MySQL and Linux/Unix

Abstract

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

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

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

Document generated on: 2016-11-04 (revision: 49749)

Table of Contents

Preface and Legal Notices

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

Legal Notices

Documentation Accessibility

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

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

Access to Oracle Support

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. For other platform-specific package formats, see the other platform-specific sections. For example, for Windows distributions, see Installing MySQL on Microsoft Windows.

To obtain MySQL, see How to Get MySQL.

MySQL compressed `tar` file binary distributions have names of the form `mysql-`*`VERSION`*`-`*`OS`*`.tar.gz`, where *`VERSION`* is a number (for example, `5.7.18`), and *`OS`* indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

> **Warning**
>
> If you have previously installed MySQL using your operating system native package management system, such as `yum` or `apt-get`, you may experience problems installing using a native binary. Make sure your previous MySQL installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check for configuration files such as `/etc/my.cnf` or the `/etc/mysql` directory and delete them.
>
> For information about replacing third-party packages with official MySQL packages, see the related Apt guide or Yum guide.

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

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

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

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

Directory	Contents of Directory
`bin`, `scripts`	`mysqld` server, client and utility programs

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

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:

```
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> mkdir mysql-files
shell> chmod 750 mysql-files
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> bin/mysql_install_db --user=mysql    # Before MySQL 5.7.6
shell> bin/mysqld --initialize --user=mysql # MySQL 5.7.6 and up
shell> bin/mysql_ssl_rsa_setup              # MySQL 5.7.6 and up
shell> chown -R root .
shell> chown -R mysql data mysql-files
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

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

> **Note**
>
> Before MySQL 5.7.4, the procedure does not assign passwords to MySQL accounts. To do so, use the instructions in Securing the Initial MySQL Accounts.

The `mysql-files` directory provides a convenient location to use as the value of the `secure_file_priv` system variable that limits import/export operations to a specific directory. See Server System Variables.

Before MySQL 5.7.5, `mysql_install_db` creates a default option file named `my.cnf` in the base installation directory. This file is created from a template included in the distribution package named `my-default.cnf`. For more information, see Server Configuration Defaults.

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

Create a mysql User and Group

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

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

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

Obtain and Unpack the Distribution

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

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

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

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

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

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

To install MySQL from a compressed `tar` file binary distribution, your system must have GNU `gunzip` to uncompress the distribution and a reasonable `tar` to unpack it. If your `tar` program supports the `z` option, it can both uncompress and unpack the file.

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

If your `tar` does not have `z` option support, use `gunzip` to unpack the distribution and `tar` to unpack it. Replace the preceding `tar` command with the following alternative command to uncompress and extract the distribution:

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

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

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

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

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

Perform Postinstallation Setup

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

Chapter 2 Installing MySQL on Linux

Table of Contents

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

- Installing with Yum using the MySQL Yum repository. For details, see Section 2.1, “Installing MySQL on Linux Using the MySQL Yum Repository”.
- Installing with APT using the MySQL APT Repository. For details, see Section 2.3, “Installing MySQL on Linux Using the MySQL APT Repository”.
- Installing with Zypper using the MySQL SLES Repository. For details, see Section 2.4, “Installing MySQL on Linux Using the MySQL SLES Repository”.
- Installing using a precompiled RPM package. For more information, see Section 2.5, “Installing MySQL on Linux Using RPM Packages from Oracle”.
- Installing using a precompiled Debian package. For more information, see Section 2.6, “Installing MySQL on Linux Using Debian Packages from Oracle”.
- Installing from a generic binary package in `.tar.gz` format. See Chapter 1, *Installing MySQL on Unix/Linux Using Generic Binaries* for more information.
- Installing using Oracle's Unbreakable Linux Network (ULN). For more information, see Installing MySQL Using Unbreakable Linux Network (ULN).
- Extracting and compiling MySQL from a source distribution. For detailed instructions, see Installing MySQL from Source.

As an alternative, you can use the package manager on your system to automatically download and install MySQL with packages from the native software repositories of your Linux distribution. These native packages are often several versions behind the currently available release. You will also normally be unable to install development milestone releases (DMRs), as these are not usually made available in the native repositories. For more information on using the native package installers, see Section 2.7, “Installing MySQL on Linux from the Native Software Repositories”.

> **Note**
>
> For many Linux installations, you will want to set up MySQL to be started automatically when your machine starts. Many of the native package installations perform this operation for you, but for source, binary and RPM solutions you may need to set this up separately. The required script, `mysql.server`, can be found

> in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. You can install it as `/etc/init.d/mysql` for automatic MySQL startup and shutdown. See `mysql.server` — MySQL Server Startup Script.

2.1 Installing MySQL on Linux Using the MySQL Yum Repository

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

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

Currently, the MySQL Yum repository for the above-mentioned platforms provides RPM packages for installing the MySQL server, client, MySQL Workbench, MySQL Utilities, Connector/ODBC, and Connector/Python (not all packages are available for all the platforms; see Installing Additional MySQL Products and Components with Yum for details).

Before You Start

As a popular, open-source software, MySQL, in its original or re-packaged form, is widely installed on many systems from various sources, including different software download sites, software repositories, and so on. The following instructions assume that MySQL is not already installed on your system using a third-party-distributed RPM package; if that is not the case, follow the instructions given in 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:

Adding[1] the MySQL Yum Repository

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

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

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

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

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

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

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

For an EL7-based system:

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

For Fedora 23:

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

For Fedora 24:

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

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

