



Linux Network Services: Databases

*This material is based on work supported by the
National Science Foundation under Grant No. 0802551*



*Any opinions, findings, and conclusions or recommendations expressed in this material are those of
the author (s) and do not necessarily reflect the views of the National Science Foundation*

Lesson Overview

In this lesson, you will explore the principles behind relational database servers, how they replaced flat files, and how they evolved. In this introduction, you will discuss the use of databases on the Internet.

Following this introduction, you will install two different database systems—MySQL database and a PostgreSQL database. The biggest difference between the two is that MySQL has a commercial version and PostgreSQL is free. However, MySQL will run on almost any operating system and is extremely well supported, while PostgreSQL is still primarily a Linux software package (although that is changing).

This lesson does not teach you SQL, the structured query language used to access data in relational databases such as MySQL and PostgreSQL. This is a subject found in many advanced database management courses. We will concentrate on the installation and setup of the packages.

Understanding dynamic databases is important because of their critical importance to business, commerce, storage, search, sorting, and Internet use.



ADD TABLE

Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

Lesson Objective:

Given the need to store and retrieve structured data, the student will install and configure a database server as per industry standards.



Lesson Outline

In this lesson, you will explore:

- ❖ History of Database Development
 - Relational Databases
 - Intro to SQL
 - Databases and Internet
 - PostgreSQL Vs MySQL
- ❖ Installation & Configuration
 - Installing MySQL
 - Installing PostgreSQL
 - Create Users
 - Create Backups



History of Database Development

In the 1960s, the computer industry began to recognize the need to efficiently manage larger quantities of data because their current indexing process was tedious and time consuming. Programmers had to “step-through” flat text files to find the data they needed. For example, if they had a list of people living in the United States, and they wanted to identify a specific person, programmers had to write a program that would step through every line of the database file to find the record with that person’s information.

The need to create a better database structure that allowed faster and better searches gave rise to new ideas about the organization of data. The 1970’s saw the first relational database. The idea behind the relational database was that every record was a fixed length and had a key. This key could be search on, or linked to other records associated with the primary record.

Required Reading

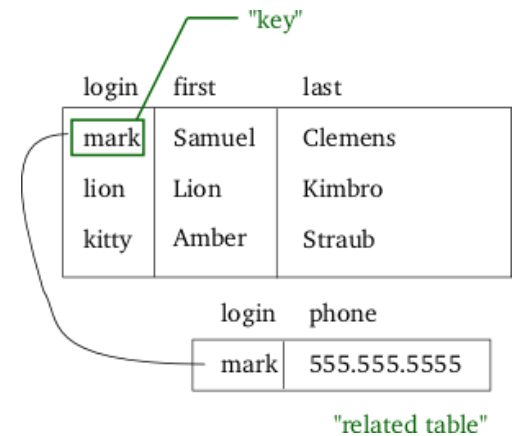
- [History of Database](#)
- [Database Management](#)

Structure of Relational Databases

For instance, a common use of a database system is to track information about users including their name, login information, various addresses, and phone numbers. In the navigational approach, all of these data fields would be placed in a single record, and unused items would simply not be placed in the database. In the relational approach, the data would be normalized into a user table, an address table, and a phone number table (for instance). Records would be created in these optional tables only if the address or phone numbers were actually provided.

Linking the information back together is the key to this system. In the relational model a person's information may be contained in multiple records spread across different tables. However, each related record is linked back to the person using a unique key, as shown in the diagram on the right.

When information is collected about a user, information stored in the optional (or related) tables would be found by searching for this key. For instance, if the login name of a user is unique, addresses and phone numbers for that user would be recorded with the login name as its key. This "re-linking" of related data back into a single collection is well optimized for databases but not as well for traditional computer languages.



Required Reading

- [Relational Database](#)

Introduction to SQL

The next step for database designers was the development, introduction, and standardization of a Structured Query Language (or SQL).

SQL was a universal set of instructions with the sole purpose of database operations. The query language was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. SQL statements contain the following elements:

1. **Clauses:** constituent components of statements and queries (sometimes optional)
2. **Expressions:** can produce scalar values or tables consisting of rows and columns
3. **Predicates:** These specify the conditions that should be evaluated (such as true / false statements)
4. **Queries:** These retrieve data based on specific values (the most important part of SQL)
5. **Statements:** These control program flow and transactions. May also affect connections to the database

Required Reading

- [Introducing SQL](#)

Introduction to SQL (Contd)

In the 1980's, the growth of the object-oriented database occurred. During this time, developers began to looking at data in databases as objects. For example, a person's data in a table may contain his or her name, address, and phone number. If you later added their height, weight, and blood type to the database, these would be added as data that belonged to the same person and would not be treated as extraneous data. This view of related data objects allowed developers to create relationships between objects and their attributes rather than to individual fields.

Another change that occurred in the late 1980's and still exists today is the need for increasing speed and reliability of databases. As the storage and usefulness of data grows, it is imperative that the data is accessible and correct. It is no longer acceptable for a record to vanish from a database.

Recommended Reading

- [What is SQL](#)

Databases And the Internet

As the speed and the reliability of the internet has grown many consumer and corporate functions are being done on large databases. Services offered by Amazon, Facebook, Google Mail, Google Applications, and Microsoft Office 2010 rely heavily on powerful databases.

Additionally many corporations are outsourcing functions that were previously managed in-house, to cloud services and databases where reliability is extremely important. Examples of these services include directory searches and medical records. Directory searches allow users to index and locate data on the Internet. The Google search engine is one example of directory search.

Medical records also use relational databases and require an extremely high level of security and reliability because doctors rely on the information contained in patient records to make life and death decisions. In today's Internet connected age, the use of fast, secure, and reliable databases is critical to the personalization and efficiency of the Internet.



MySQL Vs PostgreSQL

MySQL and PostgreSQL are widely used databases on the Internet. A comparison between the two systems must begin with how they were developed. MySQL's developers initially focused on speed while the developers of PostgreSQL focused on stability and standards. This difference meant that MySQL was most often the faster of the two. In more recent years, the differences between the two are not as significant as development has progressed. The choice to use either one is usually based on personal preferences or the popularity of the application with developers.

MySQL is widely popular among various open-source web development packages. The MyISAM engine (an older storage engine for MySQL) is often the only database engine offered by webhosting providers. Many web developers use MySQL for their development efforts. Thus, MySQL became widely popular in web development, and even labels itself as "The world's most popular open source database," a claim that may be spurious given the broad deployment of other open source database management systems (DBMS) such as SQLite, which is often used by small applications that need internal database functionality without requiring the feature set of a "full sized" DBMS.

Required Reading

- [MySQL & PostgreSQL](#)

MySQL Vs PostgreSQL (Contd)

Part of the reason MySQL is so popular is a common perception that MySQL is "easier" to use than other databases -- particularly PostgreSQL.

That perception arose years ago and has fed itself by word of mouth, to the point where accuracy has little or nothing to do with MySQL's current reputation for being comparatively easy to use. In fact, in recent years, PostgreSQL has made significant changes that have "closed the gap" or even improved its ease of use beyond that of MySQL. Of course, the validity of such claims is as open to question as those of MySQL.

In this lesson, we will install MySQL first, but only because it appears first in the alphabet!

Required Reading

- [Why PostgreSQL?](#)

Installation

MySQL & PostgreSQL

Step 1: System Prep & Update

To begin your database installation, regardless of the version you wish to install, you will need to update your Linux installation to the most recent version of software. For this lesson, we are using Fedora Linux since it is based on RedHat which is most often used in commercial server applications. Ubuntu is more often used as a desktop application.

Please follow these directions:

1. Open a terminal window (or console).
2. Type **su** and enter the root password when requested (if on an Ubuntu machine you may have to type **sudo su**)
3. Type **yum upgrade**
4. Accept any and all packages that may need to be installed along with any dependencies.
5. Do not log out or exit your console when complete.

