Build an Open Source Dropbox Clone

By PHIL CRYER

First off, if you haven't tried Dropbox, you should check it out; sync all of your computers via the Dropbox servers, their basic free service gives you 2Gigs of space and works cross-platform (Windows, Mac and Linux). I use it daily at home and work, just having a live backup of my main data for my work system, my home netbook, and any other computer I need to login to is a huge win. Plus, I have various 'shared' folders that distribute certain data to specific users and co-workers to whom I've granted access. This means work details can be updated and automatically distributed to the folks I want to review or use the data immediately. I recommend everyone try it out to see how useful it is, as it's turned into a game changer for me. So when Dropbox made headlines that they were supporting Linux, and releasing the client as open source, it got hopes up that users would be able to run their own, private Dropbox systems. In the end, it was only the client that was open source; the server would remain proprietary. While slightly disappointing, this is fine because it's a company trying to make money. I don't fault Dropbox for this, it's just that a free, portable service like that would be a killer app.

Meanwhile at work I'm working on a solution to sync large data clusters online and the project manager described it as the need for 'Dropbox on steroids'. Before I had thought it was more complicated, but after thinking about it, I realized he was right. Look, Dropbox is a great idea, but it obviously is just a melding of something similar to rsync, with something watching for file changes to initiate the sync, along with an easy- to-use front end. From there I just started looking at ways this could work, and there are more than a few; here's how I made it work.

Linux now includes inotify, which is a kernel subsystem that provides file system event notification. From there all it took was to find an application that listens to inotify and then kicks off a command when it hears of a change. I tried a few different applications like inocron, inosync and iwatch, before going with lsyncd. While all of them could work, lsyncd seemed to be the most mature, simple to configure and fast. Lsyncd uses inotify to watch a specified directory for any new, edited or removed files or directories, and then calls rsync to take care of business. So let's get started in making our own open source Dropbox clone with Debian GNU/Linux (Squeeze)

Ladies and gentlemen, start your ~~engines~~ servers!

First, you need two servers: one being the server and the other the client. (You could do this on one host if you wanted to see how it works for a proof of concept).

Install OpenSSH client and server

First you'll need to install OpenSSH on both the Client and Server. On the remote system:

```
apt-get install openssh-server
```

On the local box it's more than likely that the client is installed, but just in case:

```
apt-get install openssh-client
```

Configure SSH for Password-less Logins

You'll need to configure SSH to use password-less logins between the two hosts you want to use, as this is how rsync will pass the files back and forth. I've previously written a HOWTO on this topic, so we'll crib from there.

First, generate an SSH public key:

```
ssh-keygen -N '' -f ~/.ssh/id_dsa
```

You shouldn't have a key stored there yet, but if you do it will prompt you and ask if you want to overwrite it; make sure you overwrite it.

Enter passphrase (empty for no passphrase):

```
<Enter>
```

Enter same passphrase again:

```
<Enter>
```

We're not using pass phrases so the logins between the systems can be automated. This should only be done for scripts or applications that need this functionality, it is not for logging into servers lazily, and **it should never be done as root!**

Now, replace REMOTE_SERVER with the hostname or IP that you're going to call when you SSH to it, and copy the key over to the server:

```
ssh-copy-id REMOTE_SERVER
```

Note that if you have an older system you may not have ssh-copy-id installed, so you can do it the old way by piping the output of your key over SSH (which is good to know how to do anyway):

```
cat ~/.ssh/id_rsa.pub | ssh REMOTE_SERVER 'cat - >> ~/.ssh/
authorized_keys2'
```

Lastly, we need to set the permissions on the key file to a sane level:

```
ssh REMOTE_SERVER 'chmod 700 .ssh'
```

Now, give it a go to see if it worked:

```
ssh REMOTE_SERVER
```

You should be dropped to a prompt on the remote server without being prompted for a password. If not you may need to redo your .ssh directory, so on both servers:

```
mv ~/.ssh ~/.ssh-old
```

and goto 10

Install rsync and lsyncd

Next up is to install rsync and lsyncd. rsync is a basic command and should already be installed (you don't need to run it on the server, just the client on both systems), but to make sure you have it, and install lsyncd at the same time:

```
apt-get install rsync lsyncd
```