```
shell> sudo rpm -Uvh mysql57-community-release-el5-{version-number}.noarch.rpm
```

The installation command adds the MySQL Yum repository to your system's repository list and downloads the GnuPG key to check the integrity of the software packages. See Signature Checking Using GnuPG for details on GnuPG key checking.

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

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

> **Note**
>
> Once the MySQL Yum repository is enabled on your system, any system-wide update by the `yum update` command (or `dnf upgrade` for dnf-enabled systems) will upgrade MySQL packages on your system and also replace any native third-party packages, if Yum finds replacements for them in the MySQL Yum repository; see Upgrading MySQL with the MySQL Yum Repository and, for a discussion on some possible effects of that on your system, see Upgrading the Shared Client Libraries.

Selecting a Release Series

When using the MySQL Yum repository, the latest GA series (currently MySQL 5.7) is selected for installation by default. If this is what you want, you can skip to the next step, Installing MySQL.

Within the MySQL Yum repository, different release series of the MySQL Community Server are hosted in different subrepositories. The subrepository for the latest GA series (currently MySQL 5.7) is enabled by default, and the subrepositories for all other series (for example, the MySQL 5.6 series) are disabled by default. Use this command to see all the subrepositories in the MySQL Yum repository, and see which of them are enabled or disabled (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

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

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

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

For dnf-enabled platforms:

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

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

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

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

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

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

Verify that the correct subrepositories have been enabled and disabled by running the following command and checking its output (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

3 Installing MySQL

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

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

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

4 Starting the MySQL Server

Start the MySQL server with the following command:

```
```

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

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

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

```
shell> 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:

```
shell> mysql -uroot -p
```

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

> **Note**
>
> MySQL's validate_password plugin is installed by default. This will require that passwords contain at least one upper case letter, one lower case 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 will not be able to install these packages (and vice versa).
>
> - akonadi-mysql

Installing Additional MySQL Products and Components with Yum

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

```
shell> sudo yum --disablerepo=\* --enablerepo='mysql*-community*' list available
```

Install any packages of your choice with the following command, replacing *`package-name`* with name of the package (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

For example, to install MySQL Workbench on Fedora 24:

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

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

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

Updating MySQL with Yum

Besides installation, you can also perform updates for MySQL products and components using the MySQL Yum repository. See 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:

1. Backing Up Your Database

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

2. Adding the MySQL Yum Repository

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

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

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

After updating MySQL using the Yum repository, applications compiled with older versions of the shared client libraries should continue to work. However, *if you want to recompile applications and dynamically*

link them with the updated libraries, see Upgrading the Shared Client Libraries, for some special considerations.

Replacing a Nonnative Third-Party Distribution of MySQL

If you have installed a third-party distribution of MySQL from a nonnative software repository (that is, a software repository not provided by your own Linux distribution), follow these steps:

1. Backing Up Your Database

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

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

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

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

```
shell> yum list installed mariadb\*
MariaDB-common.i686                     10.0.4-1                        @mariadb
MariaDB-compat.i686                     10.0.4-1                        @mariadb
MariaDB-server.i686                     10.0.4-1                        @mariadb
```

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

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

```
shell> yum list installed Percona\*
Percona-Server-client-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
Percona-Server-server-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
Percona-Server-shared-55.i686     5.5.39-rel36.0.el6          @percona-release-i386
percona-release.noarch            0.1-3                       @/percona-release-0.1-3.noarch
```

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

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

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

The next step is to stop Yum from receiving packages from the nonnative repository. If the `yum-config-manager` utility is supported on your platform, you can, for example, use this command for

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

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

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

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

You can perform the same task by removing the entry for the software repository existing in one of the repository files under the `/etc/yum.repos.d/` directory. This is how the entry typically looks 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 (for dnf-enabled systems, replace `yum` in the command with `dnf`):

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

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

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

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 following Linux platforms: :

- Debian 7.x (“wheezy”)
- Debian 8.x (“jessie”)
- Ubuntu 12.04 LTS (“Precise Pangolin”)
- Ubuntu 14.04 LTS (“Trusty Tahr”)
- Ubuntu 16.04 (“Xenial Xerus”)

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.

