Chapter 1. Installing MySQL

Let's begin our learning path by installing MySQL and accessing it for the first time.

Note that we do not rely on a single version of MySQL for this book. Instead, we have drawn on our collective knowledge of MySQL in the real world. The book's core is focused on Linux operating systems (mostly Ubuntu/Debian and CentOS/RHEL or its derivatives) and on MySQL 5.7 and MySQL 8.0, because those are what we consider the "current" versions capable of production workloads. The MySQL 5.7 and 8.0 series are still under development, which means that newer versions with bug fixes and new features will continue to be released.

With MySQL becoming the most popular open source database (Oracle, which ranks first, is not open source), the demand for having a fast installation process has increased. You can think of installing MySQL from scratch as similar to baking a cake: the source code is the recipe. But even with the source code available, the recipe for building software is not easy to follow. It takes time to compile, and usually, it is necessary to install additional development libraries that expose production environments to risk. Say you want a chocolate cake; even if you have the instructions for how to make it yourself, you may not want to mess up your kitchen, or you may not have the time to bake it, so you go to a bakery and buy one instead. For MySQL, when you want it ready to use without the effort involved in compiling it, you can use the *distribution packages*.

Distribution packages for MySQL are available for diverse platforms, including Linux distributions, Windows, and macOS. These packages provide a flexible and fast way to start using MySQL. Returning to the chocolate cake example, suppose you want to change something. Maybe you want a white chocolate cake. For MySQL, we have what are called forks, which include some different options. We'll look at a few of these in the next section.

MySQL Forks

In software engineering, a *fork* occurs when someone copies the source code and starts their own path of independent development and support. The fork can follow a track close to that of the original version, as the Percona distribution of MySQL does, or drift away, like MariaDB. Because the MySQL source code is open and free, new projects can fork the code without permission from its original creator. Let's take a look at a few of the most notable forks.

MySQL Community Edition

MySQL Community Edition, also known as the *upstream* or *vanilla* version of MySQL, is the open source version distributed by Oracle. This version drives the development of the InnoDB engine and new features, and it is the first one to receive updates, new features, and bug fixes.

Percona Server for MySQL

The Percona distribution of MySQL is a free, open source, drop-in replacement for MySQL Community Edition. The development closely follows that version, focusing on improving performance and the overall MySQL ecosystem. Percona Server also includes additional enhancements like the MyRocks engine, an Audit Log plugin, and a PAM Authentication plugin. Percona was cofounded by Peter Zaitsev and Vadim Tkachenko.

MariaDB Server

Created by Michael "Monty" Widenius and distributed by the MariaDB Foundation, MariaDB Server is by far the fork that has drifted the furthest away from vanilla MySQL. In recent years it has developed new features and engines such as MariaDB ColumnStore, and it was the first database to integrate Galera 4 clustering functionality.

MySQL Enterprise Edition

MySQL Enterprise Edition is currently the only version with a commercial license (which means you need to pay to use it, like a Windows license). Also

distributed by Oracle, it contains all the functionality of the Community Edition plus exclusive features for security, backup, and high availability.

Installation Choices and Platforms

First, you must choose the MySQL version compatible with your operating system (OS). You can verify compatibility with the MySQL website. The same support policies are available for Percona Server and MariaDB.

We often hear the question: is it possible to install MySQL on an OS that is not supported? Most of the time, the answer is yes. It is possible to install MySQL on Windows 7, for example, but the risks of hitting a bug or facing unpredictable behavior (like memory leaks or underperformance) are high. Because of these risks, we do not recommend doing this for production environments.

The next step is to decide whether to install a *development* or *General Availability* (GA) release. Development releases have the newest features, but we do not recommend them for production because they are not stable. GA releases, also called *production* or *stable* releases, are meant for production use.

TIP

We highly recommend using the most recent GA release because this will include the latest stable bug fixes and performance improvements.

The last thing to decide is which distribution format to install for the operating system. For most use cases, a binary distribution fits. Binary distributions are available in native format for many platforms, such as *.rpm* packages for Linux or *.dmg* packages for macOS. The distributions are also available in generic formats, such as *.zip* archives or compressed *.tar* files (*tarballs*). On Windows, you can use the MySQL Installer to install a binary distribution.

WARNING

Watch out for whether the version is 32-bit or 64-bit. The rule of thumb is to pick the 64-bit version. Unless you are working with an ancient OS, you should *not* select the 32-bit version. This is because 32-bit processors can handle only a limited amount of RAM (4 GB or less), whereas 64-bit processors are capable of addressing much more memory.

The installation process consists of four major steps, outlined in the following sections. It's essential to follow these correctly and to set the minimum security requirements for the MySQL database.

1. Download the Distribution that You Want to Install

Each distribution has its owner and, by consequence, its source. Some Linux distributions provide default packages in their repositories. For example, on CentOS 8, the MySQL vanilla distribution is available from the default repositories. When the OS has default packages available, it is unnecessary to download MySQL from a website or configure a repository yourself, which facilitates the installation process.

We will demonstrate how to install the repositories and download the files without the need to go to the website during the installation process. However, if you do want to download MySQL yourself, you can use the following links:

- MySQL Community Server
- Percona Server for MySQL
- MariaDB Server

2. Install the Distribution

Installing consists of the elementary steps to make MySQL functional and bring it online, but not securing MySQL. For example, at this point, the MySQL root user can connect without a password, which is quite hazardous since the root user has privileges to perform every action, including dropping a database.

3. Perform Any Necessary Post-Installation Setup

This step is about making sure the MySQL server is working correctly. It is essential to make sure that your server is secure, and the first step for this is executing the *mysql_secure_installation* script. You'll change the password for the root user, disable access for the root user from a remote server, and remove the test database.

4. Run Benchmarks

Some DBAs run benchmarks for each deployment to measure whether the performance is suitable for the project they are using it for. The most common tool for this is `sysbench`. It's essential to highlight here that `sysbench` performs a *synthetic workload*, whereas when the application is running, we call it the *real workload*. Synthetic workloads usually provide reports about the maximum server performance, but they can't reproduce the real-world workload (with its inherent locks, different query execution times, stored procedures, triggers, and so on).

In the next section we'll walk through the details of the installation process for a few of the most commonly used platforms.

Installing MySQL on Linux

The Linux ecosystem is diverse and has many variants, including Red Hat Enterprise Linux (RHEL), CentOS, Ubuntu, Debian, and others. This section focuses on only the most popular ones—otherwise, this book would be entirely about the installation process!

Installing MySQL on CentOS 7

CentOS, short for Community Enterprise Linux Operating System, was founded in 2004, and Red Hat acquired it in 2014. CentOS is the community version of Red Hat, so they're pretty much identical, but CentOS is free, and support comes from the community instead of Red Hat itself. CentOS 7 was released in 2014, and its end-of-life date is in 2024.

Installing MySQL 8.0

To install MySQL 8.0 on CentOS 7 using the *yum* repository, complete the following steps.

Log in to Linux server

Usually, for security reasons, users log into Linux servers as nonprivileged users. Here is an example of a user logging into Linux from a macOS terminal using a private key:

```
$ ssh -i key.pem centos@3.227.11.227
```

After you've successfully connected, you'll see something like this in the terminal:

```
[centos@ip-172-30-150-91 ~]$
```

Become root in Linux

Once you're connected to the server, you need to become root:

```
$ sudo su - root
```

You will then see a prompt like the following in your terminal:

```
[root@ip-172-30-150-91 ~]#
```

Becoming root is important because to install MySQL it is necessary to perform tasks such as creating the MySQL user in Linux, configuring directories, and setting permissions. It is also possible to use the `sudo` command for all examples we will show that should be executed by the root user. However, if you forget to prefix a command with `sudo`, the installation process will be incomplete.

NOTE

This chapter will use the Linux root user in the majority of the examples (represented by the prompt `#` in the code lines). Another advantage of the `#` representation is that this is also the comment character in Linux. If you blindly copy/paste lines from the book, you won't run any real commands in the shell.

Configure the yum repository

Execute the following command to configure the MySQL *yum* repository:

```
# rpm -Uvh https://repo.mysql.com/mysql80-community-rel
```

Install MySQL 8.0 Community Server

Because the MySQL *yum* repository has repositories for multiple MySQL versions (5.7 and 8.0 major versions), first we have to disable all repositories:

```
# sed -i &#39;s/enabled=1/enabled=0/&#39;
/etc/yum.repos.d/mysql-community.repo
```

Next, we need to enable the MySQL 8.0 repository and execute the following command to install MySQL 8.0:

```
# yum --enablerepo=mysql80-community install mysql-comm
```

Start the MySQL service

Now, start the MySQL service with the `systemctl` command:

```
# systemctl start mysqld
```

It is also possible to start the MySQL process manually, which can be useful to troubleshoot initialization problems when MySQL is refusing to start. To start manually, indicate the location of the *my.cnf* file and which user can manipulate the database files and the process:

```
# mysqld --defaults-file=/etc/my.cnf --user=mysql
```

Discover the default password for the root user

When you install MySQL 8.0, MySQL creates a temporary password for the root user account. To identify the password of the root user account, execute the following command:

```
# grep "A temporary password" /var/log/mysqld.log
```

The command provides output like the following:

```
2020-05-31T15:04:12.256877Z 6 [Note] [MY-010454] [Serve
password is generated for root@localhost: #z?hhCCyj2aj
```

Secure the MySQL installation

MySQL provides a shell script that you can run on Unix systems, *mysql_secure_installation*, that enables you to improve the security of your server installation in the following ways:

- You can set a password for the root account.
- You can disable root access from outside the localhost.
- You can remove anonymous user accounts.
- You can remove the test database, which by default can be accessed by anonymous users.

Execute the command `mysql_secure_installation` to secure the MySQL server:

```
# mysql_secure_installation
```

It will prompt you for the current password of the root account:

```
Enter the password for user root:
```

Enter the temporary password obtained in the previous step and press Enter. The following message will appear:

```
The existing password for the user account root has exp
set a new password.

New password:
Re-enter new password:
```

NOTE

This section will cover only the basics of changing the root password to grant access to the MySQL server. We will show more details about granting privileges and creating a password policy in Chapter 8.

You will need to enter the new password for the root account twice. More recent MySQL versions come with a validation policy, which means that the new password needs to respect minimal requirements to be accepted. The default requirements are that passwords must be at least eight characters long and include:

- At least one numeric character
- At least one lowercase character
- At least one uppercase character
- At least one special (nonalphanumeric) character

Next, it will prompt you with some yes/no questions about whether you want to make some initial setup changes. To ensure maximum protection, we recommend removing anonymous users, disabling remote root login, and removing the test database (i.e., answering *yes* for all options):

```
Remove anonymous users? (Press y|Y for Yes, any other k

Disallow root login remotely? (Press y|Y for Yes, any c

Remove test database and access to it? (Press y|Y for Y
for No) : y
```

```
Reload privilege tables now? (Press y|Y for Yes, any ot
```

Connect to MySQL

This step is optional, but we use it to verify that we executed all the steps correctly. Use this command to connect to the MySQL server:

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

It will prompt for the password of the root user. Type the password and press Enter:

```
Enter password:
```

If successful, it will show the MySQL command line:

```
mysql>
```

Start MySQL 8.0 upon server start (optional)

To set MySQL to start whenever the server boots up, use the following command:

```
# systemctl enable mysqld
```

Installing MariaDB 10.5

To install MariaDB 10.5 on CentOS 7, you'll need to execute similar steps as for the vanilla MySQL distribution.

Become root in Linux

First, we need to become root. See the instructions in "Installing MySQL 8.0".

Install the MariaDB repository

The following set of commands will download the MariaDB repo and configure it for the next step. Note that in the `yum` commands, we are using the `-y` option. This option tells Linux to assume the answer is *yes* for all subsequent questions:

```
# yum install wget -y
# wget https://downloads.mariadb.com/MariaDB/mariadb_re
# chmod +x mariadb_repo_setup
# ./mariadb_repo_setup
```

Install MariaDB

With the repository configured, the next command will install the latest stable version of MariaDB and its dependencies:

```
# yum install MariaDB-server -y
```

The end of the output will be similar to this:

```
Installed:
  MariaDB-compat.x86_64 0:10.5.8-1.el7.centos

Dependency Installed:
  MariaDB-client.x86_64 0:10.5.8-1.el7.centos MariaDB-c
  0:10.5.8-1.el7.centos boost-program-options.x86_64 0:
  galera-4.x86_64 0:26.4.6-1.el7.centos        libaio.x
  0:0.3.109-13.el7               lsof.x86_64 0:4.87-6.e
  pcre2.x86_64 0:10.23-2.el7                  perl.x86_
  4:5.16.3-299.el7_9             perl-Carp.noarch 0:1.
  ...

Replaced:
  mariadb-libs.x86_64 1:5.5.64-1.el7

Complete!
```

The *Complete!* at the end of the log indicates a successful installation.

Start MariaDB

With MariaDB installed, initialize the service with the `systemctl` command:

```
# systemctl start mariadb.service
```

You can use this command to verify its status:

```
# systemctl status mariadb
```

```
  mariadb.service - MariaDB 10.5.8 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.serv
   vendor preset: disabled)
...
Feb 07 12:55:04 ip-172-30-150-91.ec2.internal systemd[1
MariaDB 10.5.8 database server.
```

Secure MariaDB

At this point, MariaDB will be running in insecure mode. In contrast to MySQL 8.0, MariaDB will have an empty root password so you can access it instantly:

```
# mysql
```

```
Welcome to the MariaDB monitor.  Commands end with ; or
Your MariaDB connection id is 44
Server version: 10.5.8-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation A

Type 'help;' or '\h' for help. Type '\c' to clear the c
statement.

MariaDB [(none)]>
```

You can execute `mysql_secure_installation` to secure MariaDB just like you would for MySQL 8.0 (see the previous section for details). There is a slight variation in output, with one extra question:

```
Switch to unix_socket authentication [Y/n] y
Enabled successfully!
Reloading privilege tables..
 ... Success!
```

Answering *yes* changes the connection from TCP/IP to Unix socket mode. We will discuss the different connection types in "MySQL 5.7 Default Files".

Installing Percona Server 8.0

Install Percona Server 8.0 on CentOS 7 using the following step.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the Percona repository

You can install the Percona *yum* repository by running the following command as root or with `sudo` :

```
# yum install https://repo.percona.com/yum/percona-rele
```

The installation creates a new repository file, */etc/yum.repos.d/percona-original-release.repo*. Now, enable the Percona Server 8.0 repository using this command:

```
# percona-release setup ps80
```

Install Percona Server 8.0

To install the server, execute this command:

```
# yum install percona-server-server
```

Initialize Percona Server 8.0 with systemctl

Once you've installed the Percona Server 8.0 binaries, start the service:

```
# systemctl start mysql
```

And validate its status:

```
# systemctl status mysql
```

```
 mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.servi
   vendor preset: disabled)
   Active: active (running) since Sun 2021-02-07 13:22:
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-sys
  Process: 14472 ExecStartPre=/usr/bin/mysqld_pre_syste
  status=0/SUCCESS)
 Main PID: 14501 (mysqld)
   Status: "Server is operational"
    Tasks: 39 (limit: 5789)
   Memory: 345.2M
   CGroup: /system.slice/mysqld.service
           └─14501 /usr/sbin/mysqld

Feb 07 13:22:14 ip-172-30-92-109.ec2.internal systemd[1
MySQL Server...
Feb 07 13:22:15 ip-172-30-92-109.ec2.internal systemd[1
Server.
```

At this point, the steps are similar to the vanilla installation. Refer to the sections on obtaining the temporary password and executing the `mysql_secure_installation` command in "Installing MySQL 8.0".

