Our customers have approximately 120,000 hands.
That's a lot of high-fives.
zendesk
PUT 'ER THERE

LHN PDFs Click to Purchase

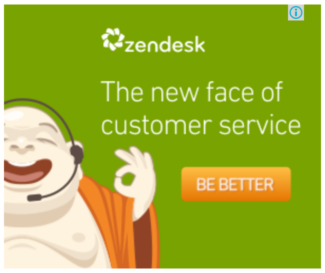
LHN

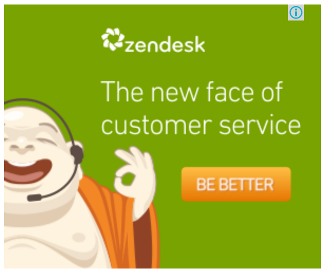
(c) Peter Harrison

HomePurchase PDFsAbout

G+1 735

Search





Quick HOWTO : Ch34 : Basic MySQL Configuration

From Linux Home Networking

Contents

- 1 Introduction
- 2 Preparing MySQL For Applications
- 3 Download and Install The MySQL Packages
- 4 Debian / Ubuntu Differences
- 5 Starting MySQL
- 6 The /etc/my.cnf File
- 7 The Location of MySQL Databases
- 8 Creating a MySQL "root" Account
- 9 Accessing The MySQL Command Line
- 10 Granting Privileges to Users
- 11 Running MySQL Scripts To Create Data Tables
- 12 Viewing Your New MySQL Databases
  - 12.1 Listing The Data Tables In Your MySQL Database
  - 12.2 Viewing Your MySQL Database's Table Structure
  - 12.3 Viewing The Contents Of A Table
- 13 Configuring Your Application
  - 13.1 Table 34.1 Required PHP and Perl RPMs for MySQL Support
- 14 Recovering / Changing Your MySQL Root Password
- 15 MySQL Database Backup
- 16 MySQL Database Restoration
- 17 MySQL Table Backup and Restoration
- 18 Very Basic MySQL Network Security
- 19 Basic MyQL Troubleshooting
  - 19.1 Connectivity Testing
  - 19.2 Test Database Access
- 20 A Common Fedora Core 1 MySQL Startup Error
- 21 Conclusion

Introduction

Most home/SOHO administrators don't do any database programming, but they sometimes need to install applications that require a MySQL database. This chapter explains the basic steps of configuring MySQL for use with a MySQL-based application in which the application runs on the same server as the database.

Preparing MySQL For Applications

In most cases the developers of database applications expect the systems administrator to be able to independently prepare a database for their applications to use. The steps to do this include:

Other Linux Home Networking Topics

- Introduction to Networking
- Linux Networking
- Simple Network Troubleshooting
- Troubleshooting Linux with Syslog
- Installing Linux Software
- The Linux Boot Process
- Configuring the DHCP Server
- Linux Users and sudo
- Windows, Linux and Samba
- Sharing Resources with Samba
- Samba Security and Troubleshooting
- Linux Wireless Networking
- Linux Firewalls Using iptables
- Linux FTP Server Setup
- Telnet, TFTP and xinetd
- Secure Remote Logins and File Copying
- Configuring DNS
- Dynamic DNS
- The Apache Web Server
- Configuring Linux Mail Servers
- Monitoring Server Performance
- Advanced MRTG For Linux
- The NTP Server
- Network-Based Linux Installation
- Linux Software RAID
- Expanding Disk Capacity
- Managing Disk Usage with Quotas
- Remote Disk Access with NFS

1. Install and start MySQL.
2. Create a MySQL "root" user.
3. Create a regular MySQL user that the application will use to access the database.
4. Create your application's database.
5. Create your database's data tables.
6. Perform some basic tests of your database structure.

The rest of the chapter is based on a scenario in which a Linux-based application named sales-test needs to be installed. After reading the sales-test manuals, you realize that you have to create a MySQL database, data tables, and a database user before you can start the application. Fortunately sales-test comes with a script to create the tables, but you have to do the rest yourself. Finally, as part of the planning for the installation, you decided to name the database salesdata and let the application use the MySQL user mysqluser to access it.