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

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 2.1 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 2.2 RPM Packages for the MySQL Enterprise Edition

Package Name	Summary
mysql-commercial-server	Database server and related tools
mysql-commercial-client	MySQL client applications and tools
mysql-commercial-common	Common files for server and client libraries
mysql-commercial-devel	Development header files and libraries for MySQL database client applications
mysql-commercial-libs	Shared libraries for MySQL database client applications
mysql-commercial-libs-compat	Shared compatibility libraries for previous MySQL installations
mysql-commercial-embedded	MySQL embedded library
mysql-commercial-embedded-devel	Development header files and libraries for MySQL as an embeddable library
mysql-commercial-test	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 2.3 MySQL Linux RPM Package Distribution Identifiers

distribution Value	Intended Use
`el5`, `el6`, `el7`	Red Hat Enterprise Linux/Oracle Linux/CentOS 5, 6, or 7
`fc23`, `fc24`	Fedora 23 or 24
`sles12`	SUSE Linux Enterprise Server 12

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

```
shell> 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, minimal 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 5:

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

Replace `yum` with `zypper` for SLES systems, and with `dnf` for dnf-enabled systems (like Fedora 23).

For Red Hat Enterprise Linux/Oracle Linux/CentOS 5 systems, there is an extra package (`mysql-version-el5-arch.rpm`) to be installed; use the following command:

```
shell> 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 5:

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

Replace `yum` with `zypper` for SLES systems, and with `dnf` for dnf-enabled systems (like Fedora 23).

For Red Hat Enterprise Linux/Oracle Linux/CentOS 5 systems:

```
shell> 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 2.4 MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone

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

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

> **Note**
>
> 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:

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

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

```
shell> 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.

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.
- The validate_password plugin is installed and enabled.
- A superuser account `'root'@'localhost'` is created. A password for the superuser is set and stored in the error log file. To reveal it, use the following command for RHEL, Oracle Linux, CentOS, and Fedora systems:

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

  Use the following command for SLES systems:

  ```
  shell> 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:

```
shell> mysql -uroot -p
```

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

> **Note**
>
> MySQL's validate_password plugin is installed by default. This will require that passwords contain at least one upper case letter, one lower case letter, one digit, and one special character, and that the total password length is at least 8 characters.

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

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

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:

```
shell> 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.

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 (currently, Debian 7 and 8, and Ubuntu 12, 14, and 16 are supported). 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 2.5 MySQL Debian and Ubuntu Installation Packages CPU Identifiers

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

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

```
shell> 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:

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

- Preconfigure the MySQL server package with the following command:

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

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

  > **Important**
  >
  > Make sure you remember the root password you set. Users who want to set a password later can leave the **password** field blank in the dialogue box and just press **OK**; 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:

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

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

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

  Here are where the files are installed on the system:

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

  > **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 Installing MySQL on Linux from the Native Software Repositories

Many Linux distributions include a version of the MySQL server, client tools, and development components in their native software repositories and can be installed with the platforms' standard package management systems. This section provides basic instructions for installing MySQL using those package management systems.

Important

Native packages are often several versions behind the currently available release. You will also normally be unable to install development milestone releases (DMRs), as these are not usually made available in the native repositories. Before proceeding, we recommend that you check out the other installation options described in Chapter 2, *Installing MySQL on Linux*.

Distribution specific instructions are shown below:

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

  Note

  For EL5, EL6, or EL7-based Linux platforms and Fedora 23 or 24, you can install MySQL using the MySQL Yum repository instead of the platform's native software repository. See Section 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:

```
root-shell> yum install mysql mysql-server mysql-libs mysql-server
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package mysql.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-libs.x86_64 0:5.1.48-2.fc13 set to be updated
---> Package mysql-server.x86_64 0:5.1.48-2.fc13 set to be updated
--> Processing Dependency: perl-DBD-MySQL for package: mysql-server-5.1.48-2.fc13.x86_64
--> Running transaction check
---> Package perl-DBD-MySQL.x86_64 0:4.017-1.fc13 set to be updated
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package               Arch          Version               Repository      Size
================================================================================
Installing:
 mysql                 x86_64        5.1.48-2.fc13         updates        889 k
 mysql-libs            x86_64        5.1.48-2.fc13         updates        1.2 M
 mysql-server          x86_64        5.1.48-2.fc13         updates        8.1 M
Installing for dependencies:
 perl-DBD-MySQL        x86_64        4.017-1.fc13          updates        136 k
Transaction Summary
================================================================================
```

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

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

```
root-shell> service mysqld start
```

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