Installing MySQL 5.7

Install MySQL 5.7 on CentOS 7 using the following steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the MySQL 5.7 repository

You can install the MySQL 5.7 *yum* repository by running the following command as root or with `sudo`:

```
# yum localinstall\
    https://dev.mysql.com/get/mysql57-community-release
```

The installation creates a new repository file, */etc/yum.repos.d/mysql-community.repo*.

Install the MySQL 5.7 binaries

To install the server, execute this command:

```
# yum install mysql-community-server -y
```

Initialize MySQL 5.7 with systemctl

Once you've installed the MySQL 5.7 binaries, start the service:

```
# systemctl start mysqld
```

And run this command to validate its status:

```
# systemctl status mysqld
```

At this point, the steps are similar to the MySQL 8.0 vanilla installation. Refer to the sections on obtaining the temporary password and executing the `mysql_secure_installation` command in "Installing MySQL 8.0".

Installing Percona Server 5.7

Install Percona Server 5.7 on CentOS 7 using the following steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the Percona repository

You can install the Percona *yum* repository by running the following command as root or with `sudo` :

```
# yum install https://repo.percona.com/yum/percona-rele
```

The installation creates a new repository file, */etc/yum.repos.d/percona-original-release.repo*. Use this command to enable the Percona Server 5.7 repository:

```
# percona-release setup ps57
```

Install the Percona Server 5.7 binaries

To install the server, execute this command:

```
# yum install Percona-Server-server-57 -y
```

Initialize Percona Server 5.7 with systemctl

Once you've installed the Percona Server 5.7 binaries, start the service:

```
# systemctl start mysql
```

And validate its status:

```
# systemctl status mysql
```

At this point, the steps are similar to the MySQL 8.0 vanilla installation. Refer to the sections on obtaining the temporary password and executing the `mysql_secure_installation` command in "Installing MySQL 8.0".

Installing MySQL on CentOS 8

The current version of CentOS is CentOS 8, and it is built on top of RHEL 8. Typically, CentOS enjoys the same ten-year support lifecycle as RHEL itself. This traditional support lifecycle would give CentOS 8 an end-of-life date in 2029. However, in December 2020, a Red Hat announcement signaled the intention to put a headstone on CentOS 8's grave much sooner—in 2021. (Red Hat will support CentOS 7 alongside RHEL 7 through 2024.) Current CentOS users will need to migrate either to RHEL itself or to the newer CentOS Stream project. Some community projects are arising, but at this point, the future of CentOS is uncertain.

However, we will share the installation steps here since many users are using RHEL 8 and Oracle Linux 8 in the industry.

Installing MySQL 8.0

The latest MySQL 8.0 version is available to install from the default AppStream repository using the MySQL module that the CentOS 8 and RHEL 8 systems enable by default. So, there is some variation from the traditional `yum` method. Let's take a look at the details.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the MySQL 8.0 binaries

Run the following command to install the `mysql-server` package and a number of its dependencies:

```
# dnf install mysql-server
```

When prompted, press **y** and then Enter to confirm that you want to proceed:

```
Output
...
Transaction Summary
============================================================
Install  50 Packages
Upgrade   8 Packages

Total download size: 50 M
Is this ok [y/N]: y
```

Start MySQL

At this point, you've installed MySQL on your server, but it isn't yet operational. The package you installed configures MySQL to run as a `systemd` service named `mysqld.service`. To start MySQL, you need to use the `systemctl` command:

```
# systemctl start mysqld.service
```

Check if the service is running

To check if the service is running correctly, run the following command:

```
# systemctl status mysqld
```

If you successfully started MySQL, the output will show that the MySQL service is active:

```
# systemctl status mysqld
```

```
mysqld.service - MySQL 8.0 database server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.servi
   vendor preset: disabled)
   Active: active (running) since Sun 2020-06-21 22:57:
  Process: 15966 ExecStartPost=/usr/libexec/mysql-check
  (code=exited, status=0/SUCCESS)
  Process: 15887 ExecStartPre=/usr/libexec/mysql-prepar
  mysqld.service (code=exited, status=0/SUCCESS)
  Process: 15862 ExecStartPre=/usr/libexec/mysql-check-
```

```
  (code=exited, status=0/SUCCESS)
 Main PID: 15924 (mysqld)
   Status: "Server is operational"
    Tasks: 39 (limit: 23864)
   Memory: 373.7M
   CGroup: /system.slice/mysqld.service
           └─15924 /usr/libexec/mysqld --basedir=/usr

Jun 21 22:57:57 ip-172-30-222-117.ec2.internal systemd[
MySQL 8.0 database server...
Jun 21 22:57:57 ip-172-30-222-117.ec2.internal systemd[
MySQL 8.0 database server.
```

Secure MySQL 8.0

As with installing MySQL 8.0 on CentOS 7, you need to execute the `mysql_secure_installation` command (see the relevant section in "Installing MySQL 8.0" for details). The main difference is that there is *not* a temporary password for CentOS 8, so when the script requests the root password, leave it blank and press Enter.

Start MySQL 8.0 upon server start (optional)

To set MySQL to start whenever the server boots up, use the following command:

```
# systemctl enable mysqld
```

Installing Percona Server 8.0

To install Percona Server 8.0 on CentOS 8, you need to install the repository first. Let's walk through the steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the Percona Server 8.0 binaries

Run the following command to install the Percona repository:

```
# yum install https://repo.percona.com/yum/percona-rele
```

When prompted, press **y** and then Enter to confirm that you want to proceed:

```
Last metadata expiration check: 0:03:49 ago on Sun 07 F
percona-release-latest.noarch.rpm
Dependencies resolved.

<snip>

Total size: 19 k
Installed size: 31 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  1/1
  Installing       : percona-release-1.0-25.noarch
  1/1
  Running scriptlet: percona-release-1.0-25.noarch
  1/1
* Enabling the Percona Original repository
<*> All done!
* Enabling the Percona Release repository
<*> All done!
The percona-release package now contains a percona-rele
can enable additional repositories for our newer produc
enable the Percona Server 8.0 repository use:

  percona-release setup ps80

Note: To avoid conflicts with older product versions, t
command may disable our original repository for some pr
information, please visit:
```

```
  https://www.percona.com/doc/percona-repo-config/percc

  Verifying: percona-release-1.0-25.noarch 1/1

Installed:
  percona-release-1.0-25.noarch
```

Enable the repository for Percona 8.0

The installation creates a new repository file in */etc/yum.repos.d/percona-original-release.repo*. Enable the Percona Server 8.0 repository using this command:

```
# percona-release setup ps80
```

The command prompts you to disable the RHEL 8 module for MySQL. You can do this now by pressing **y**:

```
* Disabling all Percona Repositories
On RedHat 8 systems it is needed to disable dnf mysql m
Percona-Server
Do you want to disable it? [y/N] y
Disabling dnf module...
Percona Release release/noarch YUM repository
6.4 kB/s | 1.4 kB     00:00
Dependencies resolved.

<snip>

Complete!
dnf mysql module was disabled
* Enabling the Percona Server 8.0 repository
* Enabling the Percona Tools repository
<*> All done!
```

Or do it manually with the following command:

```
# dnf module disable mysql
```

Install the Percona Server 8.0 binaries

You're now ready to install Percona Server 8.0 on your CentOS 8/RHEL 8 server. To avoid being prompted again about whether you want to proceed, add the `-y` to the command line:

```
# yum install percona-server-server -y
```

Start and secure Percona Server 8.0

Now that you've installed the Percona Server 8.0 binaries, you can start the `mysqld` service and set it to start at system boot:

```
# systemctl enable --now mysqld
# systemctl start mysqld
```

Check the service status

It is important to validate that you've completed all the steps successfully. Use this command to check the status of the service:

```
# systemctl status mysqld
```

```
mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.servi
   vendor preset: disabled)
   Active: active (running) since Sun 2021-02-07 01:30:
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-sys
  Process: 12864 ExecStartPre=/usr/bin/mysqld_pre_syste
  status=0/SUCCESS)
 Main PID: 12942 (mysqld)
   Status: "Server is operational"
    Tasks: 39 (limit: 5789)
   Memory: 442.6M
   CGroup: /system.slice/mysqld.service
           └─12942 /usr/sbin/mysqld
```

```
Feb 07 01:30:40 ip-172-30-92-109.ec2.internal systemd[1
Feb 07 01:30:50 ip-172-30-92-109.ec2.internal systemd[1
```

TIP

If you ever want to disable the option of MySQL starting up at boot, you can do so by running the following command:

```
# systemctl disable mysqld
```

Installing MySQL 5.7

Install MySQL 5.7 on CentOS 8 using the following steps.

Become root in Linux

First, you need to become root. See the instructions in “Installing MySQL 8.0”.

Disable the MySQL default module

Systems such as RHEL 8, Oracle Linux 8, and CentOS 8 enable the MySQL module by default. Unless this module is disabled, it masks packages provided by MySQL repositories, preventing you from installing a version different than MySQL 8.0. So, use these commands to remove this default module:

```
# dnf remove @mysql
# dnf module reset mysql && dnf module disable mysql
```

Configure the MySQL 5.7 repository

There is no MySQL repository for CentOS 8, so we’ll use the CentOS 7 repository instead as a reference. Create a new repository file:

```
# vi /etc/yum.repos.d/mysql-community.repo
```

And paste the following data into the file:

```
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/e
enabled=1
gpgcheck=0

[mysql-connectors-community]
name=MySQL Connectors Community
baseurl=http://repo.mysql.com/yum/mysql-connectors-comm
enabled=1
gpgcheck=0

[mysql-tools-community]
name=MySQL Tools Community
baseurl=http://repo.mysql.com/yum/mysql-tools-community
enabled=1
gpgcheck=0
```

Install the MySQL 5.7 binaries

With the default module disabled and the repository configured, run the following command to install the `mysql-server` package and its dependencies:

```
# dnf install mysql-community-server
```

When prompted, press **y** and then Enter to confirm that you want to proceed:

```
Output
...
Install  5 Packages

Total download size: 202 M
Installed size: 877 M
Is this ok [y/N]: y
```

Start MySQL

You've installed the MySQL binaries on your server, but it isn't yet operational. The package you installed configures MySQL to run as a

`systemd` service named `mysqld.service`. To start MySQL, you need to use the `systemctl` command:

```
# systemctl start mysqld.service
```

Check if the service is running

To check that the service is running correctly, run the following command:

```
# systemctl status mysqld
```

If you successfully started MySQL, the output will show that the MySQL service is active:

```
# systemctl status mysqld
```

```
mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.servi
   vendor preset: disabled)
   Active: active (running) since Sun 2021-02-07 18:22:
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-sys
  Process: 14396 ExecStart=/usr/sbin/mysqld --daemonize
  --pid-file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS
  (code=exited, status=0/SUCCESS)
  Process: 8137 ExecStartPre=/usr/bin/mysqld_pre_system
  status=0/SUCCESS)
 Main PID: 14399 (mysqld)
    Tasks: 27 (limit: 5789)
   Memory: 327.2M
   CGroup: /system.slice/mysqld.service
           └─14399 /usr/sbin/mysqld --daemonize
           --pid-file=/var/run/mysqld/mysqld.pid

Feb 07 18:22:02 ip-172-30-36-53.ec2.internal systemd[1]
Feb 07 18:22:12 ip-172-30-36-53.ec2.internal systemd[1]
```

Secure MySQL 5.7

At this point, the steps are similar to the MySQL 8.0 vanilla installation. Refer to the sections on obtaining the temporary password and executing the `mysql_secure_installation` command in "Installing MySQL 8.0".

Start MySQL 5.7 upon server start (optional)

To set MySQL to start whenever the server boots up, use the following command:

```
# systemctl enable mysqld
```

Installing MySQL on Ubuntu 20.04 LTS (Focal Fossa)

Ubuntu is a Linux distribution based on Debian that is composed mostly of free and open source software. Officially, there are three Ubuntu editions: Desktop, Server, and Core for IoT devices and robots. The version we will work with in this book is the Server version.

Installing MySQL 8.0

For Ubuntu, the process is slightly different since Ubuntu uses the *apt* repository. Let's walk through the steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Configure the apt repository

On Ubuntu 20.04 (Focal Fossa), you can install MySQL using the *apt* package repository. First, ensure that your system is up-to-date:

```
# apt update
```

Install MySQL 8.0

Next, install the `mysql-server` package:

```
# apt install mysql-server -y
```

The `apt install` command will install MySQL but won't prompt you to set a password or make any other configuration changes. Unlike the CentOS installation, Ubuntu initializes MySQL in *insecure mode*.

For fresh installations of MySQL, you'll want to run the database management system's (DBMS's) included security script. This script changes some of the less secure default options for remote root logins and the test database. We will address this problem in the securing step after initializing MySQL.

Start MySQL

At this point, you've installed MySQL on your server, but it isn't yet operational. To start MySQL, you need to use the `systemctl` command:

```
# systemctl start mysql
```

Check if the service is running

To check that the service is running correctly, run the following command:

```
# systemctl status mysql
```

If you successfully started MySQL, the output will show that the MySQL service is active:

```
mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service;
     vendor preset: enabled)
     Active: active (running) since Sun 2021-02-07 20:1
    Process: 3514 ExecStartPre=/usr/share/mysql/mysql-s
    (code=exited, status=0/SUCCESS)
   Main PID: 3522 (mysqld)
```

```
     Status: "Server is operational"
      Tasks: 38 (limit: 1164)
     Memory: 332.7M
     CGroup: /system.slice/mysql.service
             └─3522 /usr/sbin/mysqld

Feb 07 20:19:50 ip-172-30-202-86 systemd[1]: Starting M
Feb 07 20:19:51 ip-172-30-202-86 systemd[1]: Started My
```

Secure MySQL 8.0

At this point, the steps are similar to the vanilla installation on CentOS 7 (see "Installing MySQL 8.0"). However, MySQL 8.0 on Ubuntu is initialized unsecured, which means the root password is empty. To secure it, execute `mysql_secure_installation` :

```
# mysql_secure_installation
```

This will take you through a series of prompts to make some changes to the MySQL installation's security options, which are similar to those of the CentOS version as described earlier.

There is a small variance here because in Ubuntu it is possible to change the validation policy, which manages password strength. In this example, we are setting the validation policy to MEDIUM (1):

```
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwor
and improve security. It checks the strength of passwor
and allows the users to set only those passwords which
secure enough. Would you like to setup VALIDATE PASSWOR

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special ch
```

```
STRONG Length >= 8, numeric, mixed case, special charac

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 50
Do you wish to continue with the password provided?(Pre
key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to hav
a user account created for them. This is intended only
testing, and to make the installation go a bit smoother
You should remove them before moving into a production
environment.
```

Installing Percona Server 8

Install Percona Server 8.0 on Ubuntu 20.04 LTS using the following steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Install the GNU Privacy Guard

Oracle signs MySQL downloadable packages with GNU Privacy Guard (GnuPG), an open source alternative to the well-known Pretty Good Privacy (PGP) created by Phil Zimmermann. Most Linux distributions ship with GnuPG installed by default, but in this case you need to install it:

```
# apt-get install gnupg2 -y
```

Fetch the repository packages from the Percona website

Next, fetch the repository packages from the Percona repository with the `wget` command:

```
# wget https://repo.percona.com/apt/percona-release_lat
    _all.deb
```

Install the downloaded package with dpkg

Once downloaded, install the package with the following command:

```
# dpkg -i percona-release_latest.$(lsb_release -sc)_all
```

You can then check the repository configured in the */etc/apt/sources.list.d/percona-original-release.list* file.

