

MySQL and Linux/Unix

Abstract

This is the MySQL Linux extract from the MySQL 5.7 Reference Manual.

For legal information, see the [Legal Notices](#).

For help with using MySQL, please visit the [MySQL Forums](#), where you can discuss your issues with other MySQL users.

Document generated on: 2025-10-13 (revision: 83739)

Table of Contents

Preface and Legal Notices	v
1 Installing MySQL on Unix/Linux Using Generic Binaries	1
2 Installing MySQL on Linux	5
2.1 Installing MySQL on Linux Using the MySQL Yum Repository	6
2.2 Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository	10
2.3 Installing MySQL on Linux Using the MySQL APT Repository	12
2.4 Installing MySQL on Linux Using the MySQL SLES Repository	12
2.5 Installing MySQL on Linux Using RPM Packages from Oracle	13
2.6 Installing MySQL on Linux Using Debian Packages from Oracle	18
2.7 Deploying MySQL on Linux with Docker	19
2.7.1 Basic Steps for MySQL Server Deployment with Docker	19
2.7.2 More Topics on Deploying MySQL Server with Docker	24
2.7.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker	29
2.8 Installing MySQL on Linux from the Native Software Repositories	29
2.9 Installing MySQL on Linux with Juju	32
2.10 Managing MySQL Server with systemd	32
3 Installing MySQL on Solaris	39
3.1 Installing MySQL on Solaris Using a Solaris PKG	40
4 Installing MySQL on FreeBSD	43
5 Initializing the Data Directory	45

Preface and Legal Notices

This is the MySQL Linux extract from the MySQL 5.7 Reference Manual.

Licensing information—MySQL 5.7. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.7, see the [MySQL 5.7 Commercial Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 5.7, see the [MySQL 5.7 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.5. This product may include third-party software, used under license. If you are using a *Commercial* release of NDB Cluster 7.5, see the [MySQL NDB Cluster 7.5 Commercial Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of NDB Cluster 7.5, see the [MySQL NDB Cluster 7.5 Community Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.6. If you are using a *Commercial* release of MySQL NDB Cluster 7.6, see the [MySQL NDB Cluster 7.6 Commercial Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL NDB Cluster 7.6, see the [MySQL NDB Cluster 7.6 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

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

License Restrictions

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

Warranty Disclaimer

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

Restricted Rights Notice

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and

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

Hazardous Applications Notice

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

Trademark Notice

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

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

Third-Party Content, Products, and Services Disclaimer

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

Use of This Documentation

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

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Chapter 1 Installing MySQL 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 on Unix/Linux platforms. For Linux-generic binary distribution installation instructions with a focus on MySQL security features, refer to the [Secure Deployment Guide](#). For other platform-specific binary package formats, see the other platform-specific sections in this manual. For example, for Windows distributions, see [Installing MySQL on Microsoft Windows](#). See [How to Get MySQL](#) on how to obtain MySQL in different distribution formats.

MySQL compressed `tar` file binary distributions have names of the form `mysql-VERSION-OS.tar.gz`, where `VERSION` is a number (for example, `5.7.44`), and `OS` indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

Warnings

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

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

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

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

Or, on APT-based systems:

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

- *For MySQL 5.7.19 and later:* Support for Non-Uniform Memory Access (NUMA) has been added to the generic Linux build, which has a dependency now on the `libnuma` library; if the library has not been installed on your system, use your system's package manager to search for and install it (see the preceding item for some sample commands).
- **SLES 11:** As of MySQL 5.7.19, the Linux Generic tarball package format is EL6 instead of EL5. As a side effect, the MySQL client `bin/mysql` needs `libtinfo.so.5`.

A workaround is to create a symlink, such as `ln -s libncurses.so.5.6 /lib64/libtinfo.so.5` on 64-bit systems or `ln -s libncurses.so.5.6 /lib/libtinfo.so.5` on 32-bit systems.

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

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

Table 1.1 MySQL Installation Layout for Generic Unix/Linux Binary Package

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

Debug versions of the `mysqld` binary are available as `mysqld-debug`. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. See [Installing MySQL from Source](#).

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

```
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
$> cd /usr/local
$> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
$> ln -s full-path-to-mysql-VERSION-OS mysql
$> cd mysql
$> mkdir mysql-files
$> chown mysql:mysql mysql-files
$> chmod 750 mysql-files
$> bin/mysqld --initialize --user=mysql
$> bin/mysql_ssl_rsa_setup
$> bin/mysqld_safe --user=mysql &
# Next command is optional
$> cp support-files/mysql.server /etc/init.d/mysql.server
```

Note

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

The `mysql-files` directory provides a convenient location to use as the value for the `secure_file_priv` system variable, which limits import and export operations to a specific directory. See [Server System Variables](#).

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

Create a mysql User and Group

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

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

Note

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

Obtain and Unpack the Distribution

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

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

Obtain a distribution file using the instructions in [How to Get MySQL](#). For a given release, binary distributions for all platforms are built from the same MySQL source distribution.

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

```
$> tar 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:

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

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

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

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

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

Perform Postinstallation Setup

The remainder of the installation process involves setting distribution ownership and access permissions, initializing the data directory, starting the MySQL server, and setting up the configuration file. For instructions, see [Postinstallation Setup and Testing](#).

Chapter 2 Installing MySQL on Linux

Table of Contents

2.1 Installing MySQL on Linux Using the MySQL Yum Repository	6
2.2 Replacing a Third-Party Distribution of MySQL Using the MySQL Yum Repository	10
2.3 Installing MySQL on Linux Using the MySQL APT Repository	12
2.4 Installing MySQL on Linux Using the MySQL SLES Repository	12
2.5 Installing MySQL on Linux Using RPM Packages from Oracle	13
2.6 Installing MySQL on Linux Using Debian Packages from Oracle	18
2.7 Deploying MySQL on Linux with Docker	19
2.7.1 Basic Steps for MySQL Server Deployment with Docker	19
2.7.2 More Topics on Deploying MySQL Server with Docker	24
2.7.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker	29
2.8 Installing MySQL on Linux from the Native Software Repositories	29
2.9 Installing MySQL on Linux with Juju	32
2.10 Managing MySQL Server with systemd	32

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

Table 2.1 Linux Installation Methods and Information

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

As an alternative, you can use the package manager on your system to automatically download and install MySQL with packages from the native software repositories of your Linux distribution. These native packages are often several versions behind the currently available release. You also normally cannot 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 2.8, “Installing MySQL on Linux from the Native Software Repositories”](#).

Note

For many Linux installations, you may 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](#).

2.1 Installing MySQL on Linux Using the MySQL Yum Repository

The [MySQL Yum repository](#) for Oracle Linux, Red Hat Enterprise Linux and CentOS provides RPM packages for installing the MySQL server, client, MySQL Workbench, MySQL Utilities, MySQL Router, MySQL Shell, Connector/ODBC, Connector/Python and so on (not all packages are available for all the distributions; see [Installing Additional MySQL Products and Components with Yum](#) for details).

Before You Start

As a popular, open-source software, MySQL, in its original or re-packaged form, is widely installed on many systems from various sources, including different software download sites, software repositories, and so on. The following instructions assume that MySQL is not already installed on your system using a third-party-distributed RPM package; if that is not the case, follow the instructions given in [Upgrading MySQL with the MySQL Yum Repository](#) or [Section 2.2, “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 version of MySQL with the MySQL Yum repository:

Adding¹the MySQL Yum Repository

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

- Go to the Download MySQL Yum Repository page (<https://dev.mysql.com/downloads/repo/yum/>) in the MySQL Developer Zone.
- Select and download the release package for your platform.
- Install the downloaded release package with the following command, replacing *platform-and-version-specific-package-name* with the name of the downloaded RPM package:

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

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

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

For an EL7-based system:

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

For an EL8-based system:

```
$> sudo yum localinstall mysql57-community-release-el8-{version-number}.noarch.rpm
```

For Fedora:

MySQL 5.7 does not support Fedora; support was removed in MySQL 5.7.30. For details, see the [MySQL Product Support EOL Announcements](#).

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 [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:

```
$> 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 upgrades MySQL packages on your system and replaces any native third-party packages, if Yum finds replacements for them in the MySQL Yum repository; see [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:

```
$> 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:

```
$> sudo yum-config-manager --disable mysql57-community
$> sudo yum-config-manager --enable mysql56-community
```

For Fedora platforms:

```
$> sudo dnf config-manager --disable mysql57-community
$> 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 is used by Yum.

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

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

Disabling the Default MySQL Module

(EL8 systems only) EL8-based systems such as RHEL8 and Oracle Linux 8 include a MySQL module that is enabled by default. Unless this module is disabled, it masks packages provided by MySQL repositories. To disable the included module and make the MySQL repository packages visible, use the following command (for dnf-enabled systems, replace `yum` in the command with `dnf`):

```
$> sudo yum module disable mysql
```

Installing MySQL

Install MySQL by the following command:

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

This installs the package for MySQL server (`mysql-community-server`) and also packages for the components required to run the server, including packages for the client (`mysql-community-client`), the common error messages and character sets for client and server (`mysql-community-common`), and the shared client libraries (`mysql-community-libs`).

Starting the MySQL Server

Start the MySQL server with the following command:

```
$> sudo service mysqld start
Starting mysqld: [ OK ]
```

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

```
$> sudo service mysqld status
mysqld (pid 3066) is running.
```

At the initial start up of the server, the following happens, given that the data directory of the server is empty:

- The server is initialized.
- SSL certificate and key files are generated in the data directory.
- `validate_password` is installed and enabled.
- A superuser account '`root`'@'`localhost`' is created. A password for the superuser is set and stored in the error log file. To reveal it, use the following command:

```
$> sudo grep 'temporary password' /var/log/mysqld.log
```

Change the root password as soon as possible by logging in with the generated, temporary password and set a custom password for the superuser account:

```
$> mysql -uroot -p
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';
```

Note

`validate_password` is installed by default. The default password policy implemented by `validate_password` requires that passwords contain at least one uppercase letter, one lowercase letter, one digit, and one special character, and that the total password length is at least 8 characters.

For more information on the postinstallation procedures, see [Postinstallation Setup and Testing](#).

Note

Compatibility Information for EL7-based platforms: The following RPM packages from the native software repositories of the platforms are incompatible with the package from the MySQL Yum repository that installs the MySQL server. Once you have installed MySQL using the MySQL Yum repository, you cannot install these packages (and vice versa).