```
root-shell> chkconfig --levels 235 mysqld on
```

Which enables the MySQL server to be started (and stopped) automatically at the specified the run levels.

The database tables will have been automatically created for you, if they do not already exist. You should, however, run `mysql_secure_installation` to set the root passwords on your server.

- **Debian, Ubuntu, Kubuntu**

  > **Note**
  >
  > For Debian 7 and 8, and Ubuntu 12, 14, and 15, MySQL can be installed using the MySQL APT Repository instead of the platform's native software repository. See Section 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):

```
root-shell> apt-get install mysql-client-5.1 mysql-server-5.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.28-11 linux-headers-2.6.28-11-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-common postfix
Suggested packages:
  dbishell libipc-sharedcache-perl tinyca procmail postfix-mysql postfix-pgsql
  postfix-ldap postfix-pcre sasl2-bin resolvconf postfix-cdb
The following NEW packages will be installed
  bsd-mailx libdbd-mysql-perl libdbi-perl libhtml-template-perl
  libmysqlclient15off libmysqlclient16 libnet-daemon-perl libplrpc-perl mailx
  mysql-client-5.1 mysql-common mysql-server-5.1 postfix
0 upgraded, 13 newly installed, 0 to remove and 182 not upgraded.
Need to get 1907kB/25.3MB of archives.
After this operation, 59.5MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get: 1 http://gb.archive.ubuntu.com jaunty-updates/main mysql-common 5.1.30really5.0.75-0ubuntu10.5 [63.6kB]
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: 5.7.18 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 will install a number of packages, including the MySQL server, in order to provide the typical tools and application environment. This can mean that you install a large number of packages in addition to the main MySQL package.

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

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

```
root-shell> service mysql [start|stop]
```

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

2.8 Installing MySQL on Linux with docker

The docker deployment framework supports easy installation and configuration of MySQL servers. For instructions, see https://hub.docker.com/r/mysql/mysql-server/. This page also provides extensive documentation about using MySQL under docker.

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

As of MySQL 5.7.6, if you install MySQL using an RPM distribution on the following Linux platforms, server startup and shutdown is managed by systemd:

- Red Hat Enterprise Linux 7; Oracle Linux 7; CentOS 7
- SUSE Linux Enterprise Server 12
- Fedora 23 and 24

To obtain systemd support if you install from a source distribution, configure the distribution using the `-DWITH_SYSTEMD=1` `CMake` option. See MySQL Source-Configuration Options.

The following discussion covers these topics:

- Overview of systemd
- Configuring MySQL Using systemd
- Configuring Multiple MySQL Instances Using systemd
- Migrating from mysqld_safe to systemd

Overview of systemd

systemd provides automatic 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}
```

For the `systemctl` or `service` commands, if the MySQL service name is not `mysqld`, use the appropriate name (for example, `mysql` on SLES systems).

Support for systemd includes these files:

- `mysqld.service`: systemd service unit configuration, with details about the `mysqld` service.
- `mysqld@.service`: Like `mysqld.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`: Support script for the unit file. This script assists in creating the error log file only if its location matches the pattern /var/log/mysql*.log. 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.

On platforms for which systemd support is installed, scripts such as `mysqld_safe` and the System V initialization script are not installed because they are unnecessary. 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 using an application-specific program.

As of MySQL 5.7.13, on platforms for which systemd support is installed, systemd has the capability of managing multiple MySQL instances. For details, see Configuring Multiple MySQL Instances Using systemd. Consequently, `mysqld_multi` and `mysqld_multi.server` are not installed because they are unnecessary.

Configuring MySQL Using systemd

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
```

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
```

Support for configuration using `override.conf` was added in MySQL 5.7.7.

With systemd, the `override.conf` configuration method must be used for certain parameters, rather than settings in a `[mysqld_safe]` or `[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 will be ignored.
  - Before MySQL 5.7.10: Use `PIDFile` in `override.conf` rather than the `--pid-file` option for `mysqld_safe` or `mysqld`. 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` option for `mysqld_safe` or `mysqld`.
- 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 the value of environment values that should be in effect for 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
```

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

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

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

```
systemctl restart mysqld
```

Configuring Multiple MySQL Instances Using systemd

As of MySQL 5.7.13, on platforms for which systemd support is installed, systemd has the capability of managing multiple MySQL instances. Consequently, `mysqld_multi` and `mysqld_multi.server` are not installed because they are unnecessary.