Enable the repository

The next step is enabling Percona Server 8.0 in the repository and refreshing it:

```
# percona-release setup ps80
# apt update
```

Install the Percona Server 8.0 binaries

Then, install the `percona-server-server` package with the `apt-get install` command:

```
# apt-get install percona-server-server -y
```

Start MySQL

At this point, you've installed MySQL on your server, but it isn't yet operational. To start MySQL, you need to use the `systemctl` command:

```
# systemctl start mysql
```

Check if the service is running

To check that the service is running correctly, run the following command:

```
# systemctl status mysql
```

At this point, Percona Server will be running in insecure mode. Executing `mysql_secure_installation` will take you through a series of prompts to make some changes to your MySQL installation's security options, which are identical to those described for installing vanilla MySQL 8.0 in the previous section.

Installing MariaDB 10.5

Install MariaDB 10.5 on Ubuntu 20.04 LTS using the following steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Update the system with the apt package manager

Ensure your system is up-to-date and install the `software-properties-common` package with the following commands:

```
# apt update && sudo apt upgrade
# apt -y install software-properties-common
```

This package contains the common files for software properties like the D-Bus backend and an abstraction of the used *apt* repositories.

Import the MariaDB GPG key

Run the following command to add the repository key to the system:

```
# apt-key adv --fetch-keys \
    'https://mariadb.org/mariadb_release_signing_key.as
```

Add the MariaDB repository

After importing the repository GPG key, you need to add the *apt* repository by running the following command:

```
# add-apt-repository \
    'deb [arch=amd64] http://mariadb.mirror.globo.tech/
```

NOTE

There are different mirrors to download the MariaDB repository. In this example, we use *http://mariadb.mirror.globo.tech*.

Install the MariaDB 10.5 binaries

The next step is the installation of the MariaDB Server:

```
# apt install mariadb-server mariadb-client
```

Check if the service is running

To check if the MariaDB service is running correctly, run the following command:

```
# systemctl status mysql
```

At this point, MariaDB 10.5 will be running in insecure mode. Executing `mysql_secure_installation` will take you through a series of prompts to make some changes to your MySQL installation's security options, which are identical to those described for installing vanilla MySQL 8.0 on Ubuntu earlier in this section.

Installing MySQL 5.7

Install MySQL 5.7 on Ubuntu 20.04 LTS using the following steps.

Become root in Linux

First, you need to become root. See the instructions in "Installing MySQL 8.0".

Update the system with the apt package manager

You can ensure your system is updated and install the `software-properties-common` package with the following command:

```
# apt update -y && sudo apt upgrade -y
```

Add and configure the MySQL 5.7 repository

Add the MySQL repository by running the following commands:

```
# wget https://dev.mysql.com/get/mysql-apt-config_0.8.1
# dpkg -i mysql-apt-config_0.8.12-1_all.deb
```

At the prompt, choose "ubuntu bionic" as shown in Figure 1-1 and click OK.

Figure 1-1. Choose “ubuntu bionic”

The next prompt shows MySQL 8.0 chosen by default (Figure 1-2). With this option selected, press Enter.

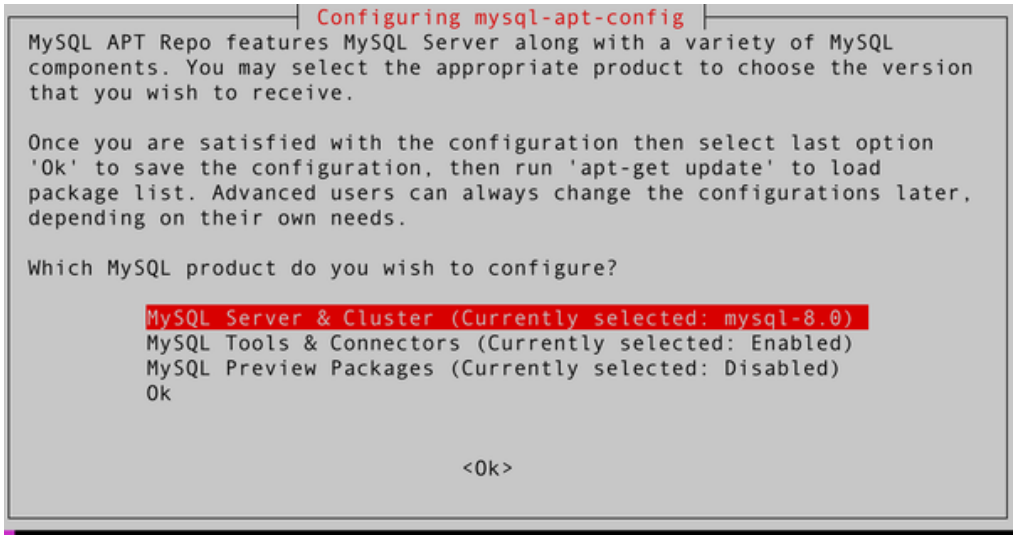

Figure 1-2. Choose the MySQL Server & Cluster option

For the next option, as shown in Figure 1-3, choose MySQL 5.7 and click OK.

Figure 1-3. Choose the MySQL 5.7 option

After returning to the main screen, click OK to exit, as shown in Figure 1-4.

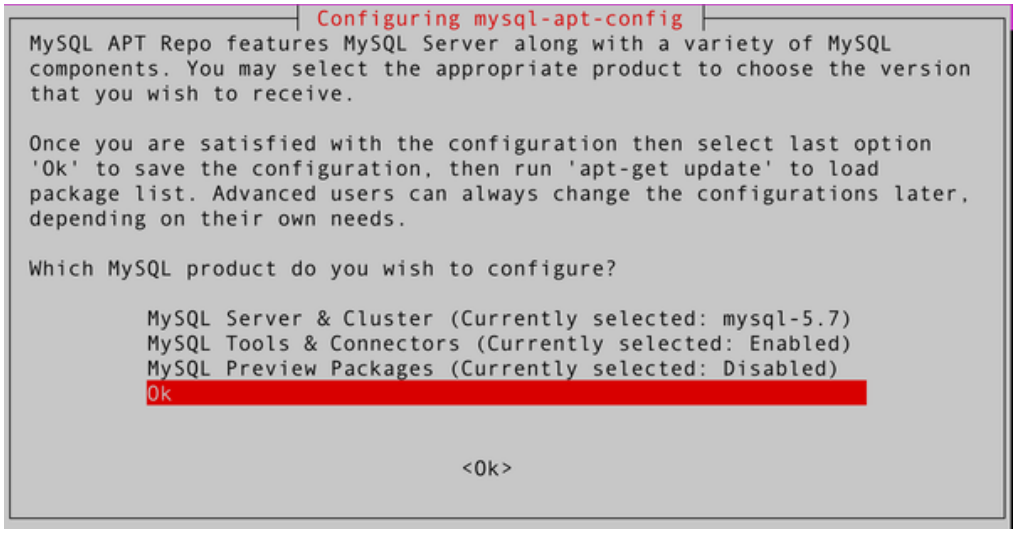

Figure 1-4. Click OK to exit

Next, you need to update the MySQL packages:

```
# apt-get update -y
```

Validate the Ubuntu policy to install MySQL 5.7:

```
# apt-cache policy mysql-server
```

Check the output to see which MySQL 5.7 version is available:

```
# apt-cache policy mysql-server
```

```
mysql-server:
  Installed: (none)
```

```
Candidate: 8.0.23-0ubuntu0.20.04.1
Version table:
   8.0.23-0ubuntu0.20.04.1 500
      500 http://br.archive.ubuntu.com/ubuntu focal-u
      500 http://br.archive.ubuntu.com/ubuntu focal-s
      Packages
   8.0.19-0ubuntu5 500
      500 http://br.archive.ubuntu.com/ubuntu focal/m
   5.7.33-1ubuntu18.04 500
      500 http://repo.mysql.com/apt/ubuntu bionic/mys
```

Install the MySQL 5.7 binaries

Now that you've verified that the MySQL 5.7 version is available (*5.7.33-1ubuntu18.04*), install it:

```
# apt-get install mysql-client=5.7.33-1ubuntu18.04 -y
# apt-get install mysql-community-server=5.7.33-1ubuntu
# apt-get install mysql-server=5.7.33-1ubuntu18.04 -y
```

The installation process will prompt you to choose the root password, as shown in Figure 1-5.

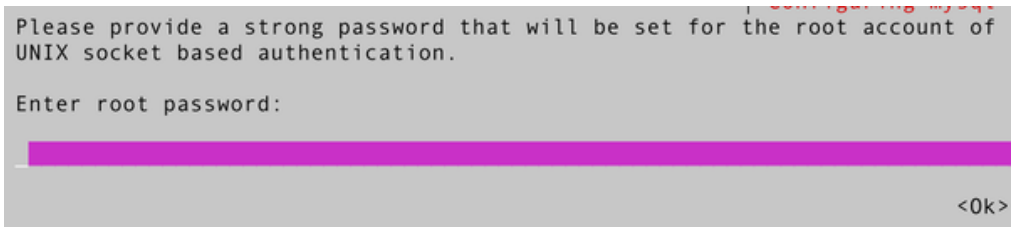

Figure 1-5. Define the root password and click OK

Check if the service is running

To check if the MySQL 5.7 service is running correctly, run the following command:

```
# systemctl status mysql
```

At this point, MySQL 5.7 will have a password set for the root user. However, you'll still want to run `mysql_secure_installation` to set the password

policy, remove remote root login and anonymous users, and remove the test database. Refer "Secure MySQL 8.0" for details.

Installing MySQL on macOS Big Sur

MySQL for macOS is available in a few different forms. Since most of the time MySQL is installed on macOS for development purposes, we will demonstrate only how to install it using the native macOS installer (the *.dmg* file). Be aware that it is also possible to use the tarball to install MySQL on macOS.

Installing MySQL 8

First, download the MySQL *.dmg* file from the MySQL website.

TIP

According to Oracle, macOS Catalina packages work for Big Sur.

Once downloaded, execute the package to start the install procedure, as shown in Figure 1-6.

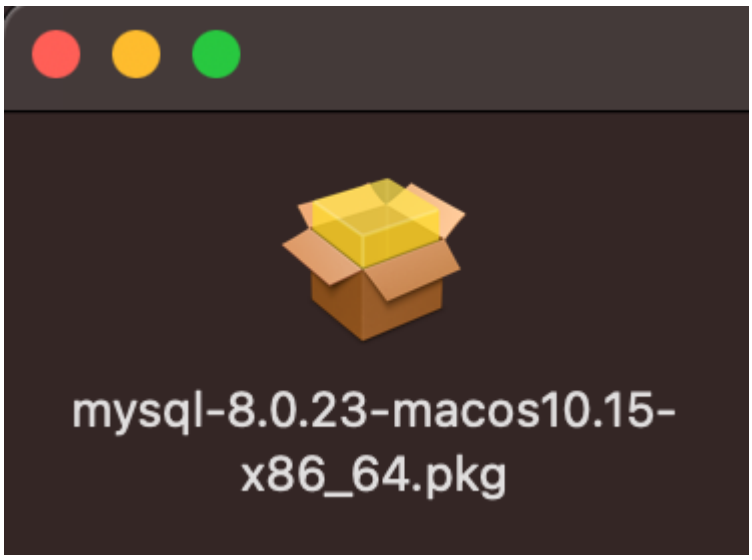

Figure 1-6. MySQL 8.0.23 *.dmg* package

Next, you need to authorize MySQL to run, as shown in Figure 1-7.

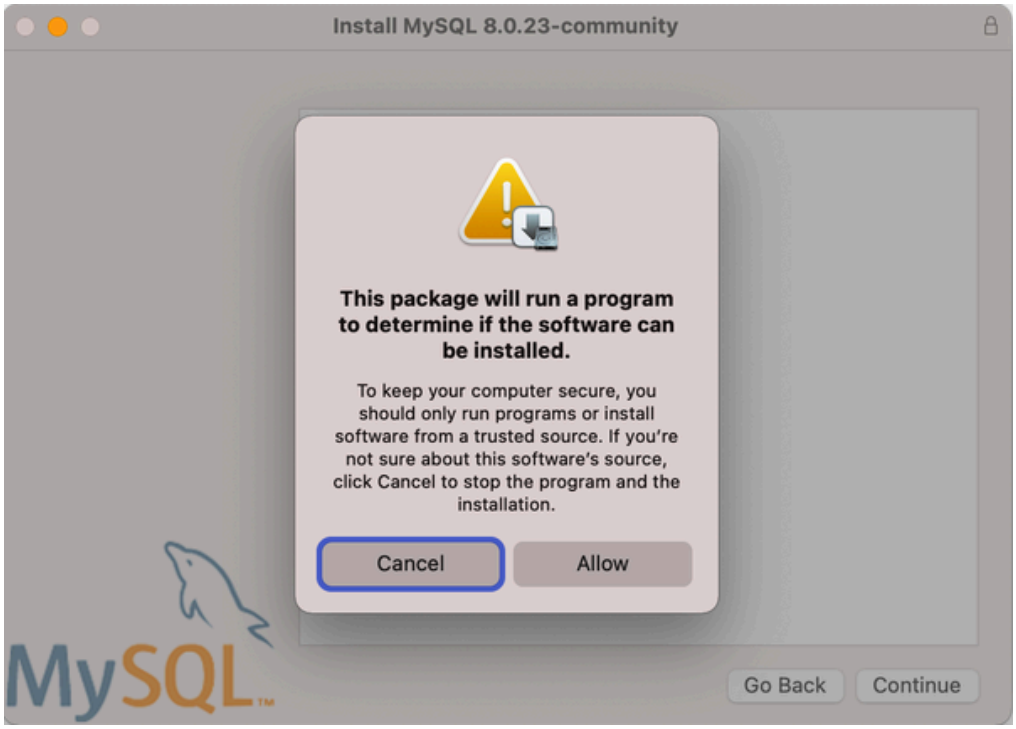

Figure 1-7. MySQL 8.0.23 authorization request

Figure 1-8 shows the installer's welcome screen.

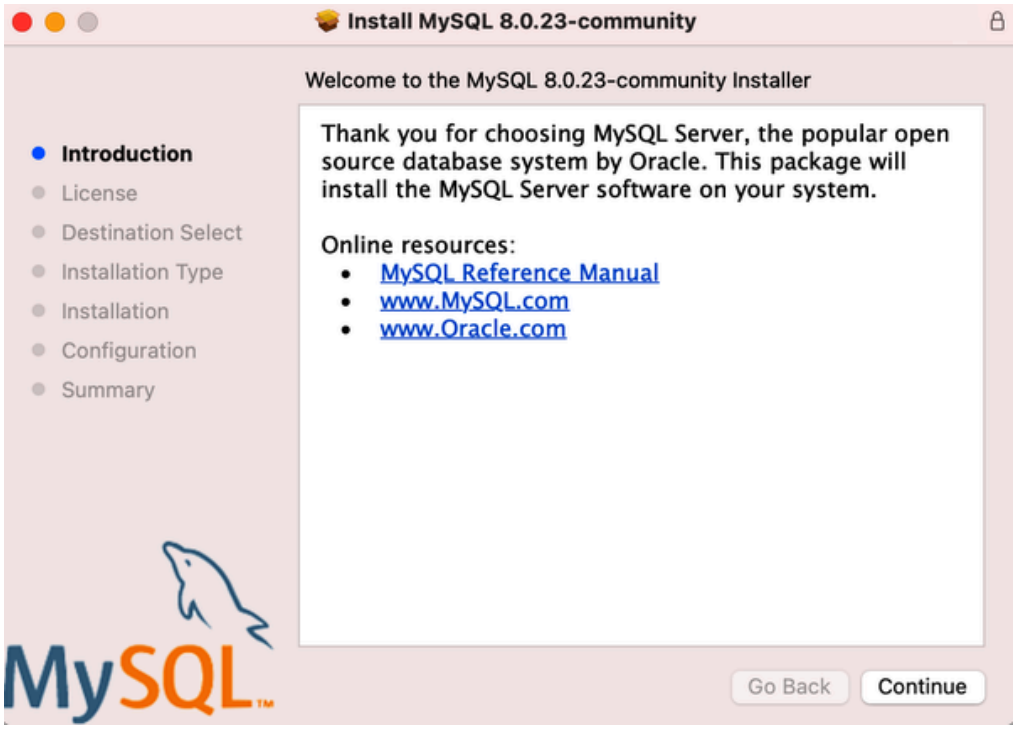

Figure 1-8. MySQL 8.0.23 initial screen

Figure 1-9 shows the license agreement. Even with open source software, it is necessary to agree to the license terms; otherwise, you can't proceed.

