How to “Dropbox” Files Between Computers on Your Home Network

Barry Block
bwb@fireflysoftware.com
28 Sep, 2014
(rev A)

Note: This tutorial is hosted on GitHub.[1] Before working through it, be sure that you have the latest revision by downloading the .pdf file from there.

Introduction

A backup strategy is only as perfect as it is *convenient*. Have you ever wished that you, as a Linux user, had a backup solution available that was as simple to use as dragging a file or folder onto a panel icon but yet it didn't require an on-line cloud service like *Dropbox*? In this tutorial I'll walk you through the steps needed to *easily* keep the important files on your computer backed up to another computer on your home network.

If you happen to have a spare computer system just laying about collecting dust, you likely have everything you need to create your own “backup server” which you could then use to keep the files on your laptop backed up. In fact, this “spare” computer system need not be collecting any dust at all. As you really only need use of some of its hard drive space and nothing more, you could use your spouse's existing computer or even your HTPC computer to back up to.

What makes all of this workable is that you only need to commandeer enough space to store *one* copy of your files (and then not necessarily *all* of your files either[2]). Clearly, *multiple* members of your household could use this approach to back up the work they've done on their computers to the *same* backup server. Each member just needs a login on the backup server that matches the login on their laptop, etc.[3]

How it Works

The backup strategy employed here makes use of *rsync*—a program that is commonly used on Linux systems for keeping two computers' file systems in sync. The bash script that is used to run rsync is designed so that either a file or folder beneath one's home folder can be dragged onto it. This effectively provides the script with the file/folder's fully qualified path. From this path alone the script has everything it needs to perform the backup. In particular, the script

1 It can be downloaded here: https://github.com/bblock/PipeWrench/blob/master/Extras/How%20to%20Dropbox%20Files%20Between%20Computers%20on%20Your%20Home%20Network.pdf?raw=true.
2 I back up only the contents of a single folder beneath my home folder, (including all its subfolders).
3 Even a “mutual”, two-way backup between two computers is possible using this method.

knows *where* to copy the file or folder on the backup server because this location is exactly the same as that of the file/folder being dragged on the “client” computer.

Just Some Preliminaries

All of the steps of this tutorial were worked out with my laptop computer and a “server” both running on a Linux live CD (in particular, the Linux Mint 17 live CD, MATE (pronounced “*may-tay*”) edition).[4] I highly recommend that anyone working through this tutorial do so using a live CD/USB, virtual PC, a junk computer etc., if only for the peace of mind it may bring in knowing that, no matter what mistakes are made, your system is reasonably safe from harm. Granted, this might not be a precaution that everyone can conveniently take, so whatever works best for you. Just be aware that there could be some risk of damage as a result of mistakes made and that you alone assume all risk of any such damage that might occur to your system if you choose to take no precaution. I'm going to assume that you'll be working from a live CD/USB and if not, you may need to adjust the instructions a bit here and there. I'll also assume that you know how to connect to the internet once you have booted into the live CD.

Setting Up the Server

With all of that out of the way, let's get started configuring the backup server. Again, you should be running from a live CD, USB or whatever and you should also be connected to your modem/router and have access to your home network and the internet.

To start, we'll need to install some software. As with other Debian-based distros, in the MATE edition of Linux Mint, the GUI way to do this would be to open up the *Synaptic Package Manager*, or “Package Manager” as it's called in the Mint menu. However, for expediency, we'll use the tried and true method of... *the terminal*. Oh, stop yer shakin'—it's not that bad! In fact, the terminal is our friend here. Trust me. ;-)

So, go ahead and open a terminal[5] and run the following command:

```
sudo apt-get update
```

To make running the command a bit easier, you can copy it into the terminal and then press the *Enter* key. Better yet, you can simply *select the above text and drag it straight into the terminal* (you can even select and drag/drop multiple commands).[6] Note that depending on whether you've selected the trailing end-of-line, you may still need to press the *Enter* key afterwards so that the command gets executed.

