A decorative border composed of a network of nodes and lines, forming a square frame with rounded corners. The nodes are represented by small circles, and the lines are thin, connecting the nodes in a grid-like pattern. The corners of the border feature a more complex, fan-like structure of nodes and lines.

OPEN SOURCE YEARBOOK

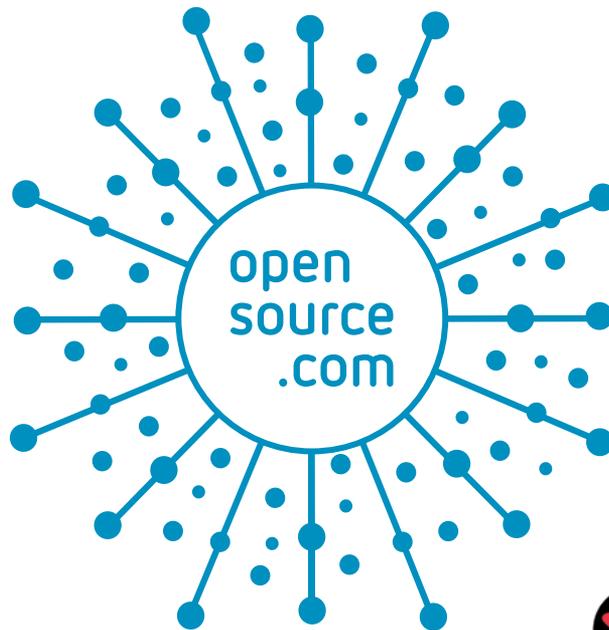
2016

Opensource.com publishes stories about creating, adopting, and sharing open source solutions. Visit [Opensource.com](https://opensource.com) to learn more about how the open source way is improving technologies, education, business, government, health, law, entertainment, humanitarian efforts, and more.

Submit a story idea: <https://opensource.com/story>

Email us: open@opensource.com

Chat with us in Freenode IRC: [#opensource.com](https://freenode.net)



SUPPORTED BY RED HAT

..... AUTOGRAPHS:

..... AUTOGRAPHS:

7 big reasons to contribute to Opensource.com:

- 1 Career benefits:** “I probably would not have gotten my most recent job if it had not been for my articles on Opensource.com.”
- 2 Raise awareness:** “The platform and publicity that is available through Opensource.com is extremely valuable.”
- 3 Grow your network:** “I met a lot of interesting people after that, boosted my blog stats immediately, and even got some business offers!”
- 4 Contribute back to open source communities:** “Writing for Opensource.com has allowed me to give back to a community of users and developers from whom I have truly benefited for many years.”
- 5 Receive free, professional editing services:** “The team helps me, through feedback, on improving my writing skills.”
- 6 We’re loveable:** “I love the Opensource.com team. I have known some of them for years and they are good people.”
- 7 Writing for us is easy:** “I couldn’t have been more pleased with my writing experience.”

Email us to learn more or to share your feedback about writing for us: <https://opensource.com/story>

Visit our Participate page to more about joining in the Opensource.com community: <https://opensource.com/participate>

Find our editorial team, moderators, authors, and readers on Freenode IRC at #opensource.com: <https://opensource.com/irc>

Twitter @opensourceway: <https://twitter.com/opensourceway>

Google+: <https://plus.google.com/+opensourceway>

Facebook: <https://www.facebook.com/opensourceway>

Instagram: <https://www.instagram.com/opensourceway>

IRC: #opensource.com on Freenode

All lead images by Opensource.com or the author under CC BY-SA 4.0 unless otherwise noted.

Dear Open Source Yearbook reader,

In 2015, Opensource.com published the first Open Source Yearbook [1], a collaboration with open source communities to collect a diverse range of stories from the year. Thanks to contributions from more than 25 writers, the 2016 edition is even bigger and highlights more than 100 organizations, projects, technologies, and events. Here are a few of the many individuals who help create the 2016 Open Source Yearbook:

- Anderson Silva—Release Engineer in IT at Red Hat
- Anna Morrow—Marketing Manager at No Starch Press
- Ben Cotton—Meteorologist, high-performance computing engineer, technical evangelist at Cycle Computing, and Opensource.com community moderator
- D Ruth Bavousett—Perl Developer at cPanel and Opensource.com community moderator
- Daniel J Walsh—Leads the RHEL Docker enablement team
- David Both—Linux and open source advocate, Opensource.com community moderator
- Gordon Haff—Red Hat’s cloud evangelist
- Greg Kroah-Hartman—Linux kernel maintainer and a Linux Foundation fellow
- Jason Baker—Technical editor and SEO specialist on Opensource.com
- Jen Wike Huger—Opensource.com content manager
- Jeremy Garcia—Founder of LinuxQuestions.org and Opensource.com community moderator
- Jono Bacon—Community manager, speaker, author, podcaster, consultant, and Opensource.com community moderator
- Josh Simmons—Community organizer and short stack web developer who works on the Google open source outreach team and sits on the OSI board of directors
- Libby Clark—Digital Content Editor at The Linux Foundation
- Máirín Duffy—Principal Interaction Designer at Red Hat
- Nithya A. Ruff—Director of Western Digital’s Open Source Office, Opensource.com community moderator
- Paul Brown—Tech journalist and editor
- Rachel Roumeliotis—Strategic Content Director at O’Reilly Media, Inc., and a Programming Chair of OSCON, O’Reilly’s Software Architecture Conference, and Fluent
- Rich Bowen—Community Liaison for the RDO project, which is a packaging of OpenStack for CentOS/Fedora/RHEL
- Richard Fontana—Senior Commercial Counsel on the Products and Technologies team at Red Hat
- Richard Gall—Copywriter, Content Strategist, and Communications Manager at Packt
- Robin Muilwijk—Advisor Internet and e-Government, and Opensource.com community moderator
- Ruth Suehle—Community leadership manager for Red Hat’s Open Source and Standards team
- Scott Nesbitt—Writer, technology coach, and Opensource.com community moderator
- Seth Kenlon—Multimedia artist, technical writer, and former Opensource.com community moderator
- Shaun McCance—Community Documentation Liaison at Red Hat
- Shawn Powers—Associate editor for *Linux Journal* and IT trainer for CBT Nuggets
- Susan Conant—Supervising Editor, Programming, O’Reilly Media, Inc.
- Tom Callaway—Education Outreach team lead at Red Hat

Thank you to everyone who contributed to the 2016 Open Source Yearbook, and to the communities who helped create, document, evangelize, and share open source technologies and methodologies throughout the year.

Best regards,

Rikki Endsley

Opensource.com community manager

[1] <https://opensource.com/yearbook/2015>

CONTENTS

WORKING

- 10 **5 initiatives that pushed the free software envelope in Europe in 2016** Paul Brown
Take a tour of top free software news from Russia, Bulgaria, The Netherlands, Germany, and the EU in 2016.
- 12 **10 open source tools for your sysadmin toolbox** Ben Cotton
Sysadmins don't lack for options when it comes to great open source software tools. We look at a few favorites.
- 14 **7 notable legal developments in open source in 2016** Richard Fontana
Learn about a few of the many open source-related legal developments that made headlines in 2016.
- 18 **Troubleshooting tips for the 5 most common Linux issues** Jeremy Garcia
Learn how to tackle the most common challenges Linux desktop users encounter.
- 20 **What's new in OpenStack in 2016: A look at the Newton release** Rich Bowen
We round up a few of the many notable updates in the latest OpenStack release.
- 23 **Why the operating system matters even more in 2017** Gordon Haff
Operating systems don't quite date back to the beginning of computing, but expect them to be around a long time to come.

- 26 **25 things to love about Linux** Jen Wike Huger
Linux turned 25 years old in 2016, so we asked our readers what they love about Linux.
- 27 **4 hot skills for Linux pros in 2017** Shawn Powers
Which in-demand skills are you brushing up on in the new year?
- 28 **Hot programming trends in 2016** Rachel Roumeliotis
Take a look at the year's hottest languages for AI projects and containers, new languages, and more programming trends.
- 30 **50 ways to avoid getting hacked in 2017** Daniel J Walsh
Paul Simon rounded up 50 ways to leave a lover, and we round up 50 ways to secure your systems.

Best Couple of 2016 Display manager and window manager

- DAVID BOTH
- 34 Our pick for Best Couple this year is actually a pair of program types—not specific commands or programs.

COLLABORATING

- 37 **10 steps to innersource in your organization in 2017** Jono Bacon
Is your company planning to implement innersource concepts in 2017? We walk through steps for getting started.
- 40 **7 cool little open source projects that stood out in 2016** D Ruth Bavousett
We look at a few innovative open source projects that stood out in 2016.
- 44 **9 lessons from 25 years of Linux kernel development** Greg Kroah-Hartman
It may be many years before we fully understand the keys to the Linux kernel's success, but there are a few lessons that stand out even now.
- 46 **A tour of Google's 2016 open source releases** Josh Simmons
We look at 7 of the exciting open source projects Google rolled out in 2016.

- 48 **Top 10 Linux news stories of 2016** Scott Nesbitt
The past year was packed with Linux anniversaries and announcements. See which ones made our top 10 list.
- 51 **2016 Hacktoberfest ignites open source participation** Ben Cotton
Registration was up more than 97% over 2015.
- 55 **Open source diversity efforts gain momentum in 2016** Nithya Ruff
Efforts to increase diversity in open source aren't new, but they are starting to show positive results. We look at the 2016 landscape.

Most Playful Top 7 Linux games of 2016

- ROBIN MUILWIJK
- 52 What were the hot Linux games of the year? We pick a few favorites.

LEARNING

62 **Publisher's picks: Top 2016 open source books**
Rikki Endsley
What were your favorite tech books of 2016? We round up a few hot releases.

66 **8 fun Raspberry Pi projects to try** Anderson Silva
We round up recent Pi projects for making a weather station, media center, security system, and more fun Pi projects to try.

Most Popular Top 10 open source projects of 2016

.....JEN WIKE HUGER

58 In our annual list of the year's top open source projects, we look back at popular projects our writers covered in 2016, plus favorites Opensource.com community moderators picked.

OLD SCHOOL

83 **How Linux got to be Linux: Test driving 1993-2003 distros** Seth Kenlon
Enjoy a trip down Linux memory lane as we take early distros for a spin.

88 **Compute like it's 1989** Seth Kenlon
Let's look back at how people used to compute, back when a "desktop" computer was so called because it took up 80% of your desktop.

92 **LinuxQuestions.org celebrates sweet 16** Jeremy Garcia
The founder of LinuxQuestions.org looks back on the site's humble beginnings and the years in between.

6 7 Reasons to Write for Us / Follow Us

94 Call for Papers / Editorial Calendar

All lead images by Opensource.com or the author under CC BY-SA 4.0 unless otherwise noted.

Most Likely to Succeed

Top open source projects to watch in 2017

.....JASON BAKER

80 Explore some of the fastest-growing new open source projects of 2016 and learn why you might want to dig a little deeper into each in the new year.

CREATING

68 **5 trends in open source documentation**
Shaun McCance
Certain trends in tech documentation stand out. We round up five top trends from 2016.

70 **11 wonderful wearable open source projects**
Ruth Suehle
Browse through a few of our favorite open source wearable projects from 2016, which feature 3D printing, Arduinos, and more.

72 **Top open source creative tools in 2016**
Máirín Duffy
Whether you want to manipulate images, edit audio, or animate stories, there's a free and open source tool to do the trick.

77 **Top open innovations in 3D printing** Tom Callaway
Open source continues to drive rapid innovation in the 3D printing industry.

5 initiatives that pushed the free software envelope in Europe in 2016

BY PAUL BROWN

THE PUBLIC SECTOR tends to lag—some would say *drag*—behind the private sector when it comes to adopting new technologies. This is also true when it comes to adopting free software: Although companies widely see free technologies as a boon, government organizations often are still locked into proprietary software and work with closed standards.

That said, some countries are making progress moving toward open source technologies.



1. Bill makes free software a priority in the Russian public sector

The draft of the bill [1], approved by the Russian Federation's State Duma (lower house) in mid-October, requires the public sector prioritize free software over proprietary alternatives, gives preference to local IT businesses that produce free software for public tenders, and recognizes the need to encourage collaboration with the global network of free software organizations and communities.

The bill is intended to reduce the dependency of the Russian public sector on non-Russian proprietary vendors, boost the local IT industry, and increase collaboration with free software organizations and communities.

2. Amendments to Bulgarian's Electronic Governance Act pave the way for free software

Moving a bit westward, and in a similar vein as the Russian law described above, Bulgaria amended its Electronic

Governance Act [2] to require that all software written for the government be open source and developed as such in a public repository (i.e., the Bulgarian government is setting up its own GitHub).

In his blog post about the news, Bozhidar Bozhanov [3], advisor to the Bulgarian Deputy Prime Minister

and the person who engineered the hacking of the Bulgarian law, says, "It means that whatever custom software the government procures will be visible and accessible to everyone. After all, it's paid by tax-payers money and they should both be able to see it and benefit from it."

This is common sentiment, pushed hard by groups such as the FSFE [4] (the Free Software Foundation [5]'s European sister organization). In fact, European governments adopting free software has become a bit of a trend in recent years. EU directives that require more transparency in procurements often make adopting free licenses for software made by or for the public administration the only viable option.

3. The Netherlands moves toward adopting open standards

The Netherlands is taking steps toward making the use of open standards [6] mandatory for public administrations in the Netherlands. A law [7] proposed by Dutch MP Astrid Oosenbrug [8] was adopted by the lower house of the parliament in October and goes into effect in 2017. Oosenbrug says the minister earlier agreed to make open standards mandatory. “Ironically, the lower house published the adopted law on its website by providing a download link to a document in a proprietary format,” Oosenbrug adds. The upper house, on the other hand, uses the Open Document Format, an ISO standard. Baby steps, I guess.

4. Members of the European Parliament vote for more free software in the public sector

The supranational organizations at the heart of the European Union are also updating their regulations. In January, the European Parliament adopted an initiative [9] that should bolster the adoption of free software within the EU’s public sector.

The initiative requires the European Commission “better promote the security advantages of open source software upgrades to users” and “increase the share of free and open source software and its reuse in and between public administrations as a solution to increase interoperability.”

The initiative, however, was overshadowed by the fact that it also considers licensing standard and essential patents under FRAND (fair, reasonable, and non-discriminatory) licenses “in order to preserve R&D and standardization incentives and foster innovation.” (As the FSFE points out in its analysis of the initiative [10], FRAND licenses are bad for free software [11].)

All of the above is well and good, but if you’re not working in or with the public sector, it all seems a bit remote. What about laws that favor the end user? Well, completely out of the left field, comes this law that they passed in Germany:

5. Germany forbids “compulsory routers”

Until recently, Internet service providers (ISPs) in Germany decided which router users had to use to connect to the Internet. Users had no say in which devices they had to pay for and install in their homes.

This changed on August 1. A new law [12] allows users choose the device that gets installed in their homes. Clients of German ISPs are now allowed by law to use any terminal device they choose. Regardless of whether it is a DSL or cable connection, the ISP will have to supply the information

you need to connect your alternative router to use the Internet and telephone network.

Open progress

The public sector does things at its own, often glacial, pace. Although there have been bold steps toward adopting free software, often the steps are tiny. That said, the trend is toward open: Data generated by public organizations is opening up, open standards are being adopted, and free software is making its way onto publicly owned servers and workstations.

Hopefully the trend will continue—and accelerate—in 2017.

Resources

- [1] <http://www.bloomberg.com/news/articles/2016-10-05/russia-weighs-replacing-ibm-microsoft-with-open-source-software>
- [2] <https://thepolicy.us/bulgaria-got-a-law-requiring-open-source-98bf626cf70a#.pg42r65iq>
- [3] <https://twitter.com/bozhobg?lang=en>
- [4] <https://fsfe.org/index.en.html>
- [5] <http://www.fsf.org/>
- [6] <https://opensource.com/resources/what-are-open-standards>
- [7] <https://joinup.ec.europa.eu/community/osor/news/nl-parliament-makes-open-standards-mandatory>
- [8] <https://twitter.com/astridoosenbrug?lang=en>
- [9] <http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+TA+P8-TA-2016-0009+0+DOC+PDF+V0//EN>
- [10] <https://fsfe.org/news/2016/news-20160128-01.en.html>
- [11] <https://fsfe.org/activities/os/why-frand-is-bad-for-free-software.en.html>
- [12] <https://fsfe.org/news/2016/news-20160725-01.en.html>

Author

Paul Brown has worked as a tech journalist, specializing in Internet trends and free software, for about 20 years. He started writing for the Spanish counter-cultural/hacker magazine @RROBA in 1996, and from there moved on as writer, editor, and later, editor-in-chief for *Linux Magazine Spain*, *Android User Spain*, *Ubuntu User Spain* and *Ubuntu User International*. He has also contributed articles to *Linux Magazine International*, *Raspberry Pi Geek*, and many other publications. He spends his free time as a volunteer teacher, teaching about open hardware to school children and writing fiction and scripts for short films and TV shows that never get made. Find Paul on Twitter: @linux_spain



10 open source tools for your sysadmin toolbox

BY BEN COTTON

SYSADMINS, no matter what platforms they work on, are awash in great open source software tools. In this article, I highlight well-known—and not-so-well-known—tools that released new versions in 2016.

Windows subsystem for Linux

“Microsoft loves Linux” has been a constant refrain from Redmond lately. With the announcement of the Windows Subsystem for Linux [1] (WSL) in the spring, this sentiment has become evident in a way never before seen. More than just an emulation layer, WSL allows Windows users to run a real Ubuntu userspace. This includes the bash shell and utilities like sed, awk, and grep. Linux sysadmins who have to parse log files occasionally on Windows servers will love this feature.

PowerShell for Linux

Of course, some sysadmins primarily work on Windows and have to switch to Linux occasionally. To help those folks, Microsoft dropped another bomb [2] over the summer: PowerShell is now open source (under the MIT license) and ported to Linux. With these two announcements, will we remember 2016 as the year the long-standing battle between Microsoft and open source communities finally came to a complete and total end?



Vim

Just because the Windows/Linux battle has been laid to rest, that doesn't mean the editor wars are over, too. The venerable Vim [3] editor, which celebrated its 25th birthday [4] in November, is still under active development. This year saw the release of version 8, the first major release in a decade. Vim 8 brings features such as support for GTK+ 3 and DirectX, asynchronous I/O for plugins, and jobs.

Git

Versioning is important for your scripts, your text files, and of course your infrastructure-as-code. The Git version control system [5] release version 2.10, which comes with a slew

of handy new features. New color controls allow, for example, git diff output to strike-through removed lines. Improved GPG signing for tags and commits is included, too. Pushes now show progress for remote post-receive operations. And for those forward-thinking users, the internal

date formatting can now handle dates beyond the year 2100.

GitLab

Git is nice on its own, but it's even better with a workflow system. GitLab [6] released version 8.11 this summer, which includes a killer feature: Issue boards. Now issues can be

visually tracked on a Kanban-style system native to GitLab. This is great for planning your infrastructure sprints without having to rely on an external tool. The other major feature in 8.11 is the ability to manage and resolve basic merge errors directly from the GitLab web interface.

SystemRescueCD

Computers are cruel, and they sometimes wind up in a bad state to torment their sysadmins. Many sysadmins carry a CD or USB disk with tools that help recover those machines. SystemRescueCD [7] is an actively developed toolset for those cases. A regular Swiss Army knife, SystemRescueCD is a bootable Linux distribution with tools for testing hardware, partitioning drives, and recovering data. Versions 4.8 and 4.9 were released in 2016, bringing updates to a variety of components, including updated filesystem tools for the ext family and BTRFS.

Clonezilla

Sometimes the best thing to do is to reimage a machine. Clonezilla [8] is the de facto standard for deploying disk images. The latest release adds support for detecting volumes encrypted with Windows bitlocker. A number of point releases over the past year have kept Clonezilla tightly tracked to the upstream Debian distribution and improved EFI support, along with a wide array of bug fixes.

Docker

Docker continued with its active container technology development in 2016. Docker [9] 1.12 added swarm mode: A way to manage a self-healing, self-organizing group. In order to provide this, a health check mechanism was added. This framework allows for service-aware determination of when a container is healthy. Another noteworthy event was the announcement that Docker containers could run natively on Windows [10] as part of a partnership between Docker and Microsoft that provides enterprise support for Docker on Windows [11].

Kubernetes

Speaking of containers, Kubernetes [12] 1.4 added more container management features in 2016. Clusters can now be created with only two commands. A dashboard UI provides 90% feature parity with the command-line tools for easier reporting and quick status awareness. Packag-

ing improvement mean that sysadmins can install Kubernetes with their favorite package managers, such as yum and apt-get.

Nextcloud

Early this summer, a group of ownCloud developers (including a co-founder) forked the project to create Nextcloud [13]. Less than two weeks later, they published their first major release. Nextcloud 10 is the second release since the fork and contains many new features. A new app allows for managing file retention policies. Improvements to the authentication system allow for automatic revocation of users with disabled LDAP accounts, user session revocation, a two-factor authentication plugin system and more.

Did I leave your favorite open source tool for sysadmins off the list?

Resources

- [1] <https://blogs.msdn.microsoft.com/wsl/2016/04/22/windows-subsystem-for-linux-overview/>
- [2] <https://azure.microsoft.com/en-us/blog/powershell-is-open-sourced-and-is-available-on-linux/>
- [3] <http://www.vim.org/>
- [4] <https://opensource.com/life/16/11/happy-birthday-vim-25>
- [5] <https://git-scm.com/>
- [6] <https://about.gitlab.com/>
- [7] <https://www.system-rescue-cd.org/>
- [8] <http://clonezilla.org/>
- [9] <https://www.docker.com/>
- [10] <https://blog.docker.com/2016/09/dockerforws2016/>
- [11] <https://blog.docker.com/2016/09/docker-microsoft-partnership/>
- [12] <http://kubernetes.io/>
- [13] <https://nextcloud.com/>

Author

Ben Cotton is a meteorologist by training and a high-performance computing engineer by trade. Ben works as a technical evangelist at Cycle Computing. He is a Fedora user and contributor, co-founded a local open source meetup group, and is a member of the Open Source Initiative and a supporter of Software Freedom Conservancy. Find him on Twitter (@FunnelFiasco) or at FunnelFiasco.com.



7 notable legal developments in open source in 2016

• BY RICHARD FONTANA

A NUMBER of interesting and notable legal developments in open source took place in 2016. These seven legal news stories stood out:

1. Victory for Google on fair use in Java API case

In 2012 the jury in the first *Oracle v. Google* trial found that Google's inclusion of Java core library APIs in Android infringed Oracle's copyright. The district court overturned [1] the verdict, holding that the APIs as such were not copyrightable (either as individual method declarations or their "structure, sequence and organization" [SSO]). The Court of Appeals for the Federal Circuit, applying 9th Circuit law, reversed [2], holding that the "declaring code and the [SSO] of the 37 Java API packages are entitled to copyright protection." The U.S. Supreme Court declined to review the case, and in 2016 a closely watched second trial was held on Google's defense of fair use. In May 2016 the jury returned a unanimous verdict in favor of Google.

As Jeff Kaufman explains [3], the verdict does not change the appellate ruling concerning API copyrightability, which, however, has limited precedential significance. Fair use involves a highly fact-specific determination, and the verdict has no obvious broader legal significance. Nonetheless the result was a clear victory for Google. Oracle has filed an appeal.

Although *Oracle v. Google* is not a "case about open source" per se, it is notable that both sides are stewards

of relevant open source platforms centered around Java development. Oracle leads the OpenJDK project, in which the APIs at issue in this case, if we regard them as copyrightable, are licensed under GPLv2 along with the Classpath Exception [4]. The Android platform, which does not implement all Java core library APIs, is licensed mostly under the Apache License 2.0. Its Java core library API implementations were generally taken from the Apache Harmony [5] project, which began as a pre-OpenJDK effort to develop an open source Java runtime. Late last year Google confirmed [6] that Android Nougat would use GPL-licensed [7] class library code from OpenJDK in place of the Apache Harmony code.

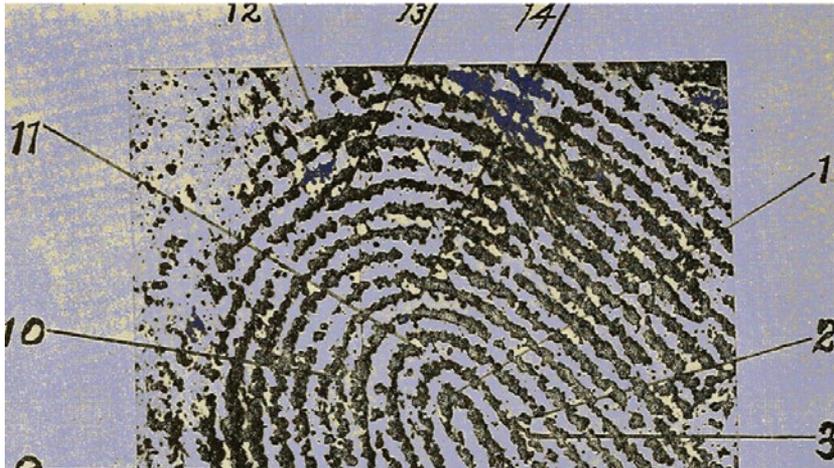


Image by: Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0

2. Censure of Patrick McHardy

Since 2014 there have been rumors of GPL enforcement lawsuits being brought against many companies in Germany by Patrick McHardy, a Linux kernel developer who was formerly the chair of the Netfilter [8] core team. There is

some discussion of the McHardy litigation in a recent Black Duck/DLA Piper slide deck [9].

Until 2016 there has been something of a taboo on open discussion of the McHardy lawsuits. This ended on July 18th, when the Netfilter project announced [10] that it would "suspend" McHardy from the Netfilter core team, the first such action it had ever taken, because "severe allegations have been brought forward against the style of his license

enforcement activities.” Although the core team had no first-hand evidence for the allegations, which were consistent and came from “trusted sources,” they noted that despite many attempts to reach McHardy he did not respond. The announcement was made in the name of the core team members, including emeritus member Harald Welte, who is well known for bringing a series of successful GPL enforcement lawsuits in Germany.

A few weeks earlier, the Netfilter core team published a statement [11] officially endorsing the Principles of Community-Oriented GPL Enforcement [12], which were released by the Software Freedom Conservancy and the FSF in 2015. The core team stated that “license enforcement is a necessary tool to ensure all parties adhere to the same set of fair rules as set forth by the license,” but then, presumably alluding to McHardy, declared that “any enforcement action should always be focused on compliance, never prioritize financial gain, never settle for less than compliance and consider legal action in court only as a last resort.” In the July 18th announcement of McHardy’s suspension, the core team said that McHardy “continues to be welcome in the project as soon as he is able to address the allegations and/or co-sign the [Conservancy/FSF Principles] in terms of any future enforcement activities.”

The next day, Karen Sandler and Bradley Kuhn of the Software Freedom Conservancy published a blog post [13] addressing the subject of McHardy. They revealed that Conservancy had engaged in largely unsuccessful attempted communications with McHardy for two years. Conservancy encouraged McHardy to co-draft the Principles with them and later invited him to endorse the Principles after they were published, but received no response from him. Sandler and Kuhn denounced McHardy for apparently refusing to endorse the Principles and failing to publicly justify his conduct of GPL enforcement.

3. Hellwig lawsuit dismissed

In 2015 Linux kernel developer Christoph Hellwig brought a copyright infringement suit against VMware in a German district court, alleging violation of the GPL in VMware’s ESXi product. Hellwig’s legal expenses were funded by the Software Freedom Conservancy. The Hellwig lawsuit attracted significant attention because it is apparently the first litigated GPL compliance case that centers on the scope of the GPL’s copyleft requirement, sometimes thought of as the “derivative work” issue.

In July 2016, as Scott Peterson has reported [14], the court dismissed the case, concluding that Hellwig had failed to identify in the VMware product the specific lines of code in which he owned copyright. The court discussed the GPL issue, but it did not address the merits. The ruling has no precedential significance for other cases. In a brief statement, Hellwig announced that he would appeal the ruling.

4. U.S. government announces Federal Source Code Policy

In August the U.S. government’s Office of Management and Budget announced the Federal Source Code Policy [15]. The policy is aimed at reducing the problem of duplicative acquisition of substantially similar code by agencies and ensuring that new custom-developed federal source code be made broadly available for reuse across the federal government. Mark Bohannon has written an article [16] on the policy.

The Federal Source Code Policy establishes a three-year pilot program that requires agencies (with some exclusions) to release at least 20% of new custom-developed software as open source each year. The policy recognizes open source as a means of enabling continual improvement resulting from improvements to the software by the broader community. The policy also announced the launch of code.gov [17], a “discoverability portal” for custom-developed code, including code released as open source under the policy.

The Federal Source Code Policy is notable for placing emphasis on adhering to proper standards for open development as well as open source licensing. Agencies releasing open source code are directed to do so in a manner that encourages engagement with existing communities, fosters growth of new communities, and facilitates contribution both by the community to the federal code and by federal employees and contractors to upstream projects. Agencies must also ensure that their open source repositories include enough information to enable reuse and participation by third parties, including details on licensing.

5. Moglen steps down as FSF general counsel

The Free Software Foundation announced [18] in October 2016 that Eben Moglen had “stepped down” as general counsel to the FSF. Moglen, who is president of the Software Freedom Law Center and a law professor at Columbia, has been one of the most influential lawyers in free software. His career in free software has been closely associated in the public mind with the FSF, for which he provided pro bono legal representation for 23 years. I expect both Moglen and the FSF to remain as engaged as ever in matters of free software legal policy, but likely with more instances of public disagreement or conflicting opinions.

6. Debian and Ubuntu ship ZFS

In the mid-2000s Sun Microsystems released its ZFS filesystem as part of OpenSolaris, licensed under the weak copyleft CDDL [19]. Efforts to port ZFS to Linux were inhibited for many years by legal concerns, including concerns about license conflicts between GPLv2 and CDDL. In recent years the “ZFS on Linux” [20] project has encouraged Linux distributions to package its ZFS kernel module.

Although packaging of ZFS in Debian was held up for some time by licensing concerns, in 2015 Debian Project Leader Lucas Nussbaum revealed [21] that Debian had

received legal advice from the Software Freedom Law Center concerning inclusion of ZFS in Debian, which he said “should unblock the situation ... and enable us to ship [ZFS] in Debian soon.” In January 2016, Nussbaum’s successor, Neil McGovern, said [22] that ZFS would be included in Debian as a DKMS package in source code form only, and would be segregated in the “contrib” archive, which contains packages that are not considered to be official Debian.

Ubuntu had included a source-only DKMS ZFS package for some time before Debian began doing so. In a blog post in February, Canonical’s Dustin Kirkland announced [23] that Ubuntu would begin shipping a binary ZFS kernel module. Following a flurry of debate over the GPL/CDDL issue, Kirkland said [24] in another blog post that Canonical had discussed the legal issues with Eben Moglen (president of SFLC) and had concluded that distribution of the binary kernel module would be compliant with both GPLv2 and CDDL. Kirkland stressed that the ZFS module was “self contained” and was not a derivative work of the kernel, and the kernel was not a derivative work of ZFS. Kirkland also argued that “[e]quivalent exceptions have existed for many years, for various other stand-alone, self-contained, non-GPL kernel modules.”

Shortly after Kirkland’s second blog post, the Software Freedom Conservancy and SFLC published conflicting [25] analyses [26] of the legality of Canonical’s distribution. (For those who don’t know: SFLC and Conservancy are independent organizations.) They agreed, however, on two basic points: (1) Debian’s distribution of a source-only module in contrib was license compliant, and (2) loadable kernel modules generally fall within the scope of the GPL copyleft on the kernel.

Conservancy claimed to be speaking on its own behalf as a Linux kernel copyright assignee as well as on behalf of kernel copyright holders participating in its GPL Compliance Project for Linux Developers [27]. In Conservancy’s view, Canonical’s distribution of the binary kernel module violates GPLv2 and thus infringes copyright on the kernel. Conservancy believes that derivative works involving GPL license incompatibilities with other free software licenses should be subjected to the same legal analysis as GPL/proprietary combinations.

According to SFLC, Canonical’s binary ZFS module must be regarded as licensed under GPLv2, since CDDL allows binaries to be under any license and any other interpretation would assume that Canonical was noncompliant with the GPL. Therefore, distribution of the ZFS binary module itself would not violate GPLv2; however, Canonical’s otherwise compliant distribution of corresponding source code for the ZFS kernel module and the Ubuntu kernel would “literally” violate GPLv2, because Canonical would be providing the ZFS filesystem source code under CDDL. There are good reasons for a community of copyright holders of a GPL project not to object to this literal GPLv2 violation,

because the conduct falls within the spirit or the “equity” of the license.

In SFLC’s view, given the tension between the literal and equitable interpretations of GPLv2, “the consensus of the kernel copyright holders’ intention ... determines which mode of interpretation is to be employed.” Here, there was no conclusive or convincing evidence of what type of interpretation the kernel copyright holders intend. SFLC argued that for as long as the kernel copyright holders choose not to object to Canonical’s distribution, it should be assumed that the consensus of the kernel licensors is to support the equitable interpretation. SFLC also pointed out that Canonical’s potential liability exposure was negligible.