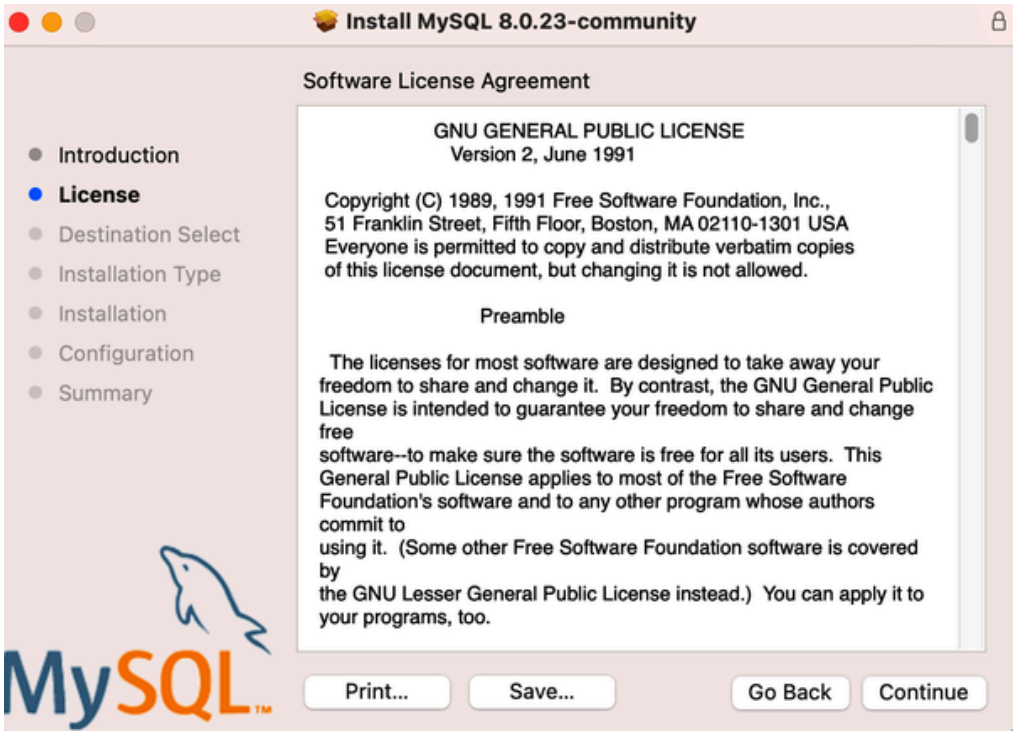

Figure 1-9. MySQL 8.0.23 license agreement

Now you can define the location and customize the installation, as shown in Figure 1-10.

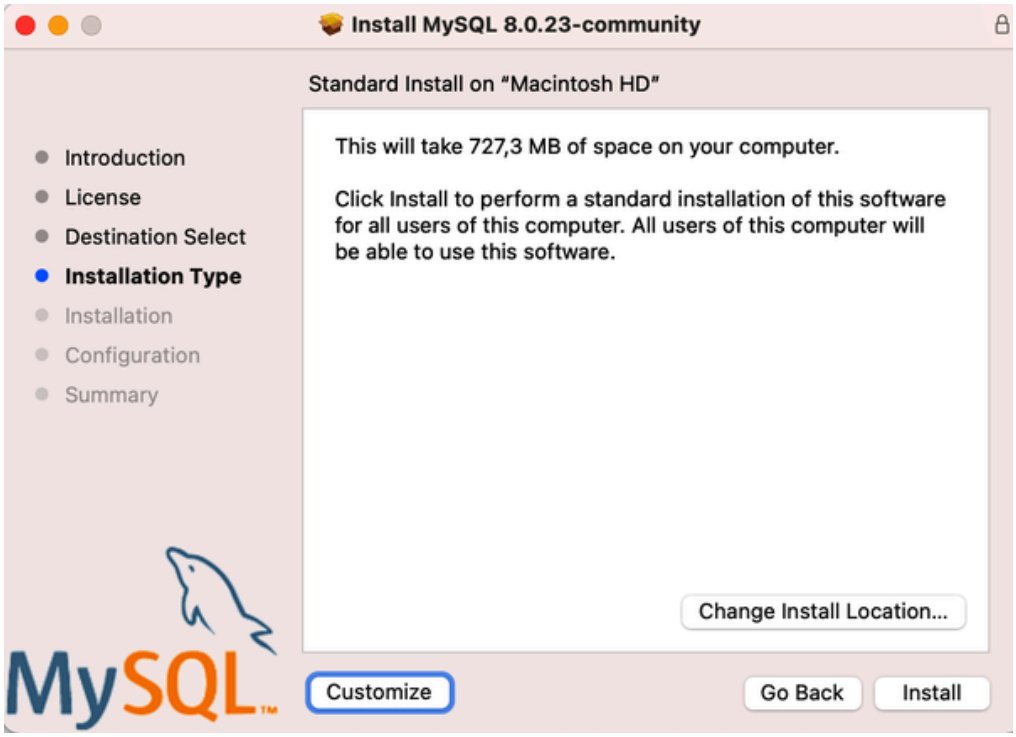

Figure 1-10. MySQL 8.0.23 installation customization

You are going to proceed with the standard installation. After clicking Install, you might get prompted to enter the macOS user password to run the installation with higher privileges, as Figure 1-11 shows.

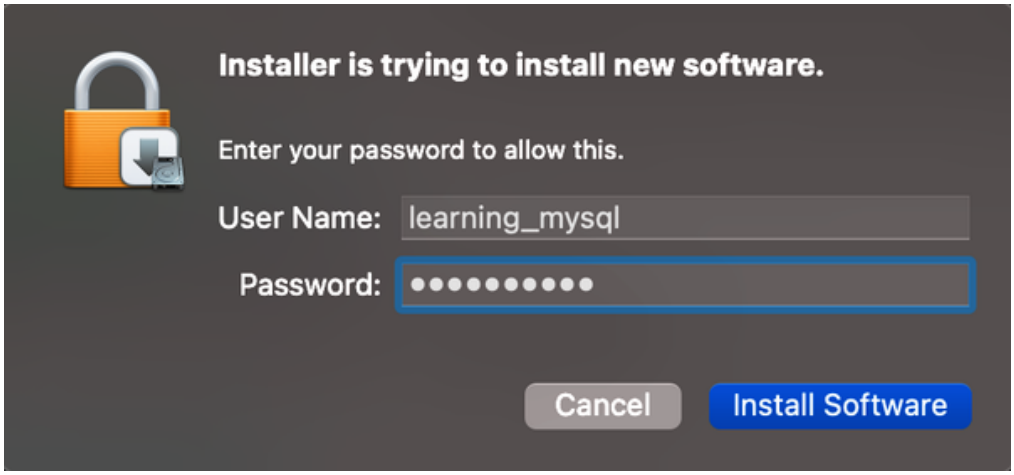

Figure 1-11. macOS authorization request

Once you've installed MySQL, the installation process will prompt you to choose the *password encryption*. You should use the newer authentication method (the default option), as shown in Figure 1-12, which is safer.

Figure 1-12. MySQL 8.0.23 password encryption

The last step consists of creating the root password and initializing MySQL, as shown in Figure 1-13.

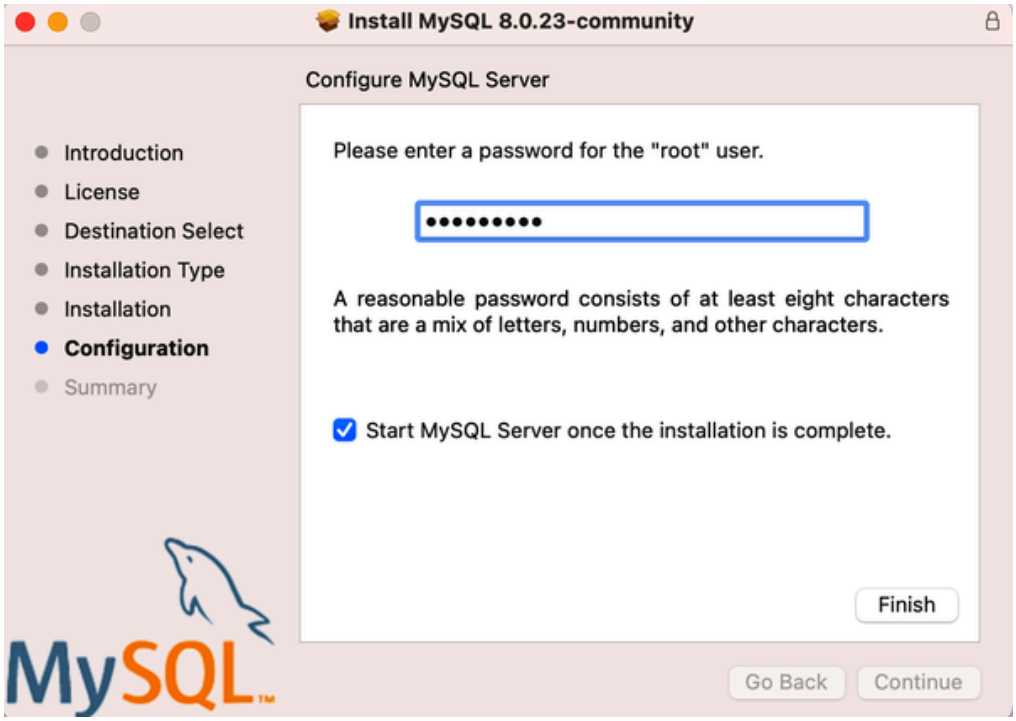

Figure 1-13. MySQL 8.0.23 root password

You've now installed MySQL Server, but it is not loaded (or started) by default. To start, open System Preferences and search for the MySQL icon, as shown in Figure 1-14.

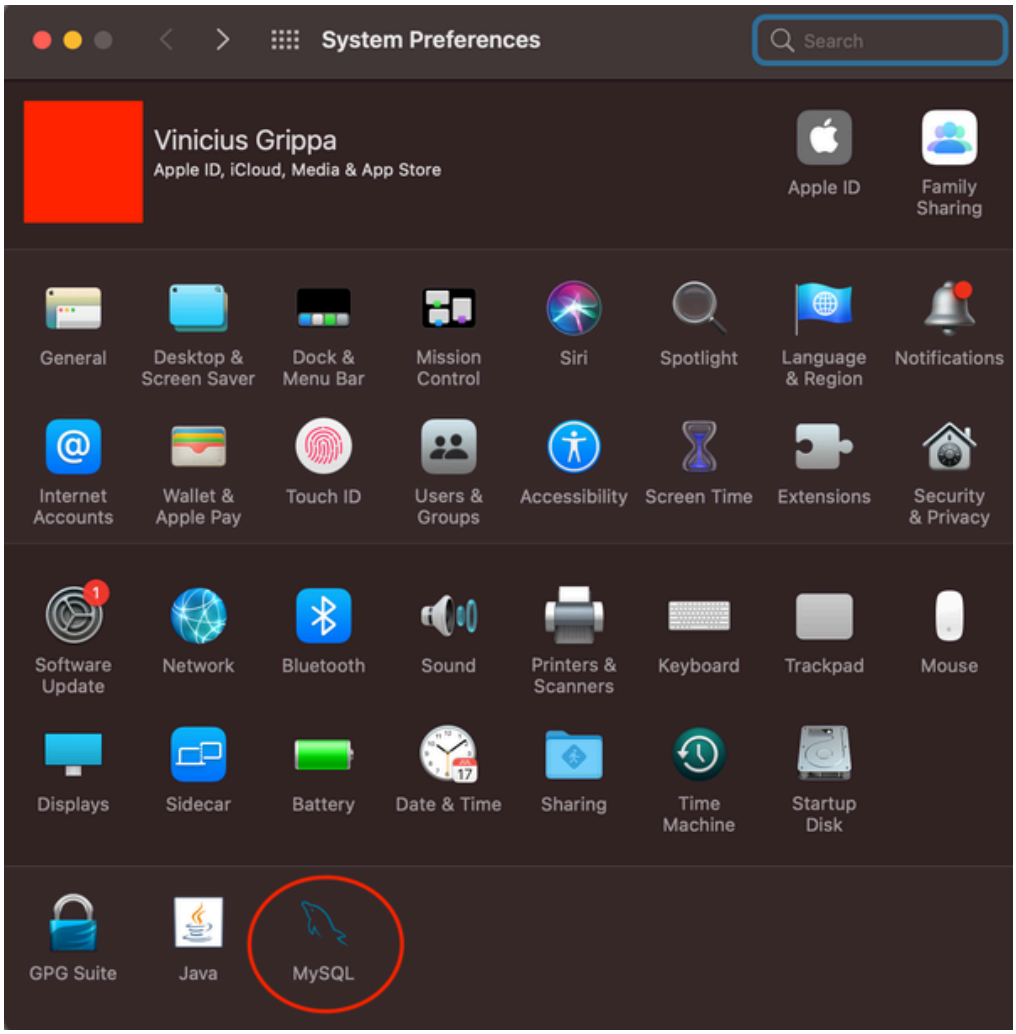

Figure 1-14. MySQL in System Preferences

Click the icon to open the MySQL panel. You should see something similar to Figure 1-15.