4 You can download the latest Mint “live CD” images at http://www.linuxmint.com/download.php. I like working from a live CD when venturing into less familiar parts of the Linux landscape as it provides a convenient means of making mistakes without being brutally punished as a result. Of course, another advantage of exploiting live CDs in Linux-based tutorials is that anyone can follow along regardless of the O/S currently installed on their system.

5 To do this in Linux Mint/MATE you would press the “Windows” key, type “term”, down-arrow once to “Terminal” and press Enter.

6 If the terminal window isn't visible during the drag operation, then simply drag over its “button” in the *window list* on the *panel* to first make it visible and then reposition the mouse over the now visible terminal and drop the text.

So, what the above command does is update the package cache so that the system has the latest package information available when packages are installed. It should generate quite a bit of text output, possibly including an error message or two.[7]

Next, we need to install SSH so that the two systems will be able to communicate. Run the following command from the terminal:

```
sudo apt-get install ssh
```

You'll be asked "Do you want to continue?" Just type "y" and press the *Enter* key.

In order for the client computer to access this (server) computer via SSH, the user on the server must have a login password. When booted from the live CD, the password for the default user ("mint") isn't set. To set it, simply run the following command:

```
passwd mint
```

You will be prompted to enter the existing password and since it isn't currently set, you just need to press the *Enter* key. Then, you'll be prompted to enter a new password (twice). Make it easy on yourself and just enter "password". This is the password that you'll later need to provide when running the two scripts.

Note: Normally, when setting up a backup server, you'd *create* a new user on that system having the same user name as the one on the client computer. In our case here, this isn't necessary because both systems are running from a Linux Mint live CD and therefore, both user names are already the same: "mint".

Before we move on to configuring the client, there's one last thing we need to do and that is to record the IP of the server so that it can be referenced by the client. From the terminal, run the following:

```
ifconfig
```

You'll find the computer's IP address listed as "inet addr" most likely in connection with either "eth0" or "wlan0", depending on how the system is connected to your modem/router.

Setting Up the Client

Next, we need to configure the computer whose files you want to back up to the backup server. As with the server, you'll need to boot this computer from a live CD and afterwards, you'll need to connect the system to your modem/router so that it has access to your home network and to the internet.

As was done previously on the server, we need to update the package cache before we can install software; but first, a new software repository (known as a Personal Package Archive or "PPA") needs to be added to the system. Open a terminal and run the following command:

```
sudo add-apt-repository ppa:bwb-s/pipewrench
```

7 If you see error messages pertaining to the cdrom, just ignore them—they shouldn't cause any problems.

Now we can update the package cache. Run this command:

```
sudo apt-get update
```

Again, you may encounter some errors here involving the cdrom. If so, just ignore them.

With the cache updated, *PipeWrench* can now be installed from its PPA. Run the following command:

```
sudo apt-get install pipewrench
```

As the name implies, *PipeWrench* is a tool that allows you to create and manage *pipes*. It will be used by the two bash scripts that will later be added to this, the client computer.

Next up, SSH needs to be installed in order to allow communication with the server. Run the following command from the terminal:

```
sudo apt-get install ssh
```

You'll be asked "Do you want to continue?" Just type "y" and press the *Enter* key.

Now that SSH is installed, we should test the connection to the server. This will provide the added benefit of finalizing the SSH configuration as you'll see. In the terminal, run the following command (replacing "<IP>" with the IP address of the server):

```
ssh mint@<IP>
```

You'll likely be presented with the message, "The authenticity of host <IP> can't be established... Are you sure you want to continue connecting (yes/no)?" Respond by typing "yes" and pressing *Enter*. Afterwards, you'll be prompted to enter the *mint* user's password which you had previously set on the server.

Having successfully entered your password, you should now be logged on to the server computer as user *mint*. This not only establishes that the SSH connectivity to the server is working but—as the above message only displays the *first* time you connect—it also keeps us from encountering this same disruptive message later when we test our two scripts.