Neil McGovern discussed his experience of the ZFS topic as Debian Project Leader in a talk [28] at Debconf. Other noteworthy statements on the ZFS issue were made by Richard Stallman [29] and by Linux kernel developer James Bottomley [30]. Little has been said about the issue in recent months.

7. Apache Software Foundation bans JSON license

For some of us involved in open source legal matters, Douglas Crockford’s JSON license [31] keeps turning up like a bad penny. The JSON license famously modifies the MIT license by adding a sentence before the warranty disclaimer: “The Software shall be used for Good, not Evil.” It is not clear whether Crockford intended the license purely as a joke, or as an oblique political statement, or both. Many who care about having a principled basis for classifying licenses as free, or open source, see the “Good, not Evil” clause as conflicting with basic definitional norms that disallow field of use restrictions and discrimination based on field of endeavor. Some have argued that the clause is not enforceable and thus should not be taken seriously; however, the FSF, which classifies the JSON license as non-free, argues [32] that it cannot be presumed that the restriction is unenforceable. Another objection to the license is that “Good” and “Evil” are undefined and thus the scope of conduct that is allowed and prohibited is highly uncertain.

The reason the JSON license is not a matter of complete obscurity is that Crockford has applied it to software that happens to have been widely adopted, including the tools JSLint [33] and JSMIn [34] and the JSON Java [35] library (“JSON-java”). Over the years Crockford has refused many requests from developers to change the license, although he has boasted [36] of having granted special permission to IBM and “its customers, partners, and minions, to use JSLint for evil.”

For many years the Apache Software Foundation, known for strict rules on licensing under which, for example, the GPL and LGPL are relegated to a forbidden “Category X,” [37] treated JSON-java as though it were in its most favored “Category A” [38] (which contains noncopyleft licenses, such as the Apache License 2.0 itself). Today several ASF proj-

ects have dependencies under the JSON license. In October 2016, in a posting [39] to the ASF's legal-discuss mailing list, Ted Dunning called on the ASF to revisit its decision, noting that the JSON license was "substantially hindering downstream adoption." After discussion, Jim Jagielski, VP of Legal Affairs for the ASF, declared [40] that "the license is NOT CatA and is NOT approved," placing the JSON license in Category X. Jagielski later clarified [41] that no new use of the JSON license by an ASF project would be allowed, but some projects already using code under the license would have a grace period of several months to transition to a replacement. The issue was covered in a November 2016 LWN.net article [42].

Because so many ASF projects have been widely adopted, the JSON license prohibition seems likely to have a significant community impact in encouraging use of open source alternatives to JSON-licensed software.

Resources

- [1] <https://opensource.com/law/12/6/oracle-v-google-and-api-copyrightability>
- [2] <http://www.ca9.uscourts.gov/content/oracle-america-inc-v-google-inc-opinion>
- [3] <https://opensource.com/law/16/6/outcome-google-v-oracle-good-open-source>
- [4] <http://openjdk.java.net/legal/gplv2+ce.html>
- [5] <https://harmony.apache.org/>
- [6] <http://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/>
- [7] <https://android.googlesource.com/platform/libcore/+29c2a3a52980b18ab26f860e9c-c712487881b081%5E%21/#F0>
- [8] <http://netfilter.org/>
- [9] <http://www.slideshare.net/blackducksoftware/litigation-and-compliance-in-the-open-source-ecosystem>
- [10] <https://marc.info/?l=netfilter-devel&m=146887464512702>
- [11] <https://www.netfilter.org/files/statement.pdf>
- [12] <https://sfconservancy.org/copyleft-compliance/principles.html>
- [13] <https://sfconservancy.org/blog/2016/jul/19/patrick-mchardy-gpl-enforcement/>
- [14] <https://opensource.com/law/16/8/gpl-enforcement-action-hellwig-v-vmware>
- [15] <https://sourcecode.cio.gov/>
- [16] <https://opensource.com/government/16/8/us-government-releases-new-policy-free-code>
- [17] <https://www.code.gov/>
- [18] <https://www.fsf.org/news/fsf-announces-change-in-general-counsel>
- [19] <https://opensource.org/licenses/CDDL-1.0>
- [20] <http://zfsonlinux.org/>
- [21] <https://lists.debian.org/debian-devel-announce/2015/04/msg00006.html>
- [22] <http://blog.halon.org.uk/2016/01/on-zfs-in-debian/>
- [23] <http://blog.dustinkirkland.com/2016/02/zfs-is-fs-for-containers-in-ubuntu-1604.html>
- [24] <http://blog.dustinkirkland.com/2016/02/zfs-licensing-and-linux.html>
- [25] <https://sfconservancy.org/blog/2016/feb/25/zfs-and-linux/>
- [26] <https://www.softwarefreedom.org/resources/2016/linux-kernel-cddl.html>
- [27] <https://sfconservancy.org/linux-compliance/>
- [28] <http://bit.ly/2i4o7Q4>
- [29] <https://www.fsf.org/licensing/zfs-and-linux>
- [30] <http://blog.hansenpartnership.com/are-gplv2-and-cddl-in-compatible/>
- [31] <http://www.json.org/license.html>
- [32] <https://www.gnu.org/licenses/license-list.en.html#JSON>
- [33] <https://github.com/douglascrockford/JSLint/blob/master/jslint.js#L15>
- [34] <https://github.com/douglascrockford/JSONMin/blob/master/jsmin.c#L16>
- [35] <https://github.com/stleary/JSON-java/blob/master/LICENSE#L13J>
- [36] <http://dev.hasenj.org/post/3272592502/ibm-and-its-minions>
- [37] <https://www.apache.org/legal/resolved#category-x>
- [38] <https://www.apache.org/legal/resolved#category-a>
- [39] <http://bit.ly/2huuMhi>
- [40] <http://bit.ly/2iFnysw>
- [41] <http://bit.ly/2huw8sr>
- [42] <https://lwn.net/Articles/707510/>

Author

Richard Fontana is Senior Commercial Counsel on the Products and Technologies team at Red Hat. Most of his work focuses on open source-related legal issues.



Troubleshooting tips for the 5 most common Linux issues

BY JEREMY GARCIA

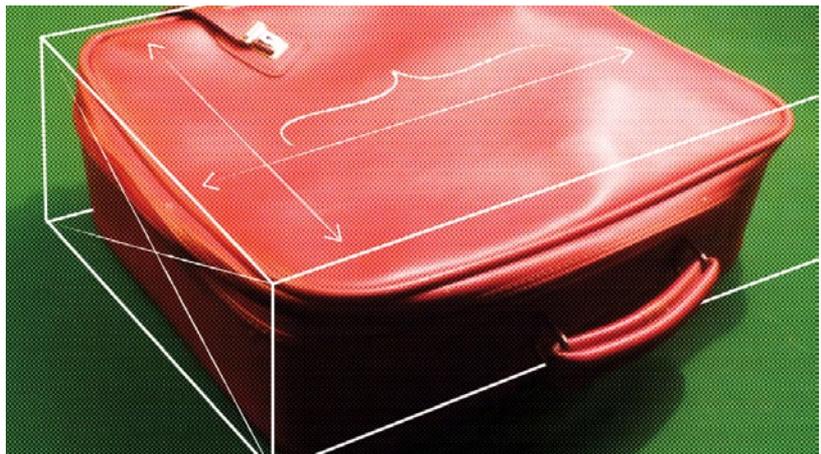
ALTHOUGH LINUX installs and operates as expected for most users, inevitably some users will run into problems. For my final article in The Queue column for the year, I thought it would be interesting to summarize the most common technical Linux issues people ran into in 2016. I posted the question to LinuxQuestions.org and on social media, and I analyzed LQ posting patterns. Here are the results.

1. Wifi drivers (especially Broadcom chips)

Generally speaking, wifi drivers—and Broadcom cards in particular—continue to be one of the most problematic technical issues facing Linux. There were hundreds of posts about this topic on LQ alone in 2016, and myriad more elsewhere. Dozens of Broadcom wireless cards are available, and detailed instructions for getting them to work with each distribution is far too involved for a single article, but the basic troubleshooting steps are the same:

- ascertain exactly which Broadcom card you have by using `lspci` to find out the PCI ID,
- determine whether the distribution you use supports that card,
- and if it does, identify the proper way to get the card working.

For example, if you have a 14e4:4315 PCI ID and are us-



ing Ubuntu, then you know the BCM4312 card is supported by installing the `firmware-b43-installer` package. The other option you have is to research the wifi card before your purchase to ensure it's fully supported by your distribution of choice out of the box.

2. Printer drivers (especially Canon and Lexmark)

Printers also continue to be problematic, with Canon and Lexmark repeatedly cited for being an issue. If you're purchasing a new printer, research compatibility before you buy. But if you are migrating from another operating system, that may not be an option. If you are doing research, the OpenPrinting [1] database and the official support channel for your distribution are the two best places to start. Note that you should ensure all functionality of a device is fully compatible, especially if it's a multifunction

product. One common complaint with Canon printers is that the drivers are often only available on non-English and sometimes obscure sites.

3. Video

Video is a nuanced topic, as simple straightforward video works extremely well out of the box on Linux. Where the issues pop up are video accelerators/acceleration; the latest video cards and newest technologies, such as NVIDIA Optimus and ATI dynamic GPU switching; installation and stability of proprietary drivers; efficient power management; and reliable suspend and resume. If you're not a gamer, do not need high-end graphics for another reason, and are not on a laptop, then you probably don't have to worry about this. If you're looking for a new laptop, be sure to research compatibility before your purchase. If you're a gamer or need the highest-end graphics, you'll need to know exactly what your requirements are and start your research there. Luckily, the situation here is improving and, Wayland teething issues aside, the situation should be quite a bit better in 2017.

4. Audio

Once again, for simple setups, audio has been easy to configure and reliable under Linux for a while. As soon as you get into professional production, echo cancellation, audio routing, unified mixing, and other complex setups, however, it can go south pretty quickly. My suggestion is to use one of the dedicated audio-related distributions if you need high-end real-time audio.

5. Installation

With a category this all-encompassing, it's almost guaranteed to be high volume. That said, I don't know that it's fair

to say Linux has wide-spread installation issues. The vast majority of installs go as expected. The sheer variety of hardware that Linux supports, and nearly infinite combinations of hardware on which Linux installs are attempted, inevitably lead to edge cases here and there. Keep in mind that end users rarely install other operating systems, such as Mac OS and Windows, as they come pre-installed on new devices.

Future looks bright

Other issues that were mentioned frequently include Bluetooth, suspend/resume, HiDPI, and touchscreens. You may see a pattern forming here—most of the issues noted in this article focus on desktop use cases. When you think about it, that makes sense. With Linux desktop adoption being relatively low, the result is that less testing and resources go into finding and fixing related issues. As desktop usage increases, you can anticipate these areas improving.

On that note, I thought it would be nice to end with a mention of one area that used to pop up frequently as a problem area for Linux, and very rarely does these days: fonts. Only a few short years ago, getting high-quality antialiased fonts were the exception. With modern distribution releases, it has become the rule.

What technical Linux issues did you find most common in 2016?

Resources

[1] <http://www.openprinting.org/printers>

Author

Jeremy Garcia is the founder of Linux-Questions.org and an ardent but realistic open source advocate. Follow Jeremy on Twitter: [@linuxquestions](https://twitter.com/linuxquestions)



What's new in OpenStack in 2016: A look at the Newton release

BY RICH BOWEN

OPENSTACK is on a six-month release cycle[1], with each release given a code name starting with consecutive letters of the alphabet. On October 7th, OpenStack Newton was released. Let's look at a few highlights from OpenStack's 2016 Newton release.

In addition to the usual enormous number of incremental improvements, the Newton release focused on ease of deployment and usability improvements, as well as improved container-management tools. It also added the Tacker project [2], for deploying and managing virtual network functions (NFV) on OpenStack.

Documentation

As always, many changes were made to the documentation [3] for this release. The networking guide was restructured, the conversion to RST was completed, training guides were improved, and new chapters were added. In addition to updates to existing translations, entire new manuals were added in Indonesian, Italian, Japanese, Korean, and Simplified Chinese.

Ceilometer

Ceilometer [4] is OpenStack's metering and alarming service. In the Newton release, Ceilometer:

- adds several new meters, including memory bandwidth statistics, and various CPU cycle and instruction count meters;
- includes support for batch recording metering with MongoDB;
- and deprecates ceilometer-dbsync to move to new ceilometer-upgrade.



Congress

Congress [5] is OpenStack's Policy as a Service project. The Newton release adds support for load-balanced policy engines for HA and high query throughput deployments. Congress now supports multi-node deployments in which different components are deployed on separate hosts.

Designate

Designate [6] is OpenStack's DNS as a Service project. Designate has a number of new features in the Newton release, such as support for new DNS servers, including TinyDNS and Knot DNS. It adds designate-worker and designate-producer services for better scaling in future releases. All services now report back to designate-central, which keeps track of what services are running and when they last checked in.

Glance

Glance [7] is OpenStack's image service, for storing VM images and snapshots. The Newton release adds vhdx as a supported disk format [8], and deprecates some older formats and store drivers.

Heat

Heat [9] is OpenStack's orchestration service, and the Newton cycle was very busy for the Heat team as they worked on adding new configuration options.

Conditional functions have been added to allow for more complex orchestration scenarios. Heat can now manage Cinder quotas. The Newton release also adds new integration with DNS service, Glance service, and Monasca service.

Horizon

Horizon [10] is the OpenStack web dashboard. The Newton release contains a number of user interface enhancements, new functionality, and bug fixes to eliminate all WARNING messages in your browser's developer panel.

Horizon used to require Nova and Glance to function, but not any longer. Now Horizon only requires Keystone, which means it can be used for deployments that don't include these services, such as Swift-only deployments.

Horizon now has better implementation of the underlying Bootstrap themes, for easier theming of your OpenStack deployment. Various HTML classes have been renamed to match Bootstrap's naming conventions.

Ironic

Ironic [11] is OpenStack's bare metal deployment service. The Newton release adds a number of new methods and options to allow you to deploy on a wider variety of platforms.

Keystone

Keystone [12] is OpenStack's authentication service, which tends to move slowly and with great deliberation because all other services rely on it. The Newton release adds the ability to cache tokens. Also, a local table may be populated with LDAP users to improve query performance. Also, Keystone can now be upgraded using a rolling upgrade.

Magnum

Magnum [13] is an API service that makes various flavors of containers into first-class resources that can be deployed on OpenStack. The Newton release adds support for several new options, including Flannel's host-gw backend, a new openSUSE driver for running a Kubernetes cluster on openSUSE, and various new options for Apache Mesos.

Manila

Manila [14] is a shared file system service for OpenStack. A number of new drivers and plugins were added to expand the number of backends that can be used for your shared file system service. Additionally, many existing drivers and plugins were enhanced to give access to additional functionality of those backends.

Mistral

Mistral [15] is OpenStack's workflow service. In Newton, Mistral now supports Magnum actions, Tacker actions, and Murano actions. Other new features include being able to call RabbitMQ directly, rather than using Oslo, and the ability to handle https requests.

Nova

Nova [16] is OpenStack's compute service. Because Nova is one of the oldest parts of OpenStack, and the largest and busiest project, listing all the changes is difficult. As with every release, changes in Nova support a wider range of features in the underlying hypervisors. Additionally, many enhancements were made to make migrations and upgrades easier. The release documentation also includes extensive information about how to upgrade your compute nodes from Mitaka to Newton.

Sahara

Sahara [17] provides a way to deploy Apache Hadoop or Apache Spark clusters on top of OpenStack. In Newton, Sahara adds support for Impala, MaR, Sentry, Kafka, CDH 5.7, and updated versions of Mahout, HBase, Drill, and MapR.

Searchlight

Searchlight [18] provides indexing and search across multi-tenant cloud resources. Searchlight first appeared in Mitaka. In Newton, Searchlight adds support for Elasticsearch 2.x, and it adds multi-thread support for indexing. You can now search Neutron security groups and rules. Index sync performance has been improved. Many other enhancements are included in this release.

Senlin

Senlin [19] provides a generic clustering service for an OpenStack cloud. Senlin is another new addition to OpenStack, and the Newton release adds various features that were not yet implemented for Mitaka, including the ability for clusters and nodes to depend on other clusters and nodes, and policy validation and profile validation APIs.

Tacker

Tacker [20] is a generic VNF manager, and NFV orchestrator to deploy and manage virtual network functions on OpenStack. Tacker is new in this release of OpenStack, and begins to build out all of the features needed for this important new area of OpenStack deployments.

Trove

Trove [21] is OpenStack's database as a service solution. Each subsequent release provides additional functionality for the various database engines that it supports. This release also contains numerous bug fixes.

Summary

With a product as large as OpenStack, summarizing what's new in a particular release is challenging. (See the full release notes [22] for more details.) Each deployment of OpenStack might use a different combination of services and projects, and so will care about different updates. Added to that, the release notes for the various projects tend to be extremely technical in nature, and often don't do a great job of calling out the changes that will actually be noticed by either operators or users.

Expect to find a growing collection of videos, on YouTube [23] and elsewhere, demonstrating some of the most interesting user-facing features in Newton. As soon as the Newton release rolled out, developers kept right on working where they left off, because not all of the features planned for Newton actually made it in.

Planning for the next release, Ocata [24], began at OpenStack Summit in Barcelona in late October 2016. The Ocata Release Schedule [25] offers a glimpse at what's ahead.

Be sure to check out Meetup.com [26] for OpenStack meetups near you, where you can talk with other OpenStack operators and compare stories.

Resources

- [1] <http://releases.openstack.org>
- [2] <https://wiki.openstack.org/wiki/Tacker>
- [3] <http://docs.openstack.org/>
- [4] <http://docs.openstack.org/developer/ceilometer/>
- [5] <https://wiki.openstack.org/wiki/Congress>
- [6] <http://docs.openstack.org/developer/designate/>
- [7] <http://docs.openstack.org/developer/glance/>
- [8] <http://docs.openstack.org/developer/glance/formats.html>
- [9] <https://wiki.openstack.org/wiki/Heat>
- [10] <http://docs.openstack.org/developer/horizon/>
- [11] <https://wiki.openstack.org/wiki/Ironic>
- [12] <http://docs.openstack.org/developer/keystone/>
- [13] <https://wiki.openstack.org/wiki/Magnum>
- [14] <https://wiki.openstack.org/wiki/Manila>
- [15] <https://wiki.openstack.org/wiki/Mistral>
- [16] <https://wiki.openstack.org/wiki/Nova>
- [17] <https://wiki.openstack.org/wiki/Sahara>
- [18] <https://wiki.openstack.org/wiki/Searchlight>
- [19] <https://wiki.openstack.org/wiki/Senlin>
- [20] <https://wiki.openstack.org/wiki/Tacker>
- [21] <https://wiki.openstack.org/wiki/Trove>
- [22] <https://releases.openstack.org/newton/>
- [23] https://www.youtube.com/results?search_query=openstack+newton
- [24] <https://releases.openstack.org/ocata/index.html>
- [25] <https://releases.openstack.org/ocata/schedule.html>
- [26] <http://meetup.com/>

Author

Rich works at Red Hat as the Community Liaison for the RDO project, which is a packaging of OpenStack for CentOS/Fedora/RHEL. He's also the Executive Vice President of the Apache Software Foundation, Open Source enthusiast, and Geocacher.



Why the operating system matters even more in 2017

BY GORDON HAFF

OPERATING SYSTEMS don't quite date back to the beginning of computing, but they go back far enough. Mainframe customers wrote the first ones in the late 1950s, with operating systems that we'd more clearly recognize as such today—including OS/360 from IBM and Unix from Bell Labs—following over the next couple of decades.

An operating system performs a wide variety of useful functions in a system, but it's helpful to think of those as falling into three general categories.

First, the operating system sits on top of a physical system and talks to the hardware. This insulates application software from many hardware implementation details.

Among other benefits, this provides more freedom to innovate in hardware because it's the operating system that shoulders most of the burden of supporting new processors and other aspects of the server design—not the application developer. Arguably, hardware innovation will become even more important [1]

as machine learning and other key software trends can no longer depend on CMOS process scaling for reliable year-over-year performance increases. With the increasingly widespread adoption of hybrid cloud architectures, the portability provided by this abstraction layer is only becoming more important.

Second, the operating system—specifically the kernel—performs common tasks that applications require. It man-

ages process scheduling, power management, root access permissions, memory allocation, and all the other low-level housekeeping and operational details needed to keep a system running efficiently and securely.

Finally, the operating system serves as the interface to both its own “userland” programs—think system utilities such as logging, performance profiling, and so forth—and applications that a user has written. The operating system should provide a consistent interface for apps through APIs (application programming interface) based on open standards [2]. Furthermore, commercially supported operating systems also bring with them business and technical relationships with third-party application providers, as well as content channels

to add other trusted content to the platform.

The computing technology landscape has changed considerably over the past couple of years. This has had the effect of shifting how we think about operating systems and what they do, even as they remain as central as ever. Consider changes in how

applications are packaged, the rapid growth of computing infrastructures, and the threat and vulnerability landscape.

Containerization

Applications running in Linux containers [3] are isolated within a single copy of the operating system running on a physical server. This approach stands in contrast to hypervisor-based virtualization in which each application is bound

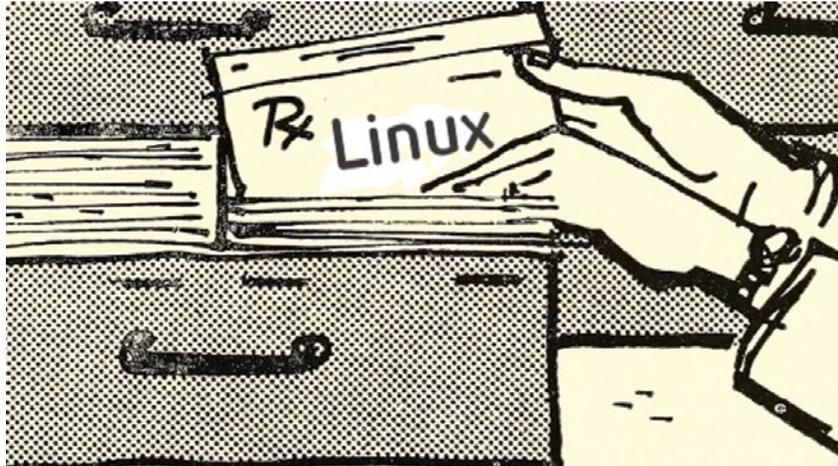


Image by: Internet Archive Book Images. Modified by Opensource.com.

to a complete copy of a guest operating system and communicates with the hardware through the intervening hypervisor. In short, hypervisors virtualize the hardware resources, whereas containers virtualize the operating system resources. As a result, containers consume few system resources, such as memory, and impose essentially no performance overhead on the application.

Containerization leans heavily on familiar operating system concepts. Containers build on the Linux kernel's process model as augmented by additional operating system features, such as namespaces (e.g., process, network, user), cgroups, and permission models to isolate containers while giving the illusion that each is a full system.

Containers have become so interesting recently by the addition of mechanisms to portably compose applications as a set of layers and move them around an environment with low overhead. In this respect, containers are the realization of a general concept that's been around for a while in various guises, but never really went mainstream. (Think application virtualization, for example.) One important change today is the greatly increased role of open source and open standards. For example, the Open Container Initiative [4], a collaborative project under the Linux Foundation, is focused on creating open industry standards around the container format and runtime.

Also significant is that container technology, together with software-defined infrastructure (such as OpenStack), is being built into and engineered together with Linux. The history of computer software clearly shows that integrating technologies into the operating system tends to lead to much wider adoption and a virtuous cycle of ecosystem development around those technologies—think TCP/IP in networking or any of a wide range of security-related features.

Scale

Another significant shift is that we increasingly think in terms of computing resources at the scale point of the datacenter rather than the individual server. This transition has been going on since the early days of the web, of course. However, today we're seeing the reimagining of high-performance computing "grid" technologies both for traditional batch workloads as well as for newer services-oriented styles.

Dovetailing neatly with containers, applications based on loosely coupled "microservices" (running in containers)—with or without persistent storage—are becoming a popular cloud-native approach. This approach, although reminiscent of Service Oriented Architecture (SOA), has demonstrated a more practical and open way to build composite applications. Microservices, through a fine-grained, loosely coupled architecture, allows for an application architecture to reflect the needs of a single well-defined application function. Rapid updates, scalability, and fault tolerance, can all be individually addressed in a composite application, whereas in

traditional monolithic apps it's much more difficult to keep changes to one component from having unintended effects elsewhere.

One important aspect to this shift from the perspective of the operating system is that it increasingly makes more sense to talk about a "computer" as an aggregated set of datacenter resources. Of course, there are still individual servers under the hood and they still must be operated and maintained—albeit in a highly automated and hands-off way. However, container scheduling and management effectively makes up the new and relevant abstraction for where workloads run and how multi-tier applications are composed—rather than the server.

The Cloud Native Computing Foundation [5] (CNCF), also under the Linux Foundation, was created to "drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self-healing multi-tenant nodes." One project under the CNCF is Kubernetes [6], an open source container cluster manager originally designed by Google, but now with a wide range of contributors [7] from Red Hat and elsewhere.

Security

All the security hardening, performance tuning, reliability engineering, and certifications that apply to the virtualized world still apply in the containerized one. And, in fact, the operating system shoulders a greater responsibility for providing security and resource isolation in a containerized and software-defined infrastructure world than in the case in which dedicated hardware or other software may be handling some of those tasks. Linux has been the beneficiary of a comprehensive toolbox of security-enforcing functionality built using the open source model, including SELinux for mandatory access controls, a wide range of userspace and kernel-hardening features, identity management and access control, and encryption.

Today, however, information security must also adapt to a changing landscape. Whether it's providing customers and partners with access to certain systems and data, allowing employees to use their own smartphones and laptops, using applications from Software-as-a-Service (SaaS) vendors, or taking advantage of pay-as-you-go utility pricing models from public cloud providers, there is no longer a single perimeter.

The open development model allows entire industries to agree on standards and encourages their brightest developers to continually test and improve technology. The groundswell of companies and other organizations providing timely security feedback for Linux and other open source software provides clear evidence of how collaborating within and among communities to solve problems is the future of technology. Furthermore, the open source development process means that when vulnerabilities are found, the entire

community of developers and vendors can work together to update code, security advisories, and documentation in a coordinated manner.

These same processes and practices apply across hybrid cloud infrastructures as the role of the operating system evolves and expands to include new capabilities like Linux containers. Furthermore, when components are reused in the form of microservices and other loosely coupled architectures, maintaining trust in the provenance of those components and their dependencies (when making up applications) becomes more important, not less.

Some things change, some don't

Priorities associated with operating system development and operation have certainly shifted. The focus today is far more about automating deployments at scale than it is about customizing, tuning, and optimizing single servers. At the same time, there's an increase in both the pace and pervasiveness of threats to a no longer clearly-defined security perimeter—requiring a systematic understanding of the risks and how to mitigate breaches quickly.

Add it all together and applications become much more adaptable, much more mobile, much more distributed, much more robust, and much more lightweight. Their placement, provisioning, and securing must become more automated. But they still need to run on something. Something solid. Something open. Something that's capable of evolving for new requirements and new types of workloads. And that something is a (Linux) operating system.

Resources

- [1] <http://bitmason.blogspot.com/2016/01/beyond-general-purpose-in-servers.html>
- [2] <https://opensource.com/resources/what-are-open-standards>
- [3] <http://bitmason.blogspot.com/2013/09/what-are-containers-anyway.html>
- [4] <https://www.opencontainers.org/>
- [5] <https://cncf.io/>
- [6] <https://opensource.com/resources/what-is-kubernetes>
- [7] http://stackalytics.com/?project_type=kubernetes-group&metric=commits

Author

Gordon Haff is Red Hat's cloud evangelist, is a frequent and highly acclaimed speaker at customer and industry events, and helps develop strategy across Red Hat's full portfolio of cloud solutions. He is the author of *Computing Next: How the Cloud Opens the Future* in addition to numerous other publications. Prior to Red Hat, Gordon wrote hundreds of research notes, was frequently quoted in publications like *The New York Times* on a wide range of IT topics, and advised clients on product and marketing strategies. Earlier in his career, he was responsible for bringing a wide range of computer systems, from minicomputers to large UNIX servers, to market while at Data General. Gordon has engineering degrees from MIT and Dartmouth and an MBA from Cornell's Johnson School.



25 reasons to love Linux

BY JEN WIKE HUGER

IN 2016, Linux turned 25 years old. To celebrate, we reached out to readers over social media and asked them why they love Linux. On August 25th, [Opensource.com](https://opensource.com) published the list.

25. There is no autopilot. I am the king of my machine.

—Anupam Datta

24. Nowadays stuff just works. No hunting for obscure firmware etc. Plug and play. Done in the open. That's truly, wow. —Jan Wildeboer

23. It is possible to customize Linux in multiple ways: via the kernel when compiling it and in user space. Plethora of free apps. —Eugene J. Markow

22. One thing I like about Linux is the fact that it's absolutely free. That includes price, ability to modify the code to your own specs, etc. No restrictive licensing, etc. —James Takac

21. Don't ask what Linux can do for you, it is already done, but what code you can apply for Linux! —Vladimir Cicovic

20. Working in the terminal is awesome. Makes me look like a badass tech guy in front of the people around me when typing those commands. —Nilesh Sarkar

19. When I ask "How do I send data via the serial port?" on a Linux forum, I get relevant answers and help. Other forums responses be like "Have you tried the parallel port?" —Eric Lovejoy

18. The freedom to edit my GUI however I see fit without worries about end user license agreements. —Jesse Woodside

17. A huge selection of applications, tools, widgets, and other software. —Nathan Leach

16. It gives you the feel of being a real programmer, or a hacker! —Sai Charan

15. There is no limit to what you can do. If you can imagine it, you can make it. —Jeroen Tuijn

14. It is less risky when it comes to virus attack... and of course its an open source! —Kefilwe Mosesanyane

13. The power to customize and create my own specific operating system to be used for any purpose. —Rasyid Sahputra

12. The main thing I like about Linux is there are no wizards. When installing software, it just does it and that includes all the dependencies. —Shaun Henderson



Image by: [Opensource.com](https://opensource.com). CC BY-SA 4.0

11. My machine loves it! The way I have total control over everything. It's a love story that is inexpressible.

—Rhitik Bhatt

10. There's always something new to learn.

—Alexander Golubets

9. Stability, resource friendly, safety. —Alwan Rosyidi

8. The freedom. —Maja Isaksson

7. Terminal <3 —Shahrukh Alizai

6. The commands :) file handling and big data analysis :) —Sum Aira

5. Light-weight, flexible, stabile, safe. —Tomasz Mikołajko

4. There is more than one topic why I love Linux. But I think because it is available for everyone for free, makes the world a little more free. —Tux von Kybermann

3. The ability to freely download, run, change, and distribute the operating system to as many computers as I want. Not everyone can afford to purchase a copy of Windows every 3-4 years. Everyone can afford Linux! —Jonathan Niccolls

2. It's the Swiss Army knife of computing. —Gary Alexander

1. Everything. —Ayoub Arahmat

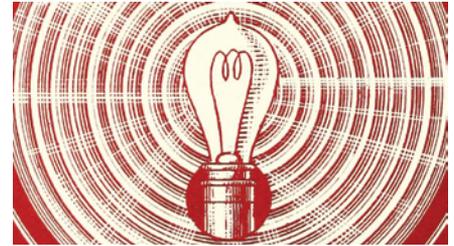
Author

Jen Wike Huger is the Content Manager for [Opensource.com](https://opensource.com). Follow her on Twitter [@jenwike](https://twitter.com/jenwike) and see her extended portfolio at [Jen.io](https://jen.io).



4 hot skills for Linux pros in 2017

BY SHAWN POWERS



ONE OF THE PROBLEMS with becoming a Linux expert is the definition is constantly changing. When I started in the Linux world, to be considered a *Linux professional*, you had to be able to compile your own kernel. Heck, if you wanted to use Linux on a laptop, you had to compile a custom kernel to even be a *user*. These days, compiling your own kernel is usually a waste of time. That's not to say it isn't important, but in the open source world we build on the successes of others, and Linux distributions provide us with kernels that work well. Although not always that drastic, the demands on IT professionals change every year.

Here are four vital skills for the Linux pro in 2017:

1. Security

I'm not talking about security experts or security consultants. With connected devices infiltrating every aspect of our lives, we need to be security conscious in every decision we make. This year, my wife and I purchased a washing machine and a refrigerator, and both of them came equipped with Bluetooth. The idea of hackers breaking into my rinse cycle might seem silly, but any foothold is a potential attack vector.

When we activate any system at work, home, or in our pockets, we should consider the security issues they might represent. And because items like Internet-enabled toasters aren't likely to get timely firmware upgrades, we need to design the rest of our systems around the idea of mundane devices getting compromised. More than ever before, we need to think about attacks coming from inside our firewalls. Don't let your files server get hacked by your blender!

2. DevOps

DevOps is no longer a new concept. For the past two or three years, we've been encouraging folks to learn about DevOps so they can succeed in the workforce. That was

good advice, but it doesn't mean we should rely completely on automation tools to do our jobs. Chef, Puppet, Ansible, Salt Stack, and similar tools are wonderful, but we need to understand what's happening behind the scenes so when something inevitably goes wrong, we know how to fix it.

