Ultimate server redux

In the first in an exciting series of features, **Jonni Bidwell** lays the foundations of what will be the *Linux Format* reference server.

Writing tutorials is a tricky game. Most of our tutorials are self-contained, which on the whole is handy—there's no need to refer to a previous issue or other source to do what needs doing. It does mean, however, that much matter is devoted to initial set up, which in many ways is not really the interesting part. So to mix things up a little we present to you, dear readers, the blueprint for our ultimate home server. In future issues we will describe optional additions, diversions and other augmentations to this, and this feature will always be freely available online in case anyone misses (or loses) this issue. In part one we shall cover the bread and butter tasks: Installing an operating system, setting up a static IP address and arming some defences against nefarious hackers.

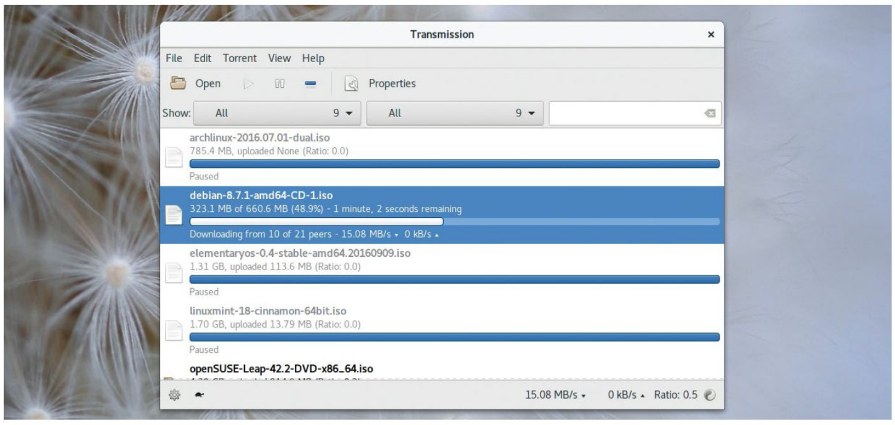
"We present to you, dear readers, the blueprint for our ultimate home server."

Canny readers will recall that we ran an Ultimate Home Server feature back in LXF213 [Features, p32]. That was generally well-received, and much of it will inform this article, but this time around things will be even more 'ultimate'. See the box (*Hardware Considerations, below*) for some hardware guidelines, but by all means feel free to improvise. There's nothing wrong with using old hardware, but there's a lot wrong with relying on it to store valuable data. So take extra care with backups if that's all that's available to you. We won't cover physically putting the machine together, but if you want some tips check out Zak Storey's Fastest Linux PC feature [See p46, LXF219].

One criticism of the 'penultimate' feature was the RAID set up: 'too complicated', 'not worth bothering with' and 'no guidelines on how to recover from a failure'. So this time around we won't bother with that. If you do plan on setting up RAID, and it's a good idea if you have some spare drives, see that feature, our *mdadm* tutorial in **LXF206** [Tutorials, p76], or our Next-Gen Filesystems feature [see p48, **LXF193**]. In particular, if you plan on using ZFS filesystem, then you'll want to invest in error correcting memory and probably will benefit from having more than 4GB.

Deploy Debian

We chose Debian for our server's OS last time and it's an excellent choice so we're jolly well going to stick with it for this outing. We don't care about getting the latest version of Gnome or a bleeding edge kernel—we want stability and security (which would be Debian's middle names, if it had middle names, but it doesn't). At the time of writing, the latest version is 8.7 and ISOs/torrents can be downloaded from **https://debian.org/CD**. A live image is available, but we have no need of trying before 'buying', so grab either the small Network Install image (~250MB) or CD 1 of the install set (you almost certainly will want the amd64 edition—32-bit architecture is officially old now). Subsequent packages can be added in later, assuming a network connection is available. Apropos to this, we'll assume that a wired connection is available. Wireless adaptors nowadays may boast about using 802.11ac, beamforming and quantum resonance (one of these is a joke) to achieve connection speeds in gigabits, but in reality this is hard to achieve. If getting a cable to your server's location is messy, then get some powerline adaptors—the tech has come a long way since the early days, where regular unplugging and plugging back in again were the order of the day.


