Best of a decade on Opensource.com 2010–2019

In celebration of our 10-year anniversary

Opensource.com/yearbook

FROM THE EDITOR

Dear reader,

As we celebrate 10 years of publishing, our focus is on the people from all over the globe, in various roles, from diverse backgrounds, who have helped us explore the multitude of ways in which open source can improve our lives—from technology and programming to farming and design, and so much more.

We are celebrating *you* because we've learned that growing this unique storytelling site demands that we do one thing better than all the rest: listen to and talk with our readers and writers.

Over the years, we've gotten better at it. We regularly hold meetings where we review how articles performed with readers from the week before and discuss why we think that's so. We brainstorm and pitch new and exciting article ideas to our writer community on a weekly basis. And we build and nurture close relationships with many writers who publish articles for us every month.

As an editor, I never would have imagined my biggest responsibility would be community management and relationship building over copy editing and calendar planning. I'm so grateful for this because it's made being a part of Opensource.com a deeply rewarding experience.

In December, we closed out a decade of publishing by reaching a new, all-time record of over 2 million reads and over 1 million readers. For us, this validates and affirms the value we've learned to place on relationships with people in a world swirling with metrics and trends.

We truly believe that openness is a better way to work, so I'd like to share some of best practices:

- If anyone out there in the big, wide world has something to share about how they use open source, got started with open source, learned best practices to get a job done or operate in a community (and the list goes on...), they can share it with us. We have hundreds of writers and writers-to-be who come to our door and generate thousands of published articles each year.
- Our lean-mean editorial team reviews and responds to all submissions, provides professional editorial services, nurtures each of our writer communities, and attends conferences and events around the world. Our goal is to care for the articles that come in to us, support and encourage our writers, and meet and know the open source communities that are important to our readers.
- The rest is all about learning and open lines of communication. What can we learn from the response we see from our readers? How can we best deliver the guides, tutorials, and stories we have to share with them? Who are the writers and writers-to-be coming to our door and what can we learn about them? How can we inspire and encourage them on their journeys?

"The Opensource.com community is open, kind, and inquisitive. Each person who writes for us brings their own individual experiences and expertise to the table. When we launched the first version of the Correspondent program in 2013—three years after our first article went live on January 25, 2010—we began with a handful of dedicated individuals. Over the past five years, that group has grown and become the heartbeat of our contributor community. They provide us with feedback on what's going on in the world of open source, they reach out to leaders and members of open source communities and projects, and they pen articles that truly change lives," says Jason Hibbets, former community manager for Opensource.com.

As we celebrate a decade, we invite you to share with us. How do *you* meet the challenges of your life with open source? What could you contribute to the conversation?

— Jen Wike Huger
Chief Editor

CONTENTS

THROUGH THE YEARS

STAND-OUT ARTICLES OF THE YEAR

Welcome to the conversation on Opensource.com

BY JIM WHITEHURST

AS THE CEO OF RED HAT, this is a day I've been looking forward to for quite some time. In my travels, I often find myself talking to people from all walks of life who see opportunities for the lessons of open source to be applied broadly to the world around us.

At Red Hat, we've used open source principles as the backbone of a successful technology company. We know there are opportunities to apply the open source way broadly in business, in government, in education, in the law, and throughout our lives.

This site is one of the ways in which Red Hat gives something back to the open source community. Our desire is to create a connection point for conversations about the broader impact that open source can have—and is having—even beyond the software world.

We think of this site as a "Red Hat community service." Meaning: all ideas are welcome, and all participants are welcome.

opensource.com is where we explore
what happens when the open source way
is applied to the world

This will not be a site for Red Hat, about Red Hat. Instead this will be a site for open source, about the future.

What you see today is only a beginning. In the spirit of open source, we are releasing the site early. We will grow its functionality and content over time based on ideas we create together [1].

With your help, this will be a place that connects people, creates dialog, and—if we dare to dream—maybe even changes the world a little bit for the better.

Good to have you here. Let's get started.

Links

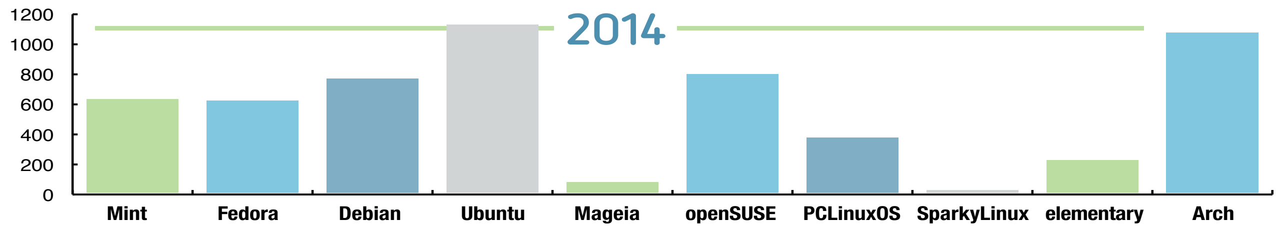
[1] https://opensource.com/should-be/10/1/what-should-opensourcecom-be

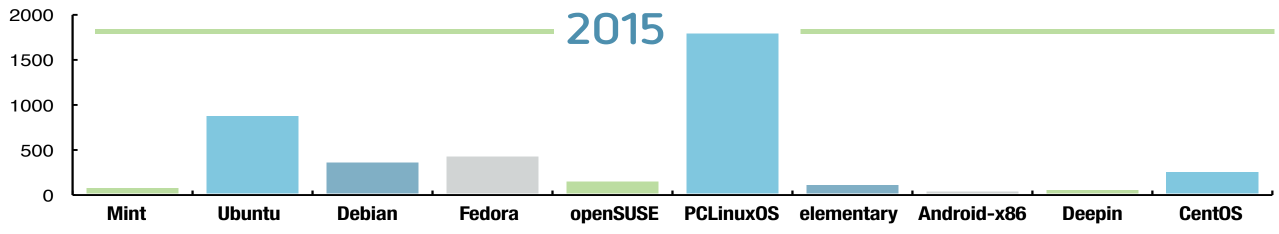
[2] https://youtu.be/J-63D6h0e38

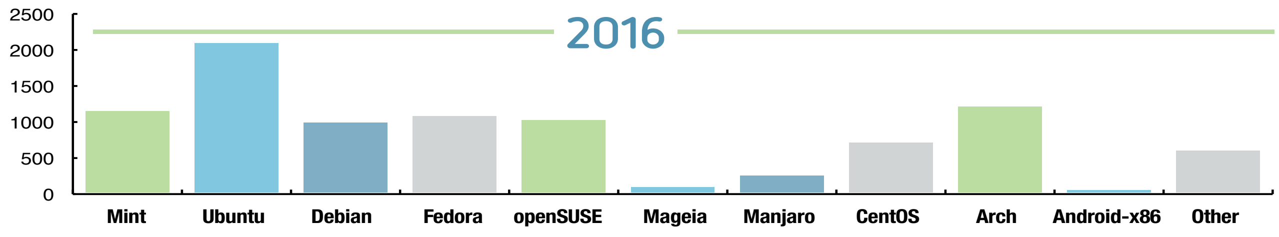
Author

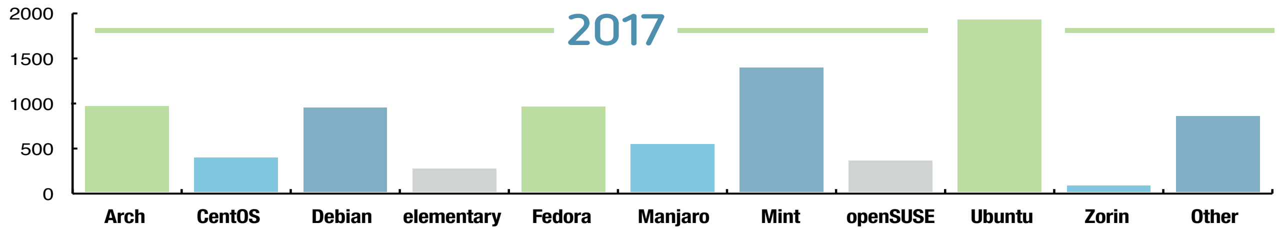
Jim Whitehurst is President and Chief Executive Officer of Red Hat, the world's leading provider of open source enterprise IT products and services. With a background in business development, finance, and global operations, Whitehurst has proven expertise in helping companies flourish—even in the most challenging economic and business environments. Since joining Red Hat in 2008, Whitehurst has grown the company, and its influence on a variety of industries, by reaching key milestones—the most notable in 2012 when Red Hat became the first $1 billion revenue open source software company.

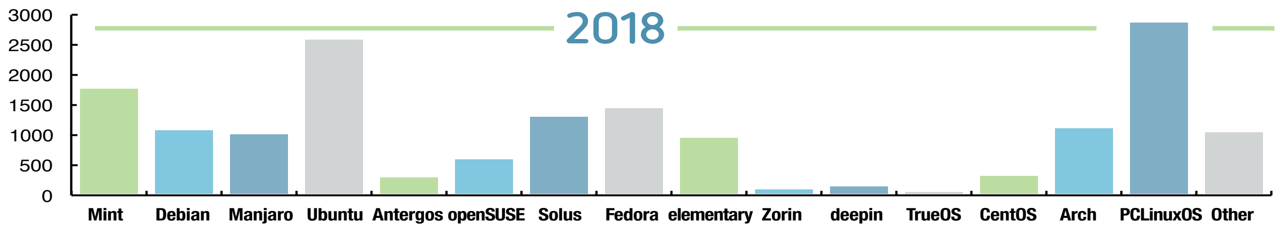
Favorite Linux distributions through the years

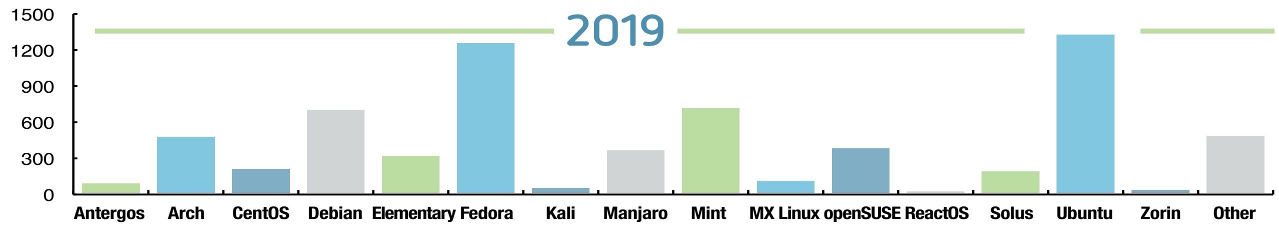












Open source news timeline

- **2013** IBM pledges $1 billion for Linux
- **2013** Motorola announces open hardware phones
- **2013** The Weather Channel goes open source
- **2014** NASA to release source code for software projects
- **2014** GitHub releases Atom text editor under MIT license
- **2014** Google reveals new container orchestration tool, Kubernetes
- **2014** Microsoft open sources more of .NET
- **2015** Rust version 1.0.0 now available
- **2015** Wordpress.com goes open source
- **2016** Perl 6 released
- **2016** Raspberry Pi 3 announced
- **2016** Code.gov shares US government open source code
- **2016** Microsoft joins the Linux Foundation
- **2017** Run Linux on your Samsung phone
- **2017** Linux dominates supercomputing
- **2018** Julia 1.0 released
- **2018** Tim Berners-Lee building a decentralized internet
- **2019** Alibaba launches the fastest open source CPU
- **2019** GNOME and KDE partner on the Linux Desktop
- **2019** System76 will ship 2 Linux laptops with Coreboot-powered open source firmware

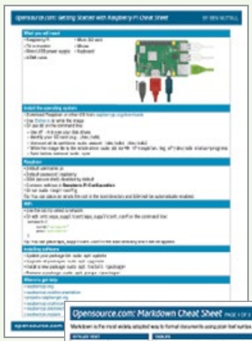
Opensource.com: Open Source Alternatives Cheat Sheet

Use this guide to help you find an open source alternative to common proprietary software tools.

Creative Tools		
Photoshop	GIMP, Pinta, Krita	https://opensource.com/life/12/6/design-without-debt-five-tools-for-designers
Dreamweaver	Aptana Studio, BlueGriffon, NetBeans, SeaMonkey, Aloha Editor	https://opensource.com/alternatives/dreamweaver
AutoCAD	BRL-CAD, FreeCAD, LibreCAD	https://opensource.com/alternatives/autocad
Video Editing	Pitivi, OpenShot, Cinelerra, KDEnlive, Blender	https://opensource.com/article/18/4/new-state-video-editing-linux
Microsoft Publisher	Scribus, LibreOffice, LaTeX or other markup languages	https://opensource.com/alternatives/microsoft-publisher
Chat Tools		
Slack	IRC, Let's Chat, Mattermost, Rocket.Chat, Riot.im	https://opensource.com/alternatives/slack
WhatsApp	Line, Riot.im, Signal, Threema, Viber, Wire,	https://opensource.com/article/19/3/open-messenger-client
Skype	Jami, Jitsi, Linphone, Riot, Wire, Pidgin	https://opensource.com/alternatives/skype
Office Tools		
Microsoft Access	LibreOffice Base, DB Browser for SQLite, Kexi, nuBuilder Forte	https://opensource.com/alternatives/access
Gmail	Roundcube, Zimbra, SquirrelMail, Rainloop, Cypht	https://opensource.com/alternatives/gmail https://opensource.com/article/19/1/productivity-tool-cypht-email
Google Docs	CryptPad	https://opensource.com/article/19/1/productivity-tool-cryptpad
Google Sheets	EtherCalc	https://opensource.com/article/19/7/get-going-ethercalc
Project Management Tools		
Project Management Tools	MyCollab, Odoo, Taiga, Phabricator, Tuleap Open ALM, Agilefant, Redmine, Gitlab, OpenProject, LibrePlan, ProjectLibre	https://opensource.com/business/16/3/top-project-management-tools-2016
Trello	Taiga, Kanboard, Wekan, Restyaboard, Taskboard	https://opensource.com/alternatives/trello
Notetaking Tools		
Evernote	Joplin, Turtl, Paperwork, Laverna, Permanote, Brainstorm	https://opensource.com/article/17/12/joplin-open-source-evernote-alternative https://opensource.com/article/17/12/using-turtl-open-source-alternative-evernote https://opensource.com/alternatives/evernote
News and Social Media		
Pocket	Wallabag	https://opensource.com/article/18/7/wallabag
Social Media	Mastodon, Textile, PixelFed	https://opensource.com/article/19/1/open-source-social-media-alternatives
Filesharing		
Dropbox	ownCloud, NextCloud, Seafile, OnionShare, Pydio Cells	https://opensource.com/alternatives/dropbox
CRM Tools		
Salesforce	Corteza	https://opensource.com/article/19/8/corteza-open-source-alternative-salesforce
CRM Tools	EspoCRM, SuiteCRM, Oro CRM, CiviCRM, Fat Free CRM, Zurmo	https://opensource.com/business/16/2/top-6-open-source-crm-tools-2016
Security		
LastPass	Bitwarden	https://opensource.com/article/18/3/managing-passwords-bitwarden https://opensource.com/life/14/7/managing-passwords-open-source-way
Personal Finance		
Personal Finance Tools	GnuCash, HomeBank, KMyMoney, Money Manager Ex, Skrooge, Spreadsheets	https://opensource.com/life/17/10/personal-finance-tools-linux
Gaming		
Minecraft, Steam	Lutris, Minetest, Terasology, Voxel.js, TrueCraft, Craft	https://opensource.com/alternatives/minecraft https://opensource.com/article/18/10/lutris-open-gaming-platform
More		
More open source alternatives		https://opensource.com/tags/alternatives https://opensource.com/alternatives

Open Source Cheat Sheets

Visit our cheat sheets collection
(https://opensource.com/downloads/cheat-sheets)
for free downloads, including:


Opensource.com: Getting Started with Raspberry Pi Cheat Sheet

Raspberry Pi: See what you need to boot your Pi, how to install the operating system, how to enable SSH and connect to WiFi, how to install software and update your system, and links for where to get further help.

Opensource.com: Python 3.7 Beginner's Cheat Sheet

Python 3.7: This cheat sheet rounds up a few built-in pieces to get new Python programmers started.

Opensource.com: Linux Common Commands Cheat Sheet

Opensource.com: Markdown Cheat Sheet

Linux Common Commands: Keep common tasks top of mind with this handy Linux commands cheat sheet.

Markdown: With this cheat sheet, you'll find yourself ready to write advanced READMEs on GitLab and GitHub.

Opensource.com: Bash Cheat Sheet

Opensource.com: Advanced SSH Cheat Sheet

Bash: Bash tricks and shortcuts to help you become more efficient at the command line.

SSH: Most people know SSH as a tool for remote login, which it is, but it can be used in many other ways.

Opensource.com: A Sysadmin's guide to Networking Cheat Sheet

Linux Networking: In this downloadable PDF cheat sheet, get a list of Linux utilities and commands for managing servers and networks.

Opensource.com

Getting started with Ansible

The open source guide to Ansible

Ansible Automation for SysAdmins: This guide is a primer to help you get started using Ansible to give you a better understanding of the capabilities and show you how to automate everyday sysadmin tasks.

Opensource.com

Getting started with Kubernetes

Find out how Kubernetes is an elegant solution to a wide range of essential business problems.

Getting Started with Kubernetes: Like Kubernetes, dump trucks are an elegant solution to a wide range of essential business problems. Download this e-book to learn more.

OPENSOURCE.COM & THE ENTERPRISERS PROJECT

DevOps Hiring Guide

DevOps Hiring Guide: This free download provides advice, tactics, and information about the state of DevOps hiring for both job seekers and hiring managers.

How I coined the term ‘open source’

BY CHRISTINE PETERSON

Christine Peterson finally publishes her account of that fateful day, 20 years ago.

IN A FEW DAYS, on February 3, the 20th anniversary of the introduction of the term “open source software” is upon us. As open source software grows in popularity and powers some of the most robust and important innovations of our time, we reflect on its rise to prominence.

I am the originator of the term “open source software” [1] and came up with it while executive director at Foresight Institute. Not a software developer like the rest, I thank Linux programmer Todd Anderson for supporting the term and proposing it to the group.

This is my account of how I came up with it, how it was proposed, and the subsequent reactions. Of course, there are a number of accounts of the coining of the term, for example by Eric Raymond and Richard Stallman, yet this is mine, written on January 2, 2006.

It has never been published, until today.

The introduction of the term “open source software” was a deliberate effort to make this field of endeavor more understandable to newcomers and to business, which was viewed as necessary to its spread to a broader community of users. The problem with the main earlier label, “free software,” was not its political connotations, but that—to newcomers—its seeming focus on price is distracting. A term was needed that focuses on the key issue of source code and that does not immediately confuse those new to the concept. The first term that came along at the right time and fulfilled these requirements was rapidly adopted: open source.

This term had long been used in an “intelligence” (i.e., spying) context, but to my knowledge, use of the term with respect to software prior to 1998 has not been confirmed. The account below describes how the term open source software [2] caught on and became the name of both an industry and a movement.

Meetings on computer security

In late 1997, weekly meetings were being held at Foresight Institute to discuss computer security. Foresight is a nonprofit think tank focused on nanotechnology and artificial intelligence, and software security is regarded as central to the reliability and security of both. We had identified free software as a promising approach to improving software security and reliability and were looking for ways to promote it. Interest in free software was starting to grow outside the programming community, and it was increasingly clear that an opportunity was coming to change the world. However, just how to do this was unclear, and we were groping for strategies.

At these meetings, we discussed the need for a new term due to the confusion factor. The argument was as follows: those new to the term “free software” assume it is referring to the price. Oldtimers must then launch into an explanation, usually given as follows: “We mean free as in freedom, not free as in beer.” At this point, a discussion on software has turned into one about the price of an alcoholic beverage. The problem was not that explaining the meaning is impossible—the problem was that the name for an important idea should not be so confusing to newcomers. A clearer term was needed. No political issues were raised regarding the free software term; the issue was its lack of clarity to those new to the concept.

Releasing Netscape

On February 2, 1998, Eric Raymond arrived on a visit to work with Netscape on the plan to release the browser code under a free-software-style license. We held a meeting that night at Foresight's office in Los Altos to strategize and refine our message. In addition to Eric and me, active participants included Brian Behlendorf, Michael Tiemann, Todd Anderson, Mark S. Miller, and Ka-Ping Yee. But at that meeting, the field was still described as free software or, by Brian, "source code available" software.

While in town, Eric used Foresight as a base of operations. At one point during his visit, he was called to the phone to talk with a couple of Netscape legal and/or marketing staff. When he was finished, I asked to be put on the phone with them—one man and one woman, perhaps Mitchell Baker—so I could bring up the need for a new term. They agreed in principle immediately, but no specific term was agreed upon.

Between meetings that week, I was still focused on the need for a better name and came up with the term "open source software." While not ideal, it struck me as good enough. I ran it by at least four others: Eric Drexler, Mark Miller, and Todd Anderson liked it, while a friend in marketing and public relations felt the term "open" had been overused and abused and believed we could do better. He was right in theory; however, I didn't have a better idea, so I thought I would try to go ahead and introduce it. In hindsight, I should have simply proposed it to Eric Raymond, but I didn't know him well at the time, so I took an indirect strategy instead.

Todd had agreed strongly about the need for a new term and offered to assist in getting the term introduced. This was helpful because, as a non-programmer, my influence within the free software community was weak. My work in nanotechnology education at Foresight was a plus, but not enough for me to be taken very seriously on free software questions. As a Linux programmer, Todd would be listened to more closely.

The key meeting

Later that week, on February 5, 1998, a group was assembled at VA Research to brainstorm on strategy. Attending—in addition to Eric Raymond, Todd, and me—were Larry Augustin, Sam Ockman, and attending by phone, Jon "maddog" Hall.

The primary topic was promotion strategy, especially which companies to approach. I said little, but was looking for an opportunity to introduce the proposed term. I felt that it wouldn't work for me to just blurt out, "All you technical people should start using my new term." Most of those attending didn't know me, and for all I knew, they might not even agree that a new term was greatly needed, or even somewhat desirable.

Fortunately, Todd was on the ball. Instead of making an assertion that the community should use this specific new term, he did something less directive—a smart thing to do with this community of strong-willed individuals. He simply used the term in a sentence on another topic—just dropped it into the conversation to see what happened. I went on alert, hoping for a response, but there was none at first. The discussion continued on the original topic. It seemed only he and I had noticed the usage.