- `akonadi-mysql`

Installing Additional MySQL Products and Components with Yum

You can use Yum to install and manage individual components of MySQL. Some of these components are hosted in sub-repositories of the MySQL Yum repository: for example, the MySQL Connectors are to be found in the MySQL Connectors Community sub-repository, and the MySQL Workbench in MySQL Tools Community. You can use the following command to list the packages for all the MySQL components available for your platform from the MySQL Yum repository:

```
$> 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:

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

For example, to install MySQL Workbench on Fedora:

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

To install the shared client libraries:

```
$> 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 [Upgrading MySQL with the MySQL Yum Repository](#) for details.

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

For supported Yum-based platforms (see [Section 2.1, “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:

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.

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](#).

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

By design, the MySQL Yum repository replaces your native third-party MySQL with the latest GA release (from the MySQL 5.7 series currently) from the MySQL Yum repository when you perform a `yum update` command on the system, or a `yum update mysql-server`.

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:

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.

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:

```
$> yum list installed mariadb\*
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:

```
$> yum list installed Percona\*
Percona-Server-client-55.i686      5.5.39-rel136.0.el6      @percona-release-i386
Percona-Server-server-55.i686      5.5.39-rel136.0.el6      @percona-release-i386
Percona-Server-shared-55.i686      5.5.39-rel136.0.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:

```
$> 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:

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

Use this command for stopping delivery from Percona:

```
$> 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 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.

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:

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

For the Percona packages we found in Step 2 above:

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

Installing MySQL with the MySQL Yum Repository

Then, install MySQL with the MySQL Yum repository by following the instructions given in [Section 2.1](#), “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](#) [9].

2.3 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 current Debian and Ubuntu releases.

Instructions for using the MySQL APT Repository are available in [A Quick Guide to Using the MySQL APT Repository](#).

2.4 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](#).

2.5 Installing MySQL on Linux Using RPM Packages from Oracle

The recommended way to install MySQL on RPM-based Linux distributions is by using the RPM packages provided by Oracle. There are two sources for obtaining them, for the Community Edition of MySQL:

- From the MySQL software repositories:
 - The MySQL Yum repository (see [Section 2.1, “Installing MySQL on Linux Using the MySQL Yum Repository”](#) for details).
 - The MySQL SLES repository (see [Section 2.4, “Installing MySQL on Linux Using the MySQL SLES Repository”](#) for details).
- From the [Download MySQL Community Server](#) page in the [MySQL Developer Zone](#).

Note

RPM distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by Oracle in features, capabilities, and conventions (including communication setup), and that the installation instructions in this manual do not necessarily apply to them. The vendor's instructions should be consulted instead.

If you have such a third-party distribution of MySQL running on your system and now want to migrate to Oracle's distribution using the RPM packages downloaded from the MySQL Developer Zone, see [Compatibility with RPM Packages from Other Vendors](#) below. The preferred method of migration, however, is to use the [MySQL Yum repository](#) or [MySQL SLES repository](#).

RPM packages for MySQL are listed in the following tables:

Table 2.2 RPM Packages for MySQL Community Edition

Package Name	Summary
mysql-community-server	Database server and related tools
mysql-community-client	MySQL client applications and tools
mysql-community-common	Common files for server and client libraries
mysql-community-devel	Development header files and libraries for MySQL database client applications
mysql-community-libs	Shared libraries for MySQL database client applications
mysql-community-libs-compat	Shared compatibility libraries for previous MySQL installations
mysql-community-embedded	MySQL embedded library
mysql-community-embedded-devel	Development header files and libraries for MySQL as an embeddable library
mysql-community-test	Test suite for the MySQL server

Table 2.3 RPM Packages for the MySQL Enterprise Edition

Package Name	Summary
mysql-commercial-server	Database server and related tools

Package Name	Summary
<code>mysql-commercial-client</code>	MySQL client applications and tools
<code>mysql-commercial-common</code>	Common files for server and client libraries
<code>mysql-commercial-devel</code>	Development header files and libraries for MySQL database client applications
<code>mysql-commercial-libs</code>	Shared libraries for MySQL database client applications
<code>mysql-commercial-libs-compat</code>	Shared compatibility libraries for previous MySQL installations
<code>mysql-commercial-embedded</code>	MySQL embedded library
<code>mysql-commercial-embedded-devel</code>	Development header files and libraries for MySQL as an embeddable library
<code>mysql-commercial-test</code>	Test suite for the MySQL server

The full names for the RPMs have the following syntax:

```
packagename-version-distribution-arch.rpm
```

The *distribution* and *arch* values indicate the Linux distribution and the processor type for which the package was built. See the table below for lists of the distribution identifiers:

Table 2.4 MySQL Linux RPM Package Distribution Identifiers

distribution Value	Intended Use
<code>el{version}</code> where <i>{version}</i> is the major Enterprise Linux version, such as <code>el8</code>	EL6 (8.0), EL7, EL8, EL9, and EL10-based platforms (for example, the corresponding versions of Oracle Linux, Red Hat Enterprise Linux, and CentOS)
<code>sles12</code>	SUSE Linux Enterprise Server 12

To see all files in an RPM package (for example, `mysql-community-server`), use the following command:

```
$> rpm -qpl mysql-community-server-version-distribution-arch.rpm
```

The discussion in the rest of this section applies only to an installation process using the RPM packages directly downloaded from Oracle, instead of through a MySQL repository.

Dependency relationships exist among some of the packages. If you plan to install many of the packages, you may wish to download the RPM bundle `tar` file instead, which contains all the RPM packages listed above, so that you need not download them separately.

In most cases, you need to install the `mysql-community-server`, `mysql-community-client`, `mysql-community-libs`, `mysql-community-common`, and `mysql-community-libs-compat` packages to get a functional, standard MySQL installation. To perform such a standard, basic installation, go to the folder that contains all those packages (and, preferably, no other RPM packages with similar names), and issue the following command for platforms *other than* Red Hat Enterprise Linux/Oracle Linux/CentOS:

```
$> sudo yum install mysql-community-{server,client,common,libs}-*
```

Replace `yum` with `zypper` for SLES.

For Red Hat Enterprise Linux/Oracle Linux/CentOS systems:

```
$> sudo yum install mysql-community-{server,client,common,libs}-* mysql-5.*
```

While it is much preferable to use a high-level package management tool like `yum` to install the packages, users who prefer direct `rpm` commands can replace the `yum install` command with the `rpm -Uvh` command; however, using `rpm -Uvh` instead makes the installation process more prone to failure, due to potential dependency issues the installation process might run into.

To install only the client programs, you can skip `mysql-community-server` in your list of packages to install; issue the following command for platforms *other than* Red Hat Enterprise Linux/Oracle Linux/CentOS:

```
$> sudo yum install mysql-community-{client,common,libs}-*
```

Replace `yum` with `zypper` for SLES.

For Red Hat Enterprise Linux/Oracle Linux/CentOS systems:

```
$> sudo yum install mysql-community-{client,common,libs}-* mysql-5.*
```

A standard installation of MySQL using the RPM packages result in files and resources created under the system directories, shown in the following table.

Table 2.5 MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone

Files or Resources	Location
Client programs and scripts	<code>/usr/bin</code>
<code>mysqld</code> server	<code>/usr/sbin</code>
Configuration file	<code>/etc/my.cnf</code>
Data directory	<code>/var/lib/mysql</code>
Error log file	For RHEL, Oracle Linux, CentOS or Fedora platforms: <code>/var/log/mysqld.log</code> For SLES: <code>/var/log/mysql/mysqld.log</code>
Value of <code>secure_file_priv</code>	<code>/var/lib/mysql-files</code>
System V init script	For RHEL, Oracle Linux, CentOS or Fedora platforms: <code>/etc/init.d/mysqld</code> For SLES: <code>/etc/init.d/mysql</code>
Systemd service	For RHEL, Oracle Linux, CentOS or Fedora platforms: <code>mysqld</code> For SLES: <code>mysql</code>
Pid file	<code>/var/run/mysql/mysqld.pid</code>
Socket	<code>/var/lib/mysql/mysql.sock</code>
Keyring directory	<code>/var/lib/mysql-keyring</code>
Unix manual pages	<code>/usr/share/man</code>
Include (header) files	<code>/usr/include/mysql</code>
Libraries	<code>/usr/lib/mysql</code>

Files or Resources	Location
Miscellaneous support files (for example, error messages, and character set files)	<code>/usr/share/mysql</code>

The installation also creates a user named `mysql` and a group named `mysql` on the system.

Notes

- The `mysql` user is created using the `-r` and `-s /bin/false` options of the `useradd` command, so that it does not have login permissions to your server host (see [Creating the mysql User and Group](#) for details). To switch to the `mysql` user on your OS, use the `--shell=/bin/bash` option for the `su` command:

```
su - mysql --shell=/bin/bash
```

- Installation of previous versions of MySQL using older packages might have created a configuration file named `/usr/my.cnf`. It is highly recommended that you examine the contents of the file and migrate the desired settings inside to the file `/etc/my.cnf` file, then remove `/usr/my.cnf`.

MySQL is not automatically started at the end of the installation process. For Red Hat Enterprise Linux, Oracle Linux, CentOS, and Fedora systems, use the following command to start MySQL:

```
$> sudo service mysqld start
```

For SLES systems, the command is the same, but the service name is different:

```
$> sudo service mysql start
```

If the operating system is systemd enabled, standard `service` commands such as `stop`, `start`, `status` and `restart` should be used to manage the MySQL server service. The `mysqld` service is enabled by default, and it starts at system reboot. Notice that certain things might work differently on systemd platforms: for example, changing the location of the data directory might cause issues. See [Section 2.10, “Managing MySQL Server with systemd”](#) for additional information.

During an upgrade installation using RPM and DEB packages, if the MySQL server is running when the upgrade occurs then the MySQL server is stopped, the upgrade occurs, and the MySQL server is restarted. One exception: if the edition also changes during an upgrade (such as community to commercial, or vice-versa), then MySQL server is not restarted.

At the initial start up of the server, the following happens, given that the data directory of the server is empty:

- The server is initialized.
- An SSL certificate and key files are generated in the data directory.
- `validate_password` is installed and enabled.
- A superuser account `'root'@'localhost'` is created. A password for the superuser is set and stored in the error log file. To reveal it, use the following command for RHEL, Oracle Linux, CentOS, and Fedora systems:

```
$> sudo grep 'temporary password' /var/log/mysqld.log
```

Use the following command for SLES systems:

```
$> sudo grep 'temporary password' /var/log/mysql/mysqld.log
```

The next step is to log in with the generated, temporary password and set a custom password for the superuser account:

```
$> mysql -uroot -p
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';
```

Note

`validate_password` is installed by default. The default password policy implemented by `validate_password` requires that passwords contain at least one uppercase letter, one lowercase letter, one digit, and one special character, and that the total password length is at least 8 characters.