Note that before Squeeze there was no official Debian package, but it's simple to build from source and install if you need to. First off, if you don't have build essentials you'll need them, as well as libxml2-dev to build the lsyncd source. Installing those is as simple as:

```
apt-get install libxml2-dev build-essential
```

Now we'll download the lsyncd code, uncompress it and build it:

```
wget http://lsyncd.googlecode.com/files/lsyncd-1.39.tar.gz
tar -zxf lsyncd-1.39.tar.gz
cd lsyncd-1.39
./configure
make; make install
```

This install does not install the configuration file, so we'll do that manually now:

```
cp lsyncd.conf.xml /etc/
```

Configure lsyncd

Next we need to edit the configuration file now located in /etc The file is a simple, well- documented XML file, and mine ended up like so – just be sure to change the source and target hosts and paths to work with your systems:

```
<lsyncd version="1.39">
  <settings>
    <logfile filename="/var/log/lsyncd"/>
    <!--Specify the rsync (or other) binary to call-->
    <binary filename="/usr/bin/rsync"/>
    <pidfile filename="/var/run/lsyncd.pid"/>
    <callopts>
      <option text="-lt%r"/>
      <option text="--delete"/>
      <exclude -file/>
    <source />
    <destination />
    </callopts>
  </settings>
  <directory>
    <source path="/var/www/sync_test"/>
    <target path="desthost::module/"/> </directory>
</lsyncd>
```

Launch lsyncd in debug for testing

We're ready to give it a go, may as well run it in debug for fun and to learn how lsyncd does what it does:

```
lsyncd --conf /etc/lsyncd.conf.xml --debug
```

Watch for errors, if none are found, continue.

Add files and watch them sync

Now we just need to copy some files into this directory on the source box:

```
/var/www/sync_test
```

And again, watch for any errors on the screen, if these come back as a failed connection it'll be an SSH/key issue; common, and not too difficult to solve. From here add some directories and watch how they're queued up, and then take a look at them on the remote box: from this point out it "just works." Now give it more to do by adding files and directories, and then the logging for errors while they sync. As it stands the system uses the source system as the preferred environment, so any files that change, or are added or removed, will be processed on the remote system. This is analogous to how Dropbox works, you can use multiple sources (your laptop, your desktop, etc) and their server serves as the remote system, keeping all the clients in line.

Reprinted with permission of the original author.
First appeared in *http://hn.my/dbclone/*.

Conclusion

You should now have a basic, working Dropbox style setup for your own personal use. I had this running and used it to sync my netbook back to my home server, and then have my work desktop sync to my home server, so both the netbook and the desktop would stay in sync without me doing anything besides putting files in the specified folder. For my week long test I ran a directory alongside my Dropbox directory just to see how they both acted, and I didn't have any failures along the way.

Epilogue

This article is an updated version of one that originally appeared on *http://fak3r.com/* in September 2009 under the title "HOWTO build your own Dropbox clone." In the year since it's publication I've received a great deal of interest in my idea, and have continuously thought of ways to improve upon it. In the configuration file for lsyncd it has a line that reads, "Specify the rsync (or other) binary to call," and this is the kind of flexibility I needed. Today I'm utilizing Unison to handle the syncing for the project, and besides having many attractive features, it's the right solution to do true two-way syncing. This fits the one-to-many Dropbox model better than rsync does. The project has now been released as open source under the name lipsync, and is available here: *https://github.com/philcryer/lipsync* Take it, try it out and improve upon it. If you have troubles ping me on my blog, contact me via GitHub or email; I'm happy to help. Thanks. ■

Phil Cryer is a husband, father, artist, music lover, hacker, open source technologist and civil liberties activist. He currently works as a senior systems engineer currently building a global, distributed, storage network. He hold a bachelor's degree in fine arts, and believes that imagination is more important than knowledge. He can be reached at *http://philcryer.com*.