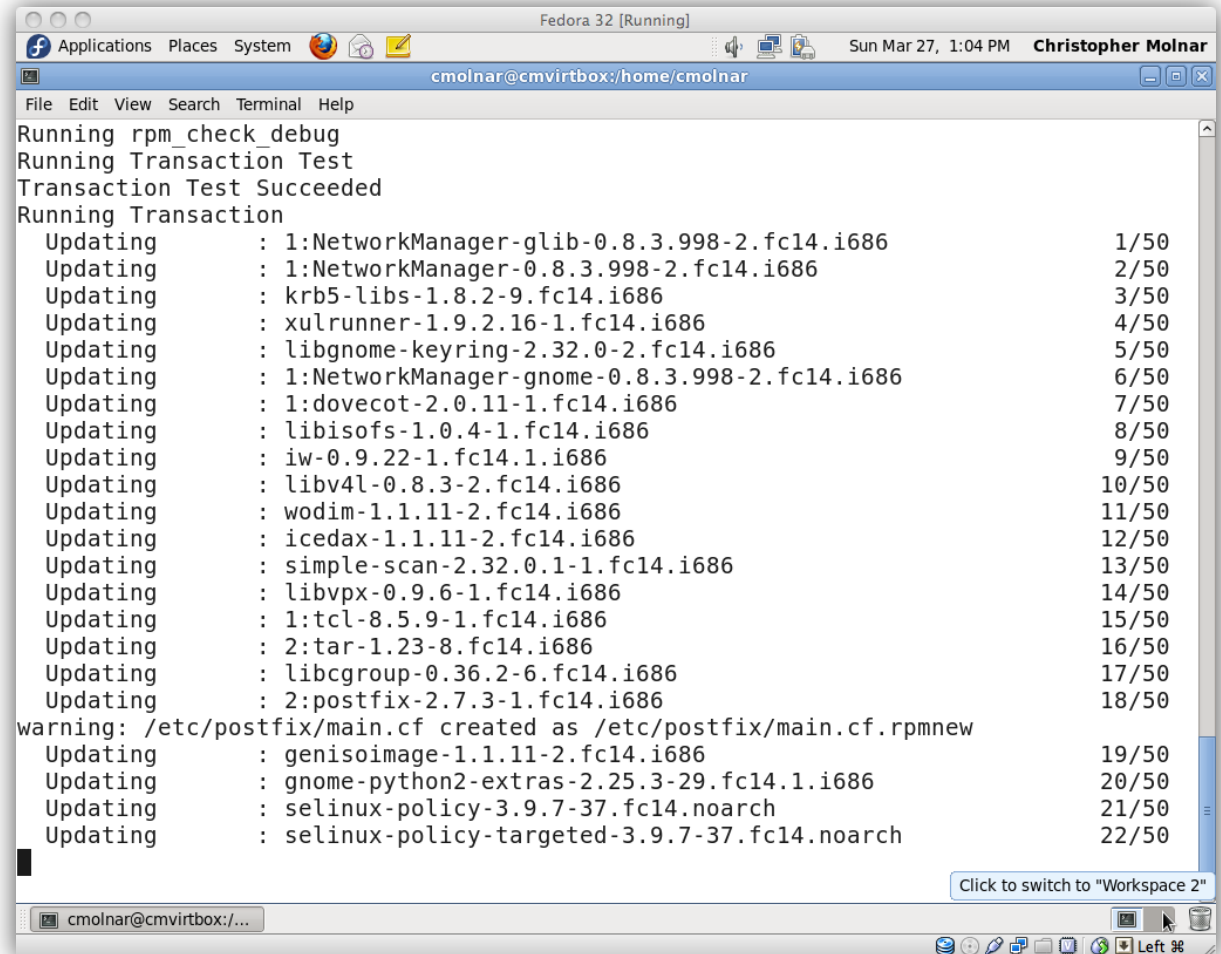
Your machine is now updated using the “yum” package manager to the most current version of all packages. Now, we can begin to install our software. As we move forward, you will notice that Step 2a instructions relate to MySQL and Step 2b relates to PostgreSQL (all the Step Xa’s are MySQL and all the Step Xb’s are PostgreSQL). This format will allow you to compare the steps required for each package.

See example of system prep and update on next screen.

Required Reading

- [MySQL on Fedora](#)
- [PostgreSQL on Fedora](#)

Example of System Updates



```
Fedora 32 [Running]
Applications Places System
cmolnar@cmvirtbox:/home/cmolnar
File Edit View Search Terminal Help
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Updating      : 1:NetworkManager-glib-0.8.3.998-2.fc14.i686           1/50
Updating      : 1:NetworkManager-0.8.3.998-2.fc14.i686             2/50
Updating      : krb5-libs-1.8.2-9.fc14.i686                       3/50
Updating      : xulrunner-1.9.2.16-1.fc14.i686                    4/50
Updating      : libgnome-keyring-2.32.0-2.fc14.i686                5/50
Updating      : 1:NetworkManager-gnome-0.8.3.998-2.fc14.i686      6/50
Updating      : 1:dovecot-2.0.11-1.fc14.i686                      7/50
Updating      : libisofs-1.0.4-1.fc14.i686                        8/50
Updating      : iw-0.9.22-1.fc14.1.i686                            9/50
Updating      : libv4l-0.8.3-2.fc14.i686                          10/50
Updating      : wodim-1.1.11-2.fc14.i686                           11/50
Updating      : icedax-1.1.11-2.fc14.i686                           12/50
Updating      : simple-scan-2.32.0.1-1.fc14.i686                   13/50
Updating      : libvpx-0.9.6-1.fc14.i686                           14/50
Updating      : 1:tcl-8.5.9-1.fc14.i686                            15/50
Updating      : 2:tar-1.23-8.fc14.i686                              16/50
Updating      : libcgroup-0.36.2-6.fc14.i686                       17/50
Updating      : 2:postfix-2.7.3-1.fc14.i686                        18/50
warning: /etc/postfix/main.cf created as /etc/postfix/main.cf.rpmnew
Updating      : genisoimage-1.1.11-2.fc14.i686                     19/50
Updating      : gnome-python2-extras-2.25.3-29.fc14.1.i686        20/50
Updating      : selinux-policy-3.9.7-37.fc14.noarch                 21/50
Updating      : selinux-policy-targeted-3.9.7-37.fc14.noarch        22/50
cmolnar@cmvirtbox:/home/cmolnar
```