To use multiple-instance capability, modify `my.cnf` to include configuration of key options for each instance. For example, to manage two instances named `replica01` and `replica02`, add something like this to the file:

```
[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
```

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`). In that 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@mysql1
```

systemd starts the server using a command such as this:

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

The result is that the `[server]`, `[mysqld]`, and `[mysqld@mysql1]` option groups are read and used for that instance of the service.

Migrating from mysqld_safe to systemd

Because `mysqld_safe` is not installed when systemd is used, options previously specified for that program (for example, in an `[mysqld_safe]` option group) must be specified another way:

- Some `mysqld_safe` options are also understood by `mysqld` and can be moved from the `[mysqld_safe]` option group to the `[mysqld]` group. This does *not* include `--pid-file` or `--open-files-limit`. To specify those options, use the `override.conf` systemd file, described previously.
- For some `mysqld_safe` options, there are similar `mysqld` options. For example, the `mysqld_safe` option for enabling `syslog` logging is `--syslog`. 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 and OpenSolaris

Table of Contents

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

- For information on installing using the native Solaris PKG format, see Section 3.1, “Installing MySQL on Solaris Using a Solaris PKG”.
- On OpenSolaris, the standard package repositories include MySQL packages specially built for OpenSolaris that include entries for the Service Management Framework (SMF) to enable control of the installation using the SMF administration commands. For more information, see Section 3.2, “Installing MySQL on OpenSolaris Using IPS”.
- 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.

To obtain a binary MySQL distribution for Solaris in tarball or PKG format, http://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, you may run into trouble even before you get the MySQL distribution unpacked, as the Solaris `tar` cannot handle long file names. This means that you may see errors when you try to unpack MySQL.

  If this occurs, you must use GNU `tar` (`gtar`) to unpack the distribution. In Solaris 10 and OpenSolaris `gtar` is normally located in `/usr/sfw/bin/gtar`, but may not be included in the default path definition.

- When using Solaris 10 for x86_64, you should mount any file systems on which you intend to store `InnoDB` files with the `forcedirectio` option. (By default mounting is done without this option.) Failing to do so will cause a significant drop in performance when using the `InnoDB` storage engine on this platform.

- If you would like MySQL to start automatically, you can copy `support-files/mysql.server` to `/etc/init.d` and create a symbolic link to it named `/etc/rc3.d/S99mysql.server`.

- If too many processes try to connect very rapidly to `mysqld`, you should see this error in the MySQL log:

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

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

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

3.1 Installing MySQL on Solaris Using a Solaris PKG

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

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

```
shell> gunzip mysql-5.7.18-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:

```
shell> pkgadd -d mysql-5.7.18-solaris11-x86_64.pkg
The following packages are available:
  1  mysql     MySQL Community Server (GPL)
               (i86pc) 5.7.18
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:
```

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

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

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

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

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

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

```
shell> mysqladmin shutdown
shell> pkgrm mysql
shell> pkgadd -d mysql-5.7.18-solaris11-x86_64.pkg
shell> mysqld_safe &
shell> mysql_upgrade
```

You should check the notes in Upgrading or Downgrading MySQL before performing any upgrade.

3.2 Installing MySQL on OpenSolaris Using IPS

OpenSolaris includes standard packages for MySQL in the core repository. The MySQL packages are based on a specific release of MySQL and updated periodically. For the latest release you must use either the native Solaris PKG, `tar`, or source installations. The native OpenSolaris packages include SMF files so that you can easily control your MySQL installation, including automatic startup and recovery, using the native service management tools.

To install MySQL on OpenSolaris, use the `pkg` command. You will need to be logged in as root, or use the `pfexec` tool, as shown in the example below:

```
shell> pfexec pkg install SUNWmysql57
```

The package set installs three individual packages, `SUNWmysql57lib`, which contains the MySQL client libraries; `SUNWmysql57r` which contains the root components, including SMF and configuration files; and `SUNWmysql57u` which contains the scripts, binary tools and other files. You can install these packages individually if you only need the corresponding components.

The MySQL files are installed into `/usr/mysql` which symbolic links for the sub directories (`bin`, `lib`, etc.) to a version specific directory. For MySQL 5.7, the full installation is located in `/usr/mysql/5.7`. The default data directory is `/var/mysql/5.7/data`. The configuration file is installed in `/etc/mysql/5.7/my.cnf`. This layout permits multiple versions of MySQL to be installed, without overwriting the data and binaries from other versions.

Once installed, you must initialize the data directory (see Chapter 5, *Initializing the Data Directory*), and use the `mysql_secure_installation` to secure your installation.

Using SMF to manage your MySQL installation

Once installed, you can start and stop your MySQL server using the installed SMF configuration. The service name is `mysql`, or if you have multiple versions installed, you should use the full version name, for example `mysql:version_57`. To start and enable MySQL to be started at boot time:

```
shell> svcadm enable mysql
```

To view the SMF logs, use this command:

```
shell> svcadm enable svc:/application/database/mysql
```

To check whether the MySQL service is running:

shell> **`svcs -xv svc:/application/database/mysql`**

To disable MySQL from starting during boot time, and shut the MySQL server down if it is running:

```
shell> svcadm disable mysql
```

To restart MySQL, for example after a configuration file changes, use the `restart` option:

```
shell> svcadm restart mysql
```

You can also use SMF to configure the data directory and enable full 64-bit mode. For example, to set the data directory used by MySQL:

```
shell> svccfg
svc:> select mysql:version_57
svc:/application/database/mysql:version_57> setprop mysql/data=/data0/mysql
```

By default, the 32-bit binaries are used. To enable the 64-bit server on 64-bit platforms, set the `enable_64bit` parameter. For example:

```
svc:/application/database/mysql:version_57> setprop mysql/enable_64bit=1
```

You must refresh the SMF after setting these options:

```
shell> svcadm refresh mysql
```

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 MySQL Program Environment Variables.