If something goes wrong during installation, you might find debug information in the error log file `/var/log/mysql.log`.

For some Linux distributions, it might be necessary to increase the limit on number of file descriptors available to `mysqld`. See [File Not Found and Similar Errors](#)

Compatibility with RPM Packages from Other Vendors. If you have installed packages for MySQL from your Linux distribution's local software repository, it is much preferable to install the new, directly-downloaded packages from Oracle using the package management system of your platform (`yum`, `dnf`, or `zypper`), as described above. The command replaces old packages with new ones to ensure compatibility of old applications with the new installation; for example, the old `mysql-libs` package is replaced with the `mysql-community-libs-compat` package, which provides a replacement-compatible client library for applications that were using your older MySQL installation. If there was an older version of `mysql-community-libs-compat` on the system, it also gets replaced.

If you have installed third-party packages for MySQL that are NOT from your Linux distribution's local software repository (for example, packages directly downloaded from a vendor other than Oracle), you should uninstall all those packages before installing the new, directly-downloaded packages from Oracle. This is because conflicts may arise between those vendor's RPM packages and Oracle's: for example, a vendor's convention about which files belong with the server and which belong with the client library may differ from that used for Oracle packages. Attempts to install an Oracle RPM may then result in messages saying that files in the RPM to be installed conflict with files from an installed package.

Installing Client Libraries from Multiple MySQL Versions. It is possible to install multiple client library versions, such as for the case that you want to maintain compatibility with older applications linked against previous libraries. To install an older client library, use the `--oldpackage` option with `rpm`. For example, to install `mysql-community-libs-5.5` on an EL6 system that has `libmysqlclient.20` from MySQL 5.7, use a command like this:

```
$> rpm --oldpackage -ivh mysql-community-libs-5.5.50-2.el6.x86_64.rpm
```

Debug Package. A special variant of MySQL Server compiled with the `debug package` has been included in the server RPM packages. It performs debugging and memory allocation checks and produces a trace file when the server is running. To use that debug version, start MySQL with `/usr/sbin/mysqld-debug`, instead of starting it as a service or with `/usr/sbin/mysqld`. See [The DBUG Package](#) for the debug options you can use.

Note

The default plugin directory for debug builds changed from `/usr/lib64/mysql/plugin` to `/usr/lib64/mysql/plugin/debug` in 5.7.21. Previously, it was necessary to change `plugin_dir` to `/usr/lib64/mysql/plugin/debug` for debug builds.

Rebuilding RPMs from source SRPMs. Source code SRPM packages for MySQL are available for download. They can be used as-is to rebuild the MySQL RPMs with the standard `rpmbuild` tool chain.

root passwords for pre-GA releases. For MySQL 5.7.4 and 5.7.5, the initial random `root` password is written to the `.mysql_secret` file in the directory named by the `HOME` environment variable. When trying to access the file, bear in mind that depending on operating system, using a command such as `sudo` may cause the value of `HOME` to refer to the home directory of the `root` system user. `.mysql_secret` is created with mode 600 to be accessible only to the system user for whom it is created. Before MySQL 5.7.4, the accounts (including `root`) 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 [Postinstallation Setup and Testing](#)."

2.6 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](#). This is the preferred method for installing MySQL on Debian-like systems, as it provides a simple and convenient way to install and update MySQL products. For details, see [Section 2.3, "Installing MySQL on Linux Using the MySQL APT Repository"](#).
- The [MySQL Developer Zone's Download Area](#). For details, see [How to Get MySQL](#). The following are some information on the Debian packages available there and the instructions for installing them:
 - Various Debian packages are provided in the MySQL Developer Zone for installing different components of MySQL on different Debian or Ubuntu platforms. The preferred method is to use the tarball bundle, which contains the packages needed for a basic setup of MySQL. The tarball bundles have names in the format of `mysql-server_MVER-DVER_CPU.deb-bundle.tar`. `MVER` is the MySQL version and `DVER` is the Linux distribution version. The `CPU` value indicates the processor type or family for which the package is built, as shown in the following table:

Table 2.6 MySQL Debian and Ubuntu Installation Packages CPU Identifiers

CPU Value	Intended Processor Type or Family
<code>i386</code>	Pentium processor or better, 32 bit
<code>amd64</code>	64-bit x86 processor

- After downloading the tarball, unpack it with the following command:

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

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

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

- Preconfigure the MySQL server package with the following command:

```
$> sudo dpkg-preconfigure mysql-community-server_*.deb
```

You are asked to provide a password for the root user for your MySQL installation. You might also be asked other questions regarding the installation.

Important

Make sure you remember the root password you set. Users who want to set a password later can leave the **password** field blank in the dialogue box

and just press **OK**; in that case, root access to the server is authenticated using the [MySQL Socket Peer-Credential Authentication Plugin](#) for connections using a Unix socket file. You can set the root password later using [mysql_secure_installation](#).

- For a basic installation of the MySQL server, install the database common files package, the client package, the client metapackage, the server package, and the server metapackage (in that order); you can do that with a single command:

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

If you are being warned of unmet dependencies by `dpkg`, you can fix them using `apt-get`:

```
sudo apt-get -f install
```

Here are where the files are installed on the system:

- All configuration files (like `my.cnf`) are under `/etc/mysql`
- All binaries, libraries, headers, etc., are under `/usr/bin` and `/usr/sbin`
- The data directory is `/var/lib/mysql`

Note

Debian distributions of MySQL are also provided by other vendors. Be aware that they may differ from those built by Oracle in features, capabilities, and conventions (including communication setup), and that the instructions in this manual do not necessarily apply to installing them. The vendor's instructions should be consulted instead.

2.7 Deploying MySQL on Linux with Docker

The Docker deployment framework supports easy installation and configuration of MySQL Server. This section explains how to use a MySQL Server Docker image.

You need to have Docker installed on your system before you can use a MySQL Server Docker image. See [Install Docker](#) for instructions.

Warning

Beware of the security concerns with running Docker containers. See [Docker security](#) for details.

The instructions for using the MySQL Docker container are divided into two sections.

2.7.1 Basic Steps for MySQL Server Deployment with Docker

Warning

The MySQL Docker images maintained by the MySQL team are built specifically for Linux platforms. Other platforms are not supported, and users using these MySQL Docker images on them are doing so at their own risk. See [the discussion here](#) for some known limitations for running these containers on non-Linux operating systems.

- [Downloading a MySQL Server Docker Image](#)

- [Starting a MySQL Server Instance](#)
- [Connecting to MySQL Server from within the Container](#)
- [Container Shell Access](#)
- [Stopping and Deleting a MySQL Container](#)
- [Upgrading a MySQL Server Container](#)
- [More Topics on Deploying MySQL Server with Docker](#)

Downloading a MySQL Server Docker Image

Important

For users of MySQL Enterprise Edition: A subscription is required to use the Docker images for MySQL Enterprise Edition. Subscriptions work by a Bring Your Own License model; see [How to Buy MySQL Products and Services](#) for details.

Downloading the server image in a separate step is not strictly necessary; however, performing this step before you create your Docker container ensures your local image is up to date. To download the MySQL Community Edition image, run this command:

```
docker pull mysql/mysql-server:tag
```

The `tag` is the label for the image version you want to pull (for example, `5.6`, `5.7`, `8.0`, or `latest`). If `:tag` is omitted, the `latest` label is used, and the image for the latest GA version of MySQL Community Server is downloaded. Refer to the list of tags for available versions on the [mysql/mysql-server page in the Docker Hub](#).

To download the MySQL Community Edition image from the Oracle Container Registry (OCR), run this command:

```
docker pull container-registry.oracle.com/mysql/mysql-server:tag
```

To download the MySQL Enterprise Edition image from the OCR, you need to first accept the license agreement on the OCR and log in to the container repository with your Docker client:

- Visit the OCR at <https://container-registry.oracle.com/> and choose **MySQL**.
- Under the list of MySQL repositories, choose `enterprise-server`.
- If you have not signed in to the OCR yet, click the **Sign in** button on the right of the page, and then enter your Oracle account credentials when prompted to.
- Follow the instructions on the right of the page to accept the license agreement.
- Log in to the OCR with your Docker client (the `docker` command) using the `docker login` command:

```
# docker login container-registry.oracle.com
Username: Oracle-Account-ID
Password: password
Login successful.
```

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

```
docker pull container-registry.oracle.com/mysql/enterprise-server:tag
```

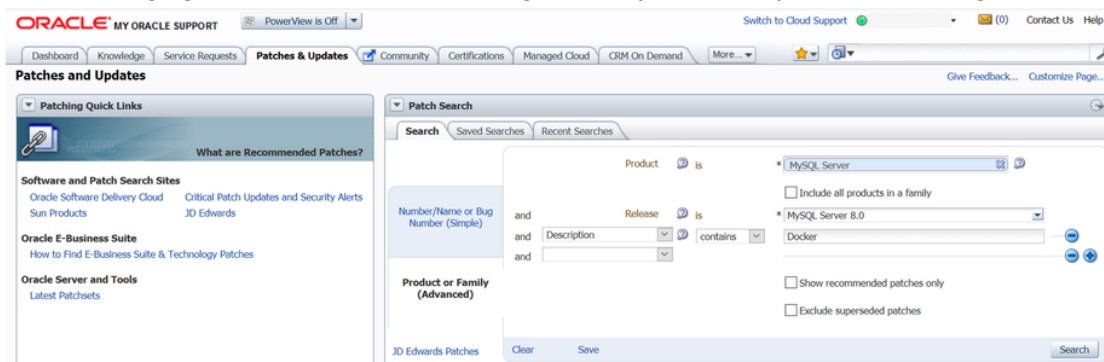
There are different choices for `tag`, corresponding to different versions of MySQL Docker images provided by the OCR:

- **8.0, 8.0.x** (x is the latest version number in the 8.0 series), **latest**: MySQL 8.0, the latest GA
- **5.7, 5.7.y** (y is the latest version number in the 5.7 series): MySQL 5.7

To download the MySQL Enterprise Edition image, visit the [My Oracle Support](#) website, sign in to your Oracle account, and perform these steps once you are on the landing page:

- Select the **Patches and Updates** tab.
- Go to the **Patch Search** region and, on the **Search** tab, switch to the **Product or Family (Advanced)** subtab.
- Enter “MySQL Server” for the **Product** field, and the desired version number in the **Release** field.
- Use the dropdowns for additional filters to select **Description—contains**, and enter “Docker” in the text field.