System prep and update
prior to install

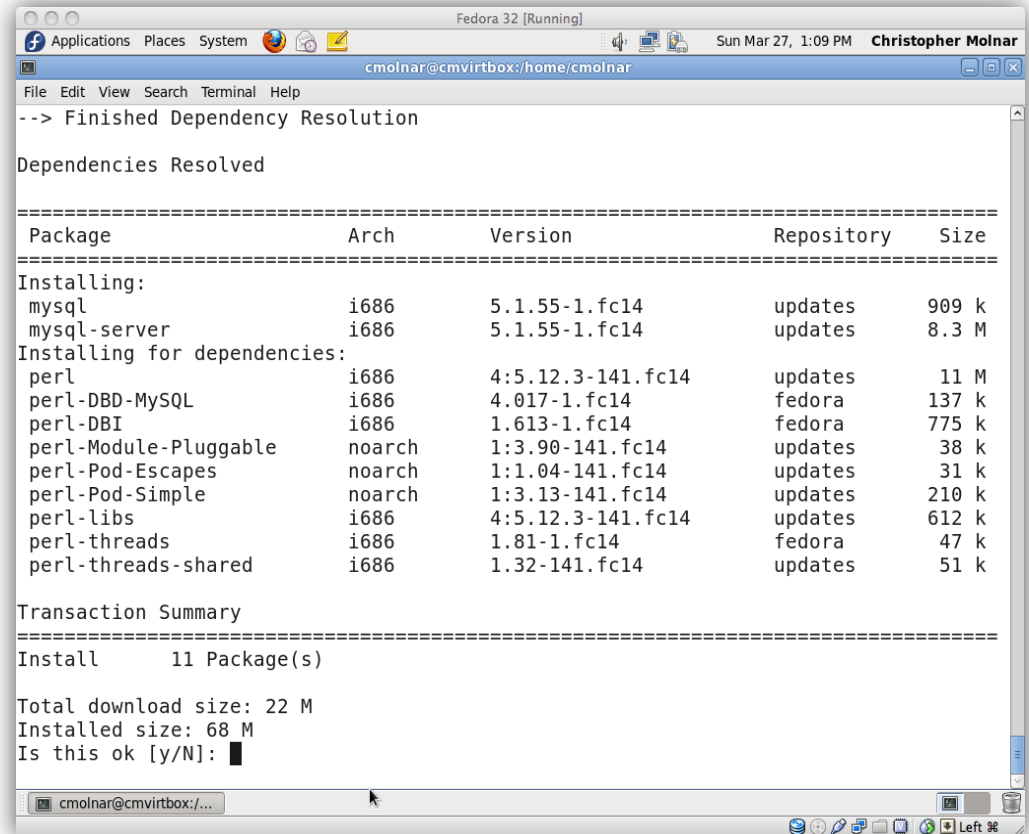
Step 2a: Installing MySQL Packages

Prior to starting and configuring MySQL, we need to install the latest packages. Follow these directions from your console while logged in as root:

1. Type: **yum install mysql mysql-server**
2. Accept any additional dependencies that need to be installed.

The MySQL package in step 1 contains all the client side (command line) utilities that are used with MySQL. The MySQL-server package contains all the server side utilities.

If you are installing a workstation that will be accessing another MySQL server, you need only install the MySQL package, not the MySQL-server package.



```
Fedora 32 [Running]
Applications Places System
cmolnar@cmvrtbox: /home/cmolnar
File Edit View Search Terminal Help
--> Finished Dependency Resolution
Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
mysql i686 5.1.55-1.fc14 updates 909 k
mysql-server i686 5.1.55-1.fc14 updates 8.3 M
Installing for dependencies:
perl i686 4:5.12.3-141.fc14 updates 11 M
perl-DBD-MySQL i686 4.017-1.fc14 fedora 137 k
perl-DBI i686 1.613-1.fc14 fedora 775 k
perl-Module-Pluggable noarch 1:3.90-141.fc14 updates 38 k
perl-Pod-Escapes noarch 1:1.04-141.fc14 updates 31 k
perl-Pod-Simple noarch 1:3.13-141.fc14 updates 210 k
perl-libs i686 4:5.12.3-141.fc14 updates 612 k
perl-threads i686 1.81-1.fc14 fedora 47 k
perl-threads-shared i686 1.32-141.fc14 updates 51 k

Transaction Summary
=====
Install 11 Package(s)

Total download size: 22 M
Installed size: 68 M
Is this ok [y/N]: █
```

Screenshot showing installation packages on Fedora

Step 2b: Installing PostgreSQL Packages

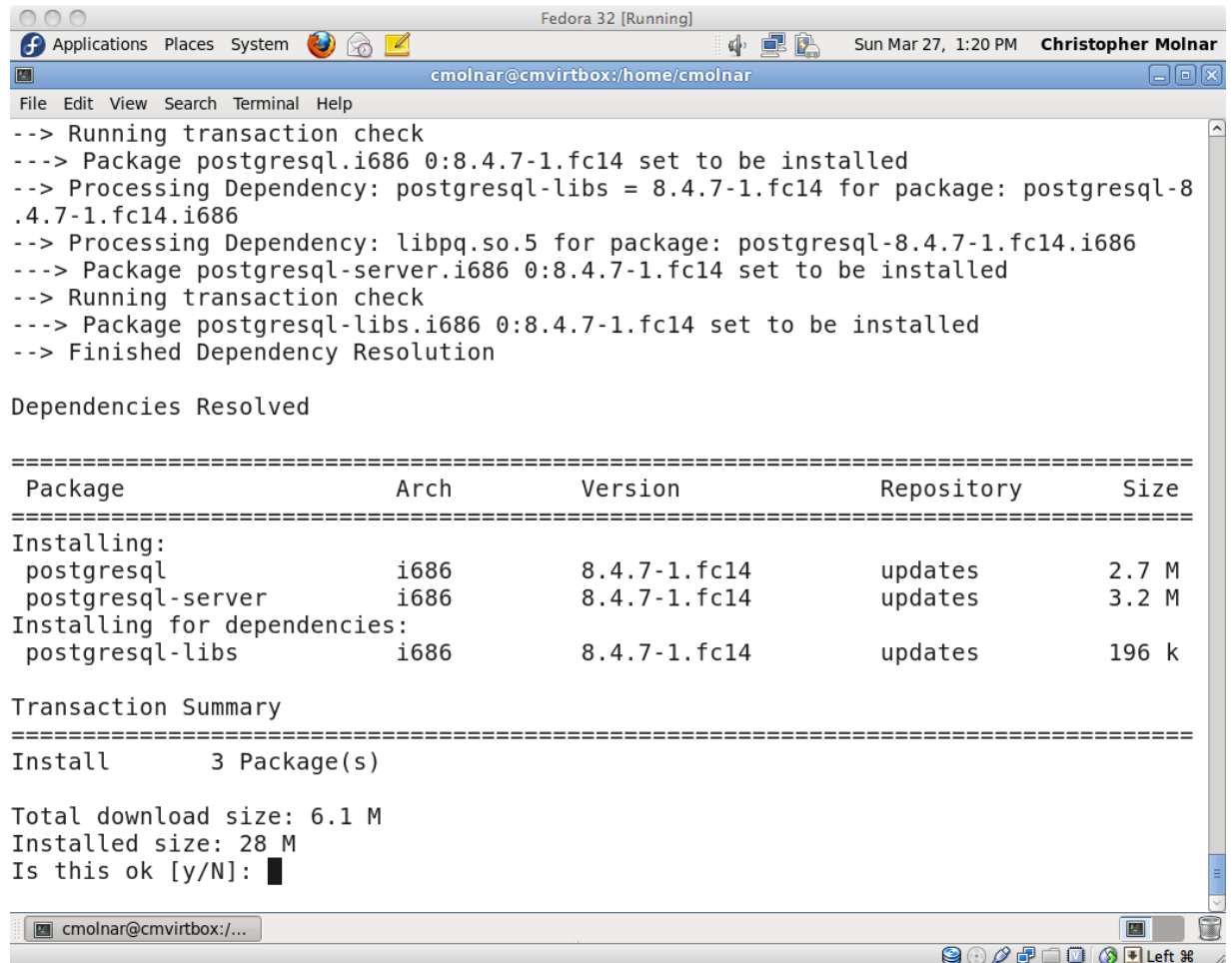
Prior to starting PostgreSQL you must install both the client and the server package. Please follow these directions from your console logged in as root. See the next screen for a screenshot of the installation process.

1. Type: **yum install postgresql postgresql-server**
2. Accept any dependencies or additional software that needs to be installed.

The PostgreSQL package in step 1 contains all the client side (command line) utilities used with PostgreSQL. The PostgreSQL-server package contains all of the server side utilities. If you are installing a workstation that will be accessing another PostgreSQL server you need only install the PostgreSQL package, not the PostgreSQL-server package.

Once the software packages are installed, we will now begin the configuration process.