Before we can continue on with the rest of the tutorial, we need to close this SSH connection that we just established with the server. If we didn't do this, then all of our subsequent commands would be executed on the *server* rather than on *this* computer—the *client!* So, to close the connection, run the following command:

```
exit
```

You should be presented with the following:

```
logout
Connection to <IP> closed.
```

Now, on to setting up the two scripts. The intent is to have each of these scripts run in a terminal when a file and/or folder is "dropped" on its respective .desktop file or "launcher".

Unfortunately, in the MATE edition of Linux Mint 17, the terminal application we've used to this point isn't correctly configured to run from such a launcher. Rather, the terminal emulator, *xterm* is configured to do so and, by default, it isn't installed. Though it would be preferable for the already installed MATE terminal to work as it should, a much easier fix is for us to just install *xterm*. So, again from the terminal, run the following command:

```
sudo apt-get install xterm
```

Next, we need to obtain a copy of the two scripts themselves, *files2sync* and *syncfiles*. The first script will be used to determine which files in the dragged folder are not in sync with the corresponding files on the backup server and the second script will be used to effectively synchronize them. Both of these scripts are included as part of a demo that is installed with *PipeWrench*. As PipeWrench is installed already, the simplest means of accessing the scripts is to run *PipeWrench* from MATE's main menu. Simply press the “Windows” key, type “pipe”, arrow down to *PipeWrench* and press the *Enter* key to run it.

When the *PipeWrench* GUI opens, click the *Help* menu/*Demos*. The demos folder should open in MATE's file manager, *Caja*. Navigate into the folder, *Dropbox Files Between Computers* and copy the two scripts *files2sync* and *syncfiles* onto the desktop.

In order for the scripts to run, they first need to be made *executable*. This is easily accomplished by running the following two commands from the terminal:

```
cd /home/mint/Desktop
chmod +x files2sync syncfiles
```

Next, we have to configure the two scripts residing on the desktop. Find the text editor *Pluma* in MATE's main menu and run it (if you're using another linux distro, use whatever text editor is appropriate). Now select and drag both scripts onto the editor to open them simultaneously.

Following the comment block at the top of each script you'll see some variables defined. Edit the variable, USERNAME so that it is “mint”. Likewise, edit the variable BUSIP to the IP that you recorded earlier for the backup server.

Note: In a real-world situation, you'd want the server's IP to be *static* so that it never changes (and thus you never have to edit the scripts); however, for the purposes of this tutorial, that won't be necessary.

When the *files2sync* script runs, it uses whatever text editor is defined by the TEXTEDITOR variable to output its results. Therefore, you'll also need to set the TEXTEDITOR variable to “pluma” (or whatever is appropriate if you're not running from the Linux Mint/MATE live CD).

Be sure to save the changes you've made to both scripts.

Next, you need to create the two launchers on the desktop that will run the scripts. For each script, you'll need to individually...

1. Right-click *on the desktop* and choose the option “Create Launcher...”.
2. For *Type*, choose “Application in Terminal”.

3. For *Name*, enter the script's name, either “files2sync” or “syncfiles”.
4. In the *Command* field, browse to the script on the desktop.
5. Optionally, you can enter a comment that will be displayed whenever you mouse over the launcher on the desktop.
6. Optionally, you can set the launcher's icon to something more meaningful than the default.

Note: Normally, you'd create both of these launchers on your desktop environment's *panel* (by right-clicking an open area of the panel and choosing “Add to Panel...”) rather than on the desktop itself so that they'd always be visible when it came time to use them.

A Test Run

At this point, both the client and server systems are configured. The final step is for us to test that everything works as expected. As part of this tutorial, we've created a number of files on the client computer's desktop. Of course, those same files don't exist on the server's desktop, so the first thing we can test is that the script, *files2sync* detects these differences.

Checking for Mismatches

From the client computer open the file manager from MATE's main menu by typing “caja”, down-arrowing to *Caja* and then pressing the *Enter* key to run the application.