Not so—memetic evolution was in action. A few minutes later, one of the others used the term, evidently without noticing, still discussing a topic other than terminology. Todd and I looked at each other out of the corners of our eyes to check: yes, we had both noticed what happened. I was excited—it might work! But I kept quiet: I still had low status in this group. Probably some were wondering why Eric had invited me at all.

Toward the end of the meeting, the question of terminology [3] was brought up explicitly, probably by Todd or Eric. Maddog mentioned "freely distributable" as an earlier term, and "cooperatively developed" as a newer term. Eric listed "free software," "open source," and "sourceware" as the main options. Todd advocated the "open source" model, and Eric endorsed this. I didn't say much, letting Todd and Eric pull the (loose, informal) consensus together around the open source name. It was clear that to most of those at the meeting, the name change was not the most important thing discussed there; a relatively minor issue. Only about 10% of my notes from this meeting are on the terminology question.

But I was elated. These were some key leaders in the community, and they liked the new name, or at least didn't object. This was a very good sign. There was probably not much more I could do to help; Eric Raymond was far better positioned to spread the new meme, and he did. Bruce Perens signed on to the effort immediately, helping set up Opensource.org and playing a key role in spreading the new term.

For the name to succeed, it was necessary, or at least highly desirable, that Tim O'Reilly agree and actively use it in his many projects on behalf of the community. Also helpful would be use of the term in the upcoming official release of the Netscape Navigator code. By late February, both O'Reilly & Associates and Netscape had started to use the term.

Getting the name out

After this, there was a period during which the term was promoted by Eric Raymond to the media, by Tim O'Reilly to business, and by both to the programming community. It seemed to spread very quickly.

On April 7, 1998, Tim O'Reilly held a meeting of key leaders in the field. Announced in advance as the first "Freeware Summit," [4] by April 14 it was referred to as the first "Open Source Summit." [5]

These months were extremely exciting for open source. Every week, it seemed, a new company announced plans to participate. Reading Slashdot became a necessity, even for those like me who were only peripherally involved. I strongly believe that the new term was helpful in enabling this rapid spread into business, which then enabled wider use by the public.

A quick Google search indicates that “open source” appears more often than “free software,” but there still is substantial use of the free software term, which remains useful and should be included when communicating with audiences who prefer it.

A happy twinge

When an early account [6] of the terminology change written by Eric Raymond was posted on the Open Source Initiative website, I was listed as being at the VA brainstorming meeting, but not as the originator of the term. This was my own fault; I had neglected to tell Eric the details. My impulse was to let it pass and stay in the background, but Todd felt otherwise. He suggested to me that one day I would be glad to be known as the person who coined the name “open source software.” He explained the situation to Eric, who promptly updated his site.

Coming up with a phrase is a small contribution, but I admit to being grateful to those who remember to credit me with it. Every time I hear it, which is very often now, it gives me a little happy twinge.

The big credit for persuading the community goes to Eric Raymond and Tim O’Reilly, who made it happen. Thanks to them for crediting me, and to Todd Anderson for his role throughout. The above is not a complete account of open source history; apologies to the many key players whose names do not appear. Those seeking a more complete account should refer to the links in this article and elsewhere on the net.

Links

[1] https://opensource.com/resources/what-open-source
[2] https://opensource.org/osd
[3] https://wiki2.org/en/Alternative_terms_for_free_software
[4] https://www.oreilly.com/pub/pr/636
[5] https://www.oreilly.com/pub/pr/796
[6] https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Alternative_terms_for_free_software.html

Author

Christine Peterson writes, lectures, and briefs the media on coming powerful technologies, especially nanotechnology, artificial intelligence, and longevity. She is Cofounder and Past President of Foresight Institute, the leading nanotech public interest group. Foresight educates the public, technical community, and policymakers on coming powerful technologies and how to guide their long-term impact.

She serves on the Advisory Board of the Machine Intelligence Research Institute, and has served on California’s Blue Ribbon Task Force on Nanotechnology and the Editorial Advisory Board of NASA’s Nanotech Briefs.

She has often directed Foresight Conferences on Molecular Nanotechnology, organized Foresight Institute Feynman Prizes, and chaired Foresight Vision Weekends.

She lectures on technology topics to a wide variety of audiences, focusing on making complex fields understandable.

Her work is motivated by a desire to help Earth’s environment and traditional human communities avoid harm and instead benefit from expected dramatic advances in technology. This goal of spreading benefits led to an interest in new varieties of intellectual property including open source software, a term she is credited with originating.

Wearing her for-profit hat, she chairs the Personalized Life Extension Conference series. In 1991 she coauthored Unbounding the Future: the Nanotechnology Revolution (Morrow, full text online), which sketches nanotechnology’s potential environmental and medical benefits as well as possible abuses. An interest in group process led to coauthoring Leaping the Abyss: Putting Group Genius to Work (knOwhere Press, 1997, full text online) with Gayle Pergamit.

Peterson holds a bachelor’s degree in chemistry from MIT.

Follow her at @lifeext

Cooperative success: Understanding the co-op business model

BY REBECCA FERNANDEZ

THERE ARE A FEW THINGS you should know about democratically run "cooperative" businesses. First, they're not all that unusual. They're also respectably profitable. And working in one doesn't require you to be a Marxist or wear patchouli.

Something old, something new

Last December, the United Nations declared 2012 to be "The International Year of Cooperatives," encouraging countries around the world to promote the establishment of these types of businesses. And with good reason.

From Spain's seventh-largest business, Mondragon Corporation [1], a federation of worker cooperatives that employs over 90,000 people, to Wisconsin's Isthmus Engineering [2], with its 29 co-owners, cooperative businesses are thriving. Arguably the most well known, Mondragon is now 54 years old and has outperformed private sector companies on employee compensation and, during several bleak recession years, unemployment rates.

Still, when people hear the word "cooperative," most have difficulty envisioning anything beyond a crunchy granola health food store. That's perhaps not surprising, considering the closest thing to employee-ownership most corporate workers experience is an Employee Stock Ownership Plan, where they can purchase (or are granted) stock in their company. These programs are designed to incentivize workers to be more productive, as they now benefit directly from the business's success. They are built around the assumption that monetary reward is the best motivation for workers.

Cooperative businesses take this idea of employee ownership and engagement one step further: *employees actually own and operate the business*. Besides the profit motive, cooperative models assume that workers value a more humanized workplace, where the input of ground-level employees can improve the business. Most cooperative businesses are founded on the democratic principle of "one worker, one vote," and no two are alike.

The many faces of the cooperative

Isthmus Engineering employs around 45 people, who are eligible to become worker-owners of the company after two years of service. According to controls engineer Ole Olson, worker-owners "get to decide how our company runs, what we build, what we don't build." The waiting period gives employees time to experience the business model, get acclimated to the environment, and understand the corporate culture.

Nearby Union Cab [3] operates a bit differently, creating new owners almost immediately, as long-time member Fred Schepartz explained to The Workers' Paradise [4]:

> "All employees who pass probation are members of Union Cab Cooperative. Period. [...] There is no caste system. Structurally, there are no members that are more equal than others. Yes, we have managers, but they have to answer to the board of directors, which is elected from the membership, by the membership. Essentially,

> management works for the employees though they are given the authority to do their jobs. [E]verybody who works at Union Cab who has passed probation is a full-fledged member. Drivers, dispatchers, phone answerers, mechanics, IT staff, accounting staff. Everybody."

Meanwhile, New York City's Cooperative Home Care Associates [5], which provides services like light housekeeping and personal care to a number of elderly and disabled individuals, focuses on transforming what are traditionally low income, entry-level, part time jobs [6] into full time positions with training, full benefits, higher wages, and guaranteed hours. The company offers a number of special services tailored to the needs of these workers, including no-interest loans up to $250, free tax preparation, and savings programs toward the $1,000 required to become a worker-owner. Meanwhile, operating decisions are made by a 14-member Board of Directors, eight being elected from among all worker-owners. The Worker Council, composed of 12 home care workers, serves as a liason between the Board of Directors and the employees, explaining decisions and advocating for needs. Each worker-owner votes to approve or deny the annual allocation of net profit determined by the Board.

There are many more stories of profitable cooperatives, from bakeries to retailers. There are also a number of organizing bodies, including the US Federation of Worker Cooperatives [7], National Cooperative Business Organization [8], and the International Co-operative Alliance [9].

Learning from success

Even Pittsburgh's United Steel Workers Union is getting in on the (cooperative) action. The union is partnering with Mondragon to explore the possibility of steel worker cooperatives. With over 40,000 manufacturing facilities closed throughout the United States during the economic recession, exploring new business models probably isn't a bad idea.

Retired steelworker Rick Kimbrough told SolidarityEconomy.net that he's all for the initiative. "Ever since they shut down our mill, I've always thought, 'Why shouldn't we own them?' If we did, they wouldn't be running away."

Links

[1] https://opensource.com/business/10/7/cooperative-success-understanding-co-op-business-model
[2] https://madison.com/wsj/business/article_6b218ae8-b911-11de-a44a-001cc4c03286.html
[3] https://www.unioncab.com/
[4] https://www.givemethisoffer.com/wim/static/wi/main3.html?tp=iw&cid=8502&v=23&gnum=6&clickid=77661227104&cachecode=HAyZFRX6td7Y%2F8fqSvcrnA%3D%3D%3AZmVkY2JhOTg3NjU0MzIxMA%3D%3D&q=cooperativeconsult+KW+cooperativeconsult.com+cooperative+consultants&dkw=cooperativeconsult.com&g=US&cc2=HAyZFRX6td7Y%2F8fqSvcrnA%3D%3D%3AZmVkY2JhOTg3NjU0MzIxMA%3D%3D&geo=US
[5] http://www.chcany.org/
[6] https://geo.coop/node/433
[7] http://www.usworker.coop/home/
[8] https://ncbaclusa.coop/
[9] https://www.ica.coop/en

Author

Rebecca Fernandez is a Principal Program Manager at Red Hat, leading projects to help the company scale its open culture. She's an Open Organization Ambassador, contributed to The Open Organization book, and maintains the Open Decision Framework. She is interested in the intersection of open source principles and practices, and how they can transform organizations for the better.

The DRM graveyard: A brief history of digital rights management in music

BY RUTH SUEHLE

THERE ARE MORE than a few reasons digital rights management (DRM) has been largely unsuccessful. But the easiest way to explain to a consumer why DRM doesn't work is to put it in terms he understands: "What happens to the music you paid for if that company changes its mind?" It was one thing when it was a theoretical question. Now it's a historical one. Rhapsody just had the next in a line of DRM music services to go—this week the company told its users than anyone with RAX files has unil November 7 to back them up in another format or lose them the next time they upgrade their systems.

The Electronic Frontier Foundation summarizes the battle [1]:

> Corporations claim that DRM is necessary to fight copyright infringement online and keep consumers safe from viruses. But there's no evidence that DRM helps fight either of those. Instead DRM helps big business stifle innovation and competition by making it easy to quash "unauthorized" uses of media and technology.

Unfortunately, the side effect in this less-than-successful attempt to fight piracy is the hours it takes users to retrieve, rip, and back up their music when a services shuts down, is sold, or simply decides DRM wasn't the right way to go (sometimes in as little as five months). The following is a brief history of the rise and fall of DRM in music services.

October 1998

The Digital Millennium Copyright Act makes DRM circumvention and circumvention tools illegal.

December 2001

Rhapsody unlimited music streaming subscription service launches with songs restricted by the company's Helix DRM.

October 2001

"Beale Screamer" cracks the Microsoft Windows Media DRM and posts a how-to on the sci-crypt Usenet board along with code for stripping the DRM from Windows Media files. In his message, he writes to music companies, "Give us more options, not fewer. If you try to take away our current rights, and dictate to us what we may or may not do, you're going to get a lot of resistance." To users he writes, "Please respect the uses I have intended this software for. I want to make a point with this software, and if you use it for purposes of violating copyrights, the message stands a very good chance of getting lost."

May 2002

Shuman Ghosemajumder proposes the Open Music Model [2], which states that subscription services free of DRM are the only successful model to beat piracy. It requires open file sharing, open file formats, open membership, open payment, and open competition.

April 21, 2003

RealNetworks (known for RealAudio, RealVideo and RealPlayer) acquires Listen.com, owner of Rhapsody and offers streaming downloads for a monthly fee.

April 28, 2003

One week later, the iTunes store launches with its songs encrypted with FairPlay DRM. It restricts users to accessing songs from only three (later five) computers and making no more than ten (later seven) copies of a CD playlist. Apple does not license its encryption, so only Apple devices can play iTunes music.

November 2003

FairPlay is cracked by Jon Lech Johansen ("DVD Jon"), previously known for his part in the DeCSS software, which was released four years earlier for decrypting DVDs.

January 2004

RealNetworks announces sale of DRM-restricted music in the RealPlayer Music Store.

August 2004

Microsoft begins certifying devices and providers with the PlaysForSure mark, noting that they had been tested and certified for compatibility with files encrypted with Windows Media DRM.

February 2005

Yahoo! Music offers unlimited music as a rebrand of LAUNCH Media at the Open Music Model's recommended $5 subscription price point, but using DRM.

October/November 2005

Consumers of Sony CDs discover the Sony rootkit in its SecuROM DRM. Removing it leaves some forced to reinstall Windows [3]. Sony settles in December. (Read a timeline of the Sony rootkit story [4].)

July 2006

The eMusic subscription service, which sells songs DRM-free, becomes the second-largest digital music service, though with an 11% market share to iTunes' 67%.

September 2006

Steve Jobs announces that Apple has 88% of the legal US music download market—still locked under DRM.

November 2006

Microsoft abandons the PlaysForSure strategy in favor of a more Apple-esque approach with the Zune player tightly tied to the Zune Marketplace. PlaysForSure music will not play on the Zune.

February 2007

Steve Jobs writes in "Thoughts on Music" [5] that it is the music companies who force Apple to use DRM in iTunes contracts, and he calls on them to relax the demand. "DRMs haven't worked, and may never work, to halt music piracy," he writes.

April 2007

EMI's music library becomes available DRM-free on iTunes for a premium charge through "iTunes Plus."

May 2007

Amazon announces it will sell DRM-free music for 99 cents/song. Shortly thereafter, Apple drops the DRM-free premium price. Customers soon discover that each of these tracks downloaded from iTunes—even the new, DRM-free ones—has the user's personal information embedded [6].

August 2007

Wal-Mart begins offering DRM-restricted mp3 downloads. Nokia Music Stoore launches to provide Nokia phones with an on-phone music store using DRM that allowed music to be played only on the phone.

February 2008

Wal-Mart decides to offer only DRM-free mp3s.

April 2008

Apple becomes the largest music seller in the US, followed by Wal-Mart and Best Buy.

March 2008

Microsoft announces that the MSN Music Store will no longer be supported and users will not be able to play their songs on any computer they do not authorize by August 31—songs definitely no longer "play for sure."

June 2008

Microsoft responds to customer outrage and agrees that MSN Music Store songs will continue to be transferrable through the end of 2011.

September 2008

Yahoo! Music Unlimited shuts down and merges into Rhapsody. It encourages users to burn their music to CDs by the end of the month, as the move to Rhapsody does not include the continued ability to access license keys for purchased music.

Wal-Mart decides to shut down its DRM system, ending support for protected files from the five months when they chose to use it.

January 2009

Apple agrees with the four major music companies that all music sold via iTunes will be sold DRM-free.

April 2009

Apple announces availability of DRM-free versions of all music in the iTunes store (but keeps it on video, audiobooks, and apps).

April 2010

Rhapsody spins off from RealNetworks.

September 2010

Nokia Music Store (Ovi Music) decides to go DRM-free.

November 2011

Rhapsody tells its users that anyone with its older RAX format files has unil November 7 to back them up in another format or lose them the next time they upgrade their systems.

Links

[1] https://www.eff.org/issues/drm
[2] https://en.wikipedia.org/wiki/Open_music_model
[3] https://www.theregister.co.uk/2005/11/01/sony_rootkit_drm/
[4] https://boingboing.net/2005/11/14/sony-anticustomer-te.html
[5] https://macdailynews.com/2007/02/06/apple_ceo_steve_jobs_posts_rare_open_letter_thoughts_on_music/
[6] https://arstechnica.com/gadgets/2007/05/eff-drm-free-itunes-files-carry-more-than-just-names-and-e-mail-addresses/

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.

Three tips for working with open source diagrams

BY MÁIRÍN DUFFY

IF YOU'RE A BIG-TIME OPEN SOURCE FANATIC like me, you probably get questions about open source alternatives to proprietary tools rather frequently. From the 'Alternatives to Microsoft® Visio®' department, here are three tips that should help designers who use Visio in an open source environment. If you need an open source option for opening Visio files, a revived open source application for creating diagrams, or a lesser-known open source tool for converting Visio® stencils, these tips are for you.

1. Opening Visio files

Open source tools haven't been able to open Visio (.VSD format) diagrams for a pretty long time. The format is commonly used for infrastructure and architecture documentation and planning—and less commonly, for user interface design and planning.

Thankfully, Summer of Code students Eilidh McAdam and Fridrich Strba implemented support for opening .VSD format files in LibreOffice Draw last summer. This support is now available in Fedora 17's version of LibreOffice.

If you prefer to use Inkscape or any other open source SVG editor (such as Karbon14 or Xara,) Eilidh McAdam produced libvisio which works with the libwpd tools for converting .VSD files to .SVG—making it possible to open them in any SVG editor in Fedora. For Fedora 17, you'll need the libvisio and libvisio-tools packages that are now available.

These two projects are covered in detail in Libre Graphics World, including full instructions on VSD-to-SVG conversion [1].

Note: There isn't a solution yet for writing out to .VSD format, but you can write out to .PDF, of course, which may serve your needs in sharing your modified and originally .VSD-format diagrams.

2. Creating diagrams in a Visio-like environment

Traditionally, Dia [2] has been the best game in town for open source diagramming, but it hasn't changed much over the years and requires a lot of tweaking and extra work to produce beautiful diagrams. For example, diagram graphics are not anti-aliased by default in Dia .9.7.1.

Inkscape [3] is another option—my preferred one—but as a general vector graphics editing tool, it doesn't have the specialized niceities and workflow of a tool specifically focused on diagramming and takes a bit of experience to use it effectively for creating diagrams.

Calligra Flow [4], which is part of the Calligra suite of productivity applications, is now available in Fedora 17. The lineage of this application can be traced back to the KOffice Kivio [5] diagramming tool, which according to Wikipedia was initially released in October 2000. In mid 2010, the Calligra project [6] split off [7] of the KOffice project; not long after, Flow maintainer Yue Liu got to work [8] improving the stencil system and included stencils.

From my initial experimentations of creating sample diagrams with it, Calligra Flow looks to be a pretty promising diagramming tool, so give it a shot!

3. Converting Visio stencils (.VXD) to .SVG

Converting Visio-format stencils to .SVG involves a pretty kludgy workflow that doesn't always work. Depending on how complex the stencils are, though, it's possible to extract the shapes from a .VXD stencil file and use them as .SVGs. It involves using an old and seemingly abandoned project, so not much is new here, but now that you know how to open .VSD files, it may be of more interest to know how to also open .VXDs files [9].

Remember, this process can be really buggy and may need some love. For example, you have to move the included 'chunks_parse_cmds.tbl' file to the directory you run the command from, in order for it to work. Even then it fails sometimes, but, it's a start!

The format for Calligra Flow stencils is open and it is ODG-based, so with some effort it is possible to manually convert VXDs to Calligra Flow stencils [10], and the project has an open call for help on that front. If you'd like to get involved, it seems like a great place to start. Have fun!

Links

[1] http://libregraphicsworld.org/blog/entry/the-ultimate-quest-of-bringing-visio-support-to-libreoffice
[2] https://wiki.gnome.org/action/show/Apps/Dia?action=show&redirect=Dia
[3] https://inkscape.org/
[4] http://www.calligra.org/flow/
[5] http://www.koffice.org/kivio/
[6] https://www.calligra.org/
[7] https://mail.kde.org/pipermail/calligra-devel/2011-June/001663.html
[8] https://www.calligra.org/news/calligra-suite-the-first-three-months/
[9] http://freshmeat.sourceforge.net/projects/vsdump
[10] https://community.kde.org/Calligra/Build_Stencils_for_Flow

Author
Máirín is a principal interaction designer at Red Hat. She is passionate about software freedom and free & open source tools, particularly in the creative domain: her favorite application is Inkscape.

How one parent fosters open source at home through DIY projects

BY CAROLYN FOX

THIS YEAR I MADE A NEW YEAR RESOLUTION to foster a more open education at home by joining a growing subculture of society. To start, I began replacing some commercial household products, such as toothpaste, with 'open source' ones. After all, there is no patent on or trademark for baking soda (2/3 cup), salt (4 teaspoons), mint oil (1 tablespoon), or melted coconut oil (2-3 tablespoons)—what you need to make homemade toothpaste. They are readily available and accessible, except for the mint oil perhaps (but you can substitute it with cinnamon or vanilla extract, or other possibilities if you just use your creative, open mind).

Once I had the necessary ingredients to make my own toothpaste, I spent less time to make it than I had previously spent in the supermarket checkout queue to buy it. It costs less money too. No fluoride, preservatives, or other chemical substances either.

Nearly everyone uses some kind of toothpaste on a daily basis, but how many of us question the futility of a commercial brand toothpaste for something that we can easily make ourselves and with an open source concept? How many of us would like our children to get excited about math, science, and history or live a simpler, greener, and healthier lifestyle by making their own toothpaste?

Plus, making your own toothpaste or shampoo, or even laundry detergent, doesn't require some special skill. It isn't laborious or time consuming or some mysterious process.

When I first started to find out about the growing do-it-yourself (DIY), or homemade, movement I was a bit intrigued and baffled—that is, until I made an analogy with open source. Then it dawned on me that this movement is an outgrowth or extension of the open source movement, and a way to overcome a crisis in education. And I am far from alone in my thoughts here. In September a technology reporter for the BBC News wrote about how Karl Marx predicted [1] a revolution would come, but Marx probably didn't envision it being a bunch of do-it-yourselfers and members of a homemade movement brigade. Without digital technology, it's true that such efforts would be minimal, but with digital technology, it has taken root.