Example of System Updates (2b)



```
Fedora 32 [Running]
Applications Places System
cmolnar@cmvrtbox:/home/cmolnar
File Edit View Search Terminal Help
--> Running transaction check
---> Package postgresql.i686 0:8.4.7-1.fc14 set to be installed
--> Processing Dependency: postgresql-libs = 8.4.7-1.fc14 for package: postgresql-8.4.7-1.fc14.i686
--> Processing Dependency: libpq.so.5 for package: postgresql-8.4.7-1.fc14.i686
---> Package postgresql-server.i686 0:8.4.7-1.fc14 set to be installed
--> Running transaction check
---> Package postgresql-libs.i686 0:8.4.7-1.fc14 set to be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version      Repository    Size
=====
Installing:
postgresql              i686      8.4.7-1.fc14 updates       2.7 M
postgresql-server      i686      8.4.7-1.fc14 updates       3.2 M
Installing for dependencies:
postgresql-libs        i686      8.4.7-1.fc14 updates       196 k
=====

Transaction Summary
-----
Install                3 Package(s)

Total download size: 6.1 M
Installed size: 28 M
Is this ok [y/N]: █

cmolnar@cmvrtbox:/...
```

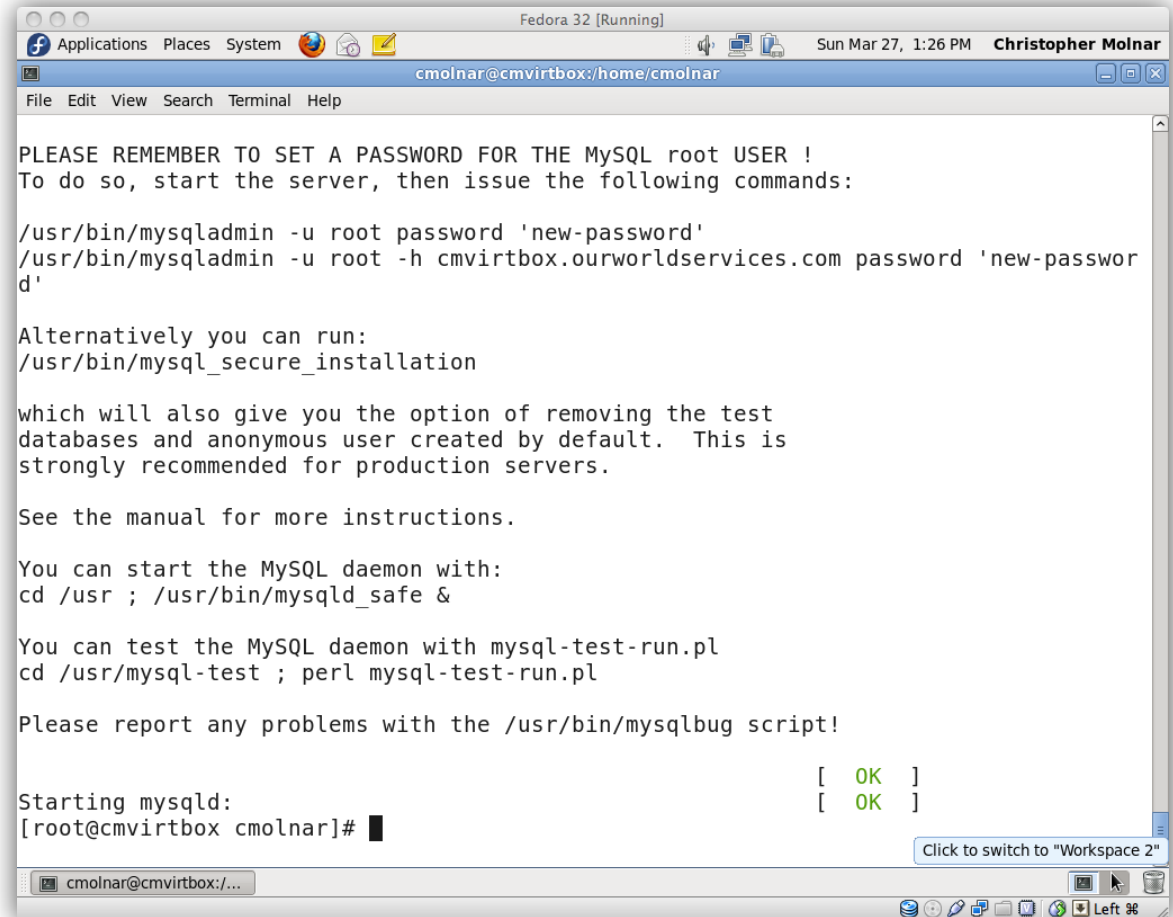
System prep and update

Step 3a: Starting MySQL Server

To start the MySQL server from your command line window, follow these instructions:

Type: **`/etc/init.d/mysqld start`**

The server will create the initial table structure and provide a reminder to set the root password for your installation.



The screenshot shows a terminal window titled 'Fedora 32 [Running]' with the user 'Christopher Molnar' on 'Sun Mar 27, 1:26 PM'. The terminal displays the following text:

```
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 cmvirtbox.ourworldservices.com password 'new-passwor
d'

Alternatively you can run:
/usr/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd /usr ; /usr/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd /usr/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /usr/bin/mysqlbug script!

Starting mysqld: [ OK ]
[root@cmvirtbox cmolnar]#
```

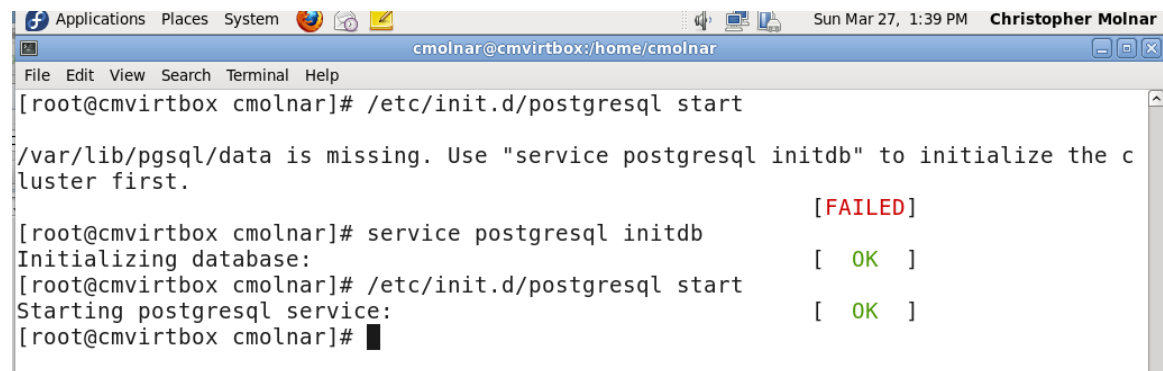
Step 3b: Starting PostgreSQL Server

To start the PostgreSQL database, use the **init.d** scripts but rather the service utility to initialize the database cluster first. From your command line terminal please follow these directions:

1. Type: **/etc/init.d/postgresql**
2. (You will get an error message stating PostgreSQL was never previously started. This is OK.)
3. Type: **service postgresql initdb**
4. Type: **/etc/init.d/postgresql start**

The server will create the initial table structure. Your next step will be to start PostgreSQL in step 4 above. At this point, your PostgreSQL database has started and running alongside your MySQL database. Next, we need to secure both structures by changing the root (administrative) passwords.