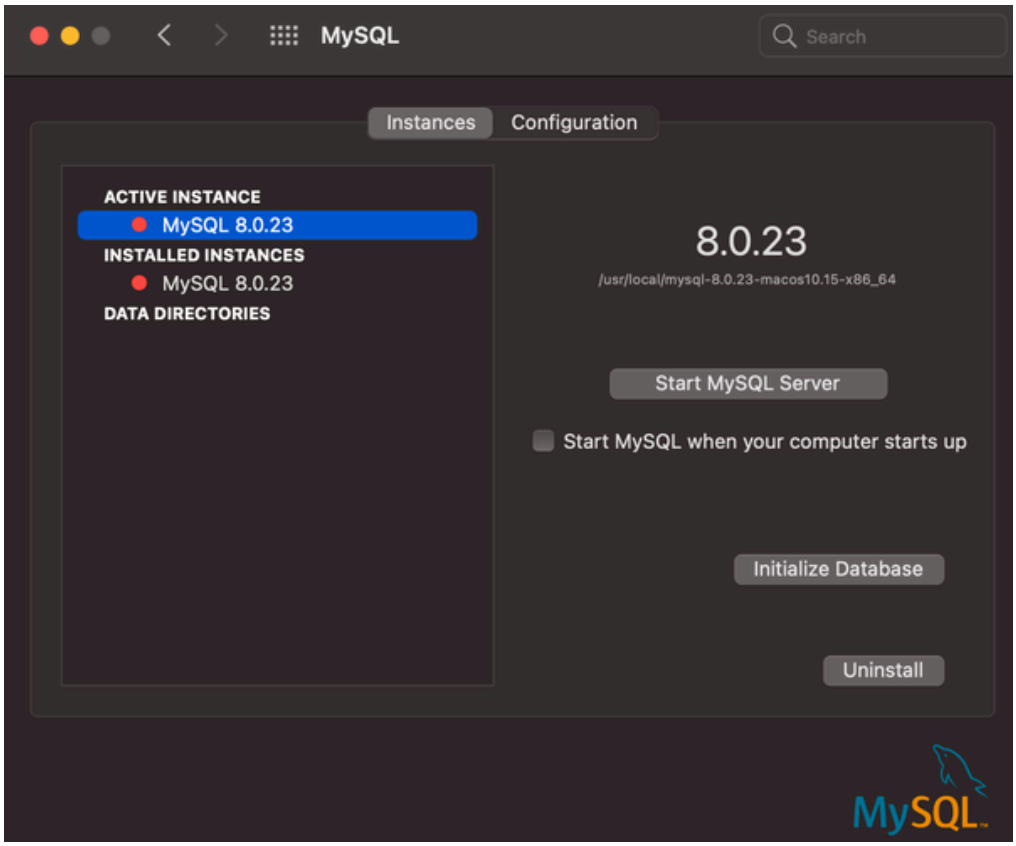

Figure 1-15. MySQL start options

Besides the obvious option, which is to start the MySQL process, there is a configuration panel (with the location of the MySQL files) and an option to reinitialize the database (you already initialized it during the installation). Start the MySQL process. You might be asked for the administrator password again.

With MySQL running, it is possible to validate the connection and confirm that MySQL Server is running correctly. You can use MySQL Workbench to test this, or install the MySQL client using `brew`:

```
$ brew install mysql-client
```

Once you've installed the MySQL client, you can connect with the password you defined in Figure 1-13. In the terminal, run the following command:

```
$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \
Your MySQL connection id is 8
```

```
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation
affiliates. Other names may be trademarks of their resp
owners.

Type help; or \h for help. Type \c to clear the current


mysql> SELECT @@version;


+-----------+
| @@version |
+-----------+
| 8.0.23    |
+-----------+
1 row in set (0.00 sec)
```

Installing MySQL on Windows 10

Oracle provides a MySQL Installer for Windows to facilitate the installation. Note that MySQL Installer is a 32-bit application, but it can install MySQL in 32-bit and 64-bit binaries. To initiate the installation process, you need to execute the installer file and choose the type of installation, as shown in Figure 1-16.

Choose the Developer Default setup type and click Next. We won't go into detail on the other options because we don't recommend using MySQL for production systems, mainly because the MySQL ecosystem is developed for Linux.

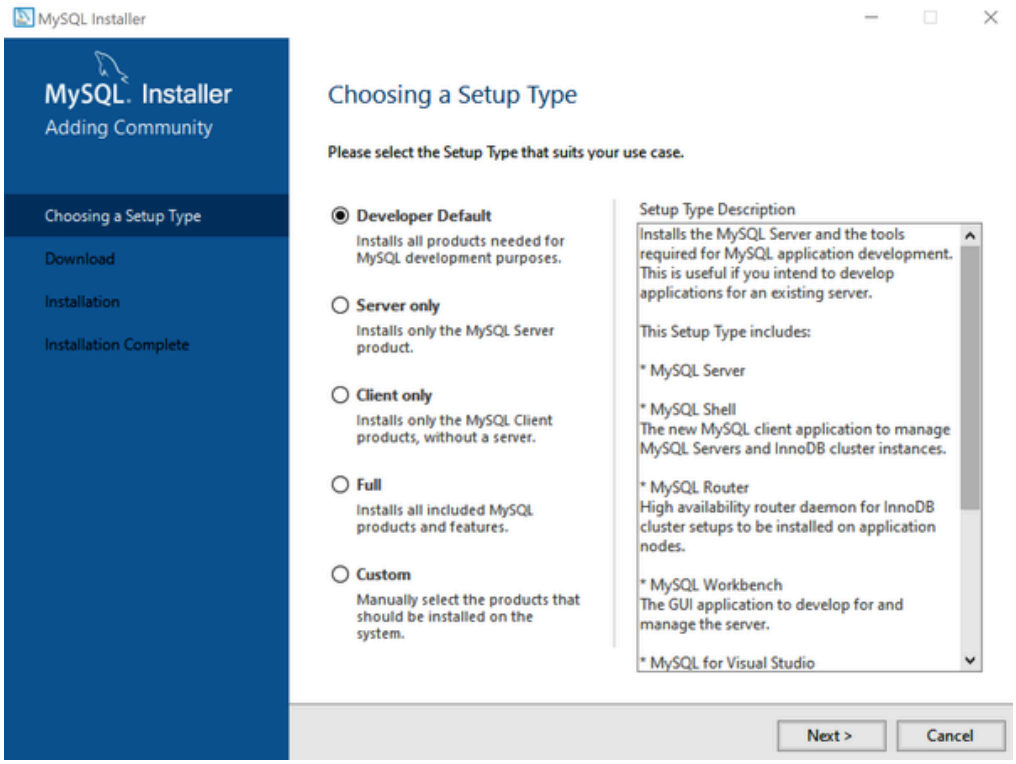

Figure 1-16. MySQL 8.0.23 Windows installation customization

Next, the installer checks whether all the requirements are satisfied (Figure 1-17).

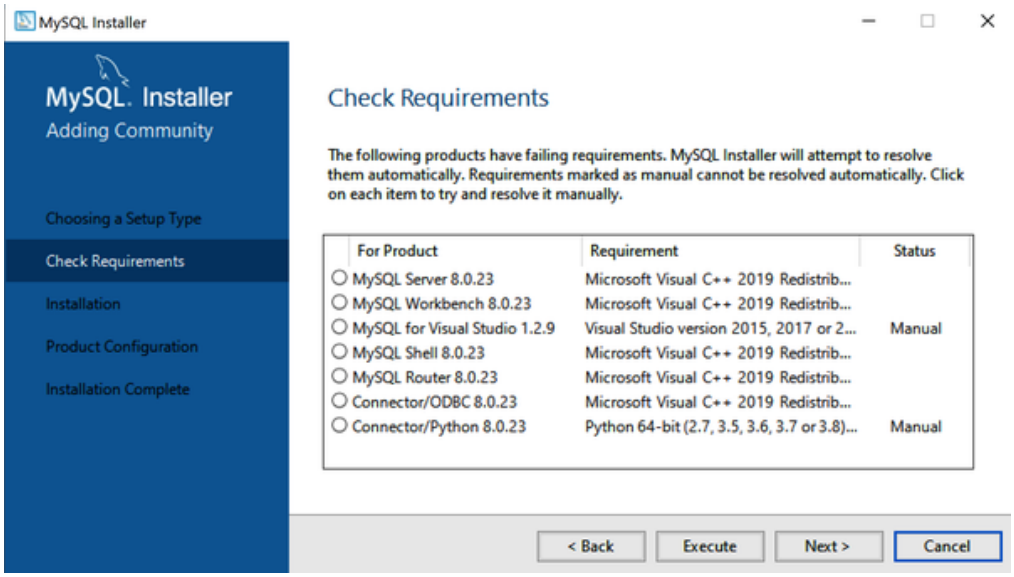

Figure 1-17. Installation requirements

Click Execute. It might be necessary to install Microsoft Visual C++ (Figure 1-18).

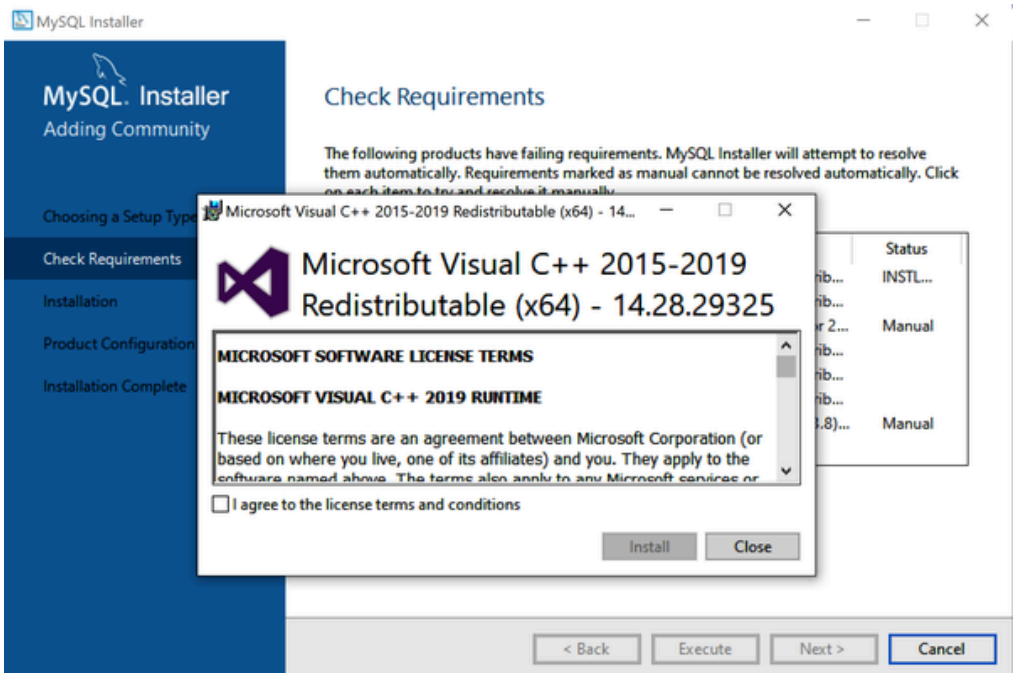

Figure 1-18. Install Microsoft Visual C++ if required

Click Next, and the installer will show the products that are ready to install (Figure 1-19).

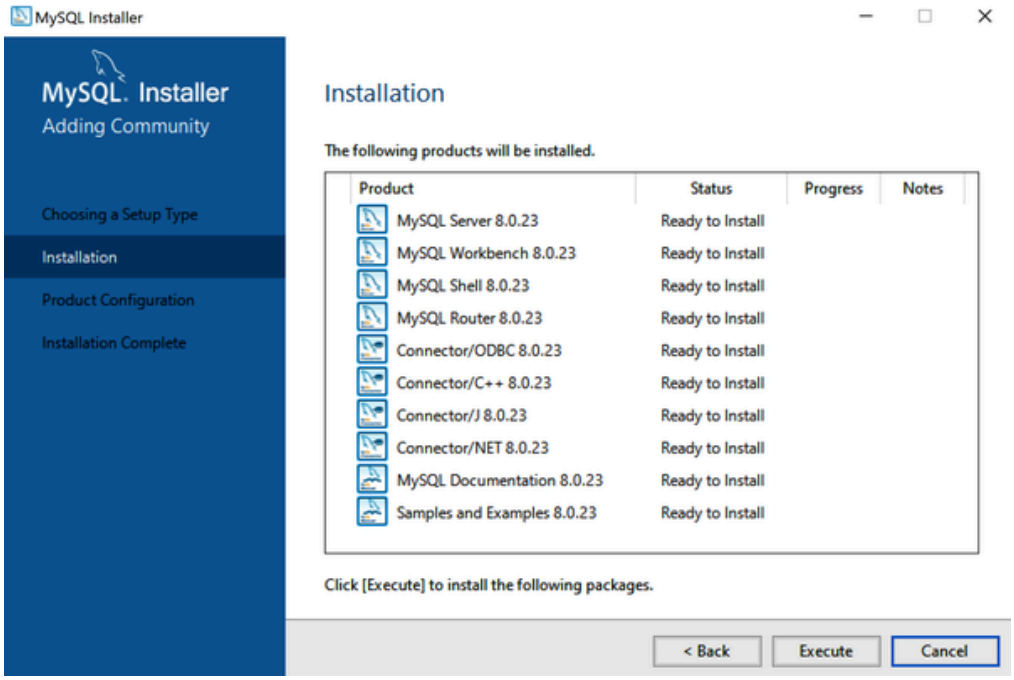

Figure 1-19. Click Execute to install the MySQL software

Click Execute and you will arrive at the screen where you can configure MySQL properties. You can use the default settings for TCP/IP and the X Protocol port, as shown in Figure 1-20, or you can customize them if you like.

Next, you will choose the authentication method. Select the newer version that is more secure, as shown in Figure 1-21.

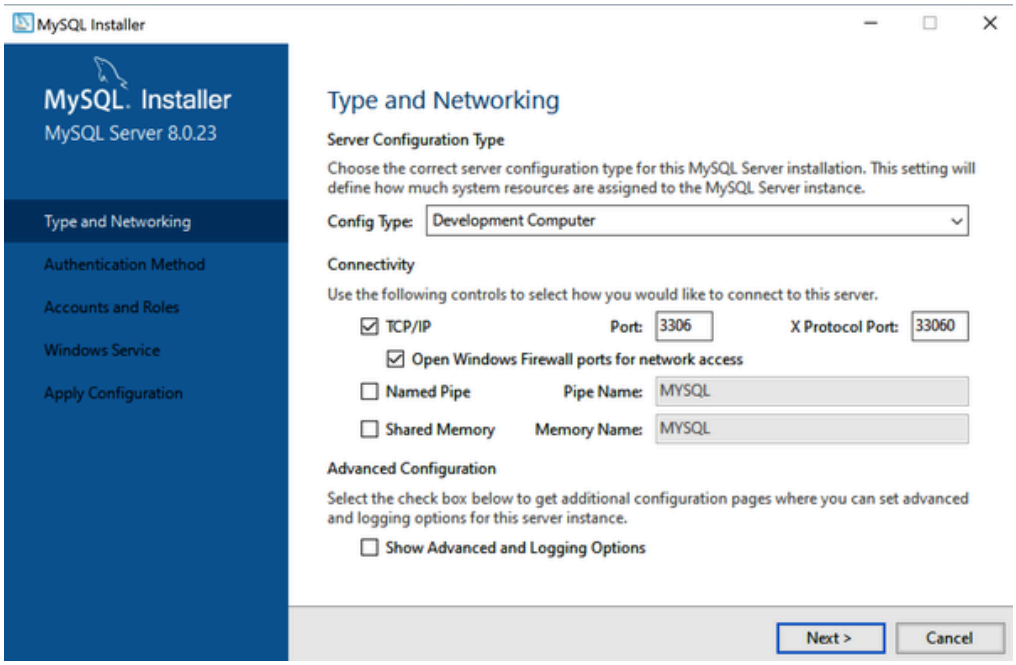

Figure 1-20. Type and networking configuration options

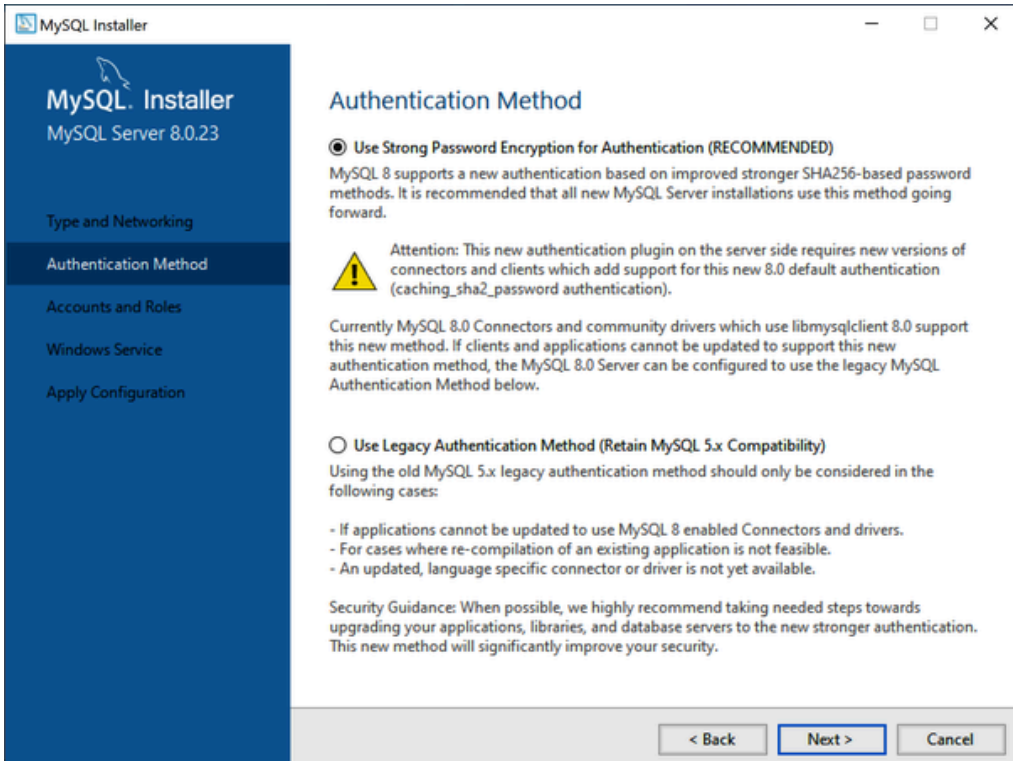

Figure 1-21. Password encryption—use SHA-256 based passwords

Next, specify the root user password and whether you want to add additional users to the MySQL database, as shown in Figure 1-22.