With DevOps' programmatic approach to computing, we still need people who can maintain, fix, and understand the systems functioning beneath the layer of code. Without Linux experts, cloud computing is a scary place to live, even if that cloud is in your own server room.

3. Development

As a system administrator for 20 years, I never had the time to learn programming. Any development skills I had were basically scripting that helped me do my job faster. Those days are over. While we need to have system administration skills in a DevOps world, we also need system administrators to have programming skills.

If you're a crusty old sysadmin like me, you've probably adopted DevOps and use it on a daily basis. If you truly want to excel, however, you need to learn how to solve problems programmatically and not think of Chef or Puppet code only as configuration files. Every IT professional needs to have at least a grasp of programming concepts, because every aspect of IT is getting abstracted at least somewhat by DevOps code.

4. Soft skills

Often the last thing we think about while preparing for a career are so-called *soft skills*—social and communication skills—and yet they are probably skills most likely to determine your success. Whether you're looking for a new job, or trying to adjust to the changing landscape of your current career, soft skills are vital.

The lines dividing the various areas of IT are blending, and the ability to communicate well makes those blurred lines an advantage instead of a stumbling block. We live in a world in which developers are spinning up servers, and operations teams are writing Ruby code to maintain server farms. These are bold new ideas in IT, and without people able to communicate between disciplines, the workplace becomes hostile quickly. Plus, IT folks have always needed to communicate effectively with people in other areas of business. If anything, that need is greater now than ever.

As you plan for 2017, what skills are you adding to your skill set?

Author

Shawn Powers has been teaching IT for more than a decade. His specialties are Linux, Chef, and integrating multiple platforms for larger networks. He has a passion for teaching others, and his enthusiasm comes through in his courses. He is an associate editor for Linux Journal. Connect with Shawn on Twitter: [@shawnp0wers](https://twitter.com/shawnp0wers)



Top programming trends in 2016

BY RACHEL ROUMELIOTIS

TECHNOLOGY IS constantly moving forward—well, maybe not always forward, but always moving. Even for someone who keeps an eye on the trends and their effect on programmers, discerning exactly where things are headed can be a challenge. My clearest glimpse into open source programming trends always comes in the fall when I work with my fellow chairs, Kelsey Hightower and Scott Hanselman, and our fantastic programming committee to sculpt the coming year's OSCON [1] (O'Reilly Open Source Convention). The proposals that we get and the number focused on specific topics turn out to be good indicators of hot trends in the open source world. What follows is an overview of the top programming trends we saw in 2016.

Languages powering AI

Out of the AI winter of the 1990s, artificial intelligence has reemerged with the computing power that it always needed to influence how we are building software. Machine learning, deep learning, natural language processing, and auto speech recognition blanket the world—from GitHub projects and job posts, to the reason behind the formation of new companies, and clearing space on our cluttered counter tops (Hey, Alexa!). And yes, even events such as OSCON are teeming with the mention of all things AI. Although the availability of computing power has paved the way, open sourcing of all things AI has thrown the industry wide open to innovation and competition. Google's TensorFlow [2], OpenAI [3], and Apache Spark [4] lead the way with pow-

erful frameworks, but there are smaller players, such as Nervana's [5] Neon [6] and Theano [7].

How has the rise in AI affected the software developer's landscape? Well, now is a good time to know Python—its agility and popularity with data engineers and scientists makes it the go-to AI programming language, followed by R, Java, and Scala.

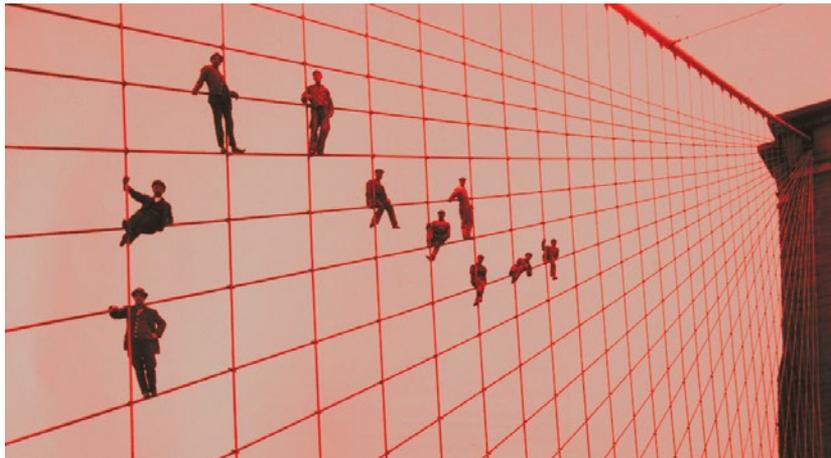


Image by: Museum of Photographic Arts. Modified by Opensource.com

Containers and Go go together like peanut butter and jelly

Go 1.0 was released [8] in March 2012. Docker [9] followed a year later, and Kubernetes [10] a year after that. In short, Go wasn't built exclusively for the future of infrastruc-

ture as we know it, but that seems to be one major hole Go is filling in the programming world. Go is specifically written in a way that Java or C++ could never have been—for a highly networked world, a world in which first-class concurrency is a necessity. If you are in or near the operations side of things, you should at the very least dip a toe into the world of Go because it is gaining steam, will be used for years to come, and will be in the backbone of many applications.

Swift transcends the Apple ecosystem

Swift was open sourced by Apple [11] in 2015, not long after the programming language started. Swift has been a hit with iOS and Mac OS X developers. That the language was easy to grok quickly became apparent, and it earned a reputation for being safer [12] than the languages it aims to replace—Objective-C and C++. How successful Swift is out in the crowded world of JavaScript frameworks [13] and other new

languages remains to be seen, but if it continues gaining popularity with the Apple faithful, there is a chance that Swift will be a viable contender in the great web world and beyond.

Java 8 vs. the functionality of JVM languages

The advent of Java 8's functional capability—namely the introduction of Lambdas—has put JVM languages like Scala and Clojure on notice. Recently, due in large part to the growth of Apache Spark, Scala was having a bit of a growth spurt. Now both Scala and Clojure are seeming to be set aside, at least for the moment, as long-time and new developers alike take a hard look at what Java 8 brings to the table. Java is now able to address concurrency and big data concerns that other programming languages specifically built to address these requirements have been doing for years. In 2017, OSCON is nearly devoid of both Scala and Clojure, not by design, but by seemingly little interest from potential speakers who submitted proposals.

Up and coming languages

And as usual, there are always more up and coming languages on the horizon intended to do something better than those that came before them, that can answer needs that weren't around when previous languages were born, or that simply start out as a crazy idea and end up changing how we think about programming. This year five languages are on the verge of making it into the big time: Rust, Elixir, Elm, Kotlin, and Perl 6.

What do the hot five bring to the industry?

- Rust [14]: Systems programming at speed and more than a modicum of safety.
- Elixir [15]: Functional, dynamic, and fault-tolerant for those larger and larger-scale apps.
- Elm [16]: More functional fun that plays with JavaScript, leaning increasingly toward being a pleasure to use.
- Kotlin [17]: This one is for the Java and JVM folks—statically typed, safe, and did I mention Java compatible?
- Perl 6 [18]: It lives! Perl 6 happens to be a new language that is expressive and feature-rich for the win.

Time will tell whether they deliver on their promises. Try them out, contribute to them, be a part of the future!

Resources

- [1] <http://conferences.oreilly.com/oscon/oscon-tx>
- [2] <https://www.tensorflow.org/>
- [3] <https://openai.com/blog/>
- [4] <http://spark.apache.org/>
- [5] <https://www.nervanasys.com/>
- [6] <https://github.com/NervanaSystems/neon>
- [7] https://github.com/benanne/nervana_theano
- [8] <https://blog.golang.org/go-version-1-is-released>
- [9] <https://opensource.com/resources/what-docker>
- [10] <https://opensource.com/resources/what-is-kubernetes>
- [11] <https://opensource.com/life/15/12/most-likely-succeed-2016>
- [12] <http://bit.ly/1dYYSYI>
- [13] <https://opensource.com/article/16/11/15-javascript-frameworks-libraries>
- [14] <https://www.rust-lang.org/en-US/>
- [15] <http://elixir-lang.org/>
- [16] <http://elm-lang.org/>
- [17] <https://kotlinlang.org/>
- [18] <https://perl6.org/>

Author

Rachel Roumeliotis, a Strategic Content Director at O'Reilly Media, Inc., leads an editorial team that covers a wide variety of programming topics ranging from full-stack web development, to open source in the enterprise, to emerging programming languages. She is a Programming Chair of OSCON, O'Reilly's Software Architecture Conference, and Fluent. She has been working in technical publishing for over 10 years, acquiring content in many areas including mobile programming, UX, computer security, and AI.



50 ways to avoid getting hacked in 2017

BY DANIEL J WALSH

WHEN I WAS YOUNG, Paul Simon released his hit song, “50 Ways to Leave Your Lover” [1]. Inspired by this song, I’ve collected 50 ways sysadmins and laypeople can avoid getting hacked:

“You just slip out the back, Jack”

1. Backup your data. If you get hit with ransomware, you don’t have to pay if you have backups.
2. Use a syncstop [2] when you have to charge your phone in a public place, or bring your own battery backup.
3. Take advantage of the auditing subsystems. There are lots of cool tools to help monitor your system. If you do have a break in, the audit system might well be able to tell you what happened and what the attacker did.
4. Speaking of logs, offloading the logs to a centralized server is always a good idea because if a hacker breaks into your system, the first thing he is going to attack is the logging system to cover his tracks. Having a good intrusion system watching the logs also helps.

“Make a new plan, Stan”

5. Run SELinux in enforcing mode (see stopdisablinglinux.com [3]). Didn’t think it would take me this long to get to that one? SELinux prevents escalations of zero day vulnerabilities. When Shell Shock [4] came out, SELinux was the only defense.
6. Run applications in the SELinux Sandbox [5] whenever possible—it was a container before containers were cool. Also follow the development of Flatpack [6], which soon should be developing sandboxing capabilities.
7. Don’t install or use Flash. Firefox no longer supports it, and hopefully most web servers are moving away from it.

8. Use confined SELinux users [7] to control what users do in your systems. If you are running a shared login system, set up users as **guest_t**.

“You don’t need to be coy, Roy”

9. Take advantage of systemd tools [8] to help secure your services. Most system attacks are going to come through a service listening on the network. Systemd provides great ways to lock down the service. For example, use **PrivateTmp=yes** [9]. PrivateTmp takes advantage of the mount namespace to set up a private **tmpfs** mount for the server’s **/tmp**. This prevents a hacked service from getting access to content in the host’s **/tmp** and potentially attacking the rest of the system based on services listening on **/tmp**.
10. **InaccessibleDirectories=/home** is a systemd unit flag that uses the mount namespace to eliminate the **/home** (or any other directory) from the services view, which makes it more difficult for a hacked service ability to attack the content.
11. **ReadOnlyDirectories=/var** is another systemd unit flag that uses the mount namespace to turn the directories



Image by: Opensource.com, CC BY-SA 4.0

contents into read-only mode. You probably should always run with `/usr` in **ReadOnlyMode**. This would prevent a hacked application from rewriting the binary, so the next time it started the service, you would already be hacked.

12. Drop capabilities from a service (**CapabilityBoundingSet=CAP_CHOWN CAP_KILL**). In the kernel, privileged processes are broken down into a series of distinct capabilities. Most services do not need many (if any), and `systemd` provides a simple switch to drop them from a service.
13. If your service is not going to use the network, then you can turn it off for the service using **PrivateNetwork=yes**. Just turning this on in a service unit file takes advantage of the network namespace and turns off all networks available to the service. Oftentimes a hacker does not actually want to break into your machine—he just wants to use it as an attack server to attack other machines. If the service can't see the network, it cannot attack it.
14. Control the devices available to your service. `Systemd` provides the **DeviceAllow** directive, which controls the devices available to the service. **DeviceAllow=/dev/null rw** will limit access to `/dev/null` and only this device node, disallowing access to any other device nodes. The feature is implemented on top of the device's cgroup controller.
15. Coming soon to a `systemd` system near you is a new feature, **ProtectSystem Strict** [10], which can turn on all of these namespaces to fully lock down the environment in which a service runs.

“Just get yourself free”

16. Don't use a cell phone without SELinux (SEAndroid [11]) in enforcing mode. Luckily, I heard that more than 90% of all Android phones now run with SEAndroid on in enforcing mode. That makes me happy. Now if we could only get those Apple guys to use SELinux.
17. Only install software from trusted sources. Don't install dodgy things you find on the Internet. This goes for your cell phone, computer system, virtual machines, containers, and so on.
18. I don't do online banking on my phone—only on my Linux computer. If a hacker steals my credit card, I lose 50 bucks; if he gets into my bank account, I lose a lot more. I guess I am old. (Get off my lawn.)
19. One cool thing I did do with my phone is set up my credit card companies to send me a text every time my credit card has been charged. That way if the number gets stolen, I will know a lot quicker.
20. When you need to communicate securely, use the Signal secure messaging app [12].

“Hop on the bus gus”

21. Run Linux on your systems. When I first hooked my father up with a computer system, I barely got home

before his system was infested with viruses. I returned and installed Linux on his system, and he has been running it ever since. I believe Linux generally is a more secure system because of the way it was designed, but I also believe the desktop is less likely to be hacked because of the smaller user base. Some would argue that Windows has improved greatly over the years, but for me, I am still sticking with what I know.

22. Only run distributions with a Security Response Team [13] watching over the security of the distribution. Enterprise Software is important.
23. Run an enterprise-level kernel. In containers, the single point of failure is the kernel. If you want to keep it secure, use an enterprise-level kernel, which means it has the latest security fixes, but is not bleeding edge. Remember the latest kernel comes with the latest security fixes, but it also comes with a ton of new code that could have vulnerabilities.

“You don't need to discuss much”

24. Most hacks are social engineering—for example, email links, web browser attacks, and phone calls. The best option here is to be educated and skeptical. No one from Nigeria is giving you money. The IRS is not calling your house demanding money. If you get a link to a web site in email from your bank, don't use the link. Type the address directly on the web browser.
25. Always keep your systems fully up to date with the latest security fixes. The number of systems that are outdated and have known security vulnerabilities is scary. Script kiddies rely on you **not** to update your system.
26. Always use HTTPS when connecting to services on the network. Chrome and Firefox now have modes to enforce this. If a web site does not support secure communications by 2016, it is probably not worth your visit.
27. Use `seccomp` [14] in your containers. This limits the attack surface on the kernel, which is the single point of failure. Limit what the processes can discuss.

“Just drop off the key, Lee”

28. Use a YubiKey [15] for storing private keys.
29. Encrypt your data on your systems. At least for laptops, keep your **homedir** and your other data directories encrypted. I was riding the subway in London a few years ago, and had my Laptop “nicked”—the door of the train car closed, and I noticed by laptop was gone and the train was pulling out of the station. Luckily, the disks were encrypted.
30. Use Let's Encrypt [16] for all your web sites. There's no reason not to run HTTPS anymore.
31. Never use the same password on different web servers. This one is difficult not to fall into the trap. Tools like Let's Encrypt help a lot. It's even better if you use ssh keys to log into systems.

32. Use two-factor authentication (2FA). Passwords have become just about useless. Using YubiKeys and the like make two-factor easy. We all have cell phones. Having a secret in your head and one generated on the phone is always better than a password.
 33. Nothing aggravates me more than websites always asking me to set up an account—can't we do better? Always use a password-generating tool for your website passwords. I am old school: I use Password Safe [17] and cut and paste into the web browser. I have heard that other people have good luck with LastPass [18] and other tools that integrate your phone and web service.
 34. Set up a service like FreeIPA [19] to use for identity services. Using tools such as Kerberos [20] for authentication and authorization makes keeping track of employees and their access to systems much easier (and it has cool crypto services). Using Active Directory is ok, but I am a little prejudiced.
 35. When you must use a password that you need to type in often, use an easily remembered sentence rather than a word. My preferred way to remember passwords is to use a phrase several words long that is easy to type.
- “And get yourself free”**
36. Use USBGuard [21] to protect your system from rogue USB devices.
 37. The past few years, I have been working on containers, so now let's dive into security on containers. First run them on a system with SELinux turned on in enforcing mode. If your system does not support SELinux, switch the distribution to one that does. SELinux is the best tool for protecting against container break out using the file system.
 38. Run your service inside of a container whenever possible. I believe this is the future—applications using OCI Image Format [22] and Linux container technology. Launch these containers with Docker, runC [23], OCID, RKT, Systemd-nspawn, and so on. Although I have often said “containers do not contain,” they do contain better than not running them inside of a container.
 39. Run your container in a VM. Virtual machines provide better isolation than containers. Running like containers on virtual machines provides you scalability and isolation from each other.
 40. Run containerized apps with different security needs on different virtual machines. Run your web service containers on virtual machines in the DMZ, but run the database containers on virtual machines outside of the DMZ.
 41. Also remember to run your virtual machines requiring the most security on different physical machines, on different virtual machines inside of containers (a.k.a., defense in depth).
 42. Run your containers in read-only mode [24]. Containers in development need to be able to write to `/usr`, but a container in production should only be able to write to `tmpfs` and volumes mounted into the container.
 43. Drop capabilities from your containers [25]. We run our processes in and outside of containers with many more “capabilities” than they need. You can make your processes more secure by dropping capabilities.
 44. Don't run your processes in containers as root [26]. Most services never need root privileges, or they need it to bind to a port < 1024 and then switch to a non-root user. I would advise always running apps as non-root.
 45. Keep your containers updated with the latest CVEs fixes. Using a system like OpenShift for building and maintaining your container images is a good idea, because it automatically rebuilds container images when a new security fix appears.
 46. An associate of mine says, “Docker is all about running random code from the Internet as root on your host.” Get your software from a trusted source. Don't grab the first Apache application that you find at docker.io. The operating system matters [27].
 47. Run your containers in production on a limited containerized optimized host, such as an Atomic Host [28], which comes with all of the security turned on, optimized for running containers, with a limited attack surface and atomic updates. What is not to like there?
 48. Use tools like OpenScap [29] to scan your systems for vulnerabilities. Sadly, new vulnerabilities are always popping up, so you must keep your scanners up to date. (Take a look at atomic scan [30] for scanning your containers, as well.)
 49. OpenScap also has features to scan for security configuration [31], such as STIGs (Security Technical Implementation Guides).
 50. Set up a special guest network for all those Christmas IoT devices your kids receive. I love my Amazon Echo and automated lights and power switches (“Alexa, turn on the Christmas Lights”), but each one of these is a Linux operating system that has questionable security.
- “There must be 50 more ways not to get hacked”**
What would you add to the list?
Josh Bressers contributed to this article.

Resources

- [1] <https://www.youtube.com/watch?v=0H5chfbcWtY>
- [2] <http://syncstop.com/>
- [3] <http://stopdisablinglinux.com/>
- [4] <http://danwalsh.livejournal.com/71122.html>
- [5] <http://danwalsh.livejournal.com/31146.html>
- [6] <http://flatpak.org/>
- [7] <http://danwalsh.livejournal.com/37404.html>
- [8] <http://0pointer.de/blog/projects/security.html>
- [9] <http://danwalsh.livejournal.com/51459.html>

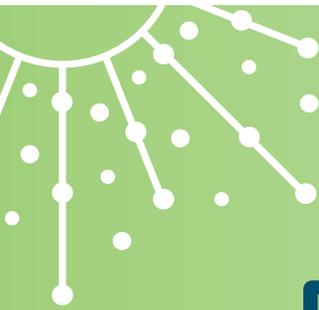
- [10] https://www.phoronix.com/scan.php?page=news_item&px=systemd-New-Protect-Tunables
- [11] <https://source.android.com/security/selinux/>
- [12] <https://whispersystems.org/>
- [13] <https://access.redhat.com/blogs/766093/posts/2695561>
- [14] <https://lwn.net/Articles/656307/>
- [15] <https://www.yubico.com/>
- [16] <https://letsencrypt.org/>
- [17] <https://pwsafe.org/>
- [18] <https://www.lastpass.com/>
- [19] https://www.freeipa.org/page/Main_Page
- [20] <https://web.mit.edu/kerberos/>
- [21] <https://github.com/dkopecek/usbguard>
- [22] <https://www.opencontainers.org/>
- [23] <https://runc.io/>
- [24] <http://www.projectatomic.io/blog/2015/12/making-docker-images-write-only-in-production/>
- [25] <http://rhelblog.redhat.com/2016/10/17/secure-your-containers-with-this-one-weird-trick/>
- [26] <https://www.projectatomic.io/blog/2016/01/how-to-run-a-more-secure-non-root-user-container/>
- [27] <https://www.opensource.com/16/12/yearbook-why-operating-system-matters>

- [28] <https://access.redhat.com/articles/rhel-atomic-getting-started>
- [29] <https://www.open-scap.org/>
- [30] <https://developers.redhat.com/blog/2016/05/02/introducing-atomic-scan-container-vulnerability-detection/>
- [31] <https://www.open-scap.org/security-policies/scap-security-guide/>

Author

Daniel Walsh has worked in the computer security field for almost 30 years. Dan joined Red Hat in August 2001. Dan leads the RHEL Docker enablement team since August 2013, but has been working on container technology for several years. He has led the SELinux project, concentrating on the application space and policy development. Dan helped developed sVirt, Secure Virtualization. He also created the SELinux Sandbox, the Xguest user, and the Secure Kiosk. Dan has a BA in Mathematics from the College of the Holy Cross and an MS in Computer Science from Worcester Polytechnic Institute.





Best Couple

Display manager and window manager

.....BY DAVID BOTH

MY SELECTION FOR BEST Couple of 2015

[1] was `ssh` and `tar`, a pair of Linux commands that work together nicely to accomplish great things. This year I have made a different type of selection for Best Couple of 2016. My choices for Best Couple this year are actually a pair of program types—not specific commands or programs.

So let's welcome our Best Couple of 2016: Put your hands together for the display manager (**dm**) and the window manager (**wm**).

These two programs, regardless of which ones you use on your Linux GUI desktop system, always work closely together to make your GUI experience smooth and seamless before you even get to your desktop.

Display manager

The display manager is a bit of code that provides the GUI login screen for your Linux desktop. After you log in to a GUI desktop, the display manager turns control over to the window manager. When you log out of the desktop, the display manager is given control again to display the login screen and wait for another login.

There are several display managers—some are provided with their respective desktops. Note that some display man-

agers are not directly associated with a specific desktop. Any of the display managers can be used for your login screen regardless of which desktop you are using. And not

all desktops have their own display managers. Such is the flexibility of Linux and well-written, modular code.

The typical desktops and display managers are shown in the Table 1.

The display manager for the first desktop (i.e., GNOME, KDE, etc.) installed is the default one. For Fedora, this is usually GDM [2], which

is the display manager for GNOME. If GNOME is not installed,



Image by: Internet Archive. Modified by Opensource.com.

Table 1: A short list of display managers.

Desktop	Display manager	Comments
GNOME	GDM	GNOME Display Manager
KDE	KDM	KDE Display Manager (up through Fedora 20)
	LightDM	Lightweight Display Manager
LXDE	LXDM	LXDE Display Manager
KDE	SDDM	Simple Desktop Display Manager (Fedora 21 and above)
	XDM	Default X Window System Display Manager



then the display manager for the installed desktop is the default. If the desktop selected during installation does not have a default display manager, then GDM is installed and used. If you use KDE as your desktop, the new SDDM [3] (Simple Desktop Display Manager) will be the display manager.

Regardless of which display manager is configured as the default at installation time, later installation of additional desktops does not automatically change the display manager used. If you want to change the display manager, you must do it yourself from the command line. Any display manager can be used, regardless of which window manager and desktop is used.

Window manager

The function of a window manager is to manage the creation, movement, and destruction of windows on a GUI desktop.

Table 2: A short list of window managers.

Desktop	Window manager	Comments
Unity	Compiz	
	Fluxbox	
	FVWM	
	IceWM	
KDE	Kwin	Starting with KDE Plasma 4 in 2008
GNOME	Metacity	Default for GNOME 2
GNOME	Mutter	Default starting with GNOME 3
	twm	A very old and simple window manager. Some distros like Fedora use it as a fallback in case no other window manager or desktop is available.
Xfce	Xfwm	

The window manager works with the X Window System [4] or the newer Wayland [5] to perform these tasks. The X Window System provides all of the graphical primitives and functions to generate the graphics for a Linux or Unix graphical user interface.

The window manager also controls the appearance of the windows it generates. This includes the functional decorative aspects of the windows, such as the look of buttons, sliders, window frames, pop-up menus, and more.

As with almost every other component of Linux, there are many different window managers from which to choose. The following list represents only a sample of the available window managers.

Note that most window managers are not directly associated with any specific desktop. In fact, some window managers can be used without any type of desktop software, such as KDE or GNOME, to provide a minimalist GUI experience for users.

How do I deal with all these choices?

In most modern distributions, the choices are made for you at installation time and are based on your selection of desktops and the preferences of the packagers of your distribution. The desktop itself can be easily changed in some distributions and the display manager can also be changed in many cases.

Prior to Fedora 18, changing the display manager was done by changing the line `PREFERRED=` in the `/etc/sysconfig/desktop` file. That file was sourced by the `/etc/X11/prefdm` file. If the file did not exist, you could create it, adding the `PREFERRED=` line (in caps) with the name and path of the preferred desktop manager. You could also set it directly in the `prefdm` file, but that change could be wiped out by an upgrade or reinstallation.

Now that `systemd` [6] has become the standard startup system in many distributions, you can set the preferred display manager in `/etc/systemd/system`, which is where the basic system startup configuration is located. There is a symbolic link (symlink) named `display-manager.service` that points to one of the display manager service units in `/usr/lib/systemd/system`. Each installed display manager has a service unit in the `/usr/lib/systemd/system` directory.

To change the active display manager, remove the existing `display-manager.service` link and replace it with the one you want to use. For example, to configure to use the KDM display manager, use the following commands:

```
cd /etc/systemd/system
rm display-manager.service
ln -s /usr/lib/systemd/system/kdm.service display-manager.service
```

The only information I could find initially about changing the window manager is in the Fedora 13 Deployment Guide [7], which is obviously fairly old and may no longer be valid. I also found information on the `wmctrl` command, which, as its name implies, provides some control over the window manager but no capability to change the window manager.

I did find that some distros and desktops have various means of changing the window manager. For example, GNOME users can use `gconf-editor` and Puppy Linux uses the `wmswitcher` command.

Conclusion

As with many other components of GNU/Linux, many different display and window managers are available. When you install most modern distributions with any kind of desktop, the installation program chooses which ones to install and activate. For most users, there should never be any need to change these choices. For others who have different needs or who are simply more adventurous, there are many options and combinations from

which to choose. With a little research, you can make interesting changes.

Resources

- [1] <https://opensource.com/business/15/12/best-couple-2015-tar-and-ssh>
- [2] <https://wiki.gnome.org/Projects/GDM>
- [3] <https://github.com/sddm>
- [4] https://en.wikipedia.org/wiki/X_Window_System
- [5] [https://en.wikipedia.org/wiki/Wayland_\(display_server_protocol\)](https://en.wikipedia.org/wiki/Wayland_(display_server_protocol))
- [6] <https://www.freedesktop.org/wiki/Software/systemd/>
- [7] https://docs.fedoraproject.org/en-US/Fedora/13/html/Deployment_Guide/s2-x-clients-winmanagers.html

Additional resources

- X Window Manager: https://en.wikipedia.org/wiki/X_window_manager
- Comparison of X window managers: https://en.wikipedia.org/wiki/Comparison_of_X_window_managers
- X Display Manager: [https://en.wikipedia.org/wiki/X_display_manager_\(program_type\)](https://en.wikipedia.org/wiki/X_display_manager_(program_type))
- Simple Desktop Display Manager: https://en.wikipedia.org/wiki/Simple_Desktop_Display_Manager
- X Window System: https://en.wikipedia.org/wiki/X_Window_System
- Wayland: [https://en.wikipedia.org/wiki/Wayland_\(display_server_protocol\)](https://en.wikipedia.org/wiki/Wayland_(display_server_protocol))
- X Window System Protocols and Architecture: https://en.wikipedia.org/wiki/X_Window_System_protocols_and_architecture

Author

David Both is a Linux and open source advocate who resides in Raleigh, North Carolina. He has been in the IT industry for more than 40 years and taught OS/2 for IBM, where he worked for more than 20 years. While at IBM, he wrote the first training course for the original IBM PC in 1981. He has taught RHCE classes for Red Hat and has worked at MCI Worldcom, Cisco, and the State of North Carolina. He has been working with Linux and open source software for almost 20 years. David has written articles for *OS/2 Magazine*, *Linux Magazine*, *Linux Journal*, and [OpenSource.com](https://opensource.com). Follow David on Twitter: [@LinuxGeek46](https://twitter.com/LinuxGeek46)



10 steps to innersource in your organization in 2017

BY JONO BACON

IN RECENT YEARS, an increasing number of organizations, often non-technology companies, have kept a keen eye on open source. Although they may be unable to use open source to the fullest extent in their products and services, they are interested in bringing the principles of open source within the walls of their organization. This “innersource” concept can provide a number of organizational benefits.

As a consultant who helps build both internal and external communities in companies, I find the major challenge facing organizations is how to put an innersource program in place, deploy resources effectively, and build growth in the program.

To help you get started, I present a high-level model that shows how you can build a consistent, predictable, and sustainable innersource program. Take this model, adjust to taste, and build a thriving community in your organization.

Understand

Fundamentally, innersource is a *cultural* change for a company. Although many people think of this as a tradi-

tionally software engineering workflow challenge, you need to focus instead on building an asynchronous, permissive, meritocratic, and collaborative environment. Of course, this encompasses development workflow, but is not limited to it.

The challenge with cultural change is that culture is an amorphous mass of ideas, opinions, habits, fears, dreams, values, and more. Before you can integrate innersource into

a company, you must understand the drivers in the existing culture, and then ensure you consider these as you roll out the innersource program.

1. Understand process and collaboration

At the core of how people work together are collaborative infrastructure and processes. These include code hosting, peer review, continuous integration, automated testing, documentation creation, knowledgebases, incentive programs, and more. You should understand all of these details, how they fit together, and where the shortfalls lay in both the wider execution of these pieces and the experiences of staff using them.

I recommend breaking the organization up by teams and then put together a map of:



Image by: Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0

- What each team consumes
- What each team produces
- How each team works
- How they interface with other teams
- The benefits and disadvantages of current systems

2. Understand the people and drivers

Outside of the collaborative nuts and bolts of how things get done, understanding the people involved is equally critical. A company brings together a multitude of people, personalities, and perspectives. You need to understand them, their goals, their fears, and their agendas. Cultural change must be mindful of the realities of the environment in which it operates. You can't build an innersource program by dictating

it to people—you need to build something that people want to use and that encourages the behaviors you want to see.

I recommend building your own organizational chart that shows:

- the breakdown of influence (key stakeholders and decision makers),
- where people deliver work (teams and key employees), and
- individual agendas and goals with each person (stop energy, people gunning for certain outcomes, intrinsic and extrinsic reward motivations, etc.).

Plan

With a firm understanding of the current environment, you then can build a roadmap of your route to a smooth, efficient, inclusive, and enjoyable innersource environment.

3. Create a strategic plan

Your first step is to build an overall strategy, a complex undertaking as you can imagine. Integrating open source principles in a company involves a huge array of different considerations, such as developer workflow, infrastructure, communication, policy (such as openness and transparency), incentivization models, segmented engagement, wider messaging, governance, and more.

Not only do you have much to do, but as your work priorities will vary (some projects are more urgently required than others), you will have limited resources, and some colleagues in the company will have limited buy-in or even seek to block the project.

To get people onboard and involved, you need to:

- build a strategy;
- define priorities;
- get a sense of resourcing; and
- factor in messaging, engagement, and roll-out for each of these pieces.

I recommend building out an overall wider strategic plan that maps to the next one or two years, covering the key areas of focus. Next, break that plan into shorter execution cycles in which you pull out key goals from the broader plan and map them to practical deliverables with metrics. This will form your backlog.

4. Build a backlog

Fundamentally, strategy is a map that tells you where you want to go. Strategy needs to be converted into practical projects and deliverables that you can apply resources to, such as development time, funds, and so on. The challenge is that each strategic objective will invariably involve a multitude of these sub-projects and goals. The best way to manage this situation is with a backlog.

Put simply, a backlog is a big, shared to-do list. When you have built out your strategy, you map all the individual projects to the backlog. This provides a place where you can discuss, refine, and improve individual deliverables. When some of these deliverables are ready for implementation, they can be pulled off the backlog into an active work plan and have resources assigned. This means that you can evolve the implementation of your strategy in the backlog, even when not actively applying resources.

5. Develop a maturity model

One of the challenges I regularly see when I work with clients who want to build innersource into their organizations is that they don't know what success looks like. Now, that sentence right there sounds like business book nonsense, but this issue is real. For example, if you want to improve developer efficiency when it comes to code review, how do you determine that this goal was achieved? How do you measure the work and determine which measurements mean you've nailed it? For many of these issues, you are building qualitative cultural change. How do we measure that?

This challenge is exacerbated by your different audiences. Although those involved in the implementation of this work will want to get to the nitty gritty of what success looks like, senior management and stakeholders won't want the details, but instead only want information on important trends. A useful approach to this is a "maturity model."