The following figure shows the search settings for a MySQL Enterprise Edition image:



- Click the **Search** button and, from the result list, select the version you want, and click the **Download** button.
- In the **File Download** dialogue box that appears, click and download the **.zip** file for the Docker image.

Unzip the downloaded **.zip** archive to obtain the tarball inside (**mysql-enterprise-server-version.tar**), and then load the image by running this command:

```
docker load -i mysql-enterprise-server-version.tar
```

You can list downloaded Docker images with this command:

```
$> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql/mysql-server	latest	3157d7f55f8d	4 weeks ago	241MB

Starting a MySQL Server Instance

To start a new Docker container for a MySQL Server, use the following command:

```
docker run --name=container_name -d image_name:tag
```

The image name can be obtained using the `docker images` command, as explained in [Downloading a MySQL Server Docker Image](#). The `--name` option, for supplying a custom name for your server container, is optional; if no container name is supplied, a random one is generated.

For example, to start a new Docker container for the MySQL Community Server, use this command:

```
docker run --name=mysql11 -d mysql/mysql-server:5.7
```

To start a new Docker container for the MySQL Enterprise Server with a Docker image downloaded from the OCR, use this command:

```
docker run --name=mysql1 -d container-registry.oracle.com/mysql/enterprise-server:5.7
```

To start a new Docker container for the MySQL Enterprise Server with a Docker image downloaded from My Oracle Support, use this command:

```
docker run --name=mysql1 -d mysql/enterprise-server:5.7
```

If the Docker image of the specified name and tag has not been downloaded by an earlier `docker pull` or `docker run` command, the image is now downloaded. Initialization for the container begins, and the container appears in the list of running containers when you run the `docker ps` command. For example:

```
$> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
a24888f0d6f4	mysql/mysql-server	"/entrypoint.sh my..."	14 seconds ago	Up 13 seconds (health: starting)

The container initialization might take some time. When the server is ready for use, the **STATUS** of the container in the output of the `docker ps` command changes from `(health: starting)` to `(healthy)`.

The `-d` option used in the `docker run` command above makes the container run in the background. Use this command to monitor the output from the container:

```
docker logs mysql1
```

Once initialization is finished, the command's output is going to contain the random password generated for the root user; check the password with, for example, this command:

```
$> docker logs mysql1 2>&1 | grep GENERATED
GENERATED ROOT PASSWORD: Axegh3kAJyDLaRuBemecis&ESh0s
```

Connecting to MySQL Server from within the Container

Once the server is ready, you can run the `mysql` client within the MySQL Server container you just started, and connect it to the MySQL Server. Use the `docker exec -it` command to start a `mysql` client inside the Docker container you have started, like the following:

```
docker exec -it mysql1 mysql -uroot -p
```

When asked, enter the generated root password (see the last step in [Starting a MySQL Server Instance](#) above on how to find the password). Because the `MYSQL_ONETIME_PASSWORD` option is true by default, after you have connected a `mysql` client to the server, you must reset the server root password by issuing this statement:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';
```

Substitute `password` with the password of your choice. Once the password is reset, the server is ready for use.

Container Shell Access

To have shell access to your MySQL Server container, use the `docker exec -it` command to start a bash shell inside the container:

```
$> docker exec -it mysql1 bash
bash-4.2#
```

You can then run Linux commands inside the container. For example, to view contents in the server's data directory inside the container, use this command:

```
bash-4.2# ls /var/lib/mysql
auto.cnf      ca.pem      client-key.pem  ib_logfile0  ibdata1  mysql      mysql.sock.lock  private_key.p
ca-key.pem    client-cert.pem  ib_buffer_pool  ib_logfile1  ibtmp1    mysql.sock  performance_schema  public_
```

Stopping and Deleting a MySQL Container

To stop the MySQL Server container we have created, use this command:

```
docker stop mysql1
```

`docker stop` sends a SIGTERM signal to the `mysqld` process, so that the server is shut down gracefully.

Also notice that when the main process of a container (`mysqld` in the case of a MySQL Server container) is stopped, the Docker container stops automatically.

To start the MySQL Server container again:

```
docker start mysql1
```

To stop and start again the MySQL Server container with a single command:

```
docker restart mysql1
```

To delete the MySQL container, stop it first, and then use the `docker rm` command:

```
docker stop mysql1
```

```
docker rm mysql1
```

If you want the [Docker volume for the server's data directory](#) to be deleted at the same time, add the `-v` option to the `docker rm` command.

Upgrading a MySQL Server Container

Important

- Before performing any upgrade to MySQL, follow carefully the instructions in [Upgrading MySQL](#). Among other instructions discussed there, it is especially important to back up your database before the upgrade.
- The instructions in this section require that the server's data and configuration have been persisted on the host. See [Persisting Data and Configuration Changes](#) for details.

Follow these steps to upgrade a Docker installation of MySQL 5.6 to 5.7:

- Stop the MySQL 5.6 server (container name is `mysql56` in this example):

```
docker stop mysql56
```

- Download the MySQL 5.7 Server Docker image. See instructions in [Downloading a MySQL Server Docker Image](#); make sure you use the right tag for MySQL 5.7.
- Start a new MySQL 5.7 Docker container (named `mysql57` in this example) with the old server data and configuration (with proper modifications if needed—see [Upgrading MySQL](#)) that have been persisted on the host (by [bind-mounting](#) in this example). For the MySQL Community Server, run this command:

```
docker run --name=mysql57 \
  --mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
  --mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
  -d mysql/mysql-server:5.7
```

If needed, adjust `mysql/mysql-server` to the correct image name—for example, replace it with `container-registry.oracle.com/mysql/enterprise-server` for MySQL Enterprise Edition images downloaded from the OCR, or `mysql/enterprise-server` for MySQL Enterprise Edition images downloaded from [My Oracle Support](#).

- Wait for the server to finish startup. You can check the status of the server using the `docker ps` command (see [Starting a MySQL Server Instance](#) for how to do that).
- Run the `mysql_upgrade` utility in the MySQL 5.7 Server container:

```
docker exec -it mysql57 mysql_upgrade -uroot -p
```

When prompted, enter the root password for your old MySQL 5.6 Server.

- Finish the upgrade by restarting the MySQL 5.7 Server container:

```
docker restart mysql57
```

More Topics on Deploying MySQL Server with Docker

For more topics on deploying MySQL Server with Docker like server configuration, persisting data and configuration, server error log, and container environment variables, see [Section 2.7.2, “More Topics on Deploying MySQL Server with Docker”](#).

2.7.2 More Topics on Deploying MySQL Server with Docker

Note

Most of the sample commands below have `mysql/mysql-server` as the Docker image repository when that has to be specified (like with the `docker pull` and `docker run` commands); change that if your image is from another repository—for example, replace it with `container-registry.oracle.com/mysql/enterprise-server` for MySQL Enterprise Edition images downloaded from the Oracle Container Registry (OCR), or `mysql/enterprise-server` for MySQL Enterprise Edition images downloaded from [My Oracle Support](#).

- [The Optimized MySQL Installation for Docker](#)
- [Configuring the MySQL Server](#)
- [Persisting Data and Configuration Changes](#)
- [Running Additional Initialization Scripts](#)
- [Connect to MySQL from an Application in Another Docker Container](#)
- [Server Error Log](#)
- [Known Issues](#)
- [Docker Environment Variables](#)

The Optimized MySQL Installation for Docker

Docker images for MySQL are optimized for code size, which means they only include crucial components that are expected to be relevant for the majority of users who run MySQL instances in Docker containers. A MySQL Docker installation is different from a common, non-Docker installation in the following aspects:

- Included binaries are limited to:

- `/usr/bin/my_print_defaults`
- `/usr/bin/mysql`
- `/usr/bin/mysql_config`
- `/usr/bin/mysql_install_db`
- `/usr/bin/mysql_tzinfo_to_sql`
- `/usr/bin/mysql_upgrade`
- `/usr/bin/mysqladmin`
- `/usr/bin/mysqlcheck`
- `/usr/bin/mysqldump`
- `/usr/bin/mysqlpump`
- `/usr/sbin/mysqld`
- All binaries are stripped; they contain no debug information.

Configuring the MySQL Server

When you start the MySQL Docker container, you can pass configuration options to the server through the `docker run` command. For example:

```
docker run --name mysql1 -d mysql/mysql-server:tag --character-set-server=utf8mb4 --collation-server=utf8mb4_col1
```

The command starts your MySQL Server with `utf8mb4` as the default character set and `utf8mb4_col1` as the default collation for your databases.

Another way to configure the MySQL Server is to prepare a configuration file and mount it at the location of the server configuration file inside the container. See [Persisting Data and Configuration Changes](#) for details.

Persisting Data and Configuration Changes

Docker containers are in principle ephemeral, and any data or configuration are expected to be lost if the container is deleted or corrupted (see discussions [here](#)). [Docker volumes](#), however, provides a mechanism to persist data created inside a Docker container. At its initialization, the MySQL Server container creates a Docker volume for the server data directory. The JSON output for running the `docker inspect` command on the container has a `Mount` key, whose value provides information on the data directory volume:

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

```
    },
    ...
}
```

The output shows that the source folder `/var/lib/docker/volumes/4f2d463cfc4bdd4baebcb098c97d7da3337195ed2c6572bc0b89f7e845d27652/_data`, in which data is persisted on the host, has been mounted at `/var/lib/mysql`, the server data directory inside the container.

Another way to preserve data is to [bind-mount](#) a host directory using the `--mount` option when creating the container. The same technique can be used to persist the configuration of the server. The following command creates a MySQL Server container and bind-mounts both the data directory and the server configuration file:

```
docker run --name=mysql1 \
--mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
--mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
-d mysql/mysql-server:tag
```

The command mounts `path-on-host-machine/my.cnf` at `/etc/my.cnf` (the server configuration file inside the container), and `path-on-host-machine/datadir` at `/var/lib/mysql` (the data directory inside the container). The following conditions must be met for the bind-mounting to work:

- The configuration file `path-on-host-machine/my.cnf` must already exist, and it must contain the specification for starting the server using the user `mysql`:

```
[mysqld]
user=mysql
```

You can also include other server configuration options in the file.

- The data directory `path-on-host-machine/datadir` must already exist. For server initialization to happen, the directory must be empty. You can also mount a directory prepopulated with data and start the server with it; however, you must make sure you start the Docker container with the same configuration as the server that created the data, and any host files or directories required are mounted when starting the container.

Running Additional Initialization Scripts

If there are any `.sh` or `.sql` scripts you want to run on the database immediately after it has been created, you can put them into a host directory and then mount the directory at `/docker-entrypoint-initdb.d/` inside the container. For example:

```
docker run --name=mysql1 \
--mount type=bind,src=/path-on-host-machine/scripts/,dst=/docker-entrypoint-initdb.d/ \
-d mysql/mysql-server:tag
```

Connect to MySQL from an Application in Another Docker Container

By setting up a Docker network, you can allow multiple Docker containers to communicate with each other, so that a client application in another Docker container can access the MySQL Server in the server container. First, create a Docker network:

```
docker network create my-custom-net
```

Then, when you are creating and starting the server and the client containers, use the `--network` option to put them on network you created. For example:

```
docker run --name=mysql1 --network=my-custom-net -d mysql/mysql-server
```

```
docker run --name=myapp1 --network=my-custom-net -d myapp
```

The `myapp1` container can then connect to the `mysql1` container with the `mysql1` hostname and vice versa, as Docker automatically sets up a DNS for the given container names. In the following example, we run the `mysql` client from inside the `myapp1` container to connect to host `mysql1` in its own container:

```
docker exec -it myapp1 mysql --host=mysql1 --user=myuser --password
```

For other networking techniques for containers, see the [Docker container networking](#) section in the Docker Documentation.

Server Error Log

When the MySQL Server is first started with your server container, a [server error log](#) is NOT generated if either of the following conditions is true:

- A server configuration file from the host has been mounted, but the file does not contain the system variable `log_error` (see [Persisting Data and Configuration Changes](#) on bind-mounting a server configuration file).
- A server configuration file from the host has not been mounted, but the Docker environment variable `MYSQL_LOG_CONSOLE` is `true` (the variable's default state for MySQL 5.7 server containers is `false`). The MySQL Server's error log is then redirected to `stderr`, so that the error log goes into the Docker container's log and is viewable using the `docker logs mysql1-container` command.

To make MySQL Server generate an error log when either of the two conditions is true, use the `--log-error` option to [configure the server](#) to generate the error log at a specific location inside the container. To persist the error log, mount a host file at the location of the error log inside the container as explained in [Persisting Data and Configuration Changes](#). However, you must make sure your MySQL Server inside its container has write access to the mounted host file.

Known Issues

- When using the server system variable `audit_log_file` to configure the audit log file name, use the [loose option modifier](#) with it, or Docker will be unable to start the server.

Docker Environment Variables

When you create a MySQL Server container, you can configure the MySQL instance by using the `--env` option (`-e` in short) and specifying one or more of the following environment variables.

Notes

- None of the variables below has any effect if the data directory you mount is not empty, as no server initialization is going to be attempted then (see [Persisting Data and Configuration Changes](#) for more details). Any pre-existing contents in the folder, including any old server settings, are not modified during the container startup.
- The boolean variables including `MYSQL_RANDOM_ROOT_PASSWORD`, `MYSQL_ONETIME_PASSWORD`, `MYSQL_ALLOW_EMPTY_PASSWORD`, and `MYSQL_LOG_CONSOLE` are made true by setting them with any strings of nonzero lengths. Therefore, setting them to, for example, "0", "false", or "no" does not make them false, but actually makes them true. This is a known issue of the MySQL Server containers.
- `MYSQL_RANDOM_ROOT_PASSWORD`: When this variable is true (which is its default state, unless `MYSQL_ROOT_PASSWORD` is set or `MYSQL_ALLOW_EMPTY_PASSWORD` is set to true), a random password for the server's root user is generated when the Docker container is started. The password

is printed to `stdout` of the container and can be found by looking at the container's log (see [Starting a MySQL Server Instance](#)).

- **MYSQL_ONETIME_PASSWORD**: When the variable is true (which is its default state, unless **MYSQL_ROOT_PASSWORD** is set or **MYSQL_ALLOW_EMPTY_PASSWORD** is set to true), the root user's password is set as expired and must be changed before MySQL can be used normally.
- **MYSQL_DATABASE**: This variable allows you to specify the name of a database to be created on image startup. If a user name and a password are supplied with **MYSQL_USER** and **MYSQL_PASSWORD**, the user is created and granted superuser access to this database (corresponding to **GRANT ALL**). The specified database is created by a **CREATE DATABASE IF NOT EXIST** statement, so that the variable has no effect if the database already exists.
- **MYSQL_USER**, **MYSQL_PASSWORD**: These variables are used in conjunction to create a user and set that user's password, and the user is granted superuser permissions for the database specified by the **MYSQL_DATABASE** variable. Both **MYSQL_USER** and **MYSQL_PASSWORD** are required for a user to be created—if any of the two variables is not set, the other is ignored. If both variables are set but **MYSQL_DATABASE** is not, the user is created without any privileges.

Note

There is no need to use this mechanism to create the root superuser, which is created by default with the password set by either one of the mechanisms discussed in the descriptions for **MYSQL_ROOT_PASSWORD** and **MYSQL_RANDOM_ROOT_PASSWORD**, unless **MYSQL_ALLOW_EMPTY_PASSWORD** is true.

- **MYSQL_ROOT_HOST**: By default, MySQL creates the `'root'@'localhost'` account. This account can only be connected to from inside the container as described in [Connecting to MySQL Server from within the Container](#). To allow root connections from other hosts, set this environment variable. For example, the value `172.17.0.1`, which is the default Docker gateway IP, allows connections from the host machine that runs the container. The option accepts only one entry, but wildcards are allowed (for example, `MYSQL_ROOT_HOST=172.*.*.*` or `MYSQL_ROOT_HOST=%`).
- **MYSQL_LOG_CONSOLE**: When the variable is true (the variable's default state for MySQL 5.7 server containers is `false`), the MySQL Server's error log is redirected to `stderr`, so that the error log goes into the Docker container's log and is viewable using the `docker logs mysqlid-container` command.

Note

The variable has no effect if a server configuration file from the host has been mounted (see [Persisting Data and Configuration Changes](#) on bind-mounting a configuration file).

- **MYSQL_ROOT_PASSWORD**: This variable specifies a password that is set for the MySQL root account.

Warning

Setting the MySQL root user password on the command line is insecure. As an alternative to specifying the password explicitly, you can set the variable with a container file path for a password file, and then mount a file from your host that contains the password at the container file path. This is still not very secure, as the location of the password file is still exposed. It is preferable to use the default settings of **MYSQL_RANDOM_ROOT_PASSWORD** and **MYSQL_ONETIME_PASSWORD** both being true.

- `MYSQL_ALLOW_EMPTY_PASSWORD`. Set it to true to allow the container to be started with a blank password for the root user.

Warning

Setting this variable to true is insecure, because it is going to leave your MySQL instance completely unprotected, allowing anyone to gain complete superuser access. It is preferable to use the default settings of `MYSQL_RANDOM_ROOT_PASSWORD` and `MYSQL_ONETIME_PASSWORD` both being true.

2.7.3 Deploying MySQL on Windows and Other Non-Linux Platforms with Docker

Warning

The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk. This section discusses some known issues for the images when used on non-Linux platforms.

Known Issues for using the MySQL Server Docker images from Oracle on Windows include:

- If you are bind-mounting on the container's MySQL data directory (see [Persisting Data and Configuration Changes](#) for details), you have to set the location of the server socket file with the `--socket` option to somewhere outside of the MySQL data directory; otherwise, the server fails to start. This is because the way Docker for Windows handles file mounting does not allow a host file from being bind-mounted on the socket file.

2.8 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 also normally cannot 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 2, Installing MySQL on Linux](#).

Distribution specific instructions are shown below:

- **Red Hat Linux, Fedora, CentOS**

Note

For a number of Linux distributions, you can install MySQL using the MySQL Yum repository instead of the platform's native software repository. See [Section 2.1, "Installing MySQL on Linux Using the MySQL Yum Repository"](#) for details.

For Red Hat and similar distributions, the MySQL distribution is divided into a number of separate packages, `mysql` for the client tools, `mysql-server` for the server and associated tools, and `mysql-libs` for the libraries. The libraries are required if you want to provide connectivity from different languages and environments such as Perl, Python and others.

To install, use the `yum` command to specify the packages that you want to install. For example:

```
#> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
--> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
--> Running transaction check
---> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
--> Finished Dependency Resolution
Dependencies Resolved

=====
Package                Arch          Version           Repository        Size
=====
Installing:
mysql                  x86_64        5.1.48-2.fc13     updates           889 k
mysql-libs             x86_64        5.1.48-2.fc13     updates           1.2 M
mysql-server          x86_64        5.1.48-2.fc13     updates           8.1 M
Installing for dependencies:
perl-DBD-MySQL        x86_64        4.017-1.fc13      updates           136 k
Transaction Summary
=====
Install      4 Package(s)
Upgrade      0 Package(s)
Total download size: 10 M
Installed size: 30 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
Processing delta metadata
Package(s) data still to download: 10 M
(1/4): mysql-5.1.48-2.fc13.x86_64.rpm | 889 kB    00:04
(2/4): mysql-libs-5.1.48-2.fc13.x86_64.rpm | 1.2 MB    00:06
(3/4): mysql-server-5.1.48-2.fc13.x86_64.rpm | 8.1 MB    00:40
(4/4): perl-DBD-MySQL-4.017-1.fc13.x86_64.rpm | 136 kB    00:00
-----
Total                                          201 kB/s | 10 MB    00:52
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing      : mysql-libs-5.1.48-2.fc13.x86_64           1/4
  Installing      : mysql-5.1.48-2.fc13.x86_64               2/4
  Installing      : perl-DBD-MySQL-4.017-1.fc13.x86_64       3/4
  Installing      : mysql-server-5.1.48-2.fc13.x86_64        4/4
Installed:
mysql.x86_64 0:5.1.48-2.fc13          mysql-libs.x86_64 0:5.1.48-2.fc13
mysql-server.x86_64 0:5.1.48-2.fc13
Dependency Installed:
perl-DBD-MySQL.x86_64 0:4.017-1.fc13
```

Complete!

MySQL and the MySQL server should now be installed. A sample configuration file is installed into `/etc/my.cnf`. An init script, to start and stop the server, is installed into `/etc/init.d/mysqld`. To start the MySQL server use `service`:

```
#> service mysqld start
```

To enable the server to be started and stopped automatically during boot, use `chkconfig`:

```
#> 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 are automatically created for you, if they do not already exist. You should, however, run `mysql_secure_installation` to set the root passwords on your server.