Transmission

File Edit Torrent View Help

Open Properties

Show: All 9 All 9

archlinux-2016.07.01-dual.iso
785.4 MB, uploaded None (Ratio: 0.0)
Paused

debian-8.7.1-amd64-CD-1.iso
323.1 MB of 660.6 MB (48.9%) - 1 minute, 2 seconds remaining
Downloading from 10 of 21 peers - 15.08 MB/s ▾ 0 kB/s ▴

elementaryos-0.4-stable-amd64.20160909.iso
1.31 GB, uploaded 113.6 MB (Ratio: 0.0)
Paused

linuxmint-18-cinnamon-64bit.iso
1.70 GB, uploaded 13.79 MB (Ratio: 0.0)
Paused

openSUSE-Leap-42.2-DVD-x86_64.iso

15.08 MB/s ▾ 0 kB/s ▴ Ratio: 0.5

› Downloading via BitTorrent is preferred, if only to see ludicrous speeds such as this.

So assuming our machine powers up, stick in the Debian disc/USB stick and reboot.

"With SSH working, you should be able to get rid of the monitor, keyboard and mouse."

You will need to disable Secure Boot if it's enabled, but UEFI installation is supported. A graphical installer is available from the boot menu, but the textual one is perfectly fine. You'll be asked the usual questions about language, location and keyboard layout. Then we must choose a hostname, we'll use lxfserver, but we know that names are powerful (Mu'adib), sentimental and hard to choose under pressure. Fortunately, the hostname can be changed at any time. Leave the domain name blank, unless you have a reason not to, and also leave the root password blank which has the effect of disabling the root account. Next, set up a regular user and password, which will be granted sudo rights for privileged commands. Next we must partition our disks. If you have only one drive, then we'll need to create a data partition on it during the installation, where as if we have two (or more) then we can follow the default scheme for our OS drive (a small EFI partition, a large ext4 partition and a small swap partition) and have a single large ext4 partition on the other drive. Set the data partition to be mounted at **/mnt/data** to save fiddling around with **/etc/fstab** later. Select 'Finish Partitioning and Write Changes to Disk', confirm and the base system will be installed. Once that's done you'll be prompted to add a network mirror, which is a good idea if you have a working network connection (it'll update packages from a server close to you) and a bad idea if you don't (it won't work), so choose appropriately. Then you can add some package groups. We don't need a desktop environment, so unselect this one, but Print Server, SSH Server and System Utilities are all »

Hardware considerations

You can make a home server out of any old bits and pieces you have lying around, but that doesn't mean you should. At least not as regards old disk drives and power supplies—these things have a habit of failing as soon as you start relying on them, and we would rather our server be reliable.

In terms of processing power, a dual-core chip from the last decade will suffice for most things you might want your server to do: host files, print and run web services. However, if you plan to use it for streaming movies around the house (e.g. with *Emby*), something more powerful (like a recent Core i5) will be better—on the fly video transcoding is quite an onerous chore for older CPUs. Memory (at least DDR3 memory) is cheap nowadays and 4GB will be more than enough for most purposes. You can get away with much less, but it's better to have more. Integrated graphics (such as are found in all modern Intel chips and AMD APUs) or the cheapest of GPU cards will be fine. Once the OS is installed we won't even need the monitor, or mouse or keyboard. We'll want a large hard drive for storing data. We'll put the operating system and data on separate partitions. Really they should be on separate drives too, and we'd encourage readers to invest extra money in another drive (a small SSD would be a good suggestion) to make this possible.

» useful. Once all this is installed you can reboot into your freshly minted server.

