymous-user accounts, none of which have passwords. The output might differ on your system, but the presence of accounts with empty passwords means that your MySQL installation is unprotected until you do something about it:

- Assign a password to each MySQL `root` account that does not have one.

- To prevent clients from connecting as anonymous users without a password, either assign a password to each anonymous account or remove the accounts.

In addition, the `mysql.db` table contains rows that permit all accounts to access the `test` database and other databases with names that start with `test_`. This is true even for accounts that otherwise have no special privileges such as the default anonymous accounts. This is convenient for testing but inadvisable on production servers. Administrators who want database access restricted only to accounts that have permissions granted explicitly for that purpose should remove these `mysql.db` table rows.

The following instructions describe how to set up passwords for the initial MySQL accounts, first for the `root` accounts, then for the anonymous accounts. The instructions also cover how to remove anonymous accounts, should you prefer not to permit anonymous access at all, and describe how to remove permissive access to test databases. Replace *`new_password`* in the examples with the password that you want to use. Replace *`host_name`* with the name of the server host. You can determine this name from the output of the preceding `SELECT` statement. For the output shown, *`host_name`* is `myhost.example.com`.

> **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 User Accounts.

You might want to defer setting the passwords until later, to avoid the need to specify them while you perform additional setup or testing. However, be sure to set them before using your installation for production purposes.

Note

On Windows, you can also perform the process described in this section during installation with MySQL Installer (see Section 5.3, “Installing MySQL on Microsoft Windows Using MySQL Installer”). 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. MySQL Workbench is available on all platforms, and also offers the ability to manage user accounts (see MySQL Workbench).

Assigning root Account Passwords

A `root` account password can be set several ways. The following discussion demonstrates three methods:

- Use the `SET PASSWORD` statement
- Use the `UPDATE` statement
- Use the `mysqladmin` command-line client program

To assign passwords using `SET PASSWORD`, connect to the server as `root` and issue a `SET PASSWORD` statement for each `root` account listed in the `mysql.user` table.

For Windows, do this:

```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'::1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('new_password');
```

The last statement is unnecessary if the `mysql.user` table has no `root` account with a host value of `%`.

For Unix, do this:

```
shell> mysql -u root
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'::1' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'host_name' = PASSWORD('new_password');
```

You can also use a single statement that assigns a password to all `root` accounts by using `UPDATE` to modify the `mysql.user` table directly. This method works on any platform:

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('new_password')
    ->     WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the password change remains unnoticed by the server until you restart it.

To assign passwords to the `root` accounts using `mysqladmin`, execute the following commands:

```
shell> mysqladmin -u root password "new_password"
shell> mysqladmin -u root -h host_name password "new_password"
```

Those commands apply both to Windows and to Unix. The double quotation marks around the password are not always necessary, but you should use them if the password contains spaces or other characters that are special to your command interpreter.

The `mysqladmin` method of setting the `root` account passwords does not work for the `'root'@'127.0.0.1'` or `'root'@'::1'` account. Use the `SET PASSWORD` method shown earlier.

After the `root` passwords have been set, you must supply the appropriate password whenever you connect as `root` to the server. For example, to shut down the server with `mysqladmin`, use this command:

```
shell> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

The `mysql` commands in the following instructions include a `-p` option based on the assumption that you have assigned the `root` account passwords using the preceding instructions and must specify that password when connecting to the server.

Assigning Anonymous Account Passwords

To assign passwords to the anonymous accounts, connect to the server as `root`, then use either `SET PASSWORD` or `UPDATE`.

To use `SET PASSWORD` on Windows, do this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('new_password');
```

To use `SET PASSWORD` on Unix, do this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR ''@'host_name' = PASSWORD('new_password');
```

To set the anonymous-user account passwords with a single `UPDATE` statement, do this (on any platform):

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> UPDATE mysql.user SET Password = PASSWORD('new_password')
    ->     WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the password change remains unnoticed by the server until you restart it.

Removing Anonymous Accounts

If you prefer to remove any anonymous accounts rather than assigning them passwords, do so as follows on Windows:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER ''@'localhost';
```

On Unix, remove the anonymous accounts like this:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DROP USER ''@'localhost';
mysql> DROP USER ''@'host_name';
```

Securing Test Databases

By default, the `mysql.db` table contains rows that permit access by any user to the `test` database and other databases with names that start with `test_`. (These rows have an empty `User` column value, which for access-checking purposes matches any user name.) This means that such databases can be used even by accounts that otherwise possess no privileges. If you want to remove any-user access to test databases, do so as follows:

```
shell> mysql -u root -p
Enter password: (enter root password here)
mysql> DELETE FROM mysql.db WHERE Db LIKE 'test%';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH` statement causes the server to reread the grant tables. Without it, the privilege change remains unnoticed by the server until you restart it.

With the preceding change, only users who have global database privileges or privileges granted explicitly for the `test` database can use it. However, if you prefer that the database not exist at all, drop it:

```
mysql> DROP DATABASE test;
```

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 5.5.7, “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 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 OS X, 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 a 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/OpenSolaris, use the service management framework (SMF) system to initiate and control MySQL startup. For more information, see Section 8.2, "Installing MySQL on OpenSolaris Using IPS".

The `mysqld_safe` and `mysql.server` scripts, Solaris/OpenSolaris SMF, and the OS X Startup Item (or MySQL Preference Pane) can be used to start the server manually, or automatically at system startup time. `mysql.server` and the Startup Item also can be used to stop the server.

The following table shows which option groups the server and startup scripts read from option files.

Table 9.1 MySQL Startup Scripts and Supported Server Option Groups

Script	Option Groups
`mysqld`	`[mysqld]`, `[server]`, `[mysqld-`*`major_version`*`]`
`mysqld_safe`	`[mysqld]`, `[server]`, `[mysqld_safe]`
`mysql.server`	`[mysqld]`, `[mysql.server]`, `[server]`

`[mysqld-`*`major_version`*`]` means that groups with names like `[mysqld-5.5]` and `[mysqld-5.6]` are read by servers having versions 5.5.x, 5.6.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. However, 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 or Downgrading MySQL

Table of Contents

This section describes the steps to upgrade or downgrade 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.

Downgrading is less common. Typically, you undo an upgrade because of some compatibility or performance issue that occurs on a production system, and was not uncovered during initial upgrade verification on the test systems. As with the upgrade procedure, perform and verify the downgrade procedure on some test systems first, before using it on a production system.

10.1 Upgrading MySQL

This section describes how to upgrade to a new MySQL version.

- Supported Upgrade Methods
- Supported Upgrade Paths
- Before You Begin
- Performing an In-Place Upgrade
- Performing a Logical Upgrade
- Upgrade Troubleshooting

> **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).