- **Debian, Ubuntu, Kubuntu**

Note

On Debian, Ubuntu, and Kubuntu, MySQL can be installed using the [MySQL APT Repository](#) instead of the platform's native software repository. See [Section 2.3, "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):

```
#> 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:
  dbshell 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 [63.6kB]
Get: 2 http://gb.archive.ubuntu.com jaunty-updates/main libmysqlclient15off 5.1.30really5.0.75-0ubuntu10.5 [
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.
2013-09-24T13:03:09.048353Z 0 [Note] InnoDB: 5.7.44 started; log sequence number 1566036
2013-09-24T13:03:10.057269Z 0 [Note] InnoDB: Starting shutdown...
2013-09-24T13:03:10.857032Z 0 [Note] InnoDB: Shutdown completed; log sequence number 1566036
* 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 installs a number of packages, including the MySQL server, in order to provide the typical tools and application environment. This can mean that you install a large number of packages in addition to the main MySQL package.

During installation, the initial database is created, and you are prompted for the MySQL root password (and confirmation). A configuration file is created in `/etc/mysql/my.cnf`. An init script is created in `/etc/init.d/mysql`.

The server is already started. You can manually start and stop the server using:

```
#> service mysql [start|stop]
```

The service is automatically added to run levels 2, 3, and 4, with stop scripts in the single, shutdown, and restart levels.

2.9 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/>.

2.10 Managing MySQL Server with systemd

If you install MySQL using an RPM or Debian package on the following Linux platforms, server startup and shutdown is managed by systemd:

- RPM package platforms:

- Enterprise Linux variants version 7 and higher
- SUSE Linux Enterprise Server 12 and higher
- Debian family platforms:
 - Debian platforms
 - Ubuntu platforms

If you install MySQL from a generic binary distribution on a platform that uses systemd, you can manually configure systemd support for MySQL following the instructions provided in the post-installation setup section of the [MySQL 5.7 Secure Deployment Guide](#).

If you install MySQL from a source distribution on a platform that uses systemd, obtain systemd support for MySQL by configuring the distribution using the `-DWITH_SYSTEMD=1` CMake option. See [MySQL Source-Configuration Options](#).

The following discussion covers these topics:

- [Overview of systemd](#)
- [Configuring systemd for MySQL](#)
- [Configuring Multiple MySQL Instances Using systemd](#)
- [Migrating from `mysqld\_safe` to systemd](#)

Note

On platforms for which systemd support for MySQL is installed, scripts such as `mysqld_safe` and the System V initialization script are unnecessary and are not installed. For example, `mysqld_safe` can handle server restarts, but systemd provides the same capability, and does so in a manner consistent with management of other services rather than by using an application-specific program.

One implication of the non-use of `mysqld_safe` on platforms that use systemd for server management is that use of `[mysqld_safe]` or `[safe_mysqld]` sections in option files is not supported and might lead to unexpected behavior.

Because systemd has the capability of managing multiple MySQL instances on platforms for which systemd support for MySQL is installed, `mysqld_multi` and `mysqld_multi.server` are unnecessary and are not installed.

Overview of systemd

systemd provides automatic MySQL server startup and shutdown. It also enables manual server management using the `systemctl` command. For example:

```
systemctl {start|stop|restart|status} mysqld
```

Alternatively, use the `service` command (with the arguments reversed), which is compatible with System V systems:

```
service mysqld {start|stop|restart|status}
```

Note

For the `systemctl` or `service` commands, if the MySQL service name is not `mysqld`, use the appropriate name. For example, use `mysql` rather than `mysqld` on Debian-based and SLES systems.

Support for systemd includes these files:

- `mysqld.service` (RPM platforms), `mysql.service` (Debian platforms): systemd service unit configuration file, with details about the MySQL service.
- `mysqld@.service` (RPM platforms), `mysql@.service` (Debian platforms): Like `mysqld.service` or `mysql.service`, but used for managing multiple MySQL instances.
- `mysqld.tmpfiles.d`: File containing information to support the `tmpfiles` feature. This file is installed under the name `mysql.conf`.
- `mysqld_pre_systemd` (RPM platforms), `mysql-system-start` (Debian platforms): Support script for the unit file. This script assists in creating the error log file only if the log location matches a pattern (`/var/log/mysql*.log` for RPM platforms, `/var/log/mysql/*.log` for Debian platforms). In other cases, the error log directory must be writable or the error log must be present and writable for the user running the `mysqld` process.

Configuring systemd for MySQL

To add or change systemd options for MySQL, these methods are available:

- Use a localized systemd configuration file.
- Arrange for systemd to set environment variables for the MySQL server process.
- Set the `MYSQLD_OPTS` systemd variable.

To use a localized systemd configuration file, create the `/etc/systemd/system/mysqld.service.d` directory if it does not exist. In that directory, create a file that contains a `[Service]` section listing the desired settings. For example:

```
[Service]
LimitNOFILE=max_open_files
PIDFile=/path/to/pid/file
Nice=nice_level
LimitCore=core_file_limit
Environment="LD_PRELOAD=/path/to/malloc/library"
Environment="TZ=time_zone_setting"
```

The discussion here uses `override.conf` as the name of this file. Newer versions of systemd support the following command, which opens an editor and permits you to edit the file:

```
systemctl edit mysqld # RPM platforms
systemctl edit mysql  # Debian platforms
```

Whenever you create or change `override.conf`, reload the systemd configuration, then tell systemd to restart the MySQL service:

```
systemctl daemon-reload
systemctl restart mysqld # RPM platforms
systemctl restart mysql  # Debian platforms
```

With systemd, the `override.conf` configuration method must be used for certain parameters, rather than settings in a `[mysqld]`, `[mysqld_safe]`, or `[safe_mysqld]` group in a MySQL option file:

- For some parameters, `override.conf` must be used because systemd itself must know their values and it cannot read MySQL option files to get them.
- Parameters that specify values otherwise settable only using options known to `mysqld_safe` must be specified using systemd because there is no corresponding `mysqld` parameter.

For additional information about using systemd rather than `mysqld_safe`, see [Migrating from `mysqld\_safe` to systemd](#).

You can set the following parameters in `override.conf`:

- To specify the process ID file:
 - As of MySQL 5.7.10: Use `override.conf` and change both `PIDFile` and `ExecStart` to name the PID file path name. Any setting of the process ID file in MySQL option files is ignored. To modify `ExecStart`, it must first be cleared. For example:

```
[Service]
PIDFile=/var/run/mysqld/mysqld-custom.pid
ExecStart=
ExecStart=/usr/sbin/mysqld --pid-file=/var/run/mysqld/mysqld-custom.pid $MYSQLD_OPTS
```

- Before MySQL 5.7.10: Use `PIDFile` in `override.conf` rather than the `--pid-file` option for `mysqld` or `mysqld_safe`. systemd must know the PID file location so that it can restart or stop the server. If the PID file value is specified in a MySQL option file, the value must match the `PIDFile` value or MySQL startup may fail.
- To set the number of file descriptors available to the MySQL server, use `LimitNOFILE` in `override.conf` rather than the `open_files_limit` system variable for `mysqld` or `--open-files-limit` option for `mysqld_safe`.
- To set the maximum core file size, use `LimitCore` in `override.conf` rather than the `--core-file-size` option for `mysqld_safe`.
- To set the scheduling priority for the MySQL server, use `Nice` in `override.conf` rather than the `--nice` option for `mysqld_safe`.

Some MySQL parameters are configured using environment variables:

- `LD_PRELOAD`: Set this variable if the MySQL server should use a specific memory-allocation library.
- `TZ`: Set this variable to specify the default time zone for the server.

There are multiple ways to specify environment variable values for use by the MySQL server process managed by systemd:

- Use `Environment` lines in the `override.conf` file. For the syntax, see the example in the preceding discussion that describes how to use this file.
- Specify the values in the `/etc/sysconfig/mysql` file (create the file if it does not exist). Assign values using the following syntax:

```
LD_PRELOAD=/path/to/malloc/library
TZ=time_zone_setting
```

After modifying `/etc/sysconfig/mysql`, restart the server to make the changes effective:

```
systemctl restart mysqld # RPM platforms
systemctl restart mysql  # Debian platforms
```

To specify options for `mysqld` without modifying systemd configuration files directly, set or unset the `MYSQLD_OPTS` systemd variable. For example:

```
systemctl set-environment MYQSLD_OPTS="--general_log=1"
systemctl unset-environment MYQSLD_OPTS
```

`MYSQLD_OPTS` can also be set in the `/etc/sysconfig/mysql` file.

After modifying the systemd environment, restart the server to make the changes effective:

```
systemctl restart mysqld # RPM platforms
systemctl restart mysql # Debian platforms
```

For platforms that use systemd, the data directory is initialized if empty at server startup. This might be a problem if the data directory is a remote mount that has temporarily disappeared: The mount point would appear to be an empty data directory, which then would be initialized as a new data directory. As of MySQL 5.7.20, to suppress this automatic initialization behavior, specify the following line in the `/etc/sysconfig/mysql` file (create the file if it does not exist):

```
NO_INIT=true
```

Configuring Multiple MySQL Instances Using systemd

This section describes how to configure systemd for multiple instances of MySQL.

Note

Because systemd has the capability of managing multiple MySQL instances on platforms for which systemd support is installed, `mysqld_multi` and `mysqld_multi.server` are unnecessary and are not installed. This is true as of MySQL 5.7.13 for RPM platforms, 5.7.19 for Debian platforms.

To use multiple-instance capability, modify the `my.cnf` option file to include configuration of key options for each instance. These file locations are typical:

- `/etc/my.cnf` or `/etc/mysql/my.cnf` (RPM platforms)
- `/etc/mysql/mysql.conf.d/mysqld.cnf` (Debian platforms)

For example, to manage two instances named `replica01` and `replica02`, add something like this to the option file:

RPM platforms:

```
[mysqld@replica01]
datadir=/var/lib/mysql-replica01
socket=/var/lib/mysql-replica01/mysql.sock
port=3307
log-error=/var/log/mysqld-replica01.log
[mysqld@replica02]
datadir=/var/lib/mysql-replica02
socket=/var/lib/mysql-replica02/mysql.sock
port=3308
log-error=/var/log/mysqld-replica02.log
```

Debian platforms:

```
[mysqld@replica01]
datadir=/var/lib/mysql-replica01
socket=/var/lib/mysql-replica01/mysql.sock
port=3307
```

