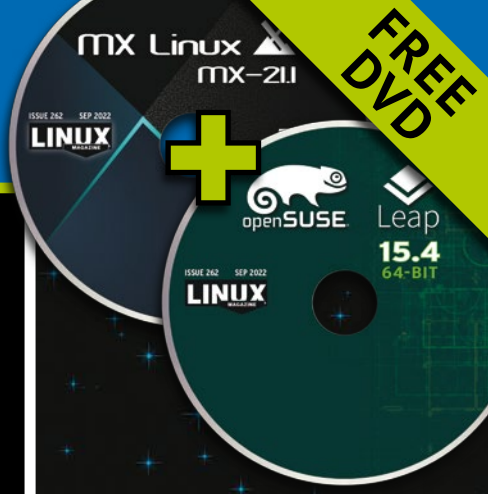
WHAT'S NEW IN
UBUNTU 22.04



LINUX PRO MAGAZINE

ISSUE 262 – SEPTEMBER 2022

Beyond 5G

Imagining an open future for mobile networks

FLUX Beamo
Cool laser cutter with Rasp Pi on the inside

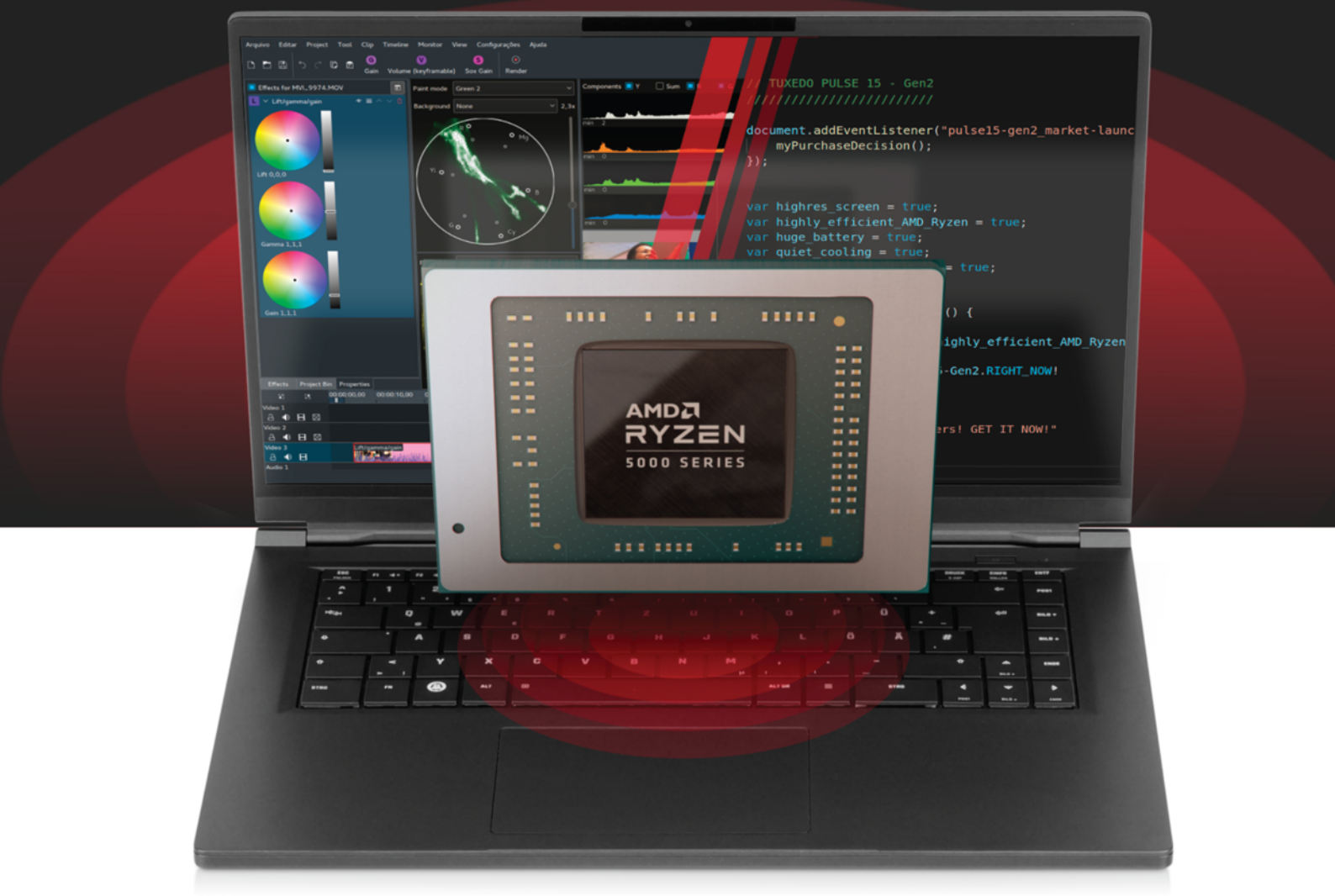
Free Social Media Tools
Get connected without getting mined

Manuskript
Planning a novel doesn't have to be crime and punishment

Practical automation with mosquitto and MQTT

10 TANTALIZING FOSS TOOLS

LINUX NEW MEDIA
The Pulse of Open Source



Qui(e)te powerful!

TUXEDO Pulse 15 - Gen2

AMD Ryzen 7 5700U-35W
8 cores | 16 threads

WQHD display
2560 x 1440 | 165 Hz

Up to 18 h of runtime
91 Wh battery

Rigid magnesium chassis
1,7 cm thin | 1,5 kg light

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS 18th ANNIVERSARY

tuxedocomputers.com

apply now!

Looking for a new job?
tuxedocomputers.com/jobs

CAP AND GOWN

Dear Reader,

A few months ago, I wrote about the need for the open source community to provide a means for the continuity of solo or small-scale open source projects when the maintainer reaches a burnout point or gets busy with something else. Does the project have to die just because the person who started it walks away?

Another angle on this problem is the question of scientific software written by research professionals. Some of the most sophisticated software in the world is created by doctoral candidates and other academic scientists. This software isn't written just to be software – it is written to test a new idea or answer a question related to a research project. Some of these programs represent years of work, but what happens when the developer graduates or gets a tenure track job? Or when the grant used to fund the research expires? More often than not, the project just stops in its tracks and slowly disappears, while the developer seeks new projects and new funding to study other questions.

Academic science is focused on journal articles, not software. The software is a means to an end, so many useful programs are abandoned, and researchers end up reinventing the wheel. Don't ask the PhDs and PhD candidates to solve this problem. No one has ever gotten a distinguished chair for maintaining already-existing software that only a few experts can even understand.

The prospects for orphaned scientific software have become a little brighter with a recent announcement from the Virtual Institute for Scientific Software (VISS) [1]. VISS, which is supported by Schmidt Futures [2], a nonprofit organization founded by former Google CEO Eric Schmidt and his wife Wendy Schmidt, is launching four software development centers at the University of Cambridge, the University of Washington Seattle, Georgia Institute of Technology, and Johns Hopkins University. These centers, which will each employ five to seven software developers, will provide development and support services for scientific projects. Initially, the centers will only work on projects associated with Schmidt Futures, but the hope is to extend that support to other worthy research.

In addition to keeping the software alive after grants end and participants move on, the centers will provide assistance in the initial development phase. Another important, but perhaps less tangible, goal will be to build the identity of the professional academic programmer. Hundreds of professional developers are working right now at universities around the world, but they are often isolated, scattered across the campus, and working independently through separate, unrelated grants. The VISS centers offer the possibility for a collective experience, with the kind of mentoring, work sharing, and synergy that is an everyday part of software development out in the wild.

According to a recent article in *Nature* [3], VISS is well aware that it can't compete with Internet giants like Amazon and Google in paying top salaries, but they are confident they can still attract high-quality talent. Many professional developers first became interested in coding through their work in science and engineering, and to some, the chance to work on scientifically relevant projects is more exciting than maxing out their salary potential. (And, to be honest, they will probably still fare pretty well compared to a lot of people hanging around a college campus.)

Given the amount of scientific software out in the world today, the addition of 20-30 coders in four small offices won't change the landscape overnight, but the VISS initiative will help to raise awareness about the need to support scientific programming, and it could offer a prototype of a permanent career path for coders who aspire to play a role in the eternal quest for scientific knowledge.

Joe Casad,
Editor in Chief

Info

[1] VISS: *https://www.schmidtfutures.com/our-work/virtual-institute-for-scientific-software/*

[2] Schmidt Futures: *https://www.schmidtfutures.com/*

[3] "Ex-Google Chief's Venture Aims to Save Neglected Science Software" by David Matthews, *Nature*, July 13, 2022: *https://www.nature.com/articles/d41586-022-01901-x*

LINUX MAGAZINE
SEPTEMBER 2022

ON THE COVER

NEWS

REVIEWS

COVER STORY

IN-DEPTH

openSUSE Leap 15.4 and MX Linux 21.1

Two Terrific Distros on a Double-Sided DVD!

openSUSE Leap 15.4
64-bit

Leap 15.4 is the latest stable release of openSUSE, the popular and well-established community-based distribution. Leap 15.5 is not scheduled to replace it until June 2023.

Because the 15.4 release is built upon a mature platform, most of its features are updates, and often behind the scenes. Several changes are enhancements and improvements to sudo and visudo, and backports and bug fixes to systemd. In addition, the release has major updates to AppArmor, as well as to productivity tools such as Firefox and LibreOffice.

Perhaps the most noticeable change is the introduction of DNF, the package manager which has largely replaced Yum in other RPM distros such as Fedora. While Yum remains in openSUSE, the 15.4 release can be configured to use DNF. In addition, the 15.4 release also introduces Microdnf, a subset of DNF that can be used to speed up package installation when the whole of DNF is not required.

Like all openSUSE releases, Leap 15.4 is a general-purpose release, suitable for all levels of users.

MX Linux 21.1
64-bit

What do you get when the lightweight antiX distribution collaborates with the once-prominent MEPIS community? Answer: A Debian derivative that has been first in page hits on DistroWatch for three years and shows few signs of slipping.

Install MX Linux, and the reasons for its popularity become obvious immediately. The antiX core makes for a speedy system, and many will approve of MX Linux's minimal reliance on systemd. Newcomers can appreciate the clear and detailed documentation, especially in the installer, and the orienting Welcome screen and tour of the desktop. For more experienced users, the desktop contains a series of tools rarely if ever matched in other distributions. These original tools include graphic interfaces for editing Bash, the boot manager, repositories, the firewall, and advanced backup techniques. Less spectacularly but equally usefully, MX Linux also includes a USB formatter, a Live USB creator, and a Samba configurer – as well as many other tools for tasks that are done from the command line in other distributions.

For more information, see this issue's Distro Walk.

Defective discs will be replaced.
Please send an email to subs@linux-magazine.com.

Web Development Essentials

Modern software applications are commonly developed for the Web. Linux Professional Institute's new Web Development Essentials program supports you in your first steps in software development. It includes learning materials suitable for both training and self-study. Upon passing the Web Development Essentials exam, you receive a certificate to prove your skills.

LPI Learning Materials are available at learning.lpi.org.
Find out more about the program and the exam at lpi.org/wde

Linux Professional Institute

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Rocky Linux 9 Has Arrived

Rocky Linux 9 is now available and is a landmark release for several reasons. First off, there has been a surge in Rocky Linux deployments, putting it ahead of CentOS Stream and AlmaLinux. But more than that, Rocky Linux includes several security enhancements and networking features to help make it a best-in-class open source operating system for businesses of all sizes.

In the new release, you'll find SHA-1 message digest for cryptographic purposes has been deprecated (as the cryptographic hash functions are no longer considered secure). In addition, you'll find OpenSSL 3.0.1 (which includes provider concept, a new versioning scheme, an improved HTTP/HTTPS client, support for new protocols/formats/algorithms, and more), OpenSSH version 8.7p1 (which includes the replacement of the SCP/RCP protocol with the more predictable SFTP protocol), SELinux performance improvements, and the automatic configuration of security compliance settings for PCI-DSS, HIPAA, DISA, and more.

As for the networking improvements, you'll find that MultiPath TCP Daemon can now be used instead of iproute2 for the configuration of MultiPath TCP endpoints. Also, NetworkManager now uses key files to store connection profiles (but still supports ifcfg). Iptables-nft and ipset are deprecated and have been replaced by the nftables framework. Finally, network-scripts has been removed in favor of NetworkManager to configure network connections.

One other major move forward for Rocky Linux is that this version was built with a community-developed, open source, cloud-native system, called Peridot. This Golang project was developed to assure new versions of Rocky Linux can be released within one week after each RHEL version. By migrating to this system, anyone can reproduce Rocky Linux from scratch, ensuring that the distribution will always be available. The source for the Peridot build system can be found on GitHub (*https://github.com/rocky-linux/peridot-releng*).

For more information about the new Rocky Linux release, be sure to read the complete release notes (*https://github.com/rocky-linux/documentation/blob/6d86674106233d3c0ab72da734de8eedee4e6549/docs/release_notes/9_0.md*).

Slimbook Upgrades CPUs in Executive Linux Ultrabook

Slimbook, well known for producing KDE Plasma-powered laptops, has given their Executive series a bit of a refresh by making them available with the Intel 12th Gen Alder Lake CPU. This new iteration adds considerably more power

(14 Cores, 20 threads, 24MB cache, and up to 4.80GHz clock speeds), improved battery life, and better graphics (via an integrated Iris Xe 4K chipset). Consumers will find two different models available with this configuration: A 14-inch 3K display, running at a 90Hz refresh rate (at 2880x1800 resolution) and a 16-inch model that sports NVIDIA RTX 3050Ti graphics with 4GB GDDR6 RAM (also at a 90Hz refresh rate).

The 14-inch model does get a beefier battery (99WHr), whereas the 16-inch model's battery is a smaller 82WHr. Both laptops include USB-C Thunderbolt 4, USB 3.2, HDMI 2.0, and USB-C 3.2 (with Display Port). The keyboards are backlit with large touchpads and the devices can be upgraded up to 64GB DDR4 3200Mhz RAM and up to 4TB NVMe SSD storage. Both versions include a 1080p full HD webcam with integrated stereo mic, WiFi 6, Bluetooth 5.1, 2W stereo speakers, and a Kensington Lock mount. You can select from numerous Linux distributions to be preinstalled (such as Ubuntu, Kubuntu, Debian, elementary OS, Pop!_OS, Linux Mint, and more).

Price starts at approximately $1,322 for the 14-inch model and $1,627 for the 16-inch. Order your new Slimbook Executive now (*https://slimbook.es/en/store/slimbook-executive*).

Fedora Linux Is Coming to the Raspberry Pi 4

Fedora Linux has been available for desktops, servers, and even IoT devices. However, if you wanted to install the OS on the Raspberry Pi 4 device, you were out of luck – until now. With the upcoming release of Fedora 37, support for the devices might well finally become a reality. Although not official, it has become a proposed change and will be implemented if it receives approval from the Fedora Engineering Steering Committee.

The reason the Raspberry Pi 4 has yet to be supported by Fedora Linux has been the lack of accelerated graphics. However, with upstream work on the kernel and Mesa (specifically the V3D GPU for both OpenGL ES and Vulkan), it's now just a matter of enabling support. The one caveat is that support for WiFi on the Raspberry Pi 400 is not a part of this (although testing for audio support is).

According to the Raspberry Pi 4 Fedora Wiki page, "The support for the Raspberry Pi ecosystem has been an ongoing evolution. The aim of this change is to support the Raspberry Pi 4 including the 4B, the 400, and the CM4 with IO board. Upstream now supports accelerated graphics using the V3D GPU for both OpenGL ES and Vulkan. There's also enhancement to wired networking with support for PTPv2 on the CM4/4B."

KaOS 2022.06 Now Available with KDE Plasma 5.25

KaOS was first created in 2013 as a Linux distribution that focuses on the KDE Desktop Environment. The original goal was to create a highly polished KDE experience, which the developers achieved quite well. And with the latest release, that experience is made even better with the addition of KDE Plasma 5.25 and a number of other additions and enhancements.

As to what else has been added to KaOS, you'll see KDE Frameworks 5.95, KDE Gear 22.04.2, Calamares 3.3, LibreOffice as the default office suite (a change from Calligra), Linux kernel 5.17 (which adds improved support for GPUs), and a new package selection addition to the onboarding Welcome screen.

Of course, at the heart of KaOS 2022.06 is KDE Plasma 5.25, which adds a host of improvements, including improved gesture support, tints for window accent colors, a much-improved touch mode, 10-bit color for Kdenlive, many enhancements to Kate, and an enhanced Settings dialog.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

ADMIN HPC
http://www.admin-magazine.com/HPC/

Sharing Linux Terminals
• Jeff Layton
Sometimes sharing a screen between two users is enormously helpful. We look at two terminal sharing tools: screen and tmux.

Performance Health Check
• Jeff Layton
Many HPC systems check the state of a node before running an application, but not very many check that the performance of the node is acceptable before running the job.

ADMIN Online
http://www.admin-magazine.com/

Three Full-Text Desktop Search Engines
• Harald Jele
Desktop search engines such as Tracker, DocFetcher, and Recoll help track down files by their content, even in massive datasets.

Stretching Devices with Limited Resources
• Federico Lucifredi
Compressed memory solutions for small memory problems.

Portable Home Directory with State-of-the-Art Security
• Martin Loschwitz
The systemd Homed service makes it easy to move your home directory, and FIDO2 or PKCS#11 can secure the stored files.

For those that are curious, LibreOffice became the default office suite, since it is now available as a single Qt application.

To read more about the latest release, check out the KaOS news page (*https://kaosx.us/news/2022/kaos06/*) and download an ISO from *https://kaosx.us/pages/download/*.

Manjaro 21.3.0 Now Available

Manjaro is an Arch-based, rolling-release Linux distribution aimed at users who want the power and flexibility of Arch, without complications. The latest release, version 21.3.0, includes plenty of newness, in the form of the Calamares installer (which now supports LUKS partitions), Gnome 42 (which includes Libadwaita), KDE Plasma 5.24 (which includes the new Overview), Xfce 4.16 (which now supports fractional scaling), and Linux kernel 5.15 LTS.

The Gnome edition of Manjaro 21.3.0 includes both GTK 4 and Libadwaita, which offers improved performance, a modern UI style, and plenty of new user interface elements. The Plasma edition makes it possible to more easily move panels around and stick them to any edge you like. The Xfce edition received quite a number of updates, especially in the area of compositing and GLX.

Since Manjaro is a rolling edition, if you already have the OS installed, an update should bring you to the latest release. If you don't have Manjaro installed, you can download the latest ISO from the official Manjaro Download page (*https://manjaro.org/download/*), where you have the choice to download the full or a minimal version of the OS. You'll find downloads for x86_64 and ARM architecture as well as unofficial spins for the Budgie, Cinnamon, i3, Sway, Mate desktops, and even a Docker image.

SpiralLinux: a New Linux Distribution Focused on Simplicity

SpiralLinux (*https://spirallinux.github.io/*) is Debian-based Linux distribution with spins for Cinnamon, Xfce, Gnome, KDE Plasma, Mate, Budgie, LXQt, and Builder (which uses the IceWM window manager for experienced users to fully configure the system to meet their needs). Each of these spins (minus "Builder") offers a simplified Linux experience that uses the official Debian Stable package repositories.

SpiralLinux includes Flatpak support built-in as well as a GUI front end for managing Flatpak packages. The distribution uses the Btrfs filesystem which includes an optimal sub-partition with Zstd transparent compression and built-in support for automatic Snapper snapshots and even zRAM swap support.

According to the developer, "Great effort has been expended in polishing the SpiralLinux default configuration for all the major desktop environments using the packages and mechanisms that Debian itself provides," said the developer when asked "why another Debian-based distro."

SpiralLinux is based on Debian 11 ("Bullseye"), is powered by kernel 5.16, and ships with apps like Firefox, LibreOffice, Thunderbird, Transmission, Pidgin, and the Synaptic Package Manager installed by default.

For more information about the latest release of SpiralLinux, make sure to visit the official release notes (*https://github.com/SpiralLinux/SpiralLinux-project/releases*).

Zack's Kernel News

Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

Random Number Sanity

Random numbers are important for security. Generally to make random numbers, you grab entropy from somewhere, like the frequency of fingers tapping a keyboard, and use that to generate as many unpredictable numbers as needed. But what if no one's typing on the keyboard? What if you run out of entropy? Should the system just sit and wait for more?

For a long time, the Linux kernel had to choose between locking up the system until it found enough entropy to make truly random numbers and providing numbers anyway, even if they weren't really random enough.

But in 2019, Linus Torvalds wrote a patch that addressed the problem "by actively generating entropy noise using the CPU cycle counter when waiting for the random number generator to initialize. This only works when you have a high-frequency time stamp counter available, but that's the case on all modern x86 CPUs, and on most other modern CPUs too."

Because of that patch, the random number generator would pretty much always have enough entropy to produce useful random numbers. Recently, Jason A. Donenfeld felt it was time for everyone to admit it was working, get over their mistrust of technology and new things, and allow the Linux random number generator to block while building up entropy.

This is inherently a controversial topic. If the system waits for a resource that never becomes available, it'll just wait forever, and then you've got a brick on your hands. There have been various attempts to introduce blocking into the random number generator, but they were taken out shortly after going into the source tree.

But Jason said, "given that the kernel has grown this mechanism for seeding itself from nothing, and that this procedure happens pretty fast, maybe there's no point any longer in having /dev/urandom give insecure bytes. In the past we didn't want the boot process to deadlock, which was understandable. But now, in the worst case, a second goes by, and the problem is resolved. It seems like maybe we're finally at a point when we can get rid of the infamous 'urandom read hole'."

He added, "This patch goes a long way toward eliminating a long overdue userspace crypto footgun. After several decades of endless user confusion, we will finally be able to say, 'use any single one of our random interfaces and you'll be fine. They're all the same. It doesn't matter'. And that, I think, is really something. Finally all of those blog posts and disagreeing forums and contradictory articles will all become correct about whatever they happened to recommend, and along with it, a whole class of vulnerabilities eliminated."

The "footgun" Jason mentioned is not some kind of obscure security-related attack vector; it's just slang for something that makes it easy to shoot yourself in the foot.

But Andy Lutomirski made just about the strongest possible objection that anyone can make against a Linux kernel patch – saying that it violated the application binary interface (ABI) – and while doing it he offered some historical perspective on this whole issue. He said:

*"This patch is 100% about a historical mistake. Way back when (not actually that long ago), there were two usable interfaces to the random number generator: /dev/random and /dev/urandom. /dev/random was, at least in principle, secure, but it blocked unnecessarily and was, therefore, incredibly slow. It was totally unsuitable for repeated use by any sort of server. /dev/urandom didn't block but was insecure if called too early. *But* urandom was also the correct interface to get best-effort-i-need-them-right-now random bits. The actual semantics that general cryptography users wanted were not available.*

"Fast forward to today. /dev/random has the correct semantics for cryptographic purposes. getrandom() also has the correct semantics for cryptographic purposes and is reliable as such – it is guaranteed to either not exist or to DTRT [do the right thing]. And best-effort users can use GRND_INSECURE or /dev/urandom.

"If we imagine that every user program we care about uses GRND_INSECURE for best-effort and /dev/random or getrandom() without GRND_INSECURE for cryptography, then we're in great shape and this patch is irrelevant.

*"But we don't get to rely on that. New kernels are supposed to be compatible with old userspace. And with *old* userspace, we do not know whether /dev/urandom users want cryptographically secure output or whether they want insecure output. And there is this window during boot that lasts, supposedly, up to 1 second; there is a massive difference.*

"So, sorry, this patch is an ABI break. You're reinterpreting any program that

Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

wanted best-effort randomness right after boot as wanting cryptographic randomness, this can delay boot by up to a second, and that's more than enough delay to be considered a break.

*"So I don't like this without a stronger justification and a clearer compatibility story. I could *maybe* get on board if you had a urandom = insecure boot option to switch back to the old behavior and a very clear message like 'random: startup of %s is delayed. Set urandom = insecure for faster boot if you do not need cryptographically secure urandom during boot', but I don't think this patch is okay otherwise.*

"Or we stick with the status quo and make the warning clearer. 'random: %s us using insecure urandom output. Fix it to use getrandom() or /dev/rando as appropriate'."

So Andy wasn't disputing that Jason's patch would work – he was pointing out that it would change the system behavior in ways that might break existing compiled userspace binaries. In other words, it would break the kernel ABI. If true, Jason's patch would have a giant hurdle to clear. In general, the only thing Linus hates worse than breaking the ABI is allowing a security hole to go unpatched. But if it can be shown that no users rely on a particular piece of the ABI, Linus has been known to allow a patch to change it.

In this case, Jason said to Andy:

"I think your analysis is a bit mismatched from the reality of the situation. That reality is that cryptographic users still find themselves using /dev/urandom, as that's been the 'standard good advice' for a very long time. And people are still encouraged to do that, either out of ignorance or out of 'compatibility'. The cryptographic problem is not going away.

"Fixing this issue means, yes, adding a 1 second delay to the small group of init system users who haven't switched to using getrandom(GRND_INSECURE) for that less common usage (who even are those users actually?). That's not breaking compatibility or breaking userspace or breaking anything; that's accepting the reality of _how_ /dev/urandom is mostly used – for crypto – and making that usage finally secure, at the expense of a 1 second delay for those other users who haven't switched to getrandom(GRND_INSECURE) yet. That seems like a _very_ small price to pay for eliminating a footgun.

"And in general, deemphasizing the rare performance of the less common usage in favor of fixing a commonly triggered footgun seems on par with how things morph and change over time. There's no actual breakage. There's no ABI change violation. What you're saying simply isn't so."

Theodore Ts'o joined the discussion on Jason's side of the argument. He said, "So long as we're only blocking for short amount of time, and only during early after the system was booted, people shouldn't care. The reason why we had to add the 'gee-I-hope-this-jitterentropy-like-hack-is-actually-secure on all architectures but it's better than the alternatives people were trying to get Linus to adopt' was because there were systems that were hanging for hours or days."

But the proof is in the pudding, and a week or so after Jason's patch went into the kernel tree, Guenter Roeck pointed out that it caused "a large number of qemu boot test failures for various architectures (arm, m68k, microblaze, sparc32, xtensa are the ones I observed). Common denominator is that boot hangs at 'Saving random seed:'. A sample bisect log is attached. Reverting this patch fixes the problem."

At this point Linus joined the conversation, saying, "Ok, it was worth trying, but yeah, it clearly causes problems for various platforms that can't do jitter entropy and have nothing else happening either."

And he reverted the patch.

Jason also commented on Guenter's post, saying, "As Linus said, it was worth a try, but I guess it just didn't work."

But the story didn't end there. Jason asked Guenter to share some of the virtual machines that seemed to break with the patch, and the two of them – and others – proceeded to track down the real reasons why Qemu had a problem with Jason's patch. As Jason put it, "if we do ever reattempt this sometime down the road, it seems like understanding everything about why the previous time failed might be a good idea."

In fact, it turned out to be a Qemu bug, rather than a problem with Jason's random number generator modifications. To Jason, this meant that "the rationale for reverting the /dev/random + /dev/urandom unification has now been fixed. That's some real tangible progress."

But he remained cautious. Jason went on to say, "I don't want to rush into trying the unification again too soon. I think if anything, the lesson from the first attempt wasn't simply, 'I should fix a few of Guenter's test cases,' but rather that the problem is fairly nuanced and will take a lot wider testing and research. However, the fact that the initial thing, across multiple platforms, that lead to the revert has been fixed gives me a decent amount of optimism that at /some point/ down the road, we'll be able to try this again. One step at a time."

Ultimately, it seems that Linus does not consider the issue in its entirety to be an unacceptable ABI violation. And he is willing, if not eager, to take something like Jason's patch if it works. So in theory, Jason may be bringing a little more sanity to random number generation in the near future.

Git Lesson from Linus

Borislav Petkov submitted a pull request to Linus Torvalds against the 5.18 kernel tree, but he noticed that his workflow had resulted in a strange diffstat. Diffstats are auto-generated summaries of which files were changed and by how much. It's one way the kernel developers can quickly see which parts of a patch might interest them. In this case, Borislav noticed Git complaining about "multiple merge bases," and he described what he thought he had done to produce that complaint:

"I needed to have prerequisite work from another tip branch: tip/locking/core which was fast-forwarded to v5.17-rc1 before it got that prerequisite work added ontop.

"So I merged tip/locking/core into tip/ras/core […] and added the RAS stuff ontop.

"However, when creating the diffstat for the pull request, it would add additional files to it from tip/locking/core even if all the tip/locking/core changes are already in your master branch."

He didn't see exactly what he had done wrong, so he asked Linus for an explanation. Linus replied:

"What you are describing is a very fundamental thing – your branch has multiple separate starting points, since you

had different branches that you merged into your tree.

"Sometimes having multiple branches doesn't actually cause that, because the different branches may all have the same base starting point.

"Git calls these things 'merge bases', because those starting points [are] what you have to take into account when merging, they are the 'base' for actually resolving the differences that come in through multiple branches.

*"And git handles that perfectly fine when merging by doing all the appropriate magic. And 'git log' has no problem with it either – you can list all the commits that are in your head but are *not* in some arbitrary number of merge bases just fine.*

"But when you do a 'git diff', things are different (and 'git request-pull' basically just does a diff to show what the thing was about).

"A 'diff' is fundamentally something you do on two end-points. You have a beginning, you have an end, and you ask 'what changed between these two end-points'.

*"But that fundamentally means that when you have multiple different merge bases, and you ask 'what changed since the beginning and the current state', your question is fundamentally ambiguous. There is not a 'the beginning'. There are *multiple* beginnings.*

"So what git will do it to pick _one_ beginning, and just use that.

"And that means that yes, the diff will show the changes since that beginning, but since the end result depends on the _other_ beginning too, it will show the changes that came from that other beginning as well.

"Sometimes those changes end up being empty, because the 'first beginning' might already have had all of that. So sometimes you might not even notice that what 'git diff' gave you was ambiguous.

"So 'git request-pull' does both a log (for the shortlog of commits) and a diff (for the diffstat), and the log should always be correct, but the diffstat will have this ambiguity problem if you have multiple merge bases."

Linus went on:

"In the general case, you aren't doing anything wrong: if you merge multiple real branches, it's just that 'git diff' cannot find a single unique point to use as the base, and you'll get some odd random diff.

"But if you are a developer who merges multiple real branches, you obviously know how to merge things, and one way to sort it out is to basically do a test-merge just for yourself:

```
# WWLS? ("What would Linus See?")
git branch -b test-merge linus
git merge my-branch
git diff -C --stat --summary ORIG_HEAD..
.. save that away ..

# go back to your real work,
# and remove that test-merge
git checkout <normal-branch>
git branch -D test-merge
```

"will generate a diffstat of what a merge (which fundamentally knows how to resolve multiple merge bases) would generate.

"(Obviously you can just do the above in a completely separate git tree too, if you don't like doing those temporary branches that might mess up your working tree).

"The other alternative is to just send me the bogus diffstat – I'm sadly quite used to it, since a number of people just do 'git request-pull', see that it's odd, don't understand why, and just let me sort it out.

"Now the good news is that people who are afraid of merges and the above kind of complexity will never actually see this situation. You can't get multiple merge bases if you don't do any merges yourself.

"So this kind of git complexity only happens to people who are supposed to be able to handle it. You clearly figured out what was going on; you didn't perhaps just realize the full story."

Borislav thanked him for the explanation and offered to write up some documentation on the topic to include in the kernel tree. But Jonathan Corbet offered a link to his own effort in March 2022 to document this situation, at `Documentation/maintainer/messy-diffstat.rst` in the kernel tree.

Borislav said he liked Jonathan's version of the documentation a lot better than whatever he himself might have scratched together, and the discussion came to an end.

When Word Has Not Yet Gone Round

Andy Shevchenko recently inadvertently put his head in the lion's mouth, pointing out that since `-Werror` had been added to the kernel build systems, all warnings at build time were now errors. And for anyone using `W=1` at build time, this would cause the GNU C Compiler (GCC) to be very finicky about identifying as many warnings as possible. The result was that Andy's kernel builds were failing left and right. He posted a patch to remove the whole `-Werror` thing in the kernel build system.

The lion's jaws did not immediately snap closed around Andy's head. Instead, Masahiro Yamada said that `-Werror` should not be enabled for any regular user who is simply building the kernel for their own use.

Andy agreed with this. However, he did start to feel the lion's hot breath all around him and offered a link to an earlier discussion, at *https://lore.kernel.org/all/YjsCpoRK7W4l6tSh@zn.tnic/*, in which Linus Torvalds had said that kernel build warnings were unacceptable, and that everyone should enable `-Werror`.

At this point, the lion did indeed chomp down upon the head of Andy.

Linus said:

"If you enabled CONFIG_WERROR, then you get CONFIG_WERROR.

"If you enabled W = 1, then you get extra warnings.

"If you enabled both, then you get extra warnings and they are errors.

"This patch is just stupid."

He went on to say:

"WERROR should be on for regular builds.

"It's W = 1 that is questionable. It enables warnings that are often false positives, and if you use W = 1 (and particularly W = 2) then that's _your_ problem.

"W = 1 is most definitely not 'regular builds'. It's only for people who want to deal with crazy compiler warnings.

"I want WERROR on as widely as possible, because I'm really sick and tired of developers not noticing when they add warnings because they did a 'regular build'.

"Stop this idiocy where you think warnings are acceptable."

Andy replied, "fair enough." Then he picked up his masticated head from the ground, reattached it, and proceeded to the next thing. ■■■

Register by October 14 for Big Discounts!

You. Dallas. SC22.

NOVEMBER 13–18

One of the world's largest gatherings of HPC researchers and professionals, SC22 is an incredible week of learning new skills, forging new friendships, solving complex problems, and viewing next-gen technology.

A new, custom-built Digital Experience is available if you prefer to attend virtually.

The International Conference for High Performance Computing, Networking, Storage, and Analysis

SC22

Dallas, TX | hpc accelerates.

Register Today!

sc22.supercomputing.org

Sponsored by: IEEE COMPUTER SOCIETY | TCHPC | acm sighpc

Open RAN and the future of mobile networks

Open Air

Open RAN brings a new spirit of openness to the radio access networks that form the foundation for the mobile revolution.

By Emil J. Khatib

Mobile networks have been a major contributor to the technology and culture of the 21st Century. Companies such as Nokia or Ericsson shaped the early generations of the mobile revolution, orchestrated by industrial alliances such as GSMA, ETSI, or the 3GPP. These international organizations created open standards that enhanced competition and helped the growth of the sector. Fast forward to 2022, and the main actors have changed, with names such as Samsung, Apple, or Huawei shaping the smartphone market, but open standards have continued to play an important role.

Like many technological developments, the precedents of mobile communications go back to World War II. The concept of mobile telephones that could help soldiers on the field moved into the civilian sphere after the war, and the 1960s, '70s, and '80s saw the emergence of portable communication systems such as citizens band (CB) radio, walkie-talkies and, finally, early cellular communications.

In the early 1990s, the second generation (2G) of mobile telephony networks made an appearance with the CDMA standard (which was mainly used in the Americas, Japan, and South Korea) and the GSM standard (which was popular in Europe, Africa, Australia, and much of Asia). 2G saw the explosion of mobile phones and effectively transformed the way we view telecommunications: It was no longer about communicating between places but about connecting individuals.

Around this time, the Internet also became a commodity, and users who were accustomed to being connected wanted to browse the web and send email from their phones. Efforts to upgrade 2G soon fell short, and the third generation (3G) began to take shape. The 3rd Generation Partnership Project (3GPP) released the UMTS specification in 1994, which started its commercialization around the turn of the millennium. 3G supported mobile broadband, paving the way for smartphones a few years later.

The relentless appetite for higher bandwidths drove the 3GPP to provide almost yearly updates to the standard. UMTS was updated with Releases 99, 2000, 4, 5, 6, and 7. Although the numbering is odd, from that point on, there was a pretty stable numbering scheme. Releases 8 (2008) through 14 (2014) defined LTE and derivatives, which became known as 4G. LTE greatly simplified the network architecture, making it possible for smaller companies to operate and maintain a 4G network.

More recently, the fifth generation (5G), defined in 3GPP Releases 15 (2018) to 17 (2022), has continued to simplify the network. With the introduction of Network Function Virtualization (NFV) and Software Defined Radio (SDR) technologies, 5G has reduced the cost for fabricating network equipment, opening the market to smaller competitors.

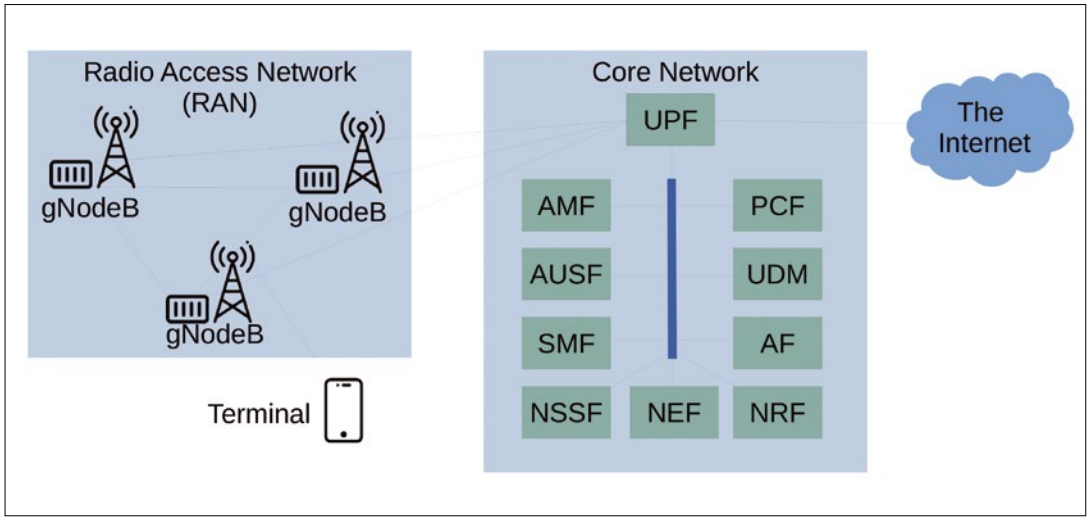


Figure 1: Main elements of a 5G mobile network.

Today the term "mobile network" usually refers to a 3GPP network, as opposed to IEEE WiFi or other types of

wireless networking. The 3GPP standard defines some elements that are common to all the generations of digital cellular networks (from GSM onward). A 5G mobile network consists of two large blocks that each contain other functions (Figure 1): the Radio Access Network (RAN) and the Core Network (Core).

The RAN contains all the elements that make a mobile network *mobile*, namely, the base stations (Figure 2) that act as access points for the terminals. This architecture has advanced significantly with the consecutive generations, and the elements of a base station have been renamed, reshaped, and reduced in size several times. Since 4G, another part of the architecture is femtocells, which are small form factor base stations with a size and shape similar to a WiFi access point. Femtocells (Figure 3) improve coverage for small offices, indoor public spaces, and homes. In 5G (and, by the way, also in 6G), base stations are better known as gNodeBs.

Figure 2: Antenna of a typical base station.

The RAN is basically a network of access points that provides connectivity below the IP layer and cannot on its own connect to the external world or provide services such as authentication or roaming. Instead, these services are functions of the core network. In other words, the core network contains all the elements that make a mobile network *a network*. (See the box entitled "Core Network" for more on the key core network functions.)

As new generations of mobile networks have advanced, these elements of the core network have become more and more open. In the 5G environment, core network functions can be programmed and distributed by different vendors, as long as they follow the 3GPP standard interfaces. This open approach has many benefits. For instance, the NRF function makes it easier to manage packages from different vendors, eliminating incompatibilities that could otherwise render a network inoperable.

This trend in opening the standards has also extended to the RAN, where a similar microservice-based architecture is emerging: Open RAN.

Open RAN

The microservice-based approach used in the 5G Core Network turned out to simplify the addition and maintenance of new functions, the integration of third-party software, and the usage of off-the-shelf computers for running the infrastructure. In other words, it was a good idea. So the next logical step was to move this good idea to the other part of the network, the RAN. In 5G, the RAN also had open specifications to a good degree (e.g., network functions, protocols, etc.). The only roadblock was that only a limited number of manufacturers had the fabrication capabilities for building the required equipment: high-frequency radio transceivers and amplifiers, carefully designed antennas, highly selective filters, and other highly specialized tools. Traditionally, this limit has reduced the number of actors in the RAN market to a few large enterprises, such as Huawei, Nokia, and Ericsson.

Figure 3: 5G femtocell (CC BY-SA 4.0, user Liumjs from commons.wikimedia.org).

Nevertheless, the conjunction of several technological advances in recent years has changed the outlook for the RAN. First and

Core Network

In 5G, the main core network functions are:

- User Plane Function (UPF): contains all the functions that are directly related with connecting the terminal with the Internet. Among these functions are routing and forwarding, Network Address Translation (NAT), packet filtering, Deep Packet Inspection (for lawful communication interception; yes, this function is defined in the standard [1]) and, of course, billing. These functions compose the user plane. All the other functions of a 5G network are called control plane functions and are needed to support the correct operation of the user plane.
- Access and Mobility Management Function (AMF): performs the roaming functionality, that is, the required operations for reassigning serving gNodeBs to the terminal and doing the handover such that the user plane can always provide connectivity to the terminal.
- Authentication Server Function (AUSF): contains the user authentication features for other core functions.
- Session Management Function (SMF): keeps track of the connections in the user plane and ensures their correct operation according to user policies for the specific terminal.
- Network Slice Selection Function (NSSF): supports the network slicing [2] functionality, which divides resources of the network among several slices or sub-networks. Each slice can support different configurations, services, etc., virtually, while sharing bare metal resources with other slices.
- Policy Control Function (PCF): contains the rules that regulate other control plane functions (such as the resource assignment to different slices) and the connections in the user plane (e.g., different quality of service (QoS) parameters).
- Unified Data Management (UDM): Acts as a centralized database containing the user data, such as session status and authentication keys. It is used by other functions such as AUSF and SMF.
- Application Function (AF): Provides services to the end user. The AF is the equivalent of a remote service over the Internet but with an increased integration into the mobile network (e.g., for using the user profile or faster user plane connections).
- Network Repository Function (NRF): acts as a centralized repository for the software packages that implement all the core network functions.
- Network Exposure Function (NEF): acts as an interface for developing new services in the 5G Core – contains SDKs, APIs for third-party AFs, and other elements.

All these network functions are implemented by software packages running over a powerful computer (which normally runs some enterprise-grade Linux distribution such as RHEL or SLE).

foremost, Software-Defined Radio (SDR) allows the usage of generic radio equipment, which can be fabricated by third parties and acquired by RAN equipment integrators. There is no need to build custom RAN-specific radio chips, which are costly and only available to a few manufacturers. An SDR board with a powerful computer and the appropriate software for the signal processing can become a base station for a RAN. Secondly, the development of cloud computing has made it possible for integrators to use cloud providers to support all the signal processing. The need for "powerful computers" then is replaced for a much more economical cloud instance where resources are provided on demand. These two technologies on their own already point at drastic cost reductions, bringing smaller competitors into play, including small enterprise companies such as Amarisoft, Yepzon, and Viavi.

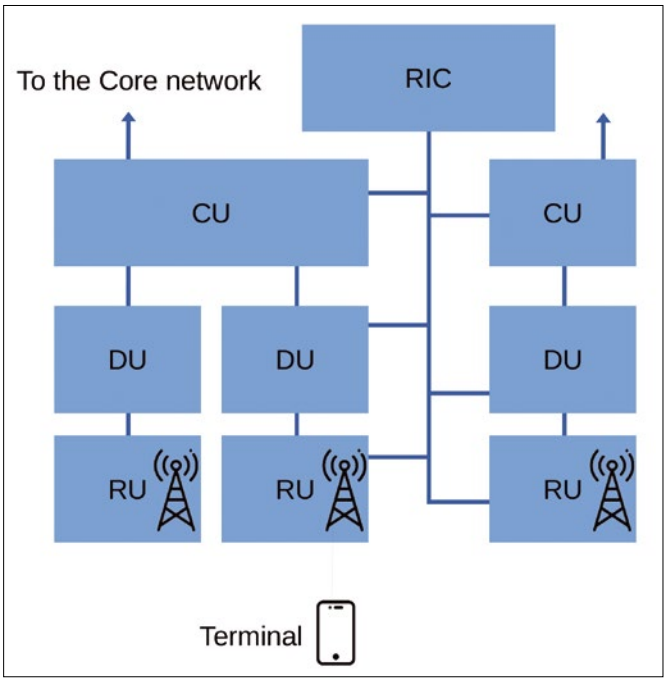


Figure 4: Components of Open RAN.

But an even higher degree of openness is possible. Although these small enterprises could each have their closed, monolithic solution, there is another way: Open RAN is a new approach that redefines the RAN as a set of microservices with an open specification of interfaces. The Open RAN specification defines functional blocks, along with the interfaces that connect them, such that a RAN integrator can make a RAN solution with off-the-shelf hardware (or cloud instances), some SDR modules, and a set of software components from different manufacturers that perfectly interact with each other. These components can be updated as vendors add new features, correct errors, or support future 3GPP releases.