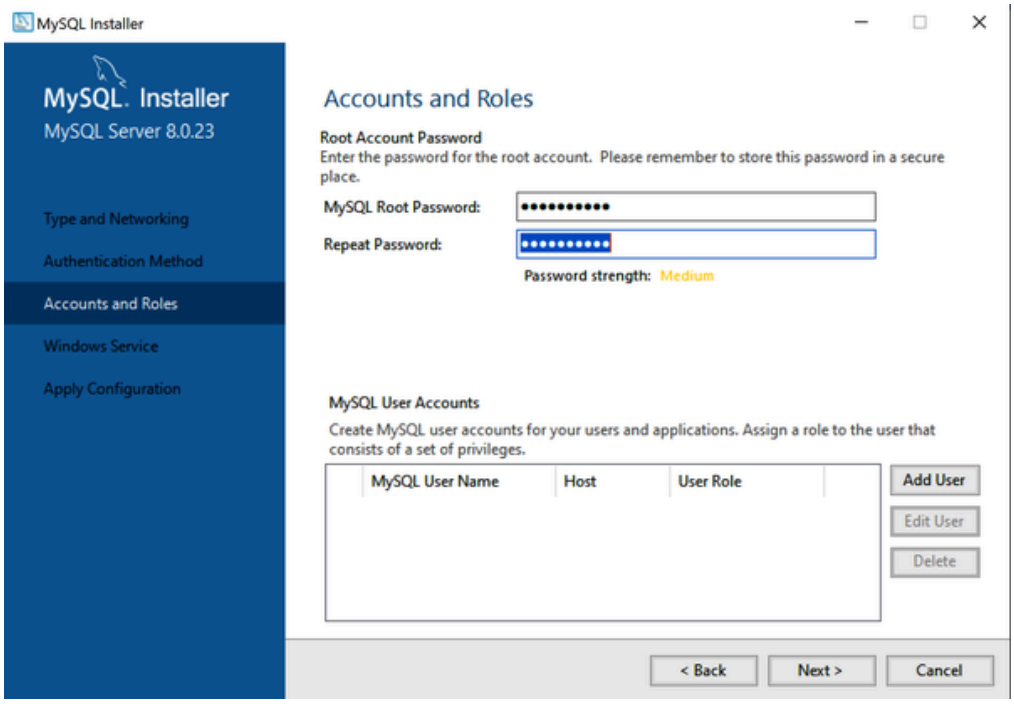

Figure 1-22. Configuring users

With the users configured, define the service name and user that will run the service, as shown in Figure 1-23.

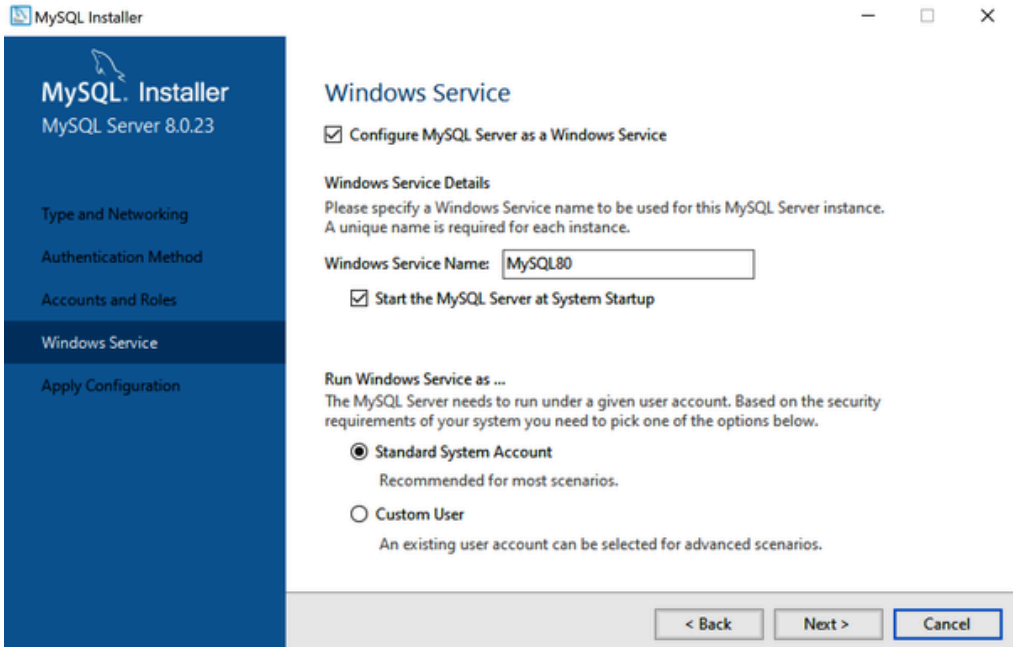

Figure 1-23. Configuring the service name

When you click Next, the installer begins configuring MySQL. Once the MySQL installer finishes its execution, you should see something like Figure 1-24.

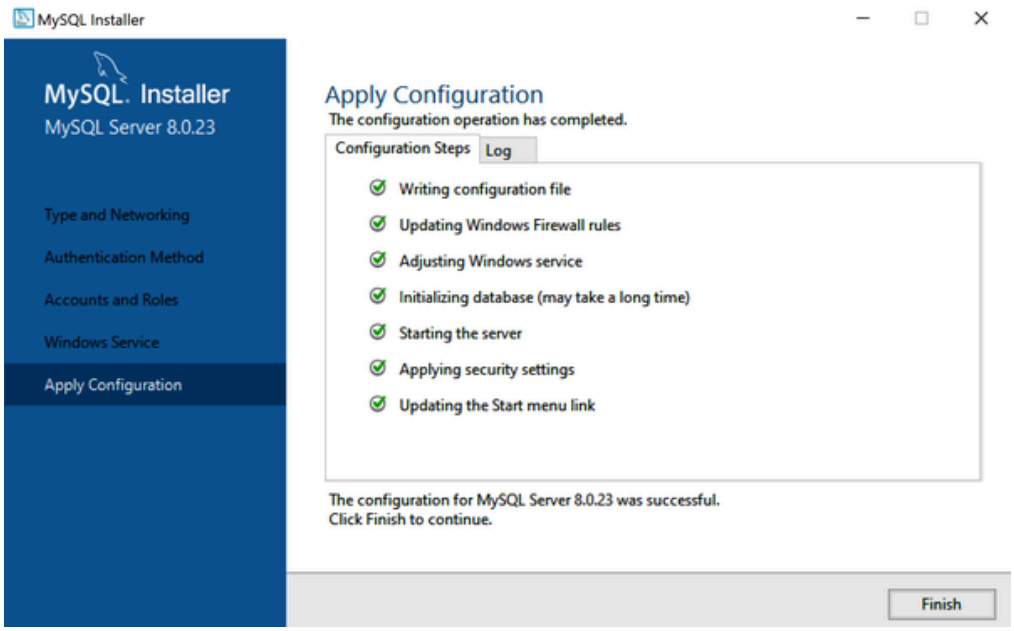

Figure 1-24. If the installation went fine, there are no errors

Now your database server is operational. Since you selected the Developer profile, the installer will go through the MySQL Router installation. MySQL Router isn't essential for this setup, and since we don't recommend Windows for production, we'll skip this part. We will dive into the details of the router in "MySQL Router".

Now you can validate your server using MySQL Workbench, as shown in Figure 1-25. You should see a MySQL connection option.

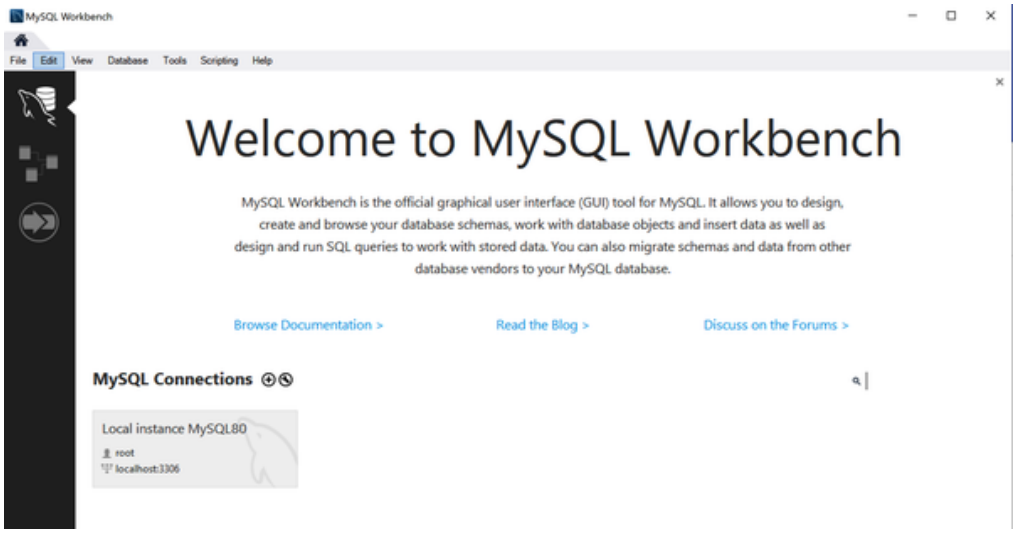

Figure 1-25. The MySQL connection option in MySQL Workbench

Double-click the connection and Workbench will prompt you to input the password, as shown in Figure 1-26.

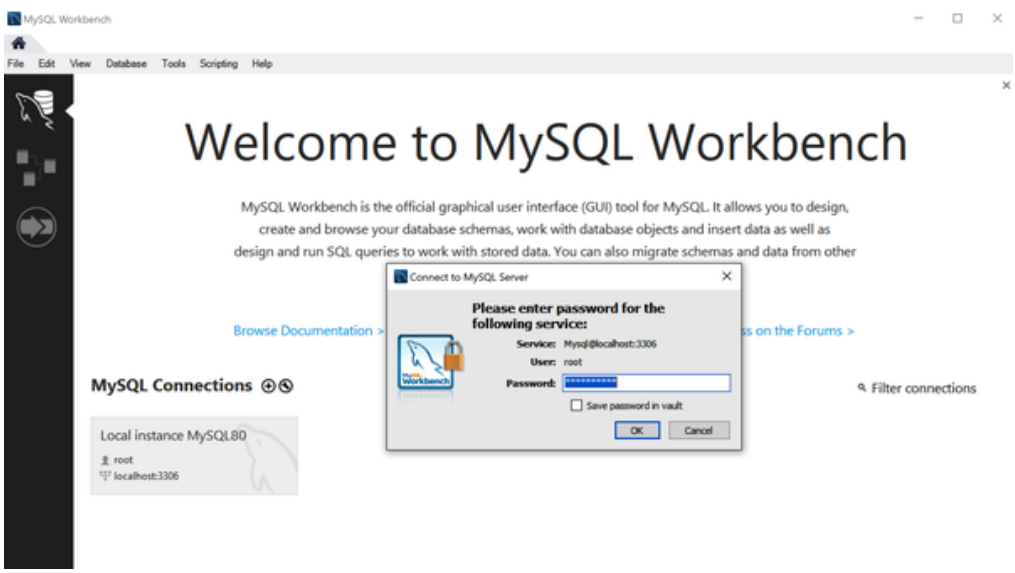

Figure 1-26. Enter the root password to connect

You can now start using MySQL in your Windows platform, as shown in Figure 1-27.

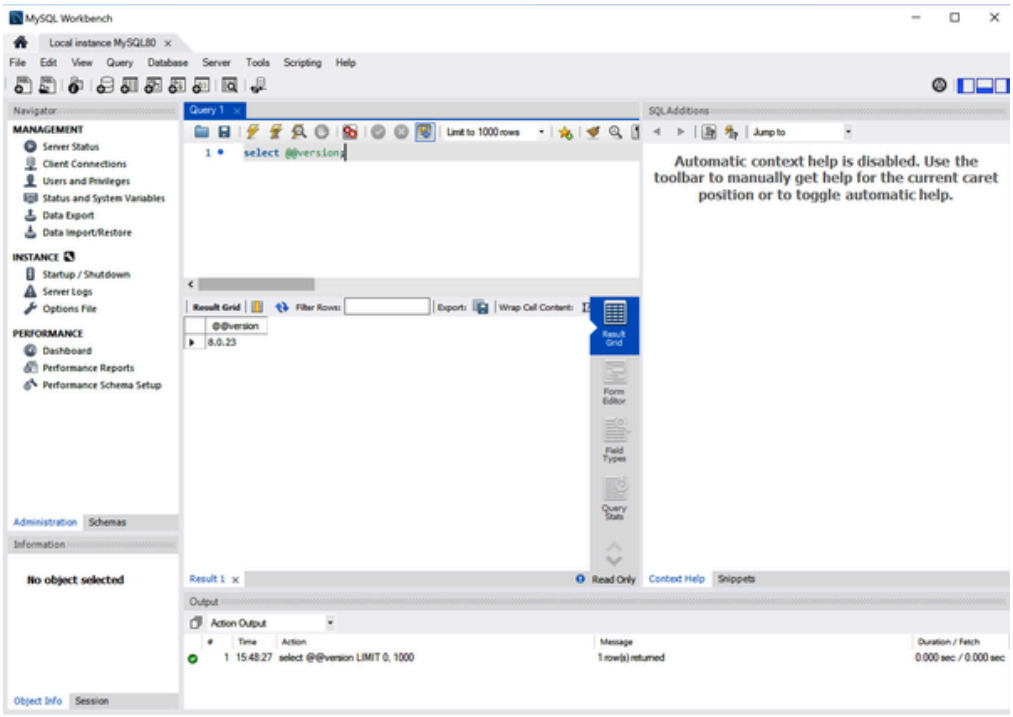

Figure 1-27. You can now begin testing your environment

The Contents of the MySQL Directory

During the installation process, MySQL creates all the files that are needed to start the server. MySQL stores its files under a directory called the *data directory*. Database administrators (DBAs) commonly refer to this as the *datadir*, which is the name of the MySQL parameter that stores the path to this directory. The default location for Linux distributions is */var/lib/mysql*. You can check its location by running the following command in the MySQL instance:

```
mysql> SELECT @@datadir;


+-----------------+
| @@datadir       |
+-----------------+
| /var/lib/mysql/ |
+-----------------+
1 row in set (0.00 sec)
```

MySQL 5.7 Default Files

The following list briefly describes the files and subdirectories typically found in the data directory:

The REDO log files

MySQL creates the redo log files as *ib_logfile0* and *ib_logfile1* in the data directory. It writes to the redo log files in a circular fashion, so the files do not grow beyond their configuration size (configured by `innodb_log_file_size`). As in any other relational database management system (RDBMS) that is ACID-compliant, the redo files are fundamental to provide data durability and the ability to recover from a crash scenario.