Chapter 5 Initializing the Data Directory

Table of Contents

After installing MySQL, you must initialize the data directory, including the tables in the `mysql` system database. For some MySQL installation methods, data directory initialization may be done automatically, as described in Postinstallation Setup and Testing. For other installation methods, including installation from generic binary and source distributions, you must initialize the data directory yourself.

This section describes how to initialize the data directory on Unix and Unix-like systems. (For Windows, see Windows Postinstallation Procedures.) For some suggested commands that you can use to test whether the server is accessible and working properly, see Testing the Server.

In the examples shown here, the server runs under the user ID of the `mysql` login account. This assumes that such an account exists. Either create the account if it does not exist, or substitute the name of a different existing login account that you plan to use for running the server. For information about creating the account, see Creating a `mysql` System User and Group, in Chapter 1, *Installing MySQL on Unix/Linux Using Generic Binaries*.

1. Change location into the top-level directory of your MySQL installation, represented here by *`BASEDIR`*:

   ```
   shell> cd BASEDIR
   ```

   `BASEDIR` is likely to be something like `/usr/local/mysql` or `/usr/local`. The following steps assume that you have changed location to this directory.

   You will find several files and subdirectories in the *`BASEDIR`* directory. The most important for installation purposes are the `bin` and `scripts` subdirectories, which contain the server as well as client and utility programs.

2. Create a directory that provides a location to use as the value of the `secure_file_priv` system variable that limits import/export operations to a specific directory. See Server System Variables.

   ```
   shell> mkdir mysql-files
   shell> chmod 750 mysql-files
   ```

3. If necessary, ensure that the distribution contents are accessible to `mysql`. If you installed the distribution as `mysql`, no further action is required. If you installed the distribution as `root`, its contents will be owned by `root`. Change its ownership to `mysql` by executing the following commands as `root` in the installation directory. The first command changes the owner attribute of the files to the `mysql` user. The second changes the group attribute to the `mysql` group.

   ```
   shell> chown -R mysql .
   shell> chgrp -R mysql .
   ```

4. If necessary, initialize the data directory, including the `mysql` database containing the initial MySQL grant tables that determine how users are permitted to connect to the server.

   Typically, data directory initialization need be done only the first time you install MySQL. If you are upgrading an existing installation, you should run `mysql_upgrade` instead (see `mysql_upgrade` —

Check and Upgrade MySQL Tables). However, the command that initializes the data directory does not overwrite any existing privilege tables, so it should be safe to run in any circumstances.

As of MySQL 5.7.6, use the server to initialize the data directory:

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

Before MySQL 5.7.6, use `mysql_install_db`:

```
shell> bin/mysql_install_db --user=mysql
```

For more information, see Section 5.1, “Initializing the Data Directory Manually Using mysqld”, or Section 5.2, “Initializing the Data Directory Manually Using mysql_install_db”, depending on which command you use.

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

   ```
   shell> mysql_ssl_rsa_setup
   ```

   For more information, see `mysql_ssl_rsa_setup` — Create SSL/RSA Files.

6. After initializing the data directory, you can establish the final installation ownership settings. To leave the installation owned by `mysql`, no action is required here. Otherwise, most of the MySQL installation can be owned by `root` if you like. The exception is that the data directory and the `mysql-files` directory must be owned by `mysql`. To accomplish this, run the following commands as `root` in the installation directory. For some distribution types, the data directory might be named `var` rather than `data`; adjust the second command accordingly.

   ```
   shell> chown -R root .
   shell> chown -R mysql data mysql-files
   ```

   If the plugin directory (the directory named by the `plugin_dir` system variable) is writable by the server, it may be possible for a user to write executable code to a file in the directory using `SELECT ... INTO DUMPFILE`. This can be prevented by making the plugin directory read only to the server or by setting the `secure_file_priv` system variable at server startup to a directory where `SELECT` writes can be performed safely. (For example, set it to the `mysql-files` directory created earlier.)

7. To specify options that the MySQL server should use at startup, put them in a `/etc/my.cnf` or `/etc/mysql/my.cnf` file. You can use such a file, for example, to set the `secure_file_priv` system variable. See Server Configuration Defaults. If you do not do this, the server starts with its default settings.

8. If you want MySQL to start automatically when you boot your machine, see Starting and Stopping MySQL Automatically.

Data directory initialization creates time zone tables in the `mysql` database but does not populate them. To do so, use the instructions in MySQL Server Time Zone Support.