I'll cover all these common tasks in detail in the remaining sections.

Download and Install The MySQL Packages

In most cases you'll probably want to install the MySQL server and MySQL client software packages. The client package gives you the ability to test the server connection and can be used by any MySQL application to communicate with the server, even if the server software is running on the same Linux box.

Note: With Fedora / Redhat the packages to install would be mysql-server and mysql and with Debian / Ubuntu the packages are mysql-server and mysql-client.

Most RedHat and Fedora Linux software product packages are available in the RPM format, whereas Debian and Ubuntu Linux use DEB format installation files. When searching for these packages, remember that the filename usually starts with the software package name and is followed by a version number, as in mysql-server-3.23.58-4.i386.rpm. (For help on downloading and installing the required packages, see Chapter 6, Installing Linux Software).

Debian / Ubuntu Differences

This chapter focuses on Fedora / CentOS / RedHat for simplicity of explanation. Whenever there is a difference in the required commands for Debian / Ubuntu variations of Linux it will be noted.

The universal difference is that the commands shown are done by the Fedora / CentOS / RedHat root user. With Debian / Ubuntu you will either have to become root using the "sudo su –" command or you can temporarily increase your privilege level to root using the "sudo <command>" command.

Here is an example of how to permanently become root:

```
user@ubuntu:~$ sudo su -
[sudo] password for peter:
root@ubuntu:~#
```

Here is an example of how to temporarily become root to run a specific command. The first attempt to get a directory listing fails due to insufficient privileges. The second attempt succeeds when the sudo keyword is inserted before the command.

```
user@ubuntu:~$  ls -l /var/lib/mysql/mysql
ls: cannot access /var/lib/mysql/mysql: Permission denied
user@ubuntu:~$ sudo ls -l /var/lib/mysql/mysql
[sudo] password for peter:
total 964
-rw-rw---- 1 mysql mysql   8820 2010-12-19 23:09 columns_priv.frm
-rw-rw---- 1 mysql mysql      0 2010-12-19 23:09 columns_priv.MYD
-rw-rw---- 1 mysql mysql   4096 2010-12-19 23:09 columns_priv.MYI
-rw-rw---- 1 mysql mysql   9582 2010-12-19 23:09 db.frm
...
...
```

- Configuring NIS
- Centralized Logins Using LDAP and RADIUS
- Controlling Web Access with Squid
- Modifying the Kernel to Improve Performance
- Basic MySQL Configuration

```
...
user@ubuntu:~$
```

Now that you have got this straight, let’s continue with the discussion.

Starting MySQL

The methodologies vary depending on the variant of Linux you are using as you’ll see next.

Fedora / CentOS / RedHat

With these flavors of Linux you can use the chkconfig command to get mysqld configured to start at boot:

```
[root@bigboy tmp]# chkconfig mysqld on
```

To start, stop, and restart mysqld after booting use the service command:

```
[root@bigboy tmp]# service mysqld start
[root@bigboy tmp]# service mysqld stop
[root@bigboy tmp]# service mysqld restart
```

To determine whether mysqld is running you can issue either of these two commands. The first will give a status message. The second will return the process ID numbers of the mysqld daemons.

```
[root@bigboy tmp]# service mysqld status
[root@bigboy tmp]# pgrep mysql
```

Note: Remember to run the chkconfig command at least once to ensure mysqld starts automatically on your next reboot.

Ubuntu / Debian

With these flavors of Linux the commands are different. Try installing the sysv-rc-conf and sysvinit-utils DEB packages as they provide commands that simplify the process. (For more on downloading and installing RPMs, see Chapter 6, "Installing Linux Software").

You can use the sysv-rc-conf command to get mysqld configured to start at boot:

```
user@ubuntu:~$ sudo sysv-rc-conf mysql on
```

To start, stop, and restart mysqld after booting the service command is the same:

```
user@ubuntu:~$ sudo service mysql start
user@ubuntu:~$ sudo service mysql stop
user@ubuntu:~$ sudo service mysql restart
```

To determine whether mysqld is running you can issue either of these two commands. The first will give a status message. The second will return the process ID numbers of the mysqld daemons.

```
user@ubuntu:~$ sudo service mysql status
user@ubuntu:~$ pgrep mysql
```

Note: Remember to run the sysv-rc-conf command at least once to ensure mysql starts automatically on your next reboot.