The auto.cnf file

MySQL 5.6 introduced the *auto.cnf* file. It has only a single `[auto]` section containing a single `server_uuid` setting and value. The `server_uuid` creates a unique signature for the server, and the replication layer uses it to communicate with different servers to replicate data.

WARNING

MySQL automatically creates the *auto.cnf* file in the data directory when initialized, and this file should not be changed. We explain the details in Chapter 9.

---

*The *.pem files*

In short, these files enable the use of encrypted connections for communication between a client and the MySQL server. Encrypted connections are a fundamental part of the network security layer to avoid unauthorized access while the data is in transit from the application to the MySQL server. MySQL 5.7 enables SSL by default and creates the certificates as well. However, it is possible to use certificates provided by different certificate authorities (CAs) in the market.

The performance_schema subdirectory

The MySQL Performance Schema is a feature for monitoring MySQL Server execution at a low level during runtime. When we can use Performance Schema to monitor a particular metric, we say that MySQL has instrumentation. For example, Performance Schema instruments can provide the number of users connected:

```
mysql> SELECT * FROM performance_schema.users;
```

```
+-----------------+----------------------+--------
| USER            | CURRENT_CONNECTIONS | TOTAL_C
+-----------------+----------------------+--------
| NULL            |                   40 |
| event_scheduler |                    1 |
| root            |                    0 |
| rsandbox        |                    2 |
| msandbox        |                    1 |
+-----------------+----------------------+--------
5 rows in set (0.03 sec)
```

NOTE

Many people are surprised to see `NULL` in the `user` column. The `NULL` value is used for internal threads or for a user session that failed to authenticate. The same applies to the `host` column in the `performance_schema.accounts` table:

```
mysql> SELECT user, host,
         total_connections AS cxns
    -> FROM performance_schema.accounts
         ORDER BY cxns DESC;
```

```
+-----------------+-----------+------+
| user            | host      | cxns |
+-----------------+-----------+------+
| NULL            | NULL      |   46 |
| rsandbox        | localhost |    3 |
| msandbox        | localhost |    2 |
| event_scheduler | localhost |    1 |
| root            | localhost |    1 |
+-----------------+-----------+------+
5 rows in set (0.00 sec)
```

Although instrumentation has existed since MySQL 5.6, it was in MySQL 5.7 that it gained many improvements and became a fundamental part of the DBA tools to investigate and troubleshoot issues at the MySQL level.

The ibtmp1 file

When the application needs to create temporary tables or MySQL needs to use an on-disk internal temporary table, MySQL creates them in a shared temporary tablespace. The default behavior is to create an auto-extending data file named *ibtmp1* that is slightly larger than 12 MB (its size is controlled by the `innodb_temp_data_file_path` parameter).

The ibdata1 file

The *ibdata1* file is probably the most famous file in the MySQL ecosystem. For MySQL 5.7 and older, it holds data for the InnoDB data dictionary, the doublewrite buffer, the change buffer, and the undo logs. It may also contain table and index data if we disable the

`innodb_file_per_table` option. When `innodb_file_per_table` is enabled, each user table has a tablespace and a dedicated file. Note that it is possible to have multiple *ibdata* files in the MySQL data directory.

NOTE

In MySQL 8.0, some of these components were removed from *ibdata1* and allocated into separate files. The remaining components are the change buffer table and index data if tables are created in the system tablespace (by disabling the `innodb_file_per_table`).

The mysql.sock file

This is a Unix socket file that the server uses for communication with local clients. This file exists only when MySQL is running, and removing it or creating the file manually may lead to problems.

NOTE

A Unix socket is an interprocess communication mechanism that allows bidirectional data exchange between processes running on the same machine. IP sockets (mainly TCP/IP sockets) are a mechanism allowing communication between processes over the network.

You can connect to MySQL Server on Linux using two methods: the TCP protocol or a socket. For security purposes, if the application and MySQL are on the same server, you can disable remote TCP connections. There are two ways to do this in MySQL Server: set the `bind-address` to `127.0.0.1` instead of the default `*` value (which accepts TCP/IP connections from everyone), or modify the `skip-networking` parameter, which disables network connections to MySQL.

The mysql subdirectory

The *mysql* directory corresponds to the MySQL system schema, which contains MySQL Server's information as it runs. For example, it includes information on users and their privileges, time zone tables, and replication. You can see the files named according to their respective table names with the `ls` command:

```
# cd /var/lib/mysql
# ls -l mysql/

-rw-r-----. 1 vinicius.grippa percona    8820 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona       0 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona    4096 Feb 20 15
columns_priv.MYI
-rw-r-----. 1 vinicius.grippa percona    9582 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona     976 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona    5120 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona      65 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona    8780 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona   98304 Feb 20 15
...
-rw-r-----. 1 vinicius.grippa percona   10816 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona    1292 Feb 20 15
-rw-r-----. 1 vinicius.grippa percona    4096 Feb 20 15
```

MySQL 8.0 Default Files

MySQL 8.0 brought a few changes in the core of the data directory structure. Some of these changes are related to implementing the new data dictionary, and others to improving database management. The following list describes the new files and changes:

The undo tablespace files

MySQL (InnoDB) uses *undo* files to undo the transactions that need to be rolled back and ensure isolated transactions whenever it needs to perform a consistent read.

From MySQL 8.0, the undo log files were separated from the system tablespace (*ibdata1*) and placed in the data directory. It is also possible to set another location by changing the `innodb_undo_directory` parameter.

The .dblwr files (introduced in version 8.0.20)

The doublewrite buffer is responsible for writing pages flushed from the buffer pool to the disk before MySQL writes the pages to the datafiles. The doublewrite filenames have the following format: *#ib_<page_size>_<file_number>.dblwr* (for example,

#ib_16384_0.dblwr, *#ib_16384_0.dblwr*). It is possible to change the location of these files by modifying the `innodb_doublewrite_dir` parameter.

The mysql.ibd file (introduced in version 8.0)

In MySQL 5.7, dictionary tables and system tables stored data and metadata in the *mysql* directory inside the *datadir*. In MySQL 8.0, this is all stored in the *mysql.ibd* file and is protected by the InnoDB mechanisms to ensure consistency.

Using the Command-Line Interface

The *mysql* binary is a simple SQL shell with input line-editing capabilities. Its use is straightforward (we already used it a few times during the installation process). To invoke it, run the following command:

```
# mysql
```

We can extend its functionality by executing queries in it:

```
# mysql -uroot -pseKret -e "SHOW ENGINE INNODB STATUS\G
```

And we can execute more advanced commands, piping them with other commands to perform more complex tasks. For example, we can extract a dump from one database, send it across the network, and restore it into another MySQL server in the same command line:

```
# mysql -e "SHOW MASTER STATUS\G" && nice -5 mysqldump
    --all-databases --single-transaction -R --master-da
    --log-error=/tmp/donor.log --verbose=TRUE | ssh mys
    1> /tmp/receiver.log 2>&1
```

MySQL 8.0 introduced *MySQL Shell*, which is way more powerful than its predecessor. MySQL Shell supports the JavaScript, Python, and SQL languages, providing development and administration capabilities for MySQL Server. We'll go into more detail about this in "MySQL Shell".

Using Docker

With the advent of virtualization and its popularization with cloud services, many platforms have emerged, including Docker. Born in 2013, Docker is a solution that offers a portable and flexible way to deploy software. It provides resource isolation through the use of Linux features like *cgroups* and *kernel namespaces*.

Docker is useful for DBAs who often need to install a specific version of MySQL, MariaDB, or Percona Server for MySQL to run some experiments. With Docker, it is possible to deploy a MySQL instance in seconds to perform some tests. Once you finish the tests, you can destroy the instance and release the operating system's resources to other tasks. All the processes of deploying a virtual machine (VM), installing packages, and configuring the database are simpler when using Docker.

Installing Docker

An advantage of using Docker is that once the service is running, the commands are the same in all operating systems. The commands being the same means that the learning curve for using Docker is faster compared to learning different Linux versions such as CentOS and Ubuntu, for example.

The process for installing Docker is, in some ways, similar to installing MySQL. For Windows and macOS you just install the binaries, and after that the service is up and running. For Linux-based operating systems without a graphic interface, the process requires configuring the repository.

Installing Docker on CentOS 7

The CentOS packages for Docker are, in general, older than the ones available to RHEL and in official Docker repositories. At the time of writing, the Docker version provided by regular CentOS repositories is 1.13.1, whereas the upstream stable version is 20.10.3. There is no difference for the purposes of this book, but we always recommend using the latest version for production environments.

Execute the following command to install the Docker package from the default CentOS repository:

```
# yum install docker -y
```

If you want to install Docker from the upstream repository to ensure that you are using the latest release, follow these steps:

1. Install `yum-utils` to enable the `yum-config-manager` command:

   ```
   # yum install yum-utils -y
   ```

2. Use `yum-config-manager` to add the *docker-ce* repository:

   ```
   # yum-config-manager \
       --add-repo \
       https://download.docker.com/linux/centos/docker-
   ```

3. Install the necessary packages:

   ```
   # yum install docker-ce docker-ce-cli containerd.io
   ```

4. Start the Docker service:

   ```
   # systemctl start docker
   ```

5. Enable the Docker service to auto-start after a system reboot:

   ```
   # systemctl enable --now docker
   ```

6. To validate whether the Docker service is running, execute the `systemctl status` command:

   ```
   # systemctl status docker
   ```

7. To verify that Docker Engine is installed correctly, you can run the *hello-world* container:

```
# docker run hello-world
```

Installing Docker on Ubuntu 20.04 (Focal Fossa)

To install the latest Docker release from the upstream repository, first remove any older versions of Docker (called *docker*, *docker.io*, or *docker-engine*). Uninstall them with this command:

```
# apt-get remove -y docker docker-engine docker.io cont
```

With the default repository removed, you can initiate the installation process:

1. Make sure that Ubuntu is up-to-date with this command:

```
# apt-get update -y
```

2. Install packages to allow *apt* to use a repository over HTTPS:

```
# apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

3. Next, add Docker's official GPG key:

```
# curl -fsSL https://download.docker.com/linux/ubunt
    apt-key add -
```

4. With the key in place, add the Docker stable repository:

```
# add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/lin
   $(lsb_release -cs) \
   stable"
```

5. Now, use the `apt` command to install the Docker packages:

```
# apt-get install -y docker-ce docker-ce-cli contain
```

6. Ubuntu will start the service for you, but you can check by running this command:

```
# systemctl status docker
```

7. To make the Docker service auto-start when the OS reboots, use:

```
# systemctl enable --now docker
```

8. Check the Docker version you installed with:

```
# docker --version
```

9. To verify that Docker Engine is installed correctly, you can run the *hello-world* container:

```
# docker run hello-world
```

Deploying the MySQL container

Once you have Docker Engine installed and running, the next step is deploying the MySQL Docker container.

WARNING

We designed the following instructions to get a test instance running quickly and easily; do not use this for a production deployment!

To deploy the latest MySQL version with Docker, execute this command:

```
# docker run --name mysql-latest \
  -p 3306:3306 -p 33060:33060 \
```

```
-e MYSQL_ROOT_HOST=% -e
MYSQL_ROOT_PASSWORD='learning_mysql' \
-d mysql/mysql-server:latest
```

Docker Engine will launch the latest version of the MySQL instance and be remotely accessible from anywhere with the specified root password. Installing MySQL with Docker means that you do not have access to any of the tools, utilities, or standard libraries available in a traditional host (bare metal or VM). You'll need to either deploy these tools separately or use commands shipped with the Docker image if you need them.

Next, connect to the MySQL container using the MySQL client:

```
# docker exec -it mysql-latest mysql -uroot -plearning_
```

Since you mapped the TCP port 3306 in the container to port 3306 on the Docker host with the parameter `-p 3306:3306`, you can connect to the MySQL database from any MySQL client (Workbench, MySQL Shell) available that can reach the host (hostname or IP) and that port.

Let's look at a few commands to manage the container.

To stop the MySQL Docker container, run:

```
# docker stop mysql-latest
```

Don't try to use `docker run` to start the container again. Instead, use this:

```
# docker start mysql-latest
```

To investigate an issue—for example, if the container isn't starting—access its logs using this command:

```
# docker logs mysql-latest
```

To remove the Docker container that you created, run:

```
# docker stop mysql-latest
# docker rm mysql-latest
```

To check which and how many Docker containers are running in the host, use:

```
# docker ps
```

It is possible to customize MySQL parametrization using command-line options to Docker Engine. To configure the InnoDB buffer pool size and the flush method, run the following:

```
# docker run --name mysql-latest \
  -p 3306:3306 -p 33060:33060  \
  -e MYSQL_ROOT_HOST=% -e
  MYSQL_ROOT_PASSWORD='strongpassword' \
  -d mysql/mysql-server:latest \
  --innodb_buffer_pool_size=256M \
  --innodb_flush_method=O_DIRECT
```

To run a MySQL version other than the latest version, first check that it is available in Docker Hub. For example, say you want to run MySQL 5.7.31. The first step is to check the official MySQL Docker Images list in Docker Hub to see if it exists.

Once you've confirmed its existence, run it with the following command:

```
# docker run --name mysql-5.7.31 \
  -p 3307:3306 -p 33061:33060  \
  -e MYSQL_ROOT_HOST=% -e \
  MYSQL_ROOT_PASSWORD='learning_mysql' \
  -d mysql/mysql-server:5.7.31
```

It is possible to run multiple MySQL Docker instances at the same time, but a potential problem is TCP port conflicts. In the previous example, note that we mapped different host ports for the *mysql-5.7.31* container (3307 and 33061). Also, the *name* of the container needs to be unique.

Deploying MariaDB and Percona Server containers

You follow the same steps described in the previous section for deploying a MySQL container to deploy a MariaDB or Percona Server container. The main difference is that they use different Docker images and have their own official repositories.

To deploy a MariaDB container, run:

```
# docker run --name maria-latest \
  -p 3308:3306 \
  -e MYSQL_ROOT_HOST=% -e \
  MYSQL_ROOT_PASSWORD='learning_mysql' \
  -d mariadb:latest
```

And for Percona Server, run:

```
# docker run --name ps-latest \
  -p 3309:3306 -p 33063:33060 \
  -e MYSQL_ROOT_HOST=% -e \
  MYSQL_ROOT_PASSWORD='learning_mysql' \
  -d percona/percona-server:latest \
  --innodb_buffer_pool_size=256M \
  --innodb_flush_method=O_DIRECT
```

NOTE

We are mapping different ports for MariaDB (`-p 3308:3306`) and Percona (`-p 3309:3306`) because we are deploying all the containers in the same host:

```
# docker ps


CONTAINER ID            IMAGE
5e487dd41c3e            percona/percona-server:latest

COMMAND                 CREATED           STATUS
"/docker-entrypoint..." About a minute ago Up 51 seconds
"docker-entrypoint..."  2 minutes ago      Up 2 minutes

PORTS                   NAMES
0.0.0.0:3309->3306/tcp, ps-latest
0.0.0.0:33063->33060/tcp
f5a217f1537b            mariadb:latest
0.0.0.0:3308->3306/tcp  maria-latest
```

If you are deploying a single container, you can use port 3306 or any custom port you might want to use.

Using Sandboxes

In software development, a *sandbox* is a testing environment that isolates code changes and allows experimentation and testing before deploying to production. DBAs primarily use sandboxes for testing new software versions, performance tests, and bug analysis, and the data present in MySQL is disposable.

NOTE

It is common in the context of MySQL databases to hear the terms *master* and *slave*. The origins of these words are clearly negative. Oracle, Percona, and MariaDB have therefore decided to change this terminology and instead use *source* and *replica*. In this book, we will use both sets of terms because you will encounter both of them, but be aware that these companies will implement the following terminology for the upcoming releases:

Old	New
master	source
slave	replica
blacklist	blocklist
whitelist	allowlist

In 2018, Giuseppe Maxia introduced DBdeployer, a tool that provides an easy and fast way to deploy MySQL and its forks. It supports diverse MySQL topologies such as master/slave (source/replica), master/master (source/source), Galera Cluster, and Group Replication.

Installing DBdeployer

The tool is developed in the *Go* language and works with macOS and Linux (Ubuntu and CentOS), and standalone executables are provided. Get the latest version here:

```
# wget https://github.com/datacharmer/dbdeployer/releas
    dbdeployer-1.58.2.linux.tar.gz
# tar -xvf dbdeployer-1.58.2.linux.tar.gz
# mv dbdeployer-1.58.2.linux /usr/local/bin/dbdeployer
```

If you have your */usr/local/bin/* directory in the `$PATH` variable, you should now be able to run the `dbdeployer` commands:

```
# dbdeployer --version
dbdeployer version 1.58.2
```

Using DBdeployer

The first step in using DBdeployer is to download the MySQL binary you want to run and unpack it into the directory where you store your binaries. We will use *Linux - Generic* tarballs since they are compatible with most Linux distributions, and we will store our binaries in the */opt/mysql* directory:

```
# wget https://dev.mysql.com/get/Downloads/MySQL-8.0/ \
    mysql-8.0.11-linux-glibc2.12-x86_64.tar.gz
# mkdir /opt/mysql
# dbdeployer --sandbox-binary=/opt/mysql/ unpack \
    mysql-8.0.11-linux-glibc2.12-x86_64.tar.gz
```

The `unpack` command will extract and move the files to the specified directory. The expected output of this operation is:

```
# dbdeployer --sandbox-binary=/opt/mysql/ unpack


mysql-8.0.11-linux-glibc2.12-x86_64.tar.gz
Unpacking tarball mysql-8.0.11-linux-glibc2.12-x86_64.t
/opt/mysql/8.0.11
.........100.........200........289
Renaming directory /opt/mysql/mysql-8.0.11-linux-glibc2
/opt/mysql/8.0.11
```

We can now use the following command to create a new standalone MySQL sandbox with the newly extracted binary:

```
# dbdeployer --sandbox-binary=/opt/mysql/ deploy single
```

And we can observe DBdeployer initializing MySQL:

```
# dbdeployer --sandbox-binary=/opt/mysql/ deploy single
```

```
Creating directory /root/sandboxes
Database installed in $HOME/sandboxes/msb_8_0_11
run 'dbdeployer usage single' for basic instructions'
. sandbox server started
```

Confirm that MySQL is running with the `ps` command:

```
# ps -ef | grep mysql
```

```
root      4249     1  0 20:18 pts/0    00:00:00 /bin/sh
--defaults-file=/root/sandboxes/msb_8_0_11/my.sandbox.c
root      4470  4249  1 20:18 pts/0    00:00:00 /opt/my
--defaults-file=/root/sandboxes/msb_8_0_11/my.sandbox.c
--basedir=/opt/mysql/8.0.11 --datadir=/root/sandboxes/m
--plugin-dir=/opt/mysql/8.0.11/lib/plugin --user=root
--log-error=/root/sandboxes/msb_8_0_11/data/msandbox.er
--pid-file=/root/sandboxes/msb_8_0_11/data/mysql_sandbo
--socket=/tmp/mysql_sandbox8011.sock --port=8011
root      4527  3836  0 20:18 pts/0    00:00:00 grep --
```

We can now connect to MySQL using DBdeployer's `use` command:

```
# cd sandboxes/msb_8_0_11/
# ./use
```

or using the default root credentials:

```
# mysql -uroot -pmsandbox -h 127.0.0.1 -P 8011
```

NOTE

We got the port information from the previous `ps` command. Remember that there are two ways to connect to MySQL: via TCP/IP or using a socket. We can also get the socket file location from the output of the `ps` command and connect with that, as shown here:

```
# mysql -uroot -pmsandbox -S/tmp/mysql_sandbox8011.sock
```

If we want to set up a replication environment with a source/replica topology, we can do it with the following command line:

```
# dbdeployer --sandbox-binary=/opt/mysql/ deploy replic
```

And we will have three `mysqld` processes running:

```
# ps -ef | grep mysql

root      4673     1  0 20:26 pts/0    00:00:00 /bin/sh
--defaults-file=/root/sandboxes/rsandbox_8_0_11/master/
root      4942  4673  1 20:26 pts/0    00:00:00
/opt/mysql/8.0.11/bin/mysqld
...
--pid-file=/root/sandboxes/rsandbox_8_0_11/master/data/
12.pid --socket=/tmp/mysql_sandbox20112.sock --port=201

root      5051     1  0 20:26 pts/0    00:00:00 /bin/sh
--defaults-file=/root/sandboxes/rsandbox_8_0_11/node1/m
root      5320  5051  1 20:26 pts/0    00:00:00
/opt/mysql/8.0.11/bin/mysqld
--defaults-file=/root/sandboxes/rsandbox_8_0_11/node1/m
...
--pid-file=/root/sandboxes/rsandbox_8_0_11/node1/data/m
3.pid --socket=/tmp/mysql_sandbox20113.sock --port=2011

root      5415     1  0 20:26 pts/0    00:00:00 /bin/sh
--defaults-file=/root/sandboxes/rsandbox_8_0_11/node2/m
root      5684  5415  1 20:26 pts/0    00:00:00
/opt/mysql/8.0.11/bin/mysqld
...
```

```
--pid-file=/root/sandboxes/rsandbox_8_0_11/node2/data/m
4.pid --socket=/tmp/mysql_sandbox20114.sock --port=2011
```

Another topology that DBdeployer can configure is Group Replication. For this example, we will define a `base-port`. By doing this, we will order DBdeployer to configure our servers starting from port 49007:

```
# dbdeployer deploy --topology=group replication --sanc
    8.0.11 --base-port=49007
```

Now let's see an example of the deployment of Galera Cluster using Percona XtraDB Cluster 5.7.32. We will indicate the `base-port`, and we want our nodes configured with the `log-slave-updates` option:

```
# wget https://downloads.percona.com/downloads/Percona-
    Percona-XtraDB-Cluster-5.7.32-31.47/binary/tarball/
    5.7.32-rel35-47.1.Linux.x86_64.glibc2.17-debug.tar.
# dbdeployer --sandbox-binary=/opt/mysql/ unpack\
    Percona-XtraDB-Cluster-5.7.32-rel35-47.1.Linux.x86_
# dbdeployer deploy --topology=pxc replication\
    --sandbox-binary=/opt/mysql/ 5.7.32 --base-port=450
```

As we've seen, it is possible to customize MySQL parameters. One interesting option is enabling MySQL replication using *global transaction identifiers*, or GTIDs (we'll discuss GTIDs in more detail in Chapter 13):

```
# dbdeployer deploy replication --sandbox-binary=/opt/m
```

Our last example shows that it is possible to deploy multiple standalone versions at once—here, we create five standalone instances:

```
# dbdeployer deploy multiple --sandbox-binary=/opt/mysc
```

The previous examples are just a small sample of DBdeployer's capabilities. The full documentation is available on GitHub. Another option to understand the universe of possibilities is to use `--help` in the command line:

```
# dbdeployer --help


dbdeployer makes MySQL server installation an easy task
Runs single, multiple, and replicated sandboxes.

Usage:
  dbdeployer [command]

Available Commands:
  admin           sandbox management tasks
  cookbook        Shows dbdeployer samples
  defaults        tasks related to dbdeployer defaults
  delete          delete an installed sandbox
  delete-binaries delete an expanded tarball
  deploy          deploy sandboxes
  downloads       Manages remote tarballs
  export          Exports the command structure in JSON
  global          Runs a given command in every sandbox
  help            Help about any command
  import          imports one or more MySQL servers int
  info            Shows information about dbdeployer er
  sandboxes       List installed sandboxes
  unpack          unpack a tarball into the binary dire
  update          Gets dbdeployer newest version
  usage           Shows usage of installed sandboxes
  versions        List available versions

Flags:
      --config string           configuration file (def
                                "/root/.dbdeployer/conf
  -h, --help                    help for dbdeployer
      --sandbox-binary string   Binary repository (defa
                                "/root/opt/mysql")
      --sandbox-home string     Sandbox deployment dire
                                "/root/sandboxes")
      --shell-path string       Which shell to use for
                                scripts (default "/usr/
      --skip-library-check      Skip check for needed l
                                cause nasty errors)
      --version                 version for dbdeployer

Use "dbdeployer [command] --help" for more information
```

Upgrading MySQL Server

If the most common question to arise is about replication, the second most common is about how to upgrade a MySQL instance. If the procedure is not well tested before it's done in production, the chances of having a problem are high. There are two types of upgrades that you can perform:

- A *major upgrade* in MySQL would be changing versions from 5.6 to 5.7 or 5.7 to 8.0. Such an upgrade is trickier and more complex than a minor upgrade because the changes to the architecture are more substantial. For example, a considerable change in MySQL 8.0 involved modifying the data dictionary, which is now transactional and encapsulated by InnoDB.
- A *minor upgrade* would be changing from MySQL 5.7.29 to 5.7.30 or MySQL 8.0.22 to MySQL 8.0.23. Most of the time, you'll need to install the new version using your distribution's package manager. A minor upgrade is simpler than a major one because it does not involve any changes in the architecture. The modifications are focused on fixing bugs, improving the performance, and optimizing the code.

To start planning for an upgrade, first choose between two strategies. These are the recommended strategies according to the documentation and are the ones we use:

In-place upgrade

This involves shutting down MySQL, replacing the old MySQL binaries or packages with the new ones, restarting MySQL in the existing data directory, and running `mysql_upgrade` .

NOTE

As of MySQL 8.0.16, the *mysql_upgrade* binary is deprecated, and the MySQL server itself executes its functionality (you can think of it as a "server upgrade"). MySQL added this change alongside the data dictionary upgrade (DD upgrade), which is a process to update the data dictionary table definitions. Benefits of the new process include:

- Faster upgrades
- Simpler process
- Better security
- Significant reduction in upgrade steps
- More easily automated
- No restarts
- Plug and play

Logical upgrade

This involves exporting the data in SQL format from the old MySQL version using a backup or export utility such as *mysqldump* or *mysqlpump*, installing the new MySQL version, and applying the SQL data to the new MySQL version. In other words, this process involves rebuilding the entire data dictionary and the user data. A logical upgrade usually takes longer than an in-place upgrade.

Regardless of your chosen strategy, it is essential to establish a rollback strategy in case something goes wrong. The rollback strategy will vary based on the upgrade plan you choose, and the database size and the topology present (if you're using replicas or Galera Cluster, for example) will influence this decision.

Here are some additional points to take into consideration when planning an upgrade:

- Upgrading from MySQL 5.7 to 8.0 is supported. However, the upgrade is only supported between GA releases. For MySQL 8.0, it is required that you upgrade from a MySQL 5.7 GA release (5.7.9 or higher). Upgrades from non-GA releases of MySQL 5.7 are not supported.
- Upgrading to the latest release is recommended before upgrading to the next version. For example, upgrade to the latest MySQL 5.7 release before upgrading to MySQL 8.0.

- Upgrades that skip versions are not supported. For example, upgrading directly from MySQL 5.6 to 8.0 is not supported.

NOTE

Based on our experience, moving from MySQL 5.6 to MySQL 5.7 is the upgrade that causes the most performance issues, especially if the application is using *derived tables* (see "Nested Queries in the FROM Clause"). MySQL 5.7 modified the `optimizer_switch` system variable, enabling the `derived_merge` setting by default, and this can hurt query performance.

Another complicating change is that MySQL 5.7 implements network encryption by default (SSL). Applications that were not using SSL in MySQL 5.6 may suffer a substantial performance hit.

Finally, MySQL 5.7 changed the `sync_binlog` default to synchronous mode. This mode is the safest but can harm performance due to the increased number of disk writes.

Let's go through an example of upgrading from MySQL 5.7 upstream to MySQL 8.0 upstream using the in-place method:

1. Stop the MySQL service. Perform a clean shutdown using `systemctl`:

```
# systemctl stop mysqld
```

2. Remove the old binaries:

```
# yum erase mysql-community -y
```

   This process only removes the binaries and does not touch the *datadir* (see "The Contents of the MySQL Directory").
3. Follow the regular steps for the installation process (see "Installing MySQL on Linux"). For example, to use MySQL 8.0 Community Version on CentOS 7 using `yum`:

```
# yum-config-manager --enable mysql80-community
```

4. Install the new binaries:

```
# yum install mysql-community-server -y
```

5. Start the MySQL service:

```
# systemctl start mysqld
```

We can observe in the logs that MySQL upgraded the data dictionary and that we're now running MySQL 8.0.21:

```
# tail -f /var/log/mysqld.log
```

```
2020-08-09T21:20:10.356938Z 2 [System] [MY-011003] [Ser
populating Data Dictionary tables with data.
2020-08-09T21:20:11.734091Z 5 [System] [MY-013381] [Ser
upgrade from '50700' to '80021' started.
2020-08-09T21:20:17.342682Z 5 [System] [MY-013381] [Ser
upgrade from '50700' to '80021' completed.
...
2020-08-09T21:20:17.463685Z 0 [System] [MY-010931] [Ser
/usr/sbin/mysqld: ready for connections. Version: '8.0.
'/var/lib/mysql/mysql.sock'  port: 3306  MySQL Communit
```

NOTE

We highly recommend before upgrading MySQL that you check the release notes. They contain a summary of the changes made and the bug fixes. Release notes are available for MySQL upstream, Percona Server, and MariaDB.

A common question is whether it's safe to upgrade to the latest major release. The answer is…it depends. As with any new product in the industry, early adopters tend to benefit from the new features, but they are testers as well, and they may discover and be affected by new bugs. When MySQL 8.0 was released, our recommendation was to wait for three minor releases before considering moving. The golden rule of this book is to test everything in advance before executing the next step. If you learn just that from this book, we will consider our mission accomplished.