5.1 Initializing the Data Directory Manually Using mysqld

This section describes how to initialize the data directory using `mysqld`, the MySQL server.

> **Note**
>
> The procedure described here is available for all platforms as of MySQL 5.7.6. Prior to 5.7.6, use `mysql_install_db` on Unix and Unix-like systems (see Section 5.2, "Initializing the Data Directory Manually Using mysql_install_db"). Prior to MySQL 5.7.7, Windows distributions include a data directory with prebuilt tables in the `mysql` database.

The following instructions assume that your current location is the MySQL installation directory, represented here by *`BASEDIR`*:

```
shell> cd BASEDIR
```

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.

On Windows, use one of these commands:

```
C:\> bin\mysqld --initialize
C:\> bin\mysqld --initialize-insecure
```

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

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

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

Regardless of platform, 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 will need to choose a new one. With the `--initialize-insecure` option, no `root` password is generated; it is assumed that you will assign a password to the account in timely fashion before putting the server into production use.

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

```
shell> 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):

```
shell> 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 (the `--defaults-file` option must be first):

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

When invoked with the `--initialize` or `--initialize-insecure` option, `mysqld` performs the following initialization sequence.

> **Note**
>
> The server writes any messages 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.

1. The server checks for the existence of the data directory as follows:

   - If no data directory exists, the server creates it.
   - If a data directory exists and 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.

2. Within the data directory, the server creates the `mysql` system database and its tables, including the grant tables, server-side help tables, and time zone tables. For a complete listing and description of the grant tables, see The MySQL Access Privilege System.

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, changes to some tablespace characteristics require setting up a whole new instance. This includes 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. The server's action with respect to a password for this account depends 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.
     ```

5. The server populates the server-side help tables if content is available (in the `fill_help_tables.sql` file). The server does not populate the time zone tables; to do so, see MySQL Server Time Zone Support.

6. If the `--init-file` option 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.

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` using the random password that the server generated during the initialization sequence:

     ```
     shell> mysql -u root -p
     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:

```
shell> mysql -u root --skip-password
```

3. After connecting, assign a new `root` password:

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

> **Note**
>
> The data directory initialization sequence performed by the server does not substitute for the actions performed by `mysql_secure_installation` or `mysql_ssl_rsa_setup`. See `mysql_secure_installation` — Improve MySQL Installation Security, and `mysql_ssl_rsa_setup` — Create SSL/RSA Files.

5.2 Initializing the Data Directory Manually Using mysql_install_db

This section describes how to initialize the data directory using `mysql_install_db`.

> **Note**
>
> The procedure described here is used on Unix and Unix-like systems prior to MySQL 5.7.6. (For Windows, MySQL distributions include a data directory with prebuilt tables in the `mysql` database.) As of MySQL 5.7.6, `mysql_install_db` is deprecated. To initialize the data directory, use the procedure described at Section 5.1, “Initializing the Data Directory Manually Using mysqld”.

The following instructions assume that your current location is the MySQL installation directory, represented here by *`BASEDIR`*:

```
shell> cd BASEDIR
```

To initialize the data directory, invoke `mysql_install_db`. This program might be located under the base directory in either `bin` or `scripts`, depending on your version of MySQL. If it is in `scripts`, adjust the following commands appropriately.

```
shell> bin/mysql_install_db --user=mysql
```

It is important to make sure that the database directories and files are owned by the `mysql` login account so that the server has read and write access to them when you run it later. To ensure this, run `mysql_install_db` as `root` and include the `--user` option as shown. Otherwise, execute the program while logged in as `mysql`, in which case you can omit the `--user` option from the command.

The `mysql_install_db` command creates the server's data directory. Under the data directory, it creates directories for the `mysql` database that holds the grant tables and (prior to MySQL 5.7.4) a `test` database that you can use to test MySQL. The program also creates privilege table entries for the initial account or accounts. For a complete listing and description of the grant tables, see The MySQL Access Privilege System.

It might be necessary to specify other options such as `--basedir` or `--datadir` if `mysql_install_db` does not identify the correct locations for the installation directory or data directory. For example:

```
```

```
shell> bin/mysql_install_db --user=mysql \
         --basedir=/opt/mysql/mysql \
         --datadir=/opt/mysql/mysql/data
```

If `mysql_install_db` generates a random password for the `root` account, start the server and assign a new password:

1. Start the server (use the first command if your installation includes `mysqld_safe`, the second it if includes systemd support):

   ```
   shell> bin/mysqld_safe --user=mysql &
   shell> systemctl start mysqld
   ```

   Substitute the appropriate service name if it differs from `mysqld`; for example, `mysql` on SLES systems.