The /etc/my.cnf File

You can define most of MySQL’s configuration parameters in the my.cnf file which may be located in either the /etc or /etc/mysql directory depending on your version of Linux.

Note: Your Linux distribution may only allow MySQL server to listen on the 127.0.0.1 localhost address. This is sufficient if the database and the application that uses it is on the same server. To allow access from remote clients you will have to let it listen to the IP address of your network interface. The bind-address directive in the [mysqld] section of the my.cnf file governs this action. In this case MySQL is listening on the address 192.168.1.100.

```
#
# File: my.cnf
#
[mysqld]
bind-address            = 192.168.1.100
```

Comment out or remove the bind-address directive if you want MySQL to listen on all IP addresses of your server. Do so only if your server resides on a secure network to reduce the risk of a malicious attack.

Note: Remember to restart MySQL after you make any changes to your configuration files. This is the only way to activate the new settings.

The Location of MySQL Databases

According to the /etc/my.cnf file, MySQL databases are usually located in a subdirectory of the /var/lib/mysql/ directory. If you create a database named test, then the database files will be located in the directory /var/lib/mysql/test.

Creating a MySQL "root" Account

MySQL stores all its username and password data in a special database named mysql. You can add users to this database and specify the databases to which they will have access with the grant command. The MySQL root or superuser account, which is used to create and delete databases, is the exception. You need to use the mysqladmin command to set your root password. Only two steps are necessary for a brand new MySQL installation.

1. Make sure MySQL is started.
2. Use the mysqladmin command to set the MySQL root password. The syntax is as follows:

```
[root@tmp bigboy]# mysqladmin -u root password new-password
```

If you want to change your password later, you will probably have to do a root password recovery.

Accessing The MySQL Command Line

MySQL has its own command line interpreter (CLI). You need to know how to access it to do very basic administration.

You can access the MySQL CLI using the mysql command followed by the -u option for the username and -p, which tells MySQL to prompt for a password. Here user root gains access:

```
[root@bigboy tmp]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 3.23.58

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

mysql>
```

Note: Almost all MySQL CLI commands need to end with a semi-colon. Even the exit command used to get back to the Linux prompt needs one too!

Creating and Deleting MySQL Databases

Many Linux applications that use MySQL databases require you to create the database beforehand using the name of your choice. The procedure is relatively simple: Enter the MySQL CLI, and use the create database command:

```
mysql> create database salesdata;
Query OK, 1 row affected (0.00 sec)

mysql>
```

If you make a mistake during the installation process and need to delete the database, use the drop database command. The example deletes the newly created database named salesdata.

```
mysql> drop database salesdata;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Note: Sometimes a dropped database may still appear listed when you use the show databases command explained further below. This may happen even if your root user has been granted full privileges to the database, and it is usually caused by the presence of residual database files in your database directory. In such a case you may have to physically delete the database sub-directory in /var/lib/mysql from the Linux command line. Make sure you stop MySQL before you do this.

```
[root@bigboy tmp]# service mysqld stop
```

Granting Privileges to Users

On many occasions you will not only have to create a database, but also have to create a MySQL username and password with privileges to access the database. It is not a good idea to use the root account to do this because of its universal privileges.

MySQL stores all its username and password data in a special database named mysql. You can add users to this database and specify the databases to which they will have access with the grant command, which has the syntax.

```
sql> grant all privileges on database.* to username@"servername" identified by 'password';
```

So you can create a user named mysqluser with a password of pinksl1p to have full access to the database named salesdata on the local server (localhost) with the grant command. If the database application's client resides on another server, then you'll want to replace the localhost address with the actual IP address of that client.

```
sql> grant all privileges on salesdata.* to mysqluser@"localhost" identified by 'pinksl1p';
```

The next step is to write the privilege changes to the mysql.sql database using the flush privileges command.