Figure 4 shows the elements defined in Open RAN. The main concept is "disaggregation." Instead of having a discrete gNodeB, OpenRAN divides the functions among three main components:

- The Radio Unit (RU): Contains all the functionalities of the gNodeB that deal with the radio signal, such as modulation and demodulation, filtering, etc. In technical terms, the RU implements the lower physical layer. You can find a definition of the physical layer in the "Reference Model" box. It is the part that normally will run within an SDR device.
- The Distributed Unit (DU): Contains the rest of the lower layers, such as the baseband processing (see the "Radio Aspects" box) of the higher physical layer, the MAC layer, and the RLC layer. The DU communicates with the RU to send

Reference Model

Figure 5 shows the reference model of the 5G network. The three main elements are the terminal, the gNodeB, and the core network. The terminal can be a smartphone, an IoT system, a laptop, or any similar device. The gNodeB is the equivalent of a switch in Ethernet, acting below the IP layer. The core network is the equivalent of the ISP network. From bottom to top, the layers of the reference model are:

- Physical layer (PHY): deals with all the radio aspects of the communications, such as modulation and demodulation. See the "Radio Aspects" box for a definition of these terms.
- Medium Access Control (MAC): defines logical channels over the PHY channels (which are defined by time and frequency). Also performs error correction.
- Radio Link Control (RLC): provides a data transfer service to upper layers over the MAC, with error recovery, segmentation and reassembly, and reestablishment of radio link when needed.
- Packet Data Convergence Protocol (PDCP): provides an interface to higher layers with one or several RLC entities. This allows multi-connectivity, the ability of connecting to more than one gNodeB at RLC layer. It also provides some throughput gains with header compression.
- Service Data Adaption Protocol (SDAP): provides mechanisms to ensure certain quality of service (QoS) to different data flows. For instance, for video streaming, a steady data flow with a relatively high bandwidth is required, while for web browsing, the QoS requirements can be relaxed. This sublayer is present in the user plane, the part of the protocol stack that is used to transmit user data.
- Radio Resource Control (RRC): this is a control plane sublayer, so the functions it contains do not deal with user data, but with the metadata required to keep the service working correctly. In this case, it contains all the required protocols to keep the radio connection open and stable. It deals with handovers (changes of serving gNodeB), changes to more robust or capable modulations, etc.
- PDU layer: only present in the user plane and only in the terminal and the core network. It contains the IP functionality and connects the terminal to the UPF, which acts as a router.
- Non Access Stratum (NAS): only present in the control plane and only in the terminal and core network. It contains the control functionality, such as the establishment of connection between the terminal and core in the user plane, authentication, and other services.
- Application: in the user plane, the Application layer contains the higher layers of the services that operate between processes in the terminal and the remote servers.

and receive data over the air, and it is called "distributed" because it is normally deployed physically near the RUs to reduce latency.

- The Central Unit (CU): Contains the higher layers, that is, functions such as routing the user and control data (PDCP layer) and assigning network resources such as channels and carriers (RRC layer). It is usually in a centralized location (hence the name) and can even be physically running in the same computer as the core.

Another element shown in Figure 4 is the RAN Intelligent Controller (RIC), which contains functions that control the overall operation of the RAN elements. The RIC includes functions such as optimization of resources, monitoring, etc., and even end user services that must be very close to the terminals. The RIC is further divided into Real Time RIC, for some lower-layer functions that require quick reaction times, and Non-Real Time RIC, for functions with longer timeframes In 6G, this element will contain ML-based solutions that continuously monitor the network, learn by reinforcement, and improve the quality of service.

Except the RU, all other components are software components that can run in any computer with the minimum CPU, memory, and storage requirements. This support for commodity hardware means a drastic cost reduction for RAN manufacturers and operators.

The RAN can then be thought of as a network containing cloud instances and off-the-shelf computers, some of them with SDR devices. Some of this infrastructure is centralized, in a

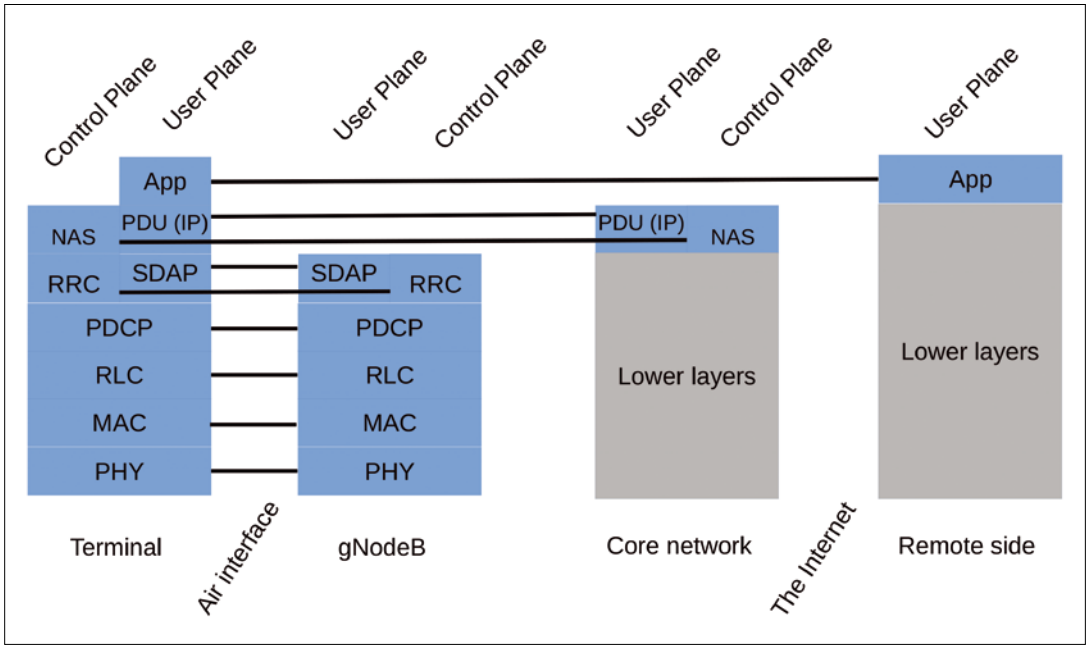


Figure 5: Layers of the 5G protocol. Only the interfaces with the terminal are described.

third-party cloud (such as AWS or Azure) or in a data center of the operator. And some of it may be distributed all over the coverage area of the operator, near where the RUs and users are located, in what is called the "network edge." Thanks to virtualization, all of this infrastructure is available to all the functions of the RAN. Virtual functions can be moved flexibly between the central infrastructure and the network edge to reduce latency or to harmonize the usage of computing resources.

Undoubtedly, as was the case with the core network, Open Source is bound to play a major role in future networks. Open interfaces open the RAN market for small enterprises or even the Open Source community, and thanks to the ease of just using off-the-shelf hardware that is easily accessible, anyone with the required knowledge, some documentation, and, of course, lots of time can start developing functions. This work would also not need to be started from scratch, as you will see in the next section.

Radio Aspects

Probably the first thing that comes to mind when talking about a mobile network is the radio technology that makes it possible. These functions are all contained in the PHY layer and are some of the most specialized parts of the network that require their own hardware. For digital radio transmission, we start with a baseband signal (Figure 6), which is made up of one or several electric signals that represent digital bits. The spectrum of this signal (its Fourier transform) is centered around 0 Hz and has a limited bandwidth. You can then modulate this signal, i.e., transform it such that its central frequency is at a specific given frequency. You can also modulate several different streams of bits in adjacent frequencies, which is called Frequency Division Multiple Access (FDMA). Each of these channels (also known as subcarriers) is dedicated to different users or data streams in different times (Transmission Time Intervals or TTI), a scheme that is called Time Division Multiple Access (TDMA). In mobile networks, Orthogonal FDMA (OFDMA, an improved version of basic FDMA) groups a set of several subcarriers into a carrier. The OFDMA subcarriers and TTIs define what is called the resource grid (Figure 6), in which each element has a defined logical function.

Open RAN Implementations

The O-RAN Alliance [3] was created to standardize the interfaces of an Open RAN implementation. The alliance is a consortium made up of large corporations, small and medium enterprises, universities, and research centers. Among them are open source vendors such as Red Hat and SUSE. The O-RAN Alliance regularly releases open specifications that anyone can access. These specifications define how the different elements of an Open RAN (as shown in Figure 4) should behave, including the protocols used in the interfaces and other elements. The alliance has also set up a partnership with the Linux Foundation, called the O-RAN Alliance Software Community, to create an open source software implementation of these specifications, which anyone will be able to downloaded and test, and to which any developer can contribute. The alliance is still in an early development stage. Some elements are already working, and some proof-of-concepts are at a showcase stage. Some other elements, such as the RU, are still missing.

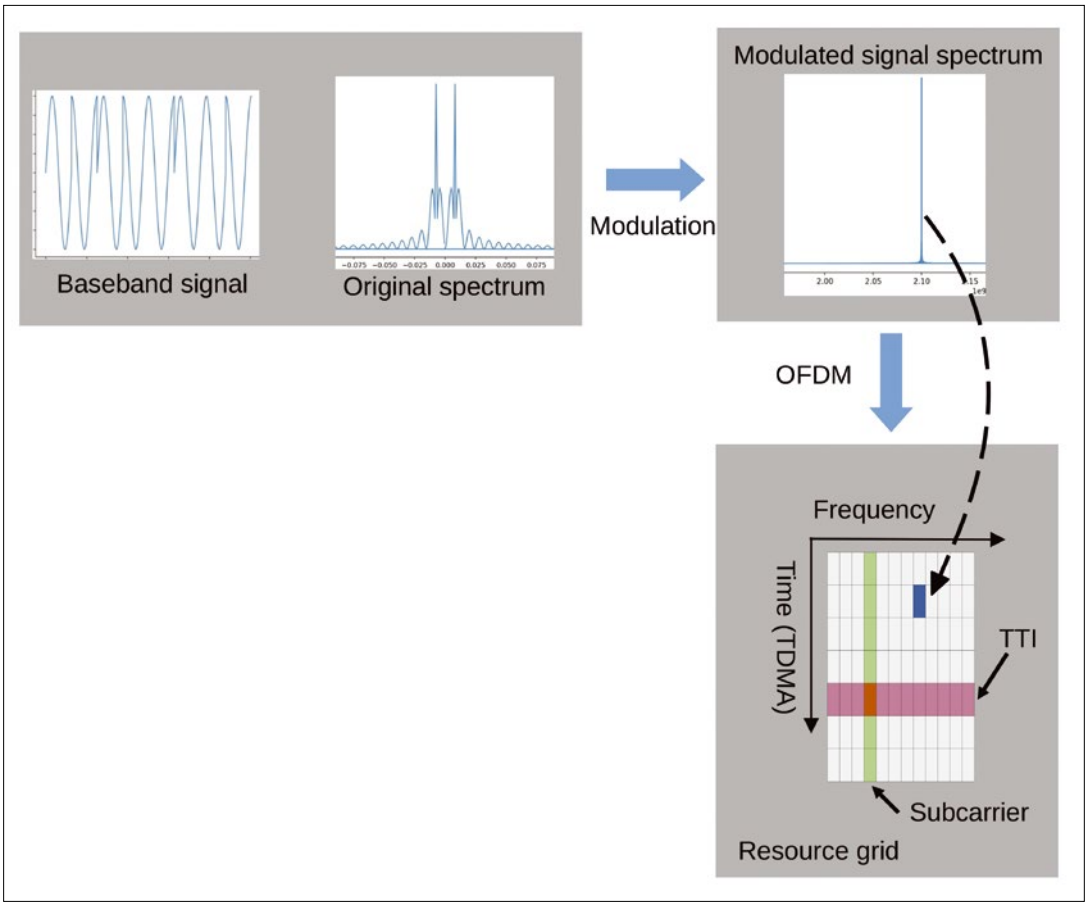


Figure 6: Process of modulation/demodulation and the resource grid resulting from TDMA and OFDMA in 5G.

The Telecom Infra Project (TIP) [4] is another partnership of large enterprises of the sector that initially was created for defining another Open RAN specification, but it soon adopted the design of the O-RAN Alliance and promoted its own software implementation: OpenRAN (note that in this case there is no space between Open and RAN). TIP works actively in creating implementations of Open RAN elements, including some radio interface designs (such as a full WiFi stack), and also

other mobile network elements, like an Open Core network. TIP also holds "plugfests," where different Open RAN vendors test the integration and interoperativity of their functions. Although these are the most known Open RAN projects, there are some other implementations, such as OpenAirInterface [5], which provides an Open Source, O-RAN Alliance compliant solution.

There is some criticism [5] on the Open RAN concept for not having achieved the promise of a fully Open Source RAN, with a focus on the difficulty of developing the radio infrastructure. In fact, there are several roadblocks that make development of SDR functions for mobile networks very difficult or outright impossible (at the moment) for small enterprises or community developers. On the one hand, the required equipment, which is not only the SDR boards, but also testing equipment to verify functionality and compliance of standards and regulations, is very expensive. On the other hand, regulations are also very restrictive: The equipment that is used in mobile networks needs to be certified by the appropriate authorities, and a special license granted by public authorities is required to transmit radio signals in most frequencies (including those where most of cellular communications take place). In the long term, there are solutions for some of these problems. For instance, expensive measuring equipment might be replaced in the future by open source SDR-based implementations. Regulatory roadblocks, on the other hand, are harder to overcome. Currently, small enterprises and research centers access the spectrum usage license through partnerships with operators, which greatly limits the development, and expensive tools such as anechoic chambers (basically Faraday chambers that block outgoing high frequency radio signals). In any case, the creativity and ingenuity of the open source community has time and again proven that it can overcome such problems.

The Role of Open Hardware

One of the main selling points of NFV in the network core and the Open RAN are the possibilities for running most of the network functions in off-the-shelf computers or common cloud instances. Regarding the underlying hardware, the first name that comes to mind to the layman when talking about "off-the-shelf" computers are x86-based systems. These systems are easy to acquire and easy to replace, and once their lifetime in the network ends, they can be reused for other purposes. Their operation and maintenance is low, documentation and expertise abounds, and they have a very high level of support from manufacturers and tools. On the downside, x86 systems are not very energy efficient, which is an increasingly important concern for operators, both in terms of carbon footprint and energy bills.

More recently, energy-efficient ARM processors are starting to be used in servers. Although ARM currently doesn't have the level of support or the quantity of software available for the x86, interest in ARM is growing by the day.

Finally, no processor discussion would be complete without mentioning RISC-V. The RISC-V platform has the advantages of ARM in power consumption (although it falls behind in terms of maturity), and it is open source hardware. Potentially, small enterprises could extend RISC-V processors to include specific functions for the mobile network. Also, thanks to its openness, it is expected that once the demand for RISC-V increases, the prices will drop.

Up to this point, I have discussed the hardware for supporting the software part of the mobile network, that is, the core network and the DU, CU, and RIC in the Open RAN. What about the RU? Although on the paper it looks just like a small part, the RU is fundamental for radio communications, and it is one of the hardest parts to work on. However, there are several options, such as the USRP devices from Ettus Research [6], which offer open source drivers. USRPs are a favorite of researchers in the field due to the large community that has gathered around them. Another commercial option is the LimeSDR [7] boards, which are open source hardware. They are more cost effective than USRPs and are starting to be adopted by the community. They are also used in several commercial RAN solutions developed by integrators such as Amarisoft. The HackRF [8] and PlutoSDR [9] devices, which are also open source, are very cheap alternatives that can be acquired by community developers. Although these devices are not as powerful as LimeSDR or USRPs, they are good starting points for learning and exploring Open RAN functions. You can also use HackRF and PlutoSDR to start developing much needed auxiliary tools, such as spectrum analyzers.

Future Perspective

It is fair to say that Open RAN has not yet achieved the dominance on cellular networks that Linux and Apache have over the Internet. But all the elements are there, and it just takes some work from motivated individuals, universities, and underdog companies who see open source as an ally to compete with the large manufacturers. In the end, it is very likely that open standards will lead the drive toward open hardware and open source software on future mobile networks. ■■■

Info

[1] 3GPP TS 33.127: Lawful Interception (LI) Architecture and Functions: *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3182*

[2] GSM Association Official Document NG.127 – E2E Network Slicing Architecture: *https://www.gsma.com/newsroom/wp-content/uploads/NG.127-v1.0-2.pdf*

[3] O-RAN Alliance homepage: *https://www.o-ran.org*

[4] TIP homepage: *https://telecominfraproject.com*

[5] Matt Kapko, "Where Are You, Open Source RAN?," available at: *https://www.sdxcentral.com/articles/analysis/where-are-you-open-source-ran/2020/12/*

[6] USRP Hardware Driver and Manual: *https://files.ettus.com/manual/page_uhd.html*

[7] LimeSDR-USB User Guide: *https://wiki.myriadrf.org/LimeSDR-USB_User_Guide*

[8] HackRF: *https://hackrf.readthedocs.io/_/downloads/en/latest/pdf/*

[9] ADALM-PLUTO for End Users: *https://wiki.analog.com/university/tools/pluto/users*

Author

Dr. Emil J. Khatib is a researcher at the University of Málaga in the field of cellular networks and industrial IoT. He also loves programming hardware and web and mobile apps. *www.emilkhatib.com*

MEPIS and antiX come together

MX Linux

MX Linux is fast, friendly, and focused on function. *By Bruce Byfield*

MX Linux is a collaboration between MEPIS and antiX, whose initials form its name [1]. AntiX is a popular lightweight distribution, while MEPIS once occupied a role similar to that of Ubuntu or Linux Mint today, in that it added usability to Debian on the desktop. Although a relative newcomer, MX Linux stands out among the hundreds of other distributions by combining the best of its ancestors, offering the speed of antiX and the user-friendliness of MEPIS's desktop tools.

MX Linux's flagship version features Xfce, with variations for older computers as well as 32- and 64-bit machines. Thirty-two and 64-bit versions are also available for KDE Plasma and Fluxbox, offering users midweight, heavyweight, and lightweight versions, plus those designed for admins and for Raspberry Pi. The installer goes through the usual steps, but it is noteworthy for the detailed help embedded in the window, which not only explains the current choices but also how to use the interface when necessary – a thoroughness unusual in installers (Figure 1). Similarly, MX Linux first boots into a Welcome screen that draws attention to the major system tools and also offers links to a tour, videos, forums, an FAQ, and the user manual, all of which provide useful starting points for exploring the desktop environment (Figure 2). For more advanced users, the *About* tab offers a summary of both system hardware partitions, boot mode, and active repositories.

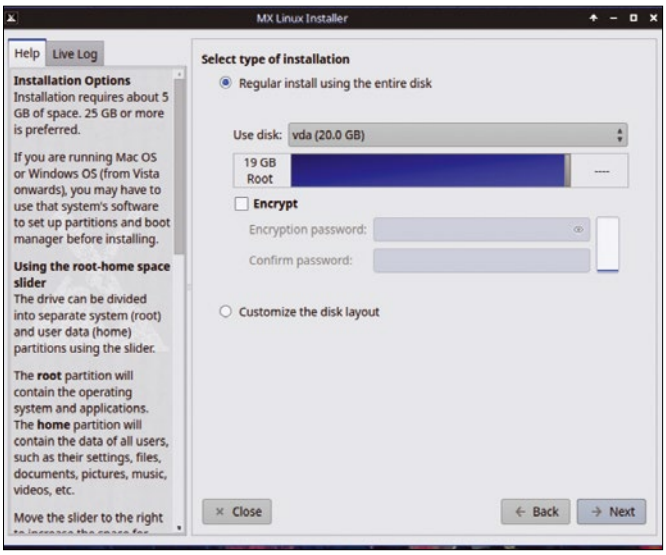


Figure 1: The installer features detailed embedded help.

Desktop and Structure

Booting into the Xfce edition, I was surprised to see an impressionistic wallpaper suitable to Wildflower (the

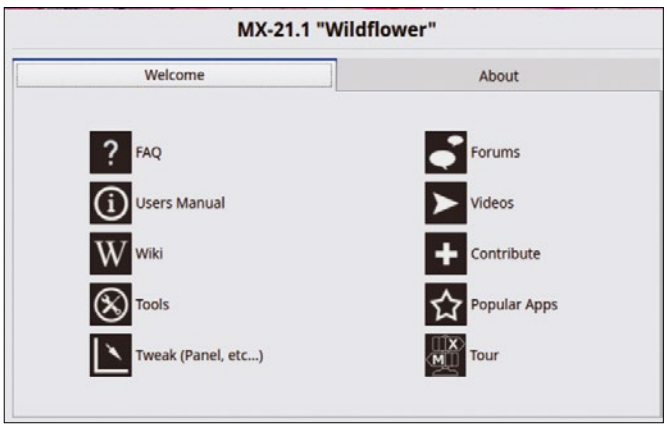


Figure 2: The Welcome screen provides a variety of orientations and help sources.

Photo by Markus Spiske on Unsplash

Figure 3: The default MX Linux Wildflower desktop.

name of the latest release) and not the abstract patterns that have become the norm in many distributions (Figure 3). Otherwise, the MX Linux desktop as a whole seems less polished than most, without the unity of design that has become common on other distributions during the past decade. Utility windows, for example, are not consistently the same size, nor do all the icons seem designed as sets. Instead, aesthetics seem a distant second to functionality – perhaps not surprising in a relatively recent distribution.

The emphasis on function is evident in the design choices. Taking a cue from Ubuntu's old Unity desktop, the panel is positioned by default on the left of the screen, maximizing the screen space for windows. The panel's contents seem meant to be system tools, although apps can be dragged to it (or the desktop) from the menu. If desired, users can make the panel into a dock. In other words, the only unusual thing about the panel is that, except for the default position, there is nothing unusual about the panel. It's like that seen in most distributions, and for that reason, it is easy to learn. The same can be said for the Whisker menu (although for some reason the developers saw fit to give it a name).

Behind the scenes, MX Linux is a relatively secure distribution, with many system utilities accessible only by the root users and prominent warnings of the possible consequences of possible actions. It depends on the Debian stable and stable-updates repositories, making it a reliable but far from cutting-edge distribution, though its own developed packages are stored in the `mx.list` repository. In fact, the manual specifically warns against trying to install Ubuntu's experimental Personal Package Archives (PPAs) or leaving other repositories permanently enabled, rightly warning that doing so "will likely result in system instabilities and lead to a re-installation." But perhaps the most unusual aspect of MX Linux is its neutral position on systemd. By default, MX Linux uses the more conservative SysVinit. However, systemd is also installed and can be activated from the Advanced options in the boot manager (Figure 4). Once systemd is installed, SysVinit can be removed with the command `systemd-sys`. The Snapshot and Live USB persistence features will not work with systemd, but the rest of MX Linux should.

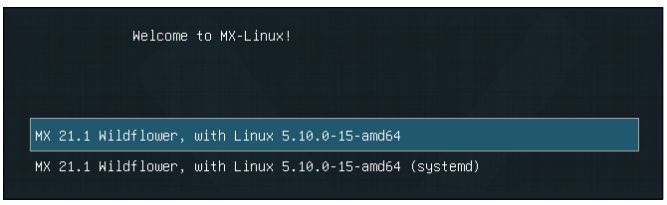


Figure 4: Systemd can be enabled from the boot manager.

Desktop Tools

The emphasis on functionality is most apparent in the array of desktop tools. Together, they make MX Linux one of the distributions that is easiest to administer from the desktop without resorting to the command line in the most common cases. Versions of some of these tools (such as Conky or Tweak) are common in many distributions.

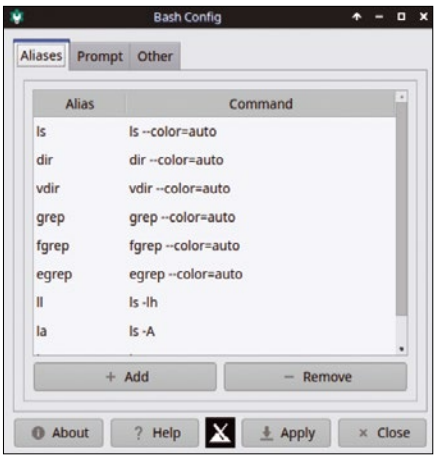


Figure 5: Bash Config provides a GUI for several command-line actions.

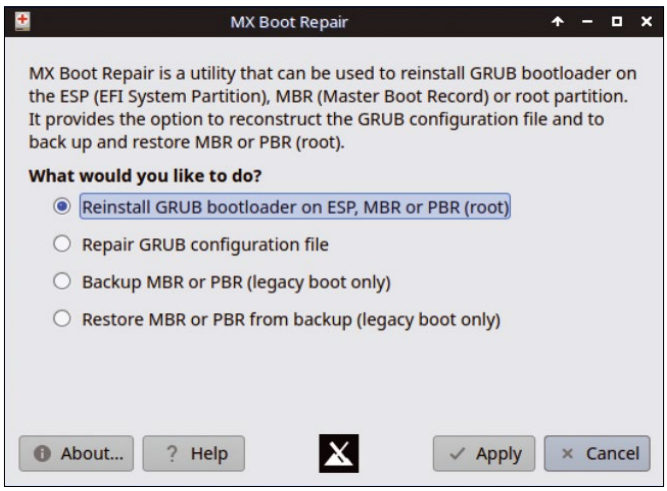


Figure 6: Boot Repair helps to restore the boot manager.

Others, like those for managing users or sound settings, are basic necessities, but many are imported from antiX or developed specifically for MX Linux.

For easier setups, MX Linux features GUIs for NVIDIA drivers and codec installers, as well as a Bash config, which has tabs for setting aliases, prompt structure, and history, three tasks which must ordinarily be done separately from the command line (Figure 5). An extensive set of unique tools are also available for maintenance, such as Chroot Rescue Scan, Boot Options, and Boot Repair (Figure 6). However, of all MX Linux's tools, probably the most useful are the ones for external devices: the Live USB Maker, which includes the ability to update the kernel on a Linux system on a flash drive (Figure 7), and Snapshot, which saves system backups for easy restoration (Figure 8). Some of these tools are perhaps useful only in the most common of cases, but all the same, each of them empowers users who might hesitate to open a command line.

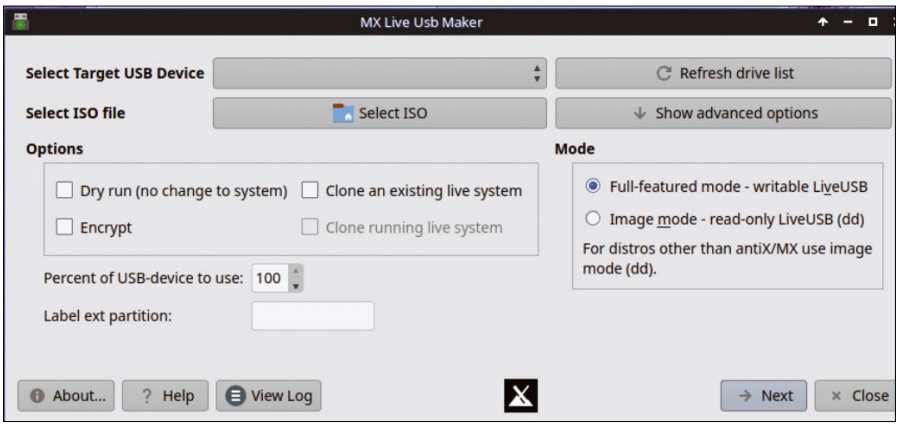


Figure 7: Live USB Maker produces and updates disk images on flash drives.

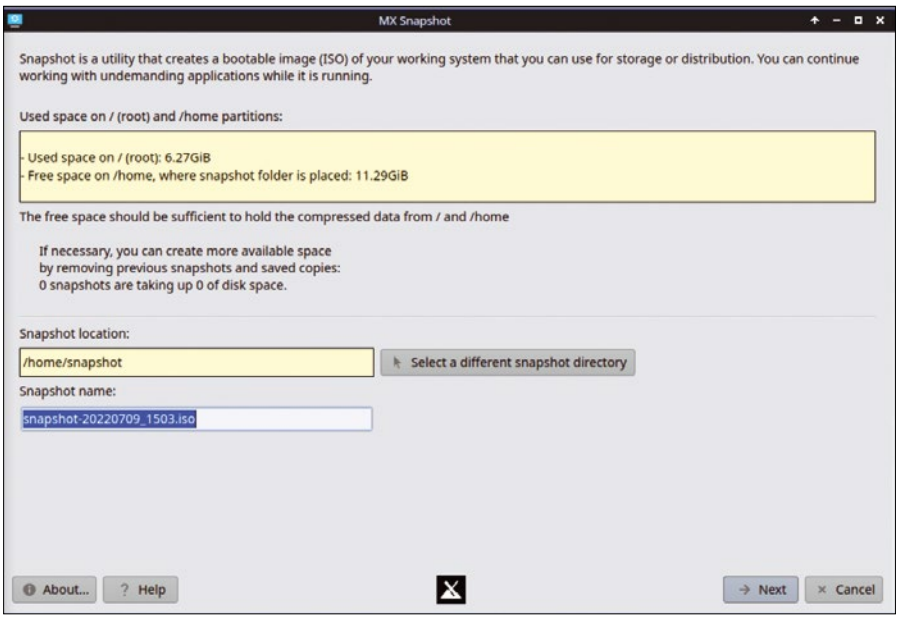


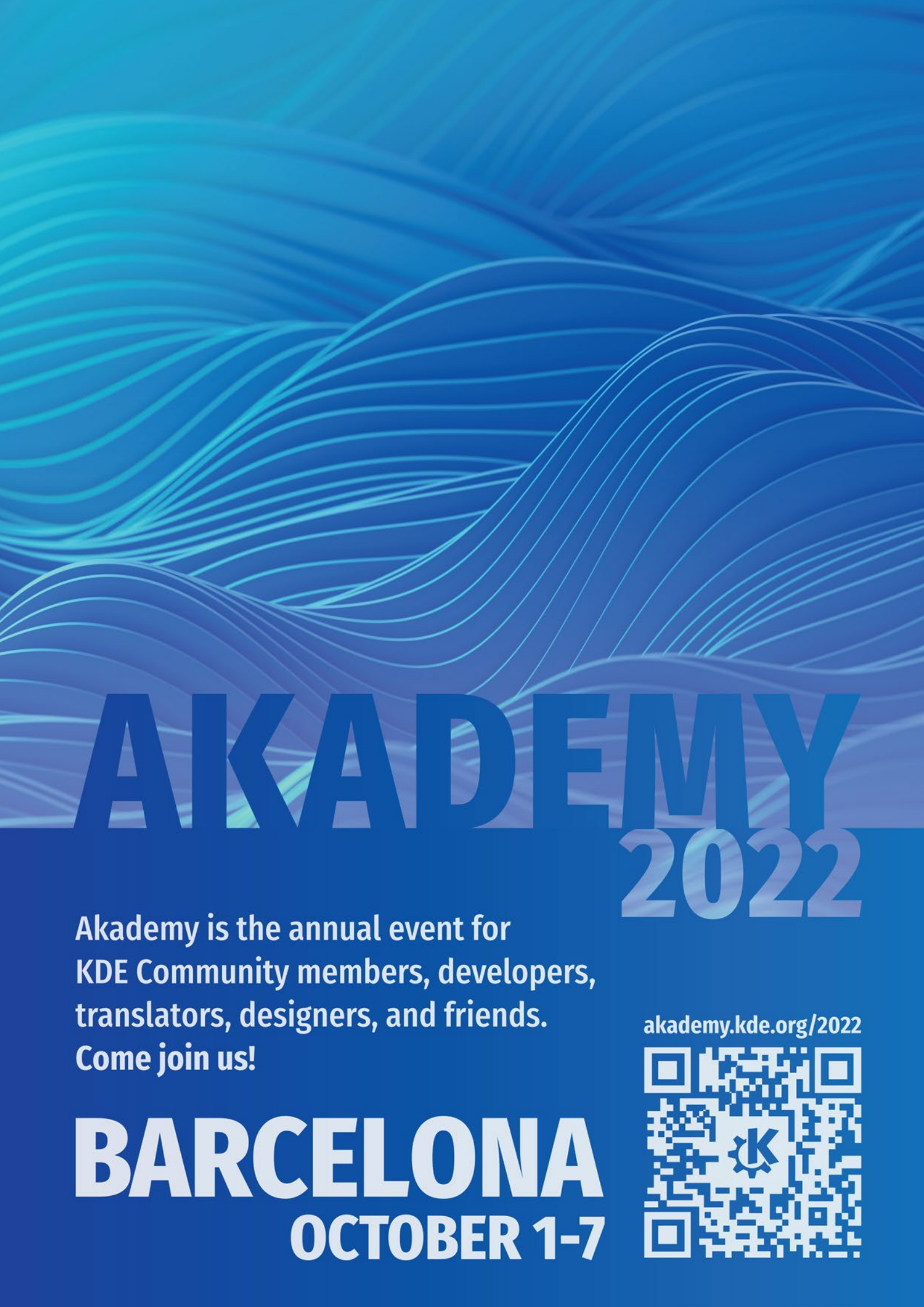
Figure 8: Inspired by a similar tool on Windows, Snapshot backs up essential system files for easy restoration.

A Work in Progress

Although the latest release is MX-21.1 – presumably, a continuation of antiX's version numbering – MX Linux strikes me as a work in progress. Its original utilities seem long overdue and many deserve to be ported to other distributions, but I would like to see what will be added next. Similarly, although function is the priority – a point obviously agreed upon by many users – MX Linux has rough edges that remind me of old versions of MEPIS – and indeed of all distributions prior to about 2005. Still, to innovate so much in its utilities while retaining much of the speed of antiX, MX Linux is definitely a distro to watch, and its popularity is unlikely to diminish any time soon. ■■■

Info

[1] MX Linux: *https://mxlinux.org/*


AKADEMY
2022
Akademy is the annual event for
KDE Community members, developers,
translators, designers, and friends.
Come join us!
BARCELONA
OCTOBER 1-7
akademy.kde.org/2022

Useful innovations in Ubuntu 22.04 LTS

The Long Haul

Ubuntu 22.04 LTS features an updated Linux kernel, numerous programming language updates, and improved virtualization and container tools, making it useful for developers and admins. *By Kristian Kißling*

Calling Ubuntu 22.04 LTS a COVID-19 release would be bad public relations, but it's not completely untrue because its predecessor 20.04 was released more or less at the onset of the pandemic. For companies using Ubuntu Desktop, Ubuntu Server, Ubuntu Cloud, and Ubuntu Core, the upgrade to "Jammy Jellyfish" (Figure 1) is well worthwhile, but there is no rush. Officially, the preceeding Ubuntu 20.04 LTS will still be supported until April 2025, with Extended Security Maintenance (ESM) for five additional years, assuming that you make an appropriate donation to Canonical.

However, users of other Ubuntu flavors, such as Kubuntu, Lubuntu, Xubuntu, and the like, can only count on official support until April 2023. Without ESM, admins will need to assess the consequences of the upgrade and compatibility issues at a somewhat less leisurely pace. If you switch to Ubuntu 22.04, the support period is extended to 2027 (or 2025 for the other flavors).

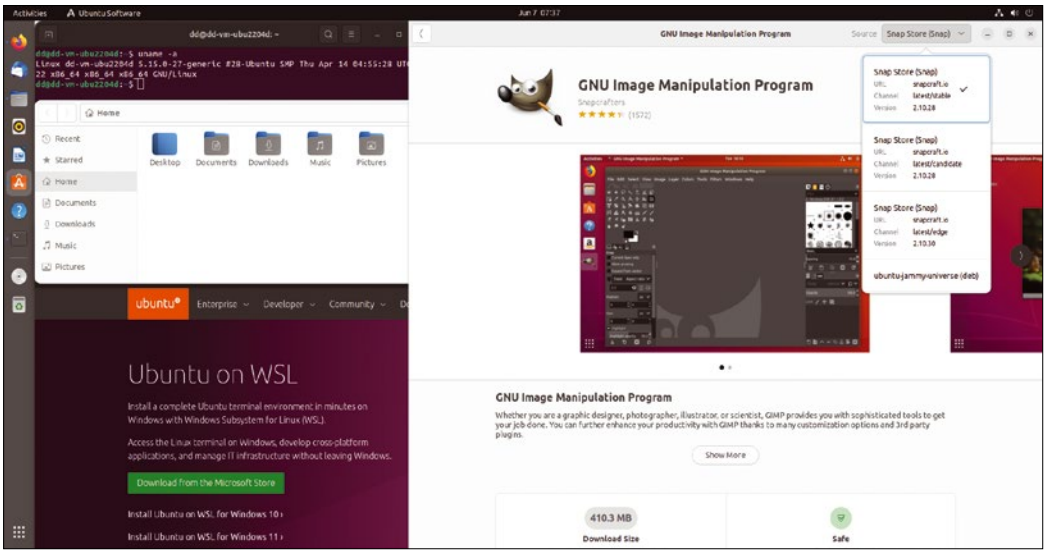


Figure 1: Ubuntu on the desktop: The software store offers both Debian and Snap packages.

Kernel Support

By default, Ubuntu 20.04 used Linux kernel version 5.4.0, while Ubuntu 22.04 has kernel version 5.15 (*linux-generic*). Canonical even uses kernel 5.17 (*linux-oem-22.04*) on certified devices. If you want, you can also use the rolling Hardware Enablement (HWE) kernel [1] (*linux-hwe-22.04*) with the LTS versions, whichs updates the distribution with the regular point releases and kernel versions.

According to Kernel.org [2], Linux kernel 5.15 will receive support for longer than other versions – specifically, until October 2023 (Figure 2). Presumably, the Ubuntu developers hope that another kernel with long-term support will have arrived on the scene by then. Otherwise, they will have to continue maintaining the kernel themselves after its shelf life expires [3].

WireGuard was already backported by the developers in Ubuntu 20.04, but there are many other innovations in kernel 5.15. For example, kernel 5.15 includes a new NTFS driver, support for

Photo by Robert Murray on Unsplash

Stable

After each mainline kernel is released, it is considered "stable." Any bug fixes for a stable kernel are backported from the mainline tree and applied by a designated stable kernel maintainer. There are usually only a few bugfix kernel releases until next mainline kernel becomes available -- unless it is designated a "longterm maintenance kernel." Stable kernel updates are released on as-needed basis, usually once a week.

Longterm

There are usually several "longterm maintenance" kernel releases provided for the purposes of backporting bugfixes for older kernel trees. Only important bugfixes are applied to such kernels and they don't usually see very frequent releases, especially for older trees.

Longterm release kernels

Version	Maintainer	Released	Projected EOL
5.15	Greg Kroah-Hartman & Sasha Levin	2021-10-31	Oct, 2023
5.10	Greg Kroah-Hartman & Sasha Levin	2020-12-13	Dec, 2026
5.4	Greg Kroah-Hartman & Sasha Levin	2019-11-24	Dec, 2025
4.19	Greg Kroah-Hartman & Sasha Levin	2018-10-22	Dec, 2024
4.14	Greg Kroah-Hartman & Sasha Levin	2017-11-12	Jan, 2024
4.9	Greg Kroah-Hartman & Sasha Levin	2016-12-11	Jan, 2023

Figure 2: The supplied kernel 5.15 will receive long-term support until 2023.

Apple's M1 chip, and a kernel-integrated Samba server, dubbed KSMBD. In addition to these major updates, there are several smaller tweaks to kernel security features. The eBPF kernel sandbox has been updated. There are some new system calls that simplify the container handling, among other things, as well as improvements to the collection of filesystems. For example, ext4, Ubuntu's standard filesystem, has been faster since kernel 5.10 thanks to a fast commit feature.

Network Binds

The server and client packages for Network File System (NFS) have been upgraded to the latest versions. NFS no longer supports mounting over UDP by default. The reason for the change is that NFS over UDP can cause data corruption on modern networks with connection speeds of more than 1Gbps – this is due to fragmentation brought about by the heavy load [4]. The new Samba v4.15.5 is also on board and, among other things, ends the experimental status of multichannel support.

SSH remains wildly popular for connecting to Ubuntu machines on the network – either as an admin or for software that then handles tasks on the target machines. OpenSSH 8.9, which is included with the new Ubuntu, disables RSA signatures by default because they use the insecure SHA-1. Disabling RSA may cause problems when communicating with older SSH servers, but that can be changed later [5]. The SCP software that comes with SSH moves and copies files between machines. The software now offers a `-s` option to use SFTP mode instead of SCP mode. For security reasons, according to the OpenSSH project, this behavior will become the default in the near future. OpenSSL v3 is also now available; it removes some legacy, insecure algorithms. Certificates that still support SHA-1 or MD5 also no longer work with OpenSSL v3.

The recently supported OpenLDAP 2.5.x is missing a few pieces, including the shell and BDB and HDB back ends. Bind v9.18, on the other hand, is now more secure, offering support for DNS over TLS (DoT) and DNS over HTTPS (DoH). The named service supports inbound and outbound zone transfers over TLS (XFR over TLS, XoT).

In terms of security, nftables now is the new back end that manages the firewall rules, taking over the job from iptables, as well as from ip6tables (IPv6), arptables (ARP), and ebtables (Ethernet bridging). The nftables developers are the same people who created iptables, and they are looking to dump the legacy ballast in the new software. The two iptables versions (for IPv4 and IPv6 addresses) still cause confusion and have forced admins to manage them in parallel, until now.

Machine Farms

Data center admins want to squeeze as many machines as possible onto a single lump of physical hardware for cost and efficiency reasons. This is where virtual machines (VMs) and containers come into play. In terms of the architecture, the Qemu virtualization software has recently outsourced the most frequently used features as modules. The new fuse3 version in Qemu 6.2.0 makes it easier to edit VM images without having root privileges and without having to boot the VM. In addition, Qemu now supports the Linux JACK sound server, which supports access with the particularly low latencies that musicians require.

Version 8.0.0 of the Libvirt virtualization API is on board and comes with hot plug support for the VirtioFS virtual filesystem. Version 4.0.0 of virt-manager, a graphical program for managing VMs on Linux, is included and provides a graphical option for configuring shared storage. VirtioFS is available here as a selectable filesystem in the settings. Virt-manager also automatically activates the Trusted Platform Module (TPM) if the VM uses UEFI. Another new default choice for x86 guests allows the host CPU to be passed through to the guests. And, last but not least, the Virtio GPU is available for most modern guest systems.

When creating VM templates, VMware users benefit from an innovation in cloud-init 22.1, which now natively supports VMware as a data source. The LXD data source dynamically reads instance data from the LXD socket and applies configuration changes that also survive reboots.

People who use VMs on a large scale usually turn to OpenStack. Despite rumors to the contrary, OpenStack is not dead, reports Canonical [6], while sending the new 2022 "Yoga" version off to do battle with its competitors. At the same time, the release notes warn that updates are not a walk in the park because OpenStack consists of many moving parts. Admins will therefore need to schedule some time for planning and testing the upgrades, and study the release notes [7].

Container Love

The container and VM manager LXD also comes with numerous new features, with version 5.0 now covering the same feature set for VMs as for containers. In multiuser operation, several users can start their projects separately. VMs now support virtual TPMs and PCI passthrough and can be migrated on-the-fly. In addition, LXD 5.0 lets you hot plug hard disks and USB sticks into VMs.

When it comes to Docker, Canonical points out that it not only offers Ubuntu

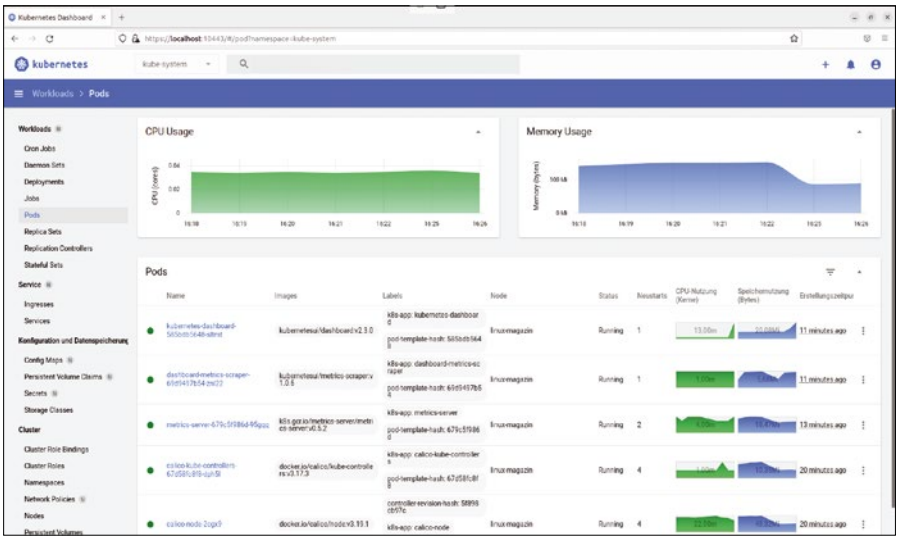


Figure 3: A local Kubernetes, including a dashboard, can be set up quite quickly thanks to MicroK8s.

itself on Docker Hub, but also numerous validated container images with MySQL, PostgreSQL, and NGINX. New additions include Grafana Loki, Apache Kafka, and Apache Cassandra.

If you are looking to build a larger container environment, you will find Kubernetes v1.23 in Ubuntu 22.04 [8]. While Canonical recommends its Charmed Kubernetes for enterprise deployments, the leaner MicroK8s (Figure 3) instead targets users who want to run the container orchestrator in edge computing or the Internet of Things (IoT) area. And, last but not least, Canonical Kubernetes also enables managed containerization with a managed Kubernetes.

Development Drive

Developers use Ubuntu because it supports numerous programming languages out of the box. And Windows users now also have an easy option for using familiar Linux tools in Windows Subsystem for Linux (WSL) 2, which now also supports Ubuntu 22.04.

Of particular interest to developers, PHP 8.1.2 is included. If you want to move up from version 7.x, note that version 8 removes some deprecated functions. As a result, some code adjustments may be needed. In return, PHP 8 promises better performance. Ruby 3.0 runs up to three times faster than its predecessor thanks to the MJIT compiler, concurrency, and static types, which is likely to go down well with its followers.

Python fans can look forward to version 3.10.4 and the Python-based Django web framework in the distribution. Django is available as version 3.2.12 with long-term support and offers asynchronous views and middleware, among other things. A word of caution: There is a risk of some incompatibility here during the upgrade. Ubuntu 22.04 also includes Go v1.18.x, Rust v1.58, and OpenJDK 11 for Java developers.

On the compiler side, Ubuntu has a great feature set with LLVM 14 and GCC 11.2.0. On the database side, PostgreSQL 14 and MySQL 8.0 impress with some new features. For PostgreSQL 14, stored procedures now return data via `OUT` parameters, simplifying the move from Oracle to PostgreSQL. MySQL admins can disable the audit log for sessions.

Point of View

With every LTS release, Canonical explains what it considers to be the highlights of the new version, which allows conclusions to be drawn about what customers have requested. This time, the company highlights native support for NVIDIA's vGPU software 14.0, among other things. This allows the virtual GPUs of many VMs to be linked together to accelerate machine learning and other scenarios with workloads that process serious amounts of data. In addition, Ubuntu 22.04 supports NVIDIA's AI Enterprise software suite, which offers advantages in a scientific context and high-performance computing.