2. Look in the `$HOME/.mysql_secret` file to find the random password that `mysql_install_db` wrote there. Then connect to the server as `root` using that password:

   ```
   shell> mysql -u root -h 127.0.0.1 -p
   Enter password: (enter the random password here)
   ```

3. After connecting, assign a new `root` password:

   ```
   mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
   ```

   After resetting the password, remove the `.mysql_secret` file; otherwise, if you run `mysql_secure_installation`, that command may see the file and expire the `root` password again as part of ensuring secure deployment.

If `mysql_install_db` did not generate a random password, you should still assign one. For instructions, see Securing the Initial MySQL Accounts. That section also describes how to remove the `test` database, if `mysql_install_db` created one and you do not want it.

If you have trouble with `mysql_install_db` at this point, see Section 5.3, “Problems Running mysql_install_db”.

5.3 Problems Running mysql_install_db

The purpose of the `mysql_install_db` program is to initialize the data directory, including the tables in the `mysql` system database. It does not overwrite existing MySQL privilege tables, and it does not affect any other data.

To re-create your privilege tables, first stop the `mysqld` server if it is running. Then rename the `mysql` directory under the data directory to save it, and run `mysql_install_db`. Suppose that your current directory is the MySQL installation directory and that `mysql_install_db` is located in the `bin` directory and the data directory is named `data`. To rename the `mysql` database and re-run `mysql_install_db`, use these commands.

```
shell> mv data/mysql data/mysql.old
shell> bin/mysql_install_db --user=mysql
```

When you run `mysql_install_db`, you might encounter the following problems:

- **`mysql_install_db` fails to install the grant tables**

You may find that `mysql_install_db` fails to install the grant tables and terminates after displaying the following messages:

```
Starting mysqld daemon with databases from XXXXXX
mysqld ended
```

In this case, you should examine the error log file very carefully. The log should be located in the directory `XXXXXX` named by the error message and should indicate why `mysqld` did not start. If you do not understand what happened, include the log when you post a bug report. See How to Report Bugs or Problems.

- **There is a `mysqld` process running**

  This indicates that the server is running, in which case the grant tables have probably been created already. If so, there is no need to run `mysql_install_db` at all because it needs to be run only once, when you first install MySQL.

- **Installing a second `mysqld` server does not work when one server is running**

  This can happen when you have an existing MySQL installation, but want to put a new installation in a different location. For example, you might have a production installation, but you want to create a second installation for testing purposes. Generally the problem that occurs when you try to run a second server is that it tries to use a network interface that is in use by the first server. In this case, you should see one of the following error messages:

  ```
  Can't start server: Bind on TCP/IP port:
  Address already in use
  Can't start server: Bind on unix socket...
  ```

  For instructions on setting up multiple servers, see Running Multiple MySQL Instances on One Machine.

- **You do not have write access to the `/tmp` directory**

  If you do not have write access to create temporary files or a Unix socket file in the default location (the `/tmp` directory) or the `TMPDIR` environment variable, if it has been set, an error occurs when you run `mysql_install_db` or the `mysqld` server.

  You can specify different locations for the temporary directory and Unix socket file by executing these commands prior to starting `mysql_install_db` or `mysqld`, where *`some_tmp_dir`* is the full path name to some directory for which you have write permission:

  ```
  shell> TMPDIR=/some_tmp_dir/
  shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
  shell> export TMPDIR MYSQL_UNIX_PORT
  ```

  Then you should be able to run `mysql_install_db` and start the server with these commands:

  ```
  shell> bin/mysql_install_db --user=mysql
  shell> bin/mysqld_safe --user=mysql &
  ```

  See How to Protect or Change the MySQL Unix Socket File, and MySQL Program Environment Variables.

There are some alternatives to running the `mysql_install_db` program provided in the MySQL distribution:

- If you want the initial privileges to be different from the standard defaults, use account-management statements such as `CREATE USER`, `GRANT`, and `REVOKE` to change the privileges *after* the grant tables have been set up. In other words, run `mysql_install_db`, and then use `mysql -u root mysql` to connect to the server as the MySQL `root` user so that you can issue the necessary statements. (See Account Management Statements.)

  To install MySQL on several machines with the same privileges, put the `CREATE USER`, `GRANT`, and `REVOKE` statements in a file and execute the file as a script using `mysql` after running `mysql_install_db`. For example:

  ```
  shell> bin/mysql_install_db --user=mysql
  shell> bin/mysql -u root < your_script_file
  ```

  This enables you to avoid issuing the statements manually on each machine.

- It is possible to re-create the grant tables completely after they have previously been created. You might want to do this if you are just learning how to use `CREATE USER`, `GRANT`, and `REVOKE` and have made so many modifications after running `mysql_install_db` that you want to wipe out the tables and start over.

  To re-create the grant tables, stop the server if it is running and remove the `mysql` database directory. Then run `mysql_install_db` again.