A maturity model breaks the different evolutionary phases of a solution into a set of expectations of what success looks like. I tend to think of these different chronological stages, in order:

- Unaware: The company is unaware of the solution.
- Explored: The solution is being explored.
- Defined: The solution is defined and executed.
- Adopted: The solution is adopted in the company.
- Optimized: The solution is being optimized and improved.

For each of these you would say what to expect. For example, if you build code review into the company, the entry for "Explored" could be "A small proportion of teams are actively experimenting with code review in non-critical codebases."

Execute

With a strategy, backlog, and maturity model in place, you know what you need to do. Now you need to make these projects happen.

6. Deliver priority projects

With your backlog in place, your first step is to decide which projects to name as priorities in your work plan. Deciding this is dependent on which work must be performed most urgently and what resources are currently available.

Although the urgency of the work is important, resources are the really defining criteria here—we can't build things with nothing.

I recommend you do this on a cadence (every two weeks, monthly, quarterly, etc.). Bring together key leaders for the program, review the backlog, define resourcing, and then finalize the work plan.

7. Communicate to different groups

With your work plan in place you can start delivery. You should be defining milestones, metrics, and regularly reviewing deliverables. Given the cultural implications for this work, communication—and in some cases, over-communication—is key. We want to ensure the wider company, key stakeholders, leadership, and others have a good sense of:

- what the strategy is,
- what the work plan is,
- how that work is being delivered, and
- what the results of the work are.

Bear in mind that these various audiences will have different communication needs. Senior leadership will need the overview and results, key leaders will require more depth, and team leads will want the details.

You will need to create a way of pulling out these overall details, design the communication for the right audiences, and update your audiences regularly (with weekly reports, for example). Also, be sure to message the entire company regularly where appropriate.

Review and improve

Bringing innersource into a company is an imperfect art and science. Companies vary, people vary, and approaches vary. You have to roll a program that meets the specific needs that you sought to understand at the beginning of this process.

As such, regularly evaluating your work and assessing how well it is going is critical, because you will need to make suitable changes. Doing this evaluation isn't easy; it can tap into people's fear of failure, but conquering such fear is important. Some things you do will not go well, and others will deliver sub-optimal results. Identifying the flaws that negatively affect your work and rectifying them is the point of your analysis.

8. Gather quantitative data

A first step is to gather data—in other words, numbers. For each project you work on, define a set of metrics you want to track and how you will read those metrics to determine success. Examples of these metrics are usage, contributions,

code, messages, participation, and other examples. You should never bring a project into the work plan unless you have a key way to measure it.

9. Survey your users

Analyzing numbers is convenient, because we usually have computers do all the work, but we also need to track human elements, such as happiness, empowerment, inclusivity, and other areas. Non-empirical analysis is difficult to do as these things don't usually map to numbers well.

A useful way to get data is to run an anonymous survey regularly that asks people how they feel about these human elements of the innersource experience. The staff must feel comfortable if they are to be critical. You need to give them express permission to criticize without consequences.

As important as the execution of the survey is, the wording of the questions and options is key, too. Wording and selections can often inadvertently influence responses, so I recommend you have a few people feed into the structure of the survey.

10. Update the strategy

Once you have this data, ask yourself and the team hard questions about what trends and patterns these illustrate and how you can refine the overall strategy, re-organize the backlog, and adjust how projects are prioritized, built, and managed with others. Of course, helping to bring innersource into a company is a complicated endeavor with countless different details, but I hope this article provided an overall framework in which you can fill in the gaps with the pieces that relate to your specific organization.

Author

Jono Bacon is a leading community manager, speaker, author, and podcaster. He is the founder of Jono Bacon Consulting, which provides community strategy/execution, developer workflow, and other services. He also previously served as director of community at GitHub, Canonical, XPRIZE, OpenAdvantage, and consulted and advised a range of organizations. Bacon is a prominent author and speaker on community management and best practice, and wrote the best-selling *The Art of Community* (O'Reilly), is the founder of the Community Leadership Summit, founder of the Community Leadership Forum, and is a regular keynote speaker at events about community management, leadership, and best practice. He also writes columns for Forbes and [Opensource.com](http://opensource.com). He lives in the San Francisco Bay Area in California with his wife, Erica, and their son, Jack. Follow him on Twitter: [@jonobacon](https://twitter.com/jonobacon)



7 cool little open source projects that stood out in 2016

BY D RUTH BAVOUCETT

IN THE EARLY DAYS of the open source movement, a lot of the attention was on operating systems, and later on large content management systems. These days, containers are mentioned regularly even in mainstream news outlets. The big tech stories are great, but they miss the other great activity in the niches of the open source space. I've rounded up seven interesting lesser-known projects from the past year. You can see more articles about projects like this in my Nooks and Crannies [1] column.

Mixxx: A DJ's Swiss Army knife

In the late 1980s, I worked as a disc jockey for a local radio station and as a mobile DJ for parties, weddings, and dances. It was a lot of fun and wasn't a hard business to start. You could set up shop with two CD players, a decent mixer and amplification system, and a *lot* of CDs. Thirty years later, what sticks in my head was lugging all those CDs. Sampling was virtually unheard of for mobile DJs back then. Commonly available computers were expensive and slow enough that playing music from a PC was risky—it would hang while buffering at some point during a show.

The technology for DJs has dramatically changed in the intervening years. An inexpensive computer can handle everything we could do in the '80s, and much more. Mixxx [2] is an open source system that acts as a mixer and sampler for a mobile or club DJ. It's incredibly feature-rich, with four

input decks and four sampling decks, tools for synchronization during cross-fades, key detection and pitch shifting for harmonic mixing, and built-in effects. You can play your mixes live, record them, or stream them over the Internet using SHOUTcast [3] or Icecast [4]. Mixxx has an amazing music library system to let you organize your music in any way you like, giving quick access to songs in the library. Mixxx has comprehensive support for DJ hardware controllers, including more than 80 of the most popular models.

Mixxx, mixing Madonna with herself. Don't judge my music!



I took a look at the Mixxx community, and it's a robust, well-organized group of dedicated souls with broad diversity. There are forums, a huge wiki, and excellent bug and release tracking all set up and well-established. The community has adopted a well-written code of conduct [5] that discourages problematic behavior among members. On the wiki, you'll find great tips on hardware [6] for use with Mixxx, and a Getting Involved page [7] that talks about how even non-programmers can get plugged into the Mixxx community. Mixxx is a C++ application and is available under the GPL v2 license [8] for Windows, OS X, and Linux. Version 2.0 came out in December 2015.

sofa: Not a place for lazy data scientists

The R project [9] is a widely used software environment for statistical computing, and its use in data analysis continues



Image by: Internet Archive Book Images. Modified by Opensource.com.

to rise. The rOpenSci project [10] is producing tools to allow R to access large repositories of science data and full-text journal articles. One of the tools from the rOpenSci team is sofa [11]. Sofa is a kit of tools for allowing easy access in R to CouchDB [12] NoSQL document databases.

To begin using sofa in a program, you create a server handle, cleverly called a cushion:

```
myCushion <- Cushion$new(  
  host = "myhost.mynet.org",  
  transport = 'https',  
  port = NULL,  
  user = 'username',  
  pass = 'mypassword'  
)
```

Once you have a cushion, you can connect to any database or create and destroy databases. A database creation is as simple as:

```
db_create(myCushion, 'felines')
```

Once you've created a JSON or XML document, inserting it into the database is easy:

```
my_kitty <- '{"name":"Midnight", "color":"black",  
  "furry":true, "size":'large', "gender":"tom"}'  
doc_create(myCushion, dbname="felines", my_kitty)
```

You can optionally specify a fourth parameter to the `doc_create` to force the document ID to a known value. If you don't use it, the default is to use an automatically generated hash key.

Ready to query? It's straightforward:

```
db_query(myCushion, dbname="felines", selector=list(size = 'large'))$docs
```

This query returns a structure with the full document, including ID and revision for all documents that have a size field of large. There are tools that let you limit your return to specific fields and much more complicated searching than this example.

Sofa is a great tool for unlocking the data in CouchDB; if big data is your game, it might be the right tool for you. All of rOpenSci's work is MIT licensed and has a contributor code of conduct [13]. The code is available on GitHub [14].

PANOPTES: Open source astronomy

I interviewed Jennifer Tong and Wilfred Gee from the PANOPTES project [15] in April. I enjoyed their OSCON conference [16] presentation, and have been following their website [17] for more information about this great project. PANOPTES (Panoptic Astronomical Networked OPTical observatory for Transiting Exoplanets Survey) is a project harnessing the power of interested citizen scientists around the world in

building a network of robotic telescopes. This global array will detect transiting exoplanets for further examination by larger earth- and space-based telescopes.

Each participant builds a robotic telescope using off-the-shelf equipment: A commercially available camera, an Arduino Micro, an Intel NUC, and other easy-to-find components. You can buy most of the components from Amazon, and the total cost is less than US\$ 5,000. These telescopes will share their data with the project servers, and image analysis from many units will go into finding potential results. When desired by the owner, an individual telescope may be taken offline for unrelated observations. This makes it an ideal project for schools and science educators, as they get to be involved in a larger global project, and have access to a quality telescope for their local teaching work.

The PANOPTES project is continuing to refine its hardware design. Beta testers for the system are welcome to build one according to the instructions on the website. Much work also is being done on the centralized observatory control system, which directs each of the robotic telescopes in their observations. This is a project worth watching not just for the science it can do, but for learning about a process for engaging people in other distributed-science teams.

OpenAPS: Enhancing life quality for Type 1 diabetes patients

One of the high-water marks of OSCON this year for me was Dana Lewis's keynote [18] about OpenAPS [19], a simplified artificial pancreas for Type 1 diabetes patients. OpenAPS uses currently available medical tools—diabetes pumps and continuous glucose monitors, paired with Raspberry Pi or Intel Edison computers. The system takes care of the complicated calculations that a pump user normally must do to keep their blood glucose levels steady. By updating every five minutes, it's doing the work in near-real-time 24-hours a day. This means less hassle for the user during the day, and better sleep at night.

The core belief of this effort is that by open sourcing the project code, they can make APS (Artificial Pancreas System) technology available to more people more quickly than current closed-source APS medical research. The OpenAPS team has taken a conservative approach to dosing to help keep it safe as well as effective.

More than 90 units have been deployed, with more than 30 of those in the summer and fall of 2016, and about a third of the OpenAPS users are children. The community is user-led, and new users are welcome. The documentation [20] for building one of your own is freely available and deeply detailed. It explains not only the how but the why; with an emphasis on patient safety.

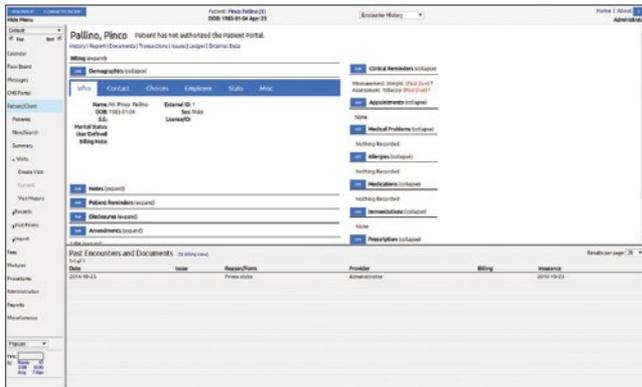
Many of us who work in IT want to make lives better by making computers do interesting things, and OpenAPS is one of the best examples I've found of using our open source skills to help our loved ones.

OpenEMR: Tools for keeping a medical practice organized

I've always had the impression that doctor's offices are quite complex places to work. Lots of diverse information must be kept and secured, and the functions that use patient information are equally diverse. Recently I was surprised to learn about OpenEMR [21], an open source practice-management system. It's been around a while, having been first developed in 2001, under another name. The first release was out in 2002, under the GPL V2.0 license.

The feature list is impressive. Besides a robust patient records system, OpenEMR has a built-in medical billing system that can take part in the major billing clearing houses using ANSI ASC X12 [22] and can use any coding system desired. OpenEMR also handles online prescription ordering, using ePrescribe [23], as well as more traditional print, fax, or email methods. If installed as a service, OpenEMR also has a patient portal system to handle communications with patients. If an office already has a popular patient portal system in use, the system can communicate via API and use that instead.

The OpenEMR Patient Information screen



OpenEMR offers a staggering list of reports, and one feature that caught my eye is that it is supported in more than 20 languages and has the ability to support multiple languages in the same clinic. This is a nice feature to have in diverse cities with large populations of non-local-language speakers, as each user can choose their own language set. OpenEMR is fully UTF-8 compliant.

With an estimated 5,000+ installs in the US alone, OpenEMR has a thriving community of users and developers. The OEMR Foundation [24] is a US charitable organization set up to support OpenEMR adoption and development to promote more affordable healthcare for all. There are highly active forums for users and developers to discuss their needs and get help with the application. More than 30 companies globally are providing commercial hosting and/or support of OpenEMR. It's not a Thneed [25] (a fine thing that all people need, according to Dr. Seuss), but it's certainly a great open source success story.

bibisco: The novelist's friend

In September 2016, I covered bibisco in my column [26]. Written by Andrea Feccomandi, bibisco [27] is an open source alternative to programs like Scrivener [28]. I was impressed with the polished feature set, and as I said then, I've been moving my own novel and other writings into it. There was only one thing holding me back from full-throated enthusiasm for this project, and that was a lack of an OS X client. Andrea packaged it for Windows and 32- and 64-bit Linux. A friend of mine made things sort of work, with a lot of finesse, on his Mac, but it was a mystery to me how to do it.

Bibisco has really revolutionized the way I'm writing my novel. For each of the scenes in a chapter, I've got a separate entry with a one-line title describing that scene. I can use these entries as a storyboard for the chapter, rearranging them as I desire. Each chapter can be tagged with locations and characters, and I can get reports on how often those appear across the book. I've made a fair bit of progress since finishing the switchover, and I couldn't be happier. There was that one nagging little problem, though. I could only work on the novel at home where I have a Windows machine; my Macbook just couldn't do it. Imagine my surprise a few days after the release of the article when Andrea commented and told the world that he had purchased a Mac so that he could release the OS X client. Then, a month later, he commented again, announcing the release of the OS X client on the website.

Pa11y: Automated accessibility testing

One oft-ignored element of web design is accessibility. Many of the guidelines are hard to test, but there are a number of specific, testable criteria that designers can use—if they have the right tool for the job. Enter Pa11y [29], a suite of tools for one-off or automated testing of web pages for accessibility against a broad list of criteria sets. Installation of the basic toolkit is easy with npm, so you can test your pages right away and get feedback and specific suggestions for improvement. If your organization would like to do ongoing or periodic testing of pages, installing the dashboard and web service is straightforward. You can see a demo of this dashboard at demo.pa11y.org [30].

The community is actively working on a new version of its website, which includes much more detailed information for developers and others wanting to contribute. The group has adopted a code of conduct [31] adapted from the Contributor Covenant [32]. They are also beginning work on a new, more refined version of the dashboard application called Sidekick. Coding has begun on that project, which the team is dedicated to designing and developing completely in the open, in a GitHub repository [33].

And more

Every year, several hundred new open source projects appear. As much as I'd love to, covering them all is not

possible. The projects in this roundup are only a few of the many worth watching in the next year. Let us know about your project—submit an article proposal [34].

Resources

1. <https://opensource.com/tags/nooks-and-crannies>
2. <http://mixxx.org>
3. <https://www.shoutcast.com/>
4. <http://icecast.org/>
5. https://github.com/mixxxdj/mixxx/blob/master/CODE_OF_CONDUCT.md
6. http://mixxx.org/wiki/doku.php/hardware_compatibility
7. http://mixxx.org/wiki/doku.php/getting_involved
8. <https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>
9. <https://www.r-project.org/>
10. <http://ropensci.org/>
11. <https://github.com/ropensci/sofa>
12. <http://couchdb.apache.org/>
13. <https://github.com/ropensci/sofa/blob/master/CONDUCT.md>
14. <https://github.com/ropensci/sofa>
15. <https://opensource.com/life/16/4/oscon-interview-wilfred-gee-panoptes>
16. <http://conferences.oreilly.com/oscon>
17. <http://www.projectpanoptes.org>
18. <https://opensource.com/life/16/5/openaps-oscon-dana-lewis>
19. <https://openaps.org>
20. <https://openaps.readthedocs.io/en/latest/index.html>
21. <http://www.open-emr.org/>
22. <http://www.x12.org>
23. http://www.open-emr.org/wiki/index.php/OpenEMR_ePrescribe
24. <http://www.oemr.org/>
25. <http://seuss.wikia.com/wiki/Thneed>
26. <https://opensource.com/life/16/9/bibisco-tool-novelists>
27. <http://www.bibisco.com>
28. <https://www.literatureandlatte.com/scrivener.php>
29. <http://pa11y.org/>
30. <http://demo.pa11y.org/>
31. <http://pa11y.github.io/contributing/code-of-conduct/>
32. <http://contributor-covenant.org/version/1/4/>
33. <https://github.com/pa11y/sidekick>
34. <https://opensource.com/story>

Author

D Ruth Bavousett has been a system administrator and software developer for a long, long time, getting her professional start on a VAX 11/780, way back when. She spent a lot of her career (so far) serving the technology needs of libraries, and has been a contributor since 2008 to the Koha open source library automation suite. Ruth is currently a Perl Developer at cPanel in Houston, and also serves as chief of staff for one large cat.



The conversation continues—in the open.

Download 3 free books today, and join us at opensource.com/open-organization



9 lessons from 25 years of Linux kernel development

• BY GREG KROAH-HARTMAN

BECAUSE THE LINUX KERNEL

community celebrated a quarter-century of development in 2016, many people have asked us the secret to the project's longevity and success. I usually laugh and joke that we really have no idea how we got here. The project has faced many disagreements and challenges along the way. But seriously, the reason we've made it this far has a lot to do with the community's capacity for introspection and change.

About 16 years ago, most of the kernel developers had never met each other in person—we'd only ever interacted over email—and so Ted T'so [1] came up with the idea of a Kernel Summit [2]. Now every year kernel developers make a point to gather in person to work out technical issues and, crucially, to review what we did right and what we did wrong over the past year. Developers can openly and honestly discuss how they interact with each other and how the development process works. And then we make changes that improve the process. We make new tools, like Git [3], and constantly change how we work together.

Over time, this evolution has created a resiliency that has allowed the project to go from one strength to the next while avoiding the forks that have split the resources of competing

projects. It may be many years before we fully understand the keys to the Linux kernel's success, but there are a few lessons that stand out even now.

1. Short release cycles are important.

In the early days of the Linux project, a new major kernel release only came once every few years. That meant considerable delays in getting new features to users, which was frustrating to users and distributors alike. But, more importantly, such long cycles meant that huge amounts of code had to be integrated at once, and that there was a great deal of pressure to get code into the next release, even if it wasn't ready.

Short cycles address all of these problems. New code is quickly made available in a stable release. Integrating new code on a nearly constant basis makes it possible to bring in even fundamental changes with minimal disruption. And developers know that if they miss one release cycle, there will be another one in two months, so there is little incentive to try to merge code prematurely.

2. Process scalability requires a distributed, hierarchical development model.

A long time ago, all changes went directly to Linus Torvalds, but that quickly proved unwieldy as no one single person can keep up with a project as diverse as an operating system kernel. Very early the idea of maintainers of different areas of the kernel came about, where the responsibility of a portion of the kernel was assigned to an individual familiar with that area. Examples of this is networking, wireless, different driver subsystems like PCI or USB, or different individual filesystems like ext2 or vfat. Spreading out the responsibility for code review and integration across many hundreds of maintainers gives the project the resources to cope with tens of thousands of changes per release, without sacrificing review or quality.

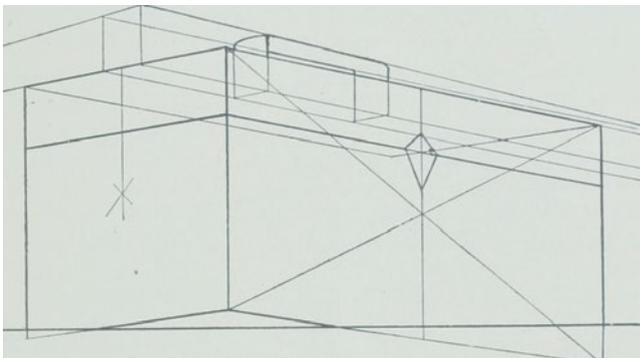


Image by: Internet Archive Book Images. Modified by Opensource.com.

3. Tools matter.

Kernel development struggled to scale until the advent of the BitKeeper [4] source-code management system changed the community's practices nearly overnight; the switch to Git brought about another leap forward. Without the right tools, a project like the kernel would simply be unable to function without collapsing under its own weight.

Without the right tools, a project like the kernel would simply be unable to function without collapsing under its own weight.

4. The kernel's strongly consensus-oriented model is important.

As a general rule, a proposed change will not be merged if a respected developer is opposed to it. This can be intensely frustrating to developers who find code they have put months into blocked on the mailing list. But it also ensures that the kernel remains suited to a wide range of users and problems. No particular user community is able to make changes at the expense of other groups. As a result, we have a single codebase that scales from tiny systems to supercomputers and that is suitable for a huge range of uses.

5. The kernel's strong "no regressions" rule is also important.

Over a decade ago the kernel developer community made the promise that if a given kernel works in a specific setting, all subsequent kernels will work there, too. If the community finds out that a change caused a regression, they work very quickly to address the issue. The rule gives users assurance that upgrades will not break their systems; as a result, they are willing to follow the kernel as it develops new capabilities.

6. Corporate participation in the process is crucial, but no single company dominates kernel development.

Some 5,062 individual developers representing nearly 500 corporations have contributed to the Linux kernel since the 3.18 release in December of 2014. The majority of developers are paid for their work—and the changes they make serve the companies they work for. But, although any company can improve the kernel for its specific needs, no company can drive development in directions that hurt the others or restrict what the kernel can do.

7. There should be no internal boundaries within the project.

Kernel developers are necessarily focused on specific parts of the kernel, but any developer can make a change to any part of the kernel if the change can be justified. As a result, problems are fixed where they originate rather than being worked around, developers have a wider view of the kernel as a whole, and even the most recalcitrant maintainer cannot indefinitely stall needed progress in any given subsystem.

8. The kernel shows that major developments can spring from small beginnings.

The original 0.01 kernel was a mere 10,000 lines of code; now it grows by more than that every two days. Some of the rudimentary, tiny features that developers are adding now will develop into significant subsystems in the future.

9. Above all, 25 years of kernel history show that sustained, cooperative effort can bring about common resources that no group would have been able to develop on its own.

Since 2005, some 14,000 individual developers from more than 1,300 different companies have contributed to the kernel. The Linux kernel, thus, has become a common resource developed on a massive scale by companies that are fierce competitors in other areas.

These takeaways, and more detailed information on Linux kernel development, can be found in the 2016 Linux Kernel Development Report [5], co-written with LWN's [6] Jon Corbet.

Libby Clark contributed to this article.

Resources

- [1] <https://think.org/tytso/>
- [2] <http://events.linuxfoundation.org/events/linux-kernel-summit>
- [3] https://opensource.com/search/apachesolr_search/git
- [4] <http://www.bitkeeper.com/>
- [5] <https://www.linux.com/publications/linux-kernel-development-how-fast-it-going-who-doing-it-what-they-are-doing-and-who-5>
- [6] <https://lwn.net/>

Author • • • • •
Greg is a Linux kernel maintainer and a Linux Foundation fellow.



A tour of Google's 2016 open source releases

BY JOSH SIMMONS

OPEN SOURCE SOFTWARE enables Google to build things quickly and efficiently without reinventing the wheel, allowing us to focus on solving new problems. We stand on the shoulders of giants, and we know it. This is why we support open source [1] and make it easy for Googlers to release the projects they're working on internally as open source.

We've released more than 20-million lines of open source code to date, including projects such as Android, Angular, Chromium, Kubernetes [2], and TensorFlow. Our releases also include many projects you may not be familiar with, such as Cartographer [3], Omnitone [4], and Yeoman [5].

Looking back at the projects we've open sourced in 2016, there's a lot to be excited about. We have released open source software [6], hardware [7], and datasets [8]. Let's take a look at some of this year's releases.

Seesaw

Seesaw [9] is a Linux Virtual Server-based [10] load-balancing platform developed in Go by our site reliability engineers. Seesaw, like many projects, was built to scratch our own itch.

From our blog post announcing its release [11]: "We needed the ability to handle traffic for unicast and any-cast VIPs, perform load balancing with NAT and DSR (also known as DR), and perform adequate health checks against the backends. Above all we wanted a platform

that allowed for ease of management, including automated deployment of configuration changes."

Vendor Security Assessment Questionnaire (VSAQ)

We assess the security of hundreds of vendors every year, and have developed a process to automate much of the initial information-gathering with VSAQ [12]. Many vendors found our questionnaires intuitive and flexible, so we decided to share them. The VSAQ framework includes four extensible questionnaire templates covering web applications, privacy programs, infrastructure, and physical and data center security. You can learn more about it in our announcement blog post [13].

OpenThread

OpenThread [14], released by Nest [15], is a complete implementation of the Thread [16] protocol for connected devices in the home. This is especially important because of the fragmentation we're seeing in this space. Development of OpenThread is supported by ARM, Microsoft, Qualcomm, Texas Instruments, and other major vendors.

Magenta

Can we use machine learning to create compelling art and music? That's the question that animates Magenta [17], a TensorFlow-based project from the Google Brain team [18]. The aim is to advance the state of the art in machine intelligence for music and art generation, and to build



Image by: Travis Wise. CC BY-SA 2.0

a collaborative community of artists, coders, and machine-learning researchers. Read the release announcement [19] for more information.

Omnitone

Virtual reality (VR) isn't nearly as immersive without spatial audio, and much of VR development is taking place on proprietary platforms. Omnitone [20] is an open library built by members of the Chrome team that brings spatial audio to the browser. Omnitone builds on standard Web Audio APIs to deliver an immersive experience and can be used alongside projects such as WebVR [21]. Find out more in our blog post announcing the project's release [22].

Science Journal

Today's smartphones are packed with sensors that can tell us interesting things about the world around us. We launched Science Journal [23] to help educators, students, and citizen scientists tap into those sensors. You can learn more about the project in our announcement blog post [24].

Cartographer

Cartographer [25] is a library for real-time simultaneous localization and mapping (SLAM) in 2D and 3D with Robot Operating System (ROS) support [26]. Combining data from a variety of sensors, this library computes positioning and maps surroundings. This is a key element of self-driving cars, UAVs, and robotics, as well as efforts to map the insides of famous buildings [27]. More information on Cartographer can be found in our blog post announcing its release [28].

This collection is just a small sampling of what we've released this year. Follow the Google Open Source Blog [29] to stay apprised of Google's open source software, hardware, and data releases.

Resources

1. <https://developers.google.com/open-source/>
2. <https://opensource.com/resources/what-is-kubernetes>
3. <https://opensource.googleblog.com/2016/10/introducing-cartographer.html>
4. <https://opensource.googleblog.com/2016/07/omnitone-spatial-audio-on-web.html>

5. <http://yeoman.io/>
6. <https://opensource.googleblog.com/2016/04/cctz-v20-now-with-more-civil-time.html>
7. <https://opensource.googleblog.com/2016/07/announcing-open-source-adc-board-for.html>
8. <https://opensource.googleblog.com/2016/10/introducing-open-images-dataset.html>
9. <https://github.com/google/seesaw>
10. <http://www.linuxvirtualserver.org/>
11. <https://opensource.googleblog.com/2016/01/seesaw-scalable-and-robust-load.html>
12. <https://github.com/google/vsaq>
13. <https://opensource.googleblog.com/2016/03/scalable-vendor-security-reviews.html>
14. <http://github.com/openthread/openthread>
15. <https://nest.com/>
16. <http://threadgroup.org/>
17. <http://github.com/tensorflow/magenta>
18. <https://research.google.com/teams/brain/>
19. <https://magenta.tensorflow.org/welcome-to-magenta>
20. <https://github.com/GoogleChrome/omnitone>
21. <https://webvr.info/>
22. <http://google-opensource.blogspot.com/2016/07/omnitone-spatial-audio-on-web.html>
23. <http://googleforeducation.blogspot.com/2016/05/inspiring-future-makers-and-scientists.html>
24. <http://google-opensource.blogspot.com/2016/08/opening-up-science-journal.html>
25. <https://github.com/googlecartographer>
26. <http://www.ros.org/about-ros/>
27. <https://www.zeitgeistminds.com/talk/5604289821016064/amit-sood-the-peoples-museum-amit-sood>
28. <http://opensource.googleblog.com/2016/10/introducing-cartographer.html>
29. <https://opensource.googleblog.com/>

Author

Josh Simmons is a community organizer and short stack web developer who works on the Google open source outreach team and sits on the OSI board of directors. You can find him on Twitter: @joshsimmons



TOP 10 Linux news stories of 2016

BY SCOTT NESBITT

LINUX made quite a few headlines in 2016. Although 2016 wasn't the much-anticipated *Year of Linux on the Desktop*, it was still a big year for the open source movement's poster child. Let's take a look at 10 of the biggest Linux news stories from the past year.

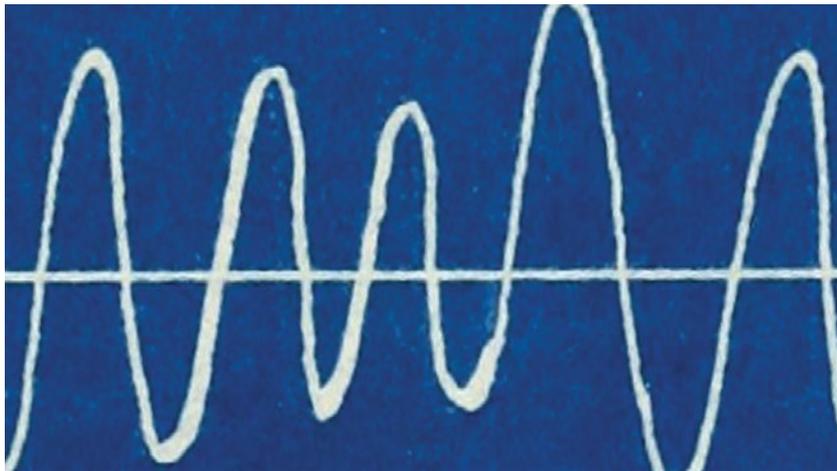


Image by: Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0

development, was pragmatic rather than ideological, the kernel was designed to be practical, and Linux managed to rally a community around itself.

1. Linux turns 25

They grow up so quickly. It's hard to believe that 25 years ago Linus Torvalds announced [1] to the comp.os.minix Usenet group that he was "doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones." Since 1991, Linux has grown beyond even Torvalds' dreams. It's not a stretch to say that Linux is everywhere. Corporations large and small use Linux, and it powers computers, mobile devices, and connected hardware. Critical infrastructure relies on the stability and flexibility of Linux.

Why, despite its humble beginnings and the opposition it's faced from certain corners of the technology world, did Linux not only survive but also thrive? Christopher Tozzi, writing at The VAR Guy website [2], suggests that Linux succeeded for four reasons: It adopted a decentralized approach to

2. Fedora 25 becomes the first major Linux distro to ship with Wayland enabled by default

As we reported in November 2016, Fedora 25 rolled out with Wayland as the default display protocol on compatible machines. Wayland was development as modern, simpler replacement for the X Window System. In a March 2016 blog post, "Why Wayland Anyway?" [3], developer Matthias Clasen outlined benefits of Wayland, including that it isolates clients from each other; it's a better fit for modern, compositing-based display system (i.e., it doesn't include unnecessary "baggage," such as core fonts or core rendering); and it will be a good foundation for enabling features that are hard to support under X (e.g., input transformation or smooth transitions between composited desktop and fullscreen clients).

Fedora 25 with Wayland received rave reviews from users. In an Ars Technica article, "Fedora 25: With Wayland, Linux has never been easier (or more handsome)" [4],

writer Scott Gilbertson says, “This is perhaps the biggest change to come in the Linux world since the move to systemd. However, unlike that systemd transition, the switch to Wayland was so seamless I had to logout and double check that I was in fact using Wayland.” And in his article on TechRepublic, “Fedora 25: Bleeding edge and bloody brilliant” [5], Jack Wallen says, “There are those in the community who believe it’s still too early to be using Wayland as the default compositor protocol. However, after using Fedora 25 for a while, it’s quite clear that Wayland is well beyond ready. GNOME on Wayland was much improved. Windows were smoother and faster to open and the stability of the desktop was a slight step ahead of X11.” Keep an eye out for Wayland to make more headlines in 2017.

3. Microsoft cozies up to Linux

Over the past couple of years, Microsoft has been backtracking on its aggressive rhetoric against Linux and open source. In fact, the company has been actively and publicly embracing its one-time sworn enemy. But in 2016, the company further shocked the Linux and open source world with several moves. Microsoft joined the Linux Foundation [6] as a Platinum member, a membership tier that includes companies such as Cisco, Fujitsu, IBM, and Intel. Other smaller—but more surprising—moves were making PowerShell [7] and SQL Server 2016 [8] available for Linux (however, only PowerShell is open source), and bringing the Bash shell to Windows 10 [9].