If you want to use Azure's confidential VMs, Ubuntu is the only Linux distribution that supports the feature. Multipass [9], a GPL software driven by Canonical, lets you start an Ubuntu VM on Windows, macOS, and Linux with a simple command and (now) also supports Apple's M1 chip. A real-time kernel (currently still in beta) is expected to find favor especially in the telecommunications industry, for example, for real-time applications in the 5G sector.

Conclusions

If you are an admin or developer, you don't need to rush to switch to Ubuntu 22.04 LTS, but gradually transitioning only makes sense, even on your servers. You should consider making the switch not only because your favorite frameworks and content management systems will eventually switch to the new programming language versions, but also because updated software may utilize some of the new features that Ubuntu 22.04 delivers. If you want to wait, most of the teething troubles should be confined to history by the time the first point release, 22.04.1, appears. Before you make the change, check through the release notes [10] for the known issues taking your infrastructure into account. ■■■

Info

[1] HWE kernel: *https://www.thomas-krenn.com/en/wiki/Ubuntu_LTS_Hardware_Enablement_Stack_information*

[2] Kernel releases: *https://www.kernel.org/category/releases.html*

[3] Plans for Ubuntu kernel: *https://discourse.ubuntu.com/t/ubuntu-desktop-gnome-plans-for-the-incoming-lts/26156/13*

[4] NFS and UDP: *https://www.mail-archive.com/kernel-packages@lists.launchpad.net/msg473086.html*

[5] SSH on older machines: *https://bugs.launchpad.net/ubuntu/+source/openssh/+bug/1961833*

[6] Ubuntu and OpenStack: *https://ubuntu.com/blog/openstack-is-dead*

[7] OpenStack "Yoga": *https://releases.openstack.org/yoga/*

[8] Kubernetes 1.23: *https://ubuntu.com/blog/kubernetes-1-23-release-top-features*

[9] Multipass: *https://multipass.run*

[10] Release notes: *https://discourse.ubuntu.com/t/jammy-jellyfish-release-notes/24668*

Exploring decentralized chat and microblogging platforms

Free Connection

Diaspora, Friendica, and Mastodon are free and decentralized microblogging platforms that keep you in control of your data. *By Erik Barwaldt*

Social media platforms are well established as an important means of communication for corporate environments as well as for personal users. Most users are aware, however, that data sovereignty is a significant problem in the social media space: Almost all leading platforms collect and aggregate user data for advertising purposes. In addition, government organizations sometimes make use of this information. In the USA, for example, providers are legally obligated to hand over data about their customers to investigative authorities such as the FBI after receiving a National Security Letter. Many users also worry about their data becoming exposed due to an attack, and centralized services pose a higher risk for attack, as the data is concentrated on a relatively small number of servers.

The problems associated with the massive, proprietary social media platforms have caused some companies and home users to pay more attention to short message services based on open source projects. These open source solutions are free of data mining, and their decentralized nature makes them less susceptible to the security problems associated with the Internet giants. This article examines some leading open source alternatives in the social media space.

All three of the tools described in this article support ActivityPub, a free protocol for supporting cross-platform social networks [1]. If two social media platforms support the ActivityPub protocol, they can exchange messages.

Diaspora

The Diaspora project [2], which is supported by the Diaspora Foundation, is a decentralized network whose servers (pods) form a globally distributed system. Diaspora's feature set is similar to industry giants Twitter and Facebook. Users can operate with a high degree of anonymity and often do not reveal their true identities. In addition, users retain all rights to their data, which rules out personalized tracking. Tech-savvy users who want to support the project can install dedicated Diaspora pods, where their data is kept secure locally.

To create a Diaspora profile, press the *Join!* button top right on the project website, and register using the green button that then appears. A small selection of suggested servers will appear for you to join. If you would like to see a complete list of all pods, including various snippets of statistical data, click on the list icon bottom right, which leads to a display showing availability information, locations, and the numbers of users (Figure 1).

To log in to one of the servers, simply click on its URL. You can only create an account on pods that have a *yes* in the *logins* table column. After you select a server, press the *Create account* button in the top right-hand corner of the splash page. All you need is a valid email address, a username, and a password. You can also upload a profile photo and specify your areas of interest. The *Awesome! Take me to diaspora* button finally takes you to the message area (Figure 2).

When you get there, you will see the message stream on the right, with various instructions appearing when you first access it. In the first box, you can directly enter a message; select the recipient in the *Public* button below. On the left is an options bar for various settings. In the activity bar at the top, you can click to select various activities and also define your user settings with the help of a drop-down menu.

To change your user profile or adjust individual settings, click on your

Photo by Mohamed Nohassi on Unsplash

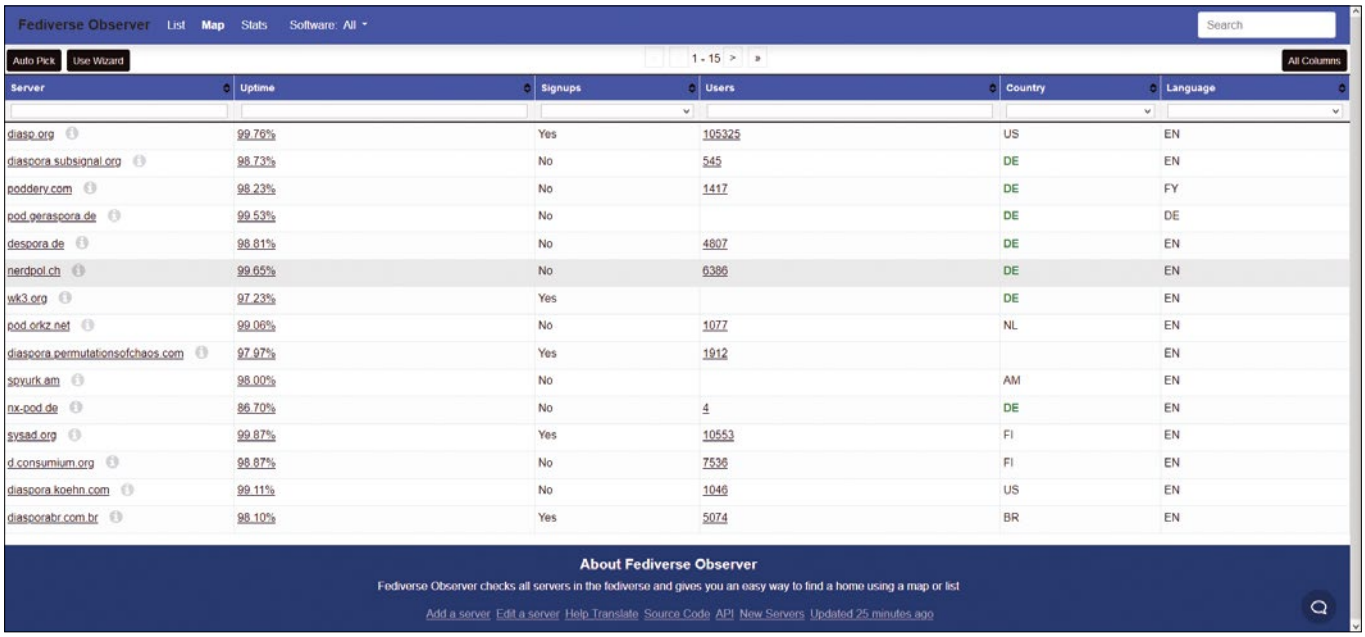


Figure 1: Diaspora lists all available servers on its website.

username in the top right-hand corner and then select *Profile*.

For many actions, emails will be sent to your stored address. The NSFW (not safe for work) option lets you highlight content that you do not want to appear while you are working. Accordingly, these messages do not appear in the conventional stream. This option is switched off by default, so you need to enable it manually.

Contacts lets you sort other users into groups, such as *Family*, *Friends*, *Work*, and *Acquaintances*. To organize one or more contacts in different groups, click on the desired group. Your contacts will then appear on the right-hand side of the window, arranged in a table.

Diaspora lets you to set up your own server, which is always integrated into the Diaspora network. To create a closed communication network based on Diaspora (without having to sacrifice functionality), you need to set up your own groups. Running your own pod involves some complex installation steps. The Diaspora developers provide somewhat outdated instructions for numerous Linux distributions.

Friendica

Friendica [3], which has been continuously maintained and developed since 2010, is one of the best-known free microblogging and chat services. Also decentralized, Friendica does not require a central server. You can connect with users on other Friendica servers or integrate contacts from Twitter, Diaspora, pump.io, and StatusNet into your message thread. Friendica also lets you share and distribute images. The software is modular and can be extended using plugins. On top of this, you can set up a Friendica server yourself and even make it publicly accessible.

On the project's website, after clicking on *Public servers* in the top right-hand corner, you will find a list of publicly accessible Friendica servers sorted by language. Choose a server, then connect to it by clicking on the given URL. The initial pages of the individual servers look almost identical. When you register, Friendica expects you to enter your name or pseudonym and an email address to match. If you already have a profile on another Friendica server, you can use it here, too. All you have to do is transfer it to the new server in the registration dialog.

After clicking the *Register* button, you will receive a message on the specified email account with a newly generated password. Using this password and your

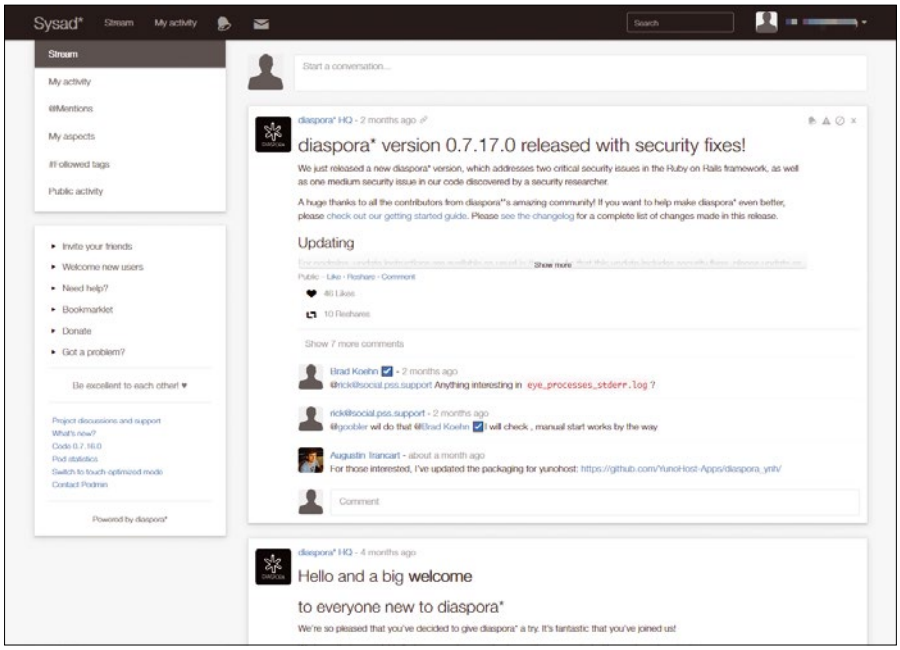


Figure 2: The Diaspora message window offers a convenient overview of available options.

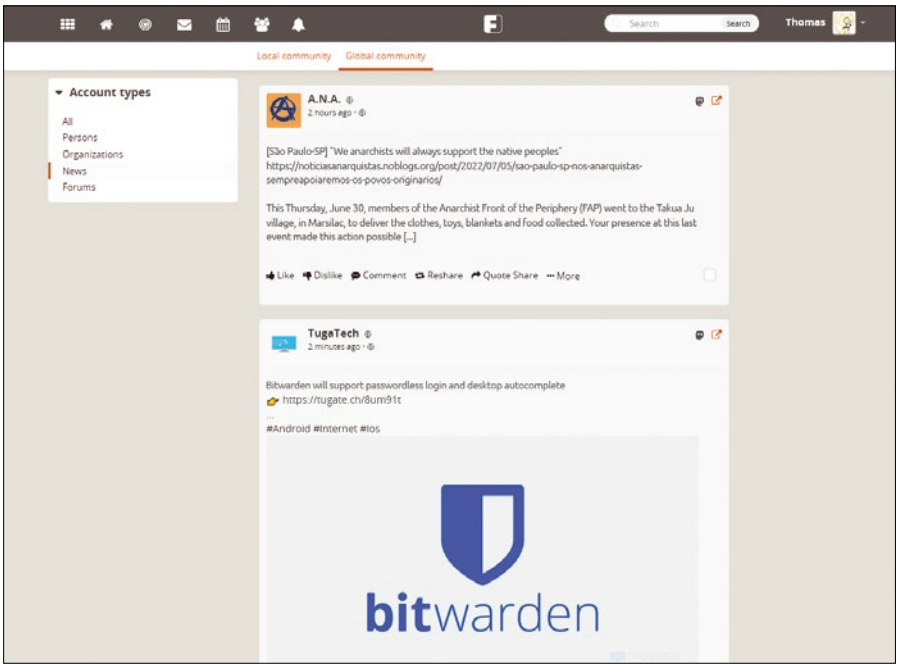


Figure 3: Friendica supports intuitive use.

username, you can now log on to the server, following which you are taken to a splash page. On the left, the various settings and administrative options are grouped together, and the message stream appears in the middle. At the top you will find several buttons for notifications and access to the most important functions (Figure 3).

To manage your own profile, click on your profile name in the top right-hand corner. In the context menu, use *Edit profile* to modify the basic settings.

Friendica offers detailed account management options that you can access by clicking on your username or avatar on the splash page. In the *Account Settings* dialog (Figure 4), you can define various security and privacy options or import contacts using a CSV file that you need to create separately. Use the *Expiration Settings* option to define how long posts remain visible before Friendica deletes them.

Friendica divides accounts into different types, which mostly differ in terms of how they handle contact requests. You can either manually accept requests for friends or followers or have them automatically sorted into categories. This does not work with accounts for discussion forums, however. You can also use several accounts simultaneously. The *Settings* | *Manage Accounts* menu lets you register your additional accounts and connect them to existing ones. If you are using a larger number of accounts in a larger organization, you will want to add administrators and authorized representatives with administrative rights.

You can access several external networks to add contacts. In Friendica, click *Contacts* | *Add New Contacts* in the main window to add a contact to your list. The name and interests search gives you an opportunity to link up with potentially like-minded people. Friendica also lets you read content from other platforms, such as Twitter, Mastodon, or Diaspora. However, to establish direct communication, you also need an account on the source server (Figure 5).

In the *Find people* settings box, you can narrow your search for contacts based on various criteria such as similar interests. If you select the *Local directory* option, only the potential contacts on the source server where you are currently logged in will appear. Use the (*Groups*) access bar to assign your Friendica contacts to groups.

Like Diaspora, Friendica lets you install your own server. Download the source code from the project's website in the *Use it* | *Install your own server* menu. Detailed documentation is available on the Friendica website [4]. Please note that you need to meet some software requirements to set up a Friendica server. These requirements include, for example, a LAMP stack and specific versions of the PHP programming language and the Apache web server. The project page lists the individual applications in detail. Friendica does not require any specific hardware, so you can use an old computer system as a server.

Mastodon

Mastodon [5], under development by Eugen Rochko since 2016, is another decentralized microblogging platform. With around 5.1 million users on 3,786 servers right now, it is one of the better-known free services.

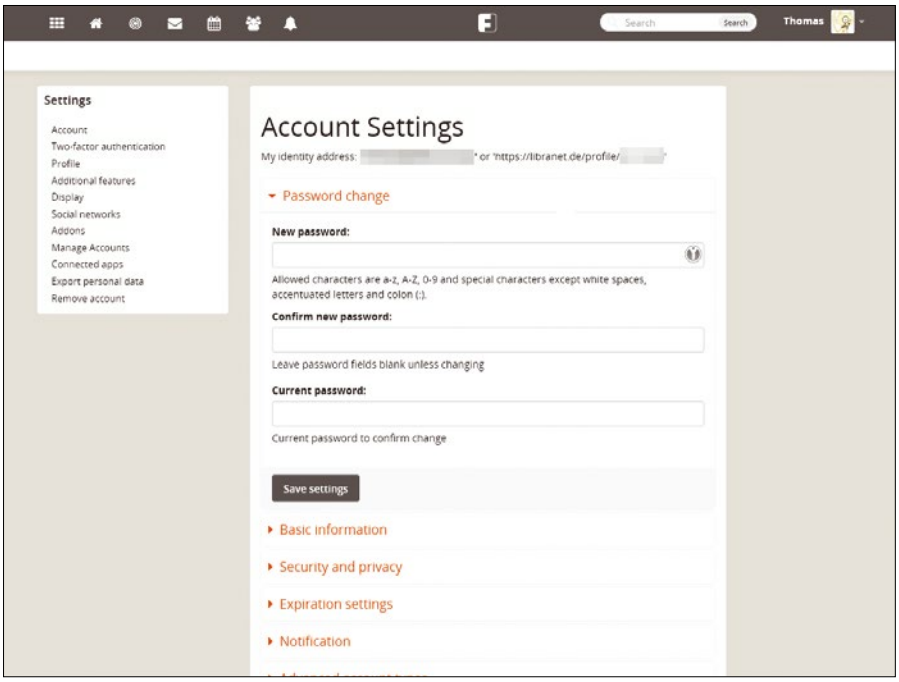


Figure 4: Friendica gives users a huge choice of account settings.

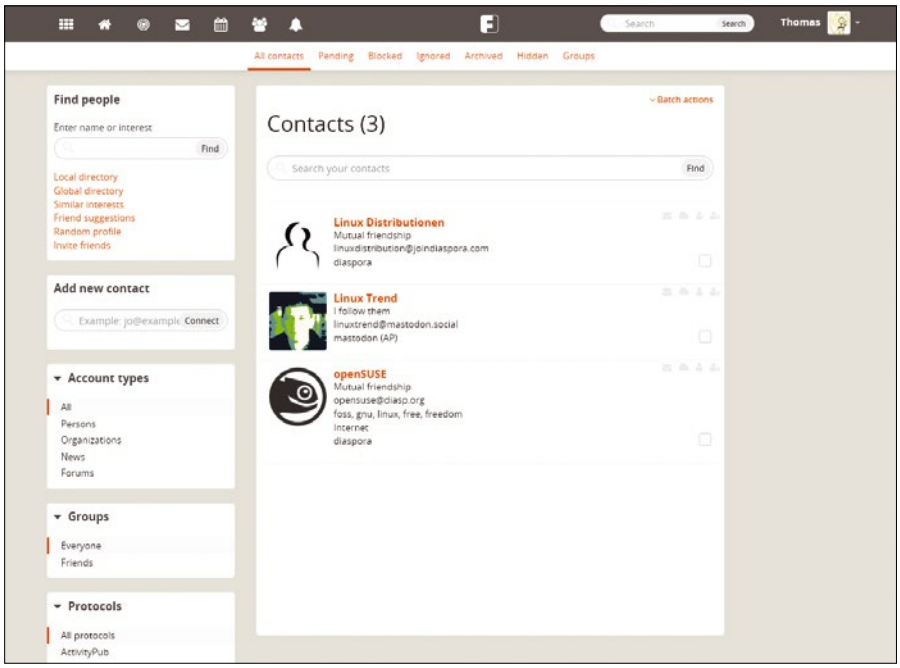


Figure 5: Contact management is unusually complex in Friendica.

Registering is just as easy as with other decentralized networks: All you need is a valid email address; just enter a freely selectable profile name and choose a password. For subsequent logins, enter the specified email address and password to access a somewhat unusual, three-column interface (Figure 6).

On the right are numerous configuration and information tools. The message threads appear at the center, and a search function and input field for the messages (*Toots*) are available on the left. In the right column, you can adjust your profile in the settings dialog. To enable two-factor authentication (2FA), go to the *Account* dialog.

The community wizard is one of Mastodon's special features; you can use the community wizard to find meaningful groups quickly. Open the *https://joinmastodon.org* page in your browser. Then click on *Get started*; you are taken to a selection screen with various communities listed by category (Figure 7).

If you join a group, you are taken to the login and registration page for the Mastodon instance, where you can log in to the server. You will then arrive at a standard start page with the standard news stream. To follow the stream on the local instance, click on *Local* in the vertical option bar on the right.

It is important to note that users can log on to multiple instances simultaneously; however, profiles are not automatically transferred between individual Mastodon servers. To use a user profile you created on several instances, save it in a CSV file and then import it.

Like its competitors, Mastodon lets users host their own servers. To host a server, you can either use dedicated

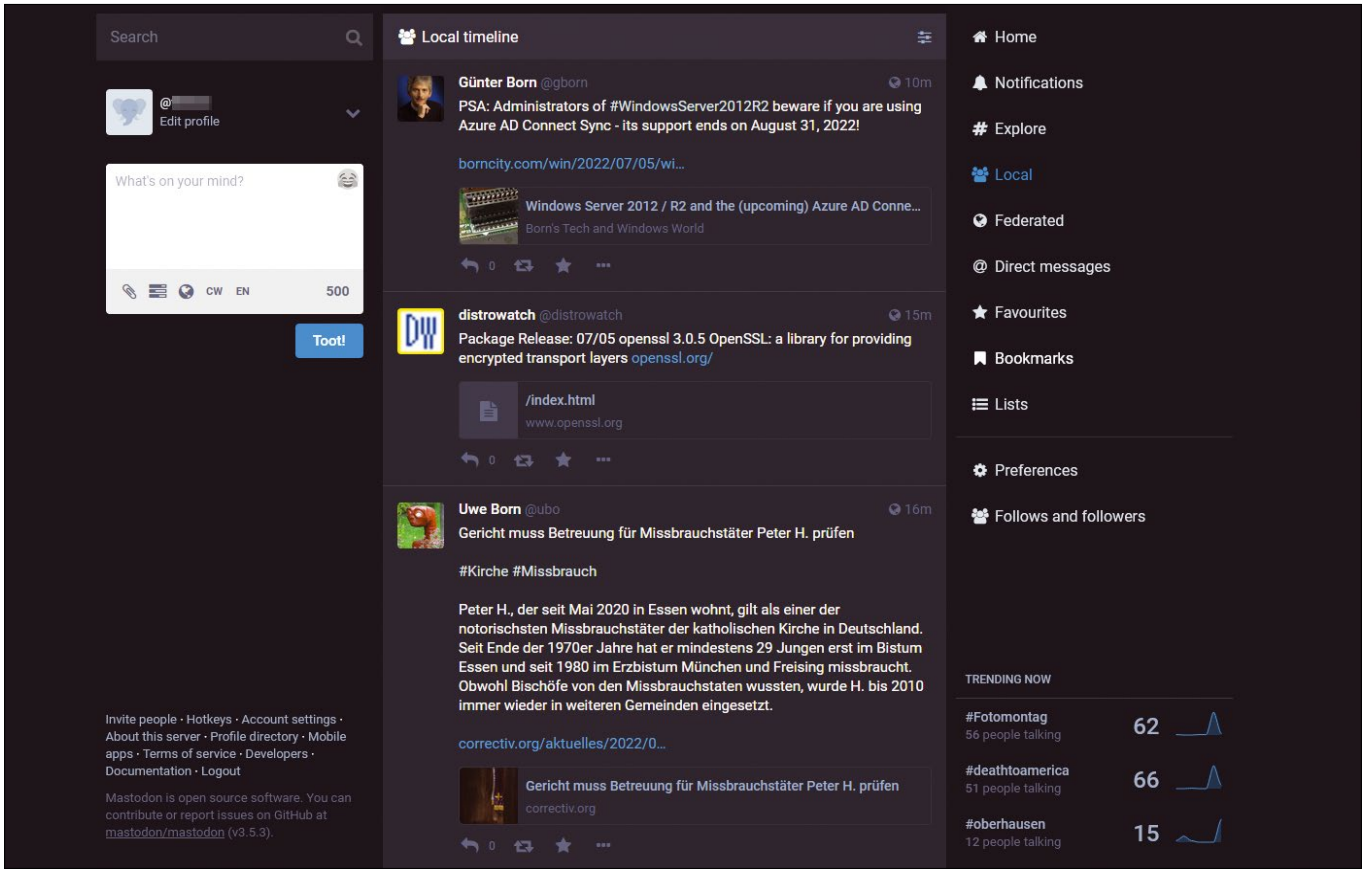


Figure 6: Mastodon uses a three-column user interface.

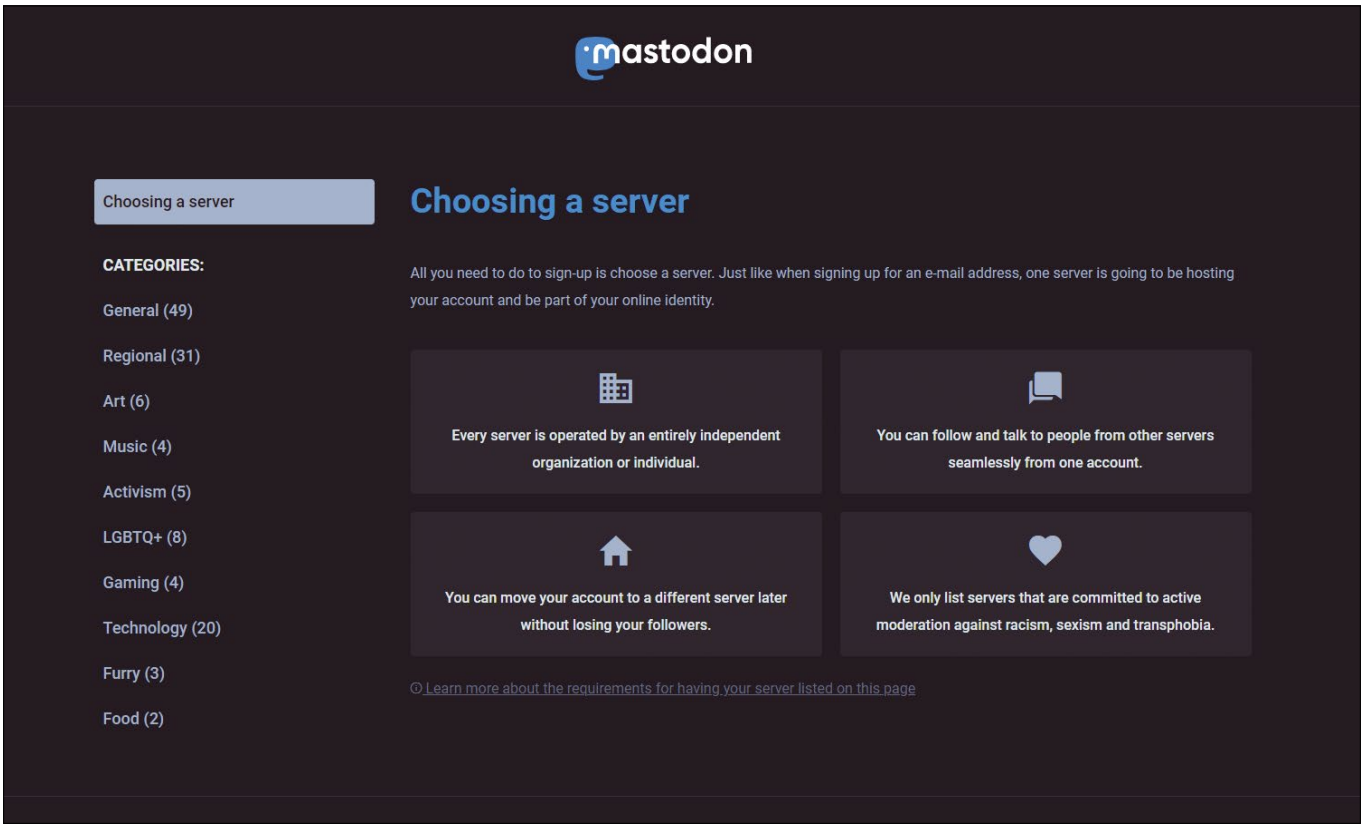


Figure 7: Mastodon offers a community wizard.

hardware or host Mastodon in the cloud. The Mastodon documentation provides detailed information on the various options for hosting a Mastodon instance [6].

Conclusions

Each of the three decentralized chat and microblogging platforms fulfills its purpose and offers a feature set that is very much on par with top dogs like Twitter. (See Table 1 for a summary of important features.) The user interfaces take a little getting used to, but thanks to their self-explanatory design, they require very little training. The decentralized approach gives users control over their data with all services, guaranteeing an essential level of security through various security functions. Setting up a separate server instance requires some manual work. Closed groups on your own instance mean that admins in smaller companies can set up internal communication platforms without security worries. ■■■

Table 1: Decentralized Social Media Networks

	Diaspora	Friendica	Mastodon
License	AGPL	AGPL	AGPL
Functions			
Registration with an email address	Yes	Yes	Yes
NSFW option	Yes	No	Restricted
2FA supported	Yes	Yes	Yes
Own server supported	Yes	Yes	Yes
Close groups supported	Yes	Yes	Yes
Profile transfer	Yes	Yes	Yes

Info

[1] ActivityPub: *https://www.w3.org/TR/activitypub/*
[2] Diaspora: *https://diasporafoundation.org*
[3] Friendica: *https://friendi.ca*
[4] Friendica installation: *https://friendi.ca/resources/installation/*
[5] Mastodon: *https://joinmastodon.org*
[6] Mastodon documentation: *https://docs.joinmastodon.org/user/run-your-own/*

Linux Magazine **is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!**

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

Photo by Nicolai Berntsen on Unsplash

Simple web scraping with Bash

Ski Report

With one line of Bash code, Pete scrapes the web and builds a desktop notification app to get the daily snow report. *By Pete Metcalfe*

While recently doing a small project, I was amazed by how much web scraping I could do with just one line of Bash. I used the text-based Lynx browser [1] and then piped the output to a `grep` search. Figure 1 shows the one-line Bash example that scrapes the current snow depth from the Sunshine Village Snow Forecast web page.

In this article, I will introduce some techniques to easily scrape web pages, and then I will create a desktop notification script that provides the daily snow forecast.

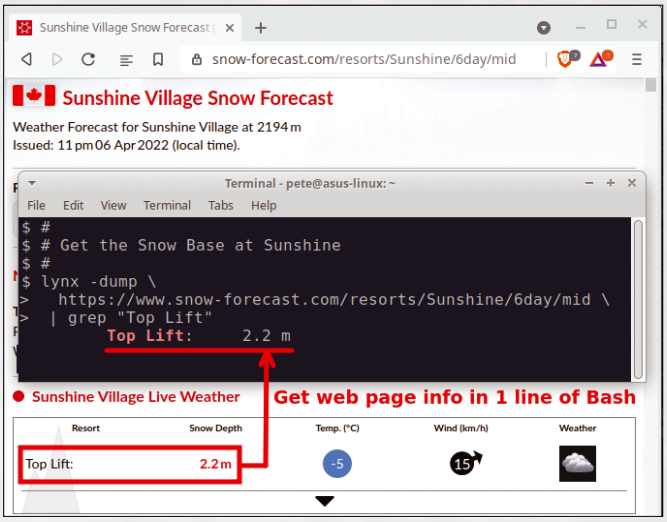


Figure 1: One line of Bash code finds the web text for the current snow depth.

The Lynx Text Browser

For my Bash web scraping, I started out by looking at using command-line tools

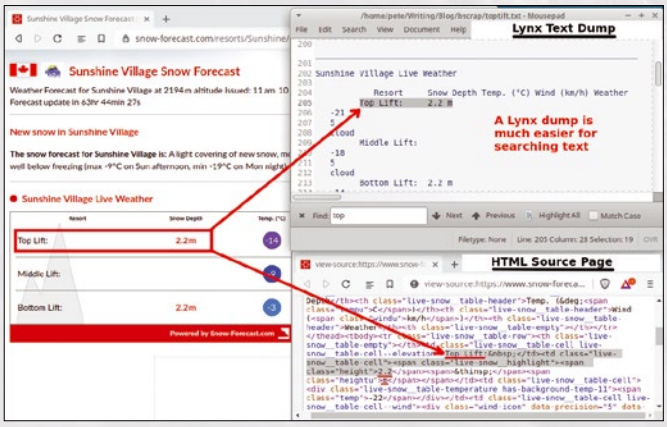


Figure 2: Lynx output removes HTML tags, encoding, and JavaScript, making it easier to search.

such as `curl` [2] with the `html2text` [3] utility. This technique definitely works, but I found that using the Lynx browser offers a one-step solution with a slightly cleaner text output.

To install Lynx on Raspian/Debian/Ubuntu, use:

```
sudo apt install lynx
```

The Lynx `-dump` option will output a web page to text with HTML tags, HTML encoding, and JavaScript removed. Figure 2 shows that a Lynx dump can greatly clean up the original web page and make searching considerably easier.

Sometimes a simple Bash `grep` search might be all that you need. However, there are many cases where some text manipulation is required. The good news is that Bash has a nice selection of line and string manipulation tools.

The example shown in Figure 3 uses line manipulation to find the current weather in Key West, Florida. A `grep` search is done on the string *"As of"*, and the option `-A 3` is used to return the requested line of data with an additional three lines. You can remove the *"As of"* line with the `tail` command if required.

It's important to note that what you see on a web page may not match the Lynx outputted text, and some trial and error testing might be required.

Figure 4 uses string manipulation to find the new snow at Sunshine Ski

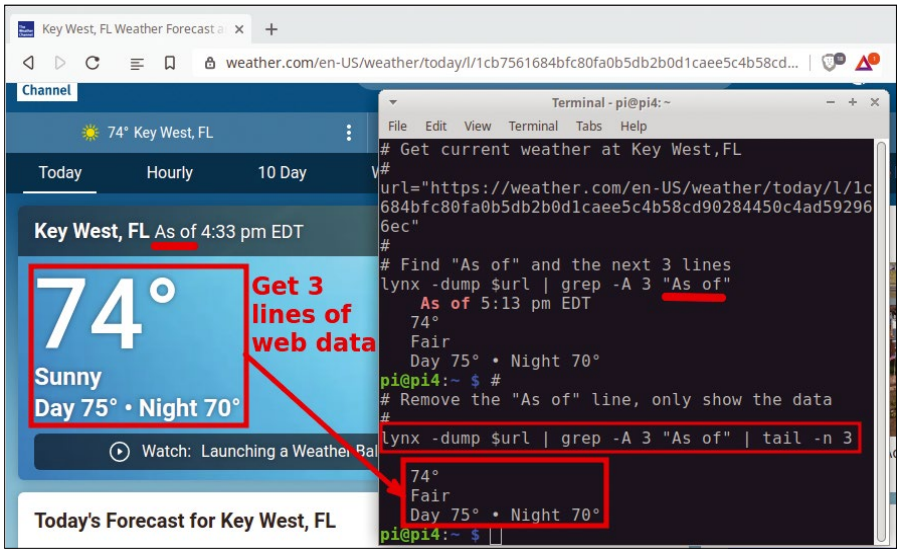


Figure 3: Using Bash line manipulation to extract web data.

Listing 1: Extracting Parts of a String

```
01 $ newsnow="5.2cm2.0"
02 $ # get the part before 'cm'
03 $ echo "${newsnow%%cm*}"
04 5.2
05 $ # get the part after 'cm'
06 $ echo "${newsnow#*cm}"
07 2.0
```

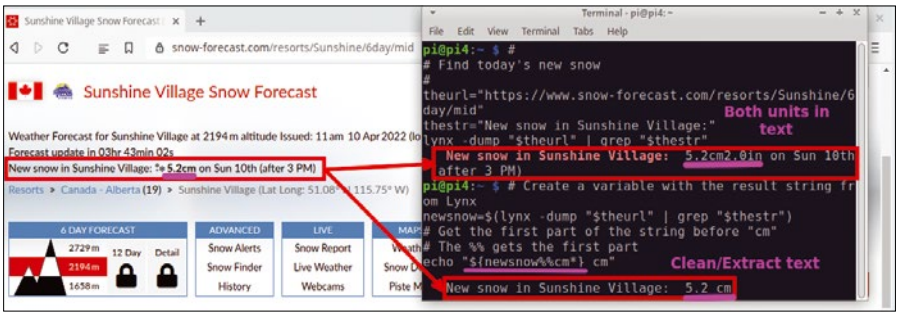


Figure 4: Here, Bash string manipulation extracts the desired web data.

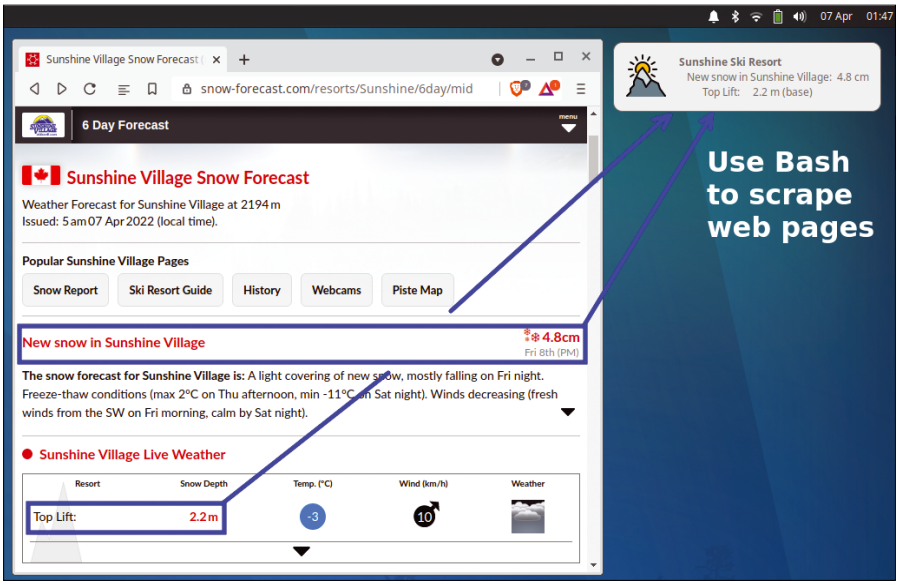


Figure 5: Using Bash web scraping, the notification script displays the daily snow report.

Listing 2: Bash Web Scraping Notification Script

```
01 #!/bin/bash
02 #
03 # skitrip.sh - show the Sunshine ski conditions in a notification
04 #
05 theurl="https://www.snow-forecast.com/resorts/Sunshine/6day/mid"
06
07 # Get the new snow depth
08 thestr="New snow in Sunshine Village:"
09 result=$(lynx -dump "$theurl" | grep "$thestr")
10 newsnow="${result%%cm*} cm"
11
12 # Get the base
13 thestr="Top Lift:"
14 base=$(lynx -dump "$theurl" | grep "$thestr")
15
16 # Show the results in a desktop notification, with 120 minute wait time
17 msg="$newsnow\n$base (base)"
18 icon="$HOME/Downloads/mountain.png"
19 notify-send -t 120000 -i "$icon"  "Sunshine Ski Resort" "$msg"
```

Resort. The resort's web page uses JavaScript to show the new snow in either centimeters or inches, but the Lynx text output displays both values and their units.

To remove parts of a string variable, you can use `%%` to extract the first part of the string and `#` to extract the last part of the string (as shown in Listing 1).

A Bash Web Scraping Project

To get excited before a family ski trip, I wanted to create a morning notification script that would show the new morning snow and the base snow.

To create the notification script (Listing 2), I used two passes with the Lynx utility. The first pass scrapes for new snow (shown in Figure 4) and then a second pass gets the snow base (shown in Figure 1). The snow results are then passed as a string (`$msg`) to the `notify-send` utility [4], which posts the message to the workstation desktop (Figure 5). You can schedule this Bash script to run every morning using either *cron* or the *at* utility.

Summary

Scraping web pages can be tricky, and the pages can change at anytime. For this reason, it is always best to check if an API is available before looking at web scraping.

Python with the Beautiful Soup library has been my go-to approach for web scraping, but it's nice know that a simple Bash alternative is also available. ■■■

Info

[1] Lynx: *https://lynx.invisible-island.net/*

[2] curl: *https://curl.se/*

[3] html2text: *https://github.com/Alir3z4/html2text/*

[4] notify-send: *https://delightlylinux.wordpress.com/2020/10/25/bash-show-notifications-from-scripts-using-notify-send/*

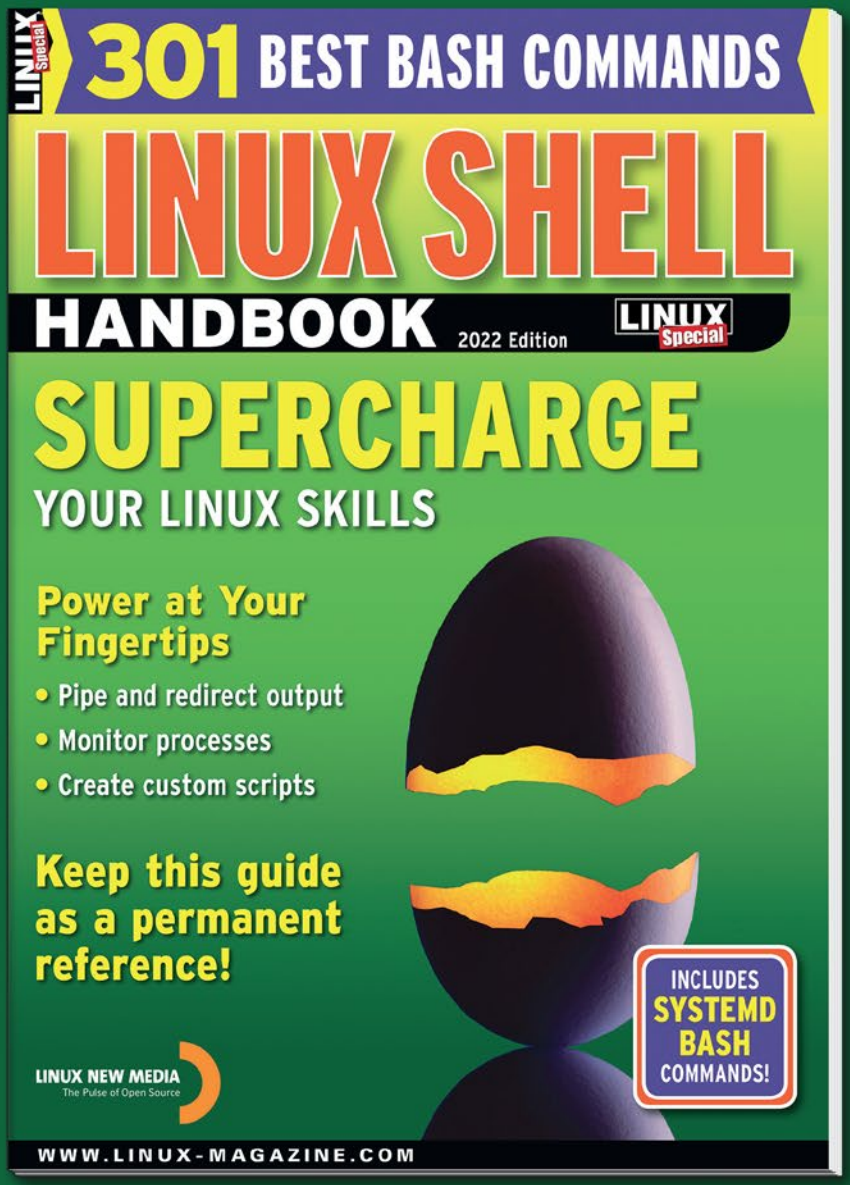
Author

You can investigate more neat projects by Pete Metcalfe and his daughters at *https://funprojects.blog*.

THINK LIKE THE EXPERTS

Linux Shell Handbook 2022 Edition

This new edition is packed with the most important utilities for configuring and troubleshooting systems.

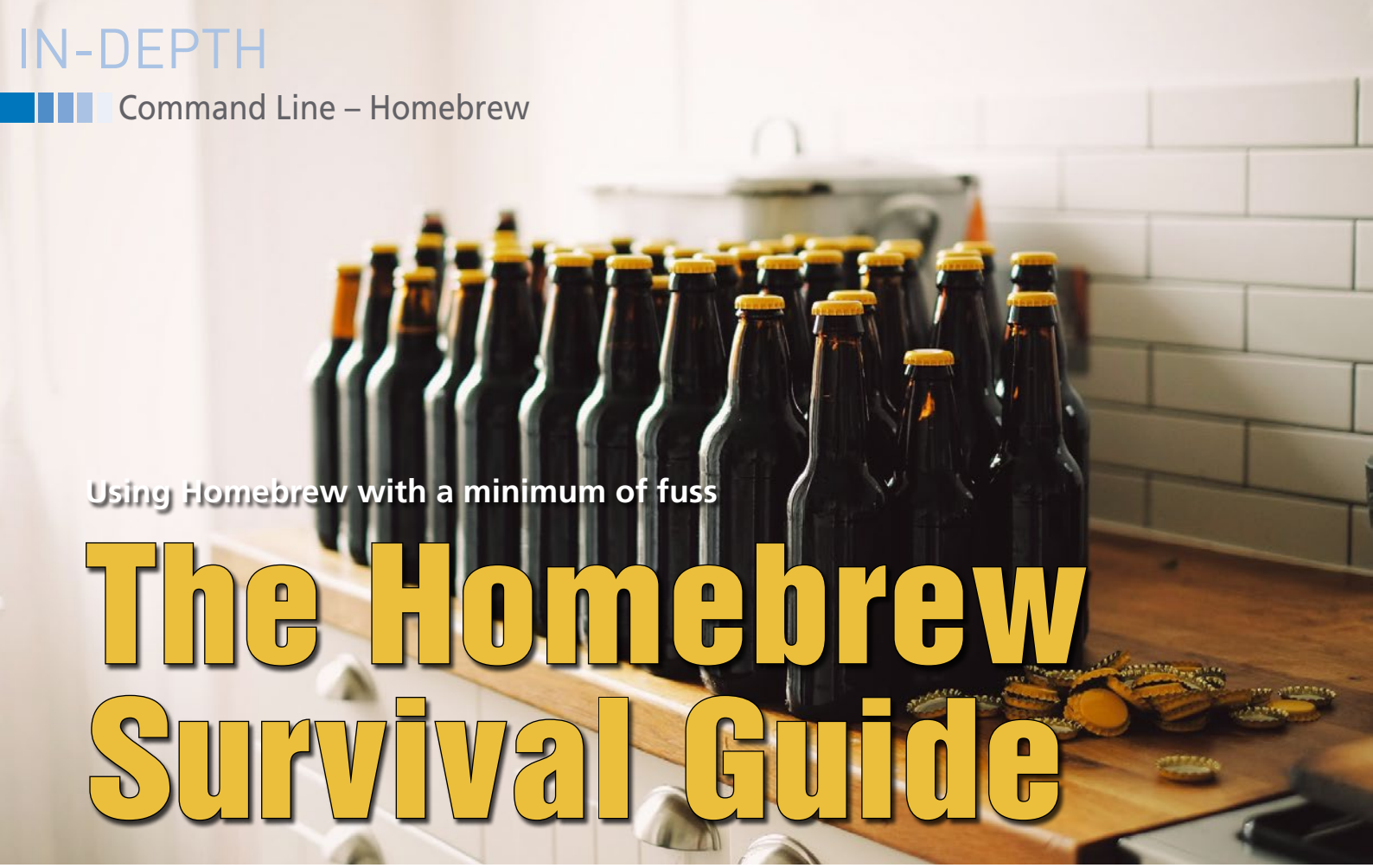


Here's a look at some of what you'll find inside:

- Customizing Bash
- Regular Expressions
- Systemd
- Bash Scripting
- Networking Tools
- And much more!