Time will tell how much this movement will affect the larger open source movement, generate a type of questioning that comes from realizing you can ditch a more expensive commercial product for one you've made yourself, and bubble down to future generations. Already, though, there are kid's sites such as DIY.org [2] that are flourishing. They bargain that children are more involved in digital technology at a younger age than previous generations and bill themselves as a place where kids can build, make, hack, grow, and earn skills—an updated digital version of the Boy Scouts. And there are future plans for a membership subscription and options for children to sell their own inventions (another thing Marx might not have imagined, or Robert Baden-Powell who was founder and Chief Scout of the Scout Movement).

DIY.org is free and open to all children age 7 and older, though it seems to be primarily aimed at boys. Good thing then that making toothpaste (for one) is, by contrast, universal and gender neutral. Perhaps this is an alternative way to connect underrepresented girls with STEM (science, technology, engineering, and mathematics) and open source, since at the moment, digital technology and the type of creative tinkering that DIY tends to foster is still initially focused on boys.

Furthermore, I think girls may be more readily interested in creative tinkering if they were first involved in something like making their own toothpaste. It would foster a sense of empowerment, autonomy, and agency. (Read: Wendy Priesnitz's idea of economic success [3] that questions a tradition of well-being based on economics alone.)

All in all, allowing children to make their own toothpaste not only becomes a principle of economics and self-reliance, but maybe a slightly subversive act too.

Links

[1] http://www.bbc.co.uk/news/technology-19347120
[2] https://diy.org/
[3] http://www.educationrevolution.org/blog/unschooling-and-feminism/

Author
Carolyn Fox is an educator, librarian, historian, and an un/homeschooling mother. She lives in Massachusetts with her UK husband and son.

Top 3 open source business intelligence and reporting tools

BY ROBIN MUILWIJK

THIS ARTICLE REVIEWS three top open source business intelligence and reporting tools. In economies of big data and open data, who do we turn to in order to have our data analysed and presented in a precise and readable format? This list covers those types of tools. The list is not exhaustive—I have selected tools that are widely used and can also meet enterprise requirements. And, this list is not meant to be a comparison—this is a review of what is available.

BIRT

BIRT [1] is part of the open source Eclipse project and was first released in 2004. BIRT is sponsored by Actuate, and recieves contributions from IBM and Innovent Solutions.

BIRT consists of several components. The main components being the Report Designer and BIRT Runtime. BIRT also provides three extra components: a Chart Engine, Chart Designer, and Viewer. With these components you are able to develop and publish reports as a standalone solution. However, with the use of the Design Engine API, which you can include in any Java/Java EE application, you can add reporting features in your own applications. For a full description and overview of it's architecture, see this overview. [2]

The BIRT Report Designer has a rich feature set [3], is robust, and performs well. It scores high in terms of usability with it's intuitive user interface. An important difference with the other tools is the fact it presents reports primarily to web. It lacks a true Report Server, but by using the Viewer on a Java application server, you can provide end users with a web interface to render and view reports.

If you are looking for support, you can either check out the BIRT community [4] or the Developer Center [5] at Actuate. The project also provides extensive documentation [6] and a Wiki [7].

BIRT is licensed under the Eclipse Public License. It's latest release 4.3.2, which runs on Windows, Linux and Mac, can be downloaded here [8]. Current development is shared through it's most recent project plan [9].

JasperReport

TIBCO recently acquired [10] JasperSoft, the company formerly behind JasperReport [11]. JasperReport is the most popular and widely used open source reporting tool. It is used in hundreds of thousands production environments. JasperReport is released as Enterprise and Community editions.

Similar to BIRT, JasperReport consists of several components such as the JasperReport Library, iReport Report Designer, JasperReport Studio, and JasperReport Server. The Library is a library of Java classes and APIs and is the core of JasperReport. iReport Designer and Studio as the report designers where iReport is a Netbeans plugin and standalone client, and Studio an Eclipse plugin. Note: iReport will be discontinued in December 2015, with Studio becoming the main designer component. For a full overview and description of the components, visit the homepage of the JasperReport community [11].

A full feature list of JasperSoft (Studio) can be viewed here [12]. Different from BIRT, JasperReport is using a pixel-perfect approach in viewing and printing it's reports. The ETL, OLAP, and Server components provide JasperReport with valuable functionality in enterprise environments, making it easier to integrate with the IT-architecture of organisations.

JasperReport is supported by excellent documentation [13], a Wiki [14], Q&A forums, and user groups [15].

Based on Java, JasperReport runs on Windows, Linux, and Mac. It's latest release 5.5 is from October 2013, and is licensed under GPL.

Pentaho

Unlike the previous two tools, Pentaho is a complete business intelligene (BI) Suite, covering the gamut from reporting to data mining. The Pentaho BI Suite encompasses several open source projects, of which Pentaho Reporting is one of them.

Like the other tools, Pentaho [16] Reporting has a rich feature set, ready for use in enterprise organisations. From visual report editor to web platform to render and view reports to end users. And report formats like PDF, HTML and more, security and role management, and the ability to email reports to users.

The Pentaho BI suite also contains the Pentaho BI Server. This is a J2EE application which provides an infrastructure to run and view reports through a web-based user interface. Other components from the suite are out of scope for this article. They can be viewed on the site from Pentaho [17], under the Projects menu. Pentaho is released as Enterprise and Community editions.

The Pentaho project provides it's community with a forum, Jira bug tracker, and some other collaboration options. It's documentation can be found on a Wiki [18].

Pentaho runs on Java Enterprise Edition and can be used on Windows, Linux, and Mac. It's latest release is version 5.0.7 from May 2014, and is licensed under GPL.

Summary

All three of these open source business intelligence and reporting tools provide a rich feature set ready for enterprise use. It will be up to the end user to do a thorough comparison and select either of these tools. Major differences can be found in report presentations, with a focus on web or print, or in the availability of a report server. Pentaho distinguishes itself by being more than just a reporting tool, with a full suite of components (data mining and integration).

Links

[1] http://www.eclipse.org/birt/
[2] http://www.eclipse.org/birt/about/architecture.php
[3] http://www.eclipse.org/birt/phoenix/project/notable4.3.php
[4] https://www.eclipse.org/birt/community/
[5] https://www.opentext.com/products-and-solutions/products/ai-and-analytics/analytics-developer-community/?utm_source=ai-analytics&utm_medium=redirect&utm_source=actuatedev&utm_medium=redirect
[6] https://www.eclipse.org/birt/documentation/
[7] https://wiki.eclipse.org/BIRT_Project
[8] https://download.eclipse.org/birt/downloads/
[9] https://www.eclipse.org/birt/about/project-organization/
[10] https://community.jaspersoft.com/founders-message-tibco
[11] https://community.jaspersoft.com/
[12] https://community.jaspersoft.com/wiki/jaspersoft-studio-features
[13] https://community.jaspersoft.com/documentation?version=10870
[14] https://community.jaspersoft.com/wiki/jaspersoft-community-wiki-0
[15] https://www.meetup.com/pro/tibco
[16] https://community.hitachivantara.com/s/article/pentaho-reporting
[17] https://community.hitachivantara.com/s/pentaho
[18] https://wiki.pentaho.com/display/COM/Community+Wiki+Home

Author

Former Opensource.com and Open Organization moderator/ambassador.

Robin writes and is active on social media to promote and advocate for open source in our businesses and lives.Follow him on Twitter @i_robin or on LinkedIn.

8 Linux file managers to try

BY DAVID BOTH

ONE OF THE MOST COMMON administrative tasks that end users and administrators alike need to perform is file management. Managing files can consume a major portion of your time. Locating files, determining which files and folders (directories) are taking the most disk space, deleting files, moving files, and simply opening files for use in an application are some of the most basic—yet frequent—tasks we do as computer users. File management programs are tools that are intended to streamline and simplify those necessary chores.

Many choices

Many people aren't aware of the wide array of choices available in file managers, nor do they realize the full capabilities of the ones they do know about. As with every aspect of Linux, there are many options available for file managers. The most common ones provided by my favorite distribution, Fedora, are:

- Midnight Commander
- Konqueror
- Dolphin
- Krusader
- Nautilus
- Thunar
- PCmanFM
- XFE

I have used each of these at various times for various reasons and they all have qualities to recommend them. Ranging from very simple to feature-packed, there is a file manager available that will meet your needs. Midnight Commander and Krusader are my favorite file managers, and I use both quite frequently, but I also find myself using Dolphin and Konqueror.

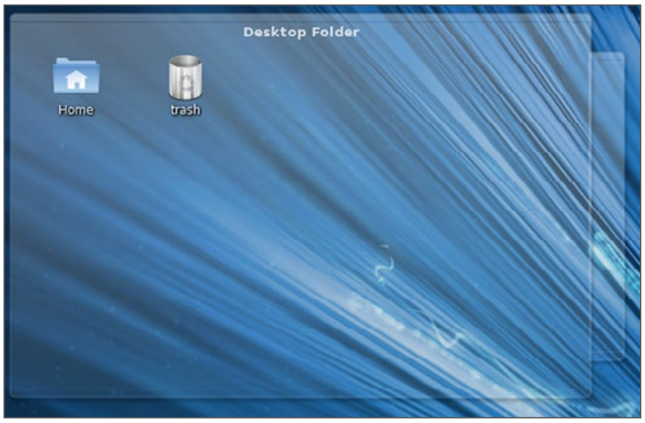
This article looks briefly at each of the file managers listed above and compares a few of their main features. Unfortunately, there is not enough space to do each of these file managers justice. I hope to have some future articles that provide a more detailed look at two or three of these powerful tools.

Each of these file managers is configurable, with Krusader and Konqueror being the most configurable of the GUI-based file managers. Midnight commander, the lone text-based file manager, is also quite configurable.

None of the file managers look by default like they do in this document. I have configured them to look like this on my systems. Except for Midnight Commander, the colors are managed in the "Application Appearance" section of the KDE System Settings application and are not configurable within the applications themselves.

Default File Manager

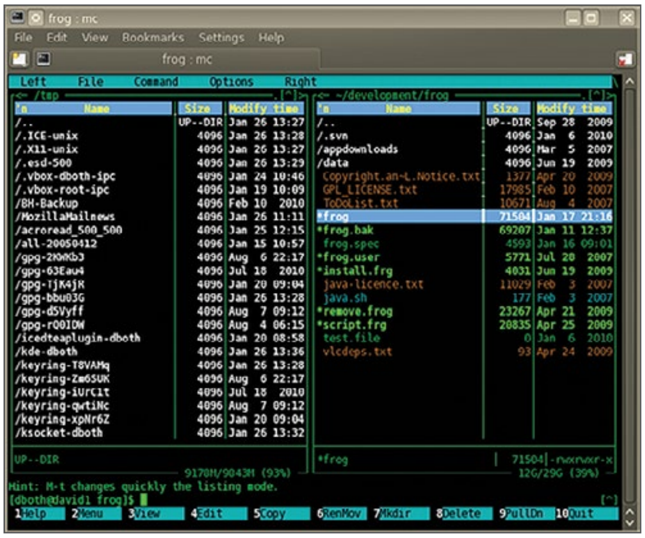
Like most Linux distributions, Fedora has a default file manager, which is currently Dolphin. The Linux desktop usually has an icon that looks like a little house—that's your home directory/folder. Click on the Home icon and the default file manager opens with your home directory as the PWD, or Present Working Directory. In current releases that use KDE 4.1 or above, the Home icon is located in the Desktop Folder along with the Trash icon, as shown below.



In KDE, the default file manager can be changed using **System Settings > Default Applications > File Manager**.

Midnight Commander

Midnight Commander [1] is a text-based Command Line Interface (CLI) program. It is particularly useful when a GUI is not available, but can also be used as a primary file manager in a terminal session even when you are using a GUI. I use Midnight Commander frequently because I often need to interact with local and remote Linux computers using the CLI. It can be used with almost any of the common shells and remote terminals through SSH.



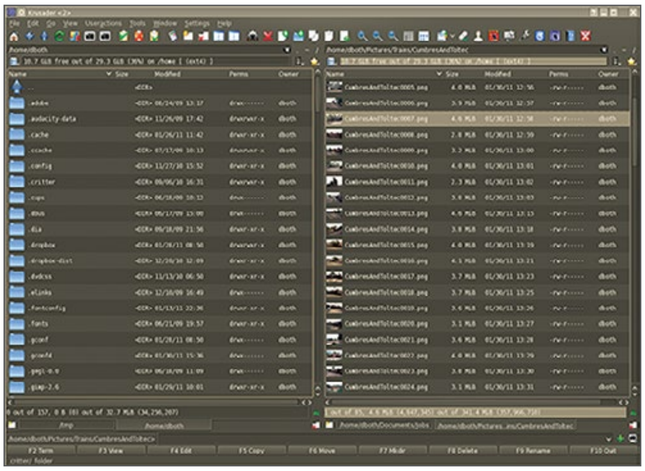
```
frog : mc
File  Edit  View  Bookmarks  Settings  Help
                 frog : mc
  Left     File     Command     Options     Right
┌<- /tmp ─────────────────────────.[^]>┐┌<- ~/development/frog ───────────────.[^]>┐
│'n         Name         │ Size  │Modify time │'n          Name         │ Size  │Modify time │
│/..                     │UP--DIR│Jan 26 13:27│/..                      │UP--DIR│Sep 28  2009│
│/.ICE-unix              │   4096│Jan 26 13:28│/.svn                    │   4096│Jan  6  2010│
│/.X11-unix              │   4096│Jan 26 13:27│/appdownloads            │   4096│Mar  5  2007│
│/.esd-500               │   4096│Jan 26 13:29│/data                    │   4096│Jun 19  2009│
│/.vbox-dboth-ipc        │   4096│Jan 24 10:46│ Copyright.an-L.Notice.txt│   1377│Apr 20  2009│
│/.vbox-root-ipc         │   4096│Jan 19 10:09│ GPL_LICENSE.txt         │  17985│Feb 10  2007│
│/BH-Backup              │   4096│Feb 10  2010│ ToDoList.txt            │  10671│Aug  4  2007│
│/MozillaMailnews        │   4096│Jan 26 11:11│*frog                    │  71504│Jan 17 21:16│
│/acroread_500_500       │   4096│Jan 25 12:15│*frog.bak                │  69207│Jan 11 12:37│
│/all-20050412           │   4096│Jan 15 10:57│ frog.spec               │   4593│Jan 16 09:01│
│/gpg-2KWKb3             │   4096│Aug  6 22:17│*frog.user               │   5771│Jul 20  2007│
│/gpg-63Eau4             │   4096│Jul 18  2010│*install.frg             │   4031│Jun 19  2009│
│/gpg-TjK4jR             │   4096│Jan 20 09:04│ java-licence.txt        │  11029│Feb  3  2007│
│/gpg-bbu03G             │   4096│Jan 26 13:28│ java.sh                 │    177│Feb  3  2007│
│/gpg-d5Vyff             │   4096│Aug  7 09:12│*remove.frog             │  23267│Apr 21  2009│
│/gpg-rQ0IDW             │   4096│Aug  4 06:15│*script.frg              │  20835│Apr 25  2009│
│/icedteaplugin-dboth    │   4096│Jan 20 08:58│ test.file               │      0│Jan  6  2010│
│/kde-dboth              │   4096│Jan 26 13:36│ vlcdeps.txt             │     93│Apr 24  2009│
│/keyring-T8VAMq         │   4096│Jan 26 13:28│                         │       │            │
│/keyring-Zm6SUK         │   4096│Aug  6 22:17│                         │       │            │
│/keyring-iUrC1t         │   4096│Jul 18  2010│                         │       │            │
│/keyring-qwtiNc         │   4096│Aug  7 09:12│                         │       │            │
│/keyring-xpNr6Z         │   4096│Jan 20 09:04│                         │       │            │
│/ksocket-dboth          │   4096│Jan 26 13:32│                         │       │            │
├────────────────────────────────────────────┤├─────────────────────────────────────────────┤
│UP--DIR                                     ││*frog                      │  71504│-rwxrwxr-x│
└──────────────────────── 9170M/9043M (93%) ─┘└──────────────────────────── 12G/29G (39%) ─┘
Hint: M-t changes quickly the listing mode.
[dboth@david1 frog]$                                                                    [^]
 1Help   2Menu   3View   4Edit   5Copy   6RenMov  7Mkdir  8Delete  9PullDn 10Quit
```

You can start Midnight Commander from the CLI with the mc command. The above image shows Midnight Commander in one tab of the Konsole program. The user interface for Midnight Commander is two text mode panes, left and right, which each display the contents of a directory. The top of each pane displays the name of the current directory for that pane. Navigation is accomplished with the arrow and tab keys. The Enter key can be used to enter a highlighted directory.

Along the very top of the Midnight Commander interface is a menu bar containing menu items for configuring Midnight Commander, the left and right panes, and for issuing various file commands. The bottom portion of the interface displays information about the file or directory highlighted in each pane, a hint feature and a line of function key labels; you can simply press the function key on your keyboard that corresponds to the function you want to perform. Between the hint line and the function keys is a command line.

Krusader

Krusader [2] is an exceptional file manager that is modeled after Midnight Commander. It uses a similar two-pane interface, but it's graphical instead of text-based. Krusader provides many features that enhance its functionality as a file manager. Krusader allows you to use the same keyboard navigation and command structure as Midnight Commander, and also allows you to use the mouse or trackball to navigate and perform all of the standard drag and drop operations you would expect on files.



The primary user interface for Krusader, much like that of Midnight Commander, is two text-mode panes—left and right—which each display the contents of a directory. The top of each pane contains the name of the current directory for that pane. In addition, tabs can be opened for each pane and a different directory can be open in each tab. Navigation is accomplished with the arrow and tab keys or the mouse. The Enter key can be used to enter a highlighted directory.

Each tab and pane can be configured to show files in one of two different Modes. In the illustration above, files are displayed in the detailed view that—in addition to the file name and an icon or preview—shows the file size, the date it was last modified, the owner, and the file permissions.

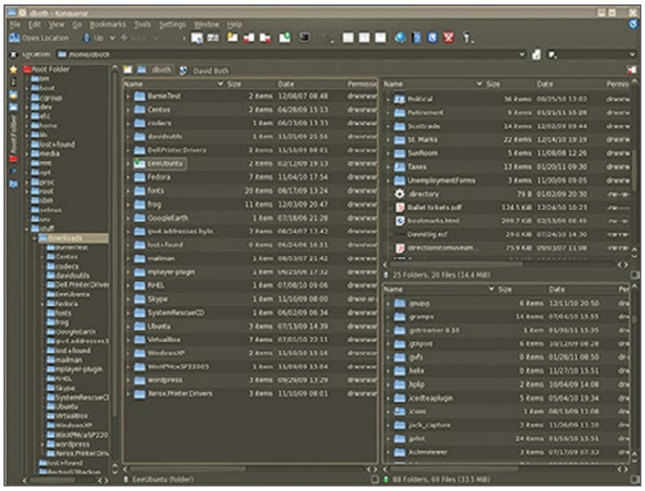
Along the very top of the Krusader GUI are a menu bar and toolbar containing menu items for configuring Krusader and managing files. The bottom portion of the interface displays a line of function key labels; you can simply press the function key on your keyboard that corresponds to the function you want to perform. At the bottom of the interface is a command line.

Krusader automatically saves the current tab and directory locations as well as other configuration items so that you will always return to the last configuration and set of directories when restarting the application.

Konqueror

Konqueror [3] is another powerful and flexible file manager with many features. It has one feature that none of the other file managers do: it doubles as a web browser. Just type the URL of the web site you want to view in the location bar.

The main thing that sets Konqeuror apart from the crowd is the ability to open multiple tabs, each of which can have one or more directory navigation panes. In the image below, one tab has been divided into three panes; one on the left side and two on the right. The sidebar at the far left is used to provide rapid navigation of the entire filesystem.



One thing I particularly like about Konqueror is that it provides an excellent high-level view of your directory structure, both in the sidebar and in the directory panels. This makes it easier to locate and delete files and directory trees that are no longer needed. It also enables easier navigation and reorganization of the directory structure.

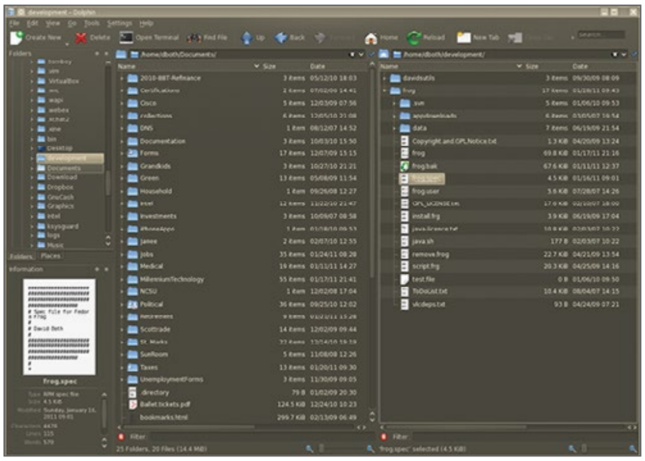
The primary user interface for Konqueror, much like that of Midnight Commander and Krusader, is text-mode panes that display the contents of a directory. Konqueror, however, allows multiple panes, and the default single pane can be split both horizontally and vertically as many times as it makes sense to do so. Konqueror also supports multiple tabs (at the top of the directory panes this time) and a different set of directories can be open in each tab. Navigation is accomplished with the arrow and tab keys or the mouse. The Enter key can be used to enter a highlighted directory. The Location widget near the top of the Konqueror GUI contains the full path of the currently selected pane.

Each tab and pane can be configured to show files in one of two different Modes. In the above image, files are displayed in the detailed view which, in addition to the file name and an icon or preview, shows the file size, the date it was last modified, the owner, and the file permissions.

Along the very top of the Graphical User Interface are a menu bar and tool bar containing menu items for configuring Konqueror and managing files. Once you have the tabs and panes set up the way you want them, you can save it so that Konqueror will always start with that configuration.

Dolphin

Dolphin [4] is very much like Konqueror and Krusader. It has two directory navigation panes and a sidebar that allows for easy filesystem navigation. It supports tabs.

