# OLD:How to secure an initial MySQL installation

How to secure an initial installation of Mysql

**!!this is the old version, please use the other wiki page!!**

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER ! To do so, start the server, then issue the following commands:

```
/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h server.wonen.amsterdam.nl password 'new-password'
```

See the manual for more instructions.

You can test the MySQL daemon with the benchmarks in the 'sql-bench' directory:

```
cd sql-bench ; perl run-all-tests
```

---

After installing MySQL on Unix, you need to initialize the grant tables, start the server, and make sure that the server works satisfactorily. You may also wish to arrange for the server to be started and stopped automatically when your system starts and stops. You should also assign passwords to the accounts in the grant tables.

On Unix, the grant tables are set up by the mysql_install_db program. For some installation methods, this program is run for you automatically:

```
   *      If you install MySQL on Linux using RPM distributions, the server RPM runs mysql_install_
db.
   *      If you install MySQL on Mac OS X using a PKG distribution, the installer runs mysql_insta
ll_db.
```

Otherwise, you will need to run mysql_install_db yourself.

The following procedure describes how to initialize the grant tables (if that has not previously been done) and then start the server. It also suggests some commands that you can use to test whether the server is accessible and working properly. For information about starting and stopping the server automatically, see Section 2.3.15.2.2, “Starting and Stopping MySQL Automatically”.

After you complete the procedure and have the server running, you should assign passwords to the accounts created by mysql_install_db. Instructions for doing so are given in Section 2.3.15.3, “Securing the Initial MySQL Accounts”.

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.

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 are located in this directory. 2.If necessary, run the mysql_install_db program to set up the initial MySQL grant tables containing the privileges that determine how users are allowed to connect to the server. You'll need to do this if you used a distribution type for which the installation procedure doesn't run the program for you.

Typically, mysql_install_db needs to be run only the first time you install MySQL, so you can skip this step if you are upgrading an existing installation, However, mysql_install_db does not overwrite any existing privilege tables, so it should be safe to run in any circumstances.

To initialize the grant tables, use one of the following commands, depending on whether mysql_install_db is located in the bin or scripts directory:

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

The mysql_install_db script creates the server's data directory. Under the data directory, it creates directories for the mysql database that holds all database privileges and the test database that you can use to test MySQL. The script also creates privilege table entries for root and anonymous-user accounts. The accounts have no passwords initially. A description of their initial privileges is given in Section 2.3.15.3, “Securing the Initial MySQL Accounts”. Briefly, these privileges allow the MySQL root user to do anything, and allow anybody to create or use databases with a name of test or starting with test_.

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, the --user option should be used as shown if you run mysql_install_db as root. Otherwise, you should execute the script while logged in as mysql, in which case you can omit the --user option from the command.

mysql_install_db creates several tables in the mysql database, including user, db, host, tables_priv, columns_priv, func, and others. See Section 5.8, “The MySQL Access Privilege System”, for a complete listing and description of these tables.

If you don't want to have the test database, you can remove it with mysqladmin -u root drop test after starting the server.

If you have trouble with mysql_install_db at this point, see Section 2.3.15.2.1, “Problems Running mysql_install_db”. 3. Start the MySQL server:

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

It is important that the MySQL server be run using an unprivileged (non-root) login account. To ensure this, the --user option should be used as shown if you run mysql_safe as system root. Otherwise, you should execute the script while logged in to the system as mysql, in which case you can omit the --user option from the command.

Further instructions for running MySQL as an unprivileged user are given in Section 5.7.5, “How to Run MySQL as a Normal User”.