Supported Upgrade Methods

Supported downgrade methods include:

- *In-Place Upgrade*: Involves shutting down the old MySQL version, replacing the old MySQL binaries or packages with the new ones, restarting MySQL on the existing data directory, and running `mysql_upgrade`.
- *Logical Upgrade*: Involves exporting existing data from the old MySQL version using `mysqldump`, installing the new MySQL version, loading the dump file into the new MySQL version, and running `mysql_upgrade`.

> **Note**
>
> MySQL recommends a `mysqldump` upgrade when upgrading from a previous release. For example, use this method when upgrading from 5.5 to 5.6.

For in-place and logical upgrade procedures, see Performing an In-Place Upgrade, and Performing a Logical Upgrade.

If you run MySQL Server on Windows, see Section 5.8, “Upgrading MySQL on Windows”.

If your current MySQL installation was installed on an Enterprise Linux platform or Fedora using the MySQL Yum Repository, see Section 10.1.2, “Upgrading MySQL with the MySQL Yum Repository”.

If your current MySQL installation was installed on Ubuntu using the MySQL APT repository, see Section 10.1.3, “Upgrading MySQL with the MySQL APT Repository”.

Supported Upgrade Paths

Unless otherwise documented, the following upgrade paths are supported:

- Upgrading from a release series version to a newer release series version is supported. For example, upgrading from 5.6.26 to 5.6.27 is supported. Skipping release series versions is also supported. For example, upgrading from 5.6.25 to 5.6.27 is supported.
- Upgrading one release level is supported. For example, upgrading from 5.5 to 5.6 is supported. Upgrading to the latest release series version is recommended before upgrading to the next release level. For example, upgrade to the latest 5.5 release before upgrading to 5.6.
- Upgrading more than one release level is supported, but only if you upgrade one release level at a time. For example, upgrade from 5.1 to 5.5, and then to 5.6. Follow the upgrade instructions for each release, in succession.
- Direct upgrades that skip a release level (for example, upgrading directly from MySQL 5.1 to 5.6) are not recommended or supported.

The following conditions apply to all upgrade paths:

- Upgrades between General Availability (GA) status releases are supported.
- Upgrades between milestone releases (or from a milestone release to a GA release) are not supported. For example, upgrading from 5.6.9 to 5.6.10 is not supported, as 5.6.9 is not a GA status release.
- For upgrades between versions of a MySQL release series that has reached GA status, you can move the MySQL format files and data files between different versions on systems with the same architecture. This is not necessarily true for upgrades between milestone releases. Use of milestone releases is at your own risk.

Before You Begin

Before upgrading, review the following information and perform the recommended steps:

- Before upgrading, protect your data by creating a backup of your current databases and log files. The backup should include the `mysql` system database, which contains the MySQL system tables. See Database Backup Methods.
- Review the Release Notes which provide information about features that are new in the MySQL 5.6 or differ from those found in earlier MySQL releases. Some of these changes may result in incompatibilities.
- Review Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.6”. This section describes changes that may require action before or after upgrading.
- Check Section 10.3, “Checking Whether Tables or Indexes Must Be Rebuilt”, to see whether changes to table formats or to character sets or collations were made between your current version of MySQL and the version to which you are upgrading. If such changes have resulted in an incompatibility between MySQL versions, you will need to upgrade the affected tables using the instructions in Section 10.4, “Rebuilding or Repairing Tables or Indexes”.
- If you use replication, review Upgrading a Replication Setup.
- 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.
- If your MySQL installation contains a large amount of data that might take a long time to convert after an in-place upgrade, you might find it useful to create a “dummy” database instance for assessing what conversions might be needed and the work involved to perform them. Make a copy of your MySQL instance that contains a full copy of the `mysql` database, plus all other databases without data. Run your upgrade procedure on this dummy instance to see what actions might be needed so that you can better evaluate the work involved when performing actual data conversion on your original database instance.
- Rebuilding and reinstalling MySQL language interaces is recommended whenever you install or upgrade to a new release of MySQL. This applies to MySQL interfaces such as PHP `mysql` extensions, the Perl `DBD::mysql` module, and the Python `MySQLdb` module.

Performing an In-Place Upgrade

This section describes how to perform an in-place upgrade. Review Before you Begin before proceeding.

> **Note**
>
> If you upgrade 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.

To perform an in-place upgrade:

1. Review the changes described in Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.6” for steps to be performed before upgrading.
2. If you use `InnoDB`, configure MySQL to perform a slow shutdown by setting `innodb_fast_shutdown` to `0`. For example:

```
mysql -u root -p --execute="SET GLOBAL innodb_fast_shutdown=0"
```

With a slow shutdown, `InnoDB` performs a full purge and change buffer merge before shutting down, which ensures that data files are fully prepared in case of file format differences between releases.

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

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

4. Upgrade the MySQL binaries or packages in place (replace the old binaries or packages with the new ones).

5. Start the MySQL 5.6 server, using the existing data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

6. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

`mysql_upgrade` examines all tables in all databases for incompatibilities with the current version of MySQL. `mysql_upgrade` also upgrades the `mysql` system database so that you can take advantage of new privileges or capabilities.

> **Note**
>
> `mysql_upgrade` should not be used when the server is running with `--gtid-mode=ON`. See GTID mode and mysql_upgrade for more information.
>
> `mysql_upgrade` does not upgrade the contents of the help tables. For upgrade instructions, see Server-Side Help.

7. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

Performing a Logical Upgrade

This section describes how to perform a logical upgrade. Review Before you Begin before proceeding.

To perform a logical upgrade:

1. Review the changes described in Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.6” for steps to be performed before upgrading.

2. Export your existing data from the previous MySQL version:

```
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.

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

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

4. Install MySQL 5.6. For installation instructions, see Chapter 1, *Installing and Upgrading MySQL*.

5. Initialize a new data directory, as described at Section 9.1, “Initializing the Data Directory”. For example:

```
scripts/mysql_install_db --user=mysql --datadir=/path/to/5.6-datadir
```

6. Start the MySQL 5.6 server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/5.6-datadir
```

7. Load the previously created dump file into the new MySQL server. For example:

```
mysql -u root -p --force < data-for-upgrade.sql
```

8. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

`mysql_upgrade` examines all tables in all databases for incompatibilities with the current version of MySQL. `mysql_upgrade` also upgrades the `mysql` system database so that you can take advantage of new privileges or capabilities.

Note

`mysql_upgrade` should not be used when the server is running with `--gtid-mode=ON`. See GTID mode and mysql_upgrade for more information.

`mysql_upgrade` does not upgrade the contents of the help tables. For upgrade instructions, see Server-Side Help.

9. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/5.6-datadir
```

Upgrade Troubleshooting

- If problems occur, such as that the new `mysqld` server does not start or that you cannot connect without a password, 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.15` to `libmysqlclient.so.16`).

- If you have created a user-defined function (UDF) with a given name and upgrade MySQL to a version that implements a new built-in function with the same name, the UDF becomes inaccessible. To correct this, use `DROP FUNCTION` to drop the UDF, and then use `CREATE FUNCTION` to re-create the UDF 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.

10.1.1 Changes Affecting Upgrades to MySQL 5.6

Before upgrading to MySQL 5.6, review the changes described in this section to identify upgrade issues that apply to your current MySQL installation and applications.

> **Note**
>
> In addition to the changes outlined in this section, review the Release Notes and other important information outlined in Before You Begin.

Changes marked as either **Known issue** or **Incompatible change** are incompatibilities with earlier versions of MySQL, and may require your attention *before you upgrade*. Our aim is to avoid these changes, but occasionally they are necessary to correct problems that would be worse than an incompatibility between releases. If any upgrade issue applicable to your installation involves an incompatibility that requires special handling, follow the instructions given in the incompatibility description. Sometimes this involves dumping and reloading tables, or use of a statement such as `CHECK TABLE` or `REPAIR TABLE`.

For dump and reload instructions, see Section 10.4, “Rebuilding or Repairing Tables or Indexes”. Any procedure that involves `REPAIR TABLE` with the `USE_FRM` option *must* be done before upgrading. Use of this statement with a version of MySQL different from the one used to create the table (that is, using it after upgrading) may damage the table. See REPAIR TABLE Syntax.

> **Note**
>
> Beginning with MySQL 5.6.6, several MySQL Server parameters have defaults that differ from previous releases. See the notes regarding these changes under Configuration Changes, particularly regarding overriding them to preserve backward compatibility if that is a concern.

- Configuration Changes
- Server Changes
- SQL Changes

Configuration Changes

- Beginning with MySQL 5.6.6, several MySQL Server parameters have defaults that differ from previous releases. The motivation for these changes is to provide better out-of-box performance and to reduce the need for the database administrator to change settings manually. These changes are subject to possible revision in future releases as we gain feedback.

  In some cases, a parameter has a different static default value. In other cases, the server autosizes a parameter at startup using a formula based on other related parameters or server host configuration,

rather than using a static value. For example, the setting for `back_log` now is its previous default of 50, adjusted up by an amount proportional to the value of `max_connections`. The idea behind autosizing is that when the server has information available to make a decision about a parameter setting likely to be better than a fixed default, it will.

The following table summarizes changes to defaults. Any of these can be overridden by specifying an explicit value at server startup.

Parameter	Old Default	New Default
`back_log`	50	Autosized using `max_connections`
`binlog_checksum`	`NONE`	`CRC32`
`--binlog-row-event-max-size`	1024	8192
`flush_time`	1800 (on Windows)	0
`innodb_autoextend_increment`	8	64
`innodb_buffer_pool_instances`	1	8 (platform dependent)
`innodb_checksum_algorithm`	`INNODB`	`CRC32 (changed back to INNODB in MySQL 5.6.7)`
`innodb_concurrency_tickets`	500	5000
`innodb_file_per_table`	`0`	`1`
`innodb_old_blocks_time`	0	1000
`innodb_open_files`	300	Autosized using `innodb_file_per_table`, `table_open_cache`
`innodb_stats_on_metadata`	`ON`	`OFF`
`join_buffer_size`	128KB	256KB
`max_allowed_packet`	1MB	4MB
`max_connect_errors`	10	100
`sync_master_info`	0	10000
`sync_relay_log`	0	10000
`sync_relay_log_info`	0	10000

With regard to compatibility with previous releases, the most important changes are:

- `innodb_file_per_table` is enabled (previously disabled).
- `innodb_checksum_algorithm` is `CRC32` (previously `INNODB` and changed back to `INNODB` in MySQL 5.6.7).
- `binlog_checksum` is `CRC32` (previously `NONE`).

Therefore, if you are upgrading an existing MySQL installation, have not already changed the values of these parameters from their previous defaults, and backward compatibility is a concern, you may want to explicitly set these parameters to their previous defaults. For example, put these lines in the server option file:

```
[mysqld]
innodb_file_per_table=0
innodb_checksum_algorithm=INNODB
```

```
binlog_checksum=NONE
```

Those settings preserve compatibility as follows:

- With the new default of `innodb_file_per_table` enabled, `ALTER TABLE` operations following an upgrade will move `InnoDB` tables that are in the system tablespace to individual `.ibd` files. Using `innodb_file_per_table=0` will prevent this from happening.
- Setting `innodb_checksum_algorithm=INNODB` permits binary downgrades after upgrading to this release. With a setting of `CRC32`, InnoDB would use checksumming that older MySQL versions cannot use.
- With `binlog_checksum=NONE`, the server can be used as a replication master without causing failure of older slaves that do not understand binary log checksums.

- As of MySQL 5.6.5, pre-4.1 passwords and the `mysql_old_password` authentication plugin are deprecated. Passwords stored in the older hash format used before MySQL 4.1 are less secure than passwords that use the native password hashing method and should be avoided. To prevent connections using accounts that have pre-4.1 password hashes, the `secure_auth` system variable is now enabled by default. (To permit connections for accounts that have such password hashes, start the server with `--secure_auth=0`.)

  DBAs are advised to convert accounts that use the `mysql_old_password` authentication plugin to use `mysql_native_password` instead. For account upgrade instructions, see Migrating Away from Pre-4.1 Password Hashing and the mysql_old_password Plugin.

  Known issue: In some early development versions of MySQL 5.6 (5.6.6 to 5.6.10), the server could create accounts with a mismatched password hash and authentication plugin. For example, if the default authentication plugin is `mysql_native_password`, this sequence of statements results in an account with a plugin of `mysql_native_password` but a pre-4.1 password hash (the format used by `mysql_old_password`):

  ```
  SET old_passwords = 1;
  CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
  ```

  The mismatch produces symptoms such as being unable to connect to the MySQL server and being unable to use `SET PASSWORD` with `OLD_PASSWORD()` or with `old_passwords=1`.

  As of MySQL 5.6.11, this mismatch no longer occurs. Instead, the server produces an error:

  ```
  mysql> SET old_passwords = 1;
  mysql> CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
  ERROR 1827 (HY000): The password hash doesn't have the expected
  format. Check if the correct password algorithm is being used with
  the PASSWORD() function.
  ```

  To deal with an account affected by a mismatch, the DBA can modify either the `plugin` or `Password` column in the account's `mysql.user` table row to be consistent with the other column:

  - Set `old_passwords` to 0, then assign a new password to the account using `SET PASSWORD` and `PASSWORD()`. This sets the `Password` column to have a 4.1 password hash, consistent with the `mysql_native_password` plugin. This is the preferred method of fixing the account.
  - Alternatively, the DBA can change the plugin to `mysql_old_password` to make the plugin match the password hash format, then flush the privileges. This is not recommended because the

`mysql_old_password` plugin and pre-4.1 password hashing are deprecated and support for them will be removed in a future version of MySQL.

Server Changes

- **Incompatible change**: It is possible for a column `DEFAULT` value to be valid for the `sql_mode` value at table-creation time but invalid for the `sql_mode` value when rows are inserted or updated. Example:

```
SET sql_mode = '';
CREATE TABLE t (d DATE DEFAULT 0);
SET sql_mode = 'NO_ZERO_DATE,STRICT_ALL_TABLES';
INSERT INTO t (d) VALUES(DEFAULT);
```

  In this case, 0 should be accepted for the `CREATE TABLE` but rejected for the `INSERT`. However, the server did not evaluate `DEFAULT` values used for inserts or updates against the current `sql_mode`. In the example, the `INSERT` succeeds and inserts `'0000-00-00'` into the `DATE` column.

  As of MySQL 5.6.13, the server applies the proper `sql_mode` checks to generate a warning or error at insert or update time.

  A resulting incompatibility for replication if you use statement-based logging (`binlog_format=STATEMENT`) is that if a slave is upgraded, a nonupgraded master will execute the preceding example without error, whereas the `INSERT` will fail on the slave and replication will stop.

  To deal with this, stop all new statements on the master and wait until the slaves catch up. Then upgrade the slaves followed by the master. Alternatively, if you cannot stop new statements, temporarily change to row-based logging on the master (`binlog_format=ROW`) and wait until all slaves have processed all binary logs produced up to the point of this change. Then upgrade the slaves followed by the master and change the master back to statement-based logging.

- **Incompatible change**: MySQL 5.6.11 and later supports `CREATE TABLE ... [SUB]PARTITION BY ALGORITHM=`*n*` [LINEAR] KEY (...)`, which can be used to create a table whose `KEY` partitioning is compatible with a MySQL 5.1 server (*n*=1). (Bug #14521864, Bug #66462) This syntax is not accepted by MySQL 5.6.10 and earlier, although it is supported in MySQL 5.5 beginning with MySQL 5.5.31. `mysqldump` in MySQL 5.5.31 and later MySQL 5.5 releases includes the `ALGORITHM` option when dumping tables using this option, but surrounds it with conditional comments, like this:

```
CREATE TABLE t1 (a INT)
/*!50100 PARTITION BY KEY */ /*!50531 ALGORITHM = 1 */ /*!50100 ()
     PARTITIONS 3 */
```

  When importing a dump containing such `CREATE TABLE` statements into a MySQL 5.6.10 or earlier MySQL 5.6 server, the versioned comment is not ignored, which causes a syntax error. Therefore, prior to importing such a dump file, you must either change the comments so that the MySQL 5.6 server ignores them (by removing the string `!50531` or replacing it with `!50611`, wherever it occurs), or remove them.

  This is not an issue with dump files made using MySQL 5.6.11 or later, where the `ALGORITHM` option is written using `/*!50611 ... */`.

- **Incompatible change**: For `TIME`, `DATETIME`, and `TIMESTAMP` columns, the storage required for tables created before MySQL 5.6.4 differs from storage required for tables created in 5.6.4 and later. This is due to a change in 5.6.4 that permits these temporal types to have a fractional part. This change can affect the output of statements that depend on the row format, such as `CHECKSUM TABLE`. After upgrading from MySQL 5.5 to MySQL 5.6.4 or later, it is recommended that you also upgrade from MySQL 5.5 to MySQL 5.6 `TIME`, `DATETIME`, and `TIMESTAMP` types. `ALTER TABLE` currently allows

the creation of tables containing temporal columns in both MySQL 5.5 and MySQL 5.6.4 (or later) binary format but this makes it more difficult to recreate tables in cases where `.frm` files are not available. Additionally, as of MySQL 5.6.4, the aforementioned temporal types are more space efficient. For more information about changes to temporal types in MySQL 5.6.4, see Storage Requirements for Date and Time Types.

As of MySQL 5.6.16, `ALTER TABLE` upgrades old temporal columns to 5.6 format for `ADD COLUMN`, `CHANGE COLUMN`, `MODIFY COLUMN`, `ADD INDEX`, and `FORCE` operations. Hence, the following statement upgrades a table containing columns in the old format:

```
ALTER TABLE tbl_name FORCE;
```

This conversion cannot be done using the `INPLACE` algorithm because the table must be rebuilt, so specifying `ALGORITHM=INPLACE` in these cases results in an error. Specify `ALGORITHM=COPY` if necessary.

When `ALTER TABLE` does produce a temporal-format conversion, it generates a message that can be displayed with `SHOW WARNINGS`: `TIME/TIMESTAMP/DATETIME columns of old format have been upgraded to the new format`.

When upgrading to MySQL 5.6.4 or later, be aware that `CHECK TABLE ... FOR UPGRADE` does not report temporal columns that use the pre-MySQL 5.6.4 format (Bug #73008, Bug #18985579). In MySQL 5.6.24, two new system variables, `avoid_temporal_upgrade` and `show_old_temporals`, were added to provide control over temporal column upgrades (Bug #72997, Bug #18985760).

- Due to the temporal type changes described in the previous incompatible change item above, importing pre-MySQL 5.6.4 tables (using `ALTER TABLE ... IMPORT TABLESPACE`) that contain `DATETIME` and `TIMESTAMP` types into MySQL 5.6.4 (or later) fails. Importing a MySQL 5.5 table with these temporal types into MySQL 5.6.4 (or later) is the mostly likely scenario for this problem to occur.

  The following procedures describe workarounds that use the original pre-MySQL 5.6.4 `.frm` file to recreate a table with a row structure that is compatible with 5.6.4 (or later). The procedures involve changing the original pre-MySQL 5.6.4 `.frm` file to use the `Memory` storage engine instead of `InnoDB`, copying the `.frm` file to the data directory of the destination instance, and using `ALTER TABLE` to change the table's storage engine type back to `InnoDB`. Use the first procedure if your tables do not have foreign keys. Use the second procedure, which has additional steps, if your table includes foreign keys.

  If the table does not have foreign keys:

  1. Copy the table's original `.frm` file to the data directory on the server where you want to import the tablespace.

  2. Modify the table's `.frm` file to use the `Memory` storage engine instead of the `InnoDB` storage engine. This modification requires changing 7 bytes in the `.frm` file that define the table's storage engine type. Using a hexidecimal editing tool:

     - Change the byte at offset position 0003, which is the `legacy_db_type`, from "`0c`" (for `InnoDB`) to "`06`" (for `Memory`), as shown below:

       ```
       00000000  fe 01 09 06 03 00 00 10  01 00 00 30 00 00 10 00
       ```

     - The remaining 6 bytes do not have a fixed offset. Search the `.frm` file for "`InnoDB`" to locate the line with the other 6 bytes. The line appears as shown below:

```
00001010  ff 00 00 00 00 00 00 06  00 49 6e 6e 6f 44 42 00  |.........InnoDB.|
```

- Modify the bytes so that the line appears as follows:

```
00001010  ff 00 00 00 00 00 00 06 00 4d 45 4d 4f 52 59 00
```

3. Run `ALTER TABLE ... ENGINE=INNODB` to add the table definition to the `InnoDB` data dictionary. This creates the `InnoDB` table with the temporal data types in the new format. For the `ALTER TABLE` operation to complete successfully, the `.frm` file must correspond to the tablespace.
4. Import the table using `ALTER TABLE ... IMPORT TABLESPACE`.

If table has foreign keys:

1. Recreate the tables with foreign keys using table definitions from `SHOW CREATE TABLE` output. The incorrect temporal column formats do not matter at this point.
2. Dump all foreign key definitions to a text file by selecting the foreign key information from `INFORMATION_SCHEMA.TABLE_CONSTRAINTS` and `INFORMATION_SCHEMA.KEY_COLUMN_USAGE`.
3. Drop all tables and complete the table import process described in steps 1 to 4 in the procedure described above for tables without foreign keys.
4. After the import operation is complete, add the foreign keys from foreign key definitions that you saved to a text file.

- **Incompatible change**: As of MySQL 5.6, the full-text stopword file is loaded and searched using `latin1` if `character_set_server` is `ucs2`, `utf16`, `utf16le`, or `utf32`. If any table was created with `FULLTEXT` indexes while the server character set was `ucs2`, `utf16`, `utf16le`, or `utf32`, repair it using this statement:

```
REPAIR TABLE tbl_name QUICK;
```

- **Incompatible change**: In MySQL 5.6.20, the patch for Bug #69477 limits the size of redo log `BLOB` writes to 10% of the redo log file size. As a result of this new limit, `innodb_log_file_size` should be set to a value greater than 10 times the largest `BLOB` data size found in the rows of your tables. No action is required if your `innodb_log_file_size` setting is already 10 times the largest `BLOB` data size or your tables contain no `BLOB` data.

  In MySQL 5.6.22, the redo log `BLOB` write limit is relaxed to 10% of the *total redo log size* (`innodb_log_file_size` * `innodb_log_files_in_group`). (Bug #19498877)

SQL Changes

- Some keywords may be reserved in MySQL 5.6 that were not reserved in MySQL 5.5. See Keywords and Reserved Words.
- The `YEAR(2)` data type has certain issues that you should consider before choosing to use it. As of MySQL 5.6.6, `YEAR(2)` is deprecated. `YEAR(2)` columns in existing tables are treated as before, but `YEAR(2)` in new or altered tables are converted to `YEAR(4)`. For more information, see YEAR(2) Limitations and Migrating to YEAR(4).
- As of MySQL 5.6.6, it is explicitly disallowed to assign the value `DEFAULT` to stored procedure or function parameters or stored program local variables (for example with a `SET var_name = DEFAULT`

statement). This was not previously supported, or documented as permitted, but is flagged as an incompatible change in case existing code inadvertently used this construct. It remains permissible to assign `DEFAULT` to system variables, as before, but assigning `DEFAULT` to parameters or local variables now results in a syntax error.

After an upgrade to MySQL 5.6.6 or later, existing stored programs that use this construct produce a syntax error when invoked. If a `mysqldump` file from 5.6.5 or earlier is loaded into 5.6.6 or later, the load operation fails and affected stored program definitions must be changed.

- In MySQL, the `TIMESTAMP` data type differs in nonstandard ways from other data types:

  - `TIMESTAMP` columns not explicitly declared with the `NULL` attribute are assigned the `NOT NULL` attribute. (Columns of other data types, if not explicitly declared as `NOT NULL`, permit `NULL` values.) Setting such a column to `NULL` sets it to the current timestamp.

  - The first `TIMESTAMP` column in a table, if not declared with the `NULL` attribute or an explicit `DEFAULT` or `ON UPDATE` clause, is automatically assigned the `DEFAULT CURRENT_TIMESTAMP` and `ON UPDATE CURRENT_TIMESTAMP` attributes.

  - `TIMESTAMP` columns following the first one, if not declared with the `NULL` attribute or an explicit `DEFAULT` clause, are automatically assigned `DEFAULT '0000-00-00 00:00:00'` (the “zero” timestamp). For inserted rows that specify no explicit value for such a column, the column is assigned `'0000-00-00 00:00:00'` and no warning occurs.

  Those nonstandard behaviors remain the default for `TIMESTAMP` but as of MySQL 5.6.6 are deprecated and this warning appears at startup:

  ```
  [Warning] TIMESTAMP with implicit DEFAULT value is deprecated.
  Please use --explicit_defaults_for_timestamp server option (see
  documentation for more details).
  ```

  As indicated by the warning, to turn off the nonstandard behaviors, enable the new `explicit_defaults_for_timestamp` system variable at server startup. With this variable enabled, the server handles `TIMESTAMP` as follows instead:

  - `TIMESTAMP` columns not explicitly declared as `NOT NULL` permit `NULL` values. Setting such a column to `NULL` sets it to `NULL`, not the current timestamp.

  - No `TIMESTAMP` column is assigned the `DEFAULT CURRENT_TIMESTAMP` or `ON UPDATE CURRENT_TIMESTAMP` attributes automatically. Those attributes must be explicitly specified.

  - `TIMESTAMP` columns declared as `NOT NULL` and without an explicit `DEFAULT` clause are treated as having no default value. For inserted rows that specify no explicit value for such a column, the result depends on the SQL mode. If strict SQL mode is enabled, an error occurs. If strict SQL mode is not enabled, the column is assigned the implicit default of `'0000-00-00 00:00:00'` and a warning occurs. This is similar to how MySQL treats other temporal types such as `DATETIME`.

  To upgrade servers used for replication, upgrade the slaves first, then the master. Replication between the master and its slaves should work provided that all use the same value of `explicit_defaults_for_timestamp`:

  1. Bring down the slaves, upgrade them, configure them with the desired value of `explicit_defaults_for_timestamp`, and bring them back up.

The slaves will recognize from the format of the binary logs received from the master that the master is older (predates the introduction of `explicit_defaults_for_timestamp`) and that operations on `TIMESTAMP` columns coming from the master use the old `TIMESTAMP` behavior.

2. Bring down the master, upgrade it, and configure it with the same `explicit_defaults_for_timestamp` value used on the slaves, and bring it back up.

10.1.2 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 off the old data files) with the MySQL Yum repository.

> **Notes**
>
> - Before performing any update to MySQL, follow carefully the instructions in Section 10.1, “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 Section 7.2, “Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository”.

Selecting a Target Series

By default, the MySQL Yum repository updates MySQL to the latest version in the release series you have chosen during installation (see Selecting a Release Series for details), which means, for example, a 5.6.x installation will NOT be updated to a 5.7.x release automatically. 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, see the general instructions given in Selecting a Release Series. For upgrading from MySQL 5.6 to 5.7, perform the *reverse* of the steps illustrated in Selecting a Release Series, disabling the subrepository for the MySQL 5.6 series and enabling that for the MySQL 5.7 series.

As a general rule, to upgrade from one release series to another, go to the next series rather than skipping a series. For example, if you are currently running MySQL 5.5 and wish to upgrade to 5.7, upgrade to MySQL 5.6 first before upgrading to 5.7.

> **Important**
>
> For important information about upgrading from MySQL 5.6 to 5.7, see Upgrading from MySQL 5.6 to 5.7.

Upgrading MySQL

Upgrade MySQL and its components by the following command, for platforms that are not dnf-enabled:

```
sudo yum update mysql-server
```

For platforms that are dnf-enabled:

```
sudo dnf upgrade mysql-server
```

Alternatively, you can update MySQL by telling Yum to update everything on your system, which might take considerably more time; for platforms that are not dnf-enabled:

```
sudo yum update
```

For platforms that are dnf-enabled:

```
sudo dnf upgrade
```

Restarting MySQL

The MySQL server always restarts after an update by Yum. Once the server restarts, run `mysql_upgrade` to check and possibly resolve any incompatibilities between the old data and the upgraded software. `mysql_upgrade` also performs other functions; see `mysql_upgrade` — Check and Upgrade MySQL Tables for details.

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, for platforms that are not dnf-enabled, update the package with the following command, replacing *`package-name`* with the name of the package:

```
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 5.6 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 will require those updated libraries on systems where the applications are deployed. And, as expected, if those libraries are not in place, the applications requiring the shared libraries will fail. So, be sure to deploy the packages for the shared libraries from MySQL on those systems. You can do this by adding 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.

10.1.3 Upgrading MySQL with the MySQL APT Repository

On Debian 7 or 8 and Ubuntu 12, 14, or 15, you can perform an in-place upgrade of MySQL and its components with the MySQL APT repository. See Upgrading MySQL with the MySQL APT Repository in A Quick Guide to Using the MySQL APT Repository.

10.2 Downgrading MySQL

This section describes how to downgrade to an older MySQL version.

- Supported Downgrade Methods
- Supported Downgrade Paths
- Before You Begin
- Performing an In-Place Downgrade
- Performing a Logical Downgrade
- Downgrade Troubleshooting

> **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).

Supported Downgrade Methods

Supported downgrade methods include:

- *In-Place Downgrade*: Involves shutting down the new MySQL version, replacing the new MySQL binaries or packages with the old ones, and restarting the old MySQL version on the new data files. In-place downgrades are supported for downgrades between GA versions within the same release series. For example, in-place downgrades are supported for downgrades from 5.6.27 to 5.6.26.
- *Logical Downgrade*: Involves using `mysqldump` to dump all tables from the new MySQL version, and then loading the dump file into the old MySQL version. Logical downgrades are supported for downgrades between GA versions within the same release series and for downgrades between release levels. For example, logical downgrades are supported for downgrades from 5.6.27 to 5.6.26 and for downgrades from 5.6 to 5.5.

Supported Downgrade Paths

Unless otherwise documented, the following downgrade paths are supported:

- Downgrading from a release series version to an older release series version is supported using all downgrade methods. For example, downgrading from 5.6.27 to 5.6.26 is supported. Skipping release series versions is also supported. For example, downgrading from 5.6.27 to 5.6.25 is supported.
- Downgrading one release level is supported using the *logical downgrade* method. For example, downgrading from 5.6 to 5.5 is supported.
- Downgrading more than one release level is supported using the *logical downgrade* method, but only if you downgrade one release level at a time. For example, you can downgrade from 5.6 to 5.5, and then to 5.1.

The following conditions apply to all downgrade paths:

- Downgrades between General Availability (GA) status releases are supported.
- Downgrades between milestone releases (or from a GA release to a milestone release) are not supported. For example, downgrading from MySQL 5.6.10 to MySQL 5.6.9 is not supported, as 5.6.9 is not a GA status release.

Before You Begin

Before downgrading, the following steps are recommended:

- Review the Release Notes for the MySQL version you are downgrading from to ensure that there are no features or fixes that you really need.
- Review Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.6”. This section describes changes that may require action before or after downgrading.

  > **Note**
  >
  > The downgrade procedures described in the following sections assume you are downgrading with data files created or modified by the newer MySQL version. However, if you did not modify your data after upgrading, downgrading using backups taken *before* upgrading to the new MySQL version is recommended. Many of the changes described in Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.6” that require action before or after downgrading are not applicable when downgrading using backups taken *before* upgrading to the new MySQL version.

- Always back up your current databases and log files before downgrading. The backup should include the `mysql` database, which contains the MySQL system tables. See Database Backup Methods.
- Use of new features, new configuration options, or new configuration option values that are not supported by a previous release may cause downgrade errors or failures. Before downgrading, it is recommended that you reverse changes resulting from the use of new features and remove configuration settings that are not supported by the release you are downgrading to.
- Check Section 10.3, “Checking Whether Tables or Indexes Must Be Rebuilt”, to see whether changes to table formats or to character sets or collations were made between your current version of MySQL and the version to which you are downgrading. If such changes have resulted in an incompatibility between MySQL versions, downgrade the affected tables using the instructions in Section 10.4, “Rebuilding or Repairing Tables or Indexes”.
- If you use XA transactions with `InnoDB`, run `XA RECOVER` before downgrading 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.

Performing an In-Place Downgrade

In-place downgrades are supported for downgrades between GA status releases within the same release series. Before proceeding, review Before You Begin.

To perform an in-place downgrade:

1. Review the changes described in Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.6” for steps to be performed before downgrading.
2. If you use `InnoDB`, configure MySQL to perform a slow shutdown by setting `innodb_fast_shutdown` to `0`. For example:

```
mysql -u root -p --execute="SET GLOBAL innodb_fast_shutdown=0"
```

With a slow shutdown, `InnoDB` performs a full purge and change buffer merge before shutting down, which ensures that data files are fully prepared in case of file format differences between releases.

3. Shut down the newer MySQL server. For example:

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

4. After the slow shutdown, remove the `InnoDB` redo log files (the `ib_logfile*` files) from the `data` directory to avoid downgrade issues related to redo log file format changes that may have occurred between releases.

```
rm ib_logfile*
```

5. Downgrade the MySQL binaries or packages in-place by replacing the newer binaries or packages with the older ones.

6. Start the older (downgraded) MySQL server, using the existing data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

7. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

8. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/existing-datadir
```

Performing a Logical Downgrade

Logical downgrades are supported for downgrades between releases within the same release series and for downgrades to the previous release level. Only downgrades between General Availability (GA) status releases are supported. Before proceeding, review Before You Begin.

To perform a logical downgrade:

1. Review the changes described in Section 10.2.1, “Changes Affecting Downgrades from MySQL 5.6” for steps to be performed before downgrading.

2. Dump all databases. For example:

```
mysqldump -u root -p
  --add-drop-table --routines --events
  --all-databases --force > data-for-downgrade.sql
```

3. Shut down the newer MySQL server. For example:

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

4. Initialize an older MySQL instance, with a new data directory. For example:

```
scripts/mysql_install_db --user=mysql
```

5. Start the older MySQL server, using the new data directory. For example:

```
mysqld_safe --user=mysql --datadir=/path/to/new-datadir
```

6. Load the dump file into the older MySQL server. For example:

```
mysql -u root -p --force < data-for-upgrade.sql
```

7. Run `mysql_upgrade`. For example:

```
mysql_upgrade -u root -p
```

8. Shut down and restart the MySQL server to ensure that any changes made to the system tables take effect. For example:

```
mysqladmin -u root -p shutdown
mysqld_safe --user=mysql --datadir=/path/to/new-datadir
```

Downgrade Troubleshooting

If you downgrade from one release series to another, there may be incompatibilities in table storage formats. In this case, use `mysqldump` to dump your tables before downgrading. After downgrading, reload the dump file using `mysql` or `mysqlimport` to re-create your tables. For examples, see Section 10.5, “Copying MySQL Databases to Another Machine”.

A typical symptom of a downward-incompatible table format change when you downgrade is that you cannot open tables. In that case, use the following procedure:

1. Stop the older MySQL server that you are downgrading to.
2. Restart the newer MySQL server you are downgrading from.
3. Dump any tables that were inaccessible to the older server by using `mysqldump` to create a dump file.
4. Stop the newer MySQL server and restart the older one.
5. Reload the dump file into the older server. Your tables should be accessible.

10.2.1 Changes Affecting Downgrades from MySQL 5.6

Before downgrading from MySQL 5.6, review the changes described in this section. Some changes may require action before or after downgrading.

System Tables

- The `mysql.user` table in MySQL 5.6 has a `password_expired` column. The `mysql.user` table in MySQL 5.5 does not. This means that an account with an expired password in MySQL 5.6 will work normally in MySQL 5.5.
- The `mysql.host` table was removed in MySQL 5.6.7. When downgrading to a previous release, startup on the downgraded server fails with an error if the `mysql.host` table is not present. You can recreate the table manually or restore it from a backup taken prior to upgrading to MySQL 5.6.7 or higher.

Data Types

- For `TIME`, `DATETIME`, and `TIMESTAMP` columns, the storage required for tables created before MySQL 5.6.4 differs from storage required for tables created in 5.6.4 and later. This is due to a change in 5.6.4 that permits these temporal types to have a fractional part. To downgrade to a version older than 5.6.4, dump affected tables with `mysqldump` before downgrading, and reload the tables after downgrading.

  The following query identifies tables and columns that may be affected by this problem. Some of them are system tables in the `mysql` database (such as `columns_priv` and `proxies_priv`). This means that `mysql` is one of the databases you must dump and reload, or server startup may fail after downgrading.

```
SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE DATA_TYPE IN ('TIME','DATETIME','TIMESTAMP')
ORDER BY TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME;
```

InnoDB

- `InnoDB` search indexes (with a type of `FULLTEXT`), introduced in MySQL 5.6.4, are not compatible with earlier versions of MySQL, including earlier releases in the 5.6 series. Drop such indexes before performing a downgrade.
- `InnoDB` small page sizes specified by the `innodb_page_size` configuration option, introduced in MySQL 5.6.4, are not compatible with earlier versions of MySQL, including earlier releases in the 5.6 series. Dump all `InnoDB` tables in instances that use a smaller `InnoDB` page size, drop the tables, and re-create and reload them after the downgrade.

Replication

- As of MySQL 5.6, the `relay-log.info` file contains a line count and a replication delay value, so the file format differs from that in older versions. See Slave Status Logs. If you downgrade a slave server to a version older than MySQL 5.6, the older server will not read the file correctly. To address this, modify the file in a text editor to delete the initial line containing the number of lines.
- Beginning with MySQL 5.6.6, the MySQL Server employs Version 2 binary log events when writing the binary log. Binary logs written using Version 2 log events cannot by read by earlier versions of MySQL Server. To generate a binary log that is written using Version 1 log events readable by older servers, start the MySQL 5.6.6 or later server using `--log-bin-use-v1-row-events=1`, which forces the server to employ Version 1 events when writing the binary log.

10.3 Checking Whether Tables or Indexes Must Be Rebuilt

A binary upgrade or downgrade is one that installs one version of MySQL “in place” over an existing version, without dumping and reloading tables:

1. Stop the server for the existing version if it is running.
2. Install a different version of MySQL. This is an upgrade if the new version is higher than the original version, a downgrade if the version is lower.
3. Start the server for the new version.

In many cases, the tables from the previous version of MySQL can be used without problem by the new version. However, sometimes changes occur that require tables or table indexes to be rebuilt, as described in this section. If you have tables that are affected by any of the issues described here, rebuild the tables

or indexes as necessary using the instructions given in Section 10.4, “Rebuilding or Repairing Tables or Indexes”.

Table Incompatibilities

If you have `ARCHIVE` tables created in MySQL 5.0, the recommended upgrade procedure is to dump the 5.0 `ARCHIVE` tables before upgrading and reload them after upgrading.

As of MySQL 5.6.4, MySQL permits fractional seconds for `TIME`, `DATETIME`, and `TIMESTAMP` column values. As a result, encoding and storage requirements for these temporal column types differ in tables created in MySQL 5.6.4 and later. This incompatibility is described in Section 10.1.1, “Changes Affecting Upgrades to MySQL 5.6”. When upgrading to MySQL 5.6.4 or later, be aware that `CHECK TABLE ... FOR UPGRADE` does not report temporal columns that use the pre-MySQL 5.6.4 format (Bug #73008, Bug #18985579). In MySQL 5.6.24, two new system variables, `avoid_temporal_upgrade` and `show_old_temporals`, were added to provide control over temporal column upgrades (Bug #72997, Bug #18985760).

Index Incompatibilities

In MySQL 5.6.3, the length limit for index prefix keys is increased from 767 bytes to 3072 bytes, for `InnoDB` tables using `ROW_FORMAT=DYNAMIC` or `ROW_FORMAT=COMPRESSED`. See Limits on InnoDB Tables for details. This change is also backported to MySQL 5.5.14. If you downgrade from one of these releases or higher, to an earlier release with a lower length limit, the index prefix keys could be truncated at 767 bytes or the downgrade could fail. This issue could only occur if the configuration option `innodb_large_prefix` was enabled on the server being downgraded.

If you perform a binary upgrade without dumping and reloading tables, you cannot upgrade directly from MySQL 4.1 to 5.1 or higher. This occurs due to an incompatible change in the `MyISAM` table index format in MySQL 5.0. Upgrade from MySQL 4.1 to 5.0 and repair all `MyISAM` tables. Then upgrade from MySQL 5.0 to 5.1 and check and repair your tables.

Modifications to the handling of character sets or collations might change the character sort order, which causes the ordering of entries in any index that uses an affected character set or collation to be incorrect. Such changes result in several possible problems:

- Comparison results that differ from previous results
- Inability to find some index values due to misordered index entries
- Misordered `ORDER BY` results
- Tables that `CHECK TABLE` reports as being in need of repair

The solution to these problems is to rebuild any indexes that use an affected character set or collation, either by dropping and re-creating the indexes, or by dumping and reloading the entire table. In some cases, it is possible to alter affected columns to use a different collation. For information about rebuilding indexes, see Section 10.4, “Rebuilding or Repairing Tables or Indexes”.

To check whether a table has indexes that must be rebuilt, consult the following list. It indicates which versions of MySQL introduced character set or collation changes that require indexes to be rebuilt. Each entry indicates the version in which the change occurred and the character sets or collations that the change affects. If the change is associated with a particular bug report, the bug number is given.

The list applies both for binary upgrades and downgrades. For example, Bug #27877 was fixed in MySQL 5.1.24, so it applies to upgrades from versions older than 5.1.24 to 5.1.24 or higher, and to downgrades from 5.1.24 or newer to versions older than 5.1.24.

In many cases, you can use `CHECK TABLE ... FOR UPGRADE` to identify tables for which index rebuilding is required. It will report this message:

```
Table upgrade required.
Please do "REPAIR TABLE `tbl_name`" or dump/reload to fix it!
```

In these cases, you can also use `mysqlcheck --check-upgrade` or `mysql_upgrade`, which execute `CHECK TABLE`. However, the use of `CHECK TABLE` applies only after upgrades, not downgrades. Also, `CHECK TABLE` is not applicable to all storage engines. For details about which storage engines `CHECK TABLE` supports, see CHECK TABLE Syntax.

These changes cause index rebuilding to be necessary:

- MySQL 5.1.24 (Bug #27877)

  Affects indexes that use the `utf8_general_ci` or `ucs2_general_ci` collation for columns that contain `'ß'` LATIN SMALL LETTER SHARP S (German). The bug fix corrected an error in the original collations but introduced an incompatibility such that `'ß'` compares equal to characters with which it previously compared different.

  Affected tables can be detected by `CHECK TABLE ... FOR UPGRADE` as of MySQL 5.1.30 (see Bug #40053).

  A workaround for this issue is implemented as of MySQL 5.1.62, 5.5.21, and 5.6.5. The workaround involves altering affected columns to use the `utf8_general_mysql500_ci` and `ucs2_general_mysql500_ci` collations, which preserve the original pre-5.1.24 ordering of `utf8_general_ci` and `ucs2_general_ci`.

10.4 Rebuilding or Repairing Tables or Indexes

This section describes how to rebuild a table, following changes to MySQL such as how data types or character sets are handled. For example, an error in a collation might have been corrected, requiring a table rebuild to update the indexes for character columns that use the collation. (For examples, see Section 10.3, “Checking Whether Tables or Indexes Must Be Rebuilt”.) You might also need to repair or upgrade a table, as indicated by a table check operation such as that performed by `CHECK TABLE`, `mysqlcheck`, or `mysql_upgrade`.

Methods for rebuilding a table include dumping and reloading it, or using `ALTER TABLE` or `REPAIR TABLE`. `REPAIR TABLE` only applies to `MyISAM`, `ARCHIVE`, and `CSV` tables.

> **Note**
>
> If you are rebuilding tables because a different version of MySQL will not 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.

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
```

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.

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, as described earlier. 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 Syntax.

For `MyISAM`, `ARCHIVE`, or `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
```

For incompatibilities introduced in MySQL 5.1.24 by the fix for Bug #27877 that corrected the `utf8_general_ci` and `ucs2_general_ci` collations, a workaround is implemented as of MySQL 5.1.62, 5.5.21, and 5.6.5. Upgrade to one of those versions, then convert each affected table using one of the following methods. In each case, the workaround altering affected columns to use the `utf8_general_mysql500_ci` and `ucs2_general_mysql500_ci` collations, which preserve the original pre-5.1.24 ordering of `utf8_general_ci` and `ucs2_general_ci`.

- To convert an affected table after a binary upgrade that leaves the table files in place, alter the table to use the new collation. Suppose that the table `t1` contains one or more problematic `utf8` columns. To convert the table at the table level, use a statement like this:

  ```
  ALTER TABLE t1
  CONVERT TO CHARACTER SET utf8 COLLATE utf8_general_mysql500_ci;
  ```

To apply the change on a column-specific basis, use a statement like this (be sure to repeat the column definition as originally specified except for the `COLLATE` clause):

```
ALTER TABLE t1
MODIFY c1 CHAR(N) CHARACTER SET utf8 COLLATE utf8_general_mysql500_ci;
```

- To upgrade the table using a dump and reload procedure, dump the table using `mysqldump`, modify the `CREATE TABLE` statement in the dump file to use the new collation, and reload the table.

After making the appropriate changes, `CHECK TABLE` should report no error.

10.5 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.

> **Note**
>
> You can copy the `.frm`, `.MYI`, and `.MYD` files for `MyISAM` tables between different architectures that support the same floating-point format. (MySQL takes care of any byte-swapping issues.) See The MyISAM Storage Engine.

Use `mysqldump --help` to see what options are available.

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 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
`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 `$HOME/.mysql_history`.
`LD_RUN_PATH`	Used to specify the location of `libmysqlclient.so`.
`LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN`	Enable the `mysql_clear_password` authentication plugin; see The Cleartext Client-Side Authentication Plugin.
`LIBMYSQL_PLUGIN_DIR`	Directory in which to look for client plugins.
`LIBMYSQL_PLUGINS`	Client plugins to preload.
`MYSQL_DEBUG`	Debug trace options when debugging.
`MYSQL_GROUP_SUFFIX`	Option group suffix value (like specifying `--defaults-group-suffix`).
`MYSQL_HISTFILE`	The path to the `mysql` history file. If this variable is set, its value overrides the default for `$HOME/.mysql_history`.
`MYSQL_HISTIGNORE`	Patterns specifying statements not to log to `$HOME/.mysql_history`.
`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_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 End-User Guidelines for Password Security.
`MYSQL_TCP_PORT`	The default TCP/IP port number.
`MYSQL_TEST_LOGIN_FILE`	The name of the `.mylogin.cnf` login path file.
`MYSQL_UNIX_PORT`	The default Unix socket file name; used for connections to `localhost`.
`PATH`	Used by the shell to find MySQL programs.
`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 Logging.

`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 default `UMASK` and `UMASK_DIR` values are `0660` and `0700`, 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.

Chapter 12 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.

Perl, and the `DBD::MySQL` module for `DBI` must be installed if you want to run the MySQL benchmark scripts; see The MySQL Benchmark Suite.

> **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 5.6.

12.1 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:

```
shell> 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:

```
shell> 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:

```
shell> cd DBI-VERSION
```

3. Build the distribution and compile everything:

```
shell> perl Makefile.PL
shell> make
shell> make test
shell> 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-

12.2 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";
```

12.3 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.

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`.