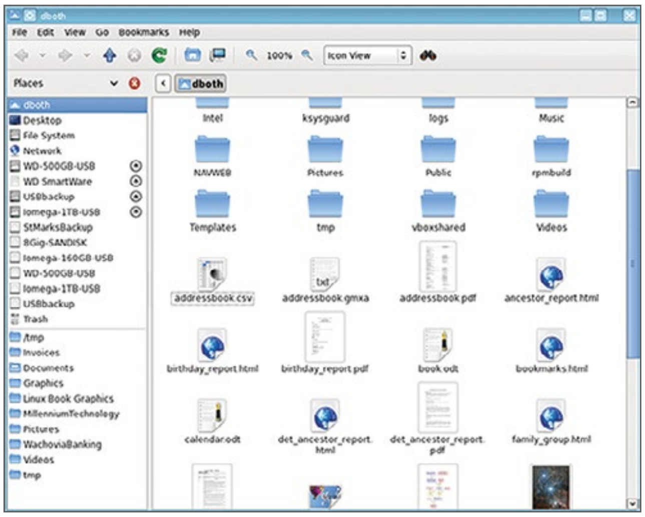


The primary user interface for Dolphin can be configured to be very similar to Konqueror and Krusader. Using two panes which each display the contents of a directory, it does not support splitting the panes. Navigation is accomplished with the arrow and tab keys or the mouse. The Enter key can be used to enter a highlighted directory. Dolphin also supports expanding the directory trees (folders) in both the sidebar navigation pane and the directory panes.

Although Dolphin does support tabs, when restarted it always reverts to the default of one pair of directory panes that display your home directory.

Nautilus

Nautilus [5] has a single directory pane with which to work. It also has a sidebar for navigation. Nautilus is a simple, decent file manager that is good for many beginners due to its simplicity. Nautilus is typically found in systems where GNOME is the desktop, but it can also be installed and used with KDE.



The primary user interface for Nautilus is fairly simple with a navigation sidebar and a single directory window in which to work. It does not support multiple tabs or splitting the panes. Navigation is accomplished with the arrow and tab keys or the mouse. The Enter key can be used to enter a highlighted directory.

Thunar

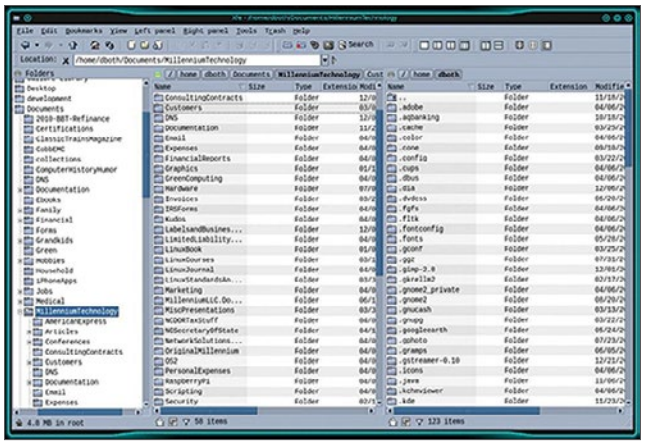
Thunar [6] is another lightweight file manager. It is so much like Nautilus in the way it looks and works and that there is nothing else to say about it.

PCmanFM

The PCManFM [7] file manager is intended to replace Nautilus and Thunar. In fact, based on the way they look and work so much alike, they may actually share some common code. These three file managers have the fewest configuration options and all share the same simple interface.

XFE

XFE [8] is one of the more interesting of the file managers as it has an interface all its own and is a bit more flexible than Nautilus, Thunar, and PCManFM.


File Edit Bookmarks View Left panel Right panel Tools Trash Help
Search
Location: /home/dboth/Documents/MillenniumTechnology
Folders
4.0 MB in root
58 items
123 items

XFE may be configured to display one or two directory panes, and the navigation bar is optional. It performs all the expected drag and drop functions, but it requires some manual configuration to associate the correct applications like LibreOffice with specific file types. It has a reasonable set of configuration options, but nowhere near those of Konqueror or Krusader.

XFE is also quite difficult about retaining its own set of "themes" and has no option to use the desktop color scheme, icons, decorations, or widgets.

Recommendations

I know that there are other file managers, one of which may be your favorite. Your choice of file manager should be the one that works best for you. GNU/Linux provides several viable choices and one will most likely meet most of your needs. If your favorite does not meet your needs for a particular task, you can always use the one that does.

All of these file managers are free of charge and distributed under some form of open source license. All are available from common, trusted repositories for Fedora and CentOS.

I plan to write some additional articles that cover some of these file managers in a bit more detail. Please leave your comments to let me know which ones you would like to know more about.

Links

[1] https://www.midnight-commander.org/
[2] http://www.krusader.org/
[3] https://konqueror.org/
[4] https://www.kde.org/applications/system/dolphin/
[5] https://wiki.gnome.org/action/show/Apps/Nautilus?action=show&redirect=Nautilus
[6] https://wiki.archlinux.org/index.php/Thunar
[7] http://wiki.lxde.org/en/PCManFM
[8] http://roland65.free.fr/xfe/

Author

David Both is an Open Source Software and GNU/Linux advocate, trainer, writer, and speaker who lives in Raleigh North Carolina. He is a strong proponent of and evangelist for the "Linux Philosophy."

David has been in the IT industry for nearly 50 years. 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 over 20 years.

David prefers to purchase the components and build his own computers from scratch to ensure that each new computer meets his exacting specifications. His primary workstation is an ASUS TUF X299 motherboard and an Intel i9 CPU with 16 cores (32 CPUs) and 64GB of RAM in a ThermalTake Core X9 case.

David has written articles for magazines including, Linux Magazine, Linux Journal. His article "Complete Kickstart," co-authored with a colleague at Cisco, was ranked 9th in the Linux Magazine Top Ten Best System Administration Articles list for 2008. David currently writes prolifically for OpenSource.com and Enable SysAdmin.

David currently has one book published, "The Linux Philosophy for SysAdmins." and is now is working on his next project, "Using and Administering Linux: Zero to SysAdmin," a self-study training course in three volumes that is scheduled for release in late 2019.

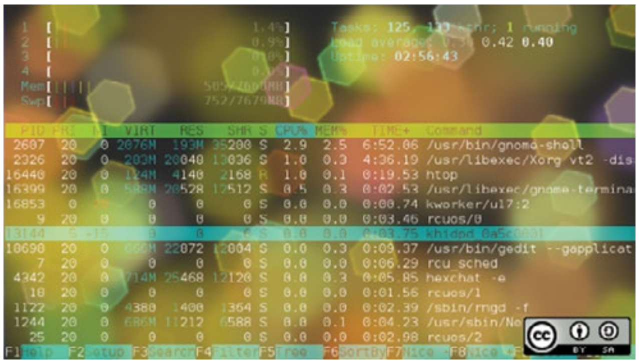
David can be reached at LinuxGeek46@both.org or on Twitter @LinuxGeek46.

4 open source tools for Linux system monitoring

BY DAVID BOTH

INFORMATION IS THE KEY to resolving any computer problem, including problems with or relating to Linux and the hardware on which it runs. There are many tools available for and included with most distributions even though they are not all installed by default. These tools can be used to obtain huge amounts of information.

This article discusses some of the interactive command line interface (CLI) tools that are provided with or which can be easily installed on Red Hat related distributions including Red Hat Enterprise Linux, Fedora, CentOS, and other derivative distributions. Although there are GUI tools available and they offer good information, the CLI tools provide all of the same information and they are always usable because many servers do not have a GUI interface but all Linux systems have a command line interface.



```
  1  [                     1.4%]   Tasks: 125, 133 thr; 1 running
  2  [                     0.9%]   Load average: 0.3  0.42 0.40
  3  [                     0.0%]   Uptime: 02:56:43
  4  [                     0.2%]
  Mem[||||           585/7680MB]
  Swp[               752/7679MB]

  PID PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 2607  20   0 2076M  193M 35200 S  2.9  2.5  6:52.06 /usr/bin/gnome-shell
 2326  20   0  203M 20040 13036 S  1.0  0.3  4:36.19 /usr/libexec/Xorg vt2 -dis
16440  20   0  124M  4140  2168 R  1.0  0.1  0:19.53 htop
16399  20   0  580M 20528 12512 S  0.5  0.3  0:02.53 /usr/libexec/gnome-termina
16853   0 -20     0     0     0 S  0.0  0.0  0:00.74 kworker/u17:2
    9  20   0     0     0     0 S  0.0  0.0  0:03.46 rcuos/0
13144   5 -15     0     0     0 S  0.0  0.0  0:03.75 khidpd_0a5c
10690  20   0  666M 22072 12004 S  0.0  0.3  0:09.37 /usr/bin/gedit --gapplicat
    7  20   0     0     0     0 S  0.0  0.0  0:06.29 rcu_sched
 4342  20   0  714M 25468 12120 S  0.0  0.3  0:05.85 hexchat -e
   18  20   0     0     0     0 S  0.0  0.0  0:01.56 rcuos/1
 1122  20   0  4380  1400  1364 S  0.0  0.0  0:02.39 /sbin/rngd -f
 1244  20   0  686M 11212  6588 S  0.0  0.1  0:04.23 /usr/sbin/No
   25  20   0     0     0     0 S  0.0  0.0  0:02.98 rcuos/2
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +
```

This article concentrates on the tools that I typically use. If I did not cover your favorite tool, please forgive me and let us all know what tools you use and why in the comments section.

My go to tools for problem determination in a Linux environment are almost always the system monitoring tools. For me, these are top, atop, htop, and glances.

All of these tools monitor CPU and memory usage, and most of them list information about running processes at the very least. Some monitor other aspects of a Linux system as well. All provide near real-time views of system activity.

Load averages

Before I go on to discuss the monitoring tools, it is important to discuss load averages in more detail.

Load averages are an important criteria for measuring CPU usage, but what does this really mean when I say that the 1 (or 5 or 10) minute load average is 4.04, for example? Load average can be considered a measure of demand for the CPU; it is a number that represents the average number of instructions waiting for CPU time. So this is a true measure of CPU performance, unlike the standard "CPU percentage" which includes I/O wait times during which the CPU is not really working.

For example, a fully utilized single processor system CPU would have a load average of 1. This means that the CPU is keeping up exactly with the demand; in other words it has perfect utilization. A load average of less than one means that the CPU is underutilized and a load average of greater than 1 means that the CPU is overutilized and that there is pent-up, unsatisfied demand. For example, a load average of 1.5 in a single CPU system indicates that one-third of the CPU instructions are forced to wait to be executed until the one preceding it has completed.

This is also true for multiple processors. If a 4 CPU system has a load average of 4 then it has perfect utilization. If it has a load average of 3.24, for example, then three of its processors are fully utilized and one is utilized at about 76%. In the example above, a 4 CPU system has a 1 minute load average of 4.04 meaning that there is no remaining capacity among the 4 CPUs and a few instructions are forced to wait. A perfectly utilized 4 CPU system would show a load average of 4.00 so that the system in the example is fully loaded but not overloaded.

The optimum condition for load average is for it to equal the total number of CPUs in a system. That would mean that every CPU is fully utilized and yet no instruction must be forced to wait. The longer-term load averages provide indication of the overall utilization trend.

Linux Journal has an excellent article [1] describing load averages, the theory and the math behind them, and how to interpret them in the December 1, 2006 issue.

Signals

All of the monitors discussed here allow you to send signals [2] to running processes. Each of these signals has a specific function though some of them can be defined by the receiving program using signal handlers.

The separate **kill** command can also be used to send signals to processes outside of the monitors. The **kill -l** can be used to list all possible signals that can be sent. Three of these signals can be used to kill a process.

SIGTERM (15): Signal 15, SIGTERM is the default signal sent by top and the other monitors when the **k** key is pressed. It may also be the least effective because the program must have a signal handler built into it. The program's signal handler must intercept incoming signals and act accordingly. So for scripts, most of which do not have signal handlers, SIGTERM is ignored. The idea behind SIGTERM is that by simply telling the program that you want it to terminate itself, it will take advantage of that and clean up things like open files and then terminate itself in a controlled and nice manner.

SIGKILL (9): Signal 9, SIGKILL provides a means of killing even the most recalcitrant programs, including scripts and other programs that have no signal handlers. For scripts and other programs with no signal handler, however, it not only kills the running script but it also kills the shell session in which the script is running; this may not be the behavior that you want. If you want to kill a process and you don't care about being nice, this is the signal you want. This signal cannot be intercepted by a signal handler in the program code.

SIGINT (2): Signal 2, SIGINT can be used when SIGTERM does not work and you want the program to die a little more nicely, for example, without killing the shell session in which it is running. SIGINT sends an interrupt to the session in which the program is running. This is equivalent to terminating a running program, particularly a script, with the **Ctrl-C** key combination.

To experiment with this, open a terminal session and create a file in /tmp named cpuHog and make it executable with the permissions rwxr_xr_x. Add the following content to the file.

```
#!/bin/bash # This little program is a cpu hog X=0;
  while [ 1 ];do echo $X;X=$((X+1));done
```

Open another terminal session in a different window, position them adjacent to each other so you can watch the results and run **top** in the new session. Run the cpuHog program with the following command:

```
/tmp/cpuHog
```

This program simply counts up by one and prints the current value of X to STDOUT. And it sucks up CPU cycles. The terminal session in which cpuHog is running should show a very high CPU usage in top. Observe the effect this has on system performance in top. CPU usage should immediately go way up and the load averages should also start to increase over time. If you want, you can open additional terminal sessions and start the cpuHog program in them so that you have multiple instances running.

Determine the PID of the cpuHog program you want to kill. Press the k key and look at the message under the Swap line at the bottom of the summary section. Top asks for the PID of the process you want to kill. Enter that PID and press **Enter**. Now top asks for the signal number and displays the default of 15. Try each of the signals described here and observe the results.

4 open source tools for Linux system monitoring

top

One of the first tools I use when performing problem determination is **top**. I like it because it has been around *since forever* and is always available while the other tools may not be installed.

The top program is a very powerful utility that provides a great deal of information about your running system. This includes data about memory usage, CPU loads, and a list of running processes including the amount of CPU time and memory being utilized by each process. Top displays system information in near real-time, updating (by default) every three seconds. Fractional seconds are allowed by top, although very small values can place a significant load the system. It is also interactive and the data columns to be displayed and the sort column can be modified.

A sample output from the top program is shown in Figure 1 below. The output from top is divided into two sections which are called the "summary" section, which is the top section of the output, and the "process" section which is the lower portion of the output; I will use this terminology for top, atop, htop and glances in the interest of consistency.

The top program has a number of useful interactive commands you can use to manage the display of data and to manipulate individual processes. Use the h command to view a brief help page for the various interactive commands. Be sure to press h twice to see both pages of the help. Use the **q** command to quit.

Summary section

The summary section of the output from top is an overview of the system status. The first line shows the system uptime and the 1, 5, and 15 minute load averages. In the example below, the load averages are 4.04, 4.17, and 4.06 respectively.

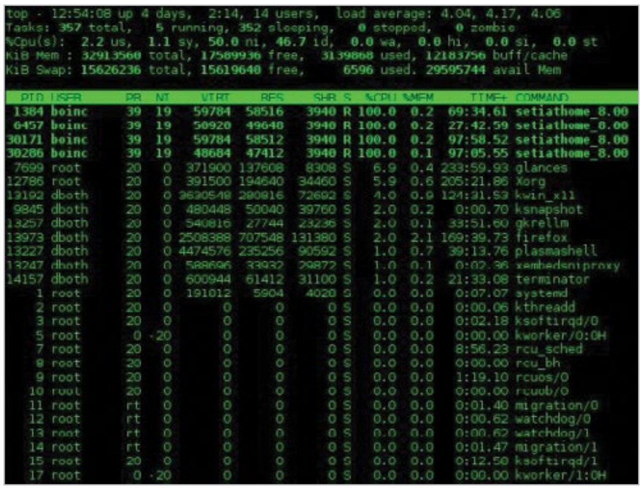
The second line shows the number of processes currently active and the status of each.

The lines containing CPU statistics are shown next. There can be a single line which combines the statistics for all CPUs present in the system, as in the example below, or one line for each CPU; in the case of the computer used for the example, this is a single quad core CPU. Press the 1 key to toggle between the consolidated display of CPU usage and the display of the individual CPUs. The data in these lines is displayed as percentages of the total CPU time available.

These and the other fields for CPU data are described below.

- **us: userspace** – Applications and other programs running in user space, i.e., not in the kernel.
- **sy: system calls** – Kernel level functions. This does not include CPU time taken by the kernel itself, just the kernel system calls.
- **ni: nice** – Processes that are running at a positive nice level.
- **id: idle** – Idle time, i.e., time not used by any running process.
- **wa: wait** – CPU cycles that are spent waiting for I/O to occur. This is wasted CPU time.
- **hi: hardware interrupts** – CPU cycles that are spent dealing with hardware interrupts.
- **si: software interrupts** – CPU cycles spent dealing with software-created interrupts such as system calls.
- **st: steal time** – The percentage of CPU cycles that a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.

The last two lines in the summary section are memory usage. They show the physical memory usage including both RAM and swap space.



```
top - 12:54:08 up 4 days,  2:14, 14 users,  load average: 4.04, 4.17, 4.06
Tasks: 357 total,   5 running, 352 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.2 us,  1.1 sy, 50.0 ni, 46.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32913560 total, 17589936 free,  3139868 used, 12183756 buff/cache
KiB Swap: 15626236 total, 15619640 free,     6596 used. 29595744 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1384 boinc     39  19   59784  58516   3940 R 100.0  0.2  69:34.61 setiathome_8.00
 6457 boinc     39  19   50920  49648   3940 R 100.0  0.2  27:42.59 setiathome_8.00
30171 boinc     39  19   59784  58512   3940 R 100.0  0.2  97:58.52 setiathome_8.00
30286 boinc     39  19   48684  47412   3940 R 100.0  0.1  97:05.55 setiathome_8.00
 7699 root      20   0  371900 137608   8308 S   6.9  0.4 233:59.93 glances
12786 root      20   0  391500 194640  34460 S   5.9  0.6 205:21.86 Xorg
13192 dboth     20   0 3630548 280816  72692 S   4.0  0.9 124:31.53 kwin_x11
 9845 dboth     20   0  480448  50040  39760 S   2.0  0.2   0:00.70 ksnapshot
13257 dboth     20   0  540816  27744  23236 S   2.0  0.1  33:51.60 gkrellm
13973 dboth     20   0 2508388 707548 131380 S   2.0  2.1 169:39.73 firefox
13227 dboth     20   0 4474576 235256  90592 S   1.0  0.7  39:13.76 plasmashell
13247 dboth     20   0  588696  33932  29872 S   1.0  0.1   0:02.36 xembedsniproxy
14157 dboth     20   0  600944  61412  31100 S   1.0  0.2  21:33.08 terminator
    1 root      20   0  191012   5904   4020 S   0.0  0.0   0:07.07 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.06 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:02.18 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   8:56.23 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   1:19.10 rcuos/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:01.40 migration/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.62 watchdog/0
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.62 watchdog/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:01.47 migration/1
   15 root      20   0       0      0      0 S   0.0  0.0   0:12.50 ksoftirqd/1
   17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
```

Figure 1: The top command showing a fully utilized 4-core CPU.

You can use the **1** command to display CPU statistics as a single, global number as shown in Figure 1, above, or by individual CPU. The **l** command turns load averages on and off. The **t** and **m** commands rotate the process/CPU and memory lines of the summary section, respectively, through off, text only, and a couple types of bar graph formats.

Process section

The process section of the output from top is a listing of the running processes in the system—at least for the number of processes for which there is room on the terminal display. The default columns displayed by top are described below. Several other columns are available and each can usually be added with a single keystroke. Refer to the top man page for details.

- **PID** – The Process ID.
- **USER** – The username of the process owner.
- **PR** – The priority of the process.
- **NI** – The nice number of the process.
- **VIRT** – The total amount of virtual memory allocated to the process.
- **RES** – Resident size (in kb unless otherwise noted) of non-swapped physical memory consumed by a process.
- **SHR** – The amount of shared memory in kb used by the process.
- **S** – The status of the process. This can be R for running, S for sleeping, and Z for zombie. Less frequently seen statuses can be T for traced or stopped, and D for uninterruptable sleep.
- **%CPU** – The percentage of CPU cycles, or time used by this process during the last measured time period.
- **%MEM** – The percentage of physical system memory used by the process.
- **TIME+** – Total CPU time to 100ths of a second consumed by the process since the process was started.
- **COMMAND** – This is the command that was used to launch the process.

Use the **Page Up** and **Page Down** keys to scroll through the list of running processes. The **d** or **s** commands are interchangeable and can be used to set the delay interval between updates. The default is three seconds, but I prefer a one second interval. Interval granularity can be as low as one-tenth (0.1) of a second but this will consume more of the CPU cycles you are trying to measure.

You can use the **<** and **>** keys to sequence the sort column to the left or right.

The **k** command is used to kill a process or the r command to renice it. You have to know the process ID (PID) of the process you want to kill or renice and that information is displayed in the process section of the top display. When killing a process, top asks first for the PID and then for the signal number to use in killing the process. Type them in and press the enter key after each. Start with signal 15, SIGTERM, and if that does not kill the process, use 9, SIGKILL.

Configuration

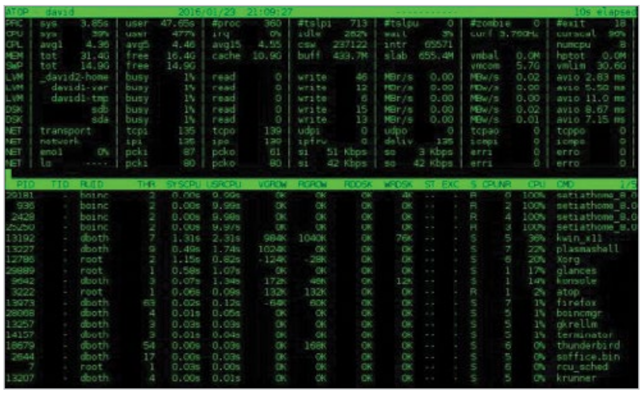
If you alter the top display, you can use the **W** (in uppercase) command to write the changes to the configuration file, ~/.toprc in your home directory.

atop

I also like atop. It is an excellent monitor to use when you need more details about that type of I/O activity. The default refresh interval is 10 seconds, but this can be changed using the interval **i** command to whatever is appropriate for what you are trying to do. atop cannot refresh at sub-second intervals like top can.

Use the **h** command to display help. Be sure to notice that there are multiple pages of help and you can use the space bar to scroll down to see the rest.