These moves have been greeted with open arms, and with more than a bit of skepticism. On one hand, analysts anticipate Microsoft’s platform to integrate and operate better with Linux and open source applications. On the other hand, many people in the open source community view Microsoft’s coziness with Linux and open source as another step in Microsoft’s embrace, extend, and extinguish [10] strategy.

4. Munich makes noise about abandoning Linux

In 2004, the city of Munich in Germany made headlines by starting a project to replace Windows and Microsoft Office on thousands of PCs with a custom Linux distribution called LiMux, OpenOffice.org, and, later, LibreOffice. That experiment seems to be over, as Munich’s government is debating a report [11] that recommends moving back to Microsoft’s products. The announcement was surprising because the city declared the shift to Linux and open source a success in 2013 [12]. Although most users haven’t had a problem with the migration, various municipal departments are pushing to go back to using Windows and Microsoft Office. But it’s not all bleak news for Linux in Munich. If the shift does happen, the city plans to use LiMux alongside Windows, and LiMux would continue to be updated.

5. Russia moves to embrace Linux

As Munich considers moving away from Linux, Russia is moving toward embracing Linux and open source. The Duma, Russia’s parliament, drafted a law [13] giving preference to open source software and requiring government agencies to justify purchases of proprietary solutions. A big part of this move is based around security concerns. Most of the country’s digital infrastructure runs on software from US-based companies, such as Microsoft, Oracle, and IBM. The Russian government claims that the software from those companies poses security risks, such as hidden backdoors into the software. Although the public and private sectors in Russia are warm to open source, a few barriers must be overcome, such as a lack of local companies who can provide support, and few open source companies have offices in Russia.

6. Linux powers the world’s fastest supercomputers

For years, Linux has been at the heart of the fastest, most powerful supercomputers on the planet. This trend continued in 2016 [14] with Linux ruling the roost as the operating system of choice for makers of supercomputers. In the TOP500 2016 [15] rankings of high performance computers, Linux powers 497 of the 500 machines on the list. That means 99.4% of the fastest supercomputers use Linux as their software heart. The other three run IBM’s AIX (a UNIX derivative), and the highest-ranking of those computers comes in at 281st place on the list.

7. Automotive Grade Linux heads toward becoming a de-facto standard

If Linux isn’t already in your car, it will be soon. That’s the idea behind the Linux Foundation’s Automotive Grade Linux [16] (AGL) project. Although Automotive Grade Linux has been kicking around for a few years, it picked up additional steam [17] in 2016 when several technology companies, including Oracle, Qualcomm, and Texas Instruments, joined the project.

“The automotive industry needs a standard open operating system and framework to enable automakers and suppliers to quickly bring smartphone-like capabilities to the car,” Dan Cauchy, the project’s general manager, says in an early 2016 project update [18]. The tech firms that joined the project along with a number of automakers are working to build a common, standard platform for controlling all aspects of a car’s operations.

8. Popular Linux distribution Mythbuntu packs it in

The year 2016 saw the end of the line for one popular Linux distribution. The developers behind Mythbuntu, the backbone of the open source MythTV digital video recorder, decided to discontinue the project [19]. Thomas Mashos, a coder with the project, says a lack of developers led to the decision. The team went from 10 developers down to 2, which caused delays to updates and new Mythbuntu releases.

2016 Hacktoberfest ignites open source participation

BY BEN COTTON

DIGITALOCEAN [1] launched Hacktoberfest [2] in 2014 to encourage contribution to open source projects. The event was a clear success, and in terms of attendance and participation goals reached, it's also clear that Hacktoberfest has become a powerful force in driving contributions to open source. The lure of a t-shirt and specific, time-limited goals help new contributors get started and encourage existing contributors to rededicate themselves and their efforts.

The third year continued the momentum. In fact, early in the month, community management manager Daniel Zaltsman [3] told Opensource.com that 2016 already surpassed last year's results.

At month's end, 10,227 of the 29,616 registered participants had opened four pull requests in order to complete Hacktoberfest successfully. Although the success rate of 34.5% was down slightly from 2015, the number of participants who completed the event was up dramatically. The number of people who completed Hacktoberfest 2016 was more than two-thirds of last year's total registrant count. Registration was up more 97% from last year, with success up 79%. And 92,569 pull requests were opened, which is an 89% increase. Momentum may be a bit of an understatement.

One interesting number that stood out in the final analysis was the nearly 1:1 ratio of registered participants to contributed projects—29,287 repositories received pull requests from Hacktoberfest 2016 participants. `homebrew-cask` [4] received the most (310), with `home-assistant` [5] (265) and `manageiq` [6] (231) close behind.

Although the basic mechanics of participation did not change for this year, DigitalOcean made a few changes to improve the experience for project maintainers. This year, maintainers were encouraged to create a Hacktoberfest label and apply that to issues where newcomers could make a contribution. GitHub provides site-wide search, so the 15,000-plus labeled issues could be found quickly by people who didn't know which project to start on, but who wanted to make a contribution. Additionally, pull requests marked "Invalid" were not counted toward a participant's total. The net result was to prevent trivial or no-op pull requests from overburdening project maintainers.

Some Hacktoberfest participants have gone on to be project maintainers. "I actually started contributing to open source in a meaningful way because of Hacktoberfest. Homebrew Cask was a convenient tool in my daily usage, and Hackto-



Image by: Library of Congress. Modified by Opensource.com.

berfest provided an extra incentive to contribute back," participant Aditya Dalal [7] told DigitalOcean. "Over time, I continued contributing and ended up as a maintainer, focusing on triaging issues and making the contribution process as simple as possible, which I like to think we have succeeded at."

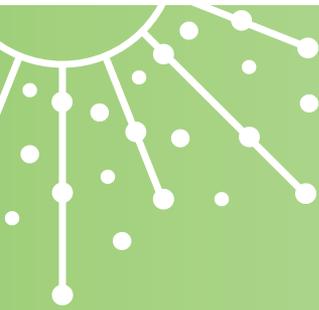
His participation illustrates the momentum that DigitalOcean has tried to build into the Hacktoberfest events. Forty events were held in 29 cities across 12 countries. For 2016, they created a Hacktoberfest-themed meetup kit to encourage social participation. Although this year got more internal teams involved, for 2017 organizers want to further increase community participation.

Interesting trivia stands out from the 2016 stats. Contributions were spread roughly equally across the days of the week, but Monday had the highest number of pull requests. Monday also happened to be the last day of the month, which meant a flurry of last-minute pull requests as people tried to earn their t-shirt. When the last day's pull requests are removed, Monday becomes the second slowest day. When normalized for the number of times the day occurred in the month, Wednesday was the most active day, with the weekends being the slowest.

Can 2017 continue building on the first three years? We'll find out next October.

Resources

- [1] <https://www.digitalocean.com/>
- [2] <https://hacktoberfest.digitalocean.com/>
- [3] <https://twitter.com/Zaltsman>
- [4] <https://github.com/caskroom/homebrew-cask/>
- [5] <https://github.com/home-assistant/home-assistant/>
- [6] <https://github.com/ManageIQ/manageiq/>
- [7] <https://github.com/adidalal>



Most Playful

Top 7 Linux games of 2016

.....BY ROBIN MULWIJK

IN THE 2015

Open Source Yearbook [1] I looked at the best open source games [2]. This year, with the continuing growth of Linux gaming, I've rounded up the top Linux games on Steam [3]. On an average day, some of these games [4] are played by almost a million players. I included both free and non-free games on my list.

Dota 2

By far the most-played game on Steam is Dota 2 [5]. At times, Dota 2 reaches close to a million concurrent online players. This action and strategy game originates from a Warcraft 3 modification. Players can select from hundreds of heroes, to team up and "battle their Dire counterparts to control a gorgeous fantasy landscape, waging campaigns of cunning, stealth, and outright warfare." Dota 2 is free to play, and exclusively available on Steam [6].

Figure 1: Dota 2 screenshot by Colony of Gamers. CC BY-NC 2.0



Image by: NASA on the Commons and Internet Archive Book Images. Modified by Opensource.com.

Counter-Strike: Global Offensive

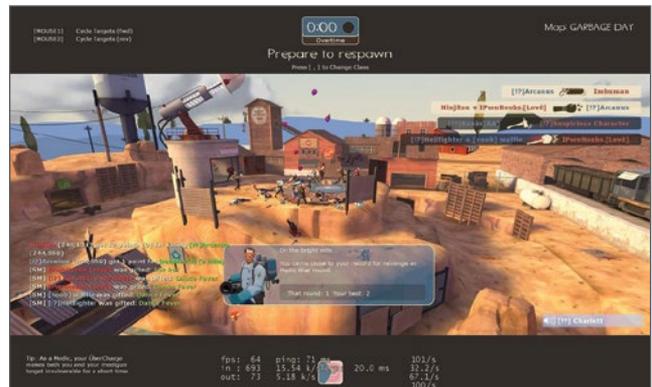
The classic first-person shooter Counter-Strike [7] game was launched 14 years ago and is now touted as the "world's #1 online action game". Counter-Strike: Global Offensive [8] has since seen many updates, such as new maps, characters, and weapons, based on

the classic Counter-Strike. Counter-Strike: Global Offensive is available on Steam [9] for US\$ 14.99. Bundle pricing is also available.

Team Fortress 2

Team Fortress 2 [10] is a multi-player action game, developed by Valve [11], creator of Steam. Team Fortress 2 has seen more than 400 updates in the past six years, and continues to be updated with new game modes, maps, and more. Team Fortress 2 is available on Steam [12] and is free to play.

Figure 2: Team Fortress 2 screenshot by Terry Robinson. CC BY-NC 2.0





ARK: Survival Evolved

ARK: Survival Evolved [13] is an action-filled role playing game that will put you through lots of adventure—from survival, to riding dinosaurs. “As a man or woman stranded naked, freezing and starving on a mysterious island, you must hunt, harvest, craft items, grow crops, and build shelters to survive,” the site explains. ARK: Survival Evolved is available on Steam [14] (Early Access) for US\$ 29.99.

Figure 3: ARK: Survival Evolved screenshot by Tamahikari Tammas. CC BY-NC 2.0



Football Manager 2017

In Football Manager 2017 [15] you can pick from 2,500 real clubs and take control over your favorite team. Football Manager 2017 claims to be “the most realistic and immersive football management game to date.” The game allows you to control transfers and who plays or sits on the bench, all while you watch the game live through the 3D match engine. Football Manager 2017 is available in the Steam store [16] for US\$ 59.99.

Garry's Mod

Garry's Mod [17] is a sandbox game and lacks any objectives. The makers provide you with tools, and leave you to

build and play. “You spawn objects and weld them together to create your own contraptions—whether that’s a car, a rocket, a catapult or something that doesn’t have a name yet—that’s up to you,” the site explains. Garry’s Mod is available on Steam [18] for \$US 9.99.

Figure 4: Garry's Mod screenshot by DoctorButtsMD. CC BY-NC 2.0



Sid Meier's Civilization V

Civilization [19] is a turn-based strategy game in which you attempt to build an empire. If you can stand the test of time, you can become ruler of the world “by establishing and leading a civilization from the Stone Age to the Information Age.”

The game allows you to play its default maps and scenarios. But you can also create your own or download player-created maps and scenarios. Civilization is available on Steam [20] for US\$ 69.99.

Honorable mentions

Because this list focuses on Linux games available on Steam, I also want to mention two other popular open source games that can be played on Linux and that have a native client:

The Battle for Wesnoth

Gamers that prefer a turn-based tactical strategy game will love The Battle for Wesnoth [21], which is free. Making this game unique is its high fantasy theme. The game allows you to build your own army, out of 200 unit types, and it includes 16 races, 6 major factions, and hundreds of years of history. As player, you are the heir to Wesnoth, and you fight to regain its throne. “The world of Wesnoth is absolutely huge and only limited by your creativity—make your own custom units, create your own maps, and write your own scenarios or even full-blown campaigns,” the site explains.

0 A.D.

0 A.D. [22] is a free historical real-time strategy game. 0 A.D. is unique in its graphics and rendering. As leader of an ancient civilization, your goal is to gather the resources you need to raise a military force and dominate your enemies. Your civilizations and battles take part over the millennium of 500 B.C. to 500 A.D. (hence, the name of this game being the midpoint: 0 A.D.)

Resources

(For links to original images in this article, visit: <http://red.ht/2i2Y5sB>)

- [1] <https://opensource.com/yearbook/2015>
- [2] <https://opensource.com/life/15/12/top-5-open-gaming>
- [3] <http://store.steampowered.com/linux>
- [4] <http://store.steampowered.com/stats/>
- [5] <http://blog.dota2.com/?l=english>
- [6] <http://store.steampowered.com/app/570/>
- [7] <http://blog.counter-strike.net/>
- [8] <http://blog.counter-strike.net/>
- [9] <http://store.steampowered.com/app/730/>
- [10] <http://www.teamfortress.com/index.php>
- [11] <http://www.valvesoftware.com/>
- [12] <http://store.steampowered.com/app/440/>
- [13] <http://www.playark.com/>
- [14] <http://store.steampowered.com/app/346110/>
- [15] <http://www.footballmanager.com/>
- [16] <http://store.steampowered.com/app/482730/>
- [17] <http://www.garrysmud.com/>
- [18] <http://store.steampowered.com/app/4000/>
- [19] <https://civilization.com/>
- [20] <http://store.steampowered.com/app/8930/>
- [21] <http://www.wesnoth.org/>
- [22] <http://play0ad.com/>

Author

Robin Muilwijk is Advisor Internet and e-Government. He also serves as a community moderator for Opensource.com and as ambassador for The Open Organization. Robin is Chair of the eZ Community Board, and Community Manager at eZ Systems. Follow Robin on Twitter: [@i_robin](https://twitter.com/i_robin)



Open source diversity efforts gain momentum in 2016

• BY NITHYA RUFF

IF SOFTWARE is pervasive, shouldn't the people building it be from everywhere and represent different voices? The broadly accepted answer is yes, that we need a diverse set of developers and technologists to build the new digital world. Further, when you look at communities that thrive, they are those that evolve and grow and bring in new voices and perspectives. Because much of the software innovation happening today involves open source software, the open source community can be an entry point for new people in technology roles. This means that the open source community must evolve to stay relevant. There has never been a better time for the open source community to welcome new community members from under-represented groups than now, and the community is rising to the challenge. Efforts to increase diversity in open source are showing results, so let's look at a few examples:

1. Foundations and organizations are increasing outreach efforts.

Over the past few years, new foundations have been created to support open source projects. For example, multiple projects fall under the umbrellas of big foundations, such as the Linux Foundation and OpenStack Foundation. Besides being a home for projects, foundations are embracing their roles in curating and sharing best practices, creating on-ramps, and supporting new people in project communities.

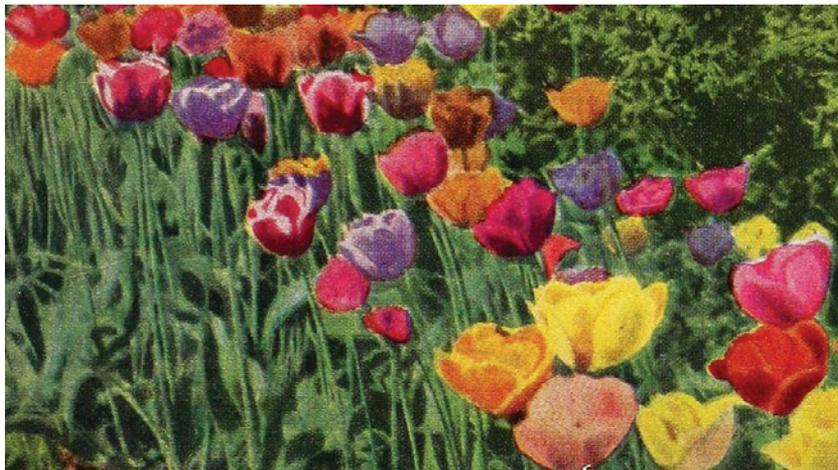


Image by: Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0

The Linux Foundation [1] and OpenStack Foundation [2] provide scholarships, travel assistance, training, mentorships, childcare, affinity groups, and more as part of their events and services. (I'm involved in the Linux Foundation-sponsored Women in Open Source events and the Women of OpenStack [WOO] group.) Since the WOO group started in 2014, more women have been attending and speaking at OpenStack-related events and contributing to OpenStack projects. More than 11% of attendees at both LinuxCon North America and OpenStack Summit in Austin [3] in 2016 were women.

The Linux Foundation, in partnership with the National Center for Women & Information Technology (NCWIT [4]), is developing inclusive speaker orientation course for events to ensure that all speakers go through training in what it means to *be inclusive*.

Many tech organizations are addressing the role their company cultures and policies play in increasing diversity. For example, Google launched its Summer of Code program in 2005, and Red Hat launched its Women in Open Source award [5] in 2015. By 2016, many organizations and events were regularly sponsoring networking opportunities for women, such as the LinuxCon North America Women in Open Source Lunch, sponsored by Intel; and the Women's Leadership Community Luncheon at Red Hat's annual summit [6].

2. Mentorships, scholarship programs, and training are scaling.

Software Freedom Conservancy's Outreachy [7], which was founded in 2010, provides internships to women and members of underrepresented groups through mentors and work experience on a specific project. Contributions from Outreachy interns is impressive: The Outreachy organization ranked #9 for kernel contributions in Linux 4.4 and #6 for Linux 4.6 releases. Read the 2016 Kernel Internship Report (PDF [8]) for more detail.

Google Summer of Code [9] (GSoC) is another successful initiative that has been matching students to open source projects and volunteer mentors for more than a decade. More than 1,200 students and 178 organizations [10] participated in GSoC 2016. Hopefully the success of these programs will help inspire more mentorship programs in 2017.

In 2016, annual Grace Hopper Celebration of Women in Computing [11] (GHC) attracted around 15,000 from more than 80 countries. The GHC annual Open Source Day is an all-day hackathon that brings attendees together to network, contribute to good causes, and build their GitHub profiles in a safe environment.

3. Conferences are improving diversity among speakers and attendees.

Conferences such as OSCON [12], LinuxCon [13], PyCon [14], SCaLE [15], and OpenStack Summit [16] play a big role in showcasing and supporting diversity. The impact of seeing women and underrepresented minorities on stage and hearing new perspectives is powerful in changing our perception of who is part of the open source community.

O'Reilly's OSCON is showing noticeable improvement in the diversity of its speakers and talks. I spoke with Rachel Roumeliotis, Strategic Content Director and OSCON Chair for O'Reilly about how OSCON raises the bar every year to showcase amazing new voices. She says organizers actively seek out new voices and perspectives, and they want to reflect the changing community and perhaps project what it could look like. (Read Opensource.com's OSCON article collection [17] for examples.)

Conferences are also investing in the next generation of programmers and contributors through dedicated co-located events. One great example of this is SCALE: The Next Generation [18], a track hosted at the Southern California Linux Expo, an annual community event. I love how SCaLE helps teach kids about programming and technology. Young community members, such as Keila Banks [19], Justin King, and Schuyler St. Leger [20], have spoken at this event.

Often the kids at conferences have relatives working in technology who are highly involved in encouraging and championing young people. Keila, for example, is the daughter of Phillip Banks, one of the SCaLE event orga-

nizers. She got her start speaking at SCALE when she was 11, and she gave a keynote at OSCON [21] in 2014, at the ripe old age of 13. When I talked to Keila, she found programming to be as natural as other subjects at school, and she loves the Python language, including its community. Young people are learning open source technical skills from programming, animation, video editing, digital drawing, and picture editing, and conferences are rapidly embracing and catering to this new generation of community members.

4. The data confirms progress.

Without data, determining whether we are making progress is difficult. We need to measure the percent of contributions from women and members of underrepresented groups, learn about projects successfully attracting and retaining a diverse group of contributors, and then share what we learn across projects. Spain-based Bitergia's Chief Data Officer Daniel Izquierdo [22] took the initiative to do this for OpenStack and for the Linux kernel and shared the results in 2016. (Read Rikki Endsley's recent interview with Daniel, "Analyzing gender diversity in the OpenStack community" [23].) Bitergia plans to do gather diversity metrics for Apache Software Foundation projects in the near future.

The data Bitergia has been collecting on the diversity of contributions and community growth over time shows trending in the right direction. We still need a new way to measure the overall health of an open source project, which means that, in addition to measuring the number of contributors, downloads, and issues resolved, we should track how diverse communities are and how welcoming they are to new contributors. These measurements could help us recognize projects that have great on-boarding and retention practices and that have built diverse and vibrant communities. Learn more about Bitergia's work in gender diversity metrics on their blog [24].

5. The definition of 'contributions' is expanding, and on-boarding is improving.

Historically non-code-related open source contributions haven't been recognized as *contributions*. Measuring code contributions and code reviews has been easier than measuring other types of contributions, such as documentation, marketing, legal representation, and graphic design. Increasingly open source projects and organizations are recognizing, encouraging, valuing, and supporting contributions in all areas of the open source ecosystem, and they are making it easier for new contributors to join communities.

GitHub has become the tool of choice for internal development in companies changing their culture to be collaborative and agile. I asked Brandon Keepers [25], head of open source at GitHub, about their initiatives. He explained that

GitHub launched a survey [26] to help them understand their changing community and its needs. His team is also working to provide best practice frameworks for project leaders to help them run better projects, which includes sample codes of conduct and README templates.

6. Allies are taking more active roles in improving diversity.

From its beginning, the Ada Initiative [27] founders recognized the roles male allies play in increasing diversity in technology. The organization was formed in 2011 by Valerie Aurora and Mary Gardiner and worked on making technical conferences harassment-free, providing training on how to overcome impostor syndrome, and provided practical skills training for male allies. Many men want to support diversity efforts and create a more inclusive culture, but may not know how to help. The Ada Initiative and NCWIT skills training [28] are examples of practical approaches to showing male allies how they can support gender equality and build inclusive cultures.

What will 2017 bring?

As 2016 comes to a close, we're seeing real progress in how we welcome new people and encourage them to stay in the open source community, but there is still much to be done. What plans does your project, organization, or community have for increasing diversity in open source in the new year? Send your story ideas to open@opensource.com.

Resources

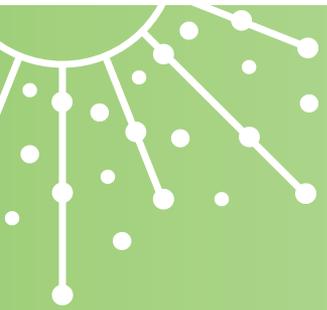
- [1] <https://www.linuxfoundation.org/>
- [2] <https://www.openstack.org/foundation>
- [3] <https://opensource.com/business/16/5/look-back-austin-openstack-summit>
- [4] <https://www.ncwit.org/>
- [5] <https://www.redhat.com/en/about/women-in-open-source>
- [6] <https://www.redhat.com/en/summit/2016/agenda>
- [7] <https://www.gnome.org/outreachy/>
- [8] http://events.linuxfoundation.org/sites/events/files/slides/outreachy_slides.pdf
- [9] <https://developers.google.com/open-source/gsoc/>

- [10] <https://developers.google.com/open-source/gsoc/2016/organizations>
- [11] <http://ghc.anitaborg.org/>
- [12] <http://conferences.oreilly.com/oscon/oscon-tx>
- [13] <http://events.linuxfoundation.org/events/linuxcon-north-america>
- [14] <http://www.pycon.org/>
- [15] <https://www.socallinuxexpo.org/scale/15x/>
- [16] <https://www.openstack.org/summit/>
- [17] <https://opensource.com/tags/oscon>
- [18] <https://www.socallinuxexpo.org/scale/14x/scale-next-generation>
- [19] <https://opensource.com/life/16/4/5-open-source-programs-automated-teens-toolbox>
- [20] <https://opensource.com/life/16/1/scale14x-interview-schuyler-st-leger>
- [21] <https://www.youtube.com/watch?v=xkTcSoQ-q5Q>
- [22] <https://bitergia.com/about/team/>
- [23] <https://opensource.com/business/16/4/openstack-summit-interview-daniel-izquierdo-bitergia>
- [24] <https://blog.bitergia.com/>
- [25] <https://opensource.com/life/15/10/ato-interview-brandon-keepers-github>
- [26] <https://github.com/github/open-source-survey>
- [27] <https://opensource.com/business/15/8/ada-initiative-legacy>
- [28] <https://www.ncwit.org/resources/10-actionable-ways-actually-increase-diversity-tech>

Author

Nithya A. Ruff is the Director of Western Digital's Open Source Office. She is the founding President of Western Digital's Women's Innovation Network (WIN), which is dedicated to the development of women's highest potential in the work place. Nithya graduated with an MS in Computer Science from NDSU and an MBA from the University of Rochester, Simon Business School. She lives in Silicon Valley and is a proud mother of two daughters. Follow Nithya on Twitter at: [@nithyaruff](https://twitter.com/nithyaruff) (The views expressed are her own and do not represent Western Digital.)





Most Popular

Top 10 open source projects of 2016BY JEN WIKE HUGER

WE CONTINUE to be impressed with the wonderful open source projects that emerge, grow, change, and evolve every year. Picking 10 to include in our annual list of top projects is no small feat, and certainly no list this short can include every deserving project.

To choose our 10, we looked back at popular open source projects our writers covered in 2016, and collected suggestions from our Community Moderators. After a round of nominations and voting by our moderators, our editorial team narrowed down the final list.

So here they are, our top 10 open source projects of 2016:

Atom

Atom [1] is a hackable text editor from GitHub. Jono Bacon wrote [2] about its “simple core” earlier this year, exclaiming approval for open source projects that give users options.

“[Atom] delivers the majority of the core features and settings that most users likely will want, but is missing many of the more advanced or specific features some users may want. ... Atom provides a powerful framework that allows pretty much any part of Atom to be changed and expanded.”

To get started contributing, read the guide [3]. To

connect with other users and the community, find Atom on GitHub [4], Discuss [5], and Slack [6].

Atom is MIT [7] licensed and the source code [8] is hosted on GitHub.

Eclipse Che

Eclipse Che [9] is a next-generation online integrated development environment (IDE) and developer workspace. Joshua Allen Holm brought us a review [10] of Eclipse Che in November 2016, which provided a look at the developer community behind the project, its innovative use of container technology, and popular languages it supports out of the box.

“The ready-to-go bundled stacks included with Eclipse Che cover most of the modern popular languages. There are stacks for C++, Java, Go, PHP, Python, .NET, Node.js, Ruby on Rails, and Android development. A Stack Library provides even more options and if that is not enough,

there is the option to create a custom stack that can provide specialized environments.”

You can test out Eclipse Che in an online hosted account [11], through a local installation [12], or in your preferred cloud provider [13]. The source code [14] can be found on GitHub under an Eclipse Public License [15].

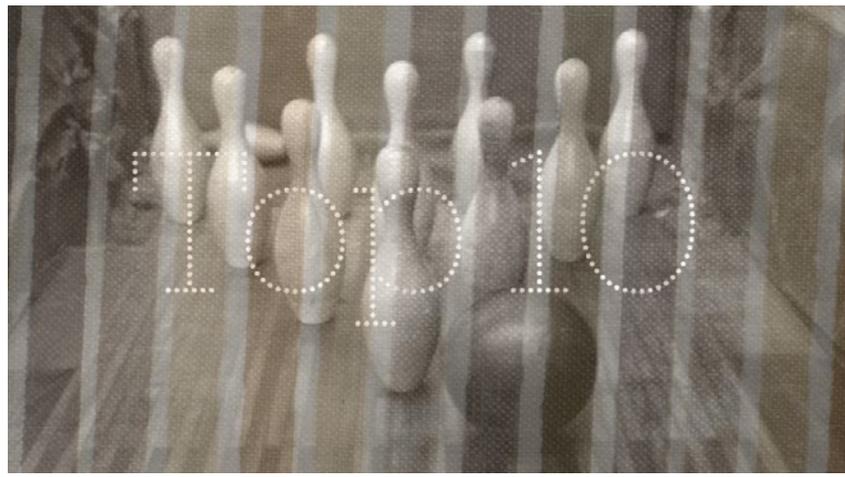


Image by: George Eastman House and Internet Archive Book Images. Modified by Opensource.com. CC BY-SA 4.0



Most Popular

FreeCAD

FreeCAD [16] is written in Python and one of the many computer-aided design—or computer-aided drafting—tools available to create design specifications for real-world objects. Jason Baker wrote about FreeCAD in “3 open source alternatives to AutoCAD” [17].

“FreeCAD can import and export from a variety of common formats for 3D objects, and its modular architecture makes it easy to extend the basic functionality with various plugins. The program has many built-in interface options, from a sketcher to renderer to even a robot simulation ability.”

FreeCAD is LGPL [18] licensed and the source code [19] is hosted on GitHub.

GnuCash

GnuCash [20] is a cross-platform open source desktop solution for managing your personal and small business accounts. Jason Baker included GnuCash in our roundup [21] of the open source alternatives to Mint and Quicken for personal finance.

GnuCash “features multi-entry bookkeeping, can import from a wide range of formats, handles multiple currencies, helps you create budgets, prints checks, creates custom reports in Scheme, and can import from online banks and pull stock quotes for you directly.”

You can find GnuCash’s source code [22] on GitHub under a GPL version 2 or 3 license [23].

An honorable mention goes to GnuCash alternative KMyMoney [24], which also received a nomination for our list, and is another great option for keeping your finances in Linux.

Kodi

Kodi [25] is an open source media center solution, formerly known as XBMC, which works on a variety of devices as a do-it-yourselfer’s tool to building a set-top box for playing movies,

TV, music, and more. It is heavily customizable, and supports numerous skins, plugins, and a variety of remote control devices (including its own custom Android remote for your phone).

Although we didn’t cover Kodi in-depth this year, it kept popping up in articles on building a home Linux music server [26], media management tools [27], and even a previous poll on favorite open source video players [28].

The source code [29] to Kodi can be found on GitHub under a GPLv2 [30] license.

MyCollab

MyCollab [31] is a suite of tools for customer relationship management, document management, and project management. Community Moderator Robin Muilwijk covered the details of the project management tool MyCollab-Project in his roundup of “Top 11 project management tools for 2016” [32].

“MyCollab-Project includes many features, like a Gantt chart and milestones, time tracking, and issue management. It also supports agile development models with its Kanban board. MyCollab-Project comes in three editions, of which the community edition [33] is the free and open source option.”

Installing MyCollab requires a Java runtime and MySQL stack. Visit the MyCollab site [34] to learn how to contribute to the project.

MyCollab is AGPLv3 licensed and the source code is hosted on GitHub.

OpenAPS

OpenAPS [35] is another project that our moderators found interesting in 2016, but also one that we have yet to cover in depth. OpenAPS, the Open Artificial Pancreas System project, is an open source project devoted to improving the lives of people with Type 1 diabetes.

The project includes “a safety-focused reference design [36], a toolset [37], and an open source reference implementation” [38] designed for device manufacturers or any individual to be able to build their own artificial pancreas device to be able to safely regulate blood glucose levels overnight by adjusting insulin levels. Although potential users should examine the project carefully and discuss it with their health-care provider before trying to build or use the system themselves, the project founders hope opening up technology will accelerate the research and development pace across the medical devices industry to discover solutions and bring them to market even faster.

OpenHAB

OpenHAB [39] is a home automation platform with a plug-gable architecture. Community Moderator D Ruth Bavousett wrote about OpenHAB [40] after buying a home this year and trying it out.

“One of the interesting modules I found was the Bluetooth binding; it can watch for the presence of specific Bluetooth-enabled devices (your smartphone, and those of your children, for instance) and take action when that device arrives or leaves—lock or unlock doors, turn on lights, adjust your thermostat, turn off security modes, and so on.”

Check out the full list of binding and bundles [41] that provide integration and communication with social networks, instant messaging, cloud IoT platforms, and more.

OpenHAB is EPL licensed and the source code is hosted on GitHub.

OpenToonz

OpenToonz [42] is production software for 2D animation. Community Moderator Joshua Allen Holm reported [43] on its open source release in March 2016, and it has been mentioned in other animation-related articles on Open-source.com, but we haven’t covered it in depth. Stay tuned for that.

In the meantime, we can tell you that there are a number of features unique to OpenToonz, including GTS, which is a spanning tool developed by Studio Ghibli, and a plug-in effect SDK [44] for image processing.

To discuss development and video research topics, check out the forum [45] on GitHub. OpenToonz source code is hosted on GitHub and the project is licensed under a modified BSD license.

Roundcube

Roundcube [46] is a modern, browser-based email client that provides much—if not all—of the functionality email users may be used to with a desktop client. Featuring support for more than 70 languages, integrated spell-checking, a drag-and-drop interface, a feature-rich address book, HTML email composition, multiple search features, PGP encryption support, threading, and more, Roundcube can work as a drop-in replacement email client for many users.

Roundcube was included along with four other solutions in our roundup of open source alternatives to Gmail [47].

You can find the source code [48] to Roundcube on GitHub under a GPLv3 license. In addition to downloading and installing the project directly, you can also find it inside many complete email server packages, including Kolab Groupware [49], iRedMail [50], Mail-in-a-Box [51], and mailcow [52].

That’s it for our list. What was your favorite open source project in 2016?