Before we do anything, we'll want to set our machine up with a static IP address. This will make it easy to find our server from other machines on the network and access its resources. By default Debian obtains an address from your router via DHCP. This is good because it means connectivity is alive without us having to do anything, but bad because the IP address you are assigned today may very well be different tomorrow. You can see what your current address is with the `ip a` command. Each network interface gets its own name, there ought to be at least a stanza for the loopback interface `lo` and your Ethernet card, probably `eth0`. The line:

```
inet 192.168.1.100/24 brd 192.168.1.255 scope
global eth0
```

in the `eth0` stanza betrays its IPv4 address (**192.168.1.100**), perhaps in the future we will deal with IPv6 addresses. We also will need the address of our router (since DHCP also provides routing and DNS information) which you may already know but if not can readily discover by typing `routel` and looking in the Gateway column. Don't worry if your router's address looks different, there are a number of different blocks reserved for private networks—some routers use **10.***, but **192.168.*** seems to be more popular. We'll suppose our router's IP is **192.168.1.254**.

Network config

Network configuration in Debian is all handled by the file ***/etc/network/interfaces***. So let's edit this file (you may wish to back it up first) with `sudo nano /etc/network/interfaces`. Replace the line:

```
iface eth0 inet dhcp
```

with the following block (leaving intact any preceding lines such as `allow-hotplug eth0`):

```
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    gateway 192.168.1.254
```

You can change the last digit of the `address` to anything less than 255 that isn't already in use on your network, but the `gateway` line must correspond with your router. Save this file with Ctrl+X, y, Enter. We also need to tell Debian to use our router for DNS lookups (which it passes to your ISP). This time edit the file ***/etc/resolv.conf*** and replace any `nameserver` lines with a single:

```
nameserver 192.168.1.254
```

You may prefer to use Google's DNS here (8.8.4.4 and 8.8.8.8), as many UK ISPs have flaky DNS servers or block certain lookups.

Now we can activate our new network configuration with:

```
$ sudo systemctl restart networking
```

And test it with:

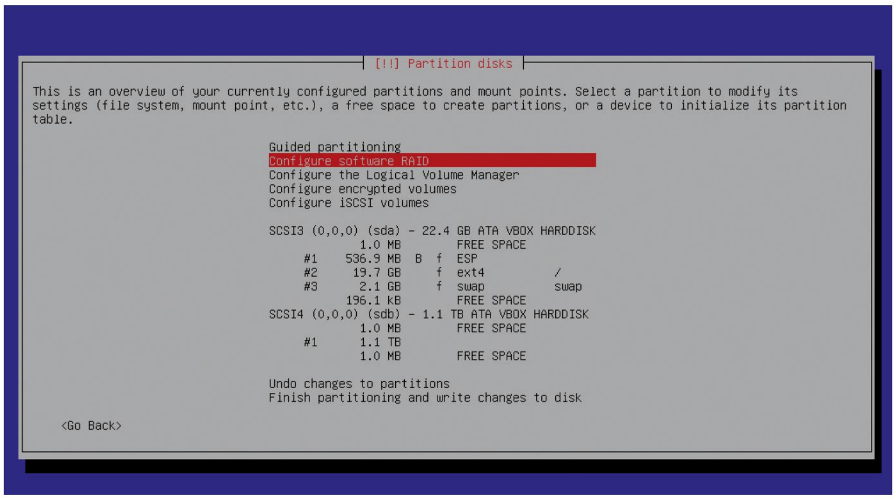
```
$ ping -c4 google.com
```

If four packets are safely returned then huzzah! Else further tweaks will be necessary.

At this point, we should be able to SSH into our server from another machine on the network. Provided we remembered to tick the SSH Server box during install, that is. If not `sudo apt-get install ssh-server` will do the trick. We can access our server from Windows via the *PuTTY* program, or through the new-fangled Windows Subsystem for Linux (WSL), although setting WSL up is beyond the scope of this feature. Or you can do it the grown-up way from a Linux box with a simple:

```
$ ssh lxfuser@192.168.1.100
```