One nice feature of atop is that it can save raw performance data to a file and then play it back later for close inspection. This is handy for tracking down internmittent problems, especially ones that occur during times when you cannot directly monitor the system. The **atopsar** program is used to play back the data in the saved file.



```
ATOP - david              2016/01/23  21:09:27              -------------              10s elapsed
PRC | sys    3.85s | user  47.65s | #proc    360 | #tslpi   713 | #tslpu     0 | #zombie    0 | #exit     18 |
CPU | sys      39% | user    477% | irq       0% | idle    282% | wait      3% | curf 3.79GHz | curscal  90% |
CPL | avg1    4.36 | avg5    4.46 | avg15   4.55 | csw   237122 | intr   65571 |              | numcpu     8 |
MEM | tot    31.4G | free   16.4G | cache  10.9G | buff  433.7M | slab  655.4M | vmbal   0.0M | hptot   0.0M |
SWP | tot    14.9G | free   14.9G |              |              |              | vmcom   5.7G | vmlim  30.6G |
LVM | _david2-home | busy      1% | read       0 | write     46 | MBr/s   0.00 | MBw/s   0.02 | avio 2.83 ms |
LVM |   david1-var | busy      1% | read       0 | write     12 | MBr/s   0.00 | MBw/s   0.00 | avio 5.50 ms |
LVM |   david1-tmp | busy      1% | read       0 | write      6 | MBr/s   0.00 | MBw/s   0.00 | avio 11.0 ms |
DSK |          sdb | busy      1% | read       0 | write     15 | MBr/s   0.00 | MBw/s   0.02 | avio 8.67 ms |
DSK |          sda | busy      1% | read       0 | write     13 | MBr/s   0.00 | MBw/s   0.01 | avio 7.15 ms |
NET | transport    | tcpi     135 | tcpo     139 | udpi       0 | udpo       0 | tcpao      0 | tcppo      0 |
NET | network      | ipi      135 | ipo      139 | ipfrw      0 | deliv    135 | icmpi      0 | icmpo      0 |
NET | eno1      0% | pcki      87 | pcko      61 | si   51 Kbps | so    3 Kbps | erri       0 | erro       0 |
NET | lo      ---- | pcki      80 | pcko      80 | si   42 Kbps | so   42 Kbps | erri       0 | erro       0 |

  PID   TID  RUID      THR  SYSCPU  USRCPU   VGROW   RGROW  RDDSK  WRDSK  ST EXC  S CPUNR   CPU  CMD        1/5
29181     -  boinc       2   0.00s   9.99s      0K      0K     0K     4K  --   -  R     0  100%  setiathome_8.0
  936     -  boinc       2   0.00s   9.99s      0K      0K     0K     0K  --   -  R     2  100%  setiathome_8.0
 2428     -  boinc       2   0.00s   9.98s      0K      0K     0K     0K  --   -  R     4  100%  setiathome_8.0
25250     -  boinc       2   0.00s   9.97s      0K      0K     0K     0K  --   -  R     3  100%  setiathome_8.0
13192     -  dboth       7   1.31s   2.31s    984K   1040K     0K    76K  --   -  S     5   36%  kwin_x11
13227     -  dboth       9   0.49s   1.74s   1024K      0K     0K     0K  --   -  S     7   22%  plasmashell
12786     -  root        2   1.15s   0.82s   -124K    -28K     0K     0K  --   -  S     6   20%  Xorg
29889     -  root        1   0.58s   1.07s      0K      0K     0K     0K  --   -  S     1   17%  glances
 9642     -  dboth       3   0.07s   1.34s    172K     48K     0K    12K  --   -  S     1   14%  konsole
 3222     -  root        1   0.06s   0.09s    132K    132K     0K     0K  --   -  R     1    2%  atop
13973     -  dboth      63   0.02s   0.12s    -64K     60K     0K     0K  --   -  S     7    1%  firefox
28068     -  dboth       4   0.01s   0.05s      0K      0K     0K     0K  --   -  S     5    1%  boincmgr
13257     -  dboth       3   0.03s   0.03s      0K      0K     0K     0K  --   -  S     5    1%  gkrellm
14157     -  dboth       3   0.01s   0.04s      0K      0K     0K     0K  --   -  S     5    1%  terminator
18679     -  dboth      54   0.00s   0.03s      0K    168K     0K     0K  --   -  S     6    0%  thunderbird
 2644     -  dboth      17   0.00s   0.03s      0K      0K     0K     0K  --   -  S     5    0%  soffice.bin
    7     -  root        1   0.03s   0.00s      0K      0K     0K     0K  --   -  S     6    0%  rcu_sched
13207     -  dboth       4   0.00s   0.01s      0K      0K     0K     0K  --   -  S     5    0%  krunner
```

Figure 2: The atop system monitor provides information about disk and network activity in addition to CPU and process data.

Summary section

atop contains much of the same information as top but also displays information about network, raw disk, and logical volume activity. Figure 2, above, shows these additional data in the columns at the top of the display. Note that if you have the horizontal screen real-estate to support a wider display, additional columns will be displayed. Conversely, if you have less horizontal width, fewer columns are displayed. I also like that atop displays the current CPU frequency and scaling factor—something I have not seen on any other of these monitors—on the second line in the rightmost two columns in Figure 2.

Process section

The atop process display includes some of the same columns as that for top, but it also includes disk I/O information and thread count for each process as well as virtual and real memory growth statistics for each process. As with the summary section, additional columns will display if there is sufficient horizontal screen real-estate. For example, in Figure 2, the RUID (Real User ID) of the process owner is displayed. Expanding the display will also show the EUID (Effective User ID) which might be important when programs run SUID (Set User ID).

atop can also provide detailed information about disk, memory, network, and scheduling information for each process. Just press the **d**, **m**, **n** or **s** keys respectively to view that data. The **g** key returns the display to the generic process display.

Sorting can be accomplished easily by using **C** to sort by CPU usage, **M** for memory usage, **D** for disk usage, **N** for network usage and **A** for automatic sorting. Automatic sorting usually sorts processes by the most busy resource. The network usage can only be sorted if the netatop kernel module is installed and loaded.

You can use the **k** key to kill a process but there is no option to renice a process.

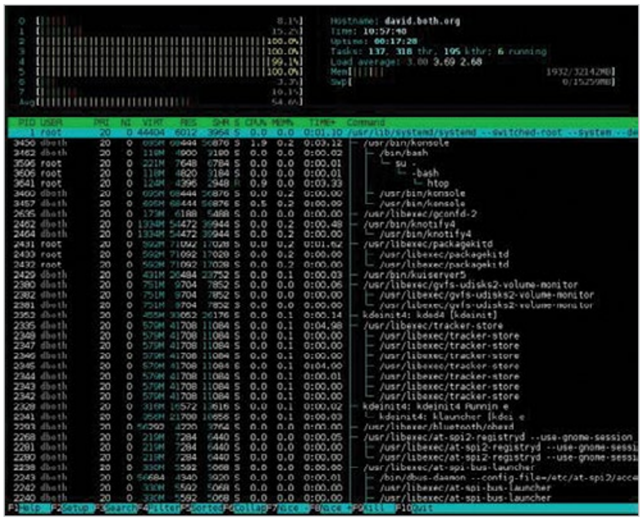
By default, network and disk devices for which no activity occurs during a given time interval are not displayed. This can lead to mistaken assumptions about the hardware configuration of the host. The **f** command can be used to force atop to display the idle resources.

Configuration

The atop man page refers to global and user level configuration files, but none can be found in my own Fedora or CentOS installations. There is also no command to save a modified configuration and a save does not take place automatically when the program is terminated. So, there appears to be now way to make configuration changes permanent.

htop

The htop program is much like top *but on steroids*. It does look a lot like top, but it also provides some capabilities that top does not. Unlike atop, however, it does not provide any disk, network, or I/O information of any type.



```
  0  [|||||                                      8.1%]   Hostname: david.both.org
  1  [||||||||                                  15.2%]   Time: 10:57:40
  2  [|||||||||||||||||||||||||||||||||||||||||100.0%]   Uptime: 00:17:28
  3  [|||||||||||||||||||||||||||||||||||||||||100.0%]   Tasks: 137, 318 thr, 195 kthr; 6 running
  4  [||||||||||||||||||||||||||||||||||||||||| 99.1%]   Load average: 3.00 3.69 2.68
  5  [|||||||||||||||||||||||||||||||||||||||||100.0%]   Mem[|||||||                    1932/32142MB]
  6  [||                                         3.3%]   Swp[                              0/15259MB]
  7  [||||||                                    10.1%]
  Avg[||||||||||||||||||||||||                  54.6%]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    1 root       20   0 44404  6012  3964 S  0.0  0.0  0:01.10 /usr/lib/systemd/systemd --switched-root --system --de
 3456 dboth      20   0  695M 68444 56876 S  1.9  0.2  0:03.12 ├─ /usr/bin/konsole
 3462 dboth      20   0  118M  4900  3180 S  0.0  0.0  0:00.02 │  ├─ /bin/bash
 3596 root       20   0  221M  7648  6784 S  0.0  0.0  0:00.01 │  │  └─ su -
 3606 root       20   0  118M  4820  3184 S  0.0  0.0  0:00.01 │  │     └─ -bash
 3641 root       20   0  124M  4396  2948 R  0.9  0.0  0:03.33 │  │        └─ htop
 3460 dboth      20   0  695M 68444 56876 S  0.0  0.2  0:00.00 │  ├─ /usr/bin/konsole
 3457 dboth      20   0  695M 68444 56876 S  0.5  0.2  0:00.09 │  └─ /usr/bin/konsole
 2635 dboth      20   0  173M  6188  5488 S  0.0  0.0  0:00.00 ├─ /usr/libexec/gconfd-2
 2462 dboth      20   0 1334M 54472 39944 S  0.0  0.2  0:00.48 ├─ /usr/bin/knotify4
 2464 dboth      20   0 1334M 54472 39944 S  0.0  0.2  0:00.00 │  └─ /usr/bin/knotify4
 2431 root       20   0  592M 71092 17028 S  0.0  0.2  0:01.62 ├─ /usr/libexec/packagekitd
 2433 root       20   0  592M 71092 17028 S  0.0  0.2  0:00.00 │  ├─ /usr/libexec/packagekitd
 2432 root       20   0  592M 71092 17028 S  0.0  0.2  0:00.00 │  └─ /usr/libexec/packagekitd
 2429 dboth      20   0  431M 26484 23752 S  0.0  0.1  0:00.03 ├─ /usr/bin/kuiserver5
 2380 dboth      20   0  751M  9704  7852 S  0.0  0.0  0:00.06 ├─ /usr/libexec/gvfs-udisks2-volume-monitor
 2382 dboth      20   0  751M  9704  7852 S  0.0  0.0  0:00.00 │  ├─ /usr/libexec/gvfs-udisks2-volume-monitor
 2381 dboth      20   0  751M  9704  7852 S  0.0  0.0  0:00.00 │  └─ /usr/libexec/gvfs-udisks2-volume-monitor
 2352 dboth      20   0  455M 33052 26176 S  0.0  0.1  0:00.14 ├─ kdeinit4: kded4 [kdeinit]
 2335 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:04.98 ├─ /usr/libexec/tracker-store
 2349 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.00 │  ├─ /usr/libexec/tracker-store
 2347 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.00 │  ├─ /usr/libexec/tracker-store
 2346 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.00 │  ├─ /usr/libexec/tracker-store
 2345 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:04.00 │  ├─ /usr/libexec/tracker-store
 2344 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.01 │  ├─ /usr/libexec/tracker-store
 2343 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.00 │  ├─ /usr/libexec/tracker-store
 2342 dboth      20   0  579M 41708 11084 S  0.0  0.1  0:00.00 │  └─ /usr/libexec/tracker-store
 2328 dboth      20   0  316M 16572 13616 S  0.0  0.1  0:00.02 ├─ kdeinit4: kdeinit4 Runnin e
 2341 dboth      20   0  356M 21700 16656 S  0.0  0.1  0:00.03 │  └─ kdeinit4: klauncher [kdei e
 2293 dboth      20   0 56292  4220  3764 S  0.0  0.0  0:00.00 ├─ /usr/libexec/bluetooth/obexd
 2268 dboth      20   0  219M  7284  6440 S  0.0  0.0  0:00.05 ├─ /usr/libexec/at-spi2-registryd --use-gnome-session
 2281 dboth      20   0  219M  7284  6440 S  0.0  0.0  0:00.00 │  ├─ /usr/libexec/at-spi2-registryd --use-gnome-sessi
 2280 dboth      20   0  219M  7284  6440 S  0.0  0.0  0:00.00 │  └─ /usr/libexec/at-spi2-registryd --use-gnome-sessi
 2238 dboth      20   0  330M  5592  5068 S  0.0  0.0  0:00.00 ├─ /usr/libexec/at-spi-bus-launcher
 2243 dboth      20   0 56684  4340  3920 S  0.0  0.0  0:00.01 │  ├─ /bin/dbus-daemon --config-file=/etc/at-spi2/acce
 2242 dboth      20   0  330M  5592  5068 S  0.0  0.0  0:00.00 │  ├─ /usr/libexec/at-spi-bus-launcher
 2240 dboth      20   0  330M  5592  5068 S  0.0  0.0  0:00.00 │  ├─ /usr/libexec/at-spi-bus-launcher
F1Help  F2Setup F3SearchF4FilterF5SortedF6CollapF7Nice -F8Nice +F9Kill  F10Quit
```

Figure 3: htop has nice bar charts to to indicate resource usage and it can show the process tree.

Summary section

The summary section of htop is displayed in two columns. It is very flexible and can be configured with several different types of information in pretty much any order you like. Although the CPU usage sections of top and atop can be toggled between a combined display and a display that shows one bar graph for each CPU, htop cannot. So it has a number of different options for the CPU display, including a single combined bar, a bar for each CPU, and various combinations in which specific CPUs can be grouped together into a single bar.

I think this is a cleaner summary display than some of the other system monitors and it is easier to read. The drawback to this summary section is that some information is not available in htop that is available in the other monitors, such as CPU percentages by user, idle, and system time.

The **F2** (Setup) key is used to configure the summary section of htop. A list of available data displays is shown and you can use function keys to add them to the left or right column and to move them up and down within the selected column.

Process section

The process section of htop is very similar to that of top. As with the other monitors, processes can be sorted any of several factors, including CPU or memory usage, user, or PID. Note that sorting is not possible when the tree view is selected.

The **F6** key allows you to select the sort column; it displays a list of the columns available for sorting and you select the column you want and press the **Enter** key.

You can use the up and down arrow keys to select a process. To kill a process, use the up and down arrow keys to select the target process and press the **k** key. A list of signals to send the process is displayed with 15, SIGTERM, selected. You can specify the signal to use, if different from SIGTERM. You could also use the **F7** and **F8** keys to renice the selected process.

One command I especially like is **F5** which displays the running processes in a tree format making it easy to determine the parent/child relationships of running processes.

Configuration

Each user has their own configuration file, ~/.config/htop/htoprc and changes to the htop configuration are stored there automatically. There is no global configuration file for htop.

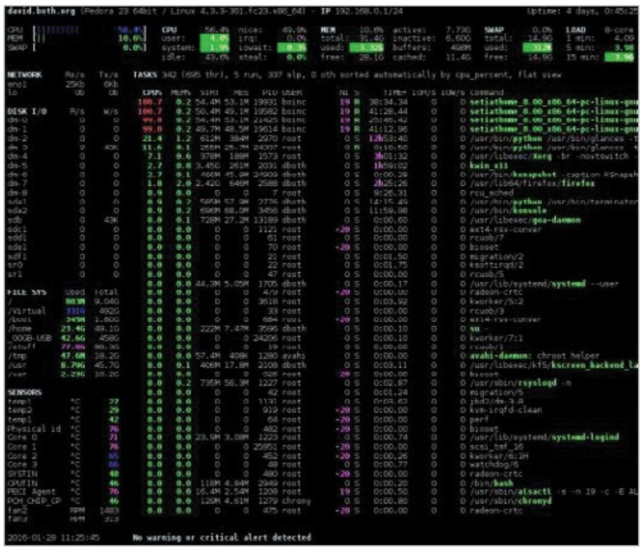
glances

I have just recently learned about glances, which can display more information about your computer than any of the other monitors I am currently familiar with. This includes disk and network I/O, thermal readouts that can display CPU and other hardware temperatures as well as fan speeds, and disk usage by hardware device and logical volume.

The drawback to having all of this information is that glances uses a significant amount of CPU resurces itself. On my systems I find that it can use from about 10% to 18% of CPU cycles. That is a lot so you should consider that impact when you choose your monitor.

Summary section

The summary section of glances contains most of the same information as the summary sections of the other monitors. If you have enough horizontal screen real estate it can show CPU usage with both a bar graph and a numeric indicator, otherwise it will show only the number.



```
david.both.org (Fedora 23 64bit / Linux 4.3.3-301.fc23.x86_64) - IP 192.168.0.1/24                              Uptime: 4 days, 0:45:25

CPU  [||||||||||     56.4%]  CPU     56.4%  nice:   49.9%  MEM    10.6%  active:    7.73G  SWAP    0.0%  LOAD   8-core
MEM  [||             10.6%]  user:    4.0%  irq:     0.0%  total: 31.4G  inactive:  6.60G  total: 14.9G  1 min:   4.09
SWAP [                0.0%]  system:  1.9%  iowait:  0.3%  used:  3.32G  buffers:   498M   used:   312K  5 min:   3.98
                             idle:   43.6%  steal:   0.0%  free:  28.1G  cached:    11.4G  free:  14.9G  15 min:  3.96

NETWORK      Rx/s   Tx/s   TASKS 342 (695 thr), 5 run, 337 slp, 0 oth sorted automatically by cpu_percent, flat view
enol         25Kb    6Kb
lo             0b     0b    CPU%  MEM%  VIRT   RES    PID USER    NI S    TIME+ IOR/s IOW/s Command
                           100.7   0.2 54.4M 53.1M  19931 boinc   19 R 38:34.34     0     0 setiathome_8.00_x86_64-pc-linux-gnu
DISK I/O      R/s    W/s   100.7   0.2 50.4M 49.1M  19582 boinc   19 R 41:28.44     0     0 setiathome_8.00_x86_64-pc-linux-gnu
dm-0            0      0    99.8   0.2 54.4M 53.1M  21425 boinc   19 R 25:40.42     0     0 setiathome_8.00_x86_64-pc-linux-gnu
dm-1            0      0    99.0   0.2 49.7M 48.5M  19614 boinc   19 R 41:12.96     0     0 setiathome_8.00_x86_64-pc-linux-gnu
dm-2            0      0    21.4   1.2  612M  384M   2970 root     0 S 12h53:40     0     0 /usr/bin/python /usr/bin/glances -t
dm-3            0    43K    11.6   0.1  255M 25.7M  24327 root     0 R  0:10.50     0     0 /usr/bin/python /usr/bin/glances -t
dm-4            0      0     7.1   0.6  378M  188M   1573 root     0 S  3h01:32     0     0 /usr/libexec/Xorg -br -novtswitch -
dm-5            0      0     2.7   0.8 3.45G  261M   2031 dboth    0 S  1h59:02     0     0 kwin_x11
dm-6            0      0     2.7   0.1  466M 45.9M  24909 dboth    0 S  0:00.28     0     0 /usr/bin/ksnapshot -caption KSnapsh
dm-7            0      0     1.8   2.0 2.42G  646M   2588 dboth    0 S  2h25:26     0     0 /usr/lib64/firefox/firefox
dm-8            0      0     0.9   0.0     0     0      7 root     0 S  9:26.31     0     0 rcu_sched
sda1            0      0     0.9   0.2  585M 57.9M   2776 dboth    0 S 14:15.49     0     0 /usr/bin/python /usr/bin/terminator
sda2            0      0     0.9   0.2  686M 68.0M   3456 dboth    0 S 11:59.88     0     0 /usr/bin/konsole
sdb             0    43K     0.0   0.1  728M 27.2M  13189 dboth    0 S  0:00.60     0     0 /usr/libexec/goa-daemon
sdc1            0      0     0.0   0.0     0     0   1121 root   -20 S  0:00.00     0     0 ext4-rsv-conver
sdd1            0      0     0.0   0.0     0     0     61 root     0 S  0:00.00     0     0 rcuob/7
sde1            0      0     0.0   0.0     0     0     70 root   -20 S  0:00.80     0     0 bioset
sdf1            0      0     0.0   0.0     0     0     21 root     0 S  0:01.50     0     0 migration/2
sr0             0      0     0.0   0.0     0     0     22 root     0 S  0:01.75     0     0 ksoftirqd/2
sr1             0      0     0.0   0.0     0     0     47 root     0 S  0:00.00     0     0 rcuob/5
                             0.0   0.0 44.3M 5.05M   1705 dboth    0 S  0:00.17     0     0 /usr/lib/systemd/systemd --user
FILE SYS     Used  Total     0.0   0.0     0     0    479 root   -20 S  0:00.00     0     0 radeon-crtc
/            883M  9.04G     0.0   0.0     0     0   3618 root     0 S  0:03.92     0     0 kworker/5:2
/virtual     333G   492G     0.0   0.0     0     0     33 root     0 S  0:00.00     0     0 rcuob/3
/boot        345M  1.60G     0.0   0.0     0     0    684 root   -20 S  0:00.00     0     0 ext4-rsv-conver
/home       23.4G  49.1G     0.0   0.0  222M 7.47M   3596 dboth    0 S  0:00.10     0     0 su -
_00GB-USB   42.6G   458G     0.0   0.0     0     0  24206 root     0 S  0:00.10     0     0 kworker/7:1
/stuff      77.0G  98.3G     0.0   0.0     0     0     19 root     0 S  0:00.00     0     0 rcuob/1
/tmp        47.6M  18.2G     0.0   0.0 57.4M  408K   1280 avahi    0 S  0:00.00     0     0 avahi-daemon: chroot helper
/usr        8.79G  45.7G     0.0   0.1  406M 17.8M   2108 dboth    0 S  0:03.11     0     0 /usr/libexec/kf5/kscreen_backend_la
/var        2.29G  18.2G     0.0   0.0     0     0    928 root   -20 S  0:00.00     0     0 bioset
                             0.0   0.2  735M 56.3M   1227 root     0 S  0:02.87     0     0 /usr/sbin/rsyslogd -n
SENSORS                      0.0   0.0     0     0     42 root     0 S  0:01.24     0     0 migration/5
temp1          °C     27     0.0   0.0     0     0   1131 root     0 S  0:03.62     0     0 jbd2/dm-3-8
temp2          °C     29     0.0   0.0     0     0    919 root   -20 S  0:00.00     0     0 kvm-irqfd-clean
temp1          °C     42     0.0   0.0     0     0     64 root   -20 S  0:00.00     0     0 perf
Physical id    °C     76     0.0   0.0     0     0    482 root   -20 S  0:00.00     0     0 bioset
Core 0         °C     71     0.0   0.0 23.9M 3.08M   1223 root     0 S  0:00.74     0     0 /usr/lib/systemd/systemd-logind
Core 1         °C     76     0.0   0.0     0     0  25951 root   -20 S  0:00.00     0     0 scsi_tmf_16
Core 2         °C     65     0.0   0.0     0     0    452 root   -20 S  0:00.26     0     0 kworker/6:1H
Core 3         °C     66     0.0   0.0     0     0     48 root     0 S  0:00.77     0     0 watchdog/6
SYSTIN         °C     48     0.0   0.0     0     0    480 root   -20 S  0:00.00     0     0 radeon-crtc
CPUTIN         °C     46     0.0   0.0  118M 4.84M   2949 root     0 S  0:00.20     0     0 /bin/bash
PECI Agent     °C     76     0.0   0.0 16.4M 2.54M   1208 root    19 S  0:00.50     0     0 /usr/sbin/alsactl -s -n 19 -c -E AL
PCH_CHIP_CP    °C     46     0.0   0.0  120M 4.61M   1279 chrony   0 S  0:00.80     0     0 /usr/sbin/chronyd
fan2          RPM   1483     0.0   0.0     0     0    475 root   -20 S  0:00.00     0     0 radeon-crtc
fan3          RPM    313

2016-01-29 11:25:45        No warning or critical alert detected
```

Figure 4: The glances interface with network, disk, filesystem, and sensor information.