If you neglected to create the grant tables before proceeding to this step, the following message appears in the error log file when you start the server:

```
mysqld: Can't find file: 'host.frm'
```

If you have other problems starting the server, see Section 2.3.15.2.3, “Starting and Troubleshooting the MySQL Server”.

```
  4.     Use mysqladmin to verify that the server is running. The following commands provide simple
tests to check whether the server is up and responding to connections:
```

```
shell> bin/mysqladmin version
shell> bin/mysqladmin variables
```

The output from mysqladmin version varies slightly depending on your platform and version of MySQL, but should be similar to that shown here:

```
shell> bin/mysqladmin version
mysqladmin  Ver 14.12 Distrib 5.0.28, for pc-linux-gnu on i686
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version          5.0.28-Max
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 14 days 5 hours 5 min 21 sec

Threads: 1  Questions: 366  Slow queries: 0
Opens: 0  Flush tables: 1  Open tables: 19
Queries per second avg: 0.000
```

To see what else you can do with mysqladmin, invoke it with the --help option. 5. Verify that you can shut down the server:

```
shell> bin/mysqladmin -u root shutdown
```

6. Verify that you can start the server again. Do this by using mysqld_safe or by invoking mysqld directly. For example:

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

If mysqld_safe fails, see Section 2.3.15.2.3, “Starting and Troubleshooting the MySQL Server”. 7. Run some simple tests to verify that you can retrieve information from the server. The output should be similar to what is shown here:

```
shell> bin/mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+
```

```
shell> bin/mysqlshow mysql
Database: mysql
+---------------------------+
|          Tables           |
+---------------------------+
| columns_priv              |
| db                        |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| proc                      |
| procs_priv                |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
```

```
shell> bin/mysql -e "SELECT Host,Db,User FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
```

```
    | %    | test   |      |
    | %    | test_% |      |
    +------+--------+------+
```

8. There is a benchmark suite in the sql-bench directory (under the MySQL installation directory) that you can use to compare how MySQL performs on different platforms. The benchmark suite is written in Perl. It requires the Perl DBI module that provides a database-independent interface to the various databases, and some other additional Perl modules:

```
    DBI
    DBD::mysql
    Data::Dumper
    Data::ShowTable
```

These modules can be obtained from CPAN (http://www.cpan.org/). See also Section 2.3.19.1, “Installing Perl on Unix”.

The sql-bench/Results directory contains the results from many runs against different databases and platforms. To run all tests, execute these commands:

```
    shell> cd sql-bench
    shell> perl run-all-tests
```

If you don't have the sql-bench directory, you probably installed MySQL using RPM files other than the source RPM. (The source RPM includes the sql-bench benchmark directory.) In this case, you must first install the benchmark suite before you can use it. There are separate benchmark RPM files named mysql-bench-VERSION-i386.rpm that contain benchmark code and data.

If you have a source distribution, there are also tests in its tests subdirectory that you can run. For example, to run auto_increment.tst, execute this command from the top-level directory of your source distribution:

```
    shell> mysql -vvf test < ./tests/auto_increment.tst
```

The expected result of the test can be found in the ./tests/auto_increment.res file. 9. At this point, you should have the server running. However, none of the initial MySQL accounts have a password, so you should assign passwords using the instructions found in Section 2.3.15.3, “Securing the Initial MySQL Accounts”.

The MySQL 5.0 installation procedure creates time zone tables in the mysql database. However, you must populate the tables manually using the instructions in Section 5.11.8, “MySQL Server Time Zone Support”. 2.3.15.2.1. Problems Running mysql_install_db

This section does not apply to MySQL Enterprise Server users.

The purpose of the mysql_install_db script is to generate new MySQL privilege tables. It does not overwrite existing MySQL privilege tables, and it does not affect any other data.

If you want to re-create your privilege tables, first stop the mysqld server if it's running. Then rename the mysql directory under the data directory to save it, and then 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 didn't start. If you do not understand what happened, include the log when you post a bug report. See Section 1.8, “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 install MySQL the first time).

```
    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 Section 5.13, “Running Multiple MySQL Ser
vers on the Same 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), 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 pathname 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 &
```

If mysql_install_db is located in the scripts directory, modify the first command to scripts/mysql_install_db.

See Section B.4.5, “How to Protect or Change the MySQL Unix Socket File”, and Appendix I, Environment Variables.

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

*

If you want the initial privileges to be different from the standard defaults, you can modify mysql_install_db before you run it. However, it is preferable to use GRANT and REVOKE to change the privileges after the grant tables have been set up. In other words, you can 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 GRANT and REVOKE statements.