Resources

- [1] <https://atom.io/>
- [2] <https://opensource.com/life/16/2/culture-pluggable-open-source>
- [3] <https://github.com/atom/atom/blob/master/CONTRIBUTING.md>
- [4] <https://github.com/atom/atom>
- [5] <http://discuss.atom.io/>
- [6] <http://atom-slack.herokuapp.com/>
- [7] <https://raw.githubusercontent.com/atom/atom/master/LICENSE.md>
- [8] <https://github.com/atom/atom>
- [9] <http://www.eclipse.org/che/>
- [10] <https://opensource.com/life/16/11/introduction-eclipse-che>
- [11] <https://www.eclipse.org/che/getting-started/cloud/>
- [12] <https://www.eclipse.org/che/getting-started/download/>
- [13] <https://bitnami.com/stack/eclipse-che>
- [14] <https://github.com/eclipse/che/>
- [15] <https://github.com/eclipse/che/blob/master/LICENSE>
- [16] <http://www.freecadweb.org/>
- [17] <https://opensource.com/alternatives/autocad>
- [18] <https://github.com/FreeCAD/FreeCAD/blob/master/COPYING>
- [19] <https://github.com/FreeCAD/FreeCAD>
- [20] <https://www.gnucash.org/>
- [21] <https://opensource.com/life/16/1/3-open-source-personal-finance-tools-linux>
- [22] <https://github.com/Gnucash/>
- [23] <https://github.com/Gnucash/gnucash/blob/master/LICENSE>
- [24] <https://kmy.money.org/>
- [25] <https://kodi.tv/>
- [26] <https://opensource.com/life/16/1/how-set-linux-based-music-server-home>
- [27] <https://opensource.com/life/16/6/tinymediamanager-catalogs-your-movie-and-tv-files>
- [28] <https://opensource.com/life/15/11/favorite-open-source-video-player>

Publisher's picks: Hot 2016 open source books

BOOK PUBLISHERS share their picks for must-read 2016 open source-related book releases.

No Starch Press

(Contributed by Anna Morrow)

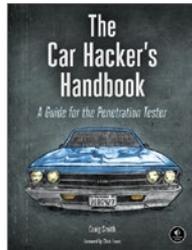
***The Car Hacker's Handbook* [1]**

By Craig Smith

304 Pages

Published: March 2016

With its focus on low-cost, open source hacking tools, *The Car Hacker's Handbook* will give you a deeper understanding of the computer systems and embedded software in modern vehicles. The book begins by examining vulnerabilities and providing detailed explanations of communications over the CAN bus and between devices and systems. You'll learn how to build an accurate threat model for your vehicle, reverse engineer the CAN bus to fake engine signals, exploit vulnerabilities, build physical and virtual test benches, and much more. If you're curious about automotive security, *The Car Hacker's Handbook* is for you.



***Electronics for Kids* [2]**

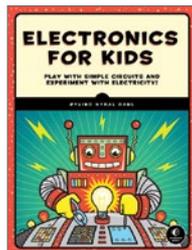
By Øyvind Nydal Dahl

328 Pages

Published: July 2016

Electronics for Kids is perfect for feeding kids' natural curiosity about electronics through hands-on projects. Kids (and the adults in their lives!) will build projects like an electronic coin tosser, an electromagnet, an electric motor, an intruder alarm, a musical instrument, a touch sensor LED circuit, and even a lemon-powered LED light. Along the way, they'll learn how current, voltage, and circuits work. With clear explanations and fun projects, this book will have kids building their own circuits in no time.

To read more from Øyvind Nydal Dahl, author of *Electronics for Kids*, visit his [OpenSource.com](https://www.opensource.com) author page [3].



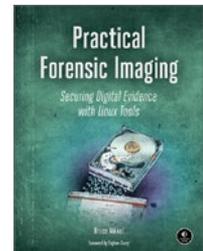
***Practical Forensic Imaging* [4]**

By Bruce Nikkel

320 Pages

Published: September 2016

When forensic investigations involve digital activity, the proper handling of media evidence is critical. *Practical Forensic Imaging* takes a detailed look at how to use open source command-line tools to secure and manage digital evidence. Author Bruce Nikkel walks you through the entire forensic acquisition process and covers practical scenarios and situations related to the imaging of storage media. *Practical Forensic Imaging* is an invaluable resource for experienced digital forensic investigators wanting to advance their Linux skills and experienced Linux administrators wanting to learn digital forensics.



***Scratch Programming Playground* [5]**

By Al Sweigart

288 Pages

Published: September 2016

Scratch is the standard language for teaching kids to program, with more than 14-million users worldwide. In *Scratch Programming Playground*, kids can learn to program by making cool games. Each game includes easy-to-follow instructions, review questions, and creative coding challenges that allow kids to make the games their own. Kids will make games like Maze Runner, Snake, a Fruit Ninja clone, a remake of Breakout, and even a game inspired by Super Mario Bros. From the author of fan favorite *Automate the Boring Stuff with Python*, *Scratch Programming Playground* proves that learning to program isn't dreary when you make a game of it. To read more from Al Sweigart, author of *Scratch Programming Playground*, visit his [OpenSource.com](https://www.opensource.com) author page [6].

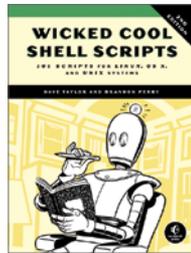


Wicked Cool Shell Scripts, 2nd Edition [7]

By Dave Taylor and Brandon Perry

392 Pages

Published: October 2016



An update to the beloved first edition, the second edition of *Wicked Cool Shell Scripts* offers a collection of useful, customizable, and fun shell scripts for solving common problems and personalizing your computing environment. This edition features 23 brand-new scripts, such as a ZIP code lookup tool, a bitcoin address information retriever, image processing and editing tools, and classic games like hangman. Whether you want to save time managing your system or just find new ways to goof off, you'll love these wicked cool scripts.

To read more from Dave Taylor, author of *Wicked Cool Shell Scripts, 2nd Edition*, visit his [Opensource.com](#) author page [8].

O'Reilly Media

(Contributed by Susan Conant)

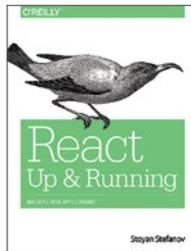
React: Up & Running [9]

Building Web Applications

By Stoyan Stefanov

222 Pages

Published: July 2016



Hit the ground running with React, the open-source technology from Facebook for building rich web applications fast. With this practical guide, Yahoo! web developer Stoyan Stefanov teaches you how to build components—React's basic building blocks—and organize them into maintainable, large-scale apps. If you're familiar with basic JavaScript syntax, you're ready to get started.

Once you understand how React works, you'll build a complete custom Whinepad app that helps users rate wines and keep notes. You'll quickly learn why some developers consider React the key to the web app development puzzle.

- Set up React and write your first “Hello world” web app
- Create and use custom React components alongside generic DOM components
- Build a data table component that lets you edit, sort, search, and export its contents
- Use the JSX syntax extension as an alternative to function calls
- Set up a lean, low-level build process that helps you focus on React
- Build a complete custom app that lets you store data on the client

- Use ESLint, Flow, and Jest tools to check and test your code as your app evolves
- Manage communication between components with Flux

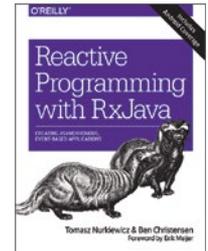
Reactive Programming with RxJava [10]

Creating Asynchronous, Event-Based Applications

By Tomasz Nurkiewicz, Ben Christensen

372 Pages

Published: October 2016



In today's app-driven era, when programs are asynchronous and responsiveness is so vital, reactive programming can help you write code that's more reliable, easier to scale, and better-performing. With this practical book, Java developers will first learn how to view problems in the reactive way, and then build programs that leverage the best features of this exciting new programming paradigm.

Authors Tomasz Nurkiewicz and Ben Christensen include concrete examples that use the RxJava library to solve real-world performance issues on Android devices as well as the server. You'll learn how RxJava leverages parallelism and concurrency to help you solve today's problems. This book also provides a preview of the upcoming 2.0 release.

- Write programs that react to multiple asynchronous sources of input without descending into “callback hell”
- Get to that aha! moment when you understand how to solve problems in the reactive way
- Cope with Observables that produce data too quickly to be consumed
- Explore strategies to debug and to test programs written in the reactive style
- Efficiently exploit parallelism and concurrency in your programs
- Learn about the transition to RxJava version 2

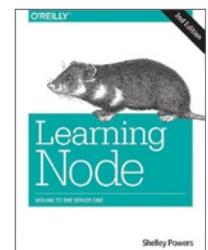
Learning Node, 2nd Edition [11]

Moving to the Server-Side

By Shelley Powers

288 Pages

Published: May 2016



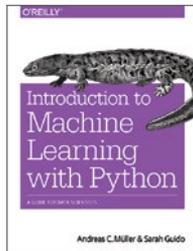
Take your web development skills from browser to server with Node—and learn how to write fast, highly scalable network applications on this JavaScript-based platform. Updated for the latest Node Long Term Support (LTS) and Node Current (6.0) releases, this hands-on edition helps you master Node's core fundamentals and gain experience with several built-in and contributed modules.

Get up to speed on Node's event-driven, asynchronous I/O model for developing data-intensive applications that are frequently accessed but computationally simple. If you're comfortable working with JavaScript, this book provides many programming and deployment examples to help you take advantage of server-side development with Node.

- Explore the frameworks and functionality for full-stack Node development
- Dive into Node's module system and package management support
- Test your application or module code on the fly with Node's REPL console
- Use core Node modules to build web applications and an HTTP server
- Learn Node's support for networks, security, and sockets
- Access operating system functionality with child processes
- Learn tools and techniques for Node development and production
- Use Node in microcontrollers, microcomputers, and the Internet of Things

Introduction to Machine Learning with Python [12]

A Guide for Data Scientists
By Andreas C. Müller, Sarah Guido
392 Pages
Published: September 2016



Machine learning has become an integral part of many commercial applications and research projects, but this field is not exclusive to large companies with extensive research teams. If you use Python, even as a beginner, this book will teach you practical ways to build your own machine learning solutions. With all the data available today, machine learning applications are limited only by your imagination.

You'll learn the steps necessary to create a successful machine-learning application with Python and the scikit-learn library. Authors Andreas Müller and Sarah Guido focus on the practical aspects of using machine learning algorithms, rather than the math behind them. Familiarity with the NumPy and matplotlib libraries will help you get even more from this book.

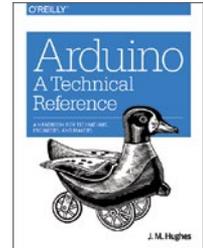
With this book, you'll learn:

- Fundamental concepts and applications of machine learning
- Advantages and shortcomings of widely used machine learning algorithms
- How to represent data processed by machine learning, including which data aspects to focus on

- Advanced methods for model evaluation and parameter tuning
- The concept of pipelines for chaining models and encapsulating your workflow
- Methods for working with text data, including text-specific processing techniques
- Suggestions for improving your machine learning and data science skills

Arduino: A Technical Reference [13]

A Handbook for Technicians, Engineers, and Makers
By J. M. Hughes
638 Pages
Published: May 2016



Rather than yet another project-based workbook, *Arduino: A Technical Reference* is a reference and handbook that thoroughly describes the electrical and performance aspects of an Arduino board and its software. This book brings together in one place all the information you need to get something done with Arduino. It will save you from endless web searches and digging through translations of datasheets or notes in project-based texts to find the information that corresponds to your own particular setup and question.

Reference features include pinout diagrams, a discussion of the AVR microcontrollers used with Arduino boards, a look under the hood at the firmware and run-time libraries that make the Arduino unique, and extensive coverage of the various shields and add-on sensors that can be used with an Arduino. One chapter is devoted to creating a new shield from scratch. The book wraps up with detailed descriptions of three different projects: a programmable signal generator, a "smart" thermostat, and a programmable launch sequencer for model rockets. Each project highlights one or more topics that can be applied to other applications.

Packt

(Contributed by Richard Gall)

C#6 and .NET Core [14]

By Mark J. Price
550 Pages
Published: March 2016



There's a lot of talk about open source going mainstream—Microsoft's launch of .NET Core in June 2016 confirmed that fact. It redefined the way the development world saw one of the most established tech giants, and appeared to be an admission that the world hadn't been

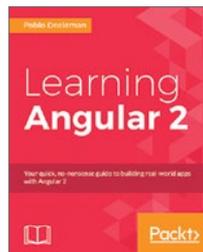
going in the direction they thought it would decades earlier. This book provides developers with a comprehensive look at Microsoft’s powerful language and impressive open-source framework, designed to give readers fluency and confidence to build cross-platform applications. At 550 pages, there’s just about everything you’d need to know, which means it acts both as a great tutorial and a reliable resource. Open source might mean rapid change and constant iteration, but this book provides readers with a stable and trusted source of knowledge.

Learning Angular 2 [15]

By Pablo Deeleman

352 Pages

Published: May 2016



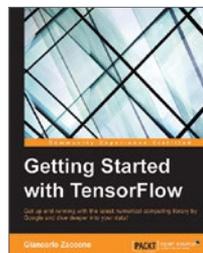
Angular 2 was easily the most anticipated software release of 2016. Google kept the world waiting—it wasn’t until May that we properly got to see what the new framework actually looked like. That was when Packt released Learning Angular 2, a no nonsense and fast paced guide to the new features of the framework. Showing readers how to build Angular 2 components, demonstrating how to get to grips with the TypeScript syntax and working with directives and services, the book brings together everything the curious and ambitious web developer needs to get started with this cutting-edge framework.

Getting Started with TensorFlow [16]

By Giancarlo Zaccone

180 Pages

Published: July 2016



TensorFlow was the surprise hit of 2016. But we probably shouldn’t have been so surprised. With machine learning becoming one of the biggest trends across the technology world, the arrival of a tool like TensorFlow, which is so easy—and enjoyable—to use, was inevitable. *Getting Started with TensorFlow* does exactly what it says on the cover—and it’s created specifically for those people that want to get up and running with machine learning as quickly as possible. From basic mathematics to neural networks and deep learning, this book proves that using software in a smart and impactful way doesn’t have to be a steep learning curve and doesn’t require you to wade through theory.

Smart Internet of Things Projects [17]

By Agus Kurniawan

258 Pages

Published: September 2016



The Internet of Things has been a buzzword for a little while now, but this year we’ve started to see it become more of a reality. And the point, really, is simple—you’ve just got to build it yourself. The book includes fun real-world projects, from building an autonomous remote control car to speech technology, and is a great way to help you get creative.

ReactJS Blueprints [18]

By Sven A. Robbestad

422 Pages

Published: July 2016



Angular 2 might have been hotly anticipated in 2016, but React has slowly been taking a hold of the JavaScript imagination for a couple of years now—in 2016, Facebook’s impressive library went mainstream. It’s also a tool that highlights the nature of the web today: dynamic, fast, and lightweight in our data intensive age. ReactJS Blueprints takes you straight into React, showing readers how to build a complete application with the library. It doesn’t just teach—it demonstrates how React works by showing you how to use it yourself.

Resources

- [1] <https://www.nostarch.com/carhacking>
- [2] <https://www.nostarch.com/electronicforkids>
- [3] <https://opensource.com/users/oyvdahl>
- [4] <https://www.nostarch.com/forensicimaging>
- [5] <https://www.nostarch.com/scratchplayground>
- [6] <https://opensource.com/users/alsweigart>
- [7] <https://www.nostarch.com/wcss2>
- [8] <https://opensource.com/users/davetaylor>
- [9] <http://shop.oreilly.com/product/0636920042266.do>
- [10] <http://shop.oreilly.com/product/0636920042228.do>
- [11] <http://shop.oreilly.com/product/0636920046936.do>
- [12] <http://shop.oreilly.com/product/0636920030515.do>
- [13] <http://shop.oreilly.com/product/0636920037880.do>
- [14] <https://www.packtpub.com/application-development/c-6-and-net-core-10>
- [15] <https://www.packtpub.com/web-development/learning-angular-2>
- [16] <https://www.packtpub.com/big-data-and-business-intelligence/getting-started-tensorflow>
- [17] <https://www.packtpub.com/hardware-and-creative/smart-internet-things-projects>
- [18] <https://www.packtpub.com/web-development/reactjs-blueprints>

8 fun Raspberry Pi projects to try

BY ANDERSON SILVA

FOR MANY OF US 2016 flew by, and we didn't complete all our New Year's resolutions or mark everything off our "2016 To Do" lists. I didn't have nearly enough time to play with the Raspberry Pi this year, and my list of projects I want to do keeps growing. In this article I've rounded up 8 recent Raspberry Pi projects that I haven't made yet, but that made it onto my "2017 To Do" list.

Recent Raspberry Pi projects to try

1. Magic Mirror²

Magic Mirror² was created by Michael Teeuw. This project allows you to convert your hall or bathroom mirror into a personal assistant. Both the Raspberry Pi 2 and 3 are supported by this project, and you can count on a rapidly growing community to help you build your own Magic Mirror². (License: MIT)

- Michael Teeuw's Twitter: [@MichMich](#) [1]
- Magic Mirror² site [2]
- Michael Teeuw, Xony Labs [3]
- Michael Teeuw's Magic Mirror² at GitHub [4]

See video demo at: <https://player.vimeo.com/video/171152845>.



Image by: Internet Archive Book Images. Modified by Opensource.com.

2. Kodi Media Center

Kodi Media Center is an open source media player that runs in several different platforms. Formerly known as XBMC Media Center, it is one of the most popular projects for running on a Raspberry Pi. (License: GPLv2)

- How to install Kodi on Raspberry Pi [5]
- Official Kodi Wiki Raspberry Pi page [6]
- Tuukka's Raspberry Pi Kodi installation tutorial [7]

See video demo at: <https://youtu.be/4oMongjNslg>.

3. Raspberry Pi Weather Station

The Raspberry Pi Weather Station is a project created by Peter Kodermac that provides all the information to build your own weather station. Peter gives you the code and recommends the best type of sensors for this scientific project.

- Peter Kodermac's email: peter@raspberrypiweather.com
- Raspberry Pi Weather Station website [8]
- Raspberry Pi Weather Station GitHub [9]

4. Fedora 25 on a Pi

The Fedora 25 project is an open source operating system known for its simplicity, which makes it an excellent product

for those new to open source development. Fedora 25 [10] was recently released and is the first Fedora release with official support for the Raspberry Pi. Fedora 25 supports the Raspberry Pi B+ version 2 and 3. Non-official spins of the Fedora distribution have been available in the past (i.e., Pidora [11]).

- Fedora Wiki's Raspberry Pi page [12]
- Pidora Fedora remix [13]

(I plan to review Fedora 25 on the Raspberry Pi in an upcoming article on [Opensource.com](http://opensource.com).)

5. Pi Hole server-level ad blocking

Raspberry Pi is useful for creating powerhouse ad blocking on a network. Instead of blocking ads at the browser level and having to manage the different extensions, why not block ads for your entire network? That's what Pi Hole will let you to do. It claims it blocks more than 100,000 ad-serving domains from your network. This may be the first project I try when I have time. (License: GPLv2)

- Pi Hole site [14]
- Pi Hole GitHub [15]

See video demo at: <https://youtu.be/9Eti3xibiho>.

6. RetroPi retro gaming machine

The RetroPi retro gaming machine is an open source project that transforms your Raspberry Pi into a time machine, allowing you to enjoy some of good old games of the '70s, '80s, '90s and 2000s. (License: GPLv3)

- RetroPi Gaming page [16]
- RetroPi setup GitHub [17]
- A list of supported emulators that run under RetroPie [18]

See video demo at: https://youtu.be/xvYX_7iRRiO.

7. Security system

A comprehensive tutorial by developer Max Williams explains how to build a security system with a Raspberry Pi, including a camera, a motion sensor, and notifications via telegram. (License: GPLv2)

- Motion-sensing Raspberry Pi security system page [19]
- Raspberry Pi Security System GitHub [20]

8. RTAndroid (Real-Time Android)

Running Android on a Raspberry Pi is being explored in earnest, most notably at the Aachen University in Germany. The project is called RTAndroid (Real-Time Android) and, although still a work in progress, it is interesting and intriguing.

- Raspberry Pi 3 Android 7.0 with Google Play and Android Root (video) [21]
- RTAndroid Raspberry Pi installation tutorial [22]

See video demo at: <https://youtu.be/Df-bMWONIYk>.

Resources

- [1] <https://twitter.com/MichMich>
- [2] <https://magicmirror.builders/>
- [3] <http://michaelteeuw.nl/tagged/magicmirror>
- [4] <https://github.com/MichMich/MagicMirror>
- [5] http://kodi.wiki/view/HOW-TO:Install_Kodi_on_Raspberry_Pi
- [6] http://kodi.wiki/view/raspberry_Pi
- [7] <http://mymediaexperience.com/raspberry-pi-xbmc-with-raspbmc/>
- [8] <http://www.raspberrypiweather.com/>
- [9] <https://github.com/peterkodermac/Raspberry-Weather>
- [10] <https://fedoramagazine.org/fedora-25-released/>
- [11] <http://pidora.ca/>
- [12] https://fedoraproject.org/wiki/Raspberry_Pi
- [13] <http://pidora.ca/>
- [14] <https://pi-hole.net/>
- [15] <https://github.com/pi-hole/pi-hole>
- [16] <https://retropie.org.uk/>
- [17] <https://github.com/RetroPie/RetroPie-Setup>
- [18] <https://github.com/RetroPie/RetroPie-Setup/wiki/Supported-Systems>
- [19] <https://www.hackster.io/FutureSharks/raspberry-pi-security-system-with-motion-detection-camera-bed172>
- [20] <https://github.com/FutureSharks/rpi-security>
- [21] <https://www.youtube.com/watch?v=Df-bMWONIYk>
- [22] <https://git.embedded.rwth-aachen.de/rtandroid/downloads/raspberry-pi/>

Author

Anderson Silva is the Sr. Manager of Red Hat's IT Platform Operations. He has been a Red Hatter since 2007. He is an RHCE and RHCA and an active Fedora Package maintainer.



5 trends in open source documentation

BY SHAUN MCCANCE

I'VE BEEN doing open source documentation for a long time. Over the past decade, there have been a lot of attitude shifts regarding authoring and publishing. Some of these trends seem to go in cycles, such as the popularity of semantic markup. The latest trends move documentation closer to code, what many have called *docs as code*. Let's look at a few of the larger themes in documentation trends:

1. Git

When I first started doing documentation work for GNOME [1], we wrote our documentation in DocBook [2] and stored it in CVS repositories alongside our code. These days, most GNOME documentation is written in Mallard [3] and stored in a Git [4] repository (after a brief stint with SVN). Although formats and tools have changed, the constant factor is that sources are stored in revision control [5], just like code.

It may seem odd to call this a trend when we've been doing it for so long, but a few things have changed, and some of that revolves around what Git has brought to the table. Git is one of the decentralized version control systems that arrived on the scene over the past decade or so. Some people continue to use decentralized version control systems the same way they used CVS or SVN, but that doesn't expose the real power of these systems. Documentation writers are increasingly proficient using Git for what it is. They're creating development, staging, and production branches, and they're merging disparate contributions. This wasn't as common just a few years ago.

Git is certainly not the only decentralized version control system. There are also Bazaar and Mercurial, to name just two, and you will find writers wielding the same power with those tools as well. But Git has taken the majority of the mind share, thanks in large part to popular Git hosting sites.

This is an area in which open source has led the trend in the overall software documentation industry. A quick glance at technical writing forums will show plenty of people across the industry looking for information on how to effectively transition to Git. In the past, they may have stored their sources on a network drive with no revision control, or they may have used a proprietary management system. Git and



Image by: Internet Archive Book Images. Modified by Opensource.com.

tools like it have drastically changed the way the entire software industry deals with documentation.

2. Lightweight languages

There have always been plenty of choices for documentation source formats. There are semantic XML formats, and SGML formats before that. There are TeX dialects and troff [6] dialects. There are the source formats of word processors, page layout tools, and help authoring tools. There are the internal formats of various wikis and content management systems. There's HTML. And there are a handful of lightweight markup languages that are designed to be easy to type in a text editor.

People are increasingly choosing lightweight markup languages for a number of reasons. They are usually easier to write, at least for simple things. They tend to play better with version control systems, because they're generally line oriented. And they can help lower the barrier to entry for new contributors, although you should be careful not to expect a change in source format alone to drive lots of contributors to your project.

Lightweight markup languages have their downsides, too. The tools for working with them tend to be limited in scope, and don't often provide the kind of data model you need to write other tools. They also don't usually provide as much semantic information. With XML formats, for example, there are a wealth of tools for translation, validation, link checking, status reporting, and various types of testing and data extraction. This kind of tooling isn't currently as extensive for lightweight formats. So although lightweight formats might ease the barrier to entry for new contributors, they can also create new barriers to long-term maintenance. As with all things, there are always trade-offs.

The three most popular lightweight formats [7] right now are Markdown, AsciiDoc, and reStructured Text. Markdown is the simplest, but it doesn't offer much for anything but the most basic documentation needs. It also comes in many different, slightly incompatible flavors, depending on which processing tool you use. AsciiDoc [8] offers more semantics

and more types of elements. It originally focused on being a front-end to DocBook, but it has grown to natively support lots of output formats. reStructuredText came from the Python community, and for a long time its use was largely limited to Python projects. It has grown in popularity lately due to hosting sites, such as Read the Docs [9].

3. Static site generators

Five years ago, the trend was to use wikis and blogging platforms to create documentation sites. They were easy to set up, and giving people accounts to contribute was easy. Particularly brave people would even open their wiki to anonymous contributions. These days, the trend is to keep sources in version control, then build and publish sites with mostly static HTML files.

Generating static sites isn't new. My first job out of college was working on internal tools used at a software company to build and publish static files for tens of thousands of pages of documentation. But static sites have become increasingly popular for projects of all sizes, for a number of reasons.

First, there are increasingly good off-the-shelf static site generators. Tools like Middleman [10] and Jekyll [11] are just as easy to deploy as a wiki or a blog. Unless you have specialized needs, you no longer have to write and maintain your own site-generating tool. Static site generators have become increasingly popular among web developers, and technical writers get to ride that wave.

Another reason static sites are more popular is that source hosting sites are easier to use, and a growing number of technical people use them. One of the draws of a wiki was that somebody could contribute without downloading anything or installing special tools. If your source files are stored in a hosting service like GitHub, anybody with a GitHub account can edit them right in their web browser and ask you to merge their changes.

4. Continuous integration

Continuous integration [12] is the key that ties the previous trends together. You can write your documentation in a simple format, store it in Git and edit it on the web using a Git hosting service, and publish a site from those sources. With continuous integration, you don't even need a human to kick off the publishing process. If you're brave, you can publish automatically after every commit to master, and you'll have a nearly wiki-like experience for writers.

Some projects will be more conservative and only publish from a production branch. But even when publishing from a branch, continuous integration removes tedious human intervention. You can also automatically publish staging sites for development branches.

Continuous integration isn't just about publishing, either. Projects can use it to automatically test their documentation for things like validity and link integrity, or to generate reports on status and coverage.

5. Hosted documentation services

Automatically publishing documentation sites with continuous integration is easier than ever, but now there are hosted services that take care of everything for you. Just pass them a Git repository, and they'll automatically build, publish, and host your documentation. The most well-known example is Read the Docs. Originally coming out of the Python community, its ease of use has made it popular for all sorts of projects.

Whether free hosted documentation sites can be financially viable remains to be seen—to keep sites like that running costs money and people hours. If the sites can't maintain a certain level of quality, people will take their documentation elsewhere. If you benefit from one of these free services, I encourage you to see how you can help financially.

I believe the hosted documentation services trend will continue. Smart people will figure out how to smooth the bumps. I also suspect we'll start seeing paid hosted documentation services for proprietary software. Open source has led the way on documentation technology over the past decade, and it will continue to do so.

Resources

- [1] <https://wiki.gnome.org/DocumentationProject>
- [2] <https://opensource.com/life/15/8/markup-lowdown>
- [3] <http://projectmallard.org/>
- [4] <https://opensource.com/resources/what-is-git>
- [5] <https://opensource.com/life/16/7/systems-administrators-should-use-revision-control>
- [6] http://cgi.csc.liv.ac.uk/~ped/teachadmin/troff_intro.html
- [7] <https://opensource.com/life/16/8/why-i-love-these-markup-languages>
- [8] <https://opensource.com/life/15/10/asciidoc>
- [9] <https://readthedocs.org/>
- [10] <https://middlemanapp.com/>
- [11] <https://jekyllrb.com/>
- [12] <https://opensource.com/business/15/7/six-continuous-integration-tools>

Author

Shaun McCance is an expert on open source documentation. He's contributed to many open source projects, including a decade-long stint as the GNOME documentation team leader. He organizes the Open Help Conference & Sprints, the only conference focused on documentation and support in open source and open communities. Shaun believes in the power of open communities, from making software to opening a neighborhood grocery store. He works as the Community Documentation Liaison at Red Hat, where he has the privilege of helping various open source projects build their documentation community.



11 wonderful wearable open source projects

BY RUTH SUEHLE

LEDS ARE ON everything, and almost everyone you know has at least tried a FitBit or similar device, whereas Google Glass didn't really take off. Despite several years of growth, whether wearable electronics are a fad, or here to keep growing from fun to truly functional is too early to tell. Judge for yourself—read through a few of our favorite wearable projects from 2016. You might even get inspired to start creating.

1. AsteroidOS

Looking for open source in your smartwatch? AsteroidOS [1] is, too. AsteroidOS is a work in progress and currently functions only with the LG G Watch and LG Watch Urbane, Sony Smartwatch 3, and ASUS ZenWatch 2. But if you have one of those watches, you can test AsteroidOS by dual booting your watch.

2. Light show jacket

Music fans should check out this light show jacket that reacts to music [2]. Created as a final project for the maker's Music Technology and Applied Electronics degree, the jacket is based around an Arduino Mega [3] connected directly to a computer and can pulse differently according to pitch or amplitude. See video demo at: <http://bit.ly/2i0qD90>.



Lead image by: The British Library and The U.S. National Archives. Modified by Opensource.com.

3. Stormtrooper voice changer

It's never too early to start planning for Halloween or the next sci-fi con. If you've been thinking it's time for a set of Stormtrooper armor, you might as well sound the part, too. Visit the Sparkfun site to learn how to build a Stormtrooper voice changer [4] to go in your helmet. See video demo at: <http://bit.ly/2ilmVUd>.

4. Cigarette-smoke detecting shirt

In more practical projects, a seventh grader used an Arduino LilyPad [5] to create a cigarette-smoke detecting shirt [6] to encourage his dad to kick the habit. If you smoke while wearing the shirt, it turns on an escalating series of LEDs labeled *stinky breath*, *yellow teeth*, and *lung cancer*. See video demo at: <http://bit.ly/2i0nDcR>.

5. Pokémon Go patches with EL Panels

Many of us spent a chunk of this summer's free time on Pokémon Go. If you're still playing, the folks over at Sparkfun can help you represent your team with these EL panel patches [7]. And of course, if you're not a Pokémon fan, you can simply replace those images with your own favorite designs. If you need a more personal connection to your team, try this similar project from 2014 to make an EL tattoo [8]. See video demo at: <http://bit.ly/2i0modP>.

6. Skintillates

Leveling up on those tattoos, the Hybrid Ecologies Lab [9] is working on Skintillates [10], electronics built into temporary tattoos so that they flex with your skin. They can work with assorted sensors for a variety of applications, such as checking your typing position or posture or controlling devices. Their goal is to replace the Arduino and Makey-Makey [11] they're currently built with and to develop their own open source development board for further flexible wearables. See video demo at: <http://bit.ly/2ilgMqS>.

7. Cosmic Bitcasting

Some of the earliest practical applications for wearables involved environmental sensors. Cosmic Bitcasting [12] detects cosmic radiation with the goal of using the data to further cosmic radiation research. You can see the detector in person at the Alchemists of Art and Science exhibition [13] at the Ars Electronica Center in Linz, Austria.

8. threeASFOUR 3D-printed clothes

Beyond electronics—and sometimes with them—3D-printed clothes and shoes are getting more interesting (albeit perhaps not more comfortable). For their Summer 2016 Interdimensional collection [14], threeASFOUR [15] won the Fashion Design Award by the Cooper-Hewitt Smithsonian Design Museum. This collection worked to incorporate traditional tailoring with 3D-printed surfaces.

9. LED matrix handbag

Finally, if you're going to have tech wearables, you're going to need to accessorize. Try this LED matrix handbag [16]. You can stream a pattern or a message across the LEDs. Not only does the maker show you how to set up the matrix and give you the code, she also shows you how to sew the bag itself. See video demo at: <http://bit.ly/2hwKCsy>.

10. 3D-printed iris-blinking goggles

Then you can top off your outfit with these 3D-printed iris-blinking goggles [17]. See video demo at: <http://bit.ly/2hYjx2g>.

11. NeoPixel tiara

Combine a 3D printer, Gemma microcontroller, sewable NeoPixels, and a soldering iron to make a fun lighted tiara [18]. See video demo at: <http://bit.ly/2ilri1f>.

Conclusion

If you want to get started building your own wearables, the LilyPad and FLORA [19], both featured in products above, are good places to start. There are plenty of resources, both for buying supplies and project ideas, available across the web. To get started with 3D-printed wearables, browse the fashion section of Thingiverse [20].