I like this summary section better than those of the other monitors; I think it provides the right information in an easily understandable format. As with atop and htop, you can press the **1** key to toggle between a display of the individual CPU cores or a global one with all of the CPU cores as a single average as shown in Figure 4, above.

Process section

The process section displays the standard information about each of the running processes. Processes can be sorted automatically **a**, or by CPU **c**, memory **m**, name **p**, user **u**, I/O rate **i**, or time **t**. When sorted automatically processes are first sorted by the most used resource.

Glances also shows warnings and critical alerts at the very bottom of the screen, including the time and duration of the event. This can be helpful when attempting to diagnose problems when you cannot stare at the screen for hours at a time. These alert logs can be toggled on or off with the **l** command, warnings can be cleared with the **w** command while alerts and warnings can all be cleared with **x**.

It is interesting that glances is the only one of these monitors that cannot be used to either kill or renice a process. It is intended strictly as a monitor. You can use the external **kill** and **renice** commands to manipulate processes.

Sidebar

Glances has a very nice sidebar that displays information that is not available in top or htop. Atop does display some of this data, but glances is the only monitor that displays the sensors data. Sometimes it is nice to see the temperatures inside your computer. The individual modules, disk, filesystem, network, and sensors can be toggled on and off using the **d**, **f**, **n**, and **s** commands, respectively. The entire sidebar can be toggled using **2**.

Docker stats can be displayed with **D**.

Configuration

Glances does not require a configuration file to work properly. If you choose to have one, the system-wide instance of the configuration file would be located in /etc/glances/glances.conf. Individual users can have a local instance at ~/.config/glances/glances.conf which will override the global configuration. The primary purpose of these configuration files is to set thresholds for warnings and critical alerts. There is no way I can find to make other configuration changes—such as sidebar modules or the CPU displays—permanent. It appears that you must reconfigure those items every time you start glances.

There is a document, /usr/share/doc/glances/glances-doc.html, that provides a great deal of information about using glances, and it explicitly states that you can use the configuration file to configure which modules are displayed. However, neither the information given nor the examples describe just how to do that.

Conclusion

Be sure to read the man pages for each of these monitors because there is a large amount of information about configuring and interacting with them. Also use the **h** key for help in interactive mode. This help can provide you with information about selecting and sorting the columns of data, setting the update interval and much more.

These programs can tell you a great deal when you are looking for the cause of a problem. They can tell you when a process, and which one, is sucking up CPU time, whether there is enough free memory, whether processes are stalled while waiting for I/O such as disk or network access to complete, and much more.

I strongly recommend that you spend time watching these monitoring programs while they run on a system that is functioning normally so you will be able to differentiate those things that may be abnormal while you are looking for the cause of a problem.

You should also be aware that the act of using these monitoring tools alters the system's use of resources including memory and CPU time. top and most of these monitors use perhaps 2% or 3% of a system's CPU time. glances has much more impact than the others and can use between 10% and 20% of CPU time. Be sure to consider this when choosing your tools.

I had originally intended to include SAR (System Activity Reporter) in this article but as this article grew longer it also became clear to me that SAR is significantly different from these monitoring tools and deserves to have a separate article. So with that in mind, I plan to write an article on SAR and the /proc filesystem, and a third article on how to use all of these tools to locate and resolve problems.

Links

[1] https://www.linuxjournal.com/article/9001?page=0,0
[2] https://en.wikipedia.org/wiki/Signal_(IPC)

Author

David Both is an Open Source Software and GNU/Linux advocate, trainer, writer, and speaker who lives in Raleigh North Carolina. He is a strong proponent of and evangelist for the "Linux Philosophy."

David has been in the IT industry for nearly 50 years. 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 over 20 years.

David prefers to purchase the components and build his own computers from scratch to ensure that each new computer meets his exacting specifications. His primary workstation is an ASUS TUF X299 motherboard and an Intel i9 CPU with 16 cores (32 CPUs) and 64GB of RAM in a ThermalTake Core X9 case.

David has written articles for magazines including, Linux Magazine, Linux Journal. His article "Complete Kickstart," co-authored with a colleague at Cisco, was ranked 9th in the Linux Magazine Top Ten Best System Administration Articles list for 2008. David currently writes prolifically for OpenSource.com and Enable SysAdmin.

David currently has one book published, "The Linux Philosophy for SysAdmins." and is now is working on his next project, "Using and Administering Linux: Zero to SysAdmin," a self-study training course in three volumes that is scheduled for release in late 2019."

David can be reached at LinuxGeek46@both.org or on Twitter @LinuxGeek46.

5 projects for Raspberry Pi at home

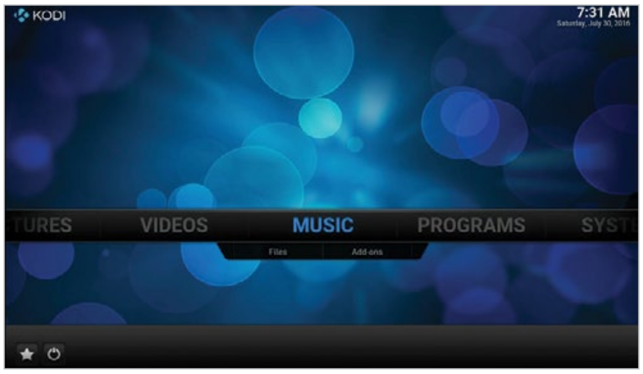
BY BEN NUTTALL

THE RASPBERRY PI [1] computer can be used in all kinds of settings and for a variety of purposes. It obviously has a place in education for helping students with learning programming and maker skills in the classroom and the hackspace, and it has plenty of industrial applications in the workplace and in factories. I'm going to introduce five projects you might want to build in your own home.

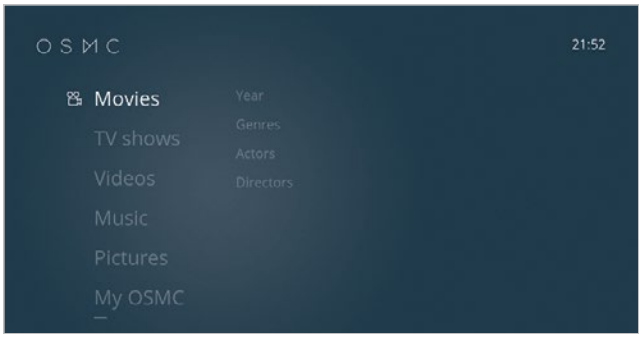
Media center

One of the most common uses for Raspberry Pi in people's homes is behind the TV running media center software serving multimedia files. It's easy to set this up, and the Raspberry Pi provides plenty of GPU (Graphics Processing Unit) power to render HD TV shows and movies to your big screen TV. Kodi [2] (formerly XBMC) on a Raspberry Pi is a great way to playback any media you have on a hard drive or network-attached storage. You can also install a plugin to play YouTube videos.

There are a few different options available, most prominently OSMC [3] (Open Source Media Center) and LibreELEC [4], both based on Kodi. They both perform well at playing media content, but OSMC has a more visually appearing user interface, while LibreElec is much more lightweight. All you have to do is choose a distribution, download the image and install on an SD card (or just use NOOBS [5]), boot it up, and you're ready to go.



LibreElec; Raspberry Pi Foundation, CC BY-SA



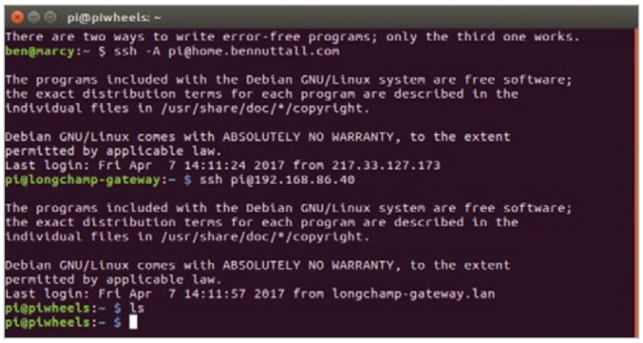
OSMC.tv, Copyright, Used with permission

Before proceeding you'll need to decide which Raspberry Pi model to use [6]. These distributions will work on any Pi (1, 2, 3, or Zero), and video playback will essentially be matched on each of these. Apart from the Pi 3 (and Zero W) having built-in Wi-Fi, the only noticeable difference is the reaction speed of the user interface, which will be much faster on a Pi 3. A Pi 2 will not be much slower, so that's fine if you don't need Wi-Fi, but the Pi 3 will noticeably outperform the Pi 1 and Zero when it comes to flicking through the menus.

SSH gateway

If you want to be able to access computers and devices on your home network from outside over the internet, you have to open up ports on those devices to allow outside traffic. Opening ports to the internet is a security risk, meaning you're always at risk of attack, misuse, or any kind of unauthorized access. However, if you install a Raspberry Pi on your network and set up port forwarding to allow only SSH access to that Pi, you can use that as a secure gateway to hop onto other Pis and PCs on the network.

Most routers allow you to configure port-forwarding rules. You'll need to give your Pi a fixed internal IP address and set up port 22 on your router to map to port 22 on your Raspberry Pi. If your ISP provides you with a static IP address, you'll be able to SSH into it with this as the host address (for example, **ssh pi@123.45.56.78**). If you have a domain name, you can configure a subdomain to point to this IP address, so you don't have to remember it (for example, **ssh** pi@home.mydomain.com).



```
pi@piwheels: ~
There are two ways to write error-free programs; only the third one works.
ben@marcy:~ $ ssh -A pi@home.bennuttall.com

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr  7 14:11:24 2017 from 217.33.127.173
pi@longchamp-gateway:~ $ ssh pi@192.168.86.40

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr  7 14:11:57 2017 from longchamp-gateway.lan
pi@piwheels:~ $ ls
pi@piwheels:~ $ 
```

However, if you're going to expose a Raspberry Pi to the internet, you should be very careful not to put your network at risk. There are a few simple procedures you can follow to make it sufficiently secure:

1. Most people suggest you change your login password (which makes sense, seeing as the default password "raspberry" is well known), but this does not protect against brute-force attacks. You could change your password and add a two-factor authentication (so you need your password and a time-dependent passcode generated by your phone), which is more secure. However, I believe the best way to secure your Raspberry Pi from intruders is to disable "password authentication" [7] in your SSH configuration, so you allow only SSH key access. This means that anyone trying to SSH in by guessing your password will never succeed. Only with your private SSH key can anyone gain access. Similarly, most people suggest changing the SSH port from the default 22 to something unexpected, but a simple Nmap [8] of your IP address will reveal your true SSH port.
2. Ideally, you would not run much in the way of other software on this Pi, so you don't end up accidentally exposing anything else. If you want to run other software, you might be better running it on another Pi on the network that is not exposed to the internet. Ensure that you keep your packages up to date by upgrading regularly, particularly the **openssh-server** package, so that any security vulnerabilities are patched.
3. Install sshblack [9] or fail2ban [10] to blacklist any users who seem to be acting maliciously, such as attempting to brute force your SSH password.

Once you've secured your Raspberry Pi and put it online, you'll be able to log in to your network from anywhere in the world. Once you're on your Raspberry Pi, you can SSH into other devices on the network using their local IP address (for example, 192.168.1.31). If you have passwords on these devices, just use the password. If they're also SSH-key-only, you'll need to ensure your key is forwarded over SSH by using the **-A** flag: **ssh -A pi@123.45.67.89**.

CCTV / pet camera

Another great home project is to set up a camera module to take photos or stream video, capture and save files, or streamed internally or to the internet. There are many reasons you might want to do this, but two common use cases are for a homemade security camera or to monitor a pet.

The Raspberry Pi camera module [11] is a brilliant accessory. It provides full HD photo and video, lots of advanced configuration, and is easy to program [12]. The infrared camera [13] is ideal for this kind of use, and with an infrared LED (which the Pi can control) you can see in the dark!

If you want to take still images on a regular basis to keep an eye on things, you can just write a short Python [14] script or use the command line tool raspistill [15], and schedule it to recur in Cron [16]. You might want to have it save them to Dropbox [17] or another web service, upload them to a web server, or you can even create a web app [18] to display them.

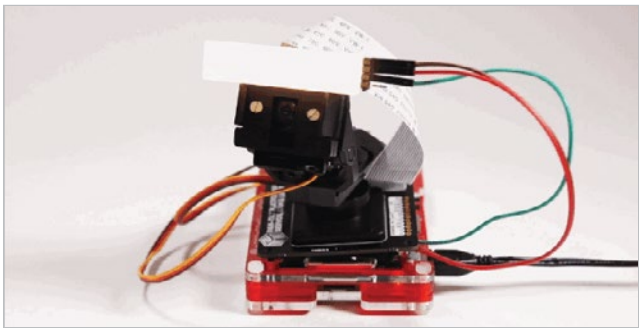
If you want to stream video, internally or externally, that's really easy, too. A simple MJPEG (Motion JPEG) example is provided in the picamera [19] documentation [20] (under "web streaming"). Just download or copy that code into a file, run it and visit the Pi's IP address at port 8000, and you'll see your camera's output live.

A more advanced streaming project, pistreaming [21], is available, which uses JSMpeg [22] (a JavaScript video player) with the web server and a websocket for the camera stream running separately. This method is more performant and is just as easy to get running as the previous example, but there is more code involved and if set up to stream on the internet, requires you to open two ports.

Once you have web streaming set up, you can position the camera where you want it. I have one set up to keep an eye on my pet tortoise:

Ben Nuttall, CC BY-SA

If you want to be able to control where the camera actually points, you can do so using servos. A neat solution is to use Pimoroni's Pan-Tilt HAT [23], which allows you to move the camera easily in two dimensions. To integrate this with pistreaming, see the project's pantilthat branch [24].


Pimoroni.com, Copyright, Used with permission

If you want to position your Pi outside, you'll need a waterproof enclosure and some way of getting power to the Pi. PoE (Power-over-Ethernet) cables can be a good way of achieving this.

Home automation and IoT

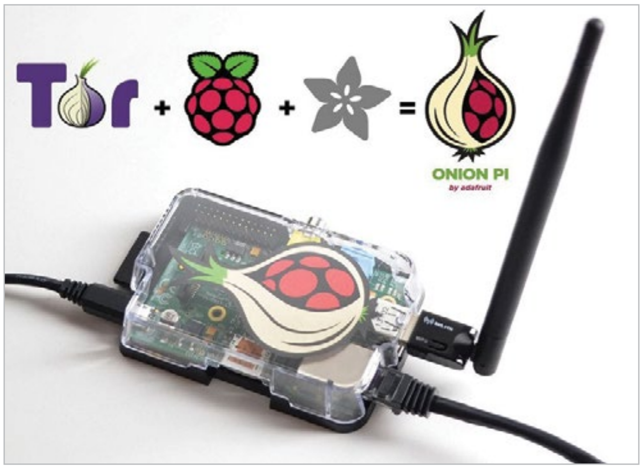
It's 2017 and there are internet-connected devices everywhere, especially in the home. Our lightbulbs have Wi-Fi, our toasters are smarter than they used to be, and our tea kettles are at risk of attack from Russia. As long as you keep your devices secure, or don't connect them to the internet if they don't need to be, then you can make great use of IoT devices to automate tasks around the home.

There are plenty of services you can buy or subscribe to, like Nest Thermostat or Philips Hue lightbulbs, which allow you to control your heating or your lighting from your phone, respectively—whether you're inside or away from home. You can use a Raspberry Pi to boost the power of these kinds of devices by automating interactions with them according to a set of rules involving timing or even sensors. One thing you can't do with Philips Hue is have the lights come on when you enter the room, but with a Raspberry Pi and a motion sensor, you can use a Python API to turn on the lights. Similarly, you can configure your Nest to turn on the heating when you're at home, but what if you only want it to turn on if there's at least two people home? Write some Python code to check which phones are on the network and if there are at least two, tell the Nest to turn on the heat.

You can do a great deal more without integrating with existing IoT devices and with only using simple components. A homemade burglar alarm, an automated chicken coop door opener, a night light, a music box, a timed heat lamp, an automated backup server, a print server, or whatever you can imagine.

Tor proxy and blocking ads

Adafruit's Onion Pi [25] is a Tor [26] proxy that makes your web traffic anonymous, allowing you to use the internet free of snoopers and any kind of surveillance. Follow Adafruit's tutorial on setting up Onion Pi and you're on your way to a peaceful anonymous browsing experience.



Onion-pi from Adafruit, Copyright, Used with permission

Pi-holeYou can install a Raspberry Pi on your network that intercepts all web traffic and filters out any advertising. Simply download the Pi-hole [27] software onto the Pi, and all devices on your network will be ad-free (it even blocks in-app ads on your mobile devices).

Links

[1] https://www.raspberrypi.org/
[2] https://kodi.tv/
[3] https://osmc.tv/
[4] https://libreelec.tv/
[5] https://www.raspberrypi.org/downloads/noobs/
[6] https://opensource.com/life/16/10/which-raspberry-pi-should-you-choose-your-project
[7] https://stackoverflow.com/questions/20898384/ssh-disable-password-authentication
[8] https://nmap.org/
[9] http://www.pettingers.org/code/sshblack.html
[10] https://www.fail2ban.org/wiki/index.php/Main_Page
[11] https://www.raspberrypi.org/products/camera-module-v2/
[12] https://opensource.com/life/15/6/raspberry-pi-camera-projects
[13] https://www.raspberrypi.org/products/pi-noir-camera-v2/
[14] https://picamera.readthedocs.io/en/release-1.13/
[15] https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md
[16] https://www.raspberrypi.org/documentation/linux/usage/cron.md
[17] https://github.com/RZRZR/plant-cam
[18] https://github.com/bennuttall/bett-bot
[19] https://picamera.readthedocs.io/en/release-1.13/recipes2.html#web-streaming
[20] https://github.com/waveform80/pistreaming
[21] https://github.com/waveform80/pistreaming
[22] https://jsmpeg.com/
[23] https://shop.pimoroni.com/products/pan-tilt-hat?variant=22408353287
[24] https://github.com/waveform80/pistreaming/tree/pantilthat
[25] https://learn.adafruit.com/onion-pi/overview
[26] https://www.torproject.org/
[27] https://pi-hole.net/

Author

Ben Nuttall is the Raspberry Pi Community Manager. In addition to his work for the Raspberry Pi Foundation, he's into free software, maths, kayaking, GitHub, Adventure Time, and Futurama. Follow Ben on Twitter @ben_nuttall.

Linux vs. Unix: What's the difference?

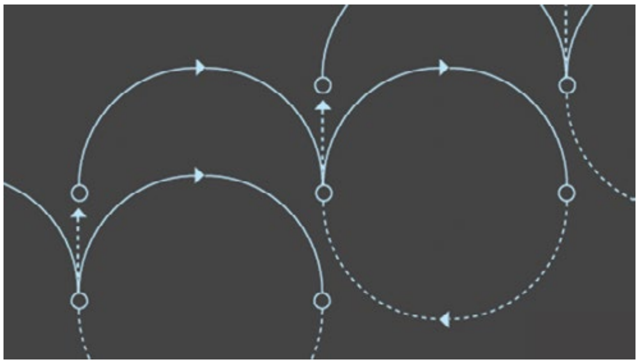
BY PHIL ESTES

Dive into the differences between these two operating systems that share much of the same heritage and many of the same goals.

IF YOU ARE A SOFTWARE DEVELOPER in your 20s or 30s, you've grown up in a world dominated by Linux. It has been a significant player in the data center for decades, and while it's hard to find definitive operating system market share reports, Linux's share of data center operating systems could be as high as 70%, with Windows variants carrying nearly all the remaining percentage. Developers using any major public cloud can expect the target system will run Linux. Evidence that Linux is everywhere has grown in recent years when you add in Android and Linux-based embedded systems in smartphones, TVs, automobiles, and many other devices.

Even so, most software developers, even those who have grown up during this venerable "Linux revolution" have at least heard of Unix. It sounds similar to Linux, and you've probably heard people use these terms interchangeably. Or maybe you've heard Linux called a "Unix-like" operating system.

So, what is this Unix? The caricatures speak of wizard-like "graybeards" sitting behind glowing green screens, writing C code and shell scripts, powered by old-fashioned, drip-brewed coffee. But Unix has a much richer history beyond those bearded C programmers from the 1970s. While articles detailing the history of Unix and "Unix vs. Linux" comparisons abound, this article will offer a high-level background and a list of major differences between these complementary worlds.