```
sql> flush privileges;
```

Running MySQL Scripts To Create Data Tables

Another common feature of prepackaged applications written in MySQL is that they may require you to not only create the database, but also to create the tables of data inside them as part of the setup procedure. Fortunately, many of these applications come with scripts you can use to create the data tables automatically.

Usually you have to run the script by logging into MySQL as the MySQL root user and automatically importing all the script file's commands with a < on the command line.

The example runs a script named create_mysql.script whose commands are applied to the newly created database named salesdata. MySQL prompts for the MySQL root password before completing the transaction. (You have to create the database first, before you can run this command successfully.)

```
[root@bigboy tmp]# mysql -u root -p salesdata < create_mysql.script
Enter password:
[root@bigboy tmp]#
```

Viewing Your New MySQL Databases

A number of commands can provide information about your newly created database. Here are some examples:

- **Login As The Database User**: It is best to do all your database testing as the MySQL user you want the application to eventually use. This will make your testing mimic the actions of the application and results in better testing in a more production-like environment than using the root account.

```
[root@bigboy tmp]# mysql -u mysqluser -p salesdata
```

- **List all your MySQL databases**: The show databases command gives you a list of all your available MySQL databases. In the example, you can see that the salesdata database has been successfully created:

```
mysql> show databases;
+-----------+
| Database  |
+-----------+
| salesdata |
+-----------+
1 row in set (0.00 sec)

mysql>
```

Listing The Data Tables In Your MySQL Database

The show tables command gives you a list of all the tables in your MySQL database, but you have to use the use command first to tell MySQL to which database it should apply the show tables command.

The example uses the salesdata database; notice that it has a table named test.

```
mysql> use salesdata;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------+
| Tables_in_salesdata |
+---------------------+
| test                |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

Viewing Your MySQL Database's Table Structure

The describe command gives you a list of all the data fields used in your database table. In the example, you can see that the table named test in the salesdata database keeps track of four fields: name, description, num, and date_modified.

```
mysql> describe test;
+---------------+-------------+------+-----+------------+----------------+
| Field         | Type        | Null | Key | Default    | Extra          |
+---------------+-------------+------+-----+------------+----------------+
| num           | int(11)     |      | PRI | NULL       | auto_increment |
| date_modified | date        |      | MUL | 0000-00-00 |                |
| name          | varchar(50) |      | MUL |            |                |
| description   | varchar(75) | YES  |     | NULL       |                |
+---------------+-------------+------+-----+------------+----------------+
6 rows in set (0.00 sec)

mysql>
```

Viewing The Contents Of A Table

You can view all the data contained in the table named test by using the select command. In this example you want to see all the data contained in the very first row in the table.

```
mysql> select * from test limit 1;
```

With a brand new database this will give a blank listing, but once the application starts and you enter data, you may want to run this command again as a rudimentary database sanity check.

Configuring Your Application

After creating and testing the database, you need to inform your application of the database name, the IP address of the database client server, and the username and password of the application's special MySQL user that will be accessing the data.

Frequently this registration process is done by the editing of a special application-specific configuration file either via a Web GUI or from the command line. Read your application's installation guide for details.

You should always remember that MySQL is just a database that your application will use to store information. The application may be written in a variety of languages with Perl and PHP being the most popular. The base PHP and Perl RPMs are installed with Fedora Linux by default, but the packages used by these languages to talk to MySQL are not. You should also ensure that you install the RPMs listed in Table 34.1 on your MySQL clients to ensure compatibility. Use the yum utility discussed in Chapter 6, "Installing Linux Software", if you are uncertain of the prerequisite RPMs needed.