With SSH working, you should be able to get rid of the monitor, keyboard and mouse attached to your server. They might come in handy later if it breaks so don't go defenestrating them just yet, but for now



```
[!!] Partition disks

This is an overview of your currently configured partitions and mount points. Select a partition to modify its
settings (file system, mount point, etc.), a free space to create partitions, or a device to initialize its partition
table.

                                    Guided partitioning
                                    Configure software RAID
                                    Configure the Logical Volume Manager
                                    Configure encrypted volumes
                                    Configure iSCSI volumes

                                    SCSI3 (0,0,0) (sda) - 22.4 GB ATA VBOX HARDDISK
                                                1.0 MB        FREE SPACE
                                         #1   536.9 MB  B  f  ESP
                                         #2    19.7 GB     f  ext4          /
                                         #3     2.1 GB     f  swap          swap
                                              196.1 kB        FREE SPACE
                                    SCSI4 (0,0,0) (sdb) - 1.1 TB ATA VBOX HARDDISK
                                                1.0 MB        FREE SPACE
                                         #1     1.1 TB
                                                1.0 MB        FREE SPACE

                                    Undo changes to partitions
                                    Finish partitioning and write changes to disk

      <Go Back>
```

› **This is how your disk layout should look if you have a separate drive for storing data on. (Yes, we did cheat and use *VirtualBox* to do this.)**

SSH keys

Logging into SSH using a key involves first generating a private key and a public key. It's your responsibility to keep the private key as secret as can be. It's convenient to have a copy on every machine you log into the server from, but also insecure—if one of those machines was stolen it represents a vector by which our server could be compromised. Ideally you should keep the private key on a USB stick and not lose it.

The public key can be just that, and a copy of it is stored on the server to verify the private key via mathematical voodoo. To generate a keypair (ideally on a machine that you'll access the server from, rather than the server itself) run the command `ssh-keygen`. Accept the default location and choose a password for your key.

This adds an extra layer of security in the event the key is purloined. Copy the key to the server with `ssh-copy-id lxfuser@192.168.1.100`, it will be appended to the file ***/home/lxfuser/.ssh/authorized_keys*** on the server. We can test it works with:

```
$ ssh lxfuser@192.168.1.100
```

All going well we shouldn't be asked for a password this time. If you want to disable password logins altogether, then you will need to add the directive:

```
PasswordAuthentication no
```

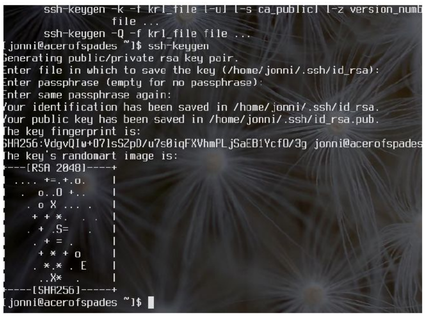
to ***/etc/ssh/sshd_config***.

The keypair itself lives in the ***~/.ssh/*** folder, in the files **id_rsa.pub** and **id_rsa.priv**. The latter file is the private key, and should be treated with care. If you copy it elsewhere, you can tell SSH to log in with `ssh -i /path/to/key` but it will be rightly fussy about permissions.

Since filesystems commonly used on USB sticks (such as FAT32 and NTFS) don't support Linux permissions, you will need to copy your key off such media (and then run:

```
chmod 600 id_rsa
```

before attempting to log in.



```
       ssh-keygen -k -f krl_file [-u] [-s ca_public] [-z version_numb
                    file ...
       ssh-keygen -Q -f krl_file file ...
[jonni@acerofspades ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jonni/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jonni/.ssh/id_rsa.
Your public key has been saved in /home/jonni/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:VdgvQIw+07lsSZpD/u7s0iqFXVhmPLjSaEB1Ycf0/3g jonni@acerofspades
The key's randomart image is:
+---[RSA 2048]----+
| .... +=.+.o.    |
|  .   o..O +..   |
|   . o X ... .   |
|    + + *.  . .  |
|   . + .S=    .  |
|    . + = .      |
|     + * + o     |
|    . *.* . E    |
|     ..X*   .    |
+----[SHA256]-----+
[jonni@acerofspades ~]$
```