If you want to install MySQL on several machines with the same privileges, you can put the 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
```

By doing this, you can avoid having to issue 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're just learning how to use 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, remove all the .frm, .MYI, and .MYD files in the mysql database directory. Then run the mysql_install_db script again.

*

You can start mysqld manually using the --skip-grant-tables option and add the privilege information yourself using mysql:

```
shell> bin/mysqld_safe --user=mysql --skip-grant-tables &
shell> bin/mysql mysql
```

From mysql, manually execute the SQL commands contained in mysql_install_db. Make sure that you run mysqladmin flush-privileges or mysqladmin reload afterward to tell the server to reload the grant tables.

Note that by not using mysql_install_db, you not only have to populate the grant tables manually, you also have to create them first.

#### 2.3.15.2.2. Starting and Stopping MySQL Automatically

Generally, you start the mysqld server in one of these ways:

- By invoking mysqld directly. This works on any platform.
- By running the MySQL server as a Windows service. This can be done on versions of Windows that support services (such as NT, 2000, XP, and 2003). The service can be set to start the server automatically when Windows starts, or as a manual service that you start on request. For instructions, see Section 2.3.8.11, “Starting MySQL as a Windows Service”.
- By invoking mysqld_safe, which tries to determine the proper options for mysqld and then runs it with those options. This script is used on Unix and Unix-like systems. See Section 5.4.1, “mysqld_safe — MySQL Server Startup Script”.
- By invoking mysql.server. This script is used primarily at system startup and shutdown on systems that use System V-style run directories, where it usually is installed under the name mysql. The mysql.server script starts the server by invoking mysqld_safe. See Section 5.4.2, “mysql.server — MySQL Server Startup Script”.
- On Mac OS X, you can install a separate MySQL Startup Item package to enable the automatic startup of MySQL on system startup. The Startup Item starts the server by invoking mysql.server. See Section 2.3.10, “Installing MySQL on Mac OS X”, for details.

The mysqld_safe and mysql.server scripts and the Mac OS X Startup Item can be used to start the server manually, or automatically at system startup time. mysql.server and the Startup Item also can be used to stop the server.

To start or stop the server manually using the mysql.server script, invoke it with start or stop arguments:

shell> mysql.server start shell> mysql.server stop

Before mysql.server starts the server, it changes location to the MySQL installation directory, and then invokes mysqld_safe. If you want the server to run as some specific user, add an appropriate user option to the [mysqld] group of the /etc/my.cnf option file, as shown later in this section. (It is possible that you will need to edit mysql.server if you've installed a binary distribution of MySQL in a non-standard location. Modify it to cd into the proper directory before it runs mysqld_safe. If you do this, your modified version of mysql.server may be overwritten if you upgrade MySQL in the future, so you should make a copy of your edited version that you can reinstall.)

mysql.server stop stops the server by sending a signal to it. You can also stop the server manually by executing mysqladmin shutdown.

To start and stop MySQL automatically on your server, you need to add start and stop commands to the appropriate places in your /etc/rc* files.

If you use the Linux server RPM package (MySQL-server-VERSION.rpm), the mysql.server script is installed in the /etc/init.d directory with the name mysql. You need not install it manually. See Section 2.3.9, “Installing MySQL on Linux”, for more information on the Linux RPM packages.

Some vendors provide RPM packages that install a startup script under a different name such as mysqld.

If you install MySQL from a source distribution or using a binary distribution format that does not install mysql.server automatically, you can install it manually. The script can be found in the support-files directory under the MySQL installation directory or in a MySQL source tree.

To install mysql.server manually, copy it to the /etc/init.d directory with the name mysql, and then make it executable. Do this by changing location into the appropriate directory where mysql.server is located and executing these commands:

shell> cp mysql.server /etc/init.d/mysql shell> chmod +x /etc/init.d/mysql

Older Red Hat systems use the /etc/rc.d/init.d directory rather than /etc/init.d. Adjust the preceding commands accordingly. Alternatively, first create /etc/init.d as a symbolic link that points to /etc/rc.d/init.d:

shell> cd /etc shell> ln -s rc.d/init.d .

After installing the script, the commands needed to activate it to run at system startup depend on your operating system. On Linux, you can use chkconfig:

shell> chkconfig --add mysql

On some Linux systems, the following command also seems to be necessary to fully enable the mysql script:

shell> chkconfig --level 345 mysql on

On FreeBSD, startup scripts generally should go in /usr/local/etc/rc.d/. The rc(8) manual page states that scripts in this directory are executed only if their basename matches the *.sh shell filename pattern. Any other files or directories present within the directory are silently ignored. In other words, on FreeBSD, you should install the mysql.server script as /usr/local/etc/rc.d/mysql.server.sh to enable automatic startup.

As an alternative to the preceding setup, some operating systems also use /etc/rc.local or /etc/init.d/boot.local to start additional services on startup. To start up MySQL using this method, you could append a command like the one following to the appropriate startup file:

/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'

For other systems, consult your operating system documentation to see how to install startup scripts.

You can add options for mysql.server in a global /etc/my.cnf file. A typical /etc/my.cnf file might look like this:

[mysqld] datadir=/usr/local/mysql/var socket=/var/tmp/mysql.sock port=3306 user=mysql

[mysql.server] basedir=/usr/local/mysql

The mysql.server script understands the following options: basedir, datadir, and pid-file. If specified, they must be placed in an option file, not on the command line. mysql.server understands only start and stop as command-line arguments.

The following table shows which option groups the server and each startup script read from option files: Script Option Groups mysqld [mysqld], [server], [mysqld-major_version] mysqld_safe [mysqld], [server], [mysqld_safe] mysql.server [mysqld], [mysql.server], [server]

[mysqld-major_version] means that groups with names like [mysqld-4.1] and [mysqld-5.0] are read by servers having versions 4.1.x, 5.0.x, and so forth. This feature can be used to specify options that can be read only by servers within a given release series.

For backward compatibility, mysql.server also reads the [mysql_server] group and mysqld_safe also reads the [safe_mysqld] group. However, you should update your option files to use the [mysql.server] and [mysqld_safe] groups instead when using MySQL 5.0.

See Section 4.3.2, “Using Option Files”. 2.3.15.2.3. Starting and Troubleshooting the MySQL Server

This section provides troubleshooting suggestions for problems starting the server on Unix. If you are using Windows, see Section 2.3.8.13, “Troubleshooting a MySQL Installation Under Windows”.

If you have problems starting the server, here are some things to try:

*

Check the error log to see why the server does not start.

*

Specify any special options needed by the storage engines you are using.

*

Make sure that the server knows where to find the data directory.

*

Make sure that the server can access the data directory. The ownership and permissions of the data directory and its contents must be set such that the server can read and modify them.

*

Verify that the network interfaces the server wants to use are available.

Some storage engines have options that control their behavior. You can create a my.cnf file and specify startup options for the engines that you plan to use. If you are going to use storage engines that support transactional tables (InnoDB, BDB, NDB), be sure that you have them configured the way you want before starting the server:

*

If you are using InnoDB tables, see Section 14.2.3, “InnoDB Configuration”.

*

If you are using BDB (Berkeley DB) tables, see Section 14.5.3, “BDB Startup Options”.

*

If you are using MySQL Cluster, see Section 15.4, “MySQL Cluster Configuration”.

Storage engines will use default option values if you specify none, but it is recommended that you review the available options and specify explicit values for those for which the defaults are not appropriate for your installation.

When the mysqld server starts, it changes location to the data directory. This is where it expects to find databases and where it expects to write log files. The server also writes the pid (process ID) file in the data directory.