Unix's beginnings

The history of Unix begins at AT&T Bell Labs in the late 1960s with a small team of programmers looking to write a multi-tasking, multi-user operating system for the PDP-7. Two of the most notable members of this team at the Bell Labs research facility were Ken Thompson and Dennis Ritchie. While many of Unix's concepts were derivative of its predecessor (Multics [1]), the Unix team's decision early in the 1970s to rewrite this small operating system in the C language is what separated Unix from all others. At the time, operating systems were rarely, if ever, portable. Instead, by nature of their design and low-level source language, operating systems were tightly linked to the hardware platform for which they had been authored. By refactoring Unix on the C programming language, Unix could now be ported to many hardware architectures.

In addition to this new portability, which allowed Unix to quickly expand beyond Bell Labs to other research, academic, and even commercial uses, several key of the operating system's design tenets were attractive to users and programmers. For one, Ken Thompson's Unix philosophy [2] became a powerful model of modular software design and computing. The Unix philosophy recommended utilizing small, purpose-built programs in combination to do complex overall tasks. Since Unix was designed around files and pipes, this model of "piping" inputs and outputs of programs together into a linear set of operations on the input is still in vogue today. In fact, the current cloud functions-as-a-service (FaaS)/serverless computing model owes much of its heritage to the Unix philosophy.

Rapid growth and competition

Through the late 1970s and 80s, Unix became the root of a family tree that expanded across research, academia, and a growing commercial Unix operating system business. Unix was not open source software, and the Unix source code was licensable via agreements with its owner, AT&T. The first known software license was sold to the University of Illinois in 1975.

Unix grew quickly in academia, with Berkeley becoming a significant center of activity, given Ken Thompson's sabbatical there in the '70s. With all the activity around Unix at Berkeley, a new delivery of Unix software was born: the Berkeley Software Distribution, or BSD. Initially, BSD was not an alternative to AT&T's Unix, but an add-on with additional software and capabilities. By the time 2BSD (the Second Berkeley Software Distribution) arrived in 1979, Bill Joy, a Berkeley grad student, had added now-famous programs such as vi and the C shell (/bin/csh).

In addition to BSD, which became one of the most popular branches of the Unix family, Unix's commercial offerings exploded through the 1980s and into the '90s with names like HP-UX, IBM's AIX, Sun's Solaris, Sequent, and Xenix. As the branches grew from the original root, the "Unix wars" [3] began, and standardization became a new focus for the community. The POSIX standard was born in 1988, as well as other standardization follow-ons via The Open Group into the 1990s.

Around this time AT&T and Sun released System V Release 4 (SVR4), which was adopted by many commercial vendors. Separately, the BSD family of operating systems had grown over the years, leading to some open source variations that were released under the now-familiar BSD license [4]. This included FreeBSD, OpenBSD, and NetBSD, each with a slightly different target market in the Unix server industry. These Unix variants continue to have some usage today, although many have seen their server market share dwindle into the single digits (or lower). BSD may have the largest install base of any modern Unix system today. Also, every Apple Mac hardware unit shipped in recent history can be claimed by BSD, as its OS X (now macOS) operating system is a BSD-derivative.

While the full history of Unix and its academic and commercial variants could take many more pages, for the sake of our article focus, let's move on to the rise of Linux.

Enter Linux

What we call the Linux operating system today is really the combination of two efforts from the early 1990s. Richard Stallman was looking to create a truly free and open source alternative to the proprietary Unix system. He was working on the utilities and programs under the name GNU, a recursive acronym meaning "GNU's not Unix!" Although there was a kernel project underway, it turned out to be difficult going, and without a kernel, the free and open source operating system dream could not be realized. It was Linus Torvald's work—producing a working and viable kernel that he called Linux—that brought the complete operating system to life. Given that Linus was using several GNU tools (e.g., the GNU Compiler Collection, or GCC [5]), the marriage of the GNU tools and the Linux kernel was a perfect match.

Linux distributions came to life with the components of GNU, the Linux kernel, MIT's X-Windows GUI, and other BSD components that could be used under the open source BSD license. The early popularity of distributions like Slackware and then Red Hat gave the "common PC user" of the 1990s access to the Linux operating system and, with it, many of the proprietary Unix system capabilities and utilities they used in their work or academic lives.

Because of the free and open source standing of all the Linux components, anyone could create a Linux distribution with a bit of effort, and soon the total number of distros reached into the hundreds. Today, distrowatch.com [6] lists 312 unique Linux distributions available in some form. Of course, many developers utilize Linux either via cloud providers or by using popular free distributions like Fedora, Canonical's Ubuntu, Debian, Arch Linux, Gentoo, and many other variants. Commercial Linux offerings, which provide support on top of the free and open source components, became viable as many enterprises, including IBM, migrated from proprietary Unix to offering middleware and software solutions atop Linux. Red Hat built a model of commercial support around Red Hat Enterprise Linux, as did German provider SUSE with SUSE Linux Enterprise Server (SLES).

Comparing Unix and Linux

So far, we've looked at the history of Unix and the rise of Linux and the GNU/Free Software Foundation underpinnings of a free and open source alternative to Unix. Let's examine the differences between these two operating systems that share much of the same heritage and many of the same goals.

From a user experience perspective, not very much is different! Much of the attraction of Linux was the operating system's availability across many hardware architectures (including the modern PC) and ability to use tools familiar to Unix system administrators and users.

Because of POSIX standards and compliance, software written on Unix could be compiled for a Linux operating system with a usually limited amount of porting effort. Shell scripts could be used directly on Linux in many cases. While some tools had slightly different flag/command-line options between Unix and Linux, many operated the same on both.

One side note is that the popularity of the macOS hardware and operating system as a platform for development that mainly targets Linux may be attributed to the BSD-like macOS operating system. Many tools and scripts meant for a Linux system work easily within the macOS terminal.

Many open source software components available on Linux are easily available through tools like Homebrew [7].

The remaining differences between Linux and Unix are mainly related to the licensing model: open source vs. proprietary, licensed software. Also, the lack of a common kernel within Unix distributions has implications for software and hardware vendors. For Linux, a vendor can create a device driver for a specific hardware device and expect that, within reason, it will operate across most distributions. Because of the commercial and academic branches of the Unix tree, a vendor might have to write different drivers for variants of Unix and have licensing and other concerns related to access to an SDK or a distribution model for the software as a binary device driver across many Unix variants.

As both communities have matured over the past decade, many of the advancements in Linux have been adopted in the Unix world. Many GNU utilities were made available as add-ons for Unix systems where developers wanted features from GNU programs that aren't part of Unix. For example, IBM's AIX offered an AIX Toolbox for Linux Applications with hundreds of GNU software packages (like Bash, GCC, OpenLDAP, and many others) that could be added to an AIX installation to ease the transition between Linux and Unix-based AIX systems.

Proprietary Unix is still alive and well and, with many major vendors promising support for their current releases well into the 2020s, it goes without saying that Unix will be around for the foreseeable future. Also, the BSD branch of the Unix tree is open source, and NetBSD, OpenBSD, and FreeBSD all have strong user bases and open source communities that may not be as visible or active as Linux, but are holding their own in recent server share reports, with well above the proprietary Unix numbers in areas like web serving.

Where Linux has shown a significant advantage over proprietary Unix is in its availability across a vast number of hardware platforms and devices. The Raspberry Pi, popular with hobbyists and enthusiasts, is Linux-driven and has opened the door for an entire spectrum of IoT devices running Linux. We've already mentioned Android devices, autos (with Automotive Grade Linux), and smart TVs, where Linux has large market share. Every cloud provider on the planet offers virtual servers running Linux, and many of today's most popular cloud-native stacks are Linux-based, whether you're talking about container runtimes or Kubernetes or many of the serverless platforms that are gaining popularity.

One of the most revealing representations of Linux's ascendancy is Microsoft's transformation in recent years. If you told software developers a decade ago that the Windows operating system would "run Linux" in 2016, most of them would have laughed hysterically. But the existence and popularity of the Windows Subsystem for Linux (WSL), as well as more recently announced capabilities like the Windows port of Docker, including LCOW (Linux containers on Windows) support, are evidence of the impact that Linux has had—and clearly will continue to have—across the software world.

Links

[1] https://en.wikipedia.org/wiki/Multics
[2] https://en.wikipedia.org/wiki/Unix_philosophy
[3] https://en.wikipedia.org/wiki/Unix_wars
[4] https://en.wikipedia.org/wiki/BSD_licenses
[5] https://en.wikipedia.org/wiki/GNU_Compiler_Collection
[6] https://distrowatch.com/
[7] https://brew.sh/

Author

Phil is a Distinguished Engineer & CTO, Container and Linux OS Architecture Strategy for the IBM Watson and Cloud Platform division. Phil is currently an OSS maintainer in the Docker (now Moby) engine project, the CNCF containerd project, and is a member of both the Open Container Initiative (OCI) Technical Oversight Board and the Moby Technical Steering Committee. Phil is a long-standing member of the Docker Captains program and has enjoyed a long relationship with key open source contributors and experts in the Docker ecosystem.

Phil is a regular speaker at industry and developer conferences as well as meetups on topics related to open source, Docker, Linux container technology and the broader container ecosystem. Phil is a recognized subject matter expert on Linux container technology and regularly assists IBM's product teams and enterprise customers in their application of container technology for their public and private cloud solutions. He maintains an active blog on container topics at https://integratedcode.us. You can find him on Twitter tweeting away as @estesp.

Create your own video streaming server with Linux

BY AARON J. PRISK

Set up a basic live streaming server on a Linux or BSD operating system.

LIVE VIDEO STREAMING is incredibly popular—and it's still growing. Platforms like Amazon's Twitch and Google's YouTube boast millions of users that stream and consume countless hours of live and recorded media. These services are often free to use but require you to have an account and generally hold your content behind advertisements. Some people don't need their videos to be available to the masses or just want more control over their content. Thankfully, with the power of open source software, anyone can set up a live streaming server.

Getting started

In this tutorial, I'll explain how to set up a basic live streaming server with a Linux or BSD operating system.

This leads to the inevitable question of system requirements. These can vary, as there are a lot of variables involved with live streaming, such as:

- **Stream quality:** Do you want to stream in high definition or will standard definition fit your needs?
- **Viewership:** How many viewers are you expecting for your videos?
- **Storage:** Do you plan on keeping saved copies of your video stream?
- **Access:** Will your stream be private or open to the world?

There are no set rules when it comes to system requirements, so I recommend you experiment and find what works best for your needs. I installed my server on a virtual machine with 4GB RAM, a 20GB hard drive, and a single Intel i7 processor core.

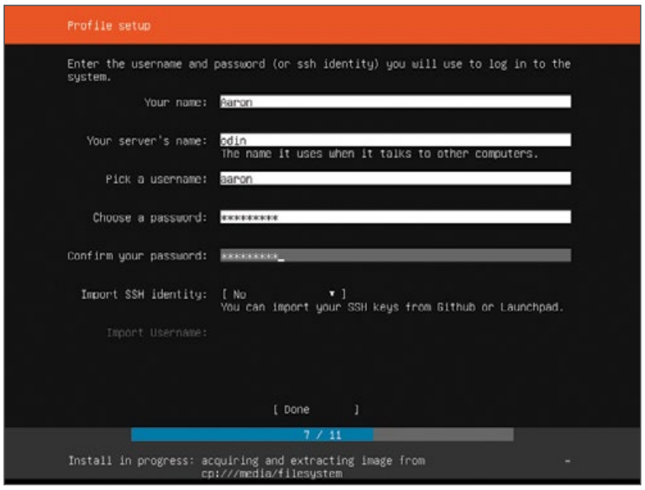
This project uses the Real-Time Messaging Protocol (RTMP) to handle audio and video streaming. There are other protocols available, but I chose RTMP because it has broad support. As open standards like WebRTC become more compatible, I would recommend that route.

It's also very important to know that "live" doesn't always mean instant. A video stream must be encoded, transferred, buffered, and displayed, which often adds delays. The delay can be shortened or lengthened depending on the type of stream you're creating and its attributes.

Setting up a Linux server

You can use many different distributions of Linux, but I prefer Ubuntu, so I downloaded the Ubuntu Server [1] edition for my operating system. If you prefer your server to have a graphical user interface (GUI), feel free to use Ubuntu Desktop [2] or one of its many flavors. Then, I fired up the Ubuntu installer on my computer or virtual machine and chose the settings that best matched my environment. Below are the steps I took.

Note: Because this is a server, you'll probably want to set some static network settings.



```
Profile setup

Enter the username and password (or ssh identity) you will use to log in to the
system.

              Your name:  Aaron

     Your server's name:  odin
                          The name it uses when it talks to other computers.

        Pick a username:  aaron

      Choose a password:  **********

  Confirm your password:  **********_

    Import SSH identity:  [ No              ▾ ]
                          You can import your SSH keys from Github or Launchpad.

       Import Username:

                          [ Done       ]

                                7 / 11

Install in progress: acquiring and extracting image from                        -
                     cp:///media/filesystem
```

After the installer finishes and your system reboots, you'll be greeted with a lovely new Ubuntu system. As with any newly installed operating system, install any updates that are available:

```
sudo apt update
sudo apt upgrade
```

This streaming server will use the very powerful and versatile Nginx web server, so you'll need to install it:

```
sudo apt install nginx
```

Then you'll need to get the RTMP module so Nginx can handle your media stream:

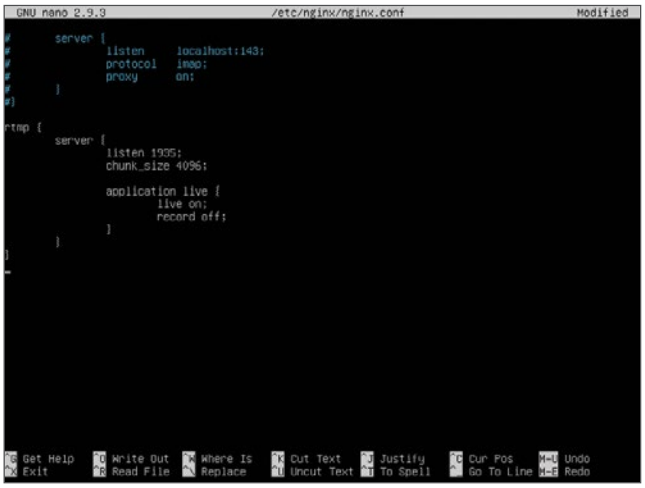
```
sudo add-apt-repository universe
sudo apt install libnginx-mod-rtmp
```

Adjust your web server's configuration so it can accept and deliver your media stream.

```
sudo nano /etc/nginx/nginx.conf
```

Scroll to the bottom of the configuration file and add the following code:

```
rtmp {
        server {
                listen 1935;
                chunk_size 4096;

                application live {
                        live on;
                        record off;
                }
        }
}
```



```
  GNU nano 2.9.3                 /etc/nginx/nginx.conf                      Modified

#       server {
#               listen     localhost:143;
#               protocol   imap;
#               proxy      on;
#       }
#}

rtmp {
        server {
                listen 1935;
                chunk_size 4096;

                application live {
                        live on;
                        record off;
                }
        }
}

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify   ^C Cur Pos     M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line  M-E Redo
```

Save the config. Because I'm a heretic, I use Nano [3] for editing configuration files. In Nano, you can save your config by pressing **Ctrl+X**, **Y**, and then **Enter**.

This is a very minimal config that will create a working streaming server. You'll add to this config later, but this is a great starting point.

However, before you can begin your first stream, you'll need to restart Nginx with its new configuration:

```
sudo systemctl restart nginx
```

Setting up a BSD server

If you're of the "beastie" persuasion, getting a streaming server up and running is also devilishly easy.

Head on over to the FreeBSD [4] website and download the latest release. Fire up the FreeBSD installer on your computer or virtual machine and go through the initial steps and choose settings that best match your environment. Since this is a server, you'll likely want to set some static network settings.

After the installer finishes and your system reboots, you should have a shiny new FreeBSD system. Like any other freshly installed system, you'll likely want to get everything updated (from this step forward, make sure you're logged in as root):

```
pkg update
pkg upgrade
```

I install Nano for editing configuration files:

```
pkg install nano
```

This streaming server will use the very powerful and versatile Nginx web server. You can build Nginx using the excellent *ports* system that FreeBSD boasts.

First, update your ports tree:

```
portsnap fetch
portsnap extract
```

Browse to the Nginx ports directory:

```
cd /usr/ports/www/nginx
```

And begin building Nginx by running:

```
make install
```

You'll see a screen asking what modules to include in your Nginx build. For this project, you'll need to add the RTMP module. Scroll down until the RTMP module is selected and press **Space**. Then Press **Enter** to proceed with the rest of the build and installation.

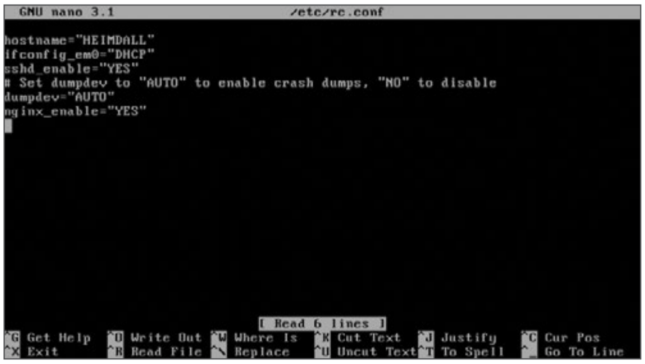
Once Nginx has finished installing, it's time to configure it for streaming purposes.

First, add an entry into **/etc/rc.conf** to ensure the Nginx server starts when your system boots:

```
nano /etc/rc.conf
```

Add this text to the file:

```
nginx_enable="YES"
```



```
  GNU nano 3.1                       /etc/rc.conf

hostname="HEIMDALL"
ifconfig_em0="DHCP"
sshd_enable="YES"
# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="AUTO"
nginx_enable="YES"

                                [ Read 6 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Next, create a webroot directory from where Nginx will serve its content. I call mine **stream**:

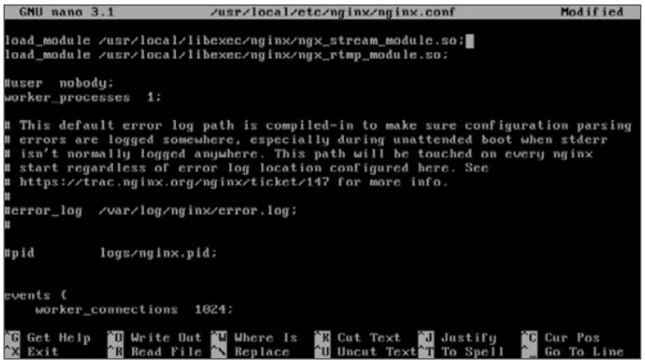
```
cd /usr/local/www/
mkdir stream
chmod -R 755 stream/
```

Now that you have created your stream directory, configure Nginx by editing its configuration file:

```
nano /usr/local/etc/nginx/nginx.conf
```

Load your streaming modules at the top of the file:

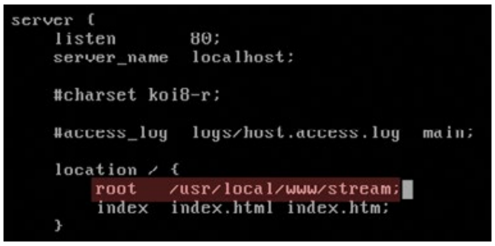
```
load_module /usr/local/libexec/nginx/ngx_stream_module.so;
load_module /usr/local/libexec/nginx/ngx_rtmp_module.so;
```



```
  GNU nano 3.1              /usr/local/etc/nginx/nginx.conf              Modified

load_module /usr/local/libexec/nginx/ngx_stream_module.so;
load_module /usr/local/libexec/nginx/ngx_rtmp_module.so;

#user  nobody;
worker_processes  1;

# This default error log path is compiled-in to make sure configuration parsing
# errors are logged somewhere, especially during unattended boot when stderr
# isn't normally logged anywhere. This path will be touched on every nginx
# start regardless of error log location configured here. See
# https://trac.nginx.org/nginx/ticket/147 for more info.
#
#error_log  /var/log/nginx/error.log;
#

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Under the **Server** section, change the webroot location to match the one you created earlier:

```
Location / {
root /usr/local/www/stream
}
```



```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   /usr/local/www/stream;
        index  index.html index.htm;
    }
```

And finally, add your RTMP settings so Nginx will know how to handle your media streams:

```
rtmp {
        server {
                listen 1935;
                chunk_size 4096;

                application live {
                        live on;
                        record off;
                }
        }
}
```

Save the config. In Nano, you can do this by pressing **Ctrl+X**, **Y**, and then **Enter**.

As you can see, this is a very minimal config that will create a working streaming server. Later, you'll add to this config, but this will provide you with a great starting point.

However, before you can begin your first stream, you'll need to restart Nginx with its new config:

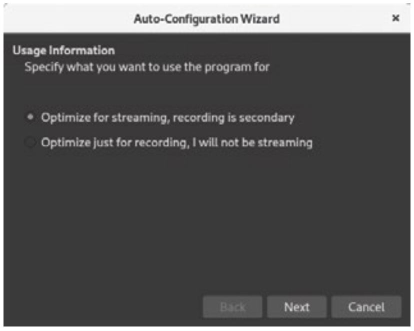
```
service nginx restart
```

Set up your streaming software

Broadcasting with OBS

Now that your server is ready to accept your video streams, it's time to set up your streaming software. This tutorial uses the powerful and open source Open Broadcast Studio (OBS).

Head over to the OBS website [5] and find the build for your operating system and install it. Once OBS launches, you should see a first-time-run wizard that will help you configure OBS with the settings that best fit your hardware.


Auto-Configuration Wizard
Usage Information
Specify what you want to use the program for
Optimize for streaming, recording is secondary
Optimize just for recording, I will not be streaming
Back
Next
Cancel

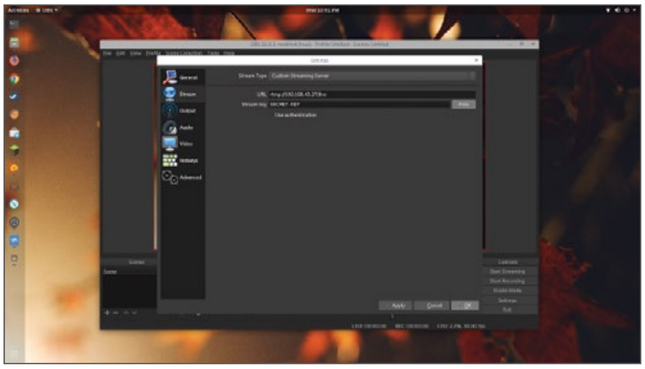
OBS isn't capturing anything because you haven't supplied it with a source. For this tutorial, you'll just capture your desktop for the stream. Simply click the **+** button under **Source**, choose **Screen Capture**, and select which desktop you want to capture.