Resources

1. <https://asteroidos.org/>
2. <http://www.instructables.com/id/Light-Show-Jacket-That-React-to-Music/>
3. <https://www.arduino.cc/en/Main/arduinoBoardMega>
4. <https://learn.sparkfun.com/tutorials/vox-imperium-storm-trooper-voice-changer>
5. <https://www.arduino.cc/en/Main/ArduinoBoardLilyPad>
6. <https://blog.arduino.cc/2016/07/27/kick-the-habit-with-a-cigarette-smoke-detecting-shirt/>
7. <https://learn.sparkfun.com/tutorials/pokmon-go-patches-with-el-panels>
8. <https://www.sparkfun.com/news/1394>
9. <http://www.hybrid-ecologies.org>
10. <http://www.hybrid-ecologies.org/projects/12-skintillates>
11. <http://www.makeymakey.com/>
12. <http://afroditipsarra.com/>
13. <http://www.aec.at/radicalatoms/en/cosmic-bitcasting/>
14. <https://vimeo.com/143647429>
15. <http://www.threeasfour.com/>
16. <http://www.geekmomprojects.com/led-matrix-handbag-2-0-how-to/>
17. <http://www.thingiverse.com/thing:1833286>
18. <http://www.thingiverse.com/thing:237034>
19. <https://www.adafruit.com/flora>
20. <http://www.thingiverse.com/explore/newest/fashion>

Author

Ruth Suehle is the community leadership manager for Red Hat's Open Source and Standards team. She's co-author of *Raspberry Pi Hacks* (O'Reilly, December 2013) and a senior editor at GeekMom, a site for those who find their joy in both geekery and parenting. She's a maker at heart who is often behind a sewing machine creating costumes, rolling fondant for an excessively large cake, or looking for the next great DIY project. You can find her on Twitter: [@suehle](https://twitter.com/suehle).



Top open source creative tools in 2016

BY MÁIRÍN DUFFY

A FEW YEARS AGO, I gave a lightning talk at Red Hat Summit that took attendees on a tour of the 2012 open source creative tools [1] landscape. Open source tools have evolved a lot in the past few years, so let's take a tour of 2016 landscape. (See the online version of this article for additional images and video links at: <http://red.ht/2ihvIvJ>)

Core applications

These six applications are the juggernauts of open source design tools. They are well-established, mature projects with full feature sets, stable releases, and active development communities. All six applications are cross-platform; each is available on Linux, OS X, and Windows, although in some cases the Linux versions are the most quickly updated. These applications are so widely known, I've also included highlights of the latest features available that you may have missed if you don't closely follow their development.

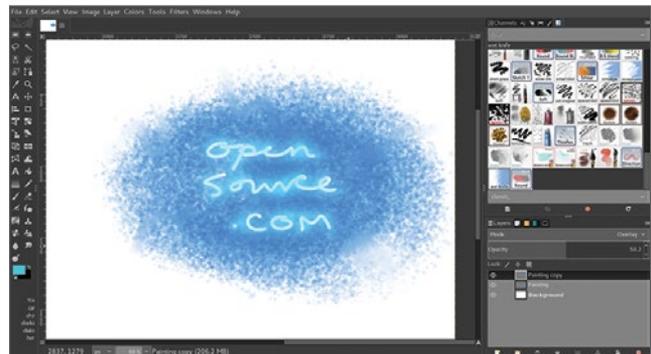
If you'd like to follow new developments more closely, and perhaps even help out by testing the latest development versions of the first four of these applications—GIMP, Inkscape, Scribus, and MyPaint—you can install them easily on Linux using Flatpak [2]. Nightly builds of each of these applications are available via Flatpak by following the instructions [3] for *Nightly Graphics Apps*. One thing to note: If you'd like to install brushes or other extensions to each Flatpak version of the app, the directory to drop the extensions in will be under the directory corresponding to the application inside the `~/ .var / app` directory.

GIMP

GIMP [4] celebrated its 20th anniversary in 2015 [5], making it one of the oldest open source creative applications out there. GIMP is a solid program for photo manipulation, basic graphic creation, and illustration. You can start using GIMP by trying simple tasks, such as cropping and resizing images, and over time work into a deep set of functionality. Available for Linux, Mac OS X, and Windows, GIMP is cross-platform and can open and export to a wide breadth of file formats, including those popularized by its proprietary analogue, Photoshop.



The GIMP team is currently working toward the 2.10 release; 2.8.18 [6] is the latest stable version. More exciting is the unstable version, 2.9.4 [7], with a revamped user interface featuring space-saving symbolic icons and dark themes, improved color management, more GEGL-based filters with split-preview, MyPaint brush support (shown in screenshot below), symmetrical drawing, and command-line batch processing. For more details, check out the full release notes [8].

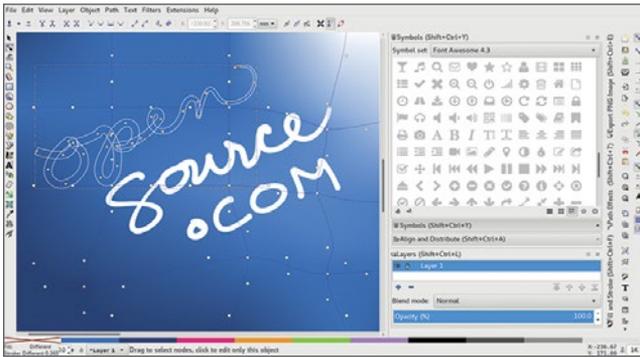


Inkscape

Inkscape [9] is a richly featured vector-based graphic design workspace. Use it to create simple graphics, diagrams, layouts, or icon art.

The latest stable version is 0.91 [10]; similarly to GIMP, more excitement can be found in a pre-release version, 0.92pre3, which was released November 2016. The premiere feature of the latest pre-release is the gradient mesh feature [11] (demonstrated in screenshot below); new features introduced in the 0.91 release include power stroke [12] for fully configurable calligraphic strokes (the “open” in “opensource.com” in the screenshot below uses power-stroke), the on-canvas measure tool, and the new symbols dialog [13] (shown in the right side of the screenshot below). (Many symbol libraries for Inkscape are available on GitHub; Xaviju's inkscape-open-symbols set [14] is fantas-

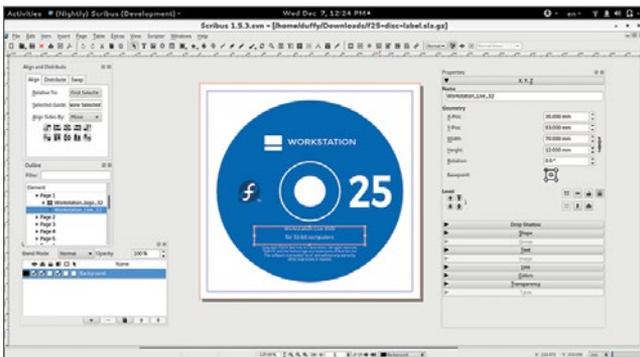
tic.) A new feature available in development/nightly builds is the *Objects* dialog that catalogs all objects in a document and provides tools to manage them.



Scribus

Scribus [15] is a powerful desktop publishing and page layout tool. Scribus enables you to create sophisticated and beautiful items, including newsletters, books, and magazines, as well as other print pieces. Scribus has color management tools that can handle and output CMYK and spot colors for files that are ready for reliable reproduction at print shops.

1.4.6 [16] is the latest stable release of Scribus; the 1.5.x [17] series of releases is the most exciting as they serve as a preview to the upcoming 1.6.0 release. Version 1.5.3 features a Krita file (*.KRA) file import tool; other developments in the 1.5.x series include the *Table* tool, text frame welding, footnotes, additional PDF formats for export, improved dictionary support, dockable palettes, a symbols tool, and expanded file format support.



MyPaint

MyPaint [18] is a drawing tablet-centric expressive drawing and illustration tool. It's lightweight and has a minimal interface with a rich set of keyboard shortcuts so that you can focus on your drawing without having to drop your pen.

MyPaint 1.2.0 [19] is the latest stable release and includes new features, such as the intuitive inking tool [20] for tracing over pencil drawings, new flood fill tool, layer groups, brush and color history panel, user interface revamp including a dark theme and small symbolic icons, and

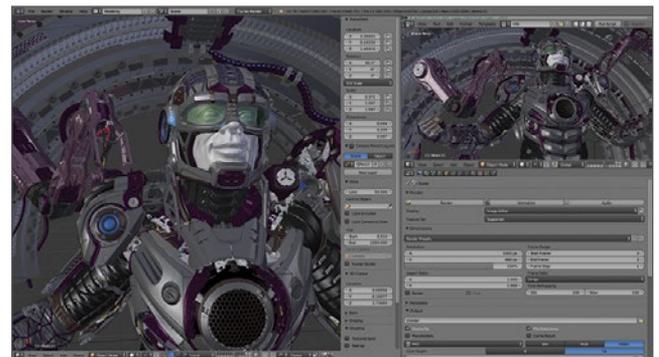
editable vector layers. To try out the latest developments in MyPaint, I recommend installing the nightly Flatpak build, although there have not been significant feature additions since the 1.2.0 release.



Blender

Initially released in January 1995, Blender [21], like GIMP, has been around for more than 20 years. Blender is a powerful open source 3D creation suite that includes tools for modeling, sculpting, rendering, realistic materials, rigging, animation, compositing, video editing, game creation, and simulation.

The latest stable Blender release is 2.78a [22]. The 2.78 release was a large one and includes features such as the revamped *Grease Pencil* 2D animation tool; VR rendering support for spherical stereo images; and a new drawing tool for freehand curves.



To try out the latest exciting Blender developments, you have many options, including:

- The Blender Foundation makes unstable daily builds [23] available on the official Blender website.
- If you're looking for builds that include particular in-development features, graphical.org [24] is a community-moderated site that provides special versions of Blender (and occasionally other open source creative apps) to enable artists to try out the latest available code and experiments.
- Mathieu Bridon has made development versions of Blender available via Flatpak. See his blog post for details: Blender nightly in Flatpak [25].

Krita

Krita [26] is a digital drawing application with a deep set of capabilities. The application is geared toward illustrators, concept artists, and comic artists and is fully loaded with extras, such as brushes, palettes, patterns, and templates.

The latest stable version is Krita 3.0.1 [27], released in September 2016. Features new to the 3.0.x series include 2D frame-by-frame animation; improved layer management and functionality; expanded and more usable shortcuts; improvements to grids, guides, and snapping; and soft-proofing.

**Video tools**

There are many, many options for open source video editing tools. Of the members of the pack, Flowblade [28] is a newcomer and Kdenlive is the established, newbie-friendly, and most fully featured contender. The main criteria that may help you eliminate some of this array of options is supported platforms—some of these only support Linux. These all have active upstreams and the latest stable versions of each have been released recently, within weeks of each other.

Kdenlive

Kdenlive [29], which was initially released back in 2002, is a powerful non-linear video editor available for Linux and OS X (although the OS X version is out-of-date). Kdenlive has a user-friendly drag-and-drop-based user interface that accommodates beginners, and with the depth experts need.

Learn how to use Kdenlive with an multi-part Kdenlive tutorial series [30] by Seth Kenlon.

- Latest Stable: 16.08.2 (October 2016)

Flowblade

Released in 2012, Flowblade [31], a Linux-only video editor, is a relative newcomer.

- Latest Stable: 1.8 (September 2016)

Pitivi

Pitivi [32] is a user-friendly free and open source video editor. Pitivi is written in Python [33] (the “P” in Pitivi), uses the GStreamer [34] multimedia framework, and has an active community.

- Latest stable: 0.97 (August 2016)
- Get the latest version with Flatpak [35]

Shotcut

Shotcut [36] is a free, open source, cross-platform video editor that started back in 2004 [37] and was later rewritten by current lead developer Dan Dennedy [38].

- Latest stable: 16.11 (November 2016)
- 4K resolution support
- Ships as a tarballed binary

OpenShot Video Editor

Started in 2008, OpenShot Video Editor [39] is a free, open source, easy-to-use, cross-platform video editor.

- Latest stable: 2.1 [40] (August 2016)

Utilities**SwatchBooker**

SwatchBooker [41] is a handy utility, and although it hasn’t been updated in a few years, it’s still useful. SwatchBooker helps users legally obtain color swatches from various manufacturers in a format that you can use with other free and open source tools, including Scribus.

GNOME Color Manager

GNOME Color Manager [42] is the built-in color management system for the GNOME desktop environment, the default desktop for a bunch of Linux distros. The tool allows you to create profiles for your display devices using a colorimeter, and also allows you to load/managed ICC color profiles for those devices.

GNOME Wacom Control

The GNOME Wacom controls [43] allow you to configure your Wacom tablet in the GNOME desktop environment; you can modify various options for interacting with the tablet, including customizing the sensitivity of the tablet and which monitors the tablet maps to.

Xournal

Xournal [44] is a humble but solid app that allows you to hand write/doodle notes using a tablet. Xournal is a useful tool for signing or otherwise annotating PDF documents.

PDF Mod

PDF Mod [45] is a handy utility for editing PDFs. PDF Mod lets users remove pages, add pages, bind multiple single PDFs together into a single PDF, reorder the pages, and rotate the pages.

SparkleShare

SparkleShare [46] is a git-backed file-sharing tool artists use to collaborate and share assets. Hook it up to a GitLab repo and you've got a nice open source infrastructure for asset management. The SparkleShare front end nullifies the inscrutability of git by providing a dropbox-like interface on top of it.

Photography

Darktable

Darktable [47] is an application that allows you to develop digital RAW files and has a rich set of tools for the workflow management and non-destructive editing of photographic images. Darktable includes support for an extensive range of popular cameras and lenses.

Entangle

Entangle [48] allows you to tether your digital camera to your computer and enables you to control your camera completely from the computer.

Hugin

Hugin [49] is a tool that allows you to stitch together photos in order to create panoramic photos.

2D animation

Synfig Studio

Synfig Studio [50] is a vector-based 2D animation suite that also supports bitmap artwork and is tablet-friendly.

Blender Grease Pencil

I covered Blender above, but particularly notable from a recent release is a refactored grease pencil feature [51], which adds the ability to create 2D animations.

Krita

Krita [52] also now provides 2D animation functionality.

Music and audio editing

Audacity

Audacity [53] is popular, user-friendly tool for editing audio files and recording sound.

Ardour

Ardour [54] is a digital audio workstation with an interface centered around a record, edit, and mix workflow. It's a little more complicated than Audacity to use but allows for automation and is generally more sophisticated. (Available for Linux, Mac OS X, and Windows.)

Hydrogen

Hydrogen [55] is an open source drum machine with an intuitive interface. It provides the ability to create and arrange various patterns using synthesized instruments.

Mixxx

Mixxx [56] is a four-deck DJ suite that allows you to DJ and mix songs together with powerful controls, including beat looping, time stretching, and pitch bending, as well as live broadcast your mixes and interface with DJ hardware controllers.

Rosegarden

Rosegarden [57] is a music composition suite that includes tools for score writing and music composition/editing and provides an audio and MIDI sequencer.

MuseScore

MuseScore [58] is a music score creation, notation, and editing tool with a community of musical score contributors.

Additional creative tools

MakeHuman

MakeHuman [59] is a 3D graphical tool for creating photorealistic models of humanoid forms.

Natron

Natron [60] is a node-based compositor tool used for video post-production and motion graphic and special effect design.

FontForge

FontForge [61] is a typeface creation and editing tool. It allows you to edit letter forms in a typeface as well as generate fonts for using those typeface designs.

Valentina

Valentina [62] is an application for drafting sewing patterns.

Calligra Flow

Calligra Flow [63] is a Visio-like diagramming tool. (Available for Linux, Mac OS X, and Windows.)

Helpful sites

There are a lot of toys and goodies to try out there. Need some inspiration to start your exploration? These websites and conference are chock-full of tutorials and beautiful creative works to inspire you get you going:

- [pixls.us \[64\]](#): Blog hosted by photographer Pat David that focuses on free and open source tools and workflow for professional photographers.
- [David Revoy's Blog \[65\]](#) The blog of David Revoy, an immensely talented free and open source illustrator, concept

Top open innovations in 3D printing

BY TOM CALLAWAY

OPEN SOURCE continues to drive rapid innovation in the 3D printing industry. This makes sense if you stop and think about it—a 3D printer exists to make other things. Combining that philosophy with free software and open source hardware helps other people participate in improving the objects that it makes, and in making the printers faster, smarter, and cleaner.

Here are a few of my favorite open source 3D printing innovations from 2016:

Prusa i3 MK2

Figure 1: Courtesy of prusa3d.com



Josef Průša is one of the core developers of the RepRap project, and the line of printers that bears his name continues to get better and better, without sacrificing any of the



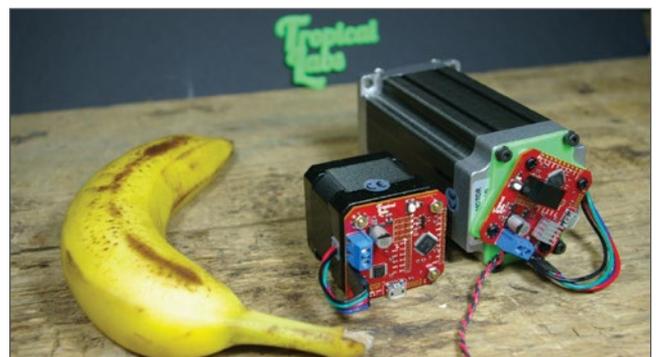
Image by: Model is "Little Lizard" by loubie, CC-BY-NC

open source goodness that makes it possible. In 2016, the Prusa i3 MK2 3D printer was released, and it is the first printer to correct its geometry in all axes automatically. This means that you no longer have to be as concerned about how precisely the printer is assembled or calibrated—the firmware included in the printer will do everything it can to ensure that the prints come out perfect. The printer uses nine special calibration points on the printer bed, combined with a probe that detects those points, then determines the deviation (if any) on the X and Y axes. This allows the printer to recalculate and adjust for any skew. It uses the same input to determine the Z height at all of the calibration points and adjust as needed. This technique is called mesh bed leveling and it eliminates most (if not all) imperfections in the bed.

There's a lot more math and engineering to explain how all of this works, but the end result is better prints every time, all powered with open source firmware and hardware. For a deeper dive into how it works, learn more on the Prusa Printers site [1] (or watch the demo video [2]).

Mechaduino

Figure 2: Hackaday.io [3]. CC BY-SA 4.0



3D printers are usually powered by NEMA stepper motors. The motors turn belts, which cause the printer to move in

three dimensions. The motors are cheap and simple, but not terribly accurate. To get accuracy, you usually want to use industrial servo motors, but they aren't cheap or simple. Enter Tropical Labs and their Mechaduino [4].

Mechaduino is an open source industrial servo control platform that provides a combination of the low cost of mass-produced stepper motors with high-resolution accuracy. Their vision is to be the “Arduino for mechatronics”. Although this innovative board will have a big impact on a number of different electronics projects, the fact that this can be a drop-in addition to most 3D printers makes it very exciting to me. Lots of people agreed, as this easily got funded via Kickstarter, reaching 855% of their goal.

3D printing with clay

Figure 3: Tom Lauerman. CC BY-SA 4.0



Most 3D printers use plastic filament or resin. A few use more exotic materials, such as sugar or chocolate, but Tom Lauerman wanted to use clay. Lauerman, an assistant professor at Penn State, designed his own 3D printer in collaboration with the University's Learning Factory program and the Center for Innovative Material Processing through Direct Digital Deposition (CIMP-3D). Unlike some academic printer projects, the Bricoleur Clay Extruder [5] wasn't built from scratch, patented, documented in an academic paper that few would ever read, and then shelved in a dusty basement on campus. Lauerman built upon existing open source 3D clay print head designs, and as a result, he remained committed to releasing all of his designs under an open source license so that others could print and improve upon his design.

Lulzbot TAZ 6

Figure 4: lulzbot.com [6]. CC BY-SA 4.0



I am a huge fan of the Lulzbot printers, and the TAZ 6, released in 2016, is no different. There is a reason that it is “Earth's highest rated desktop 3D printer,” and that reason is *freedom*. Everything that goes into the TAZ 6 is *free as in freedom* [7]—the firmware that drives it, the parts that compose it, the software that controls it—there is no secret sauce in this printer. The TAZ 6 is yet another proof point that open source does not mean sloppy or beta.

The printer itself contains a large number of 3D-printed pieces, printed on a fleet of other TAZ printers. Freedom and open source are part of the Lulzbot DNA, and the parent company (Aleph Objects) actively works with their community to come up with hacks, improvements, and modifications to make the printer more robust, accurate, and intelligent. The result of this transparency, remixing, and collaboration is a simple yet powerful 3D printing experience that appeals to an audience from students to maker gurus.

(Full disclosure: We use Lulzbot 3D printers in the Red Hat 3D printing lab at the Red Hat headquarters in Raleigh, North Carolina.)

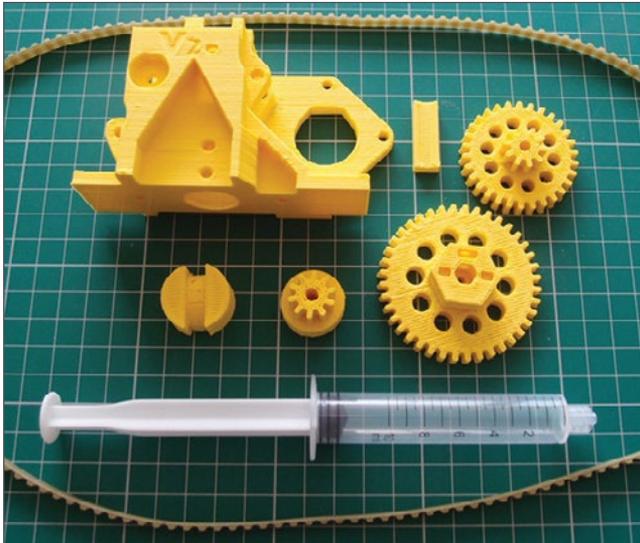
Open source bioprinting

Early in 2016, Ourobotics [9], an Irish 3D bioprinting start-up, introduced the Renegade, an open source bioprinter that can be assembled for under US\$ 900. This printer is based on Richard Horne's Universal Paste Extruder [10]. For much of the 3D printing community, bioprinting is seen as a holy grail of the technology. If we can print replacement organs, then we can improve and save lives around the world. We're

not quite there yet, but Ourobotics saw the value of innovating in the open source way to help us get to that goal faster and more affordably.

Sadly, in August, the CEO of Ourobotics, Jemma Redmond passed away unexpectedly. One of the small comforts of open source work is the knowledge that those projects can live on, even after we are gone from this world.

Figure 5: RichRap [8]. CC BY-SA 4.0



Resources

- [1] <http://prusaprinters.org/first-printer-to-automatically-correct-geometry-in-all-axes/>
- [2] <http://bit.ly/2i8Xt5r>
- [3] <https://hackaday.io/project/11224-mechaduino>
- [4] <http://tropical-labs.com/index.php/mechaduino>
- [5] <http://www.thingiverse.com/thing:1413969>
- [6] <https://www.lulzbot.com/>
- [7] <https://www.gnu.org/philosophy/free-sw.en.html>
- [8] <http://www.thingiverse.com/thing:20733>
- [9] <http://ouro-botics.com/>
- [10] <http://www.thingiverse.com/thing:20733>

Author

Tom Callaway has been a Red Hat employee since 2001 and is the co-author of *Raspberry Pi Hacks* (O'Reilly, 2014). Currently he leads the Education Outreach team at Red Hat to promote FOSS in schools. He maintains or co-maintains a large number of packages in Fedora (350+), and is responsible for managing Fedora's legal issues. Tom frequently represents Fedora and free software at conferences around the world, and tries his best not to make too big of a fool of himself. When not working, Tom enjoys geocaching, ice hockey, gaming, science fiction, 3D printing, traveling, and pinball.





Most Likely to Succeed

10 open source projects to watch in 2017

.....BY JASON BAKER

NO ONE has a crystal ball to see the future of technology. Even for projects developed out in the open, code alone can't tell us whether or not a project is destined for success—but there are hints along the way.

For example, perhaps it's not unreasonable to assume that the projects that will help shape our future are those projects that have first seen rapid growth and popularity among the developer community.

So which new projects should an open source developer watch in 2017? Let's take a look at a few projects that emerged in 2016 to achieve rapid notoriety in the GitHub community.

To develop this list, I went through GitHub with a focus on projects whose repository was created in 2016, and looked at the projects ranked by number of stars [1]. It's not a perfect system; there are, of course, repositories that contain something other than an open source project, and so these were omitted from the list. Of course, there also were many great projects introduced in 2016 whose development took place somewhere other than GitHub. Admittedly, the process of picking these 10 projects to watch for 2017 from a pool of many choices was as much of an art as a science. But I still think these projects are worth keeping an eye on in the new year.

Yarn

Yarn [2] pitches itself as providing “fast, reliable, and secure dependency management.” In short, it's a modern replacement for npm [3], a package manager built specifically for JavaScript

developers, which helped build the enthusiasm for using JavaScript across the entire application stack that seems so prevalent today. In addition to its speed and security features, yarn also features off-line installs, advanced dependency-management features, and deterministic design to ensure that packages install across multiple machines should match identically.



Image by: Opensource.com, CC BY-SA 4.0

Create React App

A new project from Facebook's incubator [4] project, Create React App [5] is, unsurprisingly, a template for creating React-based [6] applications without having to create a custom build configuration. Providing a simple command-line interface for generating new application, it's easy to create and deploy a simple application stack that gives developers the power of the React framework.

Android Architecture Blueprints

The Android Architecture Blueprints [7] repository is a great resource for learning from the UX team at Google best



Most Likely to Succeed

practices for organizing and architecting an Android app. By demonstrating several ways of handling common problems, the repository provides a starting point for creating a new application, or to inform a design decision in your existing app.

Hyper

For developers and system administrators, there are two tools that one simply cannot live without: a web browser and a terminal. Hyper [8] is an attempt to bring best attributes of a web application to a terminal emulator, creating a modern terminal experience with JavaScript, HTML, and CSS. Relying on web standards opens up customization and control to a whole new audience who can use their existing JavaScript skills to customize and optimize their terminal.

Parse server

Parse server [9] is a Node.js-based open source backend that makes it easy to migrate applications designed for Parse, after the announcement that the hosting service would be retired in early 2017. Designed to make creating web applications and APIs easier, Parse is cross-platform and works everywhere that Node.js can be deployed.

Bulma

Designing a good-looking website or web application can be difficult, and made even more so by the complexity of competing browser standards and the wide array of devices your users are viewing from. Bulma [10] is a modern CSS framework designed to be responsive and modular, easing development for UX teams trying to design interfaces that flow naturally.

TensorFlow models

TensorFlow, the Google-driven machine learning framework, was one of our top projects from the 2015 Open

Source Yearbook [11]. Looking back at the growing interest in AI over the past 12 months, finding another TensorFlow-related project in this year's batch should come as no surprise, with this repository of TensorFlow models [12] earning more than 10,000 stars. Conducting tasks from name generation and learning, image to text processing, and classification, this is a great starting point for anyone who wants to learn more about TensorFlow while getting their hands a little dirty.

Anime

If you're interested in web animation, give Anime [13] a look. Anime is a JavaScript animation engine that works with CSS, SVG, the document object, and JavaScript objects to bring animation and interactivity to any web-based project. It's cross-platform, working on all of the major browsers, and is designed to make both simple and complex animations easy to implement.

Swift Algorithm Club

Another of our top project from last year's list was Swift, the open source language from Apple that has rapidly become a developer favorite. In this year's list is the Swift Algorithm Club [14], a collection of various algorithms and

data structures implemented in Swift that you can use for learning purposes or simply drop into your application. Including numerous sorting, searching, spanning, and tree algorithms, Swift Algorithm Club is an amateur computer scientist's wishlist of code implementations.

Weex

The final entry in this year's top 10 is Weex [15], a framework designed to make developing a cross-platform user interface for mobile applications easier. Weex is designed to be fast, lightweight, and extensible, allowing you to get near native performance without having to write a different native app for each platform.

Honorable mentions

As I explained, a few new repositories were emerging on GitHub this year that, by popularity, may have made this list, but weren't strictly speaking properly-licensed open source projects. Here are a few of my favorites:

- HEAD [16]: A comprehensive list of the many uses for the "head" section of an HTML document, from providing meta information to browser directives to social sharing hints.
- Google Interview University [17]: One developer's self-study plan for moving from web developer to software engineer—essentially, a computer science knowledge checklist.
- Public APIs [18]: A list of publicly available APIs to return JSON data on just about anything you can imagine, along with links to their documentation.
- A security guide for developers [19]: A work in progress containing an outline and checklist for security-minded developers.
- *How to Be a Programmer* [20]: A book about the hard and soft skills that are necessary to master in order to be successful in a software development career.

Are there any projects you're particularly interested in watching in 2017? Let us know about them—send us an article proposal [21].

Resources:

- [1] <http://bit.ly/2ivEFgu>
- [2] <https://github.com/yarnpkg/yarn>
- [3] <https://www.npmjs.com/>
- [4] <https://github.com/facebookincubator>
- [5] <https://github.com/facebookincubator/create-react-app>
- [6] <https://facebook.github.io/react/>
- [7] <https://github.com/googlesamples/android-architecture>
- [8] <https://github.com/zeit/hyper>
- [9] <https://github.com/ParsePlatform/parse-server>
- [10] <https://github.com/jgthms/bulma>
- [11] <https://opensource.com/life/15/12/most-likely-succeed-2016>
- [12] <https://github.com/tensorflow/models>
- [13] <https://github.com/juliangarnier/anime>
- [14] <https://github.com/raywenderlich/swift-algorithm-club>
- [15] <https://github.com/alibaba/weex>
- [16] <https://github.com/joshbuechea/HEAD>
- [17] <https://github.com/jwasham/google-interview-university>
- [18] <https://github.com/toddmotto/public-apis>
- [19] <https://github.com/FallibleInc/security-guide-for-developers>
- [20] <https://github.com/braydie/HowToBeAProgrammer>
- [21] <https://opensource.com/story>

Author

Jason Baker is passionate about using technology to make the world more open, from software development to bringing sunlight to local governments. He is particularly interested in data visualization/analysis, DIY/maker culture, simulations/modeling, geospatial technologies, and cloud computing, especially OpenStack. Follow him on Twitter: @jehb



How Linux got to be Linux: Test driving 1993-2003 distros

BY SETH KENLON

A UNIQUE TRAIT of open source is that it's never truly EOL (End of Life). The disc images mostly remain online, and their licenses don't expire, so going back and installing an old version of Linux in a virtual machine and getting a precise picture of what progress Linux has made over the years is relatively simple.

We begin our journey with Slackware 1.01, posted to the **comp.os.linux.announce** newsgroup well over 20 years ago.

Slackware 1.01 (1993)

Figure 1: Slackware 1.01

```

220-----
220          R S Y N C . O S U O S L . O R G
220          Oregon State University
220          Open Source Lab
220
220          Unauthorized use is prohibited - violators will be prosecuted
220-----
220          For more information about the DSL visit:
220          http://osuosl.org/services/hosting
220
220          This host is the home to the primary archives of several
220          projects. We would prefer that only primary/secondary
220          mirrors use this service. Thanks!
220-----
220
220
220 Name (ftp.slackware.com:klaatu): anonymous
220 Password (ftp.slackware.com:anonymous):
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
    
```

The best part about trying Slackware 1.01 is that there's a pre-made image in Qemu's 2014 series [1] of free images, so you don't have to perform the install manually (don't get used to this luxury).

```

$ qemu-kvm -m 16M -drive if=ide,format=qcow2,file=slackware.qcow2 \
-netdev user,id=slirp -device ne2k_isa,netdev=slirp \
-serial stdio -redir tcp:22122::22
    
```

Many things in 1993's version of Linux works just as you'd expect. All the basic commands, such as `ls` and



Image by: Internet Archive Book Images. Modified by Opensource.com.

`cd` work, all the basic tools (`gawk`, `cut`, `diff`, `perl`, and of course Volkerding's favorite [2] `elvis`) are present and accounted for, but some of the little things surprised me. `BASH` courteously asks for confirmation when you try to tab-complete hundreds of files, and tools to inspect compressed files (such as `zless` and `zmore` and `zcat`) already existed. In more ways than I'd expected, the system feels surprisingly modern.

What's missing is any notion of package management. All installs and uninstalls are entirely manual, with no tracking.

Over all, Slackware 1.01 feels a lot like a fairly modern UNIX—or more appropriately, it feels like modern UNIX might feel to a Linux user. Most everything is familiar, but there are differences here and there. Not nearly as much a difference as you might expect from an operating system released in 1993!

Debian 0.91 (1994)

To try Debian 0.91, I used the floppy disk images available on the Ibiblio digital archive [3], originally posted in 1994. The commands to boot:

```

$ gunzip bootdisk.gz basedsk1.gz basedsk2.gz
$ qemu-system-i386 -M pc -m 64 -boot order=ac,menu=on \
-drive file=bootdisk,if=floppy,format=raw \
-drive file=debian.raw,if=ide,format=raw \
-device ne2k_isa,netdev=slirp \
-serial msmouse -vga std \
-redir tcp:22122::22 \
-netdev user,id=slirp
    
```

The bootdisk for Debian 0.91 boots to a simple shell, with clear instructions on the steps you're meant to take next.