Table 34.1 Required PHP and Perl RPMs for MySQL Support

RPM	**RPM**
php-mysql	MySQL database specific support for PHP
perl-DBI	Provides a generic Perl interface for interacting with relational databases
perl-DBD-MySQL	MySQL database specific support for Perl

Recovering / Changing Your MySQL Root Password

Sometimes you may have to recover the MySQL root password because it was either forgotten or misplaced. The steps you need are:

1) Stop MySQL

```
[root@bigboy tmp]# service mysqld stop
Stopping MySQL:  [  OK  ]
[root@bigboy tmp]#
```

2) Start MySQL in Safe mode with the mysqld_safe command and tell it not to read the grant tables with all the MySQL database passwords.

```
[root@bigboy tmp]# mysqld_safe --skip-grant-tables --skip-networking &
[1] 13007
[root@bigboy tmp]# Starting mysqld daemon with databases from /var/lib/mysql
[root@bigboy tmp]#
```

Note: In Fedora Core 3 and earlier the mysqld_safe command was named safe_mysqld and the general procedure for password recovery was different. This difference is outlined in Appendix III, "Fedora Version Differences".

3) MySQL is now running without password protection. You now have to use the familiar mysql -u root command to get the mysql> command prompt. (-p flag is not required) As expected, you will not be prompted for a password.

```
[root@bigboy tmp]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.16

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

mysql>
```

4) You will now have to use the mysql database which contains the passwords for all the databases on your system and modify the root password. In this case we are setting it to ack33nsaltf1sh.

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> UPDATE user SET Password=PASSWORD("ack33nsaltf1sh") WHERE User="root";
Query OK, 1 row affected (0.00 sec)
Rows matched: 2  Changed: 1  Warnings: 0

mysql>
```

5) Exit MySQL and restart the mysqld daemon.

```
mysql> exit
Bye
[root@bigboy tmp]# service mysqld restart
STOPPING server from pid file /var/run/mysqld/mysqld.pid
051224 17:24:56  mysqld ended

Stopping MySQL:  [  OK  ]
Starting MySQL:  [  OK  ]
[1]+  Done                    mysqld_safe --skip-grant-tables --skip-networking
[root@bigboy tmp]#
```

The MySQL root user will now be able to manage MySQL using this new password.

MySQL Database Backup

The syntax for backing up a MySQL database is as follows:

```
mysqldump --add-drop-table -u [username] -p[password] [database] > [backup_file]
```

In the previous section, you gave user mysqluser full access to the salesdata database when mysqluser used the password pinksl1p. You can now back up this database to a single file called /tmp/salesdata-backup.sql with the command

```
[root@bigboy tmp]# mysqldump --add-drop-table -u mysqluser \
  -ppinksl1p salesdata > /tmp/salesdata-backup.sql
```

Make sure there are no spaces between the -p switch and the password or else you may get syntax errors.

Note: Always backup the database named mysql too, because it contains all the database user access information.

MySQL Database Restoration

The syntax for restoring a MySQL database is:

```
mysql -u [username] -p[password] [database] < [backup_file]
```

So, using the previous example, you can restore the contents of the database with

```
[root@bigboy tmp]# mysql -u mysqluser -ppinksl1p salesdata \
  < /tmp/salesdata-backup.sql
```

Note: You may have to restore the database named mysql also, because it contains all the database user access information.

MySQL Table Backup and Restoration

Sometimes you may want to backup only one or more tables from a database. There are some practical reasons for wanting to do this. You may have a message board / forums application that uses MySQL to store its data and you want to create a brand new forum with the same users as the old one so that the users don't have to register all over again.

The MySQL SELECT statement can be used to export the data to a backup file and the LOAD command can be used to import the data back into the new database used by the new forum. In this example the data in the phpbb_users and phpbb_themes tables of the forums-db-old database are exported to files named /tmp/forums-db-users.sql and /tmp/forums-db-themes.sql respectively. The data is then imported into tables of the same name in the forums-db-new database.

```
mysql> use forums-db-old;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * INTO OUTFILE '/tmp/forums-db-users.sql' FROM phpbb_users;
Query OK, 1042 rows affected (0.03 sec)

mysql> SELECT * INTO OUTFILE '/tmp/forums-db-themes.sql' FROM phpbb_themes;
Query OK, 1038 rows affected (0.03 sec)

mysql> use forums-db-new;
Database changed
mysql> load data infile '/tmp/forums-db-users.sql' replace  into table forums-db.phpbb_users ;
Query OK, 1042 rows affected (0.06 sec)
Records: 1042  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data infile '/tmp/forums-db-themes.sql' replace into table forums-db.phpbb_themes ;
Query OK, 1038 rows affected (0.04 sec)
Records: 1038  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
```

As you can see, the syntax is fairly easy to understand. The REPLACE directive will overwrite any previously existing records with the same unique, or primary, key in the source and destination tables. The IGNORE directive will only insert records where the primary keys are different.

Very Basic MySQL Network Security

By default MySQL listens on all your interfaces for database queries from remote MySQL clients. You can see this using netstat -an. Your server will be seen to be listening on IP address 0.0.0.0 (all) on TCP port 3306.