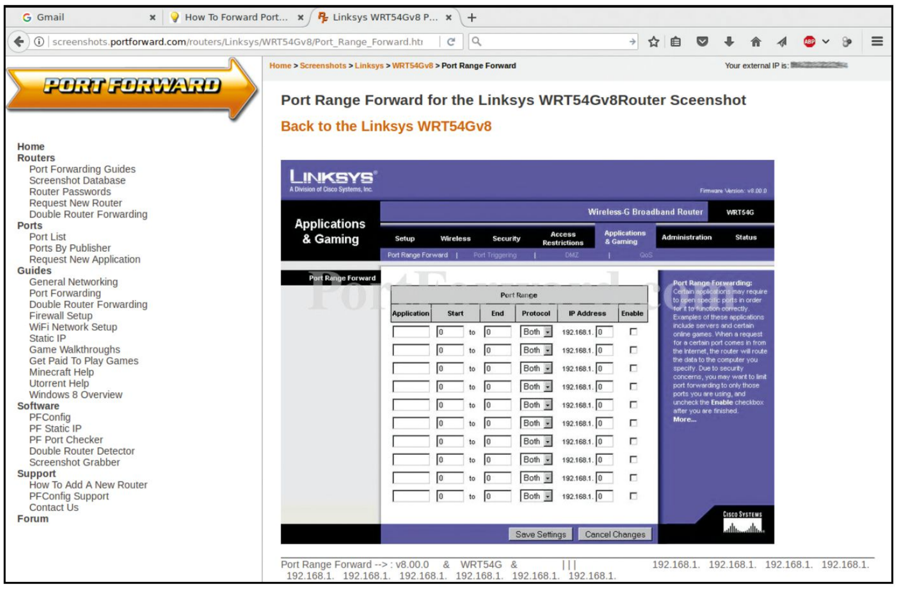
› **ssh-keygen gives you a pictorial representation of your public key—who says cryptography isn't aesthetically pleasing?**

enjoy the comfort that goes with knowing your server can be trusted to do its thing and remain accessible from any computer on your home network. If you trust fellow users of this network, then there isn't too much reason to be paranoid about choosing a lengthy and complicated password for your user. However, if you choose to make your server accessible to the whole world (see later) then a strong passwords is essential. You can even go one better and use public key crypto to better secure logins. (*See the SSH Keys box, left*).

One aspect of SSH that's often overlooked is its ability to securely transfer files via SFTP. Any modern file manager will enable you to log into your server and copy files thereto and therefrom by visiting the URI **sftp://lxfuser@192.168.1.100**. Some file managers have done away with address bars that you can actually type into, but Ctrl+L usually does the trick, Gnome's *Files* also allows you to input server addresses from the Other Locations section of the sidebar. Remember you'll only be able to do read and write to locations on the server where `lxfuser` is allowed to read and write.

Opening the floodgates

There's plenty of reason not to make your SSH service available to the whole world. But being able to do a spot of admin remotely is also pretty handy. Most home routers now have a handy interface for forwarding ports, but they're all different so you're on your own here. We want to make TCP port 22 (SSH) on our server available to the world, so we need to forward a TCP port (and it's a reasonable idea to make it different from 22, e.g. 10022 to


Gmail × | How To Forward Port... × | Linksys WRT54Gv8 P... × | +

screenshots.portforward.com/routers/Linksys/WRT54Gv8/Port_Range_Forward.htm

PORT FORWARD

Home > Screenshots > Linksys > WRT54Gv8 > Port Range Forward   Your external IP is: [redacted]

Home
Routers
- Port Forwarding Guides
- Screenshot Database
- Router Passwords
- Request New Router
- Double Router Forwarding

Ports
- Port List
- Ports By Publisher
- Request New Application

Guides
- General Networking
- Port Forwarding
- Double Router Forwarding
- Firewall Setup
- WiFi Network Setup
- Static IP
- Game Walkthroughs
- Get Paid To Play Games
- Minecraft Help
- Utorrent Help
- Windows 8 Overview

Software
- PFConfig
- PF Static IP
- PF Port Checker
- Double Router Detector
- Screenshot Grabber

Support
- How To Add A New Router
- PFConfig Support
- Contact Us

Forum

Port Range Forward for the Linksys WRT54Gv8Router Sceenshot

Back to the Linksys WRT54Gv8

LINKSYS A Division of Cisco Systems, Inc.   Firmware Version: v8.00.0

Applications & Gaming   Wireless-G Broadband Router   WRT54G

Setup | Wireless | Security | Access Restrictions | Applications & Gaming | Administration | Status