```
log-error=/var/log/mysql/replica01.log
[mysqld@replica02]
datadir=/var/lib/mysql-replica02
socket=/var/lib/mysql-replica02/mysql.sock
port=3308
log-error=/var/log/mysql/replica02.log
```

The replica names shown here use `@` as the delimiter because that is the only delimiter supported by systemd.

Instances then are managed by normal systemd commands, such as:

```
systemctl start mysqld@replica01
systemctl start mysqld@replica02
```

To enable instances to run at boot time, do this:

```
systemctl enable mysqld@replica01
systemctl enable mysqld@replica02
```

Use of wildcards is also supported. For example, this command displays the status of all replica instances:

```
systemctl status 'mysqld@replica*'
```

For management of multiple MySQL instances on the same machine, systemd automatically uses a different unit file:

- `mysqld@.service` rather than `mysqld.service` (RPM platforms)
- `mysql@.service` rather than `mysql.service` (Debian platforms)

In the unit file, `%I` and `%i` reference the parameter passed in after the `@` marker and are used to manage the specific instance. For a command such as this:

```
systemctl start mysqld@replica01
```

systemd starts the server using a command such as this:

```
mysqld --defaults-group-suffix=%I ...
```

The result is that the `[server]`, `[mysqld]`, and `[mysqld@replica01]` option groups are read and used for that instance of the service.

Note

On Debian platforms, AppArmor prevents the server from reading or writing `/var/lib/mysql-replica*`, or anything other than the default locations. To address this, you must customize or disable the profile in `/etc/apparmor.d/usr.sbin.mysqld`.

Note

On Debian platforms, the packaging scripts for MySQL uninstallation cannot currently handle `mysqld@` instances. Before removing or upgrading the package, you must stop any extra instances manually first.

Migrating from mysqld_safe to systemd

Because `mysqld_safe` is not installed on platforms that use systemd to manage MySQL, options previously specified for that program (for example, in an `[mysqld_safe]` or `[safe_mysqld]` option group) must be specified another way:

- Some `mysqld_safe` options are also understood by `mysqld` and can be moved from the `[mysqld_safe]` or `[safe_mysqld]` option group to the `[mysqld]` group. This does *not* include `--pid-file`, `--open-files-limit`, or `--nice`. To specify those options, use the `override.conf` systemd file, described previously.

Note

On systemd platforms, use of `[mysqld_safe]` and `[safe_mysqld]` option groups is not supported and may lead to unexpected behavior.

- For some `mysqld_safe` options, there are similar `mysqld` options. For example, the `mysqld_safe` option for enabling `syslog` logging is `--syslog`, which is deprecated. For `mysqld`, enable the `log_syslog` system variable instead. For details, see [The Error Log](#).
- `mysqld_safe` options not understood by `mysqld` can be specified in `override.conf` or environment variables. For example, with `mysqld_safe`, if the server should use a specific memory allocation library, this is specified using the `--malloc-lib` option. For installations that manage the server with systemd, arrange to set the `LD_PRELOAD` environment variable instead, as described previously.

Chapter 3 Installing MySQL on Solaris

Table of Contents

3.1 Installing MySQL on Solaris Using a Solaris PKG	40
---	----

Note

MySQL 5.7 supports Solaris 11 (Update 3 and later).

MySQL on Solaris is available in a number of different formats.

- For information on installing using the native Solaris PKG format, see [Section 3.1, “Installing MySQL on Solaris Using a Solaris PKG”](#).
- To use a standard `tar` binary installation, use the notes provided in [Chapter 1, *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.

Important

The installation packages have a dependency on the Oracle Developer Studio 12.5 Runtime Libraries, which must be installed before you run the MySQL installation package. See the download options for Oracle Developer Studio [here](#). The installation package enables you to install the runtime libraries only instead of the full Oracle Developer Studio; see instructions in [Installing Only the Runtime Libraries on Oracle Solaris 11](#).

To obtain a binary MySQL distribution for Solaris in tarball or PKG format, <https://dev.mysql.com/downloads/mysql/5.7.html>.

Additional notes to be aware of when installing and using MySQL on Solaris:

- If you want to use MySQL with the `mysql` user and group, use the `groupadd` and `useradd` commands:

```
groupadd mysql
useradd -g mysql -s /bin/false mysql
```

- If you install MySQL using a binary tarball distribution on Solaris, because the Solaris `tar` cannot handle long file names, use GNU `tar` (`gtar`) to unpack the distribution. If you do not have GNU `tar` on your system, install it with the following command:

```
pkg install archiver/gnu-tar
```

- You should mount any file systems on which you intend to store `InnoDB` files with the `forcedirectio` option. (By default mounting is done without this option.) Failing to do so causes a significant drop in performance when using the `InnoDB` storage engine on this platform.
- If you would like MySQL to start automatically, you can copy `support-files/mysql.server` to `/etc/init.d` and create a symbolic link to it named `/etc/rc3.d/S99mysql.server`.
- If too many processes try to connect very rapidly to `mysqld`, you should see this error in the MySQL log:

```
Error in accept: Protocol error
```

You might try starting the server with the `--back_log=50` option as a workaround for this.

- To configure the generation of core files on Solaris you should use the `coreadm` command. Because of the security implications of generating a core on a `setuid()` application, by default, Solaris does not support core files on `setuid()` programs. However, you can modify this behavior using `coreadm`. If you enable `setuid()` core files for the current user, they are generated using mode 600, and are owned by the superuser.

3.1 Installing MySQL on Solaris Using a Solaris PKG

You can install MySQL on Solaris using a binary package of the native Solaris PKG format instead of the binary tarball distribution.

Important

The installation package has a dependency on the Oracle Developer Studio 12.5 Runtime Libraries, which must be installed before you run the MySQL installation package. See the download options for Oracle Developer Studio [here](#). The installation package enables you to install the runtime libraries only instead of the full Oracle Developer Studio; see instructions in [Installing Only the Runtime Libraries on Oracle Solaris 11](#).

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

```
$> gunzip mysql-5.7.44-solaris11-x86_64.pkg.gz
```

To install a new package, use `pkgadd` and follow the onscreen prompts. You must have root privileges to perform this operation:

```
$> pkgadd -d mysql-5.7.44-solaris11-x86_64.pkg
The following packages are available:
  1  mysql      MySQL Community Server (GPL)
                        (i86pc) 5.7.44
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

The PKG installer installs all of the files and tools needed, and then initializes your database if one does not exist. To complete the installation, you should set the root password for MySQL as provided in the instructions at the end of the installation. Alternatively, you can run the `mysql_secure_installation` script that comes with the installation.

By default, the PKG package installs MySQL under the root path `/opt/mysql`. You can change only the installation root path when using `pkgadd`, which can be used to install MySQL in a different Solaris zone. If you need to install in a specific directory, use a binary `tar` file distribution.

The `pkg` installer copies a suitable startup script for MySQL into `/etc/init.d/mysql`. To enable MySQL to startup and shutdown automatically, you should create a link between this file and the init script directories. For example, to ensure safe startup and shutdown of MySQL you could use the following commands to add the right links:

```
$> ln /etc/init.d/mysql /etc/rc3.d/S91mysql
$> ln /etc/init.d/mysql /etc/rc0.d/K02mysql
```

To remove MySQL, the installed package name is `mysql`. You can use this in combination with the `pkgrm` command to remove the installation.

To upgrade when using the Solaris package file format, you must remove the existing installation before installing the updated package. Removal of the package does not delete the existing database information, only the server, binaries and support files. The typical upgrade sequence is therefore:

```
$> mysqladmin shutdown
$> pkgrm mysql
$> pkgadd -d mysql-5.7.44-solaris11-x86_64.pkg
$> mysqld_safe &
$> mysql_upgrade
```

You should check the notes in [Upgrading MySQL](#) before performing any upgrade.

Chapter 4 Installing MySQL on FreeBSD

This section provides information about installing MySQL on variants of FreeBSD Unix.

You can install MySQL on FreeBSD by using the binary distribution provided by Oracle. For more information, see [Chapter 1, *Installing MySQL on Unix/Linux Using Generic Binaries*](#).

The easiest (and preferred) way to install MySQL is to use the `mysql-server` and `mysql-client` ports available at <http://www.freebsd.org/>. Using these ports gives you the following benefits:

- A working MySQL with all optimizations enabled that are known to work on your version of FreeBSD.
- Automatic configuration and build.
- Startup scripts installed in `/usr/local/etc/rc.d`.
- The ability to use `pkg_info -L` to see which files are installed.
- The ability to use `pkg_delete` to remove MySQL if you no longer want it on your machine.

The MySQL build process requires GNU make (`gmake`) to work. If GNU `make` is not available, you must install it first before compiling MySQL.

To install using the ports system:

```
# cd /usr/ports/databases/mysql57-server
# make
...
# cd /usr/ports/databases/mysql57-client
# make
...
```

The standard port installation places the server into `/usr/local/libexec/mysqld`, with the startup script for the MySQL server placed in `/usr/local/etc/rc.d/mysql-server`.

Some additional notes on the BSD implementation:

- To remove MySQL after installation using the ports system:

```
# cd /usr/ports/databases/mysql57-server
# make deinstall
...
# cd /usr/ports/databases/mysql57-client
# make deinstall
...
```

- If you get problems with the current date in MySQL, setting the `TZ` variable should help. See [Environment Variables](#).

Chapter 5 Initializing the Data Directory

After MySQL is installed, the data directory must be initialized, including the tables in the `mysql` system database:

- For some MySQL installation methods, data directory initialization is automatic, as described in [Postinstallation Setup and Testing](#).
- For other installation methods, you must initialize the data directory manually. These include installation from generic binary and source distributions on Unix and Unix-like systems, and installation from a ZIP Archive package on Windows.

This section describes how to initialize the data directory manually for MySQL installation methods for which data directory initialization is not automatic. For some suggested commands that enable testing whether the server is accessible and working properly, see [Testing the Server](#).

- [Data Directory Initialization Overview](#)
- [Data Directory Initialization Procedure](#)
- [Server Actions During Data Directory Initialization](#)
- [Post-Initialization root Password Assignment](#)

Data Directory Initialization Overview

In the examples shown here, the server is intended to run under the user ID of the `mysql` login account. Either create the account if it does not exist (see [Create a mysql User and Group](#)), or substitute the name of a different existing login account that you plan to use for running the server.