The data directory location is hardwired in when the server is compiled. This is where the server looks for the data directory by default. If the data directory is located somewhere else on your system, the server will not work properly. You can determine what the default path settings are by invoking mysqld with the --verbose and --help options.

If the default locations don't match the MySQL installation layout on your system, you can override them by specifying options to mysqld or mysqld_safe on the command line or in an option file.

To specify the location of the data directory explicitly, use the --datadir option. However, normally you can tell mysqld the location of the base directory under which MySQL is installed and it looks for the data directory there. You can do this with the --basedir option.

To check the effect of specifying path options, invoke mysqld with those options followed by the --verbose and --help options. For example, if you change location into the directory where mysqld is installed and then run the following command, it shows the effect of starting the server with a base directory of /usr/local:

shell> ./mysqld --basedir=/usr/local --verbose --help

You can specify other options such as --datadir as well, but --verbose and --help must be the last options.

Once you determine the path settings you want, start the server without --verbose and --help.

If mysqld is currently running, you can find out what path settings it is using by executing this command:

shell> mysqladmin variables

Or:

shell> mysqladmin -h host_name variables

host_name is the name of the MySQL server host.

If you get Errcode 13 (which means Permission denied) when starting mysqld, this means that the privileges of the data directory or its contents do not allow the server access. In this case, you change the permissions for the involved files and directories so that the server has the right to use them. You can also start the server as root, but this raises security issues and should be avoided.

On Unix, change location into the data directory and check the ownership of the data directory and its contents to make sure the server has access. For example, if the data directory is /usr/local/mysql/var, use this command:

shell> ls -la /usr/local/mysql/var

If the data directory or its files or subdirectories are not owned by the login account that you use for running the server, change their ownership to that account. If the account is named mysql, use these commands:

shell> chown -R mysql /usr/local/mysql/var shell> chgrp -R mysql /usr/local/mysql/var

If the server fails to start up correctly, check the error log. Log files are located in the data directory (typically C:\Program Files\MySQL\MySQL Server 5.0\data on Windows, /usr/local/mysql/data for a Unix binary distribution, and /usr/local/var for a Unix source distribution).

Look in the data directory for files with names of the form host_name.err and host_name.log, where host_name is the name of your server host. Then examine the last few lines of these files. On Unix, you can use tail to display them:

shell> tail host_name.err shell> tail host_name.log

The error log should contain information that indicates why the server couldn't start. For example, you might see something like this in the log:

000729 14:50:10 bdb: Recovery function for LSN 1 27595 failed 000729 14:50:10 bdb: warning: ./test/t1.db: No such file or directory 000729 14:50:10 Can't init databases

This means that you did not start mysqld with the --bdb-no-recover option and Berkeley DB found something wrong with its own log files when it tried to recover your databases. To be able to continue, you should move the old Berkeley DB log files from the database directory to some other place, where you can later examine them. The BDB log files are named in sequence beginning with log.0000000001, where the number increases over time.

If you are running mysqld with BDB table support and mysqld dumps core at startup, this could be due to problems with the BDB recovery log. In this case, you can try starting mysqld with --bdb-no-recover. If that helps, you should remove all BDB log files from the data directory and try starting mysqld again without the --bdb-no-recover option.

If either of the following errors occur, it means that some other program (perhaps another mysqld server) is using the TCP/IP port or Unix socket file that mysqld is trying to use:

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