Screen shot of PostgreSQL server starting

A terminal window screenshot showing the process of starting PostgreSQL. The window title is 'cmolnar@cmvirtbox:/home/cmolnar'. The terminal output shows the following sequence of commands and responses:

```
[root@cmvirtbox cmolnar]# /etc/init.d/postgresql start
/var/lib/pgsql/data is missing. Use "service postgresql initdb" to initialize the cluster first.
[FAILED]
[root@cmvirtbox cmolnar]# service postgresql initdb
Initializing database: [ OK ]
[root@cmvirtbox cmolnar]# /etc/init.d/postgresql start
Starting postgresql service: [ OK ]
[root@cmvirtbox cmolnar]#
```

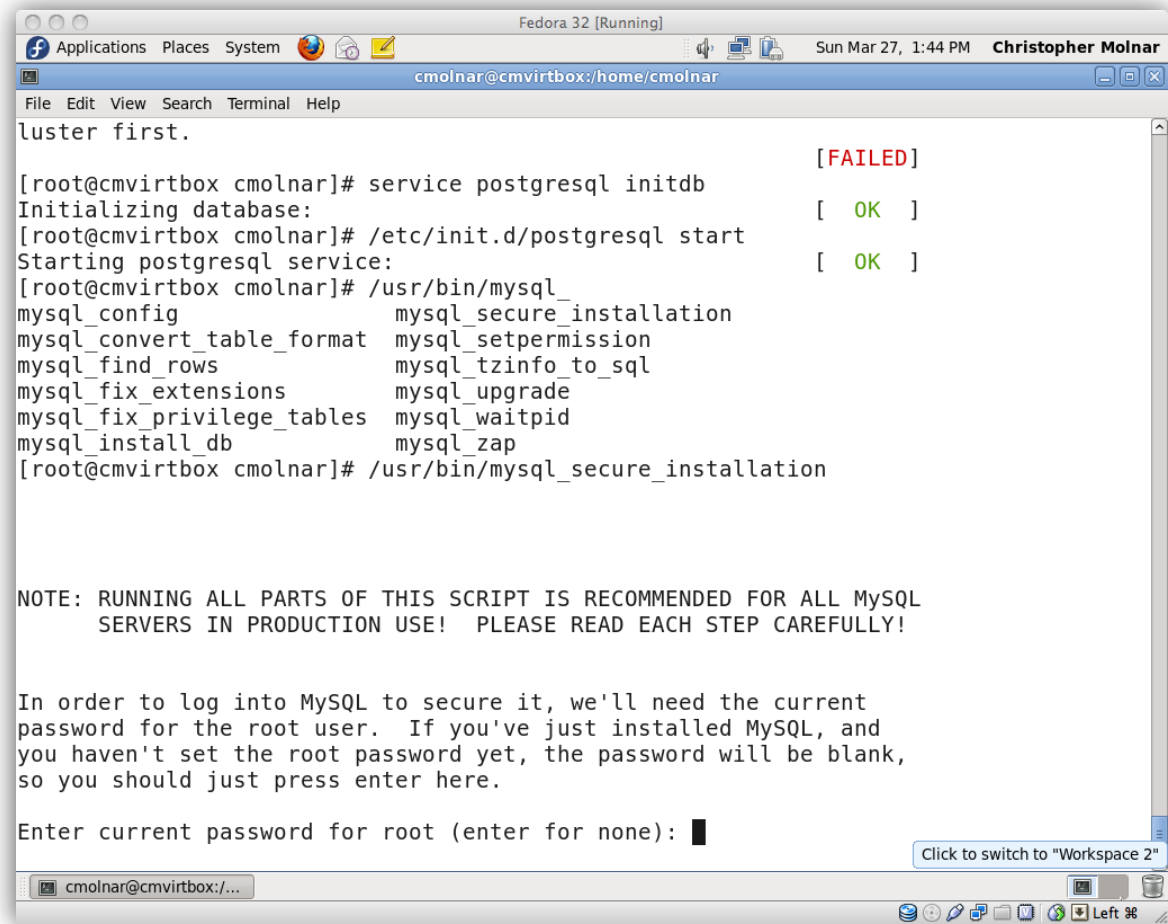
Step 4: Securing MySQL Admin Account

The first thing any Linux administrator needs to do with a MySQL database is to set the root user and administrative account passwords. The database server is installed without any of these passwords installed. Fortunately, there is a utility installed with MySQL that makes setting passwords an easy process. From your command line console please follow these instructions:

1. Type: **/usr/bin/mysql_secure_installation**
2. Press **Enter** when asked for the database root password. (Default setup does not have a password.)
3. Choose **Y** when asked if you want to set a new password.
4. Enter your new database administrator password the same way twice.
5. Choose to remove anonymous users when asked.
6. Choose to disallow root login from remote servers when prompted.
7. Choose to remove test database when prompted.
8. Choose to reload the privilege tables when prompted.
9. You should get a **complete** message when the script is done.

You just added a password for root access, and you did not allow remote root access to the database from a second machine. Later, you will set a user with admin rights. Your current configuration does not allow anonymous users (which are a database administrator's worst nightmare and provide a real security threat). One of the first rules of database security is refuse setups from others including developers. Consequently, you removed the test table. Finally, you reloaded the privileges table which includes all access-related information.

Securing MySQL Database



```
luster first.

[root@cmvirtbox cmolnar]# service postgresql initdb           [ FAILED ]
Initializing database:                                       [  OK  ]
[root@cmvirtbox cmolnar]# /etc/init.d/postgresql start
Starting postgresql service:                                 [  OK  ]
[root@cmvirtbox cmolnar]# /usr/bin/mysql_
mysql_config          mysql_secure_installation
mysql_convert_table_format  mysql_setpermission
mysql_find_rows        mysql_tzinfo_to_sql
mysql_fix_extensions   mysql_upgrade
mysql_fix_privilege_tables mysql_waitpid
mysql_install_db       mysql_zap
[root@cmvirtbox cmolnar]# /usr/bin/mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

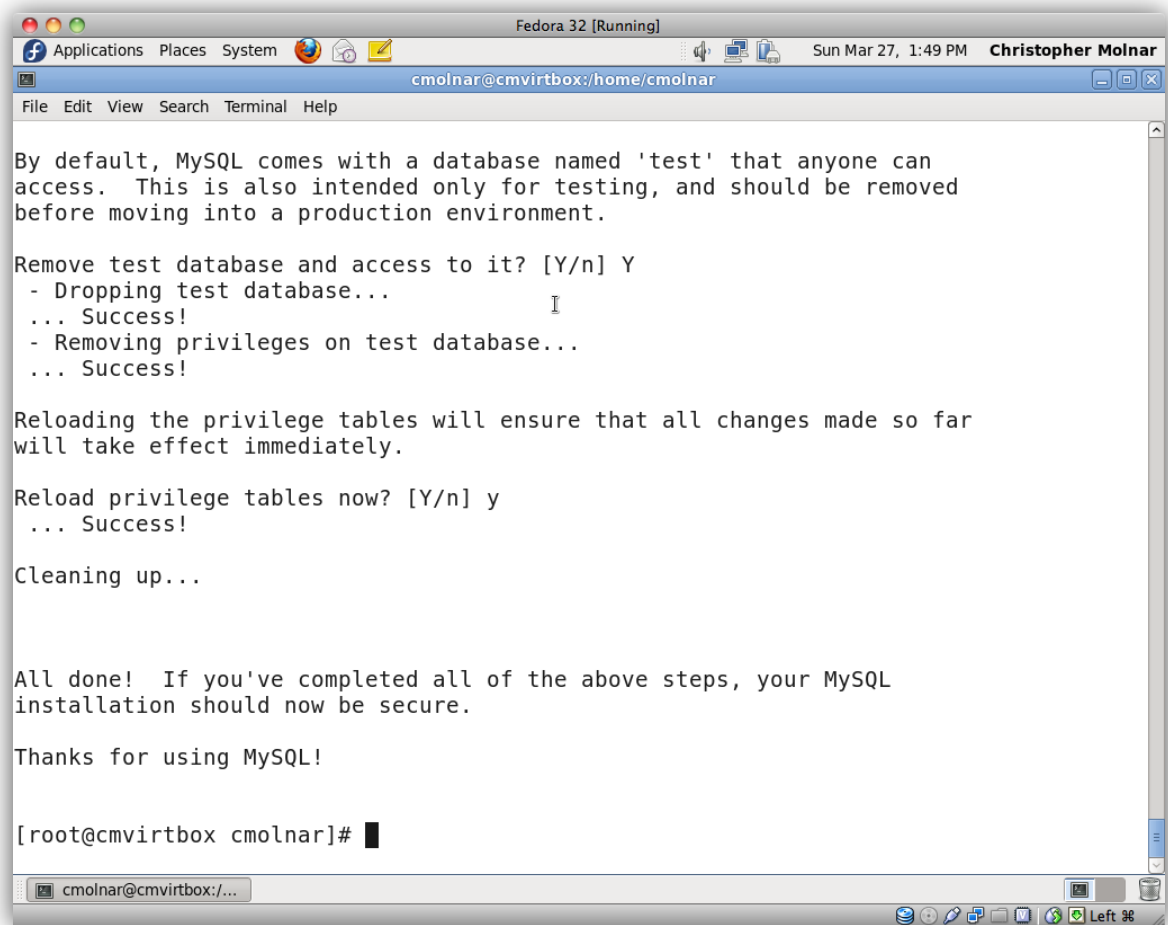
In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none): █
```