ORDER ONLINE:

shop.linuxnewmedia.com/specials



Using Homebrew with a minimum of fuss

The Homebrew Survival Guide

Homebrew, a comprehensive package manager, has been increasing in popularity thanks to its ease of use. *By Bruce Byfield*

Linux has no shortage of package managers. Besides basic ones such as RPM, DNF, and dpkg/apt-get/APT, there are supposedly universal ones such as Flatpak and Snap, and increasingly, one for each programming language. Originating in macOS and formerly called Linuxbrew on Linux, Homebrew [1] is especially popular in the Ruby on Rails community. Recently, however, it has started gaining a larger popularity due to its ease of use. If you want to install anything from a project in early development, increasingly there is a good chance that you will need Homebrew to do so. Homebrew offers the option of non-root installation, access to developing software outside your distribution's repositories, and multiple versions of applications. In addition, if you maintain multiple operating systems, you can use the same package manager and set of commands on Linux, macOS, and the Windows Subsystem for Linux.

Homebrew installs files to `/home/linuxbrew` and symlinks them to `/usr/local`, so that you do not need to be root to use it. Before installing, make sure you have all the necessary packages by running the command

```
apt install build-essential procps ➥
  curl file git
```

If you plan to run Homebrew from a regular user account, you will also need to set up the account to access sudo, because the installation script may ask for your sudo password. When you are ready, install Homebrew [2] [3] with:

```
/bin/bash -c "$(➥
  curl -fsSL ➥
  https://raw.githubusercontent.com/➥
  Homebrew/install/HEAD/install.sh)"
```

When installation succeeds, you will see the message *Installation successful*, followed by additional instructions (Figure 1). These consist of adding Homebrew to your Bash path using the series of commands in Listing 1, run one at a time.

Because none of these commands offers any feedback, you can test that Homebrew is properly installed by running

```
homebrew install hello
```

and running the `hello` command. If Homebrew is not running, run `brew update` a couple of times and `brew doctor` to see if

Listing 1: Adding Homebrew to Your Bash Path

```
test -d ~/.linuxbrew && eval "$(~/.linuxbrew/bin/brew shellenv)"
test -d /home/linuxbrew/.linuxbrew && eval "$(/home/linuxbrew/.linuxbrew/bin/brew
  shellenv)"
test -r ~/.bash_profile && echo "eval "$($(brew --prefix)/bin/brew shellenv)\"" >>
  ~/.bash_profile
echo "eval "$($(brew --prefix)/bin/brew shellenv)\" >> ~/.bash_profile
```

Photo by Adam Wilson on Unsplash

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (*http://brucebyfield.wordpress.com*). He is also cofounder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

```
bb@ilvarness:~$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)
==> Checking for `sudo` access (which may request your password)...
==> This script will install:
/home/linuxbrew/.linuxbrew/bin/brew
/home/linuxbrew/.linuxbrew/share/doc/homebrew
/home/linuxbrew/.linuxbrew/share/man/man1/brew.1
/home/linuxbrew/.linuxbrew/share/zsh/site-functions/_brew
/home/linuxbrew/.linuxbrew/etc/bash_completion.d/brew
/home/linuxbrew/.linuxbrew/Homebrew
==> The following new directories will be created:
/home/linuxbrew/.linuxbrew/bin
/home/linuxbrew/.linuxbrew/etc
/home/linuxbrew/.linuxbrew/include
/home/linuxbrew/.linuxbrew/lib
/home/linuxbrew/.linuxbrew/sbin
/home/linuxbrew/.linuxbrew/share
/home/linuxbrew/.linuxbrew/var
/home/linuxbrew/.linuxbrew/opt
/home/linuxbrew/.linuxbrew/share/zsh
/home/linuxbrew/.linuxbrew/share/zsh/site-functions
/home/linuxbrew/.linuxbrew/var/homebrew
/home/linuxbrew/.linuxbrew/var/homebrew/linked
/home/linuxbrew/.linuxbrew/Cellar
/home/linuxbrew/.linuxbrew/Caskroom
/home/linuxbrew/.linuxbrew/Frameworks

Press RETURN/ENTER to continue or any other key to abort:
```

Figure 1: The start of Homebrew's installation script. Notice the assumption of sudo and the installation directory of `/home/linuxbrew`.

there are any obvious issues (Figure 2). If you still have problems, check the online list of common issues [4].

A Note on the Jargon

An annoying feature of Homebrew is that it extends the metaphor of its name, giving different names for its directories depending on their contents, such as cellar, rack, or kegs, making it hard to know how one term relates to another. Similarly, when installing, the output talks of "Pouring" (Figure 2). Fortunately, for many basic uses, you can ignore this needless complication, but Figure 3 shows how the various terms are related to each other. Mostly, you only need to know that a formula is a package, and a manifest is a package's installation script. These two pieces of jargon break the metaphor, but perhaps that comes as a relief. Occasionally, however, it may be quickest to work with multiple formulae by using, for instance, commands that affect kegs or racks (i.e., directories with multiple formulae). If you need to know more about other terminology, a summary is available online [5].

Using the brew Command

The basics of Homebrew are almost identical to those of most package managers. That is, they consist of the basic command, followed by a subcommand or action, and then the specific package affected, if any. Commands that do not specify a formula apply to all of Homebrew, such as `brew autoremove`. Table 1 shows a list of basic commands, using `gcc` as an example. This is, of course, a different version of `gcc` than any installed from a distribution's repositories. Like Debian and other distributions, Homebrew also maintains a web page of available formulae (one the same page as the Homebrew terminology [5].

Table 1: Basic Homebrew Commands

Action	Command
Install	`brew install gcc`
Remove	`brew remove gcc`
Auto-remove dependencies	`brew autoremove`
Upgrade formula	`brew upgrade gcc`
Upgrade all formulae	`brew upgrade`
List installed formulae	`brew list`
List available formulae	`brew formulae`
Search	`brew search TEXT`

```
bb@ilvarness:~$ brew install hello
==> Downloading https://ghcr.io/v2/homebrew/core/hello/manifests/2.12.1
######################################################################## 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/hello/blobs/sha256:7935d0efdae69742f5140d514ef2e3e50d1d7cb82104cf6033ad51b900c12749
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:7935d0efdae69742f5140d514ef2e3e50d1d7cb82104cf6033
######################################################################## 100.0%
==> Pouring hello--2.12.1.x86_64_linux.bottle.tar.gz
🍺  /home/linuxbrew/.linuxbrew/Cellar/hello/2.12.1: 55 files, 650KB
==> Running `brew cleanup hello`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
bb@ilvarness:~$
```

Figure 2: Homebrew formulae installs are "Pourings." Output is conveniently emphasized by arrows and progress bars for each step.

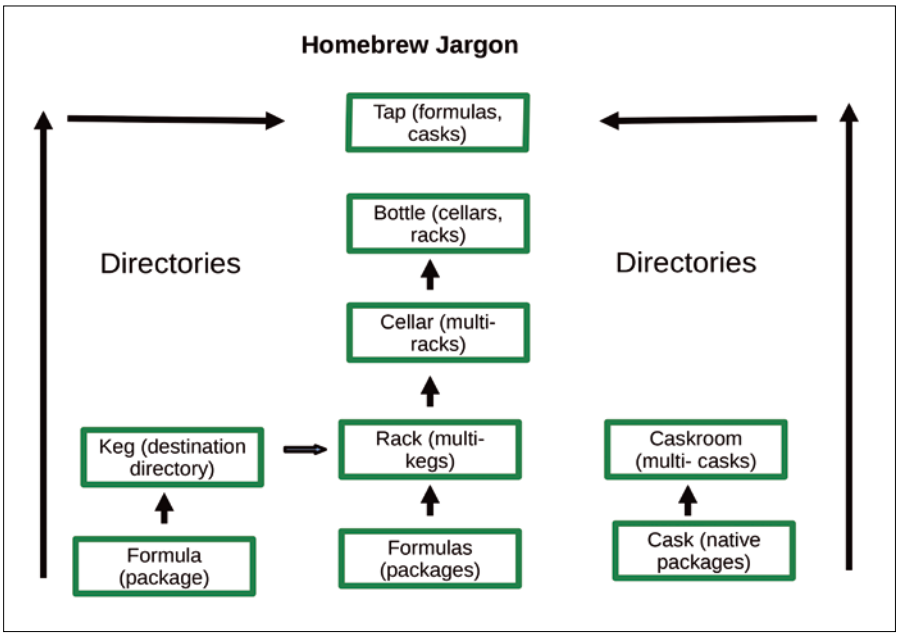


Figure 3: Homebrew has a complicated series of names that reflect the contents of each directory.

As with many package managers, these eight commands are enough for most user interactions with Homebrew. However, Homebrew also has a lengthy man page that for some reason is not installed with it [6]. Some of the options are specific to developers creating formulae, such as the spellcheck `typecheck` command or the `analytics` command for repositories, and will not be discussed here for lack of space, but a number are also useful for convenience or administration purposes. For example, among the convenient options are `completion`, which autocompletes typed commands once enabled for a Bash, Zsh, or fish shell by linking to online dictionaries [7]. Similarly, `home` or `homepage`, qualified by a formula's or cask's name, opens to the target's web page so you can learn more about them. In addition, once you have mastered Homebrew's jargon, there are sub-commands and options for dealing with formulae in groups, instead of individually.

The administrative options include a number of options that help keep a system current. For instance, cleanup action uses the `--prune DAYS` option to remove files in Homebrew's cache that are older than a certain number of days or `-s` to remove the cache entirely. Another potentially useful action is `outdated`, which lists formulae for which a newer version is available. This is especially useful when accompanied by the action `migrate`, which takes options for when a formula's name is changed in a newer version, or `bump`, which sets whether an older version should be updated.

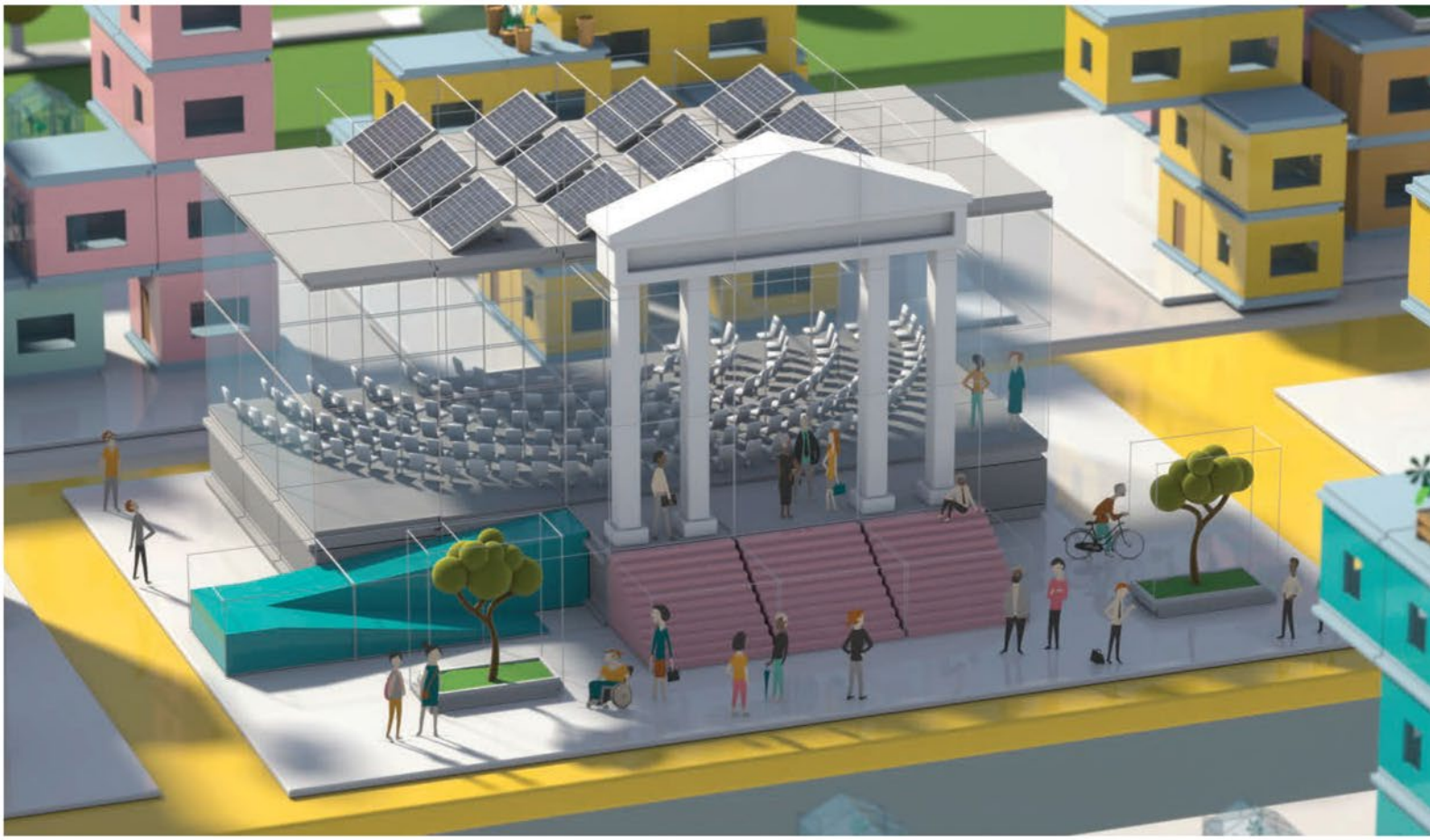
A Niche App

Homebrew is a comprehensive package manager. Despite the fact that it is only 13 years old, in many ways it is as far-reaching as the much older `apt-get` or Yum. It even includes features that could be useful in other package managers. However, access to Homebrew's advanced features is partially blocked by unnecessary jargon. As well, while its formulae must number in the thousands, they are nowhere near, for instance, Debian's 60,000 packages. It is only in the Ruby on Rails community that Homebrew is likely to dominate. For the rest of us, these notes should be enough to use Homebrew when a developer decides to use it, with a minimum of fuss and only a slight loss of convenience. ■■■

Info

[1] Homebrew: *https://brew.sh/*
[2] Install Homebrew: *https://docs.brew.sh/Installation*
[3] How to Install and Use Homebrew on Linux: *https://www.digitalocean.com/community/tutorials/how-to-install-and-use-homebrew-on-linux*
[4] Common issues: *https://docs.brew.sh/Common-Issues*
[5] Jargon and formulae: *https://docs.brew.sh/Formula-Cookbook#homebrew-terminology*
[6] Man page: *https://docs.brew.sh/Manpage*
[7] Shell completion: *https://docs.brew.sh/Shell-Completion*

Public Money
Public Code



Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

Build your own web server in a few simple steps

Self Made

If you want to learn a little bit more about the communication between a web browser and an HTTP server, why not build your own web server and take a closer look. *By Goran Mladenovic*

Programming your own web server might seem like a difficult and unnecessary undertaking. Any number of freely available web servers exist in the Linux space, from popular all-rounders like Apache or NGINX to lightweight alternatives like Cherokee or lighttpd (pronounced "lighty").

But sometimes you don't need a full-blown web server. If you just want to share a couple of HTML pages locally on your own network or offer people the ability to upload files, Linux on-board tools are all it takes. A simple shell script is fine as a basic framework that controls existing tools from the GNU treasure chest. Network communication is handled by Netcat [1], aka the Swiss army knife of TCP/IP.

Getting Ready

With a project like this, the best place to start is at the root. Because a web server is still a server at the end of the day, it needs to constantly listen on a given port and respond appropriately to requests. Usually, web servers listen on port 80 for normal requests, and port 80 generally only accepts HTTP requests without encryption. The web server I'll describe in this article listens on ports 8080 and 8081 and communicates without encryption. If you are using a firewall and want to test the server on the local network, remember to allow these two ports in the firewall.

A web server needs a root folder from which it loads the requested HTML files. It also needs a directory in which it can store uploaded files. Your first step is to define a configuration using a series of simple variables at the start of the server script (Listing 1). And you need to create the directories, along with a FIFO file, either manually or using the Bash test builtin. The `server6.sh` script, which is included with the code from this article [2], offers a solution.

In the last line of Listing 1, you can see that your own IP address is also important. You will need to modify the network device specification (the Ethernet interface `enp2s0` in this example) to suit your own system. When a web browser tries to submit a file via a web form, it needs a target address. GET requests are the simplest approach to doing this. When a browser sends a GET request, it expects the content of a web page in response, and it displays this content in the browser window.

Listing 1: Configuration

```
HTTP_HOME=http_home
HTTP_UPLOAD=${HTTP_HOME}/upload
CACHE_DATEI=${HTTP_UPLOAD}/filetoprocess
FIFO_GET=fifo_get
HTTP_GET_PORT=8080
HTTP_POST_PORT=8081
MEINE_IP=$(ip addr show <enp2s0> | grep -Eo "([0-9]{1,3}\.){3}[0-9]+" | sed 1q)
```

Sample Files

Files for testing the web server are easily scripted. The function in Listing 2 runs through a `for` loop seven times. The routine uses a *here document* (heredoc) to support the entry of HTML code almost 1:1 (third line). Heredocs let you refer to the variable set in the `for` statement, which then simply contains the sequence number.

Heredocs help to define sections of text in many programming languages. Unlike conventional output via `echo` or `printf`, line breaks, indents, and some special characters are preserved in the text. Bash also supports the use of variables in heredocs.

In this way, you can create as many HTML files as you need with just a few lines of code. You could optionally integrate additional dynamic content that you generate with a script within the heredoc.

Listing 2: Creating Sample Files

```
function create_files () {
  for x in {1..7}; do
    cat <<-FILE > ${HTTP_HOME}/datei${x}.html
      <html><head><meta charset="utf-8">
      <title>Page ${x}</title>
      </head><body>
      <p> $( date ) </p>
      <p> Page ${x} </p>
      </body></html>
    FILE
  done
}
```

Listing 3: Netcat Response

```
while true; do
  respond < $FIFO_GET | netcat -l $HTTP_GET_PORT > $FIFO_GET
done
```

Listing 4: FIFO File

```
01 function respond () {
02   read get_or_post address httpversion
03   if [ ${#address} = 1 ]; then
04     list_dir
05   elif [ ${#address} -gt 1 ]; then
06     return_file $address
07   fi
08 }
```

You'll also need to create some sample HTML files for testing your homegrown server. (See the box entitled "Sample Files.")

GET Requests

Responding to a GET request entails much more than just sending the content of a file. HTTP and HTTPS require that additional information be sent along with the transmission. If you want to know what a response from a genuine web server looks like, type the following command:

```
wget --spider -S ➥
"https://www.zeit.de/index"
```

The `wget` utility downloads a web page from the terminal. The `--spider` option tells `wget` to behave like a web spider; in other words, it won't download the actual content but will check that the content is there and will receive the transmission information associated with an HTTP request.

In the first line, the server confirms that it is happy to take the HTTP request – `HTTP/1.1 200 OK`. Further lines in the form of value pairs (such as `Connection: keep-alive`, `Content-Length:300`) are used to send back additional information or instructions.

It also appears that this service is a well-secured web server, because it does not reveal precisely what kind of server program it is. Many servers out themselves at this point as `server: nginx`, for example – not advisable, because such disclosures makes things easier for attackers. If you want Netcat to behave like a genuine web server, you'll need a way to generate this header information associated with HTTP.

Netcat

Netcat is available on virtually any Linux system and can be used for many purposes given a little creativity on the user's part, although it admittedly has some limitations. You can emulate basic network operations using Netcat, but complex interactions are difficult or impossible. You definitely don't want to try to compete with Apache or NGINX just using Netcat.

If you want Netcat to permanently listen on a port and also send different responses, you have to combine it in a loop with a FIFO file. FIFO refers to the "first in, first out" principle. This means that the information comes back out of the file in the same order in which it was sent in [3]. Listing 3 shows an example.

The FIFO file improves the communication between Netcat and the `respond` function, as shown in Listing 4. Netcat listens on the specified port and writes to the FIFO file. On the left side of the pipe, you can see the call to the function that reads the browser request. It evaluates the request and then sends a matching response, containing an HTML header and HTML data, back through the pipe to Netcat. The `respond` function decides what to return to the browser.

This variant is already a fairly powerful solution. If the length of the browser request is 1 (line 3), then it is `/`, and Netcat returns a directory listing. If the length is not equal to 1, Netcat returns the content of a file from the root directory. To get the web server to return a list of the files contained in the root folder, a very simple `ls directory_name` is all that is needed. However, the results then need to be embedded in suitable HTML code so that the links work and the browser can actually use them (Figure 1). The sed [4] stream editor is recommended for converting a directory listing into HTML code.

Listing 5 shows the functions referenced in Listing 4. In the `list_dir` function, the directory content is output with a simple `ls` command. Sed then converts the results into plain vanilla HTML. The files generated by the function from Listing 2, which reside in the root directory, already contain HTML code. The server uses the `return_file` function in line 19 of Listing 5 to send a file back to the browser with a matching header.

Because Netcat is continuously available for requests in the loop and sends a

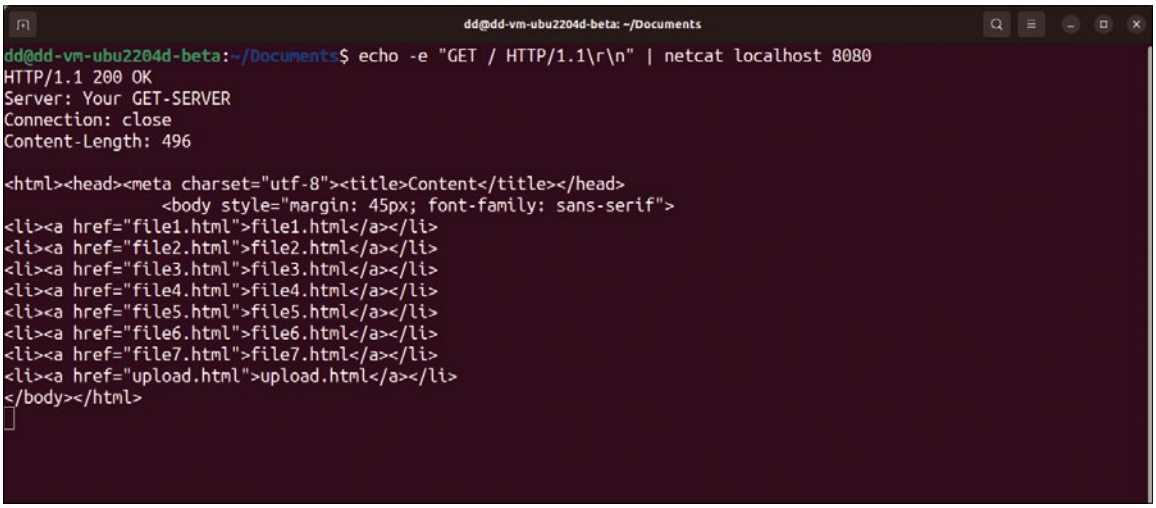


Figure 1: The DIY web server returns a listing of the root directory content.

Listing 5: Output

```
01 function list_dir () {
02   local output=$( ls --hide=upload -1 $HTTP_HOME | sed -r '
03   1 i<html><head><meta charset="utf-8"><title>Content</title></head>\
04   <body style="margin: 45px; font-family: sans-serif">
05   s#(.*)#<li><a href="\1">\1</a></li>#
06   $ a</body></html>
07   ' )
08
09   local content_length="Content-Length: $( cat <<<$output | wc --bytes )"
10
11   cat <<<$output | sed '
12   1 i HTTP/1.1 200 OK
13   1 i Server: Your GET SERVER
14   1 i Connection: close
15   1 i '"$content_length"'\n
16   '
17 }
18
19 function return_file () {
20   content=$( cat ${HTTP_HOME}/${1:1} )
21   if [[ $? -eq 0 ]]; then
22     laenge=$( cat <<<${content} | wc --bytes )
23     cat <<<${content} | sed -r '
24       1 i HTTP/1.1 200 OK
25       1 i Server: Your GET SERVER
26       1 i Connection: close
27       1 i Content-Length: '"$length"'\n'
28   else
29     cat <<-ERROR
30       HTTP/1.1 404 Not Found
31       Connection: close
32       Content-Length: 42
33
34       The requested page does not exist, sorry!
35   ERROR
36   fi
37 }
```

header and the corresponding HTML, a browser in the local network thinks it is dealing with a real web server.

However, it can also happen that the user manually requests a page in the browser that does not exist. This leads to the infamous 404 error, which you have probably seen on the web before [5]. The custom web server can also come up with this feature. If the `cat` command in the first line of the `return_file` function (line 20) throws an error, the `else` branch starting at line 28 is executed. The web browser then displays a message that the requested page does not exist.

POST Requests

Unlike GET requests, where the web browser wants to download files, there are also POST requests that allow the browser to send data to the web server. You can think of this as like posting something on social media. You type some text, add images, or even add videos in a box provided for that purpose, and then press *Post*. The content is then uploaded to the server and subsequently displayed under your profile. Our simple server only uploads files from a browser and saves them in the `uploads/` folder.

Again, the browser sends a header indicating that it wants to post something. You can easily find out what a post request looks like by running the command in Listing 6. In the browser, call the web form in the root folder and send a file (Figure 2). After a few seconds, interrupt the Netcat command by pressing Ctrl + C. The browser displays a *File arrived* message, and the file is where you redirected it. But this is not a displayable JPEG file, because the file saved here still contains the header, as shown in Figure 3.

Listing 7 shows how sed can get rid of the excess data that you do not want

Listing 6: POST Simulation

```
$ echo "File arrived" | netcat -l 8081 > upload/filex.jpg
```

Listing 7: Filtering

```
01 function run_post_server () {
02
03 message_for_post='HTTP/1.1 200 OK
04 Content-Length: 13
05 Connection: close
06
07 File arrived
08
09 while true; do
10   cat <<< $message_for_post | netcat -l $HTTP_POST_PORT >
     "${CACHE_DATEI}"
11   new_name=$( sed -r -n '/filename/{ s/(.*)(filename=")
     (.+)(".*)/\3/; p}' ${CACHE_DATEI} )
12   upload_path="${HTTP_UPLOAD}/${new_name}"
13   sed '1,/filename/d;/Content-Type/{N;d};$d'
     ${CACHE_DATEI} > "${upload_path}"
14 done
15 }
16
17 run_post_server &
```

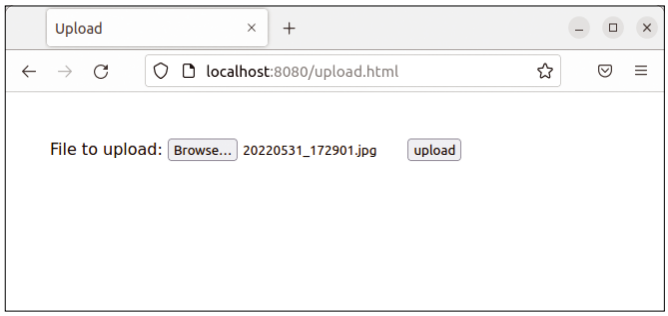


Figure 2: Using the web form to upload files such as photos or texts to the web server.

Listing 8: Sending a JPEG File

```
#!/bin/bash

header="HTTP/1.1 200 OK"
myfile="http_home/upload/IMG-20220213-WA0002.jpg"
content_length="Content-Length: $( cat $myfile | wc --bytes )"
content_type="Content-Type: image/jpeg"

cat $myfile | sed -r -e "1 i $header" -e \
    "1 i $content_length" -e \
    "1 i $content_type" -e \
    "1 i Connection: close\n" | netcat -l 8080
```

in the uploaded file. Sed handles this task in the `while` loop starting in line 9. Sed removes the header, boundary statements, the file name, and similar data. To compare this with what the data originally looked like, take a look at the cache file, which is also in the upload folder. If sed didn't remove all the ballast, the operating system would be unable to display the received files correctly.

In the background, the routine also calls the `run_post_server` function (line 17). This function contains a matching response for POST requests stating the content length in bytes and containing instructions to break down the connection after reading. Without these instructions, Firefox would simply keep the connection option, although the data has already been sent. The function launches in the background (`&`) to avoid it blocking everything as soon as the files have been sent.

Unchecked

Even if the web browser explicitly requests the root directory or another file, the web server can basically return whatever you want – you just need to declare the returned content correctly for the browsers. Listing 8 shows an

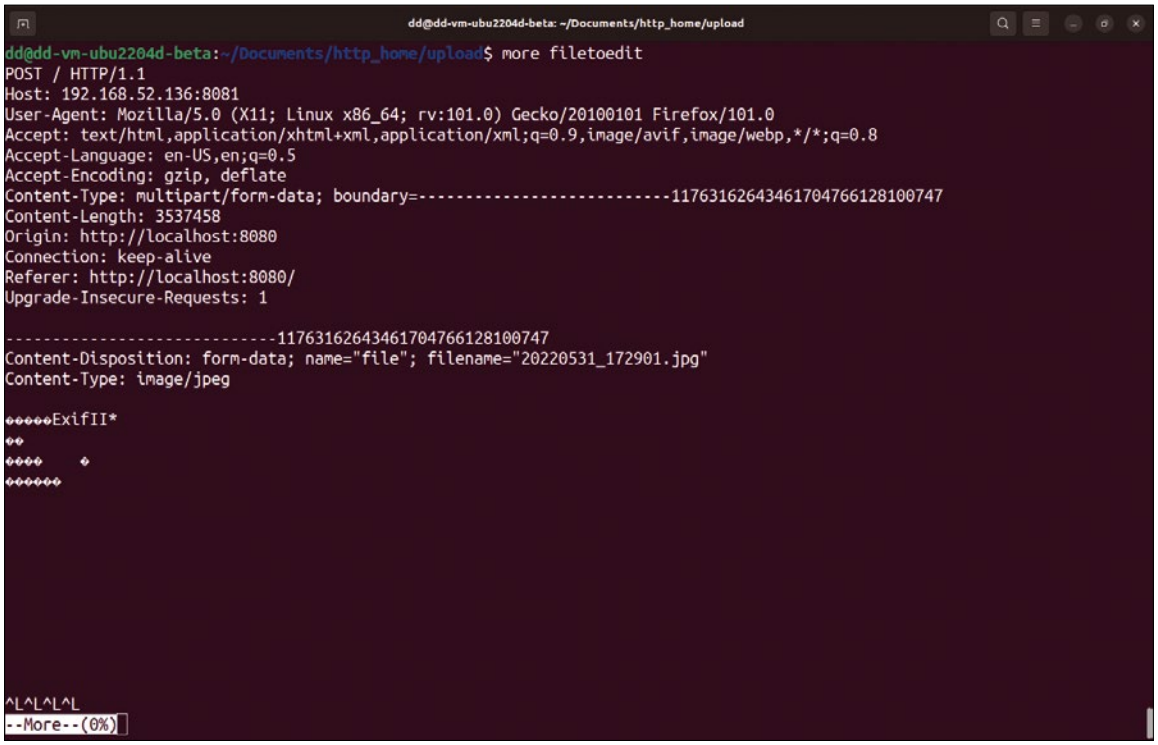


Figure 3: The upload request contains data that does not belong to the uploaded file.

example of this where the browser immediately displays a JPEG file on calling `localhost:8080` or `IP_address:8080` without any complaints.

The interesting thing here is not just that this works, but that it also represents a potential vulnerability. Apparently, most web browsers don't bother checking whether the content of the GET request and the returned page actually match. In this case, the browser asked for the index page of the web server and was given a JPEG file instead. That's something like a tennis player getting a basketball thrown at them by their opponent all of a sudden.

From experience, these idiosyncrasies, and many other features you might want to implement, are not very well documented on the web or are not documented at all. That's why it could be useful to log what Firefox and other browsers request. The function in Listing 9 starts the server. You can see two `tee` redirects there that forward all of the data to a logfile for debugging. This log will then contain the date and time, what the web browser sent as a request, and what the server sent back as a response (Figure 4). Armed with these details, you can analyze each request and response and understand what exactly is going on when the client and server talk.

For example, many browsers ask for the famous `favicon.ico` after they have talked to a server for a little while. This is the icon that you usually see at the top of the browsers' tabs. It is usually found in the web server's root folder.

If you want your own server to provide a favicon, you first need to find out what the browser request looks like and then tell the server to respond appropriately. You can tell that the web browser often asks for this file from the error message `cat: http_home/favicon.ico: file or directory not found` in the logfile.

Listing 9: Calling the Web Server

```
function run_server () {
  while true; do
    date | sed -r 's/^|$/\n/g' >> debug
    respond < $FIFO_GET | tee --append debug |
      netcat -l $HTTP_GET_PORT |
    tee --append debug > $FIFO_GET
  done
}
```

Conclusions

As you can see, a rudimentary web server is quite easy to build yourself. The web server presented in this article has a plain and simple design, but it is not intended to compete with major league players like the Apache web server or NGINX. On the other hand, your homegrown web server does have some capabilities that a typical web server can't offer. For instance, you can access the whole repertoire of shell commands to display information locally with minimal overhead. The resources consumed by the small script are also minimal. This DIY server is quite useful as an info server on your own network, and you can also use it to transfer files from one computer to another – all told, not a bad solution for small tasks. ■■■

Author

Goran Mladenovic is a hobby developer and inventor, who believes programming is a passion.

Info

[1] Netcat: *http://netcat.sourceforge.net/*

[2] Code for this Article: *https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z*

[3] Queue: *https://en.wikipedia.org/wiki/Queue_(abstract_data_type)*

[4] sed: *https://www.gnu.org/software/sed/manual/sed.html*

[5] HTTP status codes: *https://en.wikipedia.org/wiki/List_of_HTTP_status_codes*

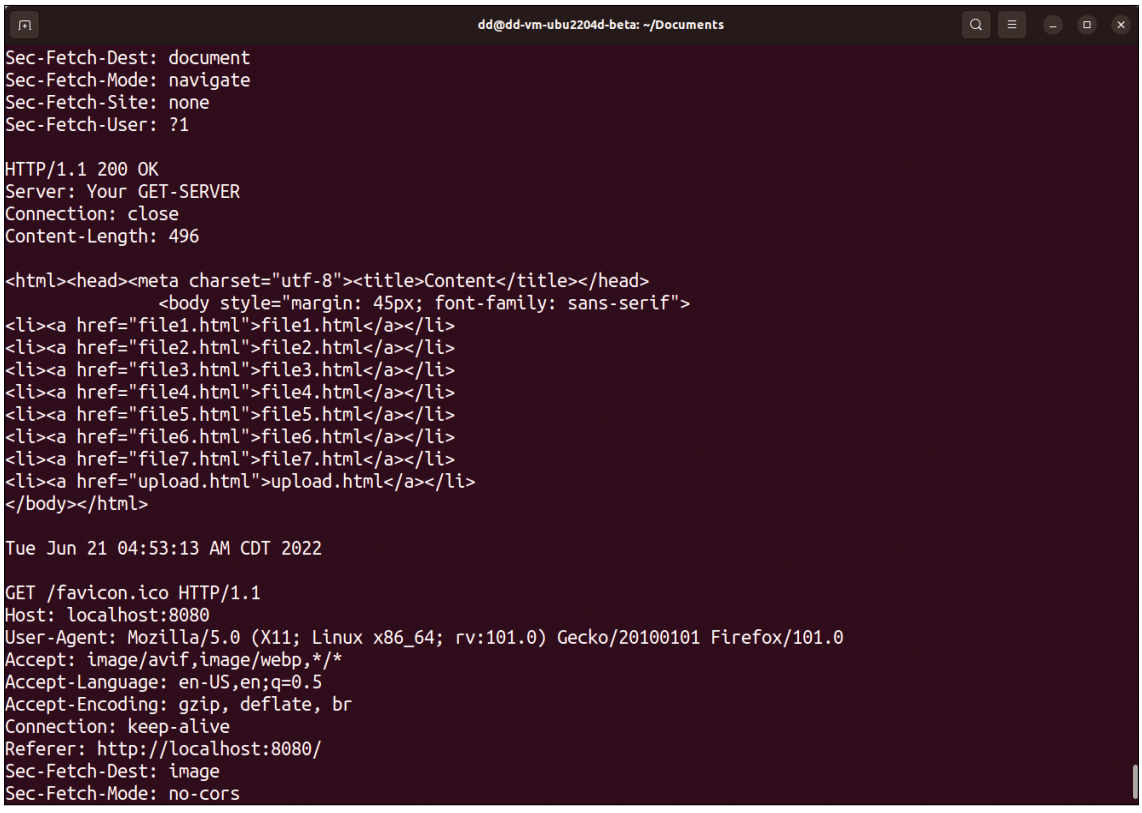


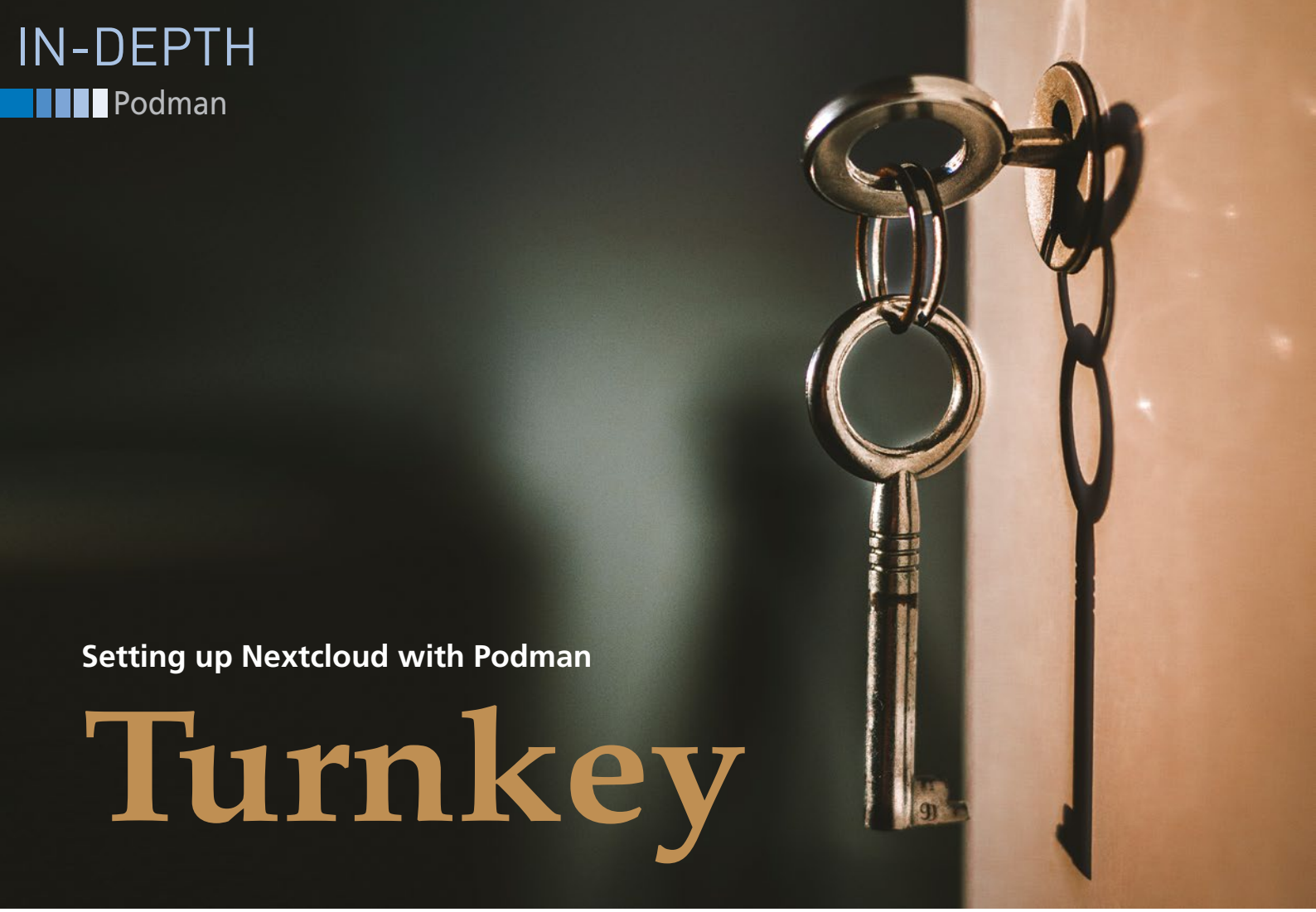
Figure 4: An excerpt from the debug file for port 8080.

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org



Setting up Nextcloud with Podman

Turnkey

Podman gives users a quick and easy way to set up a Nextcloud instance for home use.

By Ferdinand Thommes

Containers are increasingly making inroads into home networks. If you use Flatpaks or Snaps, you already use containers in everyday life. Future distributions will shed weight to a minimum, with required services running as containers of some kind. This development has been heralded by Fedora's Silverblue and Kinoite, Endless OS, MicroOS, and Intel's Clear Linux. It definitely makes sense for home users to consider the various container solutions.

Containers isolate applications through virtualization while providing a runtime environment. They make use of the filesystem and the resources of the operating system on which they run. This gives containerization the advantage of lower resource consumption compared with the traditional server approach or conventional virtualization. Where a virtual machine requires its own operating system, including a kernel, containers only store the actual applications plus any files and functions (microservices) required for execution.

Docker has long been synonymous with containers since its inception in 2013, but the advent of the Kubernetes container orchestration software has slowly started to change this perception. Recently, Podman [1] has been gaining momentum in the container sector, reaching version 4.0. After disputes between Docker and Red Hat over ongoing development, Red Hat began investing in Podman in 2017 as an application for managing containers and pods and has since cancelled support for Docker.

Podman (short for Pod Manager) has adopted the pod model introduced by Kubernetes. Pods are containers, each with individual applications running on the same server. If you want to set up Nextcloud, for example, you also need a server application, a database, and, if you want to access the service from the outside world, a reverse proxy. All of these applications run in separate containers in a pod. This offers benefits such as the ability to bind to the pod's localhost address, which means that all the containers in the pod can connect to it because of the shared network namespace.

In this article, I'll discuss the benefits of Podman and then show you a practical example by setting up Nextcloud with Podman.

Podman Benefits

While Docker is centrally controlled by a daemon, Podman does without such an instance and runs without root privileges. The containers run in the context of a normal user thanks to the use of the kernel's user namespaces based on Cgroups 2 [2]. In the container itself, however, the processes themselves run with root privileges. Inside a namespace, processes thus have different rights and user IDs than outside it. Because they are not controlled by a daemon, Podman containers can be included as systemd services [3] or controlled in a GUI using the Cockpit admin tool (Figure 1) [4].

Photo by Amol Tyagi on Unsplash

Unlike Docker, where individual components of an application run in different containers, Podman combines multiple containers in a single pod; this, in turn, avoids network problems. At the command line, Podman's behavior is almost identical to that of Docker, whose commands the software implements in the background. In addition, Podman can be used to create images of the Docker Registry repository service.

To make containers as resource-efficient as possible, you can use Buildah [5], which lets you build containers from

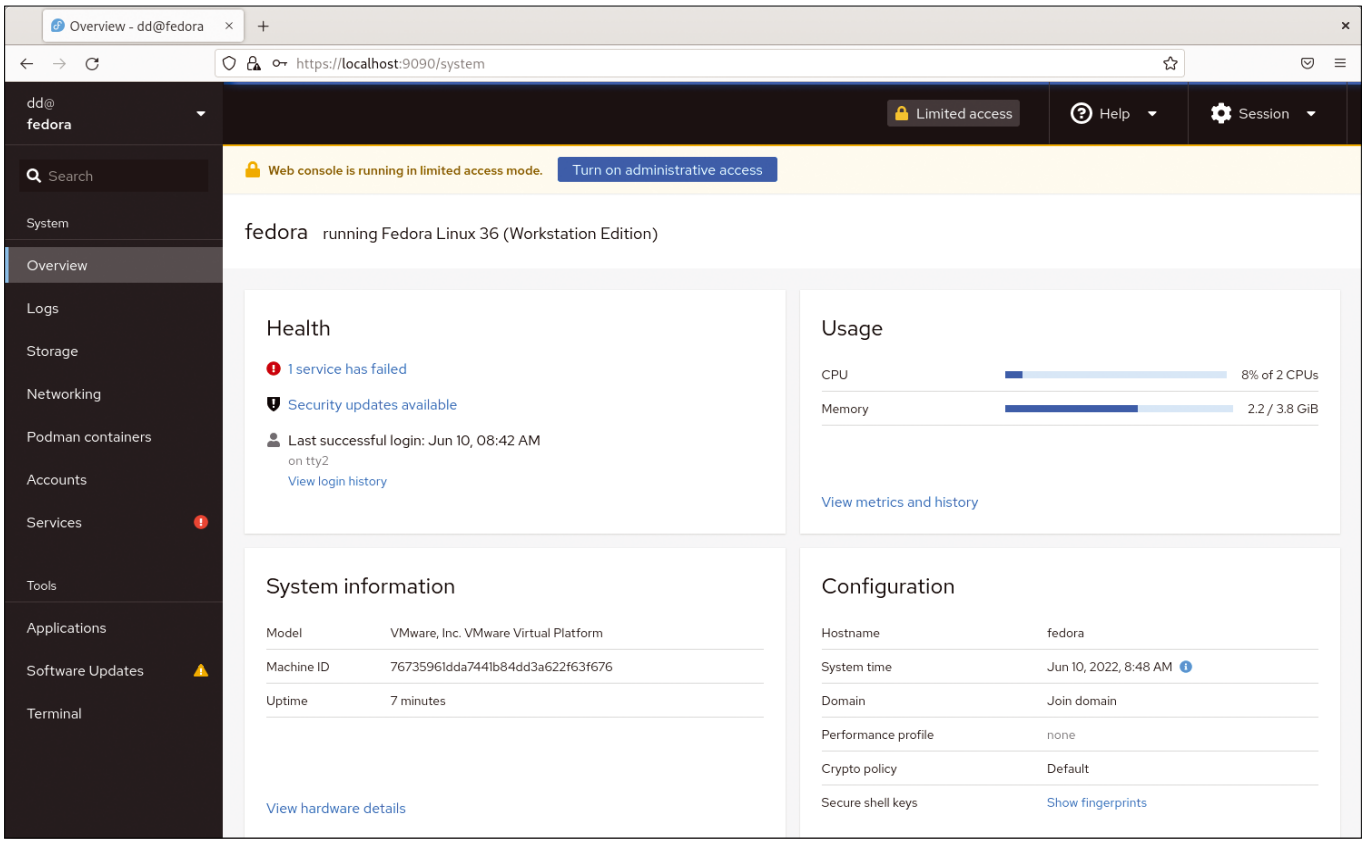


Figure 1: Cockpit, a web-based graphical user interface for servers, takes the pain out of managing local or remote computers in many ways. It recently added the ability to manage Podman.