1. Change location to the top-level directory of your MySQL installation, which is typically `/usr/local/mysql` (adjust the path name for your system as necessary):

```
cd /usr/local/mysql
```

Within this directory are several files and subdirectories, including the `bin` subdirectory that contains the server as well as client and utility programs.

2. The `secure_file_priv` system variable limits import and export operations to a specific directory. Create a directory whose location can be specified as the value of that variable:

```
mkdir mysql-files
```

Grant directory user and group ownership to the `mysql` user and `mysql` group, and set the directory permissions appropriately:

```
chown mysql:mysql mysql-files
chmod 750 mysql-files
```

3. Use the server to initialize the data directory, including the `mysql` database containing the initial MySQL grant tables that determine how users are permitted to connect to the server. For example:

```
bin/mysqld --initialize --user=mysql
```

For important information about the command, especially regarding command options you might use, see [Data Directory Initialization Procedure](#). For details about how the server performs initialization, see [Server Actions During Data Directory Initialization](#).

Typically, data directory initialization need be done only after you first install MySQL. (For upgrades to an existing installation, perform the upgrade procedure instead; see [Upgrading MySQL](#).) However, the command that initializes the data directory does not overwrite any existing `mysql` database tables, so it is safe to run in any circumstances.

Note

Initialization of the data directory might fail if required system libraries are missing. For example, you might see an error like this:

```
bin/mysqld: error while loading shared libraries:
libnuma.so.1: cannot open shared object file:
No such file or directory
```

If this happens, you must install the missing libraries manually or with your system's package manager. Then retry the data directory initialization command.

4. If you want to deploy the server with automatic support for secure connections, use the `mysql_ssl_rsa_setup` utility to create default SSL and RSA files:

```
bin/mysql_ssl_rsa_setup
```

For more information, see [mysql_ssl_rsa_setup — Create SSL/RSA Files](#).

5. In the absence of any option files, the server starts with its default settings. (See [Server Configuration Defaults](#).) To explicitly specify options that the MySQL server should use at startup, put them in an option file such as `/etc/my.cnf` or `/etc/mysql/my.cnf`. (See [Using Option Files](#).) For example, you can use an option file to set the `secure_file_priv` system variable.
6. To arrange for MySQL to start without manual intervention at system boot time, see [Starting and Stopping MySQL Automatically](#).
7. 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](#).

Data Directory Initialization Procedure

Change location to the top-level directory of your MySQL installation, which is typically `/usr/local/mysql` (adjust the path name for your system as necessary):

```
cd /usr/local/mysql
```

To initialize the data directory, invoke `mysqld` with the `--initialize` or `--initialize-insecure` option, depending on whether you want the server to generate a random initial password for the `'root'@'localhost'` account, or to create that account with no password:

- Use `--initialize` for “secure by default” installation (that is, including generation of a random initial `root` password). In this case, the password is marked as expired and you must choose a new one.
- With `--initialize-insecure`, no `root` password is generated. This is insecure; it is assumed that you assign a password to the account in timely fashion before putting the server into production use.

For instructions on assigning a new `'root'@'localhost'` password, see [Post-Initialization root Password Assignment](#).

Note

The server writes any messages (including any initial password) to its standard error output. This may be redirected to the error log, so look there if you do not see the messages on your screen. For information about the error log, including where it is located, see [The Error Log](#).

On Windows, use the `--console` option to direct messages to the console.

On Unix and Unix-like systems, it is important for the database directories and files to be owned by the `mysql` login account so that the server has read and write access to them when you run it later. To ensure this, start `mysqld` from the system `root` account and include the `--user` option as shown here:

```
bin/mysqld --initialize --user=mysql
bin/mysqld --initialize-insecure --user=mysql
```

Alternatively, execute `mysqld` while logged in as `mysql`, in which case you can omit the `--user` option from the command.

On Windows, use one of these commands:

```
bin\mysqld --initialize --console
bin\mysqld --initialize-insecure --console
```

Note

Data directory initialization might fail if required system libraries are missing. For example, you might see an error like this:

```
bin/mysqld: error while loading shared libraries:
libnuma.so.1: cannot open shared object file:
No such file or directory
```

If this happens, you must install the missing libraries manually or with your system's package manager. Then retry the data directory initialization command.

It might be necessary to specify other options such as `--basedir` or `--datadir` if `mysqld` cannot identify the correct locations for the installation directory or data directory. For example (enter the command on a single line):

```
bin/mysqld --initialize --user=mysql
--basedir=/opt/mysql/mysql
--datadir=/opt/mysql/mysql/data
```

Alternatively, put the relevant option settings in an option file and pass the name of that file to `mysqld`. For Unix and Unix-like systems, suppose that the option file name is `/opt/mysql/mysql/etc/my.cnf`. Put these lines in the file:

```
[mysqld]
basedir=/opt/mysql/mysql
datadir=/opt/mysql/mysql/data
```

Then invoke `mysqld` as follows (enter the command on a single line, with the `--defaults-file` option first):

```
bin/mysqld --defaults-file=/opt/mysql/mysql/etc/my.cnf
--initialize --user=mysql
```

On Windows, suppose that `C:\my.ini` contains these lines:

```
[mysqld]
basedir=C:\Program Files\MySQL\MySQL Server 5.7
```

```
datadir=D:\\MySQLdata
```

Then invoke `mysqld` as follows (again, you should enter the command on a single line, with the `--defaults-file` option first):

```
bin\\mysqld --defaults-file=C:\\my.ini
--initialize --console
```

Important

When initializing the data directory, you should not specify any options other than those used for setting directory locations such as `--basedir` or `--datadir`, and the `--user` option if needed. Options to be employed by the MySQL server during normal use can be set when restarting it following initialization. See the description of the `--initialize` option for further information.

Server Actions During Data Directory Initialization

Note

The data directory initialization sequence performed by the server does not substitute for the actions performed by `mysql_secure_installation` and `mysql_ssl_rsa_setup`. See [mysql_secure_installation — Improve MySQL Installation Security](#), and [mysql_ssl_rsa_setup — Create SSL/RSA Files](#).

When invoked with the `--initialize` or `--initialize-insecure` option, `mysqld` performs the following actions during the data directory initialization sequence:

1. The server checks for the existence of the data directory as follows:
 - If no data directory exists, the server creates it.
 - If the data directory exists but is not empty (that is, it contains files or subdirectories), the server exits after producing an error message:

```
[ERROR] --initialize specified but the data directory exists. Aborting.
```

In this case, remove or rename the data directory and try again.

As of MySQL 5.7.11, an existing data directory is permitted to be nonempty if every entry either has a name that begins with a period (.) or is named using an `--ignore-db-dir` option.

Note

Avoid the use of the `--ignore-db-dir` option, which has been deprecated since MySQL 5.7.16.

2. Within the data directory, the server creates the `mysql` system database and its tables, including the grant tables, time zone tables, and server-side help tables. See [The mysql System Database](#).
3. The server initializes the `system tablespace` and related data structures needed to manage `InnoDB` tables.

Note

After `mysqld` sets up the `InnoDB system tablespace`, certain changes to tablespace characteristics require setting up a whole new `instance`.

Qualifying changes include the file name of the first file in the system tablespace and the number of undo logs. If you do not want to use the default values, make sure that the settings for the `innodb_data_file_path` and `innodb_log_file_size` configuration parameters are in place in the MySQL [configuration file](#) *before* running `mysqld`. Also make sure to specify as necessary other parameters that affect the creation and location of InnoDB files, such as `innodb_data_home_dir` and `innodb_log_group_home_dir`.

If those options are in your configuration file but that file is not in a location that MySQL reads by default, specify the file location using the `--defaults-extra-file` option when you run `mysqld`.

4. The server creates a `'root'@'localhost'` superuser account and other reserved accounts (see [Reserved Accounts](#)). Some reserved accounts are locked and cannot be used by clients, but `'root'@'localhost'` is intended for administrative use and you should assign it a password.

Server actions with respect to a password for the `'root'@'localhost'` account depend on how you invoke it:

- With `--initialize` but not `--initialize-insecure`, the server generates a random password, marks it as expired, and writes a message displaying the password:

```
[Warning] A temporary password is generated for root@localhost:
iTag*AfrH5ej
```

- With `--initialize-insecure`, (either with or without `--initialize` because `--initialize-insecure` implies `--initialize`), the server does not generate a password or mark it expired, and writes a warning message:

```
[Warning] root@localhost is created with an empty password ! Please
consider switching off the --initialize-insecure option.
```

For instructions on assigning a new `'root'@'localhost'` password, see [Post-Initialization root Password Assignment](#).

5. The server populates the server-side help tables used for the `HELP` statement (see [HELP Statement](#)). The server does not populate the time zone tables. To do so manually, see [MySQL Server Time Zone Support](#).
6. If the `init_file` system variable was given to name a file of SQL statements, the server executes the statements in the file. This option enables you to perform custom bootstrapping sequences.

When the server operates in bootstrap mode, some functionality is unavailable that limits the statements permitted in the file. These include statements that relate to account management (such as `CREATE USER` or `GRANT`), replication, and global transaction identifiers.

7. The server exits.

Post-Initialization root Password Assignment

After you initialize the data directory by starting the server with `--initialize` or `--initialize-insecure`, start the server normally (that is, without either of those options) and assign the `'root'@'localhost'` account a new password:

1. Start the server. For instructions, see [Starting the Server](#).
2. Connect to the server:

- If you used `--initialize` but not `--initialize-insecure` to initialize the data directory, connect to the server as `root`:

```
mysql -u root -p
```

Then, at the password prompt, enter the random password that the server generated during the initialization sequence:

```
Enter password: (enter the random root password here)
```

Look in the server error log if you do not know this password.

- If you used `--initialize-insecure` to initialize the data directory, connect to the server as `root` without a password:

```
mysql -u root --skip-password
```

3. After connecting, use an `ALTER USER` statement to assign a new `root` password:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'root-password';
```

See also [Securing the Initial MySQL Account](#).

Note

Attempts to connect to the host `127.0.0.1` normally resolve to the `localhost` account. However, this fails if the server is run with `skip_name_resolve` enabled. If you plan to do that, make sure that an account exists that can accept a connection. For example, to be able to connect as `root` using `--host=127.0.0.1` or `--host=::1`, create these accounts:

```
CREATE USER 'root'@'127.0.0.1' IDENTIFIED BY 'root-password';  
CREATE USER 'root'@'::1' IDENTIFIED BY 'root-password';
```

It is possible to put those statements in a file to be executed using the `init_file` system variable, as discussed in [Server Actions During Data Directory Initialization](#).