Screenshot showing the process of securing a MySQL database

Required Reading
[Secure MySQL from Attack](#)

Securing MySQL Database (Contd)



```
Fedora 32 [Running]
Applications Places System
cmolnar@cmvirtbox:/home/cmolnar
File Edit View Search Terminal Help

By default, MySQL comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MySQL
installation should now be secure.

Thanks for using MySQL!

[root@cmvirtbox cmolnar]#
```

Screenshot showing the process of securing a MySQL database

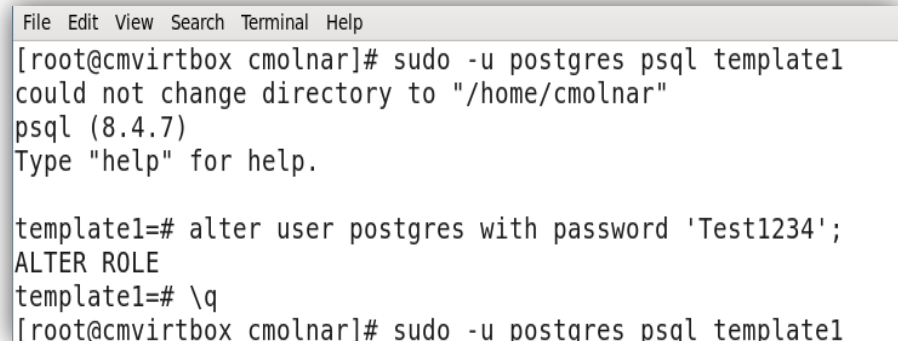
Step 4b: Securing PostgreSQL

As with MySQL, the first step is to secure PostgreSQL and disallow access to the administrator user. With PostgreSQL, the admin user is by default named “postgres.” Follow these directions from your console login. See the screen shot on the next slide for more information.

1. Type: **sudo -u postgres psql template1**
2. You will get an error message about a directory change.
3. At the template1=# prompt type:
ALTER USER postgres WITH PASSWORD 'NewAdminDatabasePassword';
4. In step 3, change the **NewAdminDatabasePassword** to a password you will remember.
5. Type: **vi /var/lib/pgsql/data/pg_hba.conf**
6. Scroll down to the bottom of the file and change the words **ident** to **md5**
7. Type: **:x** to save and exit vi.
8. Type: **/etc/init.d/postgresql restart**

Now, if you re-type the string in step 1 above, you should get a password prompt. Enter your new administrative password and you should be able to access the database.

Remember **\q** will exit. At this point, your PostgreSQL database is up and running. Next we need to setup access from a remote machine.



```
File Edit View Search Terminal Help
[root@cmvirtbox cmolnar]# sudo -u postgres psql template1
could not change directory to "/home/cmolnar"
psql (8.4.7)
Type "help" for help.

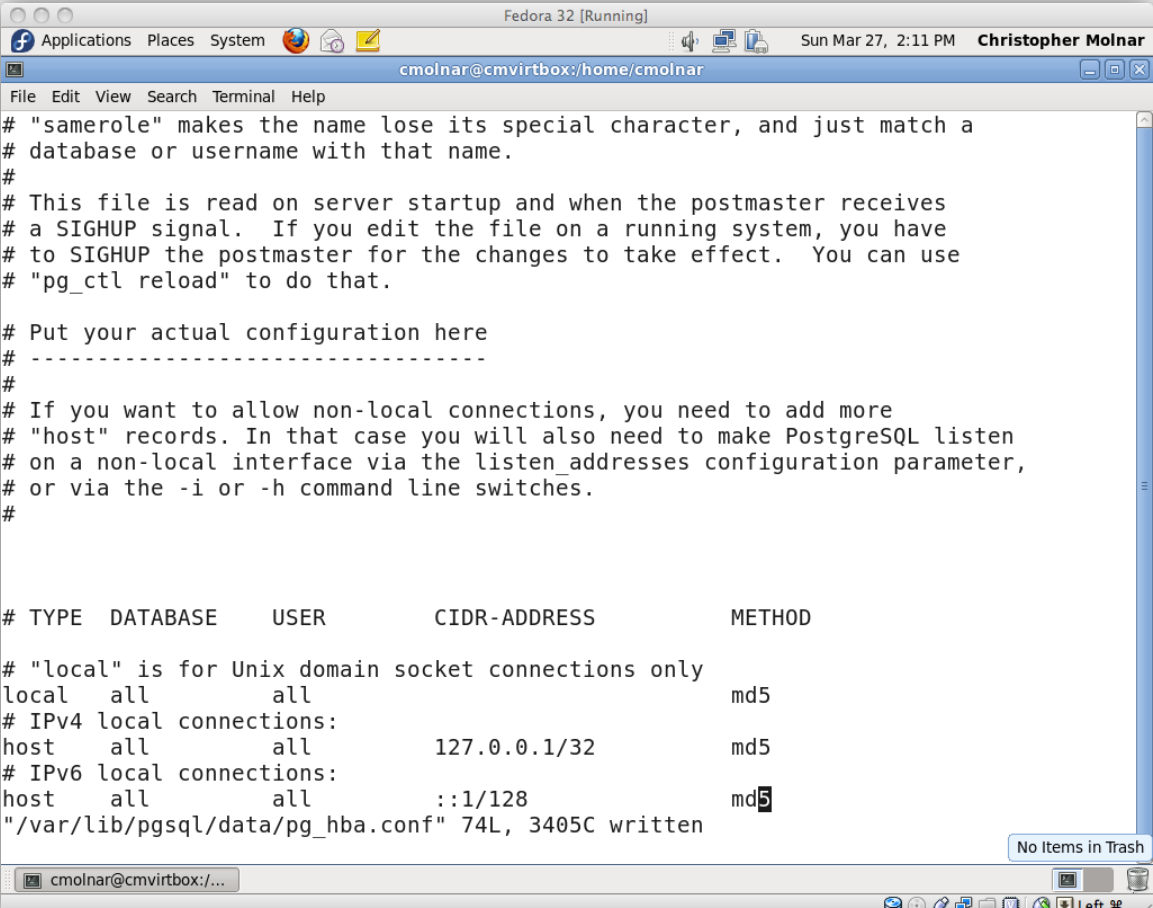
template1=# alter user postgres with password 'Test1234';
ALTER ROLE
template1=# \q
[root@cmvirtbox cmolnar]# sudo -u postgres psql template1
```

Screen shot of changes made to the PostgreSQL admin account

Securing PostgreSQL

Screenshot showing the process of securing a PostgreSQL database

Required Reading
[Secure PostgreSQL from Attack](#)



```
Fedora 32 [Running]
Applications Places System
cmolnar@cmvirtbox:/home/cmolnar
File Edit View Search Terminal Help
# "samerole" makes the name lose its special character, and just match a
# database or username with that name.
#
# This file is read on server startup and when the postmaster receives
# a SIGHUP signal.  If you edit the file on a running system, you have
# to SIGHUP the postmaster for the changes to take effect.  You can use
# "pg_ctl reload" to do that.
#
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records.  In that case you will also need to make PostgreSQL listen
# on a non-local interface via the listen_addresses configuration parameter,
# or via the -i or -h command line switches.
#
# TYPE  DATABASE  USER  CIDR-ADDRESS  METHOD
# "local" is for Unix domain socket connections only
local  all             all                                     md5
# IPv4 local connections:
host   all             all             127.0.0.1/32   md5
# IPv6 local connections:
host   all             all             ::1/128        md5
"/var/lib/pgsql/data/pg_hba.conf" 74L, 3405C written
No Items in Trash
cmolnar@cmvirtbox:/...
```


Step 5a: Creating a Working User on MySQL

Having a single administrative user and using that account for all work on a MySQL server is extremely insecure. Each database should have an admin user and each actual user should have his or her own user account. For this activity, we are going to create an additional user on your database to use for everyday activities. From your root console terminal follow these directions: (See screenshot on next slide.)