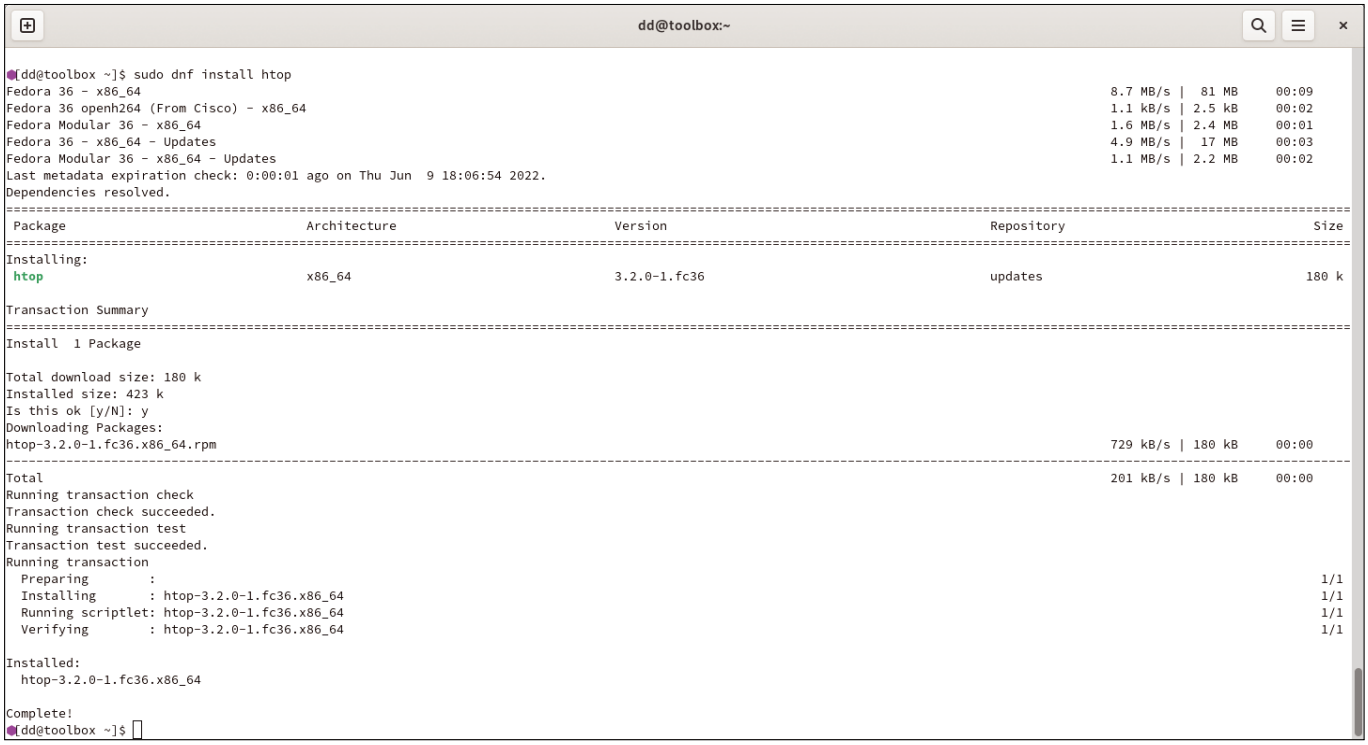


Figure 2: A handy little helper, `toolbox` has its origins in Fedora's immutable filesystems.

scratch. In particular, Buildah proves helpful in environments where you want the images to be as small as possible.

As you can see, Red Hat has elegantly solved its dependency on Docker with Podman, while providing additional functionality. For instance, Podman 4 comes with the new `podman image scp` command, which lets you copy images locally and to remote servers without detouring via a registry.

Installation

To set up a simple Nextcloud installation with Podman, I used both Fedora 36 with Podman 4.0.2 and Debian "Sid" (Siduction) with Podman 3.4.4. Apart from the Podman installation steps, the instructions are identical.

During testing, I ran Fedora 36 in a Proxmox container, whereas Siduction was installed on a laptop. To install Podman on Fedora, type:

```
sudo dnf install podman cockpit-podman
```

For Debian, use:

```
sudo apt install podman cockpit-podman
```

The Debian instructions should work on Debian Stable and its derivatives. Using older versions of Podman sometimes results in deviations in the behavior.

For even better integration between the containers and the host, you need to additionally install the `toolbox` utility [6]. After doing so, packages can be installed in the container using DNF, USB devices can be passed through, and the host's home directory can be integrated (Figure 2).

Configuration

First, you need to create three volumes for the Nextcloud installation you want to create in the Podman container (Listing 1). A volume [7] in this context acts as a storage device that Podman creates and manages, providing the ability to move and edit data between the container and the host. You can create volumes up front with the `podman volume` command or directly when setting up the containers (Figure 3).

Next, create a new network by typing

```
podman network create nextcloud-net
```

and check its properties with

```
podman network inspect nextcloud-net
```

Now it's time to create the containers, starting with the MariaDB database. As an alternative, you could integrate PostgreSQL, whereas SQLite is not a good choice for Nextcloud. The commands and specifications for setting up the database container are specified in Listing 2.

```
[dd@fedora ~]$ podman network ls
NETWORK ID    NAME        DRIVER
2f259bab93aa  podman      bridge
[dd@fedora ~]$ podman volume create nextcloud-app
nextcloud-app
[dd@fedora ~]$ podman volume create nextcloud-data
nextcloud-data
[dd@fedora ~]$ podman volume create nextcloud-db
nextcloud-db
[dd@fedora ~]$ podman volume ls
DRIVER      VOLUME NAME
local       nextcloud-app
local       nextcloud-data
local       nextcloud-db
[dd@fedora ~]$
```

Figure 3: My example Nextcloud project consists of three volumes that can be created quickly with simple commands. The volumes enhance the containers' flexibility by allowing data to be moved and edited between the containers and the host.

The `podman run \` command pops up an interactive shell where you can define the database properties [8]. Make sure you select and remember the `<DB-User-Password>` and the `<DB-Root-Password>`; you will need these later on. You can check whether this all worked by typing `podman container ls`, which shows you the running container.

The next step is to roll out Nextcloud. The same principle applies as shown in Listing 3. Again, make sure you run the `<DB-User-Password>` from the DB container and replace the `<NC-Admin>` and the `<NC-Password>` variables.

After setting up the framework, call *localhost:8080* in your web browser. Nextcloud 23 will say hello, and you can then

Listing 1: Creating Volumes

```
$ podman volume create nextcloud-app
$ podman volume create nextcloud-data
$ podman volume create nextcloud-db
```

Listing 2: MariaDB in a Container

```
podman run --detach \
  --env MYSQL_DATABASE=nextcloud \
  --env MYSQL_USER=nextcloud \
  --env MYSQL_PASSWORD=<DB-User-Password> \
  --env MYSQL_ROOT_PASSWORD=<DB-Root-Password> \
  --volume nextcloud-db:/var/lib/mysql \
  --network nextcloud-net \
  --restart on-failure \
  --name nextcloud-db \
  docker.io/library/mariadb:10
```

Listing 3: Rolling Out Nextcloud

```
podman run --detach \
  --env MYSQL_HOST=nextcloud-db.dns.podman \
  --env MYSQL_DATABASE=nextcloud \
  --env MYSQL_USER=nextcloud \
  --env MYSQL_PASSWORD=DB-User-Password \
  --env NEXTCLOUD_ADMIN_USER=<NC-Admin> \
  --env NEXTCLOUD_ADMIN_PASSWORD=<NC-Password> \
  --volume nextcloud-app:/var/www/html \
  --volume nextcloud-data:/var/www/html/data \
  --network nextcloud-net \
  --restart on-failure \
  --name nextcloud \
  --publish 8080:80 \
  docker.io/library/nextcloud:latest
```

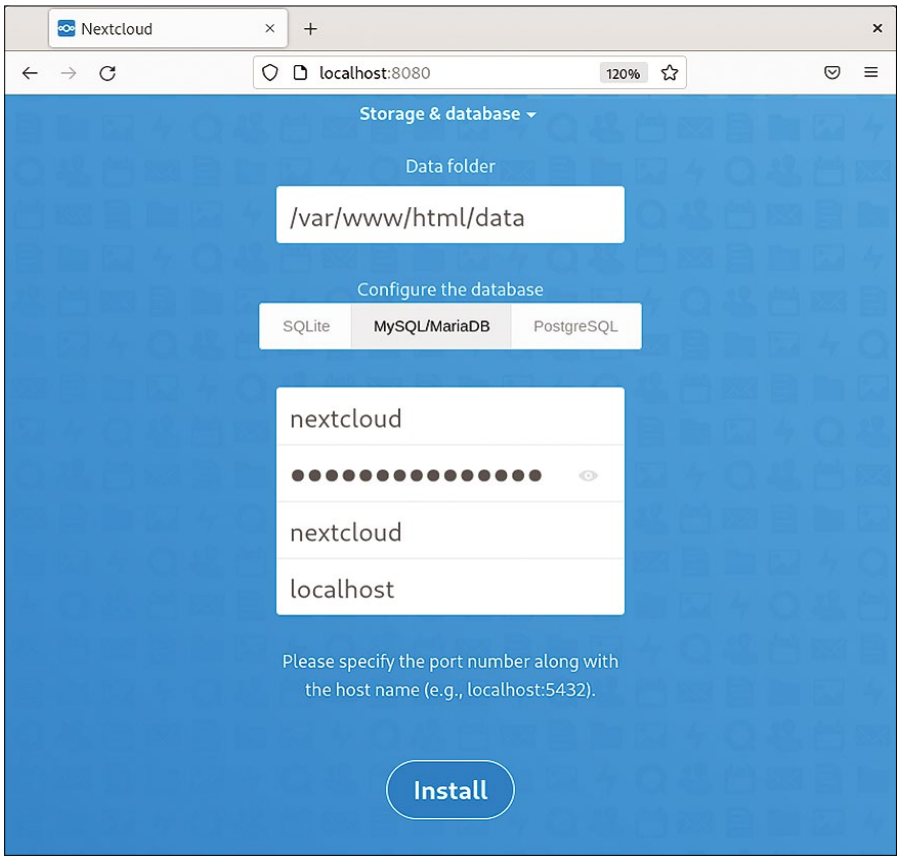


Figure 4: If everything goes according to plan, you can complete the Nextcloud installation in your browser. It's hard to imagine rolling out Nextcloud faster.

continue the installation in the GUI. You can also write the `env` parameters to a file and then include it by typing:

```
--env-file /<path>/<to>/<file>
```

Setup

The setup described here works fine for using Nextcloud on your home network. In this simple scenario, all the containers run in a single pod. If you want to access the pod from outside your home network, you will need additional containers. TLS encryption for security purposes can be implemented via the Traefik proxy, using HTTPD, NGINX, or Caddy, among others.

If you include Podman as a systemd service, you can simplify image updates via `podman auto-update` [9] by additionally specifying `label=io.containers.autoupdate=image` and using

```
podman generate systemd --new
```

Listing 4: Podman Configuration

```
/usr/share/containers/containers.conf
/etc/containers/containers.conf
$HOME/.config/containers/containers.conf
```

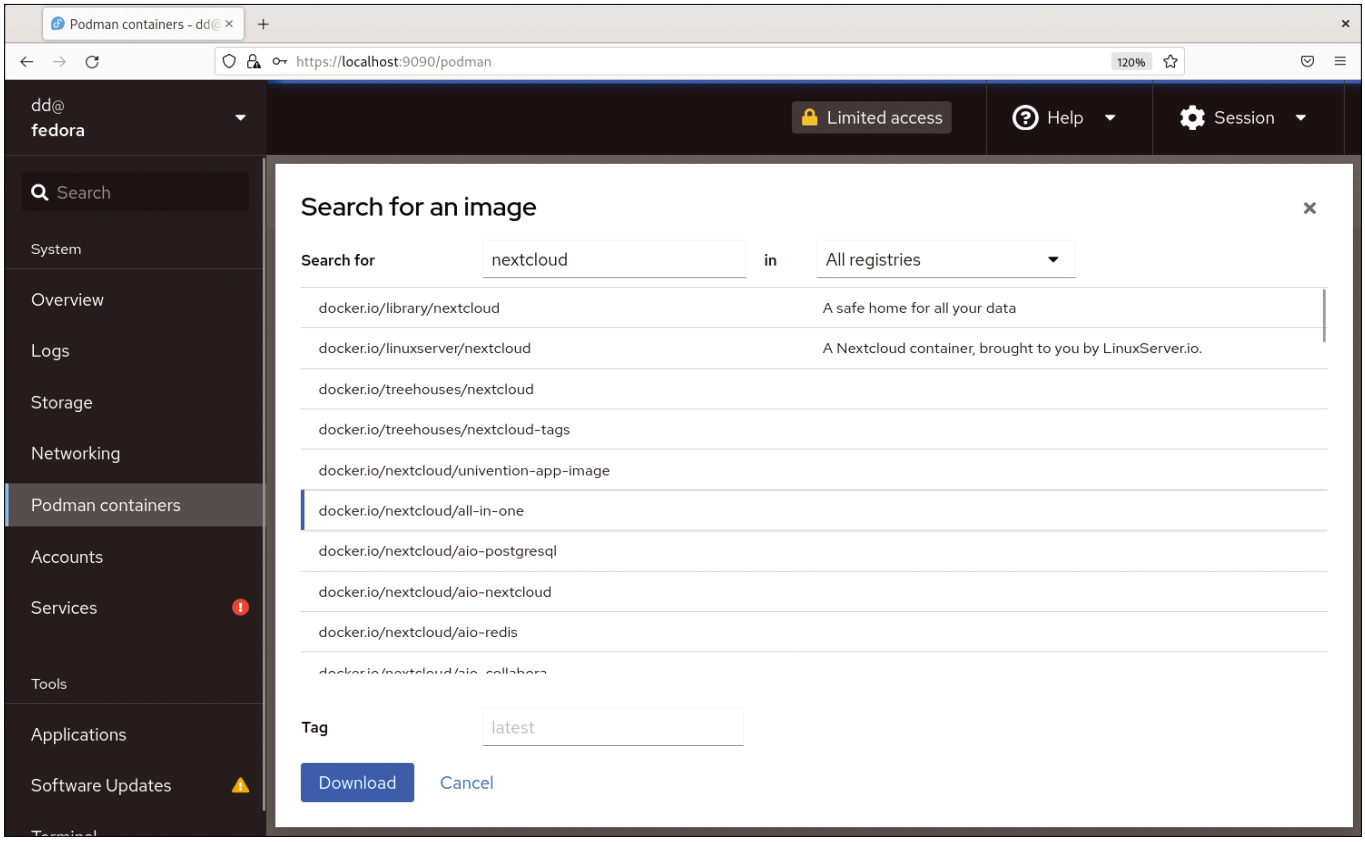


Figure 5: Embedding Podman in Cockpit makes it easier to set up the containers. This starts with downloading the required image from the Docker registry.

when creating a container. After completing the preliminary work, call Nextcloud in your browser on *localhost:8080* and complete the installation there in the usual way (Figure 4). The paths to the three most important configuration files for Podman can be found in Listing 4.

As mentioned, you can manage Podman at the command line or via the Cockpit administration interface, which you install with the *cockpit-podman*

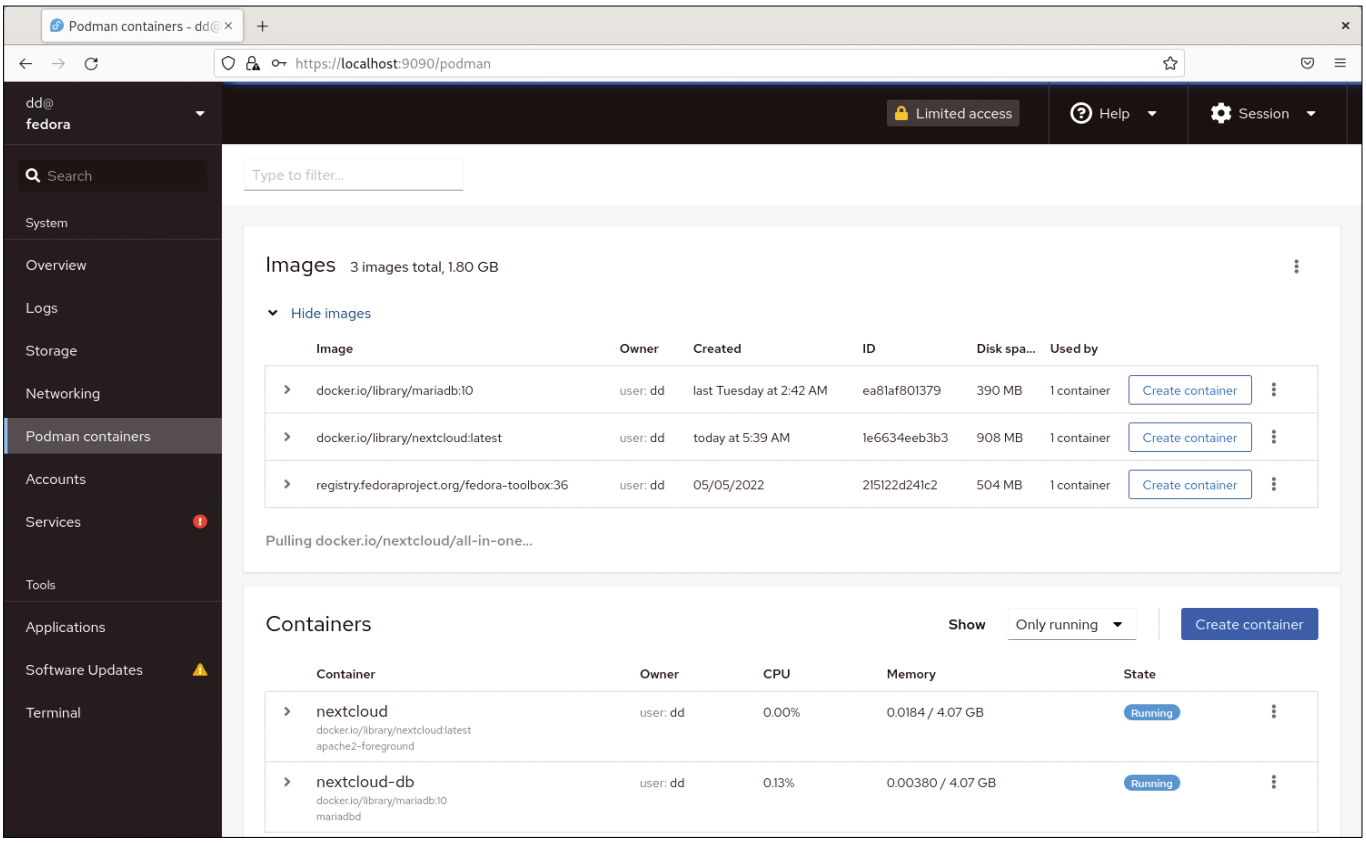


Figure 6: After downloading the images, create the containers that will host the applications.

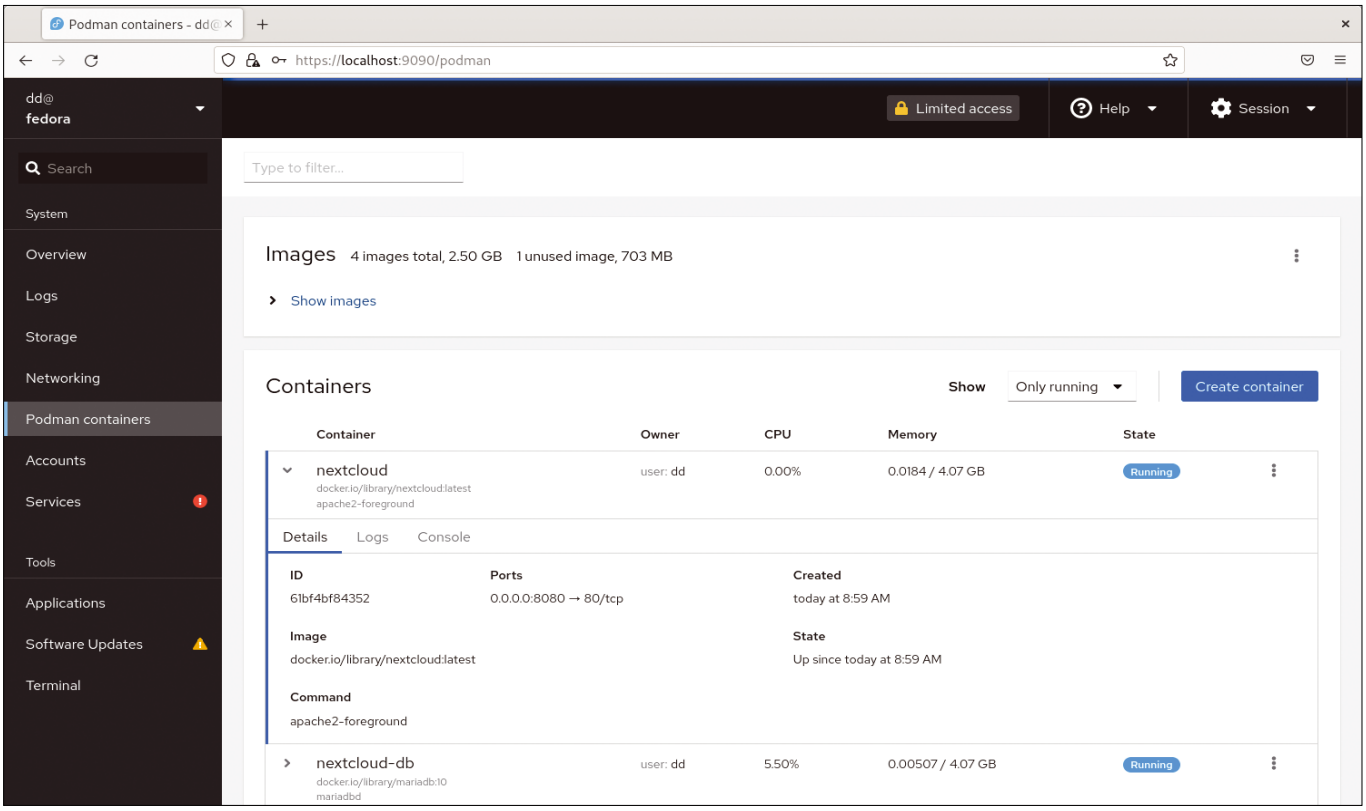


Figure 7: You can check the status of the containers against specified criteria at any time.

package. Then call Cockpit on *localhost:9090/podman* in your browser (Figure 5). The web interface gives you the ability to inspect (Figure 6) and manage (Figure 7) your images and containers.

Conclusions

The Podman man page [10] offers a useful initial overview of Podman's features, while the Podman website [1] provides a variety of references for further research. In my opinion, you can learn the Podman basics more quickly than Docker. If you already have experience with Docker, you will certainly find the transition to Podman easy. To jog muscle memory in this case, enter

```
alias docker='podman'
```

in your `.bashrc` to continue using Docker commands such as `pull`, `push`, `build`, `commit`, `tag`, and more.

I did not find any drawbacks compared to Docker during testing. Podman's biggest benefit is improved security due to eliminating the need to be root. In addition, simple administration through integration with Cockpit is a plus. Podman is well on its way to replacing Docker, although there are still some teething issues to be aware of, especially when you are working with remote machines. The simple introduction described here is a good starting place if you are looking to expand your own container skills. ■■■

Info

[1] Podman: *https://docs.podman.io/en/latest/*

[2] Namespaces: *https://en.wikipedia.org/wiki/Linux_namespaces#User_ID_(user)*

[3] Podman as a systemd service: *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/managing_containers/running_containers_as_systemd_services_with_podman*

[4] Monitoring Podman with Cockpit: *https://www.tutorialworks.com/podman-monitoring-cockpit-fedora/*

[5] Buildah: *https://buildah.io*

[6] toolbox: *https://docs.fedoraproject.org/en-US/fedora-silverblue/toolbox/*

[7] Podman volumes: *https://blog.christophersmart.com/2021/01/31/volumes-and-rootless-podman/*

[8] `podman run`: *https://docs.podman.io/en/latest/markdown/podman-run.1.html*

[9] Podman auto-update: *https://github.com/containers/podman/blob/v2.0/docs/source/markdown/podman-auto-update.1.md*

[10] Podman man page: *https://manpages.debian.org/unstable/podman/podman.1.en.html*

■■■

Photo location guessing game in Go

Treasure Hunt

A geolocation guessing game based on the popular Wordle evaluates a player's guesses based on the distance from and direction to the target location. Mike Schilli turns this concept into a desktop game in Go using the photos from his private collection. *By Mike Schilli*

After the resounding success of the Wordle [1] word guessing game, it didn't take long for the first look-alikes to rear their heads. One of the best is the entertaining Worldle [2] geography game, where the goal is to guess a country based on its shape. After each unsuccessful guess, Worldle helps the player with information about how far the guessed country's location is from the target and in which direction the target country lies.

Not recognizing the outline of the country in Figure 1, a player's first guess is the Principality of Liechtenstein. The Worldle server promptly tells the player that the target location is 5,371 kilometers away from and to the east of this tiny European state. The player's second guess is Belarus, but according to the Worldle server, from Belarus you'd have to travel 4,203 kilometers southeast to get to the target. Mongolia, the third attempt, overshoots the mark, because from there you'd have to go 3,476 kilometers to the southwest to arrive at the secret destination.

Slowly but surely, the player realizes that the secret country must be somewhere near India. And then, on the fourth try, Pakistan turns out to be correct! Worldle is a lot of fun, with a new country posted for guessing every day.

Private Snapshots

Now, instead of guessing countries, I thought it would be fun to randomly select a handful of photos from my vast cell phone photo collection, which has grown wildly over the years. Analyzing each photo's GPS data, the game engine makes sure the photos in one round were taken a great distance from one another. Initially, the computer selects a random photo from the set as a solution. It keeps this a secret, of course, and then shows the player a randomly selected photo, along with details of how many kilometers lie between the place where the random photo was taken and the target location, along with the compass direction in which the player has to move across the world to get there.

Armed with this information, the player now has to guess which picture from the remaining selection is the secret photo. The player clicks on the

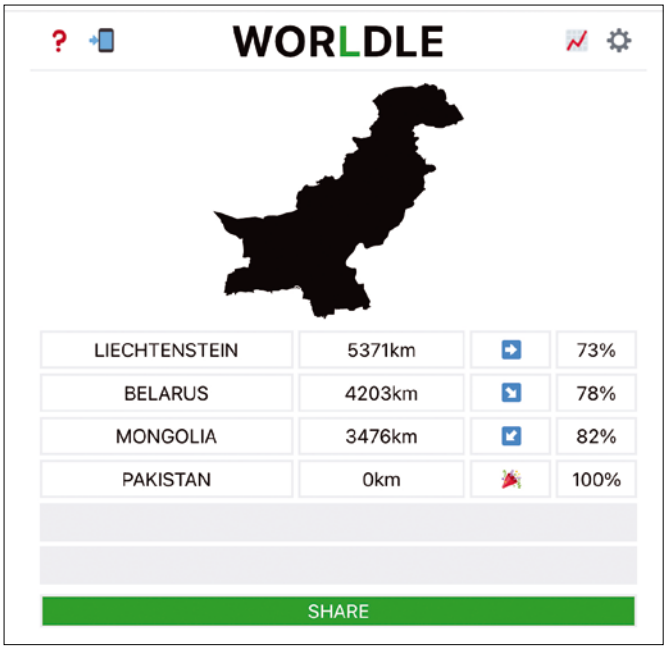


Figure 1: The original Worldle geography guessing game.

Lead Image courtesy of Mike Schilli

Author

Mike Schilli works as a software engineer in the San Francisco Bay Area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at *mschilli@perlmeister.com* he will gladly answer any questions.

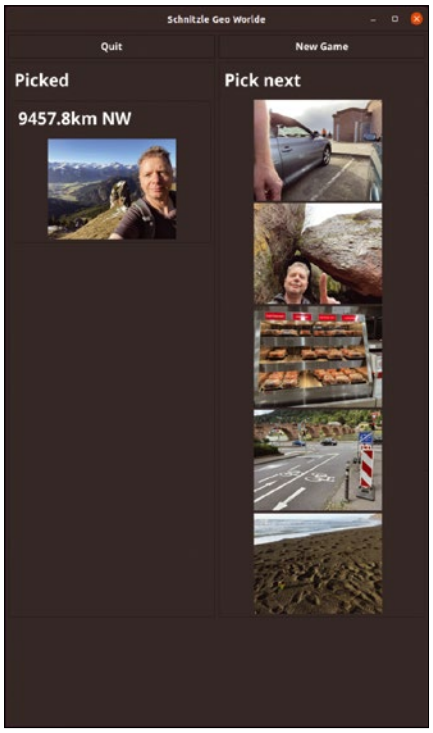


Figure 2: Starting point: Schnitzle selects a photo that is 9,457.8 kilometers from the target.

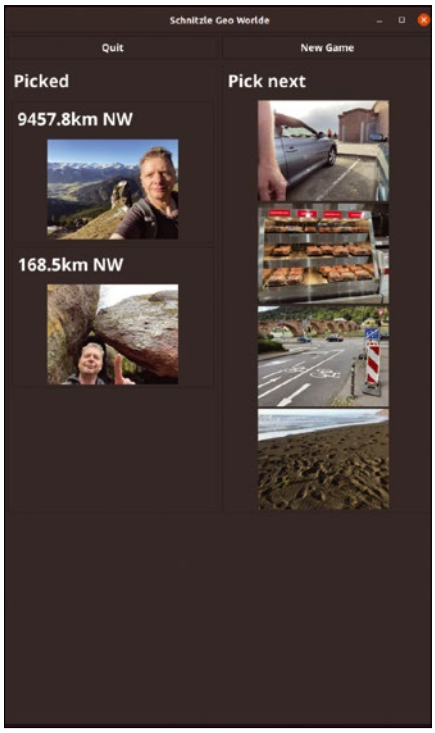


Figure 3: The player clicks on a photo of Pinnacles National Park, which is still 168.5 kilometers from the target.

suspected match and is given some feedback on the guess, again with the distance and compass direction leading to the secret location. The goal of the game is to find the solution in as few rounds as possible – a kind of treasure hunt, if you like. As a nod to the extra consonants in Wordle, my program goes by the name of Schnitzle. There are enough photos to choose from on my cell phone, and a random generator ensures that the game always selects new pictures, so it never gets boring.

And … Action!

Figure 2 shows Schnitzle in action. As a starting image, the computer has selected a snapshot depicting yours truly, hiking in the Bavarian Alps. According to the clues, the target is 9,457.8 kilometers to the northwest (NW) from the starting image. It seems highly likely that the secret photo was taken somewhere in North America! From the selection on the right, the player then clicks on the photo of Pinnacles National Park in California (Figure 3). Schnitzle reveals that the target is 168.5 kilometers in a northwest direction from this guess. What's north of the Pinnacles? Probably the San Francisco Bay Area, where I live!

In Figure 4, the player then clicks on the photo of the parking lot at the beach in Pacifica, where I often go surfing. But you still have to travel 10.2 kilometers from the beach in a northeast direction (NE) to the location of the photo the game selected. If you know the area, you can probably guess: The destination must be somewhere in the suburbs of South San Francisco, where the giant Costco supermarket is located. In fact, that shelf filled with rotisserie chicken is the solution, as the ** WINNER ** message indicates (Figure 5). There are still two unclicked photos in the right-hand column, showing a bridge in Heidelberg, Germany, and one showing the sand at Esplanade Beach in the Bay Area.

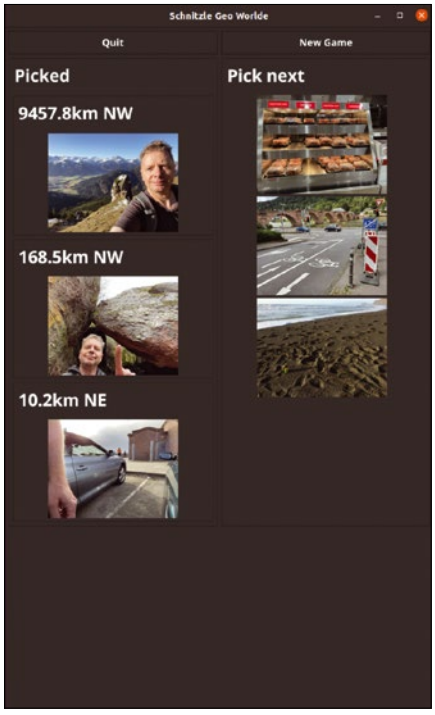


Figure 4: The Pacifica State Beach parking lot is still 10 kilometers from the target.

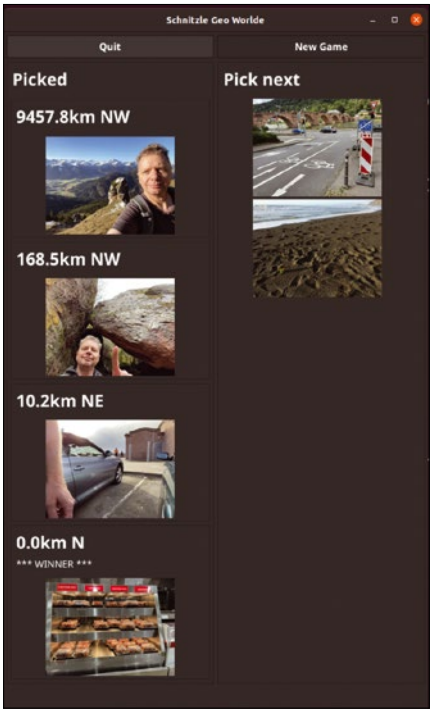


Figure 5: The solution: a shelf with rotisserie chicken at the Costco in South San Francisco.

Seek and Ye Shall Find

So how does the game work as a Go program? To sift through the entire photo collection downloaded from my cell phone to my hard disk takes some time, even though it is on a fast SSD. That's why the `finder.go` helper program in Listing 1 plumbs the depths of the cell phone photo directory set in line 18, analyzing each JPEG image found there and reading its GPS data, if available, to cache it for later.

The program feeds the results into a table in an SQLite database so that the game program can quickly select new images in each round later on, without having to comb through entire

filesystem trees on every round. You can create the required empty SQLite database with the required table that assigns GPS data to file names in next to no time with a shell command such as the one in Figure 6.

```
$ sqlite3 photos.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> CREATE TABLE files (path text, lat real, long real);
sqlite> ^D
```

Figure 6: An empty photo database, generated using the `sqlite3` client.

```
$ sqlite3 photos.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> SELECT COUNT(*) FROM files;
4162
sqlite> SELECT * FROM files ORDER BY RANDOM() LIMIT 5;
/Users/mschilli/iphone/IMG_9729.JPG|37.7695194444444|-122.410219444444
/Users/mschilli/iphone/IMG_6101.JPG|37.7421361111111|-122.436858333333
/Users/mschilli/iphone/IMG_E6814.JPG|37.8522|-122.425977777778
/Users/mschilli/iphone/IMG_0186.JPG|37.9521055555556|-122.42685
/Users/mschilli/iphone/IMG_8616.JPG|37.2432416666667|-122.435255555556
sqlite>
```

Figure 7: After running `finder`, there are 4,162 images with GPS coordinates in the database.

Before the game can begin, the program from Listing 1 needs to run once, compiled by typing:

```
go build finder.go
```

The program uses two libraries (*go-sqlite3* and *goexif2*) from GitHub. One drives the flat-file database, and the other reads the GPS headers from the JPEG photos.

To make the Go compiler do this without any complaints, first type

```
go mod init finder; go mod tidy
```

to specify a Go module to parse the libraries included in the source code, fetch them from GitHub if needed, and define their versions. When this is done, the `go build` command produces a static binary `finder` including all the compiled libraries.

Listing 1: finder.go

```
01 package main
02
03 import (
04   "database/sql"
05   "fmt"
06   _ "github.com/mattn/go-sqlite3"
07   exif "github.com/xor-gate/goexif2/exif"
08   "os"
09   "path/filepath"
10   rex "regexp"
11 )
12
13 type Walker struct {
14   Db *sql.DB
15 }
16
17 func main() {
18   searchPath := "photos"
19
20   db, err := sql.Open("sqlite3", "photos.db")
21   w := &Walker{ Db: db }
22   err = filepath.Walk(searchPath, w.Visit)
23   panicOnErr(err)
24
25   db.Close()
26 }
27
28 func (w *Walker) Visit(path string,
29   f os.FileInfo, err error) error {
30   jpgMatch := rex.MustCompile("(?i)JPG$")
31   match := jpgMatch.MatchString(path)
32   if !match {
33     return nil
34   }
35
36   lat, long, err := GeoPos(path)
37   panicOnErr(err)
38
39   stmt, err := w.Db.Prepare("INSERT INTO files
   VALUES(?,?,?)")
40   panicOnErr(err)
41   fmt.Printf("File: %s %.2f/%.2f\n", path, lat, long)
42   _, err = stmt.Exec(path, lat, long)
43   panicOnErr(err)
44   return nil
45 }
46
47 func GeoPos(path string) (float64,
48   float64, error) {
49   f, err := os.Open(path)
50   if err != nil {
51     return 0, 0, err
52   }
53
54   x, err := exif.Decode(f)
55   if err != nil {
56     return 0, 0, err
57   }
58
59   lat, long, err := x.LatLong()
60   if err != nil {
61     return 0, 0, err
62   }
63
64   return lat, long, nil
65 }
```

As shown in Figure 7, the `finder` utility from Listing 1 reads the 4,000 or so files in my phone folder in about 30 seconds and adds the photos' metadata into the `files` table in the SQLite `photos.db` flat-file database.

The call to the `Walk` function in line 22 of Listing 1 receives the `w.Visit` callback defined in line 28. The file browser calls this function for every file it finds. It always drags a type `Walker` data structure along with it as a receiver, which means that it can immediately access the `db` handle of the SQLite database opened previously.

For each file found, line 31 checks whether the file has a `.jpg` extension (upper- or lowercase) and then runs the `GeoPos()` function from line 47 to load the photo's Exif data. This will ideally include the longitude and latitude of the location where the photo was taken as floating-point numbers.

Line 39 feeds the path and GPS data into the database table with an `INSERT` statement in typical SQL syntax. Later, the main `schnitzle` program can pick up the image location and metadata from the database, when it is looking for new snapshots for a new game.

Spoiled for Choice

It is not particularly difficult to select a dozen pictures at random from a photo collection of several thousand images. What is trickier is to make sure that the locations of the photos in one round of the game are not too close to each other. Many cell phone photos are taken at home, and the prospect of navigating inch by inch between the living room, balcony, and kitchen is not exactly thrilling.

Instead, I wanted the algorithm to randomly select images, while ensuring that new exciting game scenarios are created in each round, by always presenting a good mix of different regions. The cell phone photo app's geographical view in Figure 8 illustrates how the shots can be assigned to bundled hotspots based on the GPS data. The algorithm then only ever selects one image from a given hotspot.

The k-means algorithm [3] is a massive help here; k-means is an artificial intelligence [4] method, applied to cluster information in unsupervised learning [5] (Figure 9). From a set of more or less randomly distributed points in a two- or multidimensional space, k-means determines the centers of the clusters. In the Schnitzle game, these would be locations where many cell phone photos were taken, such as at home or at various vacation destinations. The algorithm then randomly selects one image only from each of these clusters. This ensures that there will be a meaningful distance between the locations where the individual pictures were taken for each round of the game.

The `photoSet()` function starting in line 18 of Listing 2 has the task of delivering an array slice of six photos of the

Figure 8: Geo-clustering of photos on my cell phone.

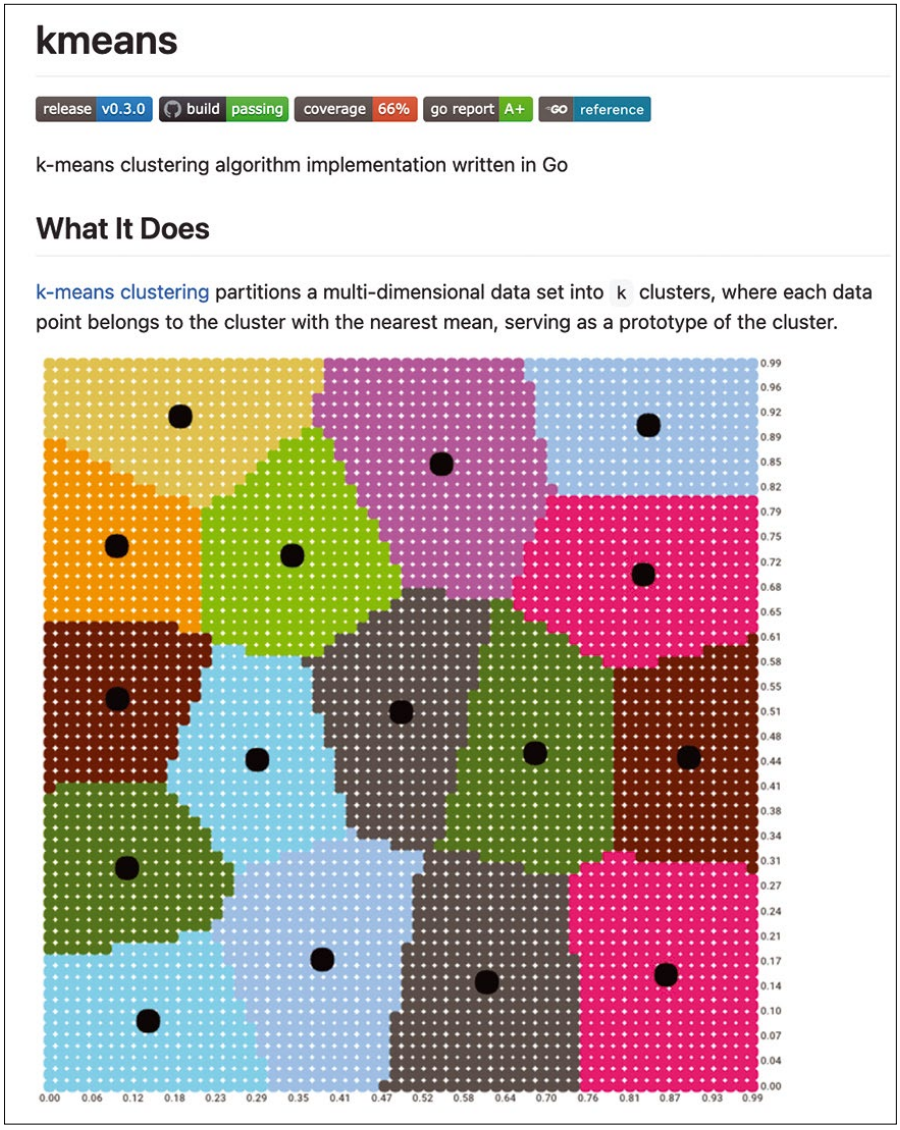


Figure 9: The *kmeans* library for Go on GitHub.

`Photo` type for a new game. Line 12 defines the `Photo` data structure, which contains `Path` for the path to the image file for one thing. On top of that, it holds the geo-coordinates read from the Exif information as `Lng` (longitude)and `Lat` (latitude), both represented as 64-bit floating-point numbers.

To do this, `photoSet()` connects to the previously created SQLite database `photos.db` starting in line 19 and runs the `SELECT` query starting in line 23 to sort through all the previously read photo files along with their GPS coordinates. After the `for` loop, which starts in line 36 and processes all the table tuples it finds, all records now exist in an array of `clusters.Observations` type elements, ready to be processed by the *kmeans* package from GitHub [6].

The call to `km.Partition()` then assigns the GPS coordinates to 10 different clusters. From these, line 60 then discards tiny clusters with fewer than three entries. This prevents the same photos from appearing time and time again in each game, not giving the algorithm a chance to deliver variety in the form of random selections from a specific cluster. The algorithm selects a maximum of six (`maxClusters`) photos from the remaining clusters and then puts them in random order with the shuffle function from the *rand* package.