Port Range Forward | Port Triggering | DMZ | QoS

Port Range Forward

Port Range

Application	Start		End	Protocol	IP Address	Enable
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐
	0	to	0	Both	192.168.1. 0	☐

Port Range Forwarding: Certain applications may require to open specific ports in order for it to function correctly. Examples of these applications include servers and certain online games. When a request for a certain port comes in from the Internet, the router will route the data to the computer you specify. Due to security concerns, you may want to limit port forwarding to only those ports you are using, and uncheck the **Enable** checkbox after you are finished. **More...**

Save Settings   Cancel Changes

CISCO SYSTEMS

Port Range Forward --> : v8.00.0 & WRT54G & | | | 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1. 192.168.1.

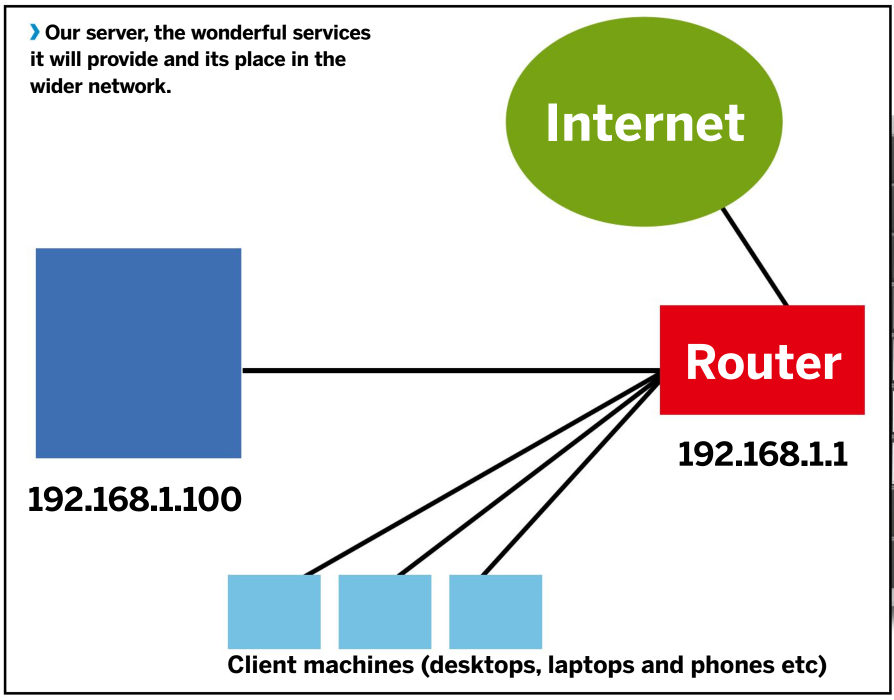
› If all else fails, the portforward.com website will help you figure out how to do just that on your router. However, the process should be straightforward.

lessen bot traffic) from our router's external IP. Unfortunately for most people, that IP address will change with the weather, but this problem can be circumvented using a dynamic DNS provider (such as the free and excellent **www.duckdns.org**). By running a script on your server, the provider is informed of any changes to the server's IP and DNS records are updated, so your server remains accessible through an invariant hostname, such as **lxfserver.duckdns.org**.

Once your server is publicly accessible, it won't take long before bots start trying to log in with common usernames and passwords. We can mitigate against this using the *fail2ban* program, which we can install with `sudo apt-get install fail2ban`. *Fail2ban* can work with any service, and once specified number of failed logins (or just requests to a web server) are recorded then a temporary ban is implemented via *iptables*. The default settings are fine for our purposes, but it's good practice to copy the main configuration file and make any changes to a local file:

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/
jail.local
```

Have a poke around in **/etc/fail2ban/filter.d** to see how the various filters work. And tune in for our next exciting instalment next month [cue moody orchestral music]. LXF


› Our server, the wonderful services it will provide and its place in the wider network.