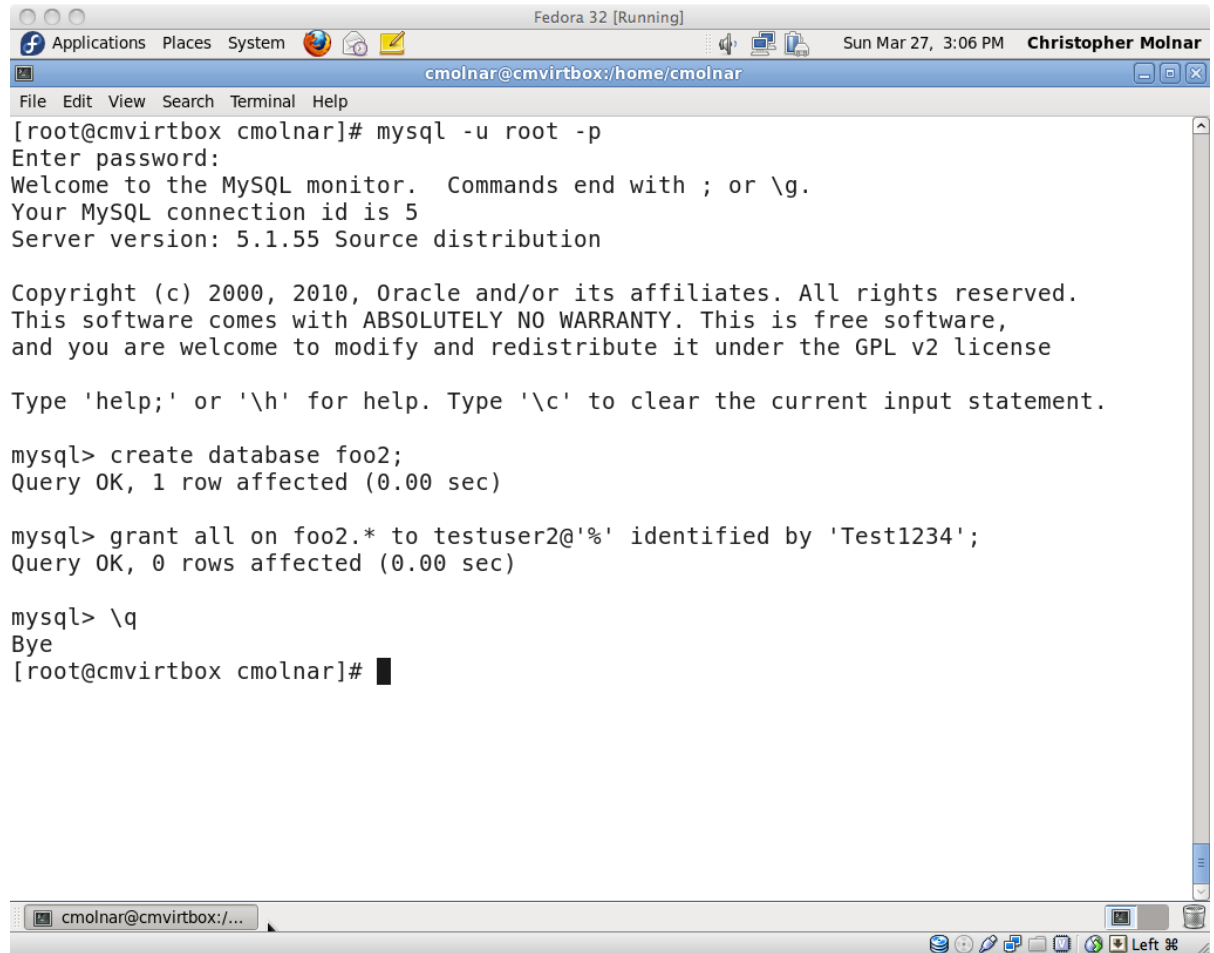
1. Return to your root console window and type: **mysql -u root -p**
2. Enter your admin password at the prompt
3. Type: **create database foo;**
4. Type: **grant all on foo.* to testuser@'%' identified by 'Test1234';**
5. Type: **\q**
6. From the shell prompt type: **mysql -u testuser -p foo**
7. Enter your new password
8. You should now be in the MySQL shell as your new user. This user now has access to control everything on the **foo** database that you created in step 3.
9. Type: **\q** to get back to the command line shell.

Now you have a configured MySQL database that is accessible under either *root* or *testuser* with passwords. There is also a single database created named “foo”. Next, we will do the same thing with your PostgreSQL installation.

Required Reading:

- ❖ [Create a user](#)
- ❖ [Create database](#)
- ❖ [Create table](#)

Securing PostgreSQL



The screenshot shows a terminal window titled "Fedora 32 [Running]" with the user "Christopher Molnar" on "Sun Mar 27, 3:06 PM". The terminal prompt is "[root@cmvirtbox cmolnar]#". The user runs "mysql -u root -p", enters a password, and is prompted to the MySQL monitor. The monitor displays the MySQL version (5.1.55) and copyright information. The user then runs "mysql> create database foo2;" and "mysql> grant all on foo2.* to testuser2@%' identified by 'Test1234';". Finally, the user runs "mysql> \q" to quit, and the terminal returns to the shell prompt "[root@cmvirtbox cmolnar]#".

```
[root@cmvirtbox cmolnar]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.1.55 Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

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

mysql> create database foo2;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on foo2.* to testuser2@%' identified by 'Test1234';
Query OK, 0 rows affected (0.00 sec)

mysql> \q
Bye
[root@cmvirtbox cmolnar]#
```

Screenshot showing the creation of a new user on a MySQL database

Step 5b: Create a Working User

Since we now have a single user “postgres” on our postgresql database system we need to create some additional users . From your root console terminal follow these directions:

1. Return to your root console window and type:
sudo -u postgres psql template1
2. Enter your admin password at the prompt.
3. Type: **create database foo;**
4. Type: **create user testuser with password 'Test1234';**
5. Type: **grant all privileges on database foo to testuser;**
6. Type: **\q**
7. From the shell prompt type: **mysql -u testuser -p foo**
8. Enter your new password
9. You should now be in the MySQL shell as your new user. This user now has access to everything on the foo database you created in step 3.
10. Type: **\q** to get back to the command line shell.

Now you have a configured PostgreSQL database that is accessible under either *root* or *testuser* with passwords. There is also a single database created named *foo*.