Use ps to determine whether you have another mysqld server running. If so, shut down the server before starting mysqld again. (If another server is running, and you really want to run multiple servers, you can find information about how to do so in Section 5.13, “Running Multiple MySQL Servers on the Same Machine”.)

If no other server is running, try to execute the command telnet your_host_name tcp_ip_port_number. (The default MySQL port number is 3306.) Then press Enter a couple of times. If you don't get an error message like telnet: Unable to connect to remote host: Connection refused, some other program is using the TCP/IP port that mysqld is trying to use. You'll need to track down what program this is and disable it, or else tell mysqld to listen to a different port with the --port option. In this case, you'll also need to specify the port number for client programs when connecting to the server via TCP/IP.

Another reason the port might be inaccessible is that you have a firewall running that blocks connections to it. If so, modify the firewall settings to allow access to the port.

If the server starts but you can't connect to it, you should make sure that you have an entry in /etc/hosts that looks like this:

127.0.0.1 localhost

This problem occurs only on systems that do not have a working thread library and for which MySQL must be configured to use MIT-pthreads.

If you cannot get mysqld to start, you can try to make a trace file to find the problem by using the --debug option. See Section H.1.2, “Creating Trace Files”. Previous / Next / Up / Table of Contents

User Comments Posted by [name withheld] on June 8 2004 9:49pm [Delete] [Edit]

You must run mysqld_install_db as the user that mysqld will run under, for example:

shell> su mysql -c mysql_install_db

Otherwise, the daemon will not be able to look in the mysql directory, and will not be able to find the file 'host.frm' inside it. Posted by A. Johan on July 16 2004 2:38pm [Delete] [Edit]

if you didn't run the mysql_install_db as the mysql user and you are unable to start the mysql daemon with an error such as "Can't find file ... host.frm", all you need to check is see if the dir and files belong to the user mysql.

/var/lib/mysql/mysql and files in it should belong to the user mysql.

just do chown and you should be ok.

cheers.

aj Posted by Sheldon Plankton on August 25 2004 4:29pm [Delete] [Edit]

I am confused Step 6 says ...

6. Verify that you can restart the server. Do this by using mysqld_safe or by invoking mysqld directly. For example:

shell> bin/mysqld_safe --user=mysql --log &

If mysqld_safe fails, see section 2.4.2.3 Starting and Troubleshooting the MySQL Server.

... is it mysqld_safe or safe_mysqld ??? Posted by José Manuel Ciges Regueiro on December 3 2004 10:04am [Delete] [Edit]

In opposition at what is said in the docs the minimun version of the Perl module MySQL::DBD for MySQL 4.1 is the 2.9004.

The compilation of MySQL::DBD version 2.9003 gives the following error: mysql.xs: In function `XS_DBD__mysql__dr__admin_internal': mysql.xs:100: too few arguments to function `mysql_shutdown'

This is because from MySQL version 4.1.3 the function mysql_shutdown has an extra parameter.

As I told before the problem is solved in the version 2.9004. Posted by [name withheld] on January 26 2005 8:57pm [Delete] [Edit]

Thanks to [name withheld] for the advice on running the mysql_install_db script as the mysql user! This appears to be **crucial**, but I didn't find that tip anywhere but in her/his comment. Posted by [name withheld] on February 6 2005 3:20pm [Delete] [Edit]

Have your server up and running but can't get a connection to it over your network? Try commenting out the poorly documented "bind-address = 127.0.0.1" in my.cnf.

While you're there, make sure that "skip-networking" is commented out. Posted by Todd Trimble on February 11 2005 3:54pm [Delete] [Edit]

It's worth noting that the package install for OpenBSD creates the "_mysql" user account.

One little underscore that can really confuse readers... Posted by Coyote Osborne on April 6 2005 9:02pm [Delete] [Edit]

I just spent what would have been a hair-pulling two hours (if I had hair) trying to install mySQL on a mac running osx 10.3.8

I downloaded the nice "easy" installer package from this site, and ran both that installer and the startup item installer. The install seemed to go fine, however, the mysql_install_db would not set up the access tables no matter what I did.

My solution was to activate the root account via netinfo manager, and do a complete reinstall as root, and then manually run mysql_install_db via su as root. Everything worked flawlessly after that. I then deactivated the root account (for the system, not mySQL). I try not to leave a root account active on these systems.