Because the *kmeans* cluster library from GitHub is not familiar with photo

Listing 2: photoset.go

```
01 package main
02
03 import (
04   "database/sql"
05   "fmt"
06   _ "github.com/mattn/go-sqlite3"
07   "github.com/muesli/clusters"
08   "github.com/muesli/kmeans"
09   "math/rand"
10 )
11
12 type Photo struct {
13   Path string
14   Lat  float64
15   Lng  float64
16 }
17
18 func photoSet() ([]Photo, error) {
19   db, err := sql.Open("sqlite3", "photos.db")
20   panicOnErr(err)
21   photos := []Photo{}
22
23   query := fmt.Sprintf("SELECT path, lat, long FROM
   files")
24   stmt, _ := db.Prepare(query)
25
26   rows, err := stmt.Query()
27   panicOnErr(err)
28
29   var d clusters.Observations
30   lookup := map[string]Photo{}
31
32   keyfmt := func(lat, lng float64) string {
33     return fmt.Sprintf("%f-%f", lat, lng)
34   }
35
36   for rows.Next() {
37     var path string
38     var lat, lng float64
39     err = rows.Scan(&path, &lat, &lng)
40     panicOnErr(err)
41     lookup[keyfmt(lat, lng)] = Photo{Path: path, Lat:
                                        lat, Lng: lng}
42     d = append(d, clusters.Coordinates{
43       lat,
44       lng,
45     })
46   }
47
48   db.Close()
49
50   maxClusters := 6
51   km := kmeans.New()
52   clusters, err := km.Partition(d, 10)
53   panicOnErr(err)
54
55   rand.Shuffle(len(clusters), func(i, j int) {
56     clusters[i], clusters[j] = clusters[j], clusters[i]
57   })
58
59   for _, c := range clusters {
60     if len(c.Observations) < 3 {
61       continue
62     }
63     rndIdx := rand.Intn(len(c.Observations))
64     coords := c.Observations[rndIdx].Coordinates()
65     key := keyfmt(coords[0], coords[1])
66     photo := lookup[key]
67     photos = append(photos, photo)
68     if len(photos) == maxClusters {
69       break
70     }
71   }
72   return photos, nil
73 }
74
75 func randPickExcept(pick []Photo, notIdx int) int {
76   idx := rand.Intn(len(pick)-1) + 1
77   if idx == notIdx {
78     idx = 0
79   }
80   return idx
81 }
```

collections, but can only sort points with X/Y coordinates, line 41 creates a `lookup` hash map. It maps the longitude and latitude of the photos to the JPEG images on the disk. When the algorithm comes back with the coordinates of a desired image later on, the program can find, load, and display the associated image.

Controlled Randomness

From the representatives of all the chosen clusters, the Schnitzle game initially needs to select a secret target picture for the player to guess. It then opens the game with a random starting image, but it would not be a good idea to pick the secret image, even by accident! The `rand.Intn(len(<N>))` standard solution in Go delivers randomly and equally distributed index positions between `0` (inclusive) and `len(<N>)` (exclusive), thus picking purely random elements from the array.

The `randPickExcept()` function starting in line 75 of Listing 2 now picks a random element from the array passed into it, without ever revealing the element that resides in the `notIdx` space. This is accomplished by the algorithm only selecting the elements in index positions `1..<N>` from the elements in the index range `0..<N>`, neglecting the first image in the list. And, if the choice happens to fall on the forbidden `notIdx` index position, the function simply delivers the `0` item, which was previously excluded from the pick, as a replacement. This way, all photos, except the secret one, have an equal probability of being picked as a starting point.

Slimming Down

Listing 3 helps to load the scaled down cell phone photos into the GUI. One difficulty here is that many cell phones have the bad habit of storing image pixels in a rotated orientation when taken and noting in the header that the image needs to be rotated through 90 or 180 degrees for display purposes [7].

This quirky behavior is handled by the *imageorient* package that Listing 3 pulls in from GitHub in line 5, auto-rotating each image before it is handed to the GUI for display. Also, nobody really wants to move massive photos around on the screen. Instead, the *nfnt/resize* package (also from GitHub) creates handy thumbnails from the large photos with the help of the `Thumbnail()` function in line 26.

Listing 4 computes the distance between the photo shoot locations of two image files and the angle from 0 to 360 degrees at which you would have to start walking to get from A to B. As the earth isn't a flat surface, calculating these numbers isn't as easy as on a two-dimensional map, but the formulas dealing with the required 3D geometry are not too complicated and already available online [8]. In line 8, the `hike()` function takes the longitude (`lng<N>`) and latitude (`lat<N>`) from the GPS data of two photos and taps into the functions of the *golang-geo* library, including `GreatCircleDistance()` and `BearingTo()` to determine the distance and bearing to travel from one photo to the other.

To convert the route's `bearing`, available as a floating-point number ranging from 0 to 360 degrees into a compass direction such as north or northeast, line 16 divides the angle by 45, rounds the result to the nearest integer, and then accesses the array slice in line 15 with that index. Index `0` is `N` for north, `1` is `NE` for northeast, and so on. If the index drops below `0`, which can happen with negative angles, line 19 simply adds the

Listing 3: image.go

```
01 package main
02
03 import (
04   "fyne.io/fyne/v2/canvas"
05   "github.com/disintegration/imageorient"
06   "github.com/nfnt/resize"
07   "image"
08   "os"
09 )
10
11 const DspWidth = 300
12 const DspHeight = 150
13
14 func dispDim(w, h int) (dw, dh int) {
15   if w > h {
16     // landscape
17     return DspWidth, DspHeight
18   }
19   // portrait
20   return DspHeight, DspWidth
21 }
22
23 func scaleImage(img image.Image) image.Image {
24   dw, dh := dispDim(img.Bounds().Max.X,
25     img.Bounds().Max.Y)
26   return resize.Thumbnail(uint(dw),
27     uint(dh), img, resize.Lanczos3)
28 }
29
30 func showImage(img *canvas.Image, path string) {
31   nimg := loadImage(path)
32   img.Image = nimg.Image
33
34   img.FillMode = canvas.ImageFillOriginal
35   img.Refresh()
36 }
37
38 func loadImage(path string) *canvas.Image {
39   f, err := os.Open(path)
40   panicOnErr(err)
41   defer f.Close()
42   raw, _, err := imageorient.Decode(f)
43   panicOnErr(err)
44
45   img := canvas.NewImageFromResource(nil)
46   img.Image = scaleImage(raw)
47
48   return img
49 }
```

Listing 4: gps.go

```
01 package main
02
03 import (
04   geo "github.com/kellydunn/golang-geo"
05   "math"
06 )
07
08 func hike(lat1, lng1, lat2, lng2 float64) (float64,
            string, error) {
09   p1 := geo.NewPoint(lat1, lng1)
10   p2 := geo.NewPoint(lat2, lng2)
11
12   bearing := p1.BearingTo(p2)
13   dist := p1.GreatCircleDistance(p2)
14
15   names := []string{"N", "NE", "E", "SE", "S", "SW",
                       "W", "NW", "N"}
16   idx := int(math.Round(bearing / 45.0))
17
18   if idx < 0 {
19     idx = idx + len(names)
20   }
21
22   return dist, names[idx], nil
23 }
```

length of the array slice to arrive at an index that addresses the array slice from the end instead.

Widget with Something Special on Top

The `schnitzle` GUI application relies on the Fyne framework to whip up a native-looking graphical application on the desktop with plain vanilla Go code. Previous articles in this column [9] have plumbed the depths of this excellent tool [10] many a time.

Fyne comes with a whole range of label, listbox, button, and other widgets out of the box, but it can't hope to cover every single special case conjured up by creative software engineers. For example, in the Schnitzle game, the guesser clicks on a photo in the right pane, to have the game move it to the left. Photos don't normally take clicks, but button widgets do. And they, in turn, define callbacks to perform the intended action in case of an alert.

With just a little code, you can quickly program extensions in Fyne to add the desired, albeit nonstandard, behavior. Listing 5 defines a new widget type named `clickImage()` starting in line 13. It is composed from a canvas object with a thumbnail image and takes a callback that it invokes when the user clicks on the photo with the mouse.

Thanks to Go's built-in inheritance mechanism for structures, `widget.BaseWidget` on line 14 derives the structure from Fyne's base widget, thus providing it with the display, shrink, or hide functions common to all widgets. In addition, the add-on widget needs to call the `ExtendBaseWidget()` function in the

Listing 5: gui.go

```
01 package main
02
03 import (
04   "fyne.io/fyne/v2"
05   "fyne.io/fyne/v2/canvas"
06   "fyne.io/fyne/v2/container"
07   "fyne.io/fyne/v2/widget"
08   "math/rand"
09   "os"
10   "time"
11 )
12
13 type clickImage struct {
14   widget.BaseWidget
15   image *canvas.Image
16   cb    func()
17 }
18
19 func newClickImage(img *canvas.Image, cb func())
   *clickImage {
20   ci := &clickImage{}
21   ci.ExtendBaseWidget(ci)
22   ci.image = img
23   ci.cb = cb
24   return ci
25 }
26
27 func (t *clickImage) CreateRenderer() fyne.WidgetRenderer {
28   return widget.NewSimpleRenderer(t.image)
29 }
30
31 func (t *clickImage) Tapped(_ *fyne.PointEvent) {
32   t.cb()
33 }
34
35 func makeUI(w fyne.Window, p fyne.Preferences) {
36   rand.Seed(time.Now().UnixNano())
37
38   var leftCard *widget.Card
39   var rightCard *widget.Card
40
41   quit := widget.NewButton("Quit", func() {
42     os.Exit(0)
43   })
44
45   var restart *widget.Button
46
47   reload := func() {
48     leftCard, rightCard = makeGame(p)
49     vbox := container.NewVBox(
50       container.NewGridWithColumns(2, quit, restart),
51       container.NewGridWithColumns(2, leftCard, rightCard),
52     )
53     w.SetContent(vbox)
54     canvas.Refresh(vbox)
55   }
56
57   restart = widget.NewButton("New Game", func() {
58     reload()
59   })
60
61   reload()
62 }
```

constructor later on in line 21 to use all of the GUI's features.

One more thing: The GUI still doesn't know how to display the new widget on the screen. This is why the `CreateRenderer()` function starting in line 27 returns an object of the `NewSimpleRenderer` type to the GUI, when the function is being called with the image as its only parameter.

In order for the photo widgets in Schnitzle to respond to mouse clicks, their `newClickImage()` constructor, defined in line 19, takes a callback that the widget later calls on mouse click events. Line 23

Listing 6: schnitzle.go

```
01 package main
02
03 import (
04   "fmt"
05   "fyne.io/fyne/v2"
06   "fyne.io/fyne/v2/app"
07   "fyne.io/fyne/v2/canvas"
08   "fyne.io/fyne/v2/container"
09   "fyne.io/fyne/v2/widget"
10   "math/rand"
11 )
12
13 func makeGame(p fyne.Preferences) (*widget.Card, *widget.
                                     Card) {
14   pickCh := make(chan int)
15   done := false
16
17   photos, err := photoSet()
18   panicOnErr(err)
19
20   photosRight := make([]Photo, len(photos))
21   copy(photosRight, photos)
22
23   pool := []fyne.CanvasObject{}
24
25   for i, photo := range photosRight {
26     idx := i
27     img := canvas.NewImageFromResource(nil)
28     img.SetMinSize(fyne.NewSize(DspWidth, DspHeight))
29     clkimg := newClickImage(img, func() {
30       if !done {
31         pickCh <- idx
32       }
33     })
34
35     pool = append(pool, clkimg)
36     showImage(img, photo.Path)
37   }
38
39   solutionIdx := rand.Intn(len(photos))
40   solution := photos[solutionIdx]
41
42   left := container.NewVBox()
43   right := container.NewVBox(pool...)
44
45   go func() {
46     for {
47       select {
48       case i := <-pickCh:
49         photo := photos[i]
50         dist, bearing, err := hike(photo.Lat, photo.Lng,
            solution.Lat, solution.Lng)
51         panicOnErr(err)
52
53         if photo.Path == solution.Path {
54           done = true
55         }
56
57         subText := ""
58         if done == true {
59           subText = " * WINNER  *"
60         }
61
62         card := widget.NewCard(fmt.Sprintf("%.1fkm %s",
            dist, bearing), subText, pool[i])
63         left.Add(card)
64         canvas.Refresh(left)
65
66         pool[i] = widget.NewLabel("")
67         pool[i].Hide()
68
69         if done == true {
70           return
71         }
72       }
73     }
74   }()
75
76   first := randPickExcept(photos, solutionIdx)
77   pickCh <- first
78   return widget.NewCard("Picked", "", left),
79     widget.NewCard("Pick next", "", right)
80 }
81
82 func panicOnErr(err error) {
83   if err != nil {
84     panic(err)
85   }
86 }
87
88 func main() {
89   a := app.NewWithID("com.example.schnitzle")
90   w := a.NewWindow("Schnitzle Geo Worlde")
91
92   pref := a.Preferences()
93   makeUI(w, pref)
94   w.ShowAndRun()
95 }
```

assigns this function to the `cb` instance variable. Later on, the `Tapped()` function (defined in line 31 and called by the GUI) simply triggers the previously defined callback in line 32, whenever the player clicks on the particular widget. There you have it: a new GUI element controlled by a clickable photo, which behaves in a similar way to a button widget.

Like the *Quit* button starting in line 41, which uses its callback to announce the end of the game when a button is pressed via `os.Exit(0)`, Listing 6 can – thanks to the new extension – create a new clickable photo in line 29. In its callback, it sends the selected index to the `pickCh` channel. At the other end of the pipe, the game apparatus picks up the index and triggers the animation, which moves the photo from the right to the left pane.

The rest of Listing 5 is devoted to arranging the widgets shown in the game with `makeUI`. The central function `reload()` loads a new game at the initial program start and whenever *New Game* is pressed.

Index Cards as a Model

The graphical elements in the game window are arranged as two horizontal buttons at the top, followed by two image columns each of the `Card` type – this relies on vertical stacking with `NewVBox()`. This standard `Card` widget from the Fyne collection displays a header (optionally a subhead) and an image to illustrate it. You can think of it as being something like an index card with a title and some media content.

Listing 6 defines the `main()` function of the game and defines the individual widgets of the left and right game columns in `makeGame()`. On top of this, you have the move mechanism that kicks in when the player clicks on a photo in the right column.

The elements in the `pool` array are the photos for the right column, each displayed as a `CanvasObject`. In contrast to this, the left-hand widget column contains the photos already selected over the course of the game. They are located in what is initially an empty `left` array. Each time a photo in the `right` widget column is clicked, the callback associated with that photo sends the index of the selection to the `pickCh` channel in line 31.

Then the Go routine defined in line 45, running concurrently, uses `select` to handle the event. It computes the distance to the target image by calling `hike` in line 50 and generates a Fyne `card` with the result in line 62. The `Add()` function appends the card at the bottom of the left column in line 63 and uses `canvas.Refresh()` to make sure the GUI displays the change.

To make the clicked photo disappear from the right column, line 66 puts an empty `Label` widget in its place and then immediately whisks it away by calling `Hide()`.

At the start of the game, line 76 uses `randPickExcept()` to pick a random photo from the right column, but avoids disclosing the solution right away. Line 77 pushes the index position into the `pickCh` channel, very much like the callback of the photo widget selected by the user will do later on, setting off the same animation and moving it to the left column.

In Go, programmers have to check results, practically after each function call, to make sure an error hasn't crept in. It always comes back as an `err` variable. If it has a value of `nil`, no error occurred. Each time, the corresponding error handler requires three lines of code and takes up vast amounts of space in listings printed in magazines.

This is why line 82 simply defines a `panicOnErr()` function that executes this test in one line of code each time and, if an error occurs, aborts the program immediately by calling `panic()`. In production environments, errors are instead handled individually and often looped through further up the call stack, but page space in printed magazines is scarce and nobody wants to read monster-sized listings!

Off We Go

You can compile the whole kit and caboodle with the commands from Listing 7. The resulting `schnitzle` binary can be launched at the command line, which causes the GUI to pop up on screen.

On Linux, the Fyne GUI uses a C wrapper from Go to tap into the *libx11-dev*, *libgl1-mesa-dev*, *libxcursor-dev*, and *xorg-dev* libraries. To install the game on Ubuntu, for example, you can fetch these libraries with the command

```
sudo apt-get install
```

to ensure that `go build` in Listing 7 actually finds the required desktop underpinnings.

Listing 7: Compiling Schnitzle

```
$ go mod init schnitzle
$ go mod tidy
$ go build schnitzle.go gui.go photoset.go image.go gps.go
```

Conclusions

Schnitzle is a lot of fun; try it out! Sometimes it's surprisingly difficult to determine the compass direction in your head in order to even walk from one part of town to another, as depicted in the photos. I was often surprised by the results, even for locations that I thought I was very familiar with. The game offers hours of entertainment for the whole family when you trawl long-forgotten photos from years gone by from the depths of a massive image collection. ■■■

Info

[1] Wordle: *https://www.nytimes.com/games/wordle/index.html*

[2] Worldle: *https://worldle.teuteuf.fr/*

[3] K-means algorithm: *https://en.wikipedia.org/wiki/K-means_clustering*

[4] "Calculating Clusters with AI Methods" by Mike Schilli, *Linux Magazine*, issue 145, December 2012, p. 62

[5] Unsupervised learning: *https://en.wikipedia.org/wiki/Unsupervised_learning*

[6] kmeans library on GitHub: *https://github.com/muesli/kmeans*

[7] "Adjusting Cell Phone Photo Orientation with Go" by Mike Schilli, *Linux Magazine*, issue 257, April 2022, p. 52

[8] "Calculate distance, bearing and more between Latitude/Longitude points": *http://www.movable-type.co.uk/scripts/latlong.html*

[9] "Game Development with Go and the Fyne Framework" by Mike Schilli, *Linux Magazine*, issue 255, February 2022, p. 32

[10] "Discarding Photo Fails with Go and Fyne" by Mike Schilli, *Linux Magazine*, issue 254, January 2022, p. 40

MakerSpace

Get creative with the FLUX Beamo laser and open source software

Beam Me Up, Fluxy!

With the FLUX Beamo laser and a Raspberry Pi Board B10001, you can execute your own laser cutting projects on a wide range of materials. *By Christa and Ian Travis*

Laser cutting mainly used to be the purview of hard-headed business owners with solid financial backing and a business plan to make sure the laser paid for its upkeep. And there were hobbyists who were more interested in the arts and crafts side of laser cutting and saw the laser as a tool for cutting and engraving materials such as wood, cloth, and acrylics in the scope of their arts and crafts projects. The trouble was, many people were more than a little worried about the Heath Robinson-style, low-budget lasers they could purchase at the time. The foundations of the laser world were to be shaken though, when FLUX – an organization founded by a "group of passionate young engineers and designers," as the manufacturer itself states – launched the Beambox laser cutting machine back in 2018. FLUX quickly followed up with the Beamo, promoted as "the world's smallest laser cutting machine" the next year.

The FLUX laser family not only meant a paradigm shift in terms of laser cutting machine pricing, but also a move towards a community-driven approach. For one thing, the FLUX laser family relies on a Raspberry Pi, in the form of the Raspberry Pi Board B10001 [1], to control the machine. And the downloadable software package that lets users send their ideas (in the form of, say, PNG images or vector diagrams) to the laser runs on your choice of operating system, whether this be Windows, macOS, or Linux. Expanding on the community idea, FLUX users can get together to exchange ideas on Facebook or at regular FLUX community meetings that take place all over the world – even in Germany, where we're based. So, laser-affine readers, let's get started with unboxing and the setup on the hardware and software side.

Unboxing

The FLUX Beamo reaches its new owner in a very large and fairly heavy cardboard box. If you are worried about the state of your back, then it's a good idea to ask a friend to help you lift the beast out of the box. Once you've done this, there are a couple of tasks to complete before you get started on your first project. Besides doing the obvious things like removing the vent hose and attaching it to the duct on rear of the laser with the clamp kindly provided by the supplier, attaching the WiFi antenna, and plugging in, there is also the software setup. This no big deal: After powering on your laser cutter, select *Network* on the Start screen (Figure 1) and then *Connect to WiFi* if you are using the

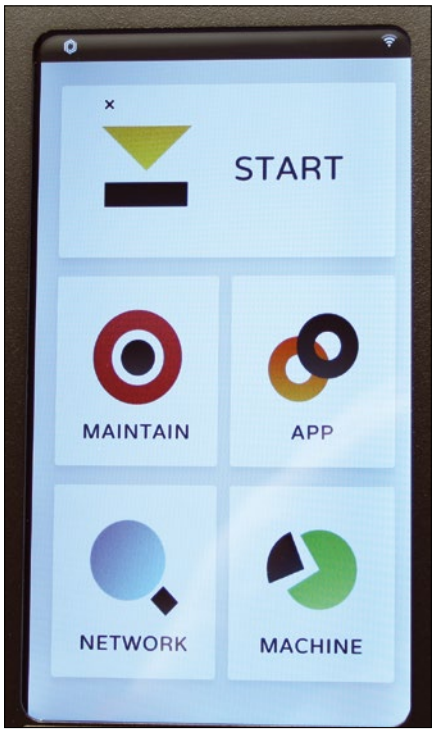


Figure 1: Flux Beamo Start screen.

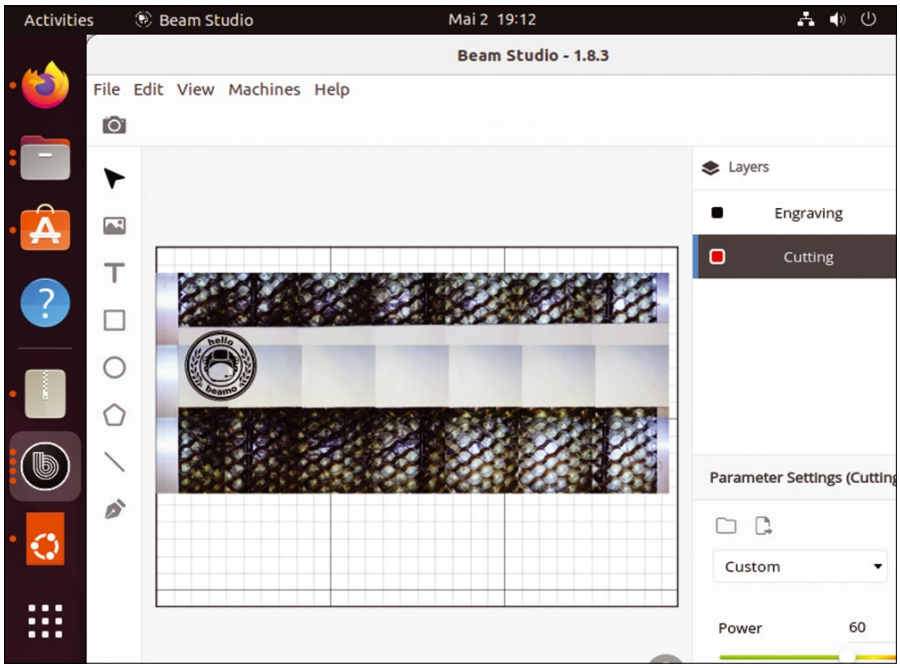


Figure 2: Beam Studio on Linux.

machine's wireless interface. Choose a WiFi network, enter the password, and let DHCP do its magic. You can alternatively use the Ethernet cable provided – this can be a good idea if the WiFi connection is too slow or unreliable. Once the machine has an IP address, make a note of the address and move on to the next stage, installing Beam Studio.

Beam Studio

If you worked with CO2 lasers prior to 2009, one of the biggest items on your wish list was probably a camera with software support that let you see what the laser was "seeing." Something that would take the guesswork out of positioning the laser head and alleviate any worries about lasering beyond the edge of the medium. Amazingly, considering its low price compared to the laser behemoths of the past, the FLUX laser family has exactly that in the form of a built-in camera and Beam Studio.

To install Beam Studio (Figure 2), go to the FLUX website [2] and select the version that matches your operating system. If you have a recent Ubuntu system, this will be *v.1.8.3 -Ubuntu-18.04 x64*. The name of the install file is `beam-studio_1.8.3_amd64.deb`, and you can install it using the Ubuntu package manager. Right-click on the downloaded file, select *Open with other application*, and then select *Software Install*. The installer prompts you for your password and then sets about installing Beam Studio. If you picked the DEB package up directly from FLUX, you can ignore the warnings about the software being potentially unsafe.

Beam Studio asks you to set up an account and then prompts you to choose how to connect to your laser cutter (Figure 3). You can also optionally choose to calibrate the camera (Figure 4).

Test Job

FLUX thoughtfully provides a tiny piece of plywood and a test job to help you get started. It's a good way of finding out if your laser is correctly aligned. If it isn't, see the "Smoke and Mirrors" box. Place

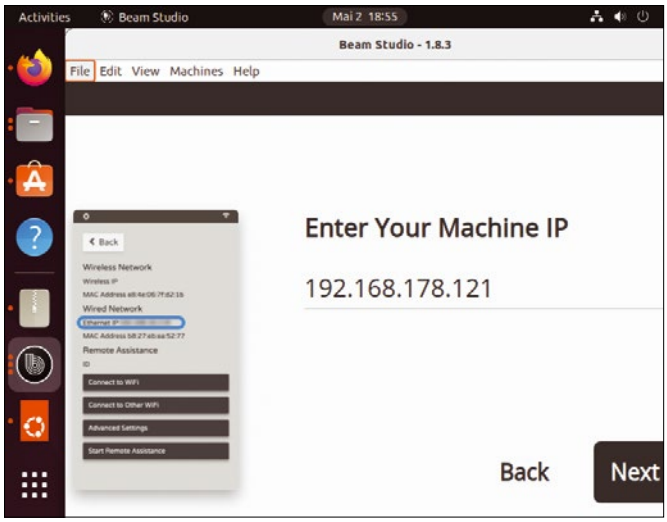


Figure 3: Selecting the laser cutter's network address in Beam Studio.

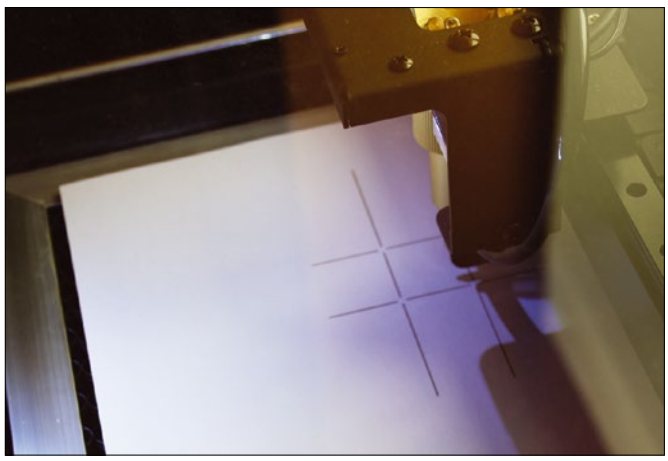


Figure 4: Calibrating the FLUX Beamo camera.

the piece of plywood on the grid and follow the procedure to set the focus of your laser – either fold down the piece of Perspex for a manual focus, or use the optional autofocus feature. Now click on the camera icon and drag a frame around the area where you placed the piece of plywood. This tells the laser cutter to switch on the camera and scan for the exact position of the wood. Then select *File* | *Example* and load the imaginatively named *Example of Beamo* file. As you can see from Figure 2, the file consists of two layers, a bitmap image with the *hello beamo* logo, and a single ring to cut out the engraving. This is an important feature. Layers let you engrave a bitmap image onto a suitable material such as wood or acrylics in one layer, and the cutting work with a simple vector image is assigned to a different layer with different parameters.

Group the two images and center them on the plywood. The default parameter settings were chosen by FLUX and will be perfect for the job using the piece of plywood provided. Press *Go*, and then in the Beamo window that appears, press the arrow button to start the job. If all of this worked, you are well on your way to getting creative with your FLUX laser. The next part of the journey involves creating your own designs on paper and scanning the results for post-processing with a software tool capable of handling both vector and bitmap images. In the Linux world, and beyond, the tool of choice here is likely to be Inkscape.

Inkscape

Installing Inkscape on Ubuntu is as easy as pie. If this amazing, free program isn't already installed, simply launch the Ubuntu Software Center and type *Inkscape* in the search box. Then type your password when prompted, and wait for Inkscape to install. Now it's time to get creative. Christa had this idea for a Christmas decoration showing the Three Wise Men, a star, and a camel. With her being one of those fortunate people who can imagine a scene in their heads and draw it on paper without needing photos or sketches as a reference, she just sat

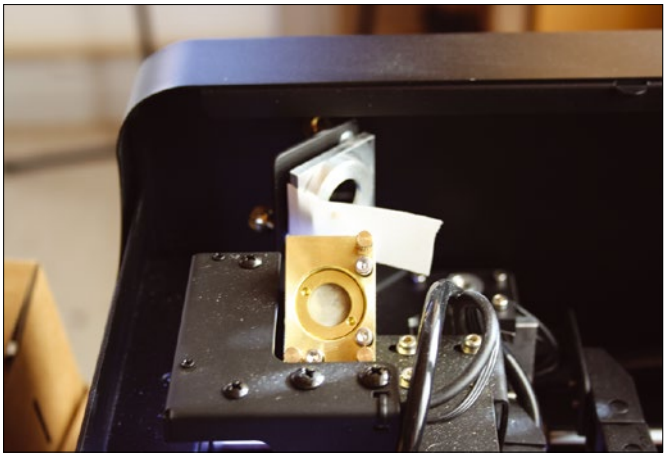


Figure 5: Adjusting the mirror alignment.

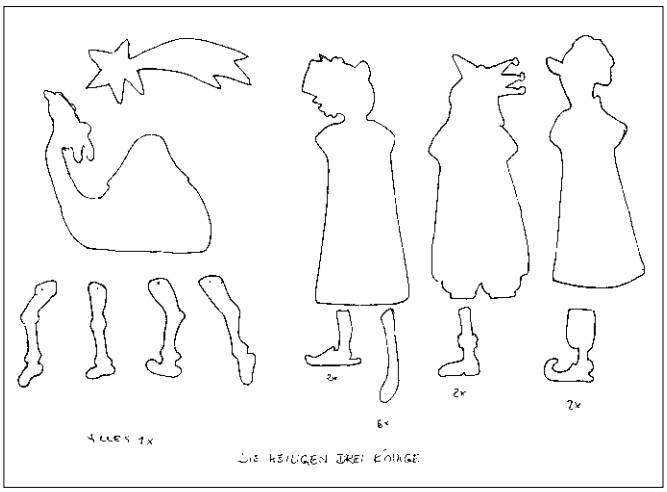


Figure 6: Scan of the Three Wise Men. Count the arms and legs!

Smoke and Mirrors

As you are probably aware, laser cutters are precision instruments. What you may not have been aware of up to this point is that the laser beam is deflected by mirrors to the laser head. Any knocks the laser's packaging took en route to its new owner are likely to cause some misalignment issues. You may get lucky and not need to align the mirrors on your machine, and you may not. We were not so lucky and had to align the first reflecting mirror (Figure 5), which involves removing a large number of Allen screws before you can start the alignment. It sounds complicated, but if we managed it, we're sure you can, too. Here's what you need to do:

1. Mirror one: With the laser switched off, attach a piece of white sticky tape (that's why it's included in the scope of supply) to mirror number one. Close the laser's lid and switch the power on. On the touch panel, tap *Maintain*. On the Maintenance screen, tap *Motors*. Then move the laser head manually to the top left position after doing so. Now press *Laser Pulse* to fire one shot at the tape (you should see a puff of smoke) and make sure you can see a dot. Now, move the laser head manually to the bottom left position. Make sure the lid is closed and fire another shot at the tape. If the dots from the laser shot line up, you can move on to the next mirror. If they don't line up, adjust the screws on mirror number one and repeat the steps until the mirrors are lined up. If the second shot misses the tape entirely, move the head to center left position to check the direction in which you need to adjust. Turning the top left screw clockwise moves the dot down and to the left. Turning the top right screw clockwise moves the dot to the right, and moving the bottom screw moves the dot upward. Easy. All done. Now for mirror number two.
2. Mirror two: This time attach a piece of sticky tape to the metal ring on mirror three. Move the laser head to the center left position. Close the lid again and fire a shot. Then move the laser head to the center right position and fire another shot. The dots need to overlap. If not, adjust the screws on the second mirror until everything lines up. The screws are in different places and do different things this time. If you are really unlucky, you may need to adjust the laser head, the laser tube, and the third reflecting mirror, too. We're not going to go into the procedures for doing so here, but again, it's all smoke and mirrors. You may need to RTFM (which naturally means Read the FLUX Manual) at this point if you really need these adjustments. But let's hope you were lucky and can move straight to the first actual laser engraving and cutting task.

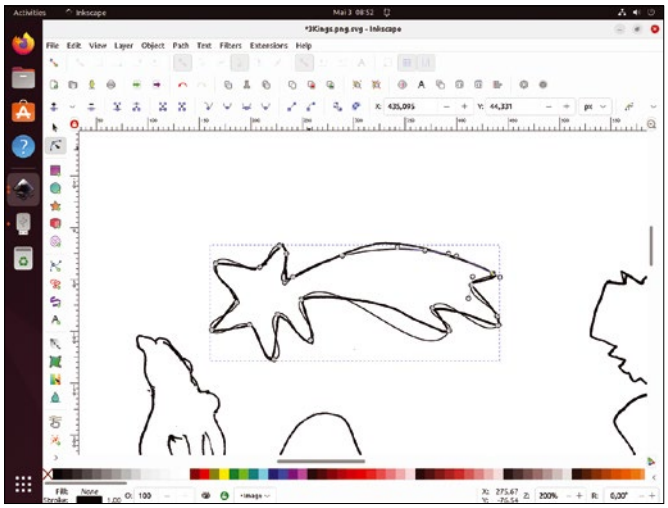


Figure 7: Tracing the original drawing in Inkscape.

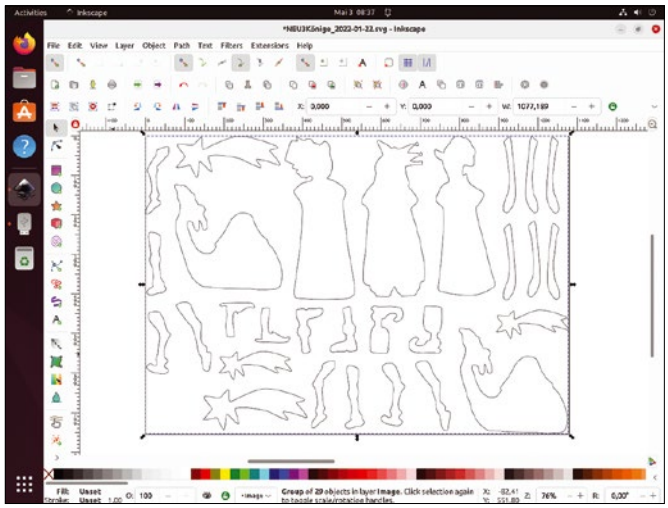


Figure 8: After tracing the original drawing, it just wouldn't fit into a DIN A4-sized space in the SVG file for the laser cutter.

down and drew the image that you can see in Figure 6.

If you look closely at the figure, you're probably wondering what happened to the wise men's arms and legs. The idea here is that, if you want to keep something symmetrical, you just do the drawing work once and then scan it. Next, you load the scan in Inkscape, trace the drawing to create a vector diagram, and copy and paste the resulting object. While the kings' torsos are needed once each, we need six arms and two of each of the legs, as you can see from Christa's notes. If you are not good at drawing, don't worry, there are plenty of free-to-use sketches out there on the web that you can download and process using similar steps to the ones we will be following.

There are two schools of thought on converting a bitmap, such as a PNG file, to a vector format (SVG in Inkscape) for laser cutting. One approach is to select *Path* | *Trace Bitmap* and let the algorithm follow the contours of the original drawing. The other is to use the original drawing as a template and do the tracing work yourself. Although the automatic trace seems to be the easier option, it will probably give you a very complex vector diagram with a huge number of nodes. Despite selecting *Path* | *Simplify* to reduce the number of nodes, post-processing is likely to take longer than the manual approach. The *Simplify* results are not always what you expect and there may be some artifacts. But this is something you need to try out for yourself, and the decision will always be driven by the results of a trace.

If you decide to do a manual trace (Figure 7) yourself, the steps are as follows: First select the *Draw Bezier curves and straight lines* tool in the buttonbar on the left of the Inkscape window. Click on a starting point and drag a line to the next point on the object you are tracing. Click again, drag again, and keep repeating until you have traced the whole object. When you are done, you will definitely need to select *Edit paths by nodes* and tidy up the curves and lines by clicking on the nodes and dragging the handles that Inkscape shows you. Once you have a vector replica of the object, you can delete the original hand-drawn object, but make sure you keep a backup copy, just in case you delete the wrong object – something that can easily happen.

After tracing an arm, we selected the object, right-clicked, selected *Copy*, and then clicked on a new location where we

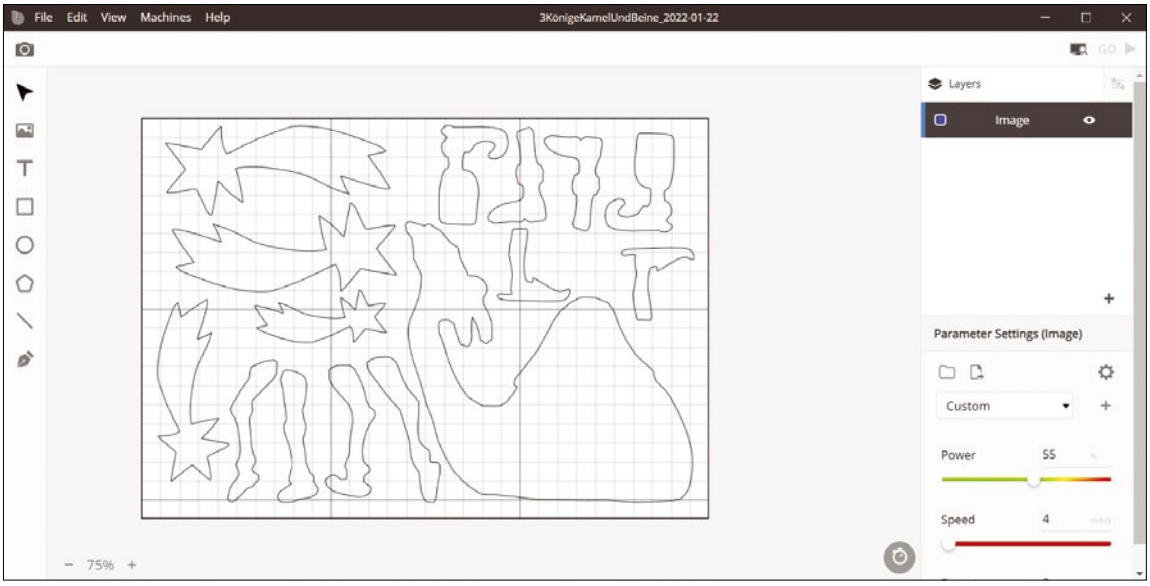


Figure 9: Splitting up the original drawing across two SVG files gave us the extra space we needed for this project.

wanted to position the copy. Then it was just a matter of keeping on pasting until we had the right number of objects. This assumes that the wood doesn't have a "good" side and a "bad" side and that you can work with mirror-symmetrical blanks. Of course, we had to do this again for the sets of legs.

On our first attempt after converting the original drawing, and making sure we had the right number of arms and legs, we just ran out of space. If you look at Figure 8, you will see that the camel is far too small compared to the other figures, and the kings' arms look a little spindly. Because the laser cutting space of the FLUX Beamo is about the size of an A4 sheet of paper, we eventually had no option but to split the drawing across two A4 sheets of plywood. This gave us bigger kings, a bigger camel, and more sturdy arms that were less likely to break during the assembly and decorating process. Figure 9 shows one of the images featuring the camel and an assortment of legs both kingly and otherwise.

The next steps should be familiar from the test job with the FLUX Beamo logo. Lay a sheet of plywood on the grid in the laser workspace. Check the setting for the laser head – you only need to do this if you are not using the same type of wood for this job as for the test job. Click the Camera icon and drag across the entire workspace to find out where the piece of wood is. Note that there is a blind area at the top of the workspace that the camera can't see but the laser head can still reach. Center the SVG vector diagram in the laser workspace, making sure it fits nicely on the sheet of wood. Then check your settings for the laser power, speed, and number of repetitions. We discovered that poplar plywood is easiest to cut, needing just one repetition rather than the two repetitions you need for heavier birch wood. You can tell that the settings are right if the figures you cut drop out of the board when the laser has cut all around them. If this doesn't happen, don't panic, just relaunch the job with another repetition but without moving or changing anything (we wouldn't want to waste that expensive wood). On a safety note, it's always tempting to watch the laser at work, but looking directly at the bright spot of the laser beam is a not good for your eyes.

Making Up the Puzzle

A quick look at Figure 10 shows what you can expect when the laser has finished its work. It's pretty, but not something that you would want to give to a friend or sell to a customer to finish off. In fact, this is where the second creative part of this build starts. But first, there is a little preparatory work to complete. As this is a hanging decoration, thin wire is a great way to connect the parts. The holes for threading the wire need to be drilled before you start decorating, and it's a good idea to clean up the edges of the parts to remove the darker laser burn marks with some steel wool.

Again, Christa has a great eye for color and decoration. She knows exactly what she wants her Three Kings to look like (Figure 11). You will need acrylic paints for the artwork, which is really a matter of your own style and sense of aesthetics. The one thing you do need to think about is that wooden decorations on thin plywood will warp if you don't paint both sides. You can paint the rear side in black if it will not be visible when the ornament is hanging, or go the whole hog and do realistic front and rear sides for more attractive results. Figure 12 shows you a fully decorated and assembled figure, ready for the next craft fair or as a unique gift.

More Applications with Lasers

There are virtually no limits to your creativity with the FLUX Beamo. Other 2D projects include laser engraving

Figure 10: A mass of blanks following laser cutting. Puzzle time!

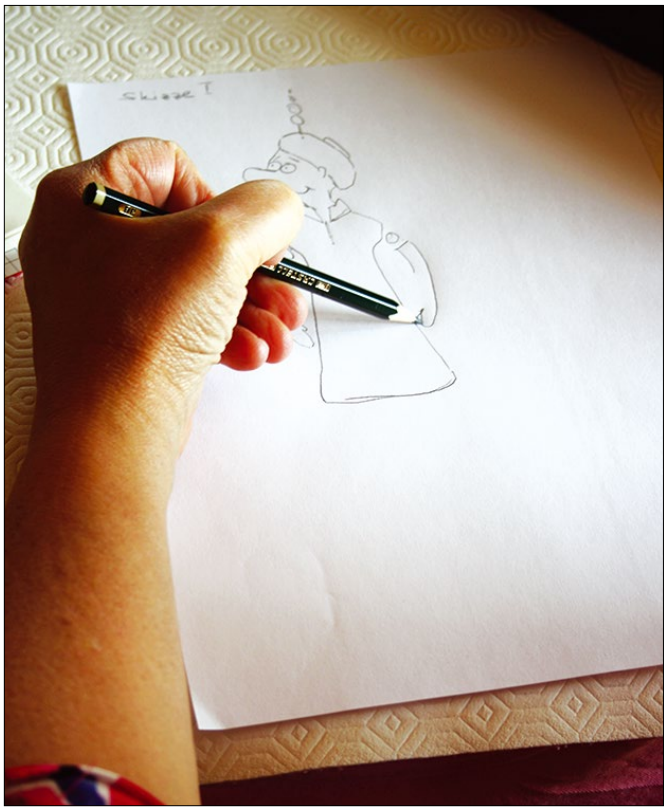


Figure 11: Creative work part one: creating an original drawing.

Figure 12: Creative work part two: assembled and decorated laser cut parts.

rubber stamps and creating printed works of art with them, or engraving photos on plywood, acrylics, stone, bone, or glass. You can even laser-engrave creative patterns or lettering on your jeans to make a fashion statement or cut cardboard to make inventive pop-up greeting cards. FLUX even offers an extension kit for engraving cylindrical objects, such as glasses with a diameter of up to 80mm if you want to surprise a friend with a personalized champagne glass, for example.

Conclusions

As this walk-through has demonstrated, you don't need to rob a bank to own a dependable laser cutting machine. There is no need to worry about vendor lock-in thanks to Beam Studio's ability to run on the Linux operating system and the fact that excellent, open source tools can be used for the tasks involved. Of course, a machine alone can't hope to replace the kind of artistic talent it takes to create original and attractive pieces of work that customers will want to purchase. ■■■

Info

[1] FLUX Raspberry Pi Board B10001: *https://www.fluxlasers.com/raspberry-pi-board-b100001.html*

[2] Beam Studio downloads: *https://flux3dp.com/downloads/*

Authors

Christa Travis is a mathematician and freelance portrait artist. **Ian Travis** owns a computer business and is responsible for translation and localization at *Linux Magazine* and *Admin Magazine*. When he's not sitting at the keyboard of some computer, he plays keyboards in a local band.

MakerSpace

Home Assistant controls microcontrollers over MQTT

Home Sweet Home

Automating your four walls does not necessarily require commercial solutions. With a little skill, you can develop your own projects on a low budget. *By Gerhard Schauer*

Home automation offers many opportunities, but equally harbors many risks. If you succeed in becoming independent of commercial products, you can save money while retaining control over what data flows where. In this case, the Message Queuing Telemetry Transport (MQTT) protocol proves to be very useful.

A previous article on Z-Wave [1] showed how you can bring the Raspberry Pi up to speed with Home Assistant and components available on the market to attain the goal of achieving automation magic in your home without human interference. Plenty of components can be addressed by Home Assistant, even without the cloud. The only limits are your wallet and your imagination.

This sequel explains how you tweak both the price and the DIY factors of the components. The Raspberry Pi from model 3 onward comes with a wireless interface that is also available in many microcontroller modules and is likely to open many doors. The glue that connects the whole thing to the Home Assistant environment looked at in the previous article is an IP-capable protocol that saw the light of day long before any Internet of Things (IoT) hype did: MQTT.

The MQTT protocol was introduced in 1999 and is a text-based protocol that runs over TCP on any IP network. Transport Layer Security (TLS) can optionally be used for encryption. The sensors and actuators communicate as clients with the broker, which acts as the communications hub. Addressing relies on what are known as "topics." Transmitters are referred to as publishers, and receivers as subscribers.

Testing MQTT

To implement the protocol on the Raspberry Pi, you first need the *mosquitto* package. For the small implementation tests that follow, you will also need the matching client. You can set up both components on the Raspberry Pi with the commands:

```
$ sudo apt-get install mosquitto
$ apt-get mosquitto-clients
```

If the broker is running after the install, you can log in to the individual topics with the client and retrieve a list of topics with a command-line call that specifies the host on which the broker is running withthe `-h` option:

```
$ mosquitto_sub -h localhost -t "#"
```

In this case, it is localhost. For example, to send the value *10* to a topic

Lead Image © shamain, 123RF.com

named *test/test1* and receive the value as a subscriber, you would use the commands:

```
$ mosquitto_pub -V mqttv311 ➥
  -h 192.168.3.7 -t test/test1 -m 10
$ mosquitto_sub -h 192.168.3.7 ➥
  -t test/test1
```

For a more practical example, I'll use MQTT to transfer the measured values of a one-wire temperature sensor of the DS18x20 type connected to a Raspberry Pi to the broker listening on another Raspberry Pi. To do this, you need to enable one-wire support up front with

```
dtoverlay=w1-gpio
```

in the `/boot/config.txt` file.