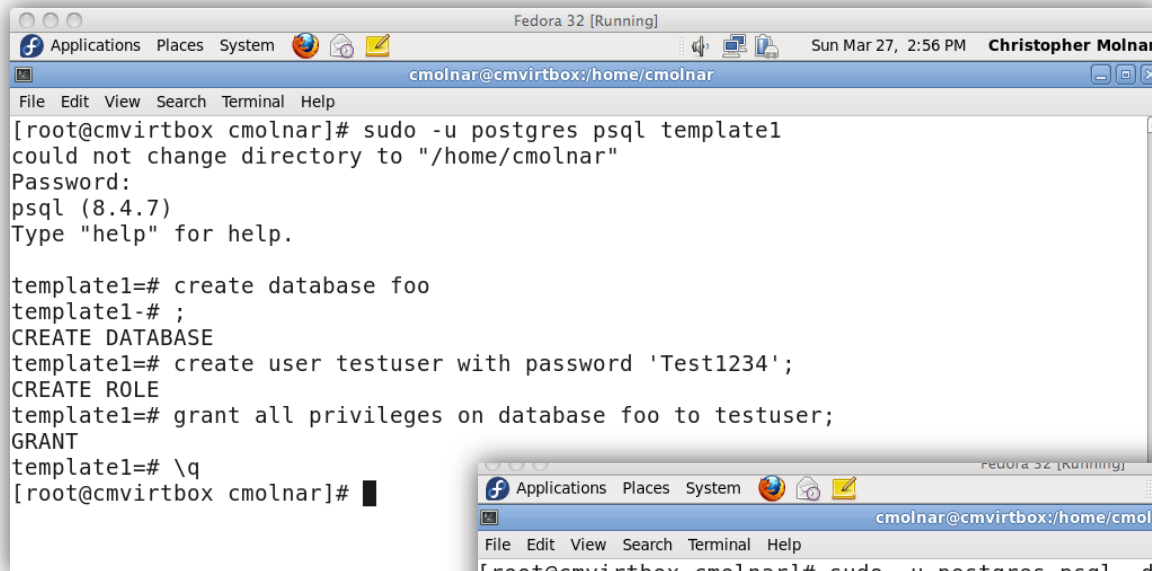
Required Reading:

- ❖ [PostgreSQL user](#)
- ❖ [PostgreSQL database](#)
- ❖ [PostgreSQL table](#)

For Review

- ❖ [Data types](#)

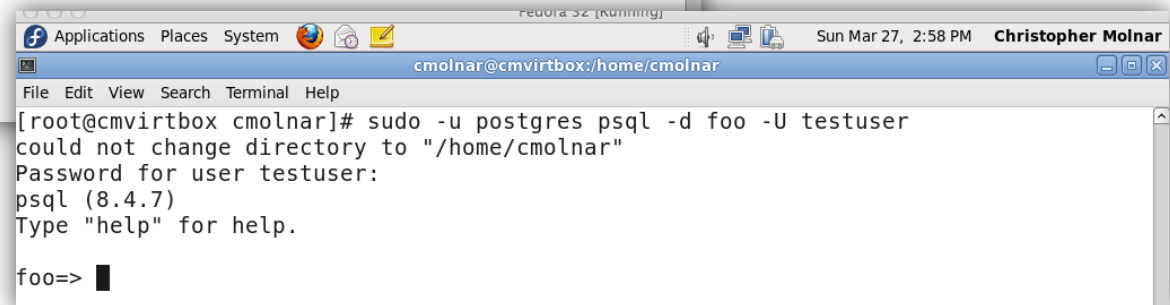
Screenshot: Adding User to PostgreSQL



```
Fedora 32 [Running]
Applications Places System Sun Mar 27, 2:56 PM Christopher Molnar
cmolnar@cmvirtbox:/home/cmolnar
File Edit View Search Terminal Help
[root@cmvirtbox cmolnar]# sudo -u postgres psql template1
could not change directory to "/home/cmolnar"
Password:
psql (8.4.7)
Type "help" for help.

template1=# create database foo
template1=# ;
CREATE DATABASE
template1=# create user testuser with password 'Test1234';
CREATE ROLE
template1=# grant all privileges on database foo to testuser;
GRANT
template1=# \q
[root@cmvirtbox cmolnar]#
```

Screenshot showing new user being added to PostgreSQL database



```
Fedora 32 [Running]
Applications Places System Sun Mar 27, 2:58 PM Christopher Molnar
cmolnar@cmvirtbox:/home/cmolnar
File Edit View Search Terminal Help
[root@cmvirtbox cmolnar]# sudo -u postgres psql -d foo -U testuser
could not change directory to "/home/cmolnar"
Password for user testuser:
psql (8.4.7)
Type "help" for help.

foo=>
```

After setting up the database and adding a user, we need to backup the database. The database structure as it exists on the filesystem does not lend itself to a proper backup/restore sequence. The best way to do this backup is to dump the data to a text file and store that text file on the backup media.

Step 6a: Create Backup on MySQL

The MySQL package is installed with a tool to backup the database. From a command line console, follow these directions:

1. Type: **cd**
2. Type: **mysqldump -a --all-databases -u root -p > backup-DATE.sql**
3. In line 2, replace **DATE** with today's date in YYYYMMDD format.
4. Enter your root (database administrators password when requested).
5. Type: **ls -la**
6. Verify that the **backup*.sql** file was created.

In step 1, we changed to your home directory using the *cd* command by itself. Then we used the *mysqldump* utility to backup all databases to a flat file. Then finally, we verified that the file was created.

This backup file would be copied to a CD/DVD/ or some other form of backup. It contains all the structures and data found in the database, so it's extremely important to keep this data in a safe, secure place.

Required Reading:

- ❖ [Database dumps](#)

Backup Process Screenshot

```
[cmolnar@cmvirtbox ~]$ cd
[cmolnar@cmvirtbox ~]$ mysqldump -a --all-databases --add-drop-table -u root -p > Backup-2011MAR27.sql
Enter password:
[cmolnar@cmvirtbox ~]$ ls -la
total 652
drwx----- . 24 cmolnar cmolnar 4096 Mar 27 15:30 .
drwxr-xr-x. 3 root root 4096 Mar 20 16:42 ..
-rw-rw-r--. 1 cmolnar cmolnar 494639 Mar 27 15:30 Backup-2011MAR27.sql
-rw-----. 1 cmolnar cmolnar 49 Mar 20 18:01 .bash_history
-rw-r--r--. 1 cmolnar cmolnar 18 Jun 22 2010 .bash_logout
```

Screenshot showing MySQL backup process

Step 6b: Create Backup on PostgreSQL

The PostgreSQL package is installed with a tool to backup the database. From a command line console, follow these directions:

1. Type: **cd**
2. Type: **sudo -u postgres pg_dumpall > backup-DATE.sql**
3. In line 2 replace **DATE** with today's date in YYYYMMDD format.
4. Enter your root (database administrators password when requested).
5. You will need to enter this password for each database that is dumped. Continue until the command line prompt returns.
6. Type: **ls -la**
7. Verify that the **backup*.sql** file was created.

In step 1, we changed to your home directory using the **cd** command by itself. Then we used the **pg_dumpall** utility to backup all databases to a flat file. Then finally, we verified that the backup file was created.

This backup file would be copied to a CD/DVD or some other form of backup. It contains all of the structures and data found in the database so it must be kept safe and secure.

Screenshot: PostgreSQL Backup

```
[root@cmvirtbox cmolnar]# sudo -u postgres pg_dumpall > testbackup.sql
could not change directory to "/home/cmolnar"
Password:
Password:
Password:
Password:
[root@cmvirtbox cmolnar]# vi testbackup.sql
[root@cmvirtbox cmolnar]# █
```

Screenshot showing PostgreSQL backup process

Required Reading:

❖ [PG_Dumpall](#)

Lesson Summary

Both MySQL and PostgreSQL are extremely fast and configurable database systems. They are used by companies such as Google, GoDaddy, Microsoft (Hotmail), and others.

In this lesson, you began by installing MySQL and PostgreSQL using Yum. Then we configured MySQL and PostgreSQL to enhance security. We removed the anonymous database and user from the MySQL installation and then created the database structure in PostgreSQL.

Then we used the **create** and **grant** commands to create a new database in each of our servers and allowed our testuser to access this new database. Finally, we used **pg_dumpall** and **mysqldump** to backup the databases to a flat file.

Further configuration of the database server would allow you to open it to network access, allow access from Java or C programs, and add more databases and users if required.

Required Reading:

- ❖ [Getting started with MySQL](#)