I'm only mentioning this simply because I couldn't find any tips that worked for me, and this seemed to solve the problem. Posted by Andy Canfield on May 2 2005 6:05am [Delete] [Edit]

I had a problem with step 1: 'cd BASEDIR'. My MySQL was installed with the Linux system, so where is BASEDIR? I studied the script "mysql_install_db" and figured this out: Use 'locate' or 'find' for the file "my_print_defaults". That should be BASEDIR/bin/print_my_defaults or BASEDIR/extra/print_my_defaults. In my case BASEDIR was "/usr". Posted by Matthew Bennett on May 14 2005 12:10pm [Delete] [Edit]

Under Unix/Linux, if you have already run mysql_install_db as root, you may find that running it as the mysql user still doesn't fix the problem - mainly because, as mentioned above, the mysql_install_db utility won't write over databases that are already there (and probably wouldn't be able to even if it tried)

The solution then is to go to wherever your databases are stored (mine are in /var/lib/mysql) and, as root, type:

chown -R mysqluser mysql

where "mysqluser" is the name of whatever user uses the database (most systems have a "mysql" user specifically set up for this purpose) and "mysql" refers to the directory (in my case, /var/lib/mysql/mysql)

Note: I'm no security expert, I'm only using mysql on a small test system... this may be unsafe to do in a deployment system. If someone could advise, please do...)

Posted by Kees Hink on June 22 2005 11:47am [Delete] [Edit]

On OpenBSD, after running

bin/mysqld_safe --user=_mysql &

mysql used to stop immediately. Logfiles showed the message

050622 13:29:53 mysqld started 050622 13:29:53 Can't start server : Bind on unix socket: Permission denied 050622 13:29:53 Do you already have another mysqld server running on socket: /var/run/mysql/mysql.sock ?

Turns out root was owner of /var/run/mysql, so i turned ownership over to _mysql:

chown _mysql /var/run/mysql/ Posted by Keisy bui on October 17 2005 12:30am [Delete] [Edit]

Restart MySQL

I installed MySQL-server-4.1.14-0.glibc23.i386.rpm on Linux. After running

rpm -i MySQL-server-4.1.....rpm

the server started straight away. Then i realised that mysql is installed in /usr and /usr was my BASEDIR.

Then under /usr, i set up the password for the root:

/bin/mysqladmin -u root password 'somepass'

Then i tried to view all databases using

bin/mysqlshow

i got the access denied message, then i tried:

bin/mysqlshow -u root -p

the system then asked me for the password, then 2 databases mysql and test showed up :)

After that, i decided to stop and restart MySQL bin/mysqladmin -u root shutdown -p (and typed in the password)

bin/safe_mysqld -user root -p somepass

but i could not restart it, i dont know if the command was incorrect.

then i did a series of chown:

cd /var/lib/mysql chown -R root mysql chown -R myusername mysql chown -R mysql mysql

I then restarted MySQL using:

cd /usr bin/safe)mysqld --user=root &

and the server started.

Note: so far i run everything as 'root' user of the OS.

Posted by Fausto Rodriguez Zapata on November 18 2005 1:26pm [Delete] [Edit]

After installing the mysql-debug-5.0.15-osx10.4-powerpc.pkg package and running the mysql server through installed preference pane I got the following error when I tried to set the anonymous passwords:

shell> mysql -u root Access denied for user 'root'@'localhost' (using password: NO)

So I first followed the directions in A.4.1. How to Reset the Root Password to reset the root password and then I modified the anonymous and root passwords as in this section. Posted by Cameron Spitzer on April 16 2006 10:49pm [Delete] [Edit]

Watch out when you run perl run-all-tests. I got a 270 MB log file. You might want to turn query logging off. Posted by Shane Crawford on May 9 2006 2:45am [Delete] [Edit]

OS X: 10.4.6

Installed via Darwin Ports ('port install mysql5')

$sudo /opt/local/lib/mysql5/bin/mysql_install_db --user=mysql

$cd /opt/local/var/db

$ ls -l drwx------ 9 mysql admin 306 May 8 21:19 mysql5

$cd /opt/local/share/mysql5/mysql

$sudo ./mysql.server start

$mysqladmin ping mysqld is alive

Posted by [name withheld] on August 15 2006 9:04pm [Delete] [Edit] Using RedHat Enterprise Linux 4:

You need to disable selinux to start the service. So do the following:

1. /usr/sbin/setenforce 0 2. /etc/init.d/mysql start 3. /usr/sbin/setenforce 1

---

Revision #3
Created 2026-04-13 18:27:52 CEST by Philip
Updated 2026-04-13 19:31:34 CEST by Philip