After connecting the sensor to the standard one-wire GPIO port, the temperature values can be queried as four-digit integers in `/sys/bus/w1/devices/28-0417c1b1f7ff/w1_slave`. The value `28-0417c1b1f7ff` is the unique ID of the sensor on the bus. The complete command sequence is:

```
$ mosquitto_pub -h 192.168.3.7 ➥
  -t wohn/temp/28-0417c1b1f7ff ➥
  -m $value
$ mosquitto_sub -h localhost ➥
  -t wohn/temp/28-0417c1b1f7ff
```

To transfer the values to the Home Assistant environment, you need to enter the broker you are using in the `configuration.yaml` file; in this case, it is running on the same host (Listing 1, lines 1-2). The entries in the lines that follow take care of subscribing to the appropriate topic and fielding the sensor data in Home Assistant; the last line also specifies degrees Celsius as the unit.

ESP8266 Controller

The theoretical side is now covered, but to use a separate Raspberry Pi for each remote actuator or sensor seems a little unrealistic in practice. The ESP8266 controller will come in handy here. On the one hand, you can source low-priced and useful developer boards that use this controller; on the other hand, the chip is used in various products from the Far East, and you can take control of many of these simply by flashing them with freely available firmware.

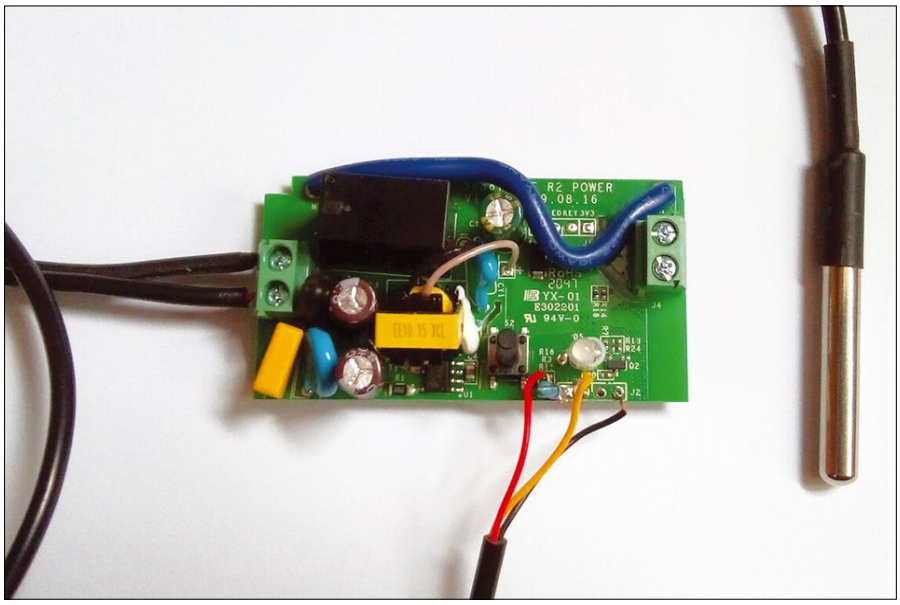


Figure 1: The Sonoff Basic WiFi smart switch with temperature sensor.

For the tests here, I used a Sonoff Basic WiFi smart switch [2] (Figure 1). This controllable relay switch has a screw connection for a 220V mains supply with switchable output.

CAUTION: There is a danger to life if this hardware is not used correctly. In this article, I am restricting the supply voltage to the module to 3.3V.

The firmware is Tasmota. The software lets you configure a variety of supported sensors and components on the existing GPIO ports of the controller, such as the temperature sensor I referred to earlier. Tasmota uses MQTT to transmit the data from the sensors.

Listing 1: configuration.yaml

```
01 mqtt:
02   broker: localhost
03
04 sensor:
05   - platform: mqtt
06     name: temp1
07     state_topic: "living/temp/28-0417c1b1f7ff"
08     unit_of_measurement: '°C'
```

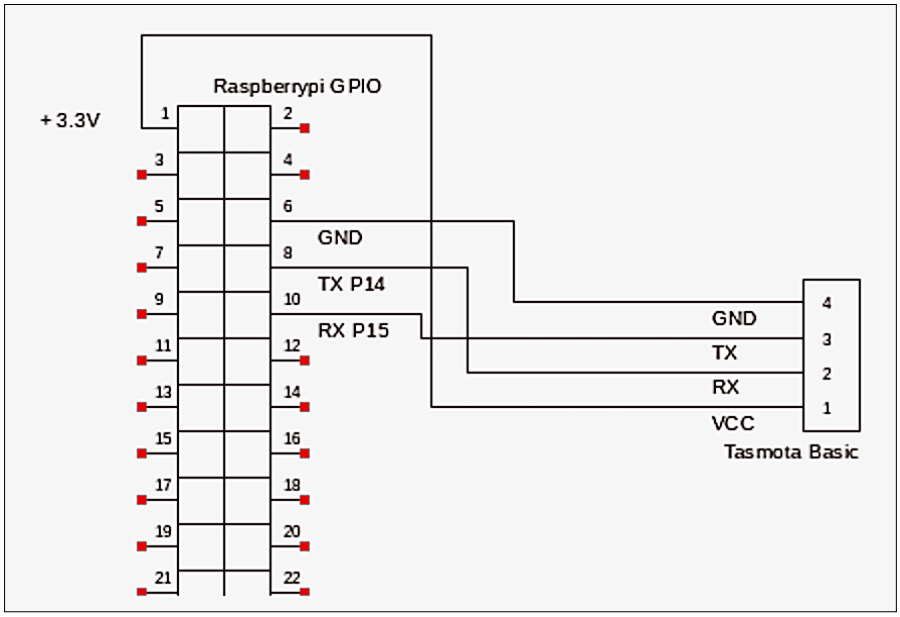


Figure 2: Wiring the Sonoff Basic to the Raspberry Pi.

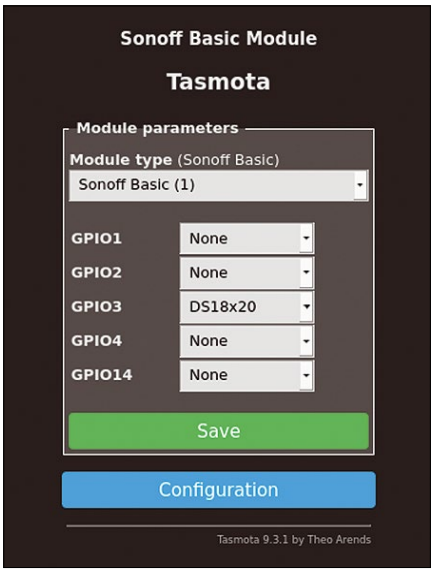


Figure 3: A convenient web GUI helps you set up the temperature sensor.

Flashing Tasmota

Among the several ways to flash Tasmota, one popular approach is to use a USB serial adapter. A description can be found on the Getting Started page of the project [3]. Because the Raspberry Pi comes with a serial interface, I will do the flashing directly through the GPIO.

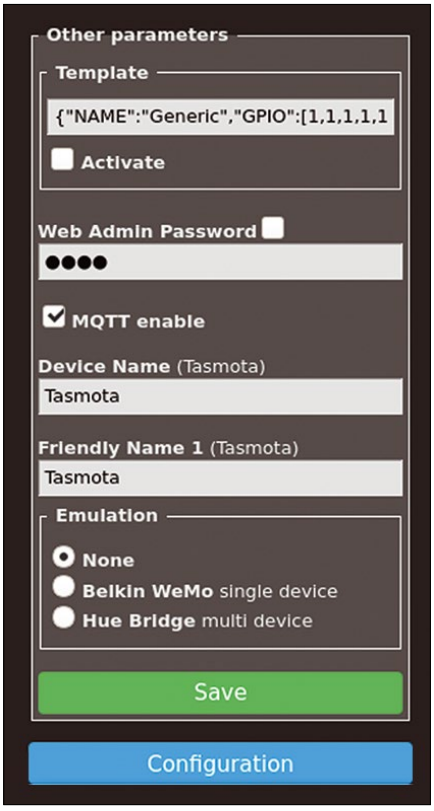


Figure 4: Just a few parameters let you prepare Tasmota for transferring data to Home Assistant.

The wiring required for this is shown in Figures 1 and 2. The `esptool.py` utility from the PiOS repository provides the software.

The Tasmota firmware, along with a massive collection of information, can be found in the project's GitHub repository [4]. I downloaded the sensor variant for the tests because I wanted to integrate the temperature sensor:

```
$ wget -c ➥
  http://ota.tasmota.com/tasmota/➥
  release-9.4.0/tasmota-sensors.bin
```

After connecting the module as shown, you have to hold down the button on the Sonoff module while powering it on; the LED does not light up. Next, use the command

```
$ esptool --port /dev/ttyAMA0 chip_id
```

to check whether you can access the Sonoff. An ID must be shown.

Finally, flash Tasmota with the command:

```
$ esptool ➥
  --port /dev/ttyAMA0 write_flash ➥
  -fm dout 0x0 tasmota-sensors.bin
```

This operation should take about a minute. When the module is restarted, it reports for duty as the new WiFi access point, which you can reach at *http://192.168.4.1* and on which you configure the WiFi network to be used. All further setup steps are then performed with the IP address assigned by your own WiFI network.

Setting Up Tasmota

Before proceeding, you first need to connect the temperature sensor to one of the GPIOs used for flashing (RX/TX), which is now free. You can do this with a pull-up resistor and the 3.3V and GND connections [5]. After wiring the elements, it's time to configure the sensor; its name will have already appeared in the GUI (Figure 3).

You also need to assign a password for the web GUI and activate MQTT (Figure 4), which you will be using to deliver the data to Home Assistant. The broker and topic are configured in the corresponding entries in the web interface, where you will also find the Tasmota console, which lets you issue commands directly and set up the parameterization.

At this point at the latest, I recommend studying the very good documentation from Tasmota's GitHub repository. For this project, I used a command that makes the later integration with Home Assistant by autodiscovery very easy by telling Tasmota to announce all available topics (`setoption19 on`). At the end of the day, Tasmota advertises itself in the GUI as a web switch with a temperature sensor (Figure 5). You can also view the

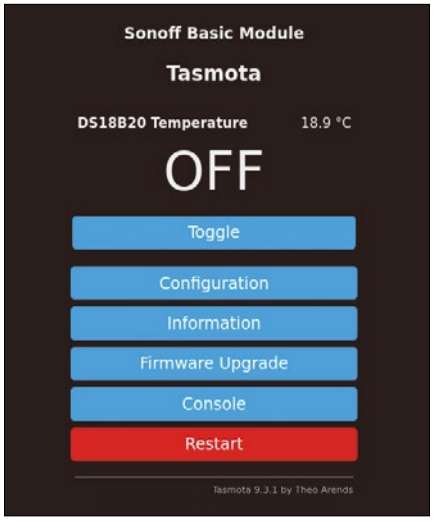


Figure 5: Hardly distinguishable from commercial products: the web-based user interface of Tasmota.

Figure 6: A few simple steps are all it takes to integrate the MQTT protocol into Home Assistant.

MQTT publishing in the Console, if needed.

Integration

To make the components available in Home Assistant, you now need to set up MQTT integration in the GUI. To do so, all you need is the IP address and port of the broker you are using; by default, this is set to 1883. After the setup, the new MQTT devices should be displayed (Figure 6).

After clicking on the *entities* URL, both the switch and the temperature sensor are shown as available Home Assistant entities, and you can now use them in the usual way (Figure 7). In combination with the relay in the Sonoff module, the familiar automation features of Home Assistant can now be used to implement, say, a simple thermostat control.

Conclusions

The Tasmota option associated with Home Assistant's autodiscovery of MQTT integration easily makes this DIY solution as convenient as the commercial counterparts described in the previous article. Homebrewing turns out to be worthwhile – and great fun, too. ■

Author

Gerhard Schauer is a self-employed electronics engineer living in the southern part of Germany. He writes maker articles because learning by doing and maintaining control of technology is the "right" way.

Info

[1] "Z-Wave Home Assistant" by Gerhard Schauer, *Linux Magazine*, issue 248, July 2021, pg.56, *https://www.linuxpromagazine.com/Issues/2021/248/Z-Wave-Home-Assistant/*

[2] Sonoff Basic R2 Power: *https://tasmota.github.io/docs/devices/Sonoff-Basic/#sonoff-basic-r2*

[3] Getting Started: *https://tasmota.github.io/docs/Getting-Started/*

[4] Tasmota repository: *https://tasmota.github.io*

[5] Wiring Tasmota: *https://tasmota.github.io/docs/DS18x20/*

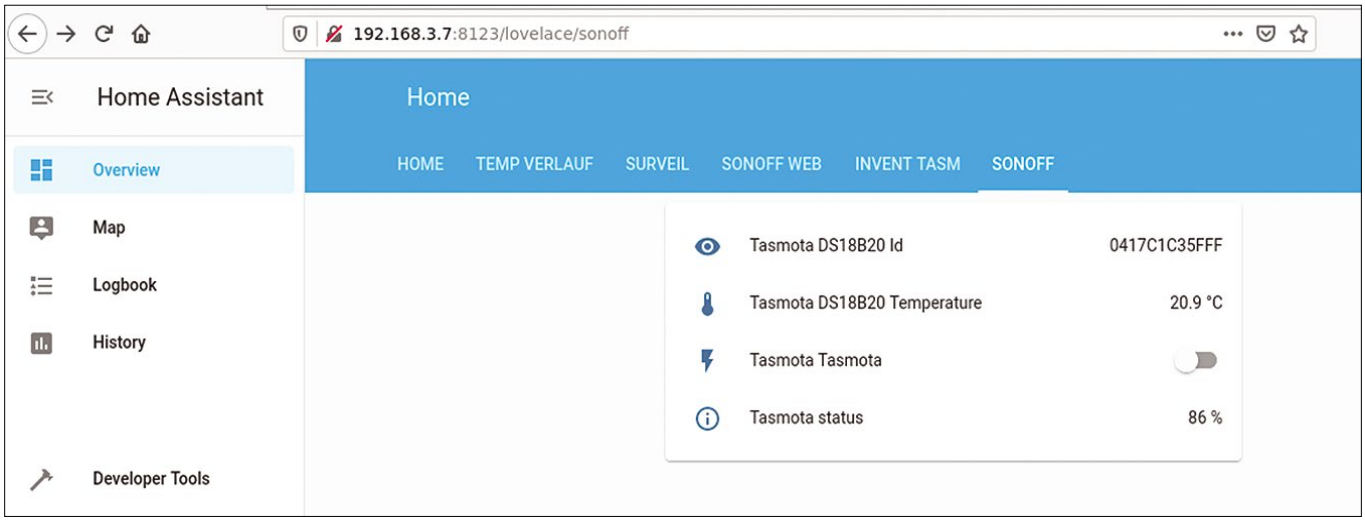


Figure 7: Home Assistant presents the integrated sensors in a clear-cut and neat way.

Hone your skills with special editions!

Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

Some of the best articles are the ones that don't just teach about a tool – they teach about the reason for the tool. When you read one of those articles, you aren't just learning about Linux: You are learning about some other aspect of life on Earth that brings value to the reading experience. This month we feature one such article. You'll learn about the Manuskript editor and novel-writing tool, but you'll also get an introduction to the *snowflake method*, an outlining technique that takes the intimidation out of writing long fiction works. And while we're on the subject of creating things in original ways, how about building a slide presentation using the Go programming language? Read on for a study of Presentation as Code.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

Jon "maddog" Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

MADDOG'S DOGHOUSE

Maddog considers the history of chess as a metaphor to grow the desktop Linux user base. BY JON "MADDOG" HALL

Growing Linux Desktop

There is a story about the game of chess and how it was invented centuries ago. After the game was invented, the ruler of the country was so happy that he offered the inventor a reward for their efforts. The inventor requested a grain of rice on the first square, two grains on the second square, four grains on the third square, and so forth. The ruler, speaking quickly, said "Let it be done." The problem was that there was not enough grain in the entire kingdom.

Today we might think of this as "two to the 64th power," a number so large that it is fairly unimaginable. Even two to the 32nd is over four billion, and, if you start with a number greater than two, it gets very large very fast. It is this concept that I was thinking about when I asked Linus Torvalds to port Linux from the 32-bit Intel platform to the 64-bit DEC Alpha.

Today, I would like to apply this concept to a different subject, that of desktop Linux, one of the few places where Linux does not have the penetration that it should have. Few people would say that Linux does not have penetration in the high performance computing market, the server marketplace, the embedded system space, and the cell phone space (as Android), but we have been stymied in the desktop space to the point where "The Year of Desktop Linux" has been a running joke for at least two decades.

In the early days of Linux, there were some legitimate desktop issues. Linux was cited as being hard to install, and Microsoft Windows was "easy." People ignored the fact that all laptops and desktops that people would normally buy already had some version of Microsoft Windows running on it, because that is the way that distributors and retailers sold it.

Very few end users ever installed Microsoft Windows, they just re-installed it from backup images already created for the system. Periodically, people would install new devices, but these devices would come with binary device drivers created for various versions of Microsoft Windows by the device manufacturer.

Today, there are few issues of installing mainstream distributions of Linux. With recent announcements of GPU manufacturers restructuring their drivers to include more open source and to integrate that open source with the kernel, there will probably be even fewer issues.

Other complaints were a lack of applications, which over time have been addressed by more applications being ported and use of virtualization, simulation, and emulation. More than that, this is a catch-22 issue: If there were more people using Linux on the desktop, more application vendors would support Linux.

Lack of games was another famous issue, which has lost traction with more games being ported, Steam creating a Linux platform for games, and game consoles becoming less and less expensive and becoming a platform used by many gamers. Additionally, native porting of games would be increased by having a greater desktop presence.

One of the real issues why Linux does not have more desktop presence is simply marketing and sales. The number of advertisements, product placements, and other marketing events that can be produced by FOSS companies is dwarfed by the marketing that can be done by Microsoft.

When I worked for Digital Equipment Corporation (DEC), a rule of thumb in the software industry said that 36 percent of the retail price of software was spent in sales and marketing. In other words, more than one-third of what people paid had nothing to do with engineering or even manufacturing the distribution media. Today, with the software either pre-installed or downloaded for installation, sales and marketing would probably be more than 50 percent of the cost, and (as in the case of bundled software) sales and marketing of third-party software (known as "bloatware" to readers) may actually make money for the original equipment manufacturer (OEM).

FOSS software finds it hard to compete with that marketing model on the desktop.

However, FOSS can do marketing using the chessboard model.

Each of the existing Linux users could spend some time with two Windows users and help them install Linux on their desktop. Help them find beginning books on Linux, help them find the applications they need, help them find online help groups and really get them started, and then next year we would have three times the number of Linux desktop users.

It is even better if those Windows users are teachers, town council members, government officials, business people, etc.

Then the next year each of those three find two more Windows users. This might sound like too little and too late, but we are starting with millions. There are approximately two billion desktop users. It is estimated that three percent of these are Linux users (I am using very conservative numbers), so that is 60 million Linux desktop users.

I think it is time to play chess. ■■■

Building Slide Presentations with present
Presentation as Code

The Golang package present may be the key to making attractive slide presentations with less work and hassle. BY ANKUR KUMAR

Creating slide presentations has been a necessary part of technical life for a long time, but creating crisp and beautiful slides using the popular traditional tools requires a lot of tedious work. I have always been intrigued by the elegant presentations in Golang community talks, but there was no clear-cut information available on how those beautiful presentations were rendered. In researching, I stumbled upon a Golang package named, not surprisingly, `present` [1], which renders amazing presentation slides from markup text description. For many years now, `present` has been my go-to tool for creating and delivering impressive presentations.

Getting Started

There is no separate installation step needed to start using the `present` utility. It's just a statically linked binary that is grab-and-run; there's no need to set up any other runtime dependencies. You do need the Golang compilation toolchain already set up on your machine if you want to run the `present` command natively. Alternatively, you can run `present` out of the box, provided Docker Engine is installed on your machine (which is very common nowadays). I personally took the Docker route to use `present` without doing any extra work. You can use the Dockerfile (Listing 1) and script (Listing 2) to fetch and run `present` to display your slides on your local machine.

Listing 1: Dockerfile

```
01 FROM golang:alpine
02
03 RUN apk add --no-cache git \
04   && go get golang.org/x/tools/cmd/present
05
06 COPY run.sh /usr/local/bin/
07
08 ENTRYPOINT ["run.sh"]
09 CMD ["-notes"]
```

To create a Docker image from which you can launch `present`, use the following command:

```
docker build . -t present
```

You can also launch the `present` container to serve your slides from a bind-mounted directory (e.g., files in your current directory), by executing the command:

```
docker run -d --rm ➥
  -v ${PWD}/files:/src/files:ro ➥
  -p 58888:8888 present
```

Now open your web browser and access *localhost:58888* to ensure everything is running to deliver your presentation. Figure 1 shows the

Listing 2: run.sh

```
01 #! /bin/sh
02
03 PORT=${PORT_PRESENT:-8888}
04
05 exec present -http "0.0.0.0:${PORT}" -content /src/files "$@"
```

Figure 1: The `present` home page in a web browser without a presentation.

Figure 2: After creating `mytest.slide`, you can now click on a slide deck in the browser.

`present` container accessed by its localhost endpoint in my browser. Don't be confused about it not showing a presentation – I'll create the presentation next.

Slides Basics

Now it's time to dirty your hands with `present`'s slides description language. The preferred way to craft your slides is using the CommonMark markup language [2]. (You can also use legacy syntax [1] to describe your slides, but that's not discussed here.) To start, create a file named `mytest.slide` (Listing 3) in the bind-mounted `./files` directory.

Now refresh your `present` endpoint browser page, and the newly created slide file link should appear (Figure 2).

If you click *mytest.slide* now, you'll see an elegant three-slide presentation (the first and last slides are shown in Figures 3 and 4). Congratulations, you have created a presentation – without the frustration of dragging and dropping and the impossible formatting in a slide creation tool. You can move between the slides using the left and right arrow keys and also create a PDF using the Ctrl+P print dialog.

The slide description starts with a header block with the presentation topic prefixed with `#` and a space. Optionally, you could put other metadata such as a subtitle, date, tags, summary, and old URL lines after the `#`. An optional author block follows the header section, with at least a space in between them. The author block can contain your name, title, email, URL, twitter handle, etc. `present` automatically renders the last slide with the author block (Figure 4). It only puts author info that is not an email, URL, or twitter handle on the first slide (Figure 3). You could have multiple author blocks, each separated with at least a space between them. Any line starting with `//` is treated as a comment.

The presentation slide sections follow the author blocks. A slide section starts with `##` followed by at least one space. Each slide could have a subsection too, starting with `###` and followed by at least one space. The content of a slide is governed by CommonMark to format text.

Listing 3: mytest.slide

```
01 # My Test Presentation
02
03 Free Libre Open Source Software Hacker
04 A FLOSS Hacker, FLOSS Universe
05 flosshacker@flossuniverse.com
06 https://url/
07 @flosshacker
08
09 ## TBD
```

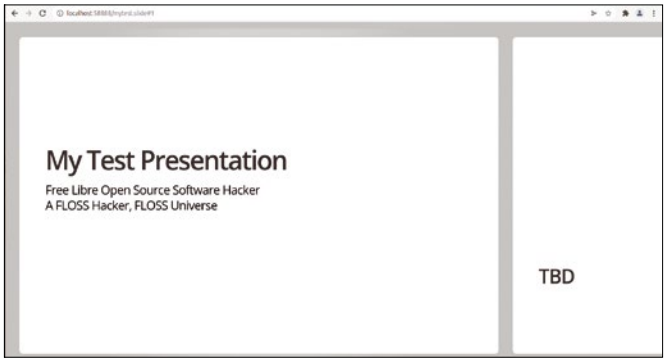


Figure 3: The first slide rendered by `present`.

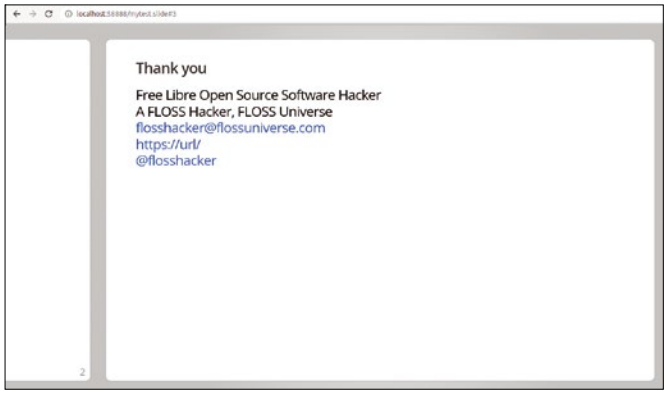


Figure 4: The last slide rendered by `present`.

Figure 5: A rendered slide using various common markup features.

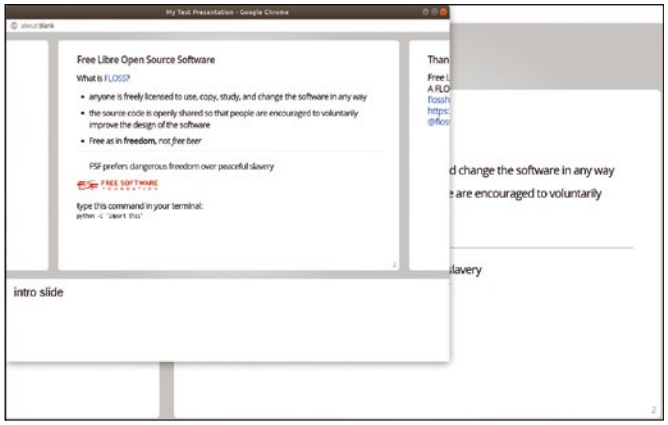


Figure 6: Pressing *N* in the browser displays a pop-up notes window.

To add more content to the slides, update your *mytest.slide* file with the code in Listing 4 and refresh the browser page rendering the presentation.

Now you should see common markup features (bold/italic text, lists, block quotes, links, images, horizontal rules, in-line code, etc.) implemented on your slide (Figure 5). To learn more about these markup features, see the CommonMark documentation [2].

Lines starting with a colon are treated as presenter notes. Pressing *N* in the browser showing your presentation opens a separate pop-up window displaying the notes (Figure 6).

More Enriched Slides

If you want more than text, hyperlinks, and logos in your slides, you can use `present` description language features to render eye-candy slides. To do this, `present` provides a number of special

Listing 4: mytest.slide Basic Content

```
01 # My Test Presentation
02
03 Free Libre Open Source Software Hacker
04 A FLOSS Hacker, FLOSS Universe
05 flosshacker@flossuniverse.com
06 https://url/
07 @flosshacker
08
09 ## Free Libre Open Source Software
10 What is [FLOSS](https://en.wikipedia.org/wiki/Free_and_
                  open-source_software)?
11 - anyone is freely licensed to use, copy, study, and
     change the software in any way
12 - the source code is openly shared so that people are
     encouraged to voluntarily improve the design of the
     software
13 - Free as in **freedom**, not _free beer_
14
15 ---
16 > FSF prefers dangerous freedom over peaceful slavery
17
18 ![FSF][1]
19
20 [1]: https://upload.wikimedia.org/wikipedia/commons/
       thumb/4/4a/Free_Software_Foundation_logo_and_
       wordmark.svg/260px-Free_Software_Foundation_logo_and_
       wordmark.svg.png
21
22 type this command in your terminal:
23 ```
24 python -c 'import this'
25 ```
26
27 : intro slide
```

Listing 5: mytest.slide Images and Sources Content

```
01 # My Test Presentation
02
03 Free Libre Open Source Software Hacker
04 A FLOSS Hacker, FLOSS Universe
05 flosshacker@flossuniverse.com
06 https://url/
07 @flosshacker
08
09 ## Free Libre Open Source Software
10 What is [FLOSS](https://en.wikipedia.org/wiki/Free_and_
                  open-source_software)?
11 - anyone is freely licensed to use, copy, study, and
     change the software in any way
12 - the source code is openly shared so that people are
     encouraged to voluntarily improve the design of the
     software
13 - Free as in **freedom**, not _free beer_
14
15 ---
16 > FSF prefers dangerous freedom over peaceful slavery
17
18 ![FSF][1]
19
20 [1]: https://upload.wikimedia.org/wikipedia/commons/
         thumb/4/4a/Free_Software_Foundation_logo_and_
         wordmark.svg/260px-Free_Software_Foundation_logo_
         and_wordmark.svg.png
21
22 type this command in your terminal:
23 ```
24 python -c 'import this'
25 ```
26
27 : intro slide
28
29 ## More FLOSS
30 ---
31
32 .background matrix.png
33
34 .image logofsforg.png 50 _
35 .caption FSF Logo
36
37 .code -edit -numbers hellofloss.c
38
39 .play hellofloss.go
```

commands using invocations. Any line starting with a dot character is an invocation. These commands include adding images, setting background, showing/editing/running code, creating a hyperlink, injecting video, including figure captions, and more. As an example, use the code in Listing 5 to once again update your *mytest.slide* file content, put the necessary images and sources into your presentation directory, and refresh the browser page rendering the presentation. Figure 7 shows a slide rendered using various invocations.

The `.image` invocation optionally takes height and width arguments. If any of these arguments are specified as an underscore, then scaling preserves the aspect ratio of the image. The `.background` invocation doesn't take any argument except the image. If your image is not big enough, then the background command will fill your slide with a repeated pattern of the mentioned image.

Both `.code` and `.play` invocations can strip the unnecessary code from the respective source and only display the necessary portion of the code. The `-edit` command enables you make modifications in the displayed code during your presentation. The `play` command displays code with a *Run* button to run the Go program from the browser.

Figure 7: A slide rendered using invocations.

Now you have enough knowledge to render rich presentations using the `present` description language. You can explore the full details of the `present` description language in the package documentation [1]. For more example slides and scripts, check out my GitHub repository [3].

Conclusion

The `present` package is a great addition to your day-to-day toolkit for rendering sophisticated-looking presentations with minimal effort. This utility is a natural fit for modern As-a-Code and GitOps workflows for automatically generating presentations without any extra effort. In the age of IAC and containers, creating stunning presentations couldn't be simpler or more automated than this. ■■■

Info

[1] present: *https://pkg.go.dev/golang.org/x/tools/present*

[2] CommonMark: *https://commonmark.org/help/*

[3] Docker scripts and example slides: *https://github.com/richnusgeeks/devops/tree/master/PresentationAsCode*

The Author

Ankur Kumar is a passionate free and open source software (FOSS) hacker and researcher and seeker of mystical life knowledge. He explores cutting-edge technologies, ancient sciences, quantum spirituality, various genres of music, mystical literature, and art. You can connect with Ankur on (*https://www.linkedin.com/in/richnusgeeks*) and explore his GitHub site at (*https://github.com/richnusgeeks*) for other useful FOSS pieces.

Turn your ideas into reality!

This is not your ordinary computer magazine! *MakerSpace* focuses on technology you can use to build your own stuff.

If you're interested in electronics but haven't had the time or the skills (yet), studying these maker projects might be the final kick to get you started.

This special issue will help you dive into:

- **Raspberry Pi**
- **Arduino**
- **Retro Gaming**
- **and much more!**

MakerSpace #02

ALSO LOOK FOR MAKERSPACE #01 AND ORDER ONLINE:

shop.linuxnewmedia.com/specials

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

There's a collective groan in Graham's household whenever he gets a new device and finds a terminal prompt. The latest victim to his nmap skills is an LG OLED television! BY GRAHAM MORRISON

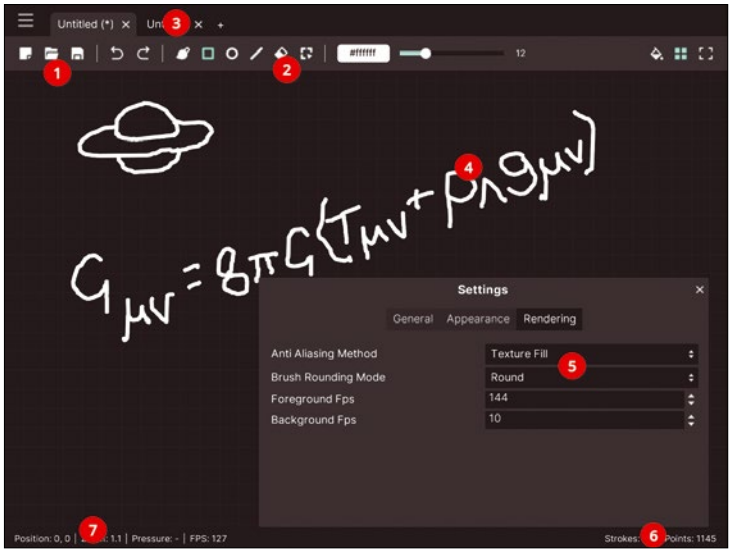
Virtual whiteboard

Lorien

There are reasons why whiteboards are synonymous with places of learning: They're brilliant for education and brainstorming, they're dynamic and adaptable, and they're collaborative. It's no surprise then that whiteboard functionality has been digitized, not just in schools with touchscreens, but also online where complex information can be presented on a virtual canvas that scrolls left and right and zooms out. A similar idea has been used with note-taking applications on the desktop. These usually mimic a notepad interface for drawing sketches and adding mental doodles rather than finding a way to map more complex ideas across a larger canvas. Even Apple's own note-taking application works in the same way with the next version promising to unshackle the fixed canvas into an infinitely scrollable canvas. Which is something this wonderful application, Lorien, can already do.

Lorien is a whiteboard with an infinite canvas. What this means is that you can start doodling anywhere on the background and then scroll up, down, left, or right to add further annotations or notes, just as you might on an infinitely large university whiteboard. This approach extends to zooming in and out of the canvas so that you could, in theory, fill a virtual microdot full of information behind a single pixel on your screen. This is possible because Lorien isn't storing bitmaps of your doodles and texts, but instead an algorithmic representation of everything you add along with their relative relationships to one another. This is how Inkscape works, how SVG files represent vectors, and how Minecraft rebuilds your house somewhere within a never-ending landscape. It allows huge bitmap images on your screen to be saved as tiny files and redrawn perfectly at any scale. It also means you get infinite undo and redo, as well as SVG export of your doodling. Unlike Inkscape, however, Lorien is only intended for your own notes and brainstorming and not for artwork. To best accomplish this, Lorien has a tabbed interface to work with multiple files at once and will accept files dragged onto the canvas view.

That doesn't mean the drawing tools are simplistic, though. There are brush, eraser, line, and rectangle tools, along with different brush strokes and palettes, but you can't fine tune your drawing in the way you can with something like Krita. Lorien is instead intended to be used to make quick notes and sketches without too much thought or artistic finesse. Your work can still look beautiful, thanks to the anti-aliasing and the vectorized drawing, but only in the way great ideas might when they're written down on the back of an envelope. Lorien is also intended to be used with a stylus, perhaps even with Gnome's new RDP and touchscreen support, where you can use pressure to dynamically change the brush size. But you can also use your boring old mouse, where the middle button is used to drag the canvas, and zooming in and out is mapped to the wheel. Files can then be saved natively or exported as an SVG for importing into another application. Everything is rendered perfectly without taxing your CPU. This is likely because Lorien is built with Godot rather than Qt or GTK, which allows Lorien to take advantage of the same performance acceleration used by games. Even more excitingly, it might mean we see other similarly excellent applications coming from what was once predominately a games engine in the future.



1. Format support: Along with Lorien's own file format, documents can be saved as SVG files. **2. Tools:** Draw, sketch, and type to add notes to your own whiteboard. **3. Tabs:** Work on multiple documents at once or enable the distraction-free mode. **4. Infinite canvas:** The background you work on can be any size and scale. **5. High frame rate:** Thanks to using Godot, Lorien renders more like a game than a desktop application. **6. Vectors:** Everything is a vector, keeping file sizes small and quality as high as you need. **7. Stylus support:** Lorien is really designed for use with a stylus where it can use the pressure to change the drawing weight.

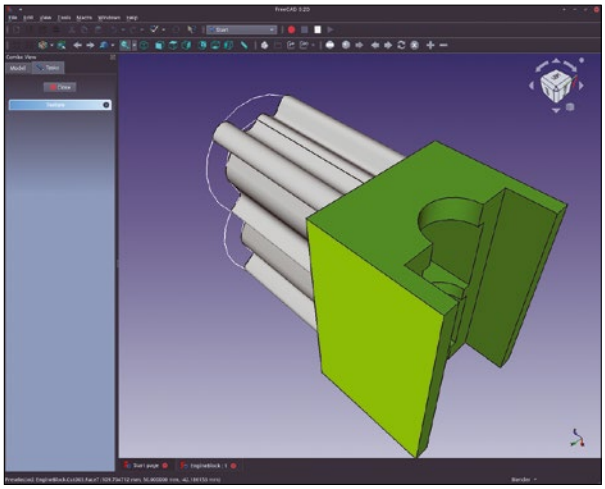
Project Website
https://github.com/mbrlabs/Lorien

3D design

FreeCAD 0.20

A major 0.20 release gives us the perfect excuse to revisit this mainstay of 3D modeling and design. FreeCAD has been around for 20 years, and as a result, it's complicated. Much like Blender, this isn't an application you're going to master in 20 minutes. But also like Blender, it's capable of brilliant and professional results, from 3D printing to circuit design. There are several different ways of working with FreeCAD, with the most common being to develop your design from a two-dimensional sketch in the X and Y plane. FreeCAD uses the idea of workbenches to change its configuration to suit whatever way you work, and there are several that help with this stage by creating a 2D top-down view, for example. With that done, you first create a sketch before more menu diving to select the shapes you want to add. These are created interactively on the canvas, letting you drag their edges and points to fit the size you need.

In this way, FreeCAD is part Gimp and part Blender with one important difference: constraints. Constraints, such as the length of a side or the angle of something, are an important concept in FreeCAD because they define the shape of an object without any ambiguity. The object creation process involves going through each parameter and fixing it in place to create a constraint. When a shape is fully constrained, it turns green and is considered solved. Everything can be created by clicking and dragging a point or line, but also made exact by editing parameters, like a more interactive OpenSCAD. But like OpenSCAD, objects can also be fully programmed in Python if you wish. Fortunately, this new release helps massively with usability, including decent help text and improving lots of user interface elements, making them more intuitive and easier to use. It's still complicated, but also a lot easier than ever before.



FreeCAD includes elements of OpenCASCADE Technology (OCCT) to perform operations on complex 3D shapes.

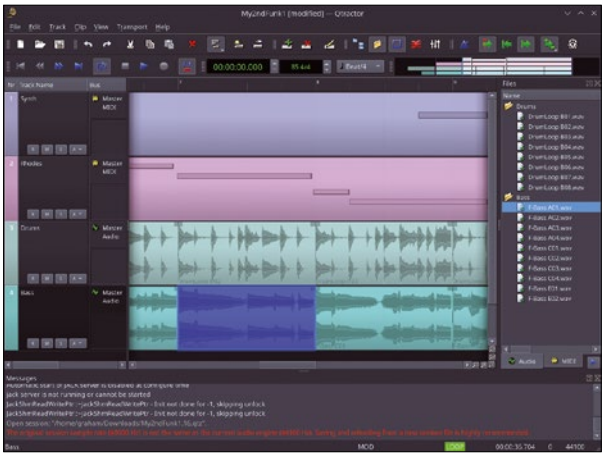
Project Website
https://www.freecadweb.org

Audio plugin API

CLAP

This isn't a normal discovery because, rather than being something you can run, CLAP has been developed to help other people create things to run. Specifically, it's an API to help programmers write audio effects and synthesizers that can work with any audio platform. This should be a solved problem because that kind of API has been around for decades – most famously in both Steinberg's VST plugin architecture and Apple's Audio Unit app. These two dominate professional audio, and VST is popular on Linux where native plugins can be built or Windows VST plugins run through Wine. The problem is that neither of these APIs are open, and VST is still controlled by Steinberg. There are older open source alternatives, including LV2, LADSPA, and DSSI, but developers complain about their complexity and fragmentation, and each has failed to reach a critical mass of use outside of Linux, which is vital if any format is to succeed.

CLAP (short for Clever Audio Plugin) is a new open source audio API that's been designed specifically to be easy, adaptable, and efficient. It's also potentially much more capable than any other plugin format. It allows for per-note automation and modulation, parameter offsets, non-destructive performance sound modification, and per-voice parameters. None of this can be easily accomplished with the alternatives, making these genuine reasons to choose CLAP above all the other formats regardless of its license. Even more impressively, CLAP has been developed by two important companies, Bitwig and u-he. Both companies produce Linux versions of their software, with Bitwig in particular becoming hugely successful as a challenger to the industry standard Ableton Live. Bitwig is already a host for CLAP plugins, and u-he is in the process of converting theirs, as are other Linux-friendly software producers. It's important to note that CLAP's ambition is to succeed across all platforms, and its MIT license and a promise of open collaboration are just as important for this. Hopefully, it will be hugely successful.



Bitwig's own software is one of the first hosts for CLAP, as is the entirely open source and brilliant Qtractor.

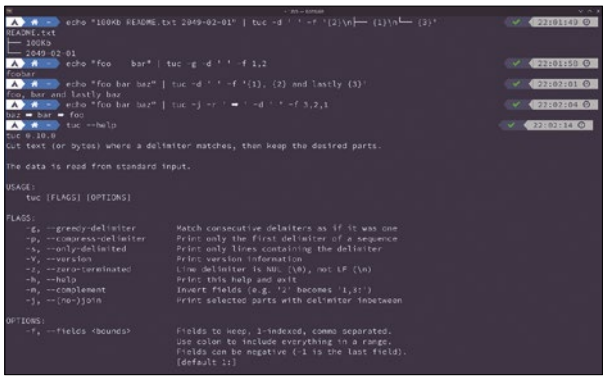
Project Website
https://github.com/free-audio/clap

Cut replacement

tuc

One of the most underused and underrated commands that you find built-in to the average terminal, `cut` has been around for a long time. While many of us can usually guess its function, our finite memory means it's usually easier to construct an equivalent command from pipes or even copy and paste than bother with the `cut --help`. The `cut` command helps you to extract various parts of a file, or piped input, by specifying ranges, delimiting characters, and even byte positions. It's powerful, but it's no longer the only cut command you should have in your path. That's because there's a new alternative called `tuc`, a cunningly named tool that does everything `cut` does while also filling in gaps in its functionality. And it does far more than reversing the order of text.

One of `cut`'s best uses is to process and extract data from a comma-separated list, and this is a good example of something that `tuc` improves upon. `tuc` is good at extracting the same data in the same way, but as its name implies, it's particularly good at rearranging the order of the fields it extracts. You can do this by simply specifying the new order for each delimited value (`-f`) after specifying a delimiter (`-d`) to use, such as a comma. You can then put the same or different delimiters into the output text or pipe the output for more processing. It's also easy to replace fields or their delimiters or reformat the output using references to the various values retrieved from the input stream. It sounds abstract when written like this, but `tuc` is actually easy to use and solves many of the problems you find when trying to import or process a list of values with `cut`.



You easily reverse a list of items or even entire lines of text with `tuc`, thank to its ability to handle negative indexes.

Project Website

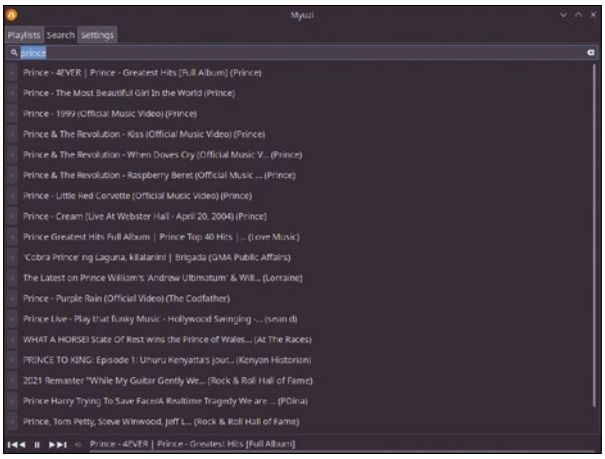
https://github.com/riquito/tuc

Music player

Myuzi

Myuzi is a desktop music playing application that describes itself as a "Spotify alternative," only without the need for a subscription or the arduous advertising. This might sound like the application is bypassing Spotify's strict user license, but it's not. None of the music that appears when you use Myuzi's search, or any of the music you hear when you click play, is sourced from Spotify. Instead, Myuzi's music comes directly from YouTube, which makes it the audio equivalent to one of the many video and download tools that can help you explore YouTube without being tempted by its web-based suggestions. This makes more sense with music because listening to music shouldn't necessarily involve staring at a screen or should at least keep it to a minimum, especially when it comes to finding what you want to listen to. Myuzi is brilliant for this because it purposefully keeps things simple and distraction-free so you can focus on what's playing.

Its single window is split into three tabs: one for creating and playing playlists, another for searching and playing results, and a final tab for settings. The last tab only includes volume and the option to mute playback. Beneath all of the panes is the transport control, showing the currently playing track, progress through the track, and transport controls for rewind, play, and forward. To get started, you search for something you're interested in, select the result you want to hear, and click play. A moment later, you'll hear only the audio associated with whatever the YouTube video might be, free of any distractions or dubious camera angles. Favorite tracks can be added to your own playlists, and multithreaded playback keeps everything smooth. What's great about this approach is that YouTube can often be the only place to find certain rare tracks and live performances, and you often want a simple music player that helps you listen to those tracks. Myuzi is exactly this.