When *Caja* opens, navigate to your home folder (recall, you're the user, *mint* and so your home folder is “/home/mint” which should be open by default).[8] In this folder you should see a subfolder named *Desktop*. Of course, this is the actual folder where files on your “visual” desktop are stored. Now drag the *Desktop* subfolder onto the *launcher* (not the script itself) that you created earlier on the (visual) desktop for the script *files2sync*. A terminal window (*xterm*) should open with a prompt asking you to enter your password. Enter the password that you had previously set for the server.

If all goes well, the text editor *Pluma* should open with the output from the script. Following is the output that I obtained:

```
files2sync rel 1.0

Source: "/home/mint/Desktop"

Mismatches:

C /home/mint/Desktop/files2sync | 7265 2014-09-23 18:06:35
C /home/mint/Desktop/files2sync.desktop | 259 2014-09-23 18:10:07
C /home/mint/Desktop/syncfiles | 5611 2014-09-23 18:06:07
C /home/mint/Desktop/syncfiles.desktop | 254 2014-09-23 18:10:36
C /home/mint/Desktop/ubiquity.desktop | 7795 2014-09-23 17:02:28
S /home/mint/Desktop/ubiquity.desktop | 7795 2014-09-23 17:14:07
```

8 In *Caja*, you can press Ctrl-L to see the current folder location displayed in text form.

Interpreting the Output

You'll notice that each line displayed beneath “Mismatches:” is preceded either by a “C” or an “S”. Each line thus represents a file either on the *client* or the *server*. Three types of mismatches can be output by *files2sync*:

1. A file exists on the client computer but not on the server. Each of these mismatches is represented in the output as a single, *unpaired* line beginning with “C”.

2. A file exists on the server but not on the client computer. Each of these mismatches is represented in the output as a single, *unpaired* line beginning with “S”.

3. A file exists on both computers but either their sizes or their modification times don't match. These mismatches are represented in the output as *two* consecutive (*paired*) lines whose file paths are identical but one line begins with an “C” and the other begins with an “S”.

As regards the output that I received from the *files2sync* script, the first four lines represent files that exist on the client but not on the server and the last two lines represent a single file, (ubuquity.desktop) who's modification time differs on the two systems.[9]

Resolving Mismatches

Once you've identified mismatches using the *files2sync* script, you'll usually want to resolve them right away. This is done using the *syncfiles* script. In our case, we've already identified file mismatches between the two computers insofar as the *Desktop* folder is concerned. We now have two options for resolving them—we can either drag each mismatched file individually onto the *syncfiles* launcher or we can more easily just drag the *Desktop* folder onto the launcher (much like we did earlier when running the *files2sync* script).

So, to resolve the mismatches, simply drag the *Desktop* folder onto the *syncfiles* launcher that you created earlier on the desktop. Again, you'll be prompted for your password. Afterwards, if all goes well, the terminal will just close.

Having synced the *Desktop* folder, check once more to see if any files on the desktop are mismatched by dragging the *Desktop* folder back onto the *files2sync* launcher. This time, *files2sync* should report that there are no mismatched files on the desktop:

```
files2sync rel 1.0

Source: "/home/mint/Desktop"

No mismatches
```

To make sure that the script isn't somehow lying to you, you can log back into the server and confirm that the files you created on the client's desktop actually exist on the server's desktop.

9 This difference was caused by my booting the two computers using two different Linux Mint live CD releases. My laptop (client) was booted using a 64-bit live CD and my “server” was booted from a 32-bit live CD.

From the terminal, run the following command (replacing “<IP>” with the IP address of the server):

```
ssh mint@<IP>
```

Again, you'll be prompted to enter the *mint* user's password.

Now that you're logged onto the server, you can list the contents of the server's desktop by running these two commands:

```
cd /home/mint/Desktop
ls -l
```

You ought to see listed the four files that were originally added to the client's desktop. :-)

Well, that about wraps it up for this tutorial. If you found it to be helpful, please leave a comment at Linux Mint's community portal (http://community.linuxmint.com/tutorial). Just do a search by the title of this tutorial. Thanks!