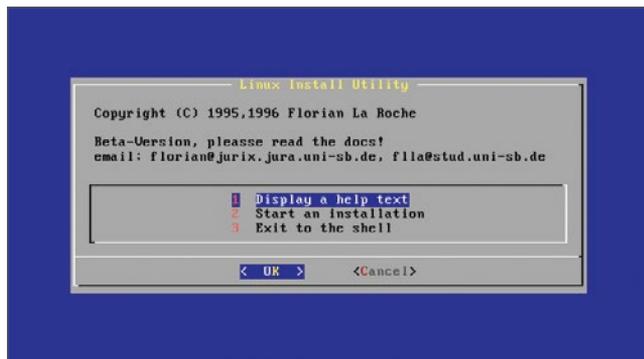
The install process is surprisingly smooth. It works off of a menu system with seven steps—from partitioning a hard drive and writing the ext2 filesystem to it, all the way through to copying the basedsk images. This provided a minimal Debian install with many of the familiar conventions any modern Linux user would expect from their OS.

Debian is now famous for its package management system, but there are mere hints of that in this early release. The dpkg command exists, but it's an interactive menu-based system—a sort of clunky aptitude, with several layers of menu selections and, unsurprisingly, a fraction of available packages.

Even so, you can sense the convenience factor in the design concept. You download three floppy images and end up with a bootable system, and then use a simple text menu to install more goodies. I sincerely see why Debian made a splash.

Jurix/S.u.S.E. (1996)

Figure 2: Jurix installation



A pre-cursor to SUSE, Jurix shipped with binary .tgz packages organized into directories resembling the structure of Slackware's install packages. The installer itself is also similar to Slackware's installer.

```
$ qemu-system-i386 -M pc -m 1024 \
  -boot order=ac,menu=on \
  -drive \
    file=jurix/install,if=floppy,format=raw \
  -drive file=jurix.img,if=ide \
  -drive file=pkg.raw,if=ide,format=raw \
  -device ne2k_isa,netdev=sliirp \
  -serial msmouse -vga std \
  -redir tcp:22122::22 \
  -netdev user,id=sliirp
```

Because I wasn't specifically looking for the earliest instance, Jurix was the first Linux distribution I found that really "felt" like it intended the user to use a GUI environment. XFree86 [4]

is installed by default, so if you didn't intend to use it, you had to opt out.

An example /usr/lib/X11/XF86Config (this later became Xorg.conf) file was provided, and that got me 90% of the way to a GUI, but fine-tuning vsync, hsync, and ramdac colormap overrides took me an entire weekend until I finally gave up.

Installing new packages on Jurix was simple; find a .tgz on your sources drive, and run a routine tar command: \$ su -c 'tar xzvf foo.tgz -C /' The package gets unzipped and unarchived to the root partition, and ready to use. I did this with several packages I hadn't installed to begin with, and found it easy, fast, and reliable.

SUSE 5.1 (1998)

Figure 3: FVWM running on SuSE 5.1



I installed SUSE 5.1 from a InfoMagic CD-ROM purchased from a software store in Maryland in 1998.

```
$ qemu-system-i386 -M pc-0.10 -m 64 \
  -boot order=ad,menu=on \
  -drive file=floppy.raw,if=floppy,format=raw \
  -cdrom /dev/sr0 \
  -drive file=suse5.raw,if=ide,format=raw \
  -vga cirrus -serial msmouse
```

The install process was convoluted compared to those that came before. YaST volleyed configuration files and settings between a floppy disk and the boot CD-ROM, requiring several reboots and a few restarts as I tried to understand the sequence expected from me. Once I'd failed the process twice, I got used to the way YaST worked, and the third time was smooth and very much a hint at the Linux user experience to come in later years.

A GUI environment was my main goal for SUSE 5.1. The configuration process was familiar, with a few nice graphical tools (including a good XF86Setup frontend) to

help test and debug mouse and monitor problems. It took less than an hour to get a GUI up and running, and most of the delay was caused by my own research on what resolutions and color depths Qemu's virtualized video card could handle.

Included desktops were fvwm, fvwm2, and ctwm. I used fvwm, and it worked as expected. I even discovered tkDesk, a dock and file manager combo pack that is surprisingly similar to Ubuntu's Unity launcher bar.

The experience was, over all, very pleasant, and in terms of getting a successful desktop up and running, SUSE 5.1 was a rousing success.

Red Hat 6.0 (1999)

Figure 4: Red Hat 6 running GIMP 1.x



The next install disc I happened to have lying around was Red Hat 6.0. That's not RHEL 6.0—just Red Hat 6.0. This was a desktop distribution sold in stores, before RHEL or Fedora existed. The disc I used was purchased in June 1999.

```
$ qemu-system-i386 -M pc-0.10 -m 512 \
  -boot order=ad,menu=on \
  -drive file=redhat6.raw,if=ide,format=raw \
  -serial msmouse -netdev user,id=slirp \
  -vga cirrus -cdrom /dev/sr0
```

The installation was fully guided and remarkably fast. You never have to leave the safety of the install process, whether choosing what packages to install (grouped together in **Workstation**, **Server**, and **Custom** groups), partitioning a drive, or kicking off the install.

Red Hat 6 included an xf86config application to step you through X configuration, although it strangely allowed some mouse emulation options that X later claimed were invalid. It beat editing the Xf86Config file, but getting X correct was still clearly not a simple task.

The desktop bundled with Red Hat 6 was, as it still is, GNOME, but the window manager was an early Enlightenment [5], which also provided the main sound daemon. Xdm and gdm were both provided as login managers so that normal users could log in without having the permission to start or kill X itself, which is particularly important on multi-user systems.

Certain staple applications are missing; gedit didn't exist yet, there's no grand unified office application, and there was no package manager to speak of. GnoRPM, a GUI interface for RPM installation, review, and removal, was the closest to a yum or PackageKit experience it had, and gnotepad+ is the GUI text editor (Emacs notwithstanding, obviously).

Over all, though, the desktop is intuitive. Unlike later implementations of GNOME, this early version featured a panel at the bottom of the screen, with an application menu and launcher icons and virtual desktop control in a central location. I can't imagine a user of another operating system at the time finding this environment foreign.

Red Hat 6 was a strong entry for Linux, which was obviously moving seriously toward being a proper desktop OS.

Mandrake 8.0 (2001)

Figure 5: Mandrake: A turning point in Linux



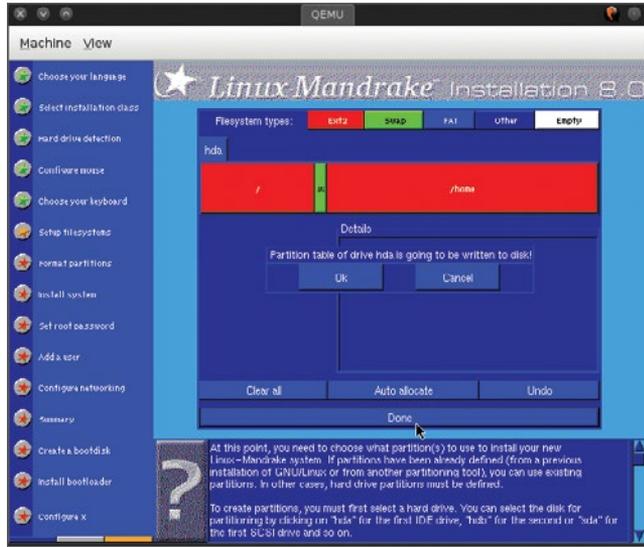
Mandrake 8.0 was released in 2001, so it would have been compared to, for instance, Apple OS 9.2 and Windows ME.

I fell back on fairly old emulated tech to be safe.

```
$ qemu-system-i386 \
  -M pc-0.10 -m 2048 \
  -boot order=ad,menu=on \
  -drive file=mandrake8.qcow2 \
  -usb -net nic,model=rtl8139 \
  -netdev user,id=slirp \
  -vga cirrus \
  -cdrom mandrake-8.0-i386.iso
```

I'd thought the Red Hat installation process had been nice, but Mandrake's was amazing. It was friendly, it gave the user a chance to test configurations before continuing, it was easy and fast, and it worked almost like magic. I didn't even have to import my XF86Config file, because Mandrake's installer got it right.

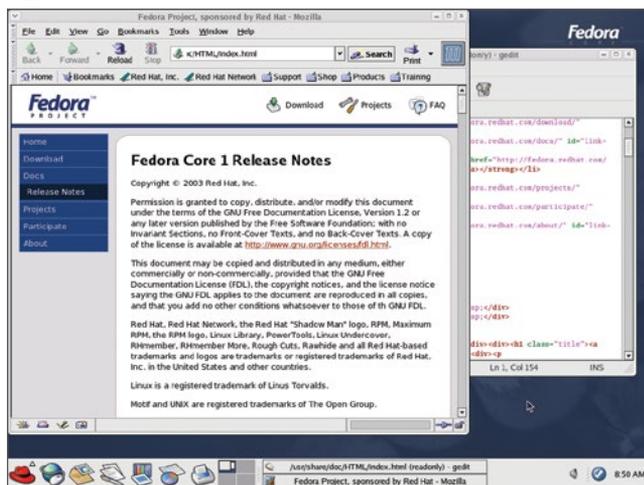
Figure 6: Mandrake 8.0 installer



Using the Mandrake desktop is a lot like using any given desktop of the time, actually. I was a little surprised at how similar the experience was. I feel certain that if I'd somehow stumbled into Mandrake Linux at this time, it actually wouldn't have been beyond my ability, even as a young and not very technical user. The interfaces are intuitive, the documentation helpful, and the package management quite natural, for a time when it still wasn't yet the mental default for people to just go to a website and download an installer for whatever software they wanted.

Fedora 1 (2003)

Figure 7: Blue Fedora, Red Hat



In 2003, the new Fedora Core distribution was released. Fedora Core was based on Red Hat, and was meant to carry on the banner of desktop Linux once Red Hat Enterprise Linux (RHEL) became the flagship product of the company.

Nothing particularly special is required to boot the old Fedora Core 1 disc:

```
$ qemu-system-i386 -M pc \
  -m 2048 -boot order=ac,menu=on \
  -drive file=fedora1.qcow2 -usb \
  -net nic,model='rtl8139' -netdev user \
  -vga cirrus -cdrom fedora-1-i386-cd1.iso
```

Installing Fedora Core is simple and familiar; it uses the same installer as Fedora and Red Hat for the next 9 years. It's a graphical interface that's easy to use and easy to understand.

Figure 8: Anaconda GUI



The Fedora Core experience is largely indistinguishable from Red Hat 6 or 7. The GNOME desktop is polished, there are all the signature configuration helper applications, and the presentation is clean and professional.

A *Start Here* icon on the desktop guides the user toward three locations: an *Applications* folder, the *Preferences* panel, and *System Settings*. A red hat icon marks the applications menu, and the lower GNOME panel holds all the latest Linux application launchers, including the OpenOffice office suite and the Mozilla browser.

The future

By the early 2000s, it's clear that Linux has well and truly hit its stride. The desktop is more polished than ever, the applications available want for nothing, the installation is easier and more efficient than other operating systems. In fact, from the early 2000s onward, the relationship between the user

and the system is firmly established and remains basically unchanged even today. There are some changes, and of course several updates and improvements and a staggering amount of innovation.

Project names come and go:

- Mandrake became Mandriva and then Mageia [6];
- Fedora Core became just Fedora [7];
- Ubuntu [8] popped up from Debian [9] and helped make “Linux” a household term;
- Valve has made SteamOS [10] the official basis for its gaming platform; and
- Slackware [11] quietly continues to this day.

Whether you’re new to Linux, or whether you’re such an old hand that most of these screenshots have been more biographical than historical, it’s good to be able to look back at how one of the largest open source projects in the world has developed. More importantly, it’s exciting to think of where Linux is headed and how we can all be a part of that, starting now, and for years to come.

Resources

- [1] <http://www.qemu-advent-calendar.org/2014>
- [2] <http://www.slackware.com/~volkerdi/>
- [3] <https://ibiblio.org/pub/historic-linux/distributions/debian-0.91/debian-0.91/dist>
- [4] <http://www.xfree86.org/>
- [5] <http://enlightenment.org>
- [6] <http://mageia.org>
- [7] <http://fedoraproject.org>
- [8] <http://ubuntu.com>
- [9] <http://debian.org>
- [10] <http://store.steampowered.com/steam>
- [11] <http://slackware.com>

Author

Seth Kenlon is an independent multimedia artist, free culture advocate, and UNIX geek. He is one of the maintainers of the Slackware-based multimedia production project, <http://slackrmedia.ml>.



WRITE FOR US

Would you like to write for [Opensource.com](https://opensource.com)? Our editorial calendar includes upcoming themes, community columns, and topic suggestions: <https://opensource.com/calendar>

Learn more about writing for Opensource.com at: <https://opensource.com/writers>

We’re always looking for open source-related articles on the following topics:

Big data: Open source big data tools, stories, communities, and news.

Command-line tips: Tricks and tips for the Linux command-line.

Containers: Getting started with containers, best practices, security, news, projects, and case studies.

Education: Open source projects, tools, solutions, and resources for educators, students, and the classroom.

Geek culture: Open source-related geek culture stories.

Hardware: Open source hardware projects, maker culture, new products, howtos, and tutorials.

High-performance computing: Open source tools, programs, projects, and howtos for research and science.

Programming: Share your favorite scripts, tips for getting started, tricks for developers, tutorials, and tell us about your favorite programming languages and communities.

Security: Tips and tricks for securing your systems, best practices, checklists, tutorials and tools, case studies, and security-related project updates.

Compute like it's 1989

BY SETH KENLON

FOR MANY OF US, when we look around at the state of computing in 2016, we nod and think, “Yes, today is what I expected when I thought about what The Future would be like.” Sure, we haven’t got flying cars yet, but today’s technology is flashy. We swipe fingers across screens instead of pressing buttons, and I’m told we are all very excited about all of the latest virtual reality headsets and augmented reality gadgets.

So now seems as good a time as any to look back at how people of the past used to compute, and back to the days when a “desktop” computer was so

called because it took up 80 percent of your desktop. The days when the term “computer” actually meant “a machine for computation.”

Why bother looking back 30-year-old computing? After all, the computers back then were clunky, slow, and awkward, weren’t they? Sure, they were, but the great thing about living in The Future is that we have the power to look back at the Old Ways and cherry pick information from them for modern technology. The truth is, there’s power in simplicity, and old computing was simple out of necessity.

Survival of the thriftiest

Have you ever played a survival video game? The kind in which you must survive a zombie apocalypse by hoarding canned food and using rolling pins as weapons? Computers are a little like that.

People make complaints in modern computing about bloat, and the cavalier answer is usually “CPU cycles are cheap!”—or in other words, “I’d prefer to treat the symptom rather than the cause.” In the old days, CPU cycles were prohibitively expensive, and even now, “cheap” isn’t the same as “free.”



Image by: LSE Library. Modified by Opensource.com.

For \$89, you can get a PocketCHIP [1] and a keyboard, and have a tiny computer ready to use at any moment, but its CPU cycles are not cheap. What you save on its cost and power requirement you pay for in computing power, but you’ll hardly notice, as long as you’re okay with simplified computing.

The same is true for server slices or instances on shared servers, or within Crouton [2]. The computer is there for the taking, but you have to learn to conserve your resources, whether it’s bandwidth or CPU cycles or RAM.

A workflow of your own

Old computers weren’t the pre-fab ready-to-wear appliances that we’re used to now. Hardware is neat, but I’m not talking about the hardware. The stuff we interact with on a daily basis is the software, and “integrated” software is a relatively new idea.

Back before big monolithic applications got dumped onto the unsuspecting public, software came in batches on “shareware” diskettes and BBSes. If you wanted to create animation, you got software to help you draw images, used more software to string images into an animated sequence,

used other software to produce sound effects, and finally used software to combine the sound and the image.

There was freedom in that, both for your computer, which didn't need to be capable of running that one big app that tried to do everything, as well as for yourself because it was up to you which applications you strung together to get the job done.

Diving in

One way to compute like it's 1989 is to go and get a Raspberry Pi or a PocketCHIP and dive in to the wonderful world of low-powered living. The good news is that running Linux today is surprisingly similar to running UNIX or Linux in the '90s. The bulk of the commands you know and love are there, and many of the applications and the general sensibility have endured.

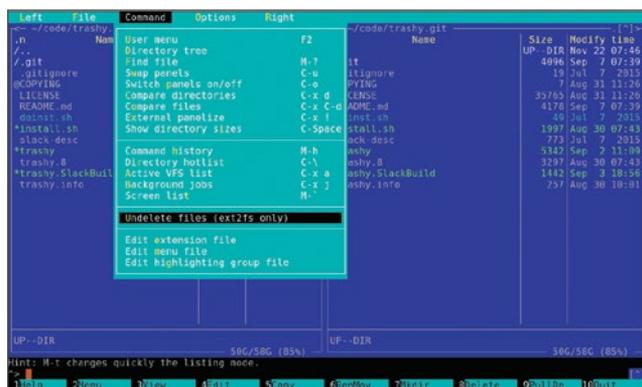
Most everything you do on your computer now can be done in a terminal, which obviously is the lightest of light-weight interfaces.

File management

The old style of file management was far more direct than modern interfaces allow. Modern computers have automagical file handling, according to a mimetype database. Mimetypes are useful when your main interaction with a file is pointing at it and double-clicking to open, but that becomes largely irrelevant when you yourself are making the call about which application to use. It might surprise you to learn how much faith we tend to put in auto-detection now. For instance, who knew you could run a valid text edit command with `sed` on a dynamic library or that you can see metadata in a sound file with `lss`? When you stop relying on pre-programmed decisions, then that is when you end up learning something new.

Most modern Linux users have at least heard of file management commands such as `cp` and `mv`, and that's an entirely valid and certainly the most efficient way to manage files. It's not the only option, though. If you yearn for that happy medium between constructing valid BASH commands and the intuitiveness of graphical interfaces, then look to GNU Midnight Commander [3].

Figure 1: Midnight Commander



Midnight Commander (invoked as `mc`) is a DOS-style utility that provides a split-pane file manager in your terminal. It's primarily keyboard-driven, although being written with the "ncurses" toolkit, it can also receive mouse-clicks.

The interface is wonderfully intuitive. All you need to know is that it's a file manager, and from there you can quickly learn it. Contextual commands are listed at the bottom of the window, each being assigned to a Function Key and a full menu (a "pull down" accessed with F9) is always available at the top of the window.

Personal customization notwithstanding, there are always two panels in Midnight Commander, and you switch between them with the Tab key. All the usual file operations are either a menu selection or a keypress away. It's intuitive enough that you can poke at it for a few minutes and fall into using it without introduction, and yet so efficient in design that you can quickly become a power user and control it with emacs-like key combos. It is, for all intents and purposes, a "desktop" for an MS- or Pro-DOS experience.

Networking

To a lot of people, the Internet and the "www" subdomain are one and the same. What many people don't realize is that the "www" subdomain is really just the "World Wide Web" part of the larger Internet and generally it's the part that serves content over HTTP(S).

What was called "Web 2.0" is a lot heavier than the Internet of old. With all the background videos, JavaScript pop-up requests for your email address, pleas for you to disable your ad-blocker, alerts about cookies, warnings that your browser is out of date, and everything else the modern web tries to throw into your browser, visiting the "www" in a non-mainstream browser is almost impossible. Luckily, there's a lot more happening on the Internet than just social media and comment wars.

Web browsers have conditioned most of us to view the web as a place in which you go and "hang out." You sit and idle; you go to it, but never bring it home. This, of course, isn't strictly true—you're downloading bits to a temporary cache, but that's all abstracted from you by the browser. You can still visit the modern web whilst living a digital retro lifestyle, but it's less about loitering and more about getting stuff done. The earliest Linux distributions shipped with Lynx and ELinks, which provide the typical HTTP experience modern web users are used to, but there were and are many other ways to interact with the Internet:

Atom and RSS

The advantages of these is that they are a "push" model instead of a "pull." You don't have to go out and check a website to see whether there are updated news items. The software sends you an alert instead. Much of my daily web browsing is taken care of with one look at newsbeuter [4] or Mashpodder [5]. Once you start using RSS and Atom, you

might just find that the sheen of HTTP is a lot duller than it seemed before.

Of the two, newsbeuter is the easiest to configure and use. Install it from your distribution's repository, and then launch it once to force it to instantiate its configuration file. Once that's done, you have only to edit your `~/.newsbeuter/ur1s` file; a simple line-delimited list of feeds you want to check. A sample from my current `ur1s` file:

```
$ head ~/.newsbeuter/ur1s
https://opensource.com/feed
http://slackware-change1og.oprod.net/atom_feed/
http://fedoraplanet.org/rss20.xml
https://planetkde.org/rss20.xml
http://planet.qt.io/rss20.xml
http://planetpython.org/rss20.xml
https://www.linux.com/feeds/rss
http://gnuworldorder.info/ogg.atom.xml
http://monsterjavaguns.com/podcast/feed
http://twodeeten.blogspot.com/feeds/posts/default
```

Wget, curl, fetch

Regardless of the UNIX you're running, you have available some command to access the network and fetch a file. It's browsing the web without the browsing, and it's sublime. Unfortunately, many modern sites obfuscate where the actual content is (if you're using wget or similar, then you're not clicking on their ads), but for the practical websites out there, a quick download command is liberating and efficient.

Git

Git itself is great, but another good thing about git's popularity is that people have actually started hosting blogs and other content in git repositories, which means you can easily grab that content using just a UNIX shell.

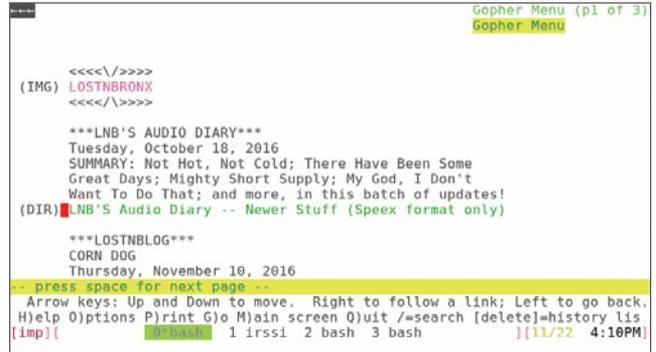
SSH

Thriving community servers are out there that are open to new users, and you can also build your own. You can find a list of free shell accounts [6], and now that you can get a computer for \$35, setting up your own is one install and port forward away, even if only as an experiment to see how many friends you can get to join you.

Gopher

There's an unusual amount of nostalgia out there for the Gopher protocol. It's not the greatest system ever invented (Gopher servers sometimes have trouble parsing Gopher's markup), but it does underscore one point that much of the web seems to have forgotten: It's all about the content, not the ads. When your site is serving lists of text and binary files, you inherit an objectivity that just gets lost in modern sites. The Lynx browser still recognizes Gopher, so start your journey with it [7].

Figure 2: Floodgap Public Gopher Proxy



Email

Of course there's always email, the original social network. Too many people these days fall back on Gmail and other providers that use over-complex web interfaces that cease to function at even the slightest variation of browser vendor or version. There is a better way, and that is MutT [8]. It's a lightweight, simple, efficient, and effective email client with more customized config files than you could ever need. Better yet, it has next-to-transparent PGP integration, so you can start encrypting those emails from end to end.

Your-Protocol-Here

Don't forget Usenet, Tor, GNUnet, and more. There are too many ways to access the worldwide network to list. If you look, you'll find all kinds of interesting lightweight technologies lying around out there.

Graphics without X

On some devices, an X server just isn't practical—sure, it might be possible, but you just know it's eating up a lot of precious RAM. You might run a lightweight desktop, but you'll still inherit the overhead of running X in the first place.

Mostly, when computing the Old Way, you don't need much by way of a graphical interface. A GUI just clutters things up, gets your hands away from the keyboard where they belong, and is painfully inefficient. If ever you're going to wish for a graphical display, it will be when you're online or checking email. People love graphics on the Internet, and people love embedding images into email.

Don't startx yet. What if I told you that you don't need to run X to display graphics on your screen? Thanks to the Linux framebuffer device, `/dev/fb0`, you can do just that.

There are a few different utilities to draw images straight to your screen without a graphic server. These don't work over remote or emulated connections (SSH, screen, tmux), but as long as you're sitting at the physical computer you're using, you can direct all kinds of output straight to the physical screen attached to it.

To view images, there's `fb i` (framebuffer image viewing) and its successor, `f i m` (Fbi IMproved). Both essentially do the same thing. Point it at a bitmap file and it'll paint the

picture on your display, abruptly, without fanfare or apology. You can use various controls; you can zoom, pan, or step through a slideshow. It's easy and immediate, and it's exactly what you need.

You can even play video without X, believe it or not. You need to make sure your username is a member of the “video” and “audio” groups (this is usually a default on even the bare-bones Linux distributions), and then:

```
$ mplayer -vo fbdev my_movie.mp4
```

Understand, this isn't a gimmicky “convert your images to ASCII” scenario—these tools actually display the images and video on your screen without a GUI. Depending on which shell you're using, painting pixels this way can confuse your input. If your shell starts to act funny after using fbdev, use the reset command and everything ought to return to normal.

The “you” in UNIX

UNIX training and training videos [9] produced in the 1980s made it abundantly clear that the intent of the operating

system was and is to empower users to take small commands and string them together to accomplish complex tasks. The individual's workflows were meant to be unique and infinitely extensible.

Beyond the mimicry of old computer interfaces and the rejection of modern network chatter is the enduring principle that computerists should be eager to find new tools, useful programs, and exciting ways to piece things together to accomplish tasks and to make life better for everyone. In other words, put the “you” in UNIX.

Resources

- [1] <https://getchip.com/pages/pocketchip>
- [2] <https://github.com/dnschneid/crouton>
- [3] <https://www.midnight-commander.org/>
- [4] <http://newsbeuter.org/>
- [5] <https://github.com/hirozed/mashpodder>
- [6] <http://shells.red-pill.eu>
- [7] <http://gopher.floodgap.com/gopher/>
- [8] <http://mutt.org>
- [9] <https://archive.org/details/UNIX1985>



LinuxQuestions.org celebrates sweet 16

BY JEREMY GARCIA

IN MY November 2016
OpenSource.
com column, The Queue [1],
I answered a multi-part question
received via email:

Why did you start LinuxQuestions.org [2]? And can you tell us a little about the history of the site?



Image by: Opensource.com. CC BY-SA 4.0

The answer

Computers, programming, and technology in general have always fascinated me. When I was in high school, I started working for a local ISP that used UNIX almost exclusively. The “UNIX way” just clicked and made a lot of sense to me. It wasn’t long before I wanted to run something similar at home. The ISP used SCO, which is fairly ironic in retrospect, so home use really wasn’t possible due to the high cost and licensing restrictions of the product. Searching for an alternative quickly lead me to Linux.

I purchased *The Linux Bible* [3] from a local bookstore, so my first distribution was Yggdrasil [4]. Although the last official release of Yggdrasil was in 1995, it was a popular option early on and ended up being the first Linux distribution available as a live CD. I’ve used Linux as my main operating system ever since. I like to tinker and understand how things work, so the fact that I could get an operating system that allowed me not only to see how things worked, but also to modify how things worked, enthralled me.

Fast forward to 2000, and I had just started my first job that was dedicated solely to working with Linux. I had been using Linux for a while at this point and wanted to give something back to a community that I felt had given me quite a bit. The first post on LinuxQuestions.org,

a welcome message, was June 25, 2000.

The initial LinuxQuestions.org site consisted of a logo crudely designed by me, a forum, and a short-lived news portal that was based on Slash [5], which was the Perl-based code running Slashdot at the time.

In the beginning, LinuxQuestions.org was basically just me answering questions posted by other members. I figured someday the site would grow to maybe a few hundred people, so to say it has grown far beyond my initial expectations is a monumental understatement. I still fondly remember the first time a member I didn’t personally know answered a question. Today we have more than 500,000 members and millions of posts.

History highlights

As for the history and progression of the site, so much comes to mind that it’s difficult to pick just a few moments, but here are highlights:

Moderators

A year in, we added our first moderator, which was a big step in growing the site and I remember being fairly nervous about it working out. We now have a team of more than 20 moderators who are spread out around the world. They’re responsible for everything from cleaning up spam, to enforcing the community guidelines, to ensuring new members

feel welcome. The amount of dedication and hard work they put into LinuxQuestions.org is truly remarkable, and we quite simply would not be the friendly, welcoming, vibrant place we are today without this group.

Figure 1: LinuxQuestions.org original logo



Members Choice Awards

On a bit of a whim, I started the Members Choice Awards a year later, and it's been fun to watch them grow as an annual event from there. The MCAs allow the community to vote on their favorite projects/products in a variety of categories, some of which change each year. Many projects really get into it, and it's nice to be able to not only recognize the winners, but also gain additional exposure for great open source projects in the process.

There were 11 categories the first year, and winners included Red Hat Linux, KDE, Enlightenment, Quake III, StarOffice, and MySQL.

In 2015, we had 35 categories and winners in the same categories were Linux Mint, KDE (GNOME has won during the interim), Openbox, 0 A.D., LibreOffice, and MariaDB. (Voting for the 16th Members Choice Awards will open soon. Follow us on Twitter [@linuxquestions](#) [6] to be notified when the polls open.)

Linux distributions forum

In 2002, the first distribution started officially participating. Although Linux From Scratch [7] led the way, we now have more than 30 distributions in our official Distributions Forum [8] program, and all distributions are welcome. If you're associated with a distribution that would like to participate, contact us for more information; there is no cost and the requirements are minimal. I think having participation from such a diverse group of people helps create the unique atmosphere we have.

First conference booth

We got a chance to exhibit at LinuxWorld in New York a few years in, and the feedback we got was really energizing. For those who don't remember LinuxWorld, it was one of the first large events focused on Linux and Open Source, with attendance topping 20,000 at its peak. We had moderators fly in from multiple countries, and people from all over the world visited our booth to tell us how much they liked the site and how much it had helped them, or who just stopped by to say hello because they wanted to meet us. Linus Torvalds even stopped by each booth in the .org Pavilion. It was a humbling experience, and one that has happened many times since. If you'd have told me when I founded the site that I'd have experiences such as that one, I'd certainly not have believed you.

Podcast launches

In 2004 we launched a podcast and it really changed how members interacted with the site. Although we no longer produce the LQ Podcast [9] or LQ Radio [10], the archives are still available and it's something I consider reviving from time to time.

Membership milestones

The 100,000th member, 1-millionth thread, and 5-millionth post milestones all stick out as memorable. The numbers involved are so far beyond my initial expectations that it's difficult to articulate. We even had a contest in which the member who guessed the correct date and time the 100,000th member would register won a gratis LinuxQuestions.org shirt. The correct answer ended up being March 13, 2004 between 7-8pm EST.

Community

With all that said, I think the best part about LinuxQuestions.org for me is hearing members say how much they enjoy participating, or getting a message from someone saying they'd have given up on Linux if the site and community didn't exist.

In the end, it's really about community. The open source world is full of smart, energetic, talented, people, and I'm absolutely a better person for having been exposed to it. I've also made quite a few good friends along the way.

Although running such a large community can be challenging at times, it's extremely rewarding and I'm both honored and humbled that so many people have chosen to take time out of their lives to participate and have allowed us to be part of the Linux ecosystem for so long. We'll continue to attempt to improve each and every day, so if you have any feedback for us, please let me know.

Resources

- [1] <https://opensource.com/tags/queue-column>
- [2] <http://www.linuxquestions.org/>
- [3] <http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118999878.html>
- [4] https://en.wikipedia.org/wiki/Yggdrasil_Linux/GNU/X
- [5] <http://www.slashcode.com/www.slashcode.com/>
- [6] <https://twitter.com/linuxquestions>
- [7] <http://linuxfromscratch.org/>
- [8] <http://www.linuxquestions.org/questions/linux-distributions-5/>
- [9] <http://radio.linuxquestions.org/linux/lq-podcasts>
- [10] <http://radio.linuxquestions.org/linux/lq-radio>

Would you like to write for us?

Our editorial calendar includes upcoming themes, community columns, and topic suggestions: <https://opensource.com/calendar>

Happy Pi Day!

To celebrate Pi Day, we're rounding up a series on the Raspberry Pi. What projects have you created? What solutions to common problems have you found? What do you do with your Raspberry Pi?

Containers

How are you or your organization using Linux containers to get work done, to push innovation forward, and to find new solutions to technical problems?

Open Hardware and DIY

Show off your tutorials and demos of hardware in the wild, and tell us about projects you work on and how you use open hardware. Let's see those DIY projects that automate your appliances and up your geek fashion cred.

Entertainment and Geek Culture

We're looking for geek culture stories and articles about how open source tools, projects, and communities keep us entertained.

Back to School

Which open source tools are helpful for the classroom? How are open source technologies being used or taught in your schools? We're always excited to hear how open source is improving education, so send us your stories.

Programming

Show off your scripts, tips for getting started, tricks for developers, and tutorials, and tell us about your favorite programming languages and communities.

Supercomputing

We want to hear about your high-performance computing projects and big data discoveries. Send us your stories!

Email story proposals to open@opensource.com