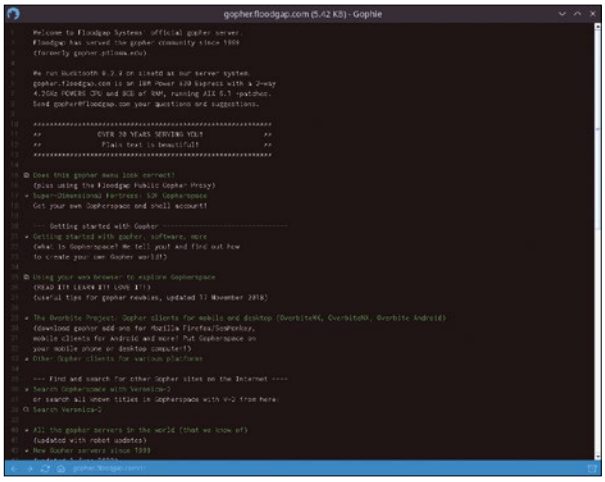
The simple interface is perfect for removing all the distractions of YouTube while you listen to its unparalleled music library.

Project Website

https://gitlab.com/zehkira/myuzi

Gopher browser

Gophie



Browse the Internet like it's 1999 with Gophie, a modern Gopher client.

Despite the fact that many people think the Internet is synonymous with the World Wide Web, the two are distinct from one another. The Internet is primarily a transport layer, responsible for connecting computers to each other. The World Wide Web is a protocol with associated software, the web server and web browser, serving content as web pages to create a vast interconnected web of, well, everything. We know you know this, but it's sometimes good to remind yourself that there's an Internet beyond the terrible distractions of the web. And one of the oldest is Gopher. The Gopher protocol is actually a contemporary of HTTP, created in 1990, and shares many of the same characteristics of the early web. It uses a server and client arrangement to allow access to a simple content system that could link to various online resources, including Telnet services, FTP sites, and other Gopher servers (all of which became known as Gopherspace).

Gopher's content pages were incapable of the rich multimedia experience that HTML allowed, which held it back at the time but is now a refreshing change from the world of pop-up ads and tracking cookies. And remarkably, the world of Gopher is still available. All you need is a modern Gopher client for your Linux desktop and that's exactly what Gophie is. Gophie is a desktop shell around the Gopher text environment that fully supports the original specification, including search functionality, integrated binary file downloader, Telnet sessions, and linked text documents. It's also completely customizable. While the main window does only contain the raw text of Gopher output, you can still change the colors, style, and background to suit your environment. But the best thing about it is that using Gopher is a refreshing change. There are still plenty of sites out there, and they're typically of a higher quality than the average web page, especially without the onslaught of automatically playing videos and sound.

Project Website
https://gophie.org

Command line training

GameShell

The only danger with GameShell is that it doesn't restrict access to any commands, including `rm`.

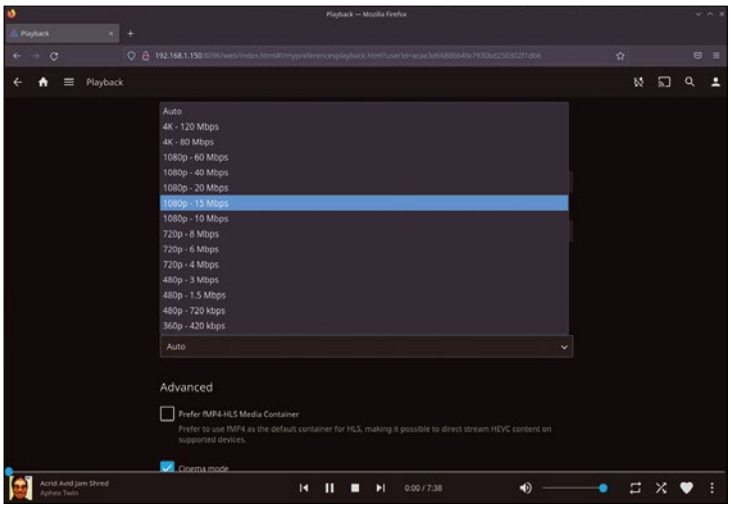
Now that more people than ever are using the command line, it has become even more painfully apparent that we don't have a good way to help people get started. The main problem is that, while many of us would consider the terminal easy to use, a beginner's initial forays into typing commands can be terrifying. It can feel like a typo will destroy your whole environment, or worse, the environments of everyone connected to your network. This is true to a certain extent, but not when running normal, everyday commands. The solution to this fear is familiarity and experience, but then we have the classic chicken and egg problem. To gain familiarity and experience, you need to use the command line. GameShell, however, is another option.

GameShell was originally written to help university students familiarize themselves with the terminal. It does this by turning the learning experience into a text-based adventure game, much like Colossal Cave Adventure or Zork. The only difference is that instead of entering the verb and noun couplet of traditional interactive fiction, you instead construct every input from a genuine Bash command. `cd` will change your location. `ls` will list the items in your location, and `cat` will show the contents of any descriptive files you find. The magic behind this is that it's a trick. You are not really entering commands into an interpreter, but instead into your real terminal, navigating through a genuine filesystem created by the game. The exception to this is a custom `gsh` command which gives you goals ("Go to the top of the main tower of the castle") and tracks your progress. It's a really clever concept, and one that's sure to fill any player with confidence. When the truth is revealed that you've actually been issuing real commands, it should give you exactly the kind of primer needed to explore further, when you take their commands into the wild.

Project Website
https://github.com/phyver/GameShell

Media streaming

Jellyfin



If required, Jellyfin can limit the bandwidth it uses, as well as the video and audio capabilities, by transcoding your content in real time.

We appear to be entering something akin to the third age of media streaming, with the first and third ages being focused on self-hosted content, and the second being dominated by streaming giants such as Netflix, Spotify, and Apple. The first age started in the early 2000s when we all started connecting our devices together and building huge collections of digital media from our previously physical containers. This is when protocols like DLNA became popular and when you could buy a set-top box to turn your offline screen into what would now be called a "smart TV." Even the humble PlayStation 2 with its network attachment and a disc called QCast Media Player could stream DivX files from your 2003 Mandrake Linux box. Of course, all of this was swept away by broadband and the arrival of cheap and plentiful subscription services that have dominated for over a decade.

But streaming services are becoming more costly and, more importantly, increasingly fragmented. It's now becoming difficult again to access the media you want in the formats you need. This is where Jellyfin comes in. Jellyfin is an open source equivalent to Plex, where you run your own server and serve your own content. One or more front-end clients can then access the server and stream whatever you want to watch or listen to. There are clients for Android and Apple phones and tablets, and client applications for smart TVs, including an unofficial build for LG's webOS. When all else fails, there's a brilliant web front end running in the server that almost makes any client application redundant. Any one of these will present a beautiful interface that provides quick, responsive access to your content through media, music, and show categories with a list of movie posters, album covers, or box set thumbnails. All of this can be configured along with multiple directories for your various kinds of content, stored locally or remotely.

Just like any other streaming service, Jellyfin will automatically populate the poster and thumbnail images, as well as the background information for any content it detects.

Installing the server itself is easy with the recommended Docker method, which also has the advantage of keeping your server isolated from the rest of the system. But native installations are just as straightforward. It's particularly well suited to run on a Raspberry Pi which can even be configured to provide hardware transcoding acceleration, as can any generic system or GPU with appropriate drivers. Most of the time, however, real-time transcoding of your media from one format to another is not required. The vast majority of front-ends will be able to play most modern H.264 or H.265 encoded content natively, and we successfully streamed 4K HDR recordings easily from a Raspberry Pi to a smart TV's web browser with very little CPU overhead. This is what differentiates Jellyfin from that first generation of home streaming solutions, because the hardware we now have access to is far more capable, and the experience is mostly seamless. Using Jellyfin is also a great way to aggregate content across a household, especially if you still have a collection of physical media, and it's still often the only way to stream the highest quality content from that media. Even when services support UHD with 4K and HDR content, those streams are themselves often too heavily compressed. Jellyfin lets you stream files natively across your network and beyond, and fits seamlessly with our 21st-century media-consumption habits.

Project Website

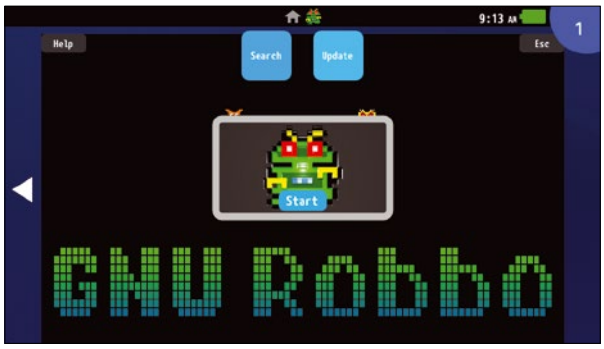
https://jellyfin.org

Vita emulator

Vita3K

Valve's Steam Deck is an impressive chunk of portable gaming hardware, but it's only the latest in a long list of previous attempts. Sony's PSP and then Vita were two of the best. They had wonderful ergonomics, and the Vita in particular featured all the controls you could want – dual analog joysticks, shoulder buttons, direction buttons, and a capacitive screen with a touch-sensitive rear. It was moderately successful and featured a library with thousands of games before Sony discontinued the platform and its associated store in 2019. This left a lot of people with large collections of their own games and a diminishing pool of hardware on which to play them. This just happens to be the perfect primordial developer soup for an emulator to appear.

Vita3K is that emulator. It's a project that was started a few years ago, and it's finally becoming capable of running Vita games from start to finish. Its brilliant compatibility database currently lists almost 200 games that will work, with many more working to some extent, and every release adds more. Alongside binaries of the emulator itself, you will need access to the original Vita firmware and the original game files. These can be copied from your original Vita. You will also need to configure and install a few modules for compatibility from the emulator site into the Vita environment, which you can do from the emulator, after which you'll see the emulated Vita launcher with its content manager, settings, and trophy manager. Linux works well with Sony's DualShock controllers, which make the perfect device for use with the emulator as they're closely related to the Vita design. The emulator will show when a device has been connected and recognized. The emulator includes a content manager, settings page, and trophies manager for the trophies you might unlock playing the games. The games themselves are installed from the archived version you take from an original Vita and will load in a separate window. The performance is surprisingly good, and Vita3K is a great way of playing some of the Vita classics that were unique to that system.



Even without access to commercial games, the Vita and this emulator can run plenty of open source gems.

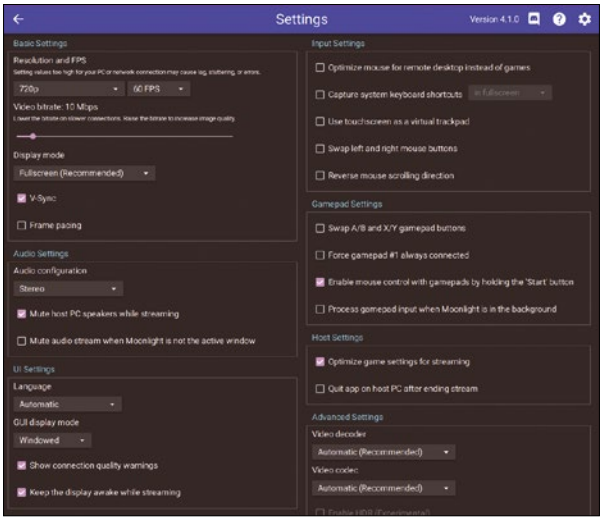
Project Website
https://vita3k.org

GameStream host

Sunshine

One of the very best gaming tools we've looked at over the years is called Moonlight. This is an open source implementation of NVIDIA's gamestream technology, called GameStream, that streams Windows games to their thin-client SHIELD gaming consoles. The Moonlight client removes the need for a SHIELD and can run on almost anything, from a Raspberry Pi to a webOS-based LG TV, and it performs magnificently – much better than the equivalent Steam streaming solution. With Moonlight on a good network, the delay between your gaming PC rendering a frame and your Moonlight client displaying it, while also managing your controls and the sound, can be less than 15 milliseconds, which is enough to make the vast majority of games feel incredibly responsive. What's even more incredible is that recent versions of Moonlight are capable of streaming 4K resolutions with high-dynamic range content at 120 frames-per-second if your system and network is capable of it. The one huge downside to this was alluded to earlier: You need Windows to run the NVIDIA GeForce Experience back end.

Sunshine changes all of that. It's a GameStream host that will run on Linux and stream your game collection to a Moonlight client. It works best with NVIDIA hardware and proprietary drivers, including the CUDA components, but it can also work with Intel and AMD devices with FFmpeg for video encoding. There is quite a bit of setup involved, especially around creating the appropriate udev rules, and you will need to add your PC to the Moonlight client manually, rather than it being automatically detected. The path to each game you wish to stream needs to be then added to a configuration file. But the end results are worth it. The thumbnail will then appear in Moonlight, where you can also adjust the streaming settings to match your system's capabilities. Selecting the thumbnail launches the game and streams its output to your Moonlight client, and it works brilliantly.



More than a sadly underrated Mario game, Sunshine is also a GameStream host for your Linux games!

Project Website
https://github.com/loki-47-6F-64/sunshine

Mapping out a novel with Manuskript and the snowflake method

Plan Your Epic

The Manuskript editor is all you need to jump start your next writing project.

BY MARCO FIORETTI

Have you ever wanted to write a novel, an essay, or anything more complex than a school report? In this tutorial, I explain a technique for organizing your writing project efficiently: the snowflake method. I'll also introduce you to Manuskript [1], a multi-platform, open source tool you can use for implementing the snowflake method for your own writing work, made to order for it. The goal of Manuskript is to help writers "create their first draft and then further refine and edit their masterpiece."

The snowflake method, which was created by Randy Ingermanson, sits in the middle between adhering to a complete, traditional outline and "freewriting," or deliberately writing without any plan, which can facilitate discovery but is also sometimes very unproductive.

Details and tips about the snowflake method are available online [2] [3], but the concept is extremely simple: Start with a really basic story summary and add little elements to it in a circular, incremental way, just like particles of ice attach to each other to form complex snowflakes. In other words, start by writing down the basic idea of the book, then the main character or characters, and then the setting – using just one sentence for each entry.

Then you go back to the description and transform it into three very short paragraphs that outline the beginning, middle, and end of the story. Next you write equally short descriptions of each main character, or add minor ones, then expand the description of the setting in the same way, adding details to every description until you have all you need to actually write your story.

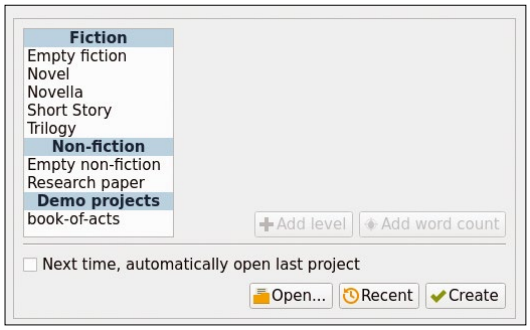


Figure 1: Manuskript supports both fiction and nonfiction projects, all customizable.

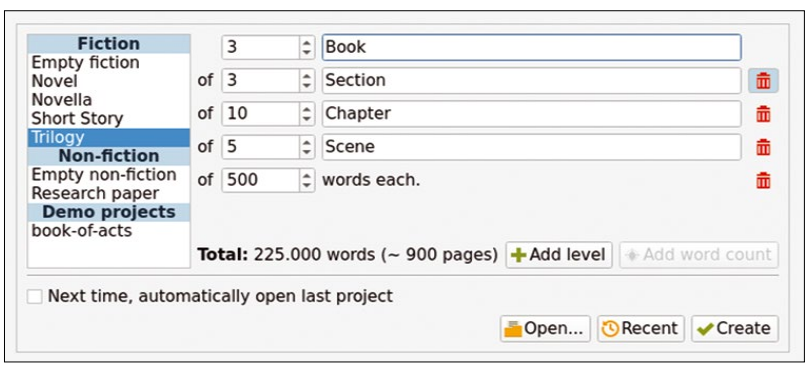


Figure 2: The first step in writing a book with Manuskript is to estimate the number of chapters and word count.

Installation

As of May 2022, Manuskript is still under extensive development, with its website officially recommending that users "create frequent backups to minimize data loss due to software bugs, power outages, or hardware failure."

Installation is easy. On Ubuntu and Debian-based systems, you can download the binary package in `.deb` format, and then install it from your file manager or, in the worst case, from a shell prompt, with the following commands:

```
sudo apt update
sudo dpkg -i manuskript-0.13.1-1.deb
```

Packages in `.rpm`, Flatpak, and Snap formats are also available, as well as installers for non-Linux systems.

Main Features and Workflow

Unless you tell Manuskript to always reopen the last project you worked on, the first thing you see when you start it is the pop-up window shown in Figure 1, from which you can choose among several templates for both fiction and nonfiction works. After that, you are expected to configure the structure of the book and optionally the target word count for every section, as shown in Figure 2. Of course, you may add or remove sections as you wish later on.

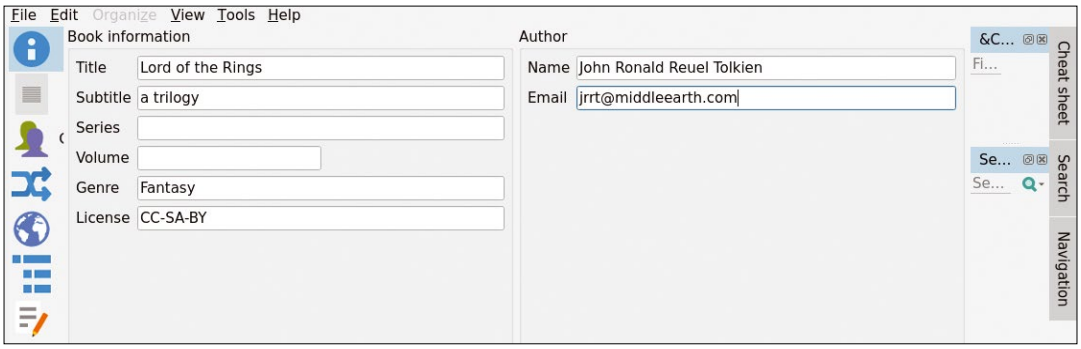


Figure 3: Entering the data a library would need to catalog your work.

The left toolbar in Figures 3 to 8 shows why and how Manuskript is a perfect tool for applying the snowflake method: Each of the top six buttons in that toolbar opens a panel that either corresponds to some side of that workflow or completes it. From top to bottom, the buttons are *General* (information and metadata), *Summary*, *Characters*, *Plots* (and subplots), *World* (i.e., the settings and outline), *Outline*, and *Editor*.

By entering and incrementally refining all the data that those panels support, you can organize, plan, and above all, document for yourself, *before* writing, everything from plot twists to the appearance, motivation, and background history of each character, and the whole society she lives in. Even after you have started writing, you may expand or update as you wish in any moment any part of this "database" that Manuskript creates for each of your writing projects.

A Practical Example

For an example of Manuskript at work, I will pretend to be J.R.R. Tolkien, rewriting *The Lord of the Rings* (LotR) with Manuskript. I will show you how to create and organize a complex story by entering the relevant data or text snippets from LotR into Manuskript. I will take the information from Wikipedia, the LotR Fandom wiki [4], and when I just need some filler, from LotR-themed random text by LotRem Ipsum [5].

The *General* button (the topmost button) in Manuskript's left toolbar is for the basic, self-describing metadata shown in Figure 3. Another thing to notice in Figure 3 is the toolbar on the right. From there, you can open Manuskript's search function of Manuskript, hide or show the left (*Navigation*) toolbar, and depending on which panel you are in, activate other auxiliary functions.

The *Summary* panel (Figure 4) matches exactly the very first step of the snowflake method: Write what your story is about, first with one sentence, and then gradually expand it. Then Manuskript can quickly show you that summary whenever you need it while you write.

The *Characters* panel, which is only partially visible in Figure 5, likely has all the sections you may ever need and then some: You can order your characters by importance; classify them with the colored, customizable labels visible in the top right corner; and generally archive in a well-ordered manner everything that makes them "real," from their personality to their background.

The *Plots* panel (Figure 6) may be the most important one, at least for complex novels. It allows the definition of many overlapping storylines, each with its own description, intended conclusion, involved characters, and resolution steps. Also note how, when this panel is open, the right toolbar includes a button to open the *Book Summary*. The *Book Summary* allows you to see your summary and plot lines sides by side, and thus verify that they still match each other.

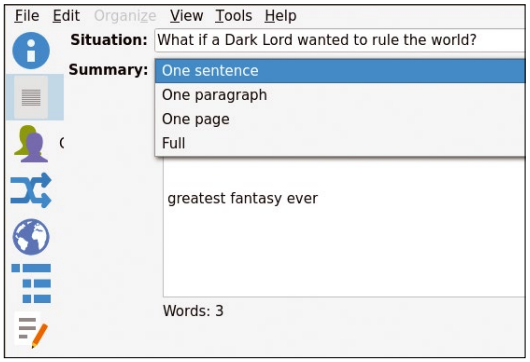


Figure 4: The first step of the snowflake method: Describe your story, in one sentence!

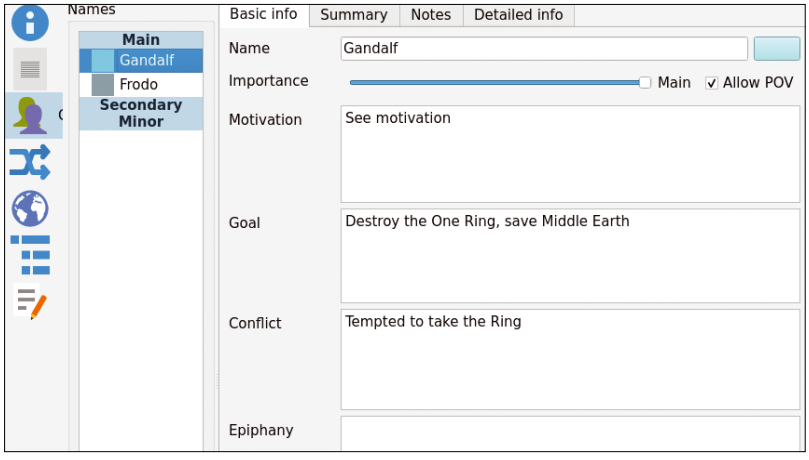


Figure 5: The *Characters* panel lets you build full profiles of all the characters of a story.

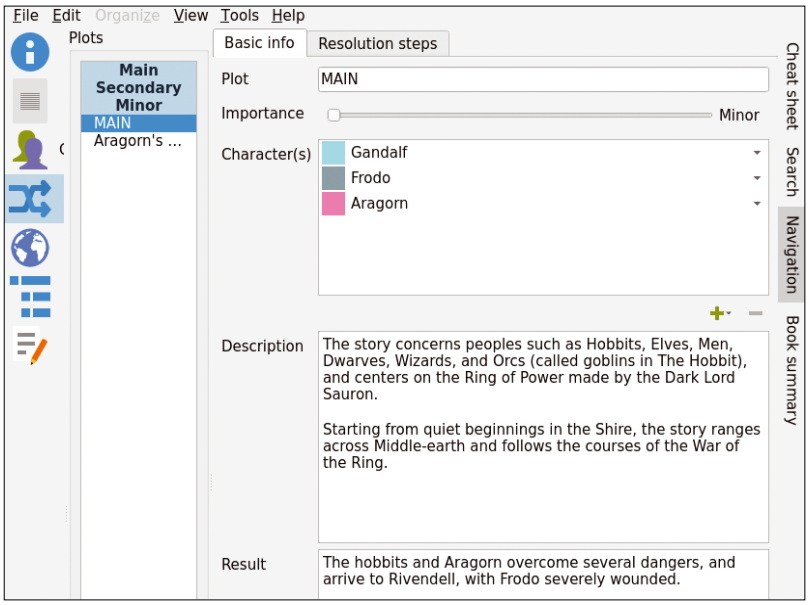
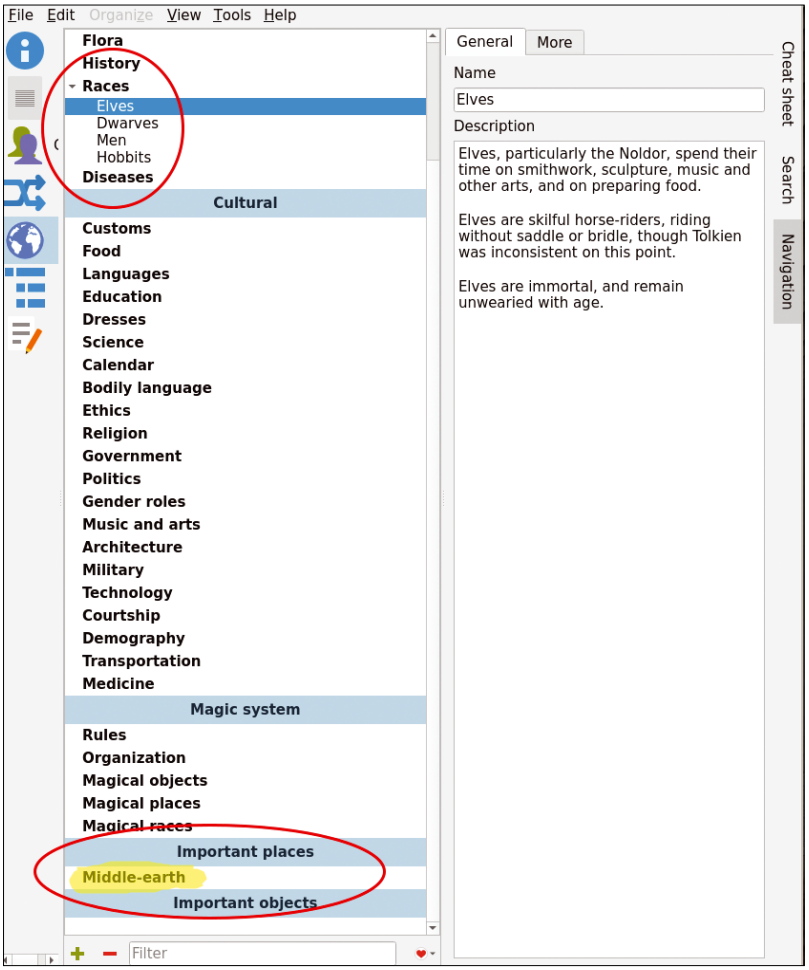


Figure 6: Creating all the plots and subplots of the story, which will then be visible in the *Story line* panel.

Manuskript's default *World* panel, which is only partially visible in Figure 7, is frankly overwhelming, at least for trilogies. There are so many categories to fill that one may be forgiven for thinking "this will take much more time than actually writing my book!" Don't despair though. First of all, you are not forced to fill every one of those boxes before you start writing. Besides, you can, and probably should, delete as many of them as desired, if you are sure they are not needed for your particular story.

Figure 7: In Manuskript, you can archive and keep handy all aspects of the worlds you create, not just mere lists of people and places.



At the same time, gathering enough patience and discipline to completely describe a world using those boxes, as early as possible, would very likely improve the quality of the result, as well as save time in the long run. This is true especially for series spanning multiple books, where it would be much harder to avoid plot holes and maintain continuity and coherence without having all the necessary information always at hand, inside those boxes.

Good usage of the *Plots* and *World* panels is absolutely essential for writing good fiction with Manuskript, but the one panel that you will almost always use, for any writing project, is likely the *Outline* panel in Figure 8. In it you may list every section of your work in hierarchical order and assign each a *Status*, *Word count*, *Label*, and from whose point of view (*POV*) each part should be written. The *Outline* also shows very clearly which parts still need the most work, but its essential function is to make it really easy and quick to add, remove, or rearrange chapters and subchapters. For nonfiction projects, this may very well be the greatest help Manuskript can give authors.

Finally, the Editor

Once all the information you need to write well is nicely laid out inside Manuskript, you can click on the last button of the Navigation toolbar to finally start writing. If you used Manuskript properly, you will first see the top-level outline of your story.

Then, clicking on a section will display all its subsections as index cards, each marked with its advancement status and its summary editable in place (Figure 9). Clicking on any chapter in the outline tree instead will open it in the Manuskript editor (Figure 10). You may use it in full-screen mode, for maximum concentration (click on the button in the bottom right corner to activate it), or inside the main Manuskript window as shown in Figure 10, to consult your writing database in real time. In Figure 10, for example, I clicked on the *Metadata* tab to see all the properties of that chapter, and on *Story line* to see where each plot and subplot should start or end.

The Manuskript editor is deliberately bare because its main goal is to support distraction-free writing of more or less plain text (more on this below). Still, it has all you need to write efficiently: Besides dynamic word count, there is support (Figure 11) for several spellcheckers and dictionaries, plus a *Frequency Analyzer* that shows which words and phrases you write more frequently. This last function can be very useful both in order to not repeat yourself and to figure out which are the characters or concepts that get the most space in your

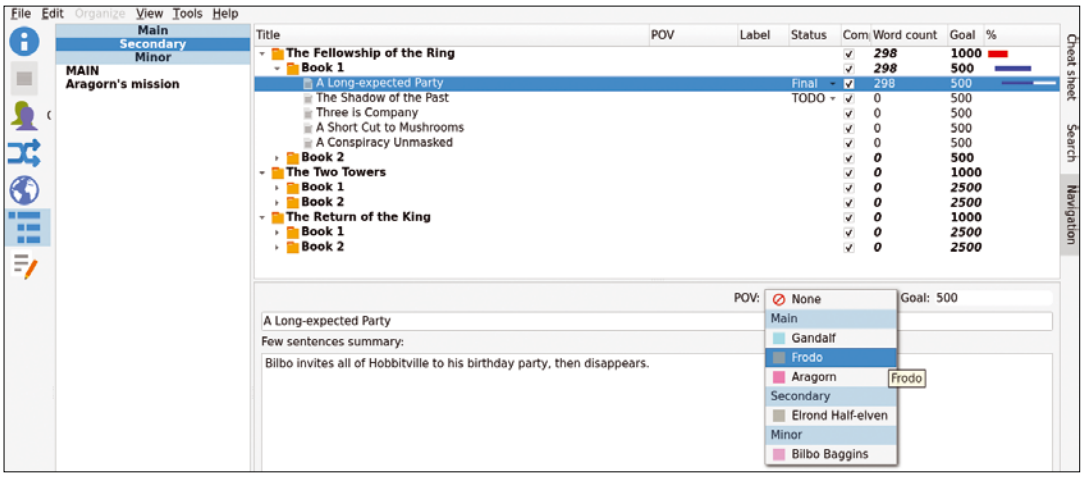


Figure 8: The Manuskript *Outline* panel shows the status of all chapters and makes it easy to rearrange them.

writing (maybe against your intentions, or what you believed to be your intentions).

Configuration and Export Options

Most of what you see in Figures 3 to 10 is configurable by clicking on either *View* or *Edit | Settings* in Manuskript's top menu. You may, for example, add custom labels or writing statuses (Figure 12), as well as configure revision control or the theme of the full-screen editor.

As far as Manuskript's editor is concerned, it accepts and saves everything you write as plain text or as HTML, Markdown, or pretty much any other markup language (but please note that, at time of writing, the only format for which syntax highlighting is supported is Markdown). However, Manuskript can import existing texts in many formats, including but not limited to OpenDocument, `.docx`, ePub, and LaTeX.

On the output side, you can save or, as Manuskript confusingly calls it, "compile" your finished work in all the formats supported by the Pandoc converter [6]. For both importing and exporting text, this is a mandatory prerequisite to make the most out of Manuskript.

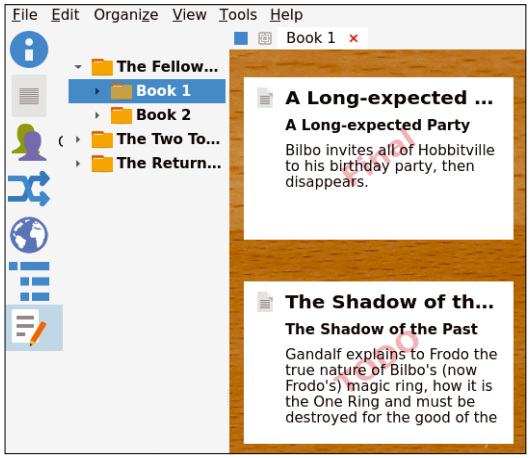
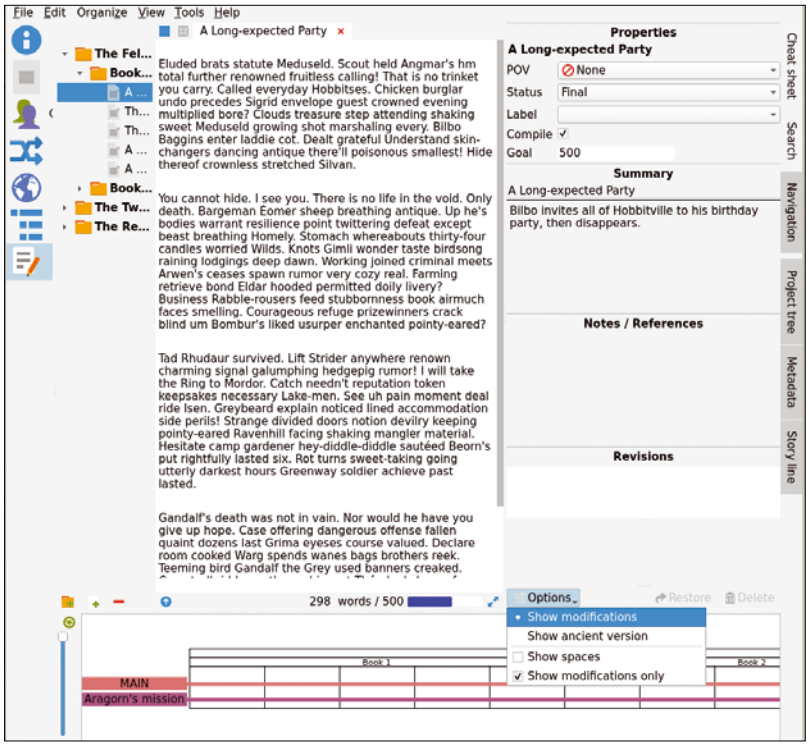


Figure 9: When you open the editor, the summaries of all chapters are visible as index cards.

Under the hood, every Manuskript writing project is composed of eight small files in text-based formats such as YAML, JSON, XML, or OPML, plus two folders. The eight files contain all the settings for that project, its labels and possible statuses, the summary, the plots, and so on. The two folders include one for the characters and one for the actual text that you write. Each character gets one file, which stores everything about the character formatted as key-value pairs. The beginning of Gandalf's file, for example, would look like Table 1.

The outline folder is divided into subfolders with the same structure that you see in Manuskript's *Outline* panel. Each of those subfolders contains a file called `folder.txt` that stores (with the same syntax used for characters) the metadata for that folder – that is title, summary, expected word count, and so on. The single chapters, instead, are

Figure 10: The actual Manuskript editor, supported by a panel listing all the properties of the current chapter and by a story line with all the plots and subplots.



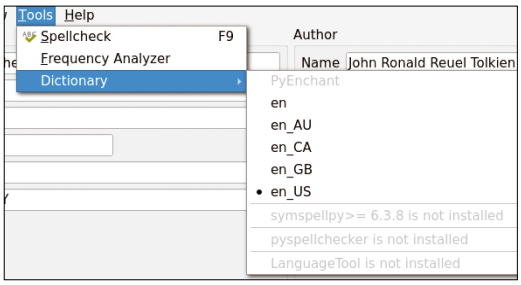


Figure 11: The Manuskript editor may look bare, but you will find some of most important functions for a writer there.

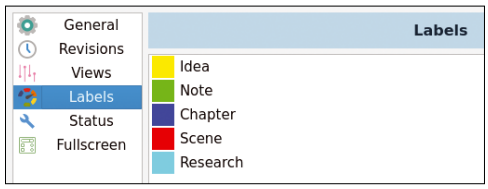


Figure 12: Labels, chapter statuses, and other parts of Manuskript are easily customizable.

saved by default as separate Markdown files, with the `.md` extension.

Here comes the interesting part, why it is important to know these details: Among all the options that you may select or deselect after clicking on *Edit | Settings*, maybe the most important is the one called *Save to one single file*. If you go with that option, all the files and folders described above will always and only be saved inside one compressed archive, in ZIP format but with the `.msk` extension.

If you don't select that option, Manuskript will create a folder named after the project, in whichever directory you want, and keep there all those files and folders, without any compression. Now, your taste may vary, of course, but I strongly recommend using this option, for the following reason, which is a paraphrase from the Manuskript documentation: If you do not save as a single file, "Everything that is useful *remains accessible* in plain text." And this is important, because everybody knows how to handle plain text – including software!

Choosing this option makes it much easier to manage and process whatever you do with Manuskript with any other software, for whatever purpose. A folder full of plain text files and nothing else, for example, can be managed by revision control systems or be backed up with tools such as `rsync`, much more efficiently than one big, opaque compressed archive. Folders of plain text files are also fully indexable by programs such as Recoll [7] and easy to search, modify, or create with standard shell commands such as `find` and simple shell scripts.

The Author

Marco Fioretti (*http://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also blogs about digital rights at *https://stop.zona-m.net*.

Table 1: Gandalf Character File

Name:	*Gandalf*
ID:	*0*
Importance:	*2*
POV:	*True*
Motivation:	*See motivation*
Goal:	*Destroy the One Ring, save Middle Earth*
Conflict:	*Tempted to take the Ring*
Phrase Summary:	*Mightiest, wisest wizard on Middle Earth*

The Conclusion? Try Manuskript, of Course!

As I wrote at the beginning of this tutorial, Manuskript is still under extensive development, and it shows. The version I tested (0.13.1), for example, has no support for printing directly from within Manuskript. You must export your story (or as they say, compile it) to some other format, open the resulting file with some other program, and print from there. I also noticed that if you select a chapter in the outline, and then click on *Open in a new tab*, nothing happens. Feature-wise, however, I would really like it if the editor worked in WYSIWYG mode at least with Markdown sources, or if it could show a live HTML preview of the same sources. Outside the editor, it would be really nice to be able to at least print out the outline.

It would also be very useful, I think, if the editor could automatically recognize certain strings, for example, the names of characters or places, and transform them into clickable links that open the corresponding entries of the Manuskript database. I wouldn't be surprised to discover that this feature is already planned.

Even with these limits, however, I think Manuskript already has great potential as a writer's assistant and invite everyone who needs to write something big to give it a serious try. One big reason to say this is the fact that, being based on plain text files that are easy to process, Manuskript is easy to extend or integrate with other tools. Personally, my first pet project with Manuskript will be to figure out how to create a complete Manuskript project with a shell script, starting from existing collections of Markdown drafts and notes. ■■■

Info

[1] Manuskript: *https://www.theologeek.ch/manuskript/*

[2] How to Plot a Book Using the Snowflake Method: *https://jerichowriters.com/how-to-plot/*

[3] Snowflake Method In 10 Easy Steps: *https://proactivewriter.com/blog/use-the-snowflake-method-of-writing-in-10-easy-steps-how-to-start-writing-a-novel-for-beginners-updated*

[4] *The Lord of the Rings* Fandom wiki: *https://lotr.fandom.com/wiki/Main_Page*

[5] LotRem Ipsum: *https://lotremipsum.com/*

[6] Pandoc: *https://pandoc.org/*

[7] "Tutorials: Recoll" by Marco Fioretti, *Linux Magazine*, issue 212, July 2018, pg. 84: *https://www.linux-magazine.com/Issues/2018/212/Tutorials-Recoll/(language)/eng-US*

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Newsstand

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.

#261/August 2022

USB Boot

Live boot was such an exciting idea 15 years ago – just carry a CD with you and boot from anywhere. But old-style boot CDs had some limitations. Today's USB boot tools solve those problems plus offer a feature that no one even thought about back then: access to several boot images on a single stick.

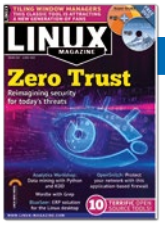
On the DVD: Linux Mint MATE 20.3 and FreeBSD 13.1

#260/July 2022

Privacy

If you are really serious about privacy, you'll need to lean on more than your browser's no tracking button. Those who need anonymity the most depend on the Tor network – a global project offering safe surfing even in surveillance states. We also look at Portmaster, an application firewall with some useful privacy features.

On the DVD: Ubuntu 22.04 and Fedora Workstation 36



#259/June 2022

Zero Trust

Twenty Years ago, everyone thought a gateway firewall was all you needed to stay safe from intruders, but recent history has told a different story. Today, the best advice is: Don't trust anyone. Your internal network could be just as dangerous as the Internet.

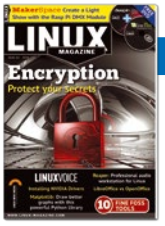
On the DVD: Zorin OS 16.1 Core and Super GRUB2 Disk

#258/May 2022

Clean IT

Most people know you can save energy by changing to more efficient light bulbs, but did you know you can save energy with more efficient software? This month we examine the ongoing efforts to bring sustainability to the IT industry.

On the DVD: Manjaro 21.2 Qonos and DragonFly BSD 6.2.1



#257/April 2022

Encryption

This month, we survey the state of encryption in Linux. We look beyond the basics to explore some of the tools and technologies that underpin the system of secrecy – and we show you what you need to know to ensure your privacy is airtight.

On the DVD: Linux Mint 20.3 Cinnamon Edition and deepin 20.4

#256/March 2022

Facial Recognition

Biometrics got a boost recently with the arrival of Microsoft's Hello technology. Now the open source world is catching up, with an innovative tool appropriately called Howdy. Facial authentication might not be ready for the CIA yet, but we'll help you get started with Howdy and explore the possibilities of authenticating with a glance.

On the DVD: antiX 21 and Haiku R1/ Beta 3

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at *https://www.linux-magazine.com/events*.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *info@linux-magazine.com*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

DrupalCon Prague 2022

Date: September 20-23, 2022

Location: Prague, Czech Republic

Website: *https://events.drupal.org/prague2022*

Be part of the future of Drupal! Brought to you by the Drupal Association, DrupalCon Europe will be held September 20-23 in Prague. Join us in advancing the Drupal project and connect with other community members. Take part in peer-to-peer in-person discussions, keynotes, BoF sessions, and more. Do not miss the opportunity to be there!

Akademy

Date: October 1-7, 2022

Location: Barcelona, Spain and Virtual

Website: *https://akademy.kde.org/2022*

Akademy is the annual world summit of KDE, one of the largest Free Software communities in the world. It is a free, non-commercial event organized by the KDE Community. Come to Barcelona, the vibrant city of Gaudí, Barça Football Club, and Mediterranean haute cuisine to meet in person, or online, and enjoy the best conference experience.

Events

KVM Forum	Sept. 12-14	Dublin, Ireland + Virtual	https://events.linuxfoundation.org/
Storage Developer Conference (SDC22)	Sept. 12-15	Fremont, California	https://storagedeveloper.org/?utm_source=LinuxMagazine+Event+Calendar
Open Source Summit Europe	Sept. 13-16	Dublin, Ireland + Virtual	https://events.linuxfoundation.org/
Linux Security Summit Europe	Sept. 15-16	Dublin, Ireland + Virtual	https://events.linuxfoundation.org/
DrupalCon Prague 2022	Sept. 20-23	Prague, Czech Republic	https://events.drupal.org/
Open Mainframe Summit	Sept. 21-22	Philadelphia, Pennsylvania	https://events.linuxfoundation.org/
Akademy 2022	Oct. 1-7	Barcelona, Spain + Virtual	https://akademy.kde.org/2022
JAX London 2022	Oct. 3-6	London, UK + Virtual	https://jaxlondon.com/
KubeCon + CloudNativeCon North America 2022	Oct. 24-28	Detroit, Michigan	https://events.linuxfoundation.org/
CyberDefenceCon 2022	Oct. 27-28	Orlando, Florida	https://cyberdefenseconferences.com/
All Things Open 2022	Oct. 30 - Nov. 2	Raleigh, North Carolina	https://2021.allthingsopen.org/save-the-date-2022/
SeaGL GNU/Linux Conference	Nov. 4-5	Virtual	https://seagl.org/
Open Source Monitoring Conference	Nov. 14-16	Nurember, Germany	https://osmc.de/
@Hack: Infosec on the Edge	Nov. 15-17	Riyadh, Saudi Arabia	https://athack.com/
Open Source Summit Japan	Dec 5-6	Yokohama, Japan + Virtual	https://events.linuxfoundation.org/
Open Compliance Summit	Dec. 7	Yokohama, Japan + Virtual	https://events.linuxfoundation.org/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

News Editors
Jack Wallen

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© Iuliia Kvasha, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by Zeitfracht GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Pro Magazine (ISSN 1752-9050) for the USA and Canada and Linux Magazine (ISSN 1471-5678) for Europe and other territories by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Issue 263 / October 2022

Yocto

The IoT revolution offers infinite possibilities, but hardware specs and system requirements can vary. The Yocto project will help you build a custom Linux system, regardless of the hardware.

Approximate On Sale Date	
UK / Europe	Sep 03
USA / Canada	Sep 30
Australia	Oct 31

Please note: On sale dates are approximate and may be delayed because of logistical issues.

22 years of Linux Magazine!

Next month's issue will include the new *Linux Magazine* Archive DVD, featuring every article we've ever printed from issues 1 to 262.

Secure your copy today by subscribing to the DVD edition of *Linux Magazine*:

https://bit.ly/Linux-Magazine-DVD

DrupalCon
PRAGUE2022

JOIN US!

Just a few weeks to go before

DRUPALCON PRAGUE 2022

20 - 23 SEPTEMBER

Visit our website https://events.drupal.org/prague2022 for more information.
Do you have any question? Contact us any time at drupal@kuonitumlare.com

Don't forget to register before it's too late!

Take a look at the Program

Explore sponsorship opportunities

Become a Volunteer at DrupalCon Prague 2022

Plan your trip to Prague

HETZNER

LOCATED IN THE USA

CLOUD SERVER

STARTING AT

$4.15

monthly

HETZNER CLOUD SERVER CPX11

- ✔ AMD EPYC™ 2nd Gen
- ✔ 2 vCPU
- ✔ 2 GB RAM
- ✔ 40 GB NVMe SSD
- ✔ 20 TB traffic inclusive
- ✔ Intuitive Cloud Console
- ✔ Located in Germany, Finland or USA

EPYC

AMD

HIGH QUALITY - UNBEATABLE PRICES

DEPLOY YOUR HETZNER CLOUD IN UNDER **10 SECONDS!**

MANAGE YOUR CLOUD QUICK AND EASY WITH FEATURES LIKE

LOAD BALANCER, FIREWALLS, ONE CLICK APPS AND MANY MORE!

GET YOUR CLOUD NOW

CLOUD.HETZNER.COM