```
[root@bigboy tmp]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
...
...
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
...
...
[root@bigboy tmp]#
```

The problem with this is that it exposes your database to MySQL queries from the Internet. If your SQL database is going to be accessed only by applications running on the server itself, then you can force it to listen only to the equivalent of its loopback interface. Here's how.

1) Edit the /etc/my.cnf file and use the bind-address directive in the [mysqld] section to define the specific IP address on which MySQL listens for connections.

```
[mysqld]
bind-address=127.0.0.1
```

2) Restart MySQL. The netstat -an command will show MySQL listening on only the loopback address on TCP port 3306, and your application should continue to work as expected.

Basic MyQL Troubleshooting

You can confirm whether your MySQL installation has succeeded by performing these few simple steps.

Connectivity Testing

In the example scenario, network connectivity between the database and the application will not be an issue because they are running on the same server.

In cases where they are not, you have to use the troubleshooting techniques in Chapter 4, "Simple Network Troubleshooting", to test both basic connectivity and access on the MySQL TCP port of 3306.

Test Database Access

The steps outlined earlier are a good test of database access. If the application fails, then retrace your steps to create the database and register the database information into the application. MySQL errors are logged automatically in the /var/log/mysqld.log file; investigate this file at the first sign of trouble.

Sometimes MySQL will fail to start because the host table in the mysql database wasn't created during the installation, this can be rectified by running the mysql_install_db command.

```
[root@bigboy tmp]# service mysqld start
Timeout error occurred trying to start MySQL Daemon.
Starting MySQL:  [FAILED]
[root@bigboy tmp]# tail /var/log/mysql.log
...
...
050215 19:00:33  mysqld started
050215 19:00:33  /usr/libexec/mysqld: Table 'mysql.host' doesn't exist
050215 19:00:33  mysqld ended
...
...
[root@bigboy tmp]# mysql_install_db
...
...
[root@bigboy tmp]# service mysqld start
Starting MySQL:  [  OK  ]
[root@bigboy tmp]#
```

A Common Fedora Core 1 MySQL Startup Error

You may notice that you can start MySQL correctly only once under Fedora Core 1. All subsequent attempts result in the message "Timeout error occurred trying to start MySQL Daemon.".

```
[root@bigboy tmp]# /etc/init.d/mysqld start
Timeout error occurred trying to start MySQL Daemon.
Starting MySQL:  [FAILED]
[root@bigboy tmp]#
```

This is caused by the MySQL startup script incorrectly attempting to do a TCP port ping to contact the server. The solution is:

1) Edit the script /etc/rc.d/init.d/mysqld.

2) Search for the two mysqladmin lines with the word ping in them and insert the string "-u $RANDOM" before the word "ping":

```
if [ -n "`/usr/bin/mysqladmin -u $RANDOM ping 2> /dev/null`" ]; then
if !([ -n "`/usr/bin/mysqladmin -u $RANDOM ping 2> /dev/null`" ]); then
```

3) Restart MySQL.

After doing this MySQL should function correctly even after a reboot.

Conclusion

MySQL has become one of the most popular Linux databases on the market and it continues to improve each day. If you have a large project that requires the installation of a database, then I suggest seeking the services of a database administrator (DBA) to help install and fine-tune the operation of MySQL. I also suggest, no matter the size of the project, that you practice an application installation on a test Linux system to be safe. It doesn't necessarily have to be the same application. You can find free MySQL-based applications using a Web search engine, and you can use these to become familiar with the steps outlined in this chapter before beginning your larger project.