Click OK, and you should see OBS mirroring your desktop.

Now it's time to send your newly configured video stream to your server. In OBS, click **File > Settings**. Click on the **Stream** section, and set **Stream Type** to **Custom Streaming Server**.

In the URL box, enter the prefix **rtmp://** followed the IP address of your streaming server followed by **/live**. For example, **rtmp://IP-ADDRESS/live**.

Next, you'll probably want to enter a Stream key—a special identifier required to view your stream. Enter whatever key you want (and can remember) in the **Stream key** box.



Stream Type: Custom Streaming Server

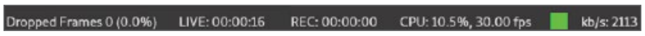
URL: rtmp://192.168.43.27/live

Stream key: SECRET-KEY

Click **Apply** and then **OK**.

Now that OBS is configured to send your stream to your server, you can start your first stream. Click **Start Streaming**.

If everything worked, you should see the button change to **Stop Streaming** and some bandwidth metrics will appear at the bottom of OBS.



Dropped Frames 0 (0.0%) LIVE: 00:00:16 REC: 00:00:00 CPU: 10.5%, 30.00 fps 🟩 kb/s: 2113

If you receive an error, double-check Stream Settings in OBS for misspellings. If everything looks good, there could be another issue preventing it from working.

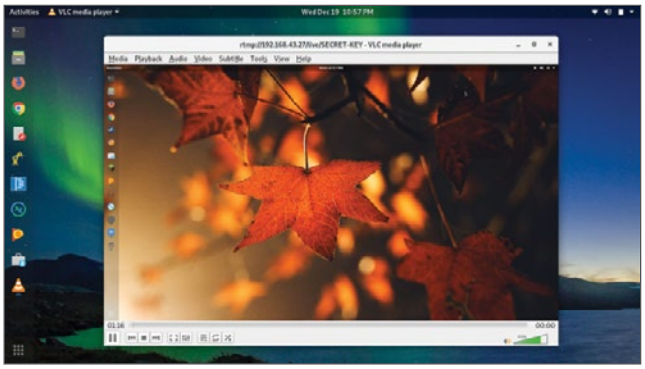
Viewing your stream

A live video isn't much good if no one is watching it, so be your first viewer!

There are a multitude of open source media players that support RTMP, but the most well-known is probably VLC media player [6].

After you install and launch VLC, open your stream by clicking on **Media > Open Network Stream**. Enter the path to your stream, adding the Stream Key you set up in OBS, then click **Play**. For example, **rtmp://IP-ADDRESS/live/SECRET-KEY**.

You should now be viewing your very own live video stream!



rtmp://192.168.43.27/live/SECRET-KEY - VLC media player

Where to go next?

This is a very simple setup that will get you off the ground. Here are two other features you likely will want to use.

- **Limit access:** The next step you might want to take is to limit access to your server, as the default setup allows anyone to stream to and from the server. There are a variety of ways to set this up, such as an operating system firewall, .htaccess file [7], or even using the built-in access controls in the RTMP module [8].
- **Record streams:** This simple Nginx configuration will only stream and won't save your videos, but this is easy to add. In the Nginx config, under the RTMP section, set up the recording options and the location where you want to save your videos. Make sure the path you set exists and Nginx is able to write to it.

```
application live {
            live on;
            record all;
            record_path /var/www/html/recordings;
            record_unique on;
}
```

The world of live streaming is constantly evolving, and if you're interested in more advanced uses, there are lots of other great resources you can find floating around the internet. Good luck and happy streaming!

Links

[1] https://www.ubuntu.com/download/server
[2] https://www.ubuntu.com/download/desktop
[3] https://www.nano-editor.org/
[4] https://www.freebsd.org/
[5] https://obsproject.com/
[6] https://www.videolan.org/vlc/index.html
[7] https://httpd.apache.org/docs/current/howto/htaccess.html
[8] https://github.com/arut/nginx-rtmp-module/wiki/Directives#access

Author

School IT Director – Open Source Evangelist – Technology Enthusiast – Husband – Dad. Follow him at @AaronPrisk

What you probably didn't know about **sudo**

BY PETER CZANIK

Think you know everything about sudo? Think again.

EVERYBODY KNOWS **SUDO**, RIGHT? This tool is installed by default on most Linux systems and is available for most BSD and commercial Unix variants. Still, after talking to hundreds of **sudo** users, the most common answer I received was that **sudo** is a tool to complicate life.

There is a root user and there is the **su** command, so why have yet another tool? For many, **sudo** was just a prefix for administrative commands. Only a handful mentioned that when you have multiple administrators for the same system, you can use **sudo** logs to see who did what.

So, what is **sudo**? According to the sudo website [1]:

> "Sudo allows a system administrator to delegate authority by giving certain users the ability to run some commands as root or another user while providing an audit trail of the commands and their arguments."

By default, **sudo** comes with a simple configuration, a single rule allowing a user or a group of users to do practically anything (more on the configuration file later in this article):

```
%wheel ALL=(ALL) ALL
```

In this example, the parameters mean the following:

- The first parameter defines the members of the group.
- The second parameter defines the host(s) the group members can run commands on.
- The third parameter defines the usernames under which the command can be executed.
- The last parameter defines the applications that can be run.

So, in this example, the members of the **wheel** group can run all applications as all users on all hosts. Even this really permissive rule is useful because it results in logs of who did what on your machine.

Aliases

Of course, once it is not just you and your best friend administering a shared box, you will start to fine-tune permissions. You can replace the items in the above configuration with lists: a list of users, a list of commands, and so on. Most likely, you will copy and paste some of these lists around in your configuration.

This situation is where aliases can come handy. Maintaining the same list in multiple places is error-prone. You define an alias once and then you can use it many times. Therefore, when you lose trust in one of your administrators, you can remove them from the alias and you are done. With multiple lists instead of aliases, it is easy to forget to remove the user from one of the lists with elevated privileges.

Enable features for a certain group of users

The **sudo** command comes with a huge set of defaults. Still, there are situations when you want to override some of these. This is when you use the **Defaults** statement in the configuration. Usually, these defaults are enforced on every user, but you can narrow the setting down to a subset of users based on host, username, and so on. Here is an example that my generation of sysadmins loves to hear about: insults. These are just some funny messages for when someone mistypes a password:

```
czanik@linux-mewy:~> sudo ls
[sudo] password for root:
Hold it up to the light --- not a brain in sight!
```

```
[sudo] password for root:
My pet ferret can type better than you!
[sudo] password for root:
sudo: 3 incorrect password attempts
czanik@linux-mewy:~>
```

Because not everyone is a fan of sysadmin humor, these insults are disabled by default. The following example shows how to enable this setting only for your seasoned sysadmins, who are members of the **wheel** group:

```
Defaults !insults
Defaults:%wheel insults
```

I do not have enough fingers to count how many people thanked me for bringing these messages back.

Digest verification
There are, of course, more serious features in **sudo** as well. One of them is digest verification. You can include the digest of applications in your configuration:

```
peter ALL =
sha244:11925141bb22866afdf257ce7790bd6275feda80b3b241c108b79c88
  /usr/bin/passwd
```

In this case, **sudo** checks and compares the digest of the application to the one stored in the configuration before running the application. If they do not match, **sudo** refuses to run the application. While it is difficult to maintain this information in your configuration—there are no automated tools for this purpose—these digests can provide you with an additional layer of protection.

Session recording
Session recording is also a lesser-known feature of **sudo**. After my demo, many people leave my talk with plans to implement it on their infrastructure. Why? Because with session recording, you see not just the command name, but also everything that happened in the terminal. You can see what your admins are doing even if they have shell access and logs only show that **bash** is started.

There is one limitation, currently. Records are stored locally, so with enough permissions, users can delete their traces. Stay tuned for upcoming features.

Plugins
Starting with version 1.8, **sudo** changed to a modular, plugin-based architecture. With most features implemented as plugins, you can easily replace or extend the functionality of **sudo** by writing your own. There are both open source and commercial plugins already available for **sudo**.

In my talk, I demonstrated the **sudo_pair** plugin, which is available on GitHub [2]. This plugin is developed in Rust, meaning that it is not so easy to compile, and it is even more difficult to distribute the results. On the other hand, the plugin provides interesting functionality, requiring a second admin to approve (or deny) running commands through **sudo**. Not just that, but sessions can be followed on-screen and terminated if there is suspicious activity.

In a demo I did during a recent talk at the All Things Open conference, I had the infamous:

```
czanik@linux-mewy:~> sudo  rm -fr /
```

command displayed on the screen. Everybody was holding their breath to see whether my laptop got destroyed, but it survived.

Logs
As I already mentioned at the beginning, logging and alerting is an important part of **sudo**. If you do not check your **sudo** logs regularly, there is not much worth in using **sudo**. This tool alerts by email on events specified in the configuration and logs all events to **syslog**. Debug logs can be turned on and used to debug rules or report bugs.

Alerts
Email alerts are kind of old-fashioned now, but if you use **syslog-ng** for collecting your log messages, your **sudo** log messages are automatically parsed. You can easily create custom alerts and send those to a wide variety of destinations, including Slack, Telegram, Splunk, or Elasticsearch. You can learn more about this feature from my blog on syslong-ng.com [3].

Configuration
We talked a lot about **sudo** features and even saw a few lines of configuration. Now, let's take a closer look at how **sudo** is configured. The configuration itself is available in **/etc/sudoers**, which is a simple text file. Still, it is not recommended to edit this file directly. Instead, use **visudo**, as this tool also does syntax checking. If you do not like vi, you can change which editor to use by pointing the **EDITOR** environment variable at your preferred option.

Before you start editing the **sudo** configuration, make sure that you know the root password. (Yes, even on Ubuntu, where root does not have a password by default.) While **visudo** checks the syntax, it is easy to create a syntactically correct configuration that locks you out of your system.

When you have a root password at hand in case of an emergency, you can start editing your configuration. When it comes to the **sudoers** file, there is one important thing to remember: This file is read from top to bottom, and the last setting wins. What this fact means for you is that you should start with generic settings and place exceptions at the end, otherwise exceptions are overridden by the generic settings.

You can find a simple **sudoers** file below, based on the one in CentOS, and add a few lines we discussed previously:

```
Defaults !visiblepw
Defaults always_set_home
Defaults match_group_by_gid
Defaults always_query_group_plugin
Defaults env_reset
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE
  KDEDIR LS_COLORS"
Defaults env_keep += "MAIL PS1 PS2 QTDIR USERNAME
  LANG LC_ADDRESS LC_CTYPE"
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
Defaults:%wheel insults
Defaults !insults
Defaults log_output
```

This file starts by changing a number of defaults. Then come the usual default rules: The **root** user and members of the **wheel** group have full permissions over the machine. Next, we enable insults for the **wheel** group, but disable them for everyone else. The last line enables session recording.

The above configuration is syntactically correct, but can you spot the logical error? Yes, there is one: Insults are disabled for everyone since the last, generic setting overrides the previous, more specific setting. Once you switch the two lines, the setup works as expected: Members of the **wheel** group receive funny messages, but the rest of the users do not receive them.

Configuration management

Once you have to maintain the **sudoers** file on multiple machines, you will most likely want to manage your configuration centrally. There are two major open source possibilities here. Both have their advantages and drawbacks.

You can use one of the configuration management applications that you also use to configure the rest of your infrastructure. Red Hat Ansible, Puppet, and Chef all have modules to configure **sudo**. The problem with this approach is that updating configurations is far from real-time. Also, users can still edit the **sudoers** file locally and change settings.

The **sudo** tool can also store its configuration in LDAP. In this case, configuration changes are real-time and users cannot mess with the **sudoers** file. On the other hand, this method also has limitations. For example, you cannot use aliases or use **sudo** when the LDAP server is unavailable.

New features

There is a new version of **sudo** right around the corner. Version 1.9 will include many interesting new features. Here are the most important planned features:

- A recording service to collect session recordings centrally, which offers many advantages compared to local storage:
  - It is more convenient to search in one place.
  - Recordings are available even if the sender machine is down.
  - Recordings cannot be deleted by someone who wants to delete their tracks.
- The **audit** plugin does not add new features to **sudoers**, but instead provides an API for plugins to easily access any kind of **sudo** logs. This plugin enables creating custom logs from **sudo** events using plugins.
- The **approval** plugin enables session approvals without using third-party plugins.
- And my personal favorite: Python support for plugins, which enables you to easily extend **sudo** using Python code instead of coding natively in C.

Conclusion

I hope this article proved to you that **sudo** is a lot more than just a simple prefix. There are tons of possibilities to fine-tune permissions on your system. You cannot just fine-tune permissions, but also improve security by checking digests. Session recordings enable you to check what is happening on your systems. You can also extend the functionality of **sudo** using plugins, either using something already available or writing your own. Finally, given the list of upcoming features you can see that even if **sudo** is decades old, it is a living project that is constantly evolving.

If you want to learn more about **sudo**, here are a few resources:

- The sudo website [4]
- The sudo blog [5]
- Follow us on Twitter [6]

Links

[1] https://www.sudo.ws/
[2] https://github.com/square/sudo_pair/
[3] https://www.syslog-ng.com/community/b/blog/posts/alerting-on-sudo-events-using-syslog-ng
[4] https://www.sudo.ws/
[5] https://blog.sudo.ws/
[6] https://twitter.com/sudoproject

Author

Peter is an engineer working as evangelist at Balabit, the company that developed syslog-ng. He assists distributions to maintain the syslog-ng package, follows bug trackers, helps users and talks regularly at conferences (SCALE, All Things Open, FOSDEM, LOADays, and others). In his limited free time he is interested in non-x86 architectures, and works on one of his PPC or ARM machines. Follow him at @PCzanik

Pylint: Making your Python code consistent

BY MOSHE ZADKA

Pylint is your friend when you want to avoid arguing about code complexity.

PYLINT IS A HIGH-LEVEL PYTHON style enforcer. While flake8 [1] and black [2] will take care of "local" style: where the newlines occur, how comments are formatted, or find issues like commented out code or bad practices in log formatting.

Pylint is extremely aggressive by default. It will offer strong opinions on everything from checking if declared interfaces are actually implemented to opportunities to refactor duplicate code, which can be a lot to a new user. One way of introducing it gently to a project, or a team, is to start by turning *all* checkers off, and then enabling checkers one by one. This is especially useful if you already use flake8, black, and mypy [3]: Pylint has quite a few checkers that overlap in functionality.

However, one of the things unique to Pylint is the ability to enforce higher-level issues: for example, number of lines in a function, or number of methods in a class.

These numbers might be different from project to project and can depend on the development team's preferences. However, once the team comes to an agreement about the parameters, it is useful to enforce those parameters using an automated tool. This is where Pylint shines.

Configuring Pylint

In order to start with an empty configuration, start your `.pylintrc` with

```
[MESSAGES CONTROL]

disable=all
```

This disables all Pylint messages. Since many of them are redundant, this makes sense. In Pylint, a `message` is a specific kind of warning.

You can check that all messages have been turned off by running `pylint`:

```
$ pylint <my package>
```

In general, it is not a great idea to add parameters to the `pylint` command-line: the best place to configure your `pylint` is the `.pylintrc`. In order to have it do *something* useful, we need to enable some messages.

In order to enable messages, add to your `.pylintrc`, under the `[MESSAGES CONTROL]`.

```
enable=<message>,

       ...
```

For the "messages" (what Pylint calls different kinds of warnings) that look useful. Some of my favorites include `too-many-lines`, `too-many-arguments`, and `too-many-branches`. All of those limit complexity of modules or functions, and serve as an objective check, without a human nitpicker needed, for code complexity measurement.

A *checker* is a source of *messages:* every message belongs to exactly one checker. Many of the most useful messages are under the design checker [4]. The default numbers are usually good, but tweaking the maximums is straightfoward: we can add a section called `DESIGN` in the `.pylintrc`.

```
[DESIGN]

max-args=7

max-locals=15
```

Another good source of useful messages is the refactoring checker. Some of my favorite messages to enable there are `consider-using-dict-comprehension`, `stop-iteration-return` (which looks for generators which use `raise StopIteration` when `return` is the correct way to stop the iteration). and `chained-comparison`, which will suggest using syntax like `1 <= x < 5` rather than the less obvious `1 <= x && x > 5`

Finally, an expensive checker, in terms of performance, but highly useful, is `similarities`. It is designed to enforce "Don't Repeat Yourself" (the DRY principle) by explicitly looking for copy-paste between different parts of the code. It only has one message to enable: `duplicate-code`. The default "minimum similarity lines" is set to 4. It is possible to set it to a different value using the `.pylintrc`.

```
[SIMILARITIES]

min-similarity-lines=3
```

Pylint makes code reviews easy

If you are sick of code reviews where you point out that a class is too complicated, or that two different functions are basically the same, add Pylint to your Continuous Integration [5] configuration, and only have the arguments about complexity guidelines for your project *once*.

Links

[1] https://opensource.com/article/19/5/python-flake8
[2] https://opensource.com/article/19/5/python-black
[3] https://opensource.com/article/19/5/python-mypy
[4] https://pylint.readthedocs.io/en/latest/technical_reference/features.html#design-checker
[5] https://opensource.com/business/15/7/six-continuous-integration-tools

Author

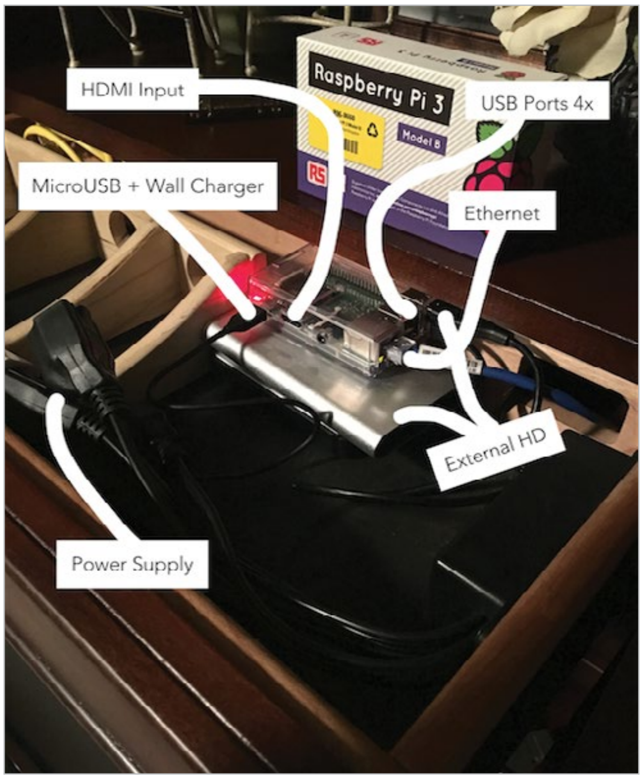
Moshe has been involved in the Linux community since 1998, helping in Linux "installation parties". He has been programming Python since 1999, and has contributed to the core Python interpreter. Moshe has been a DevOps/SRE since before those terms existed, caring deeply about software reliability, build reproducibility and other such things. He has worked in companies as small as three people and as big as tens of thousands — usually some place around where software meets system administration. Follow him at @moshezadka

How to set up a personal web server with a Raspberry Pi

BY RASPBERRYPIGUY

A PERSONAL WEB SERVER is "the cloud," except you own and control it as opposed to a large corporation.

Owning a little cloud has a lot of benefits, including customization, free storage, free Internet services, a path into open source software, high-quality security, full control over your content, the ability to make quick changes, a place to experiment with code, and much more. Most of these benefits are immeasurable, but financially these benefits can save you over $100 per month.



Raspberry Pi as a web server, by Raspberry Pi Guy, CC-BY-SA 4.0

I could have used AWS, but I prefer complete freedom, full control over security, and learning how things are built.

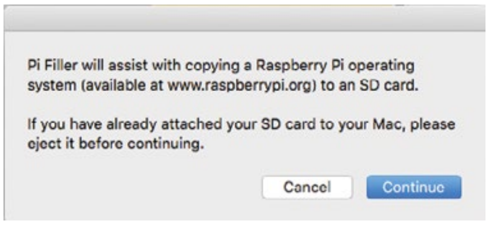
- Self web-hosting: No BlueHost or DreamHost
- Cloud storage: No Dropbox, Box, Google Drive, Microsoft Azure, iCloud, or AWS
- On-premise security
- HTTPS: Let's Encrypt
- Analytics: Google
- OpenVPN: Do not need private Internet access (at an estimated $7 per month)

Things I used:

- Raspberry Pi 3 Model B
- MicroSD Card (32GB recommended, Raspberry Pi Compatible SD Cards [1])
- USB microSD card reader
- Ethernet cable
- Router connected to Wi-Fi
- Raspberry Pi case
- Amazon Basics MicroUSB cable
- Apple wall charger
- USB mouse
- USB keyboard
- HDMI cable
- Monitor (with HDMI input)
- MacBook Pro

Step 1: Setting up the Raspberry Pi

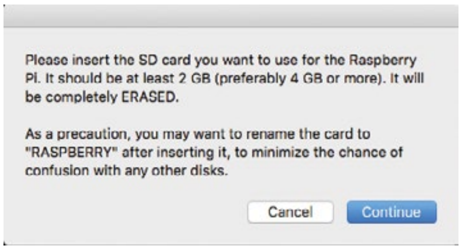
Download the most recent release of Raspbian (the Raspberry Pi operating system). Raspbian Jessie [2] ZIP version is ideal[1]. Unzip or extract the downloaded file. Copy it onto the SD card. Pi Filler [3] makes this process easy. Download Pi Filer 1.3 [4] or the most recent version. Unzip or extract the downloaded file and open it. You should be greeted with this prompt:



Pi Filler will assist with copying a Raspberry Pi operating system (available at www.raspberrypi.org) to an SD card.

If you have already attached your SD card to your Mac, please eject it before continuing.

Make sure the USB card reader has NOT been inserted yet. If it has, eject it. Proceed by clicking Continue. A file explorer should appear. Locate the uncompressed Raspberry Pi OS file from your Mac or PC and select it. You should see another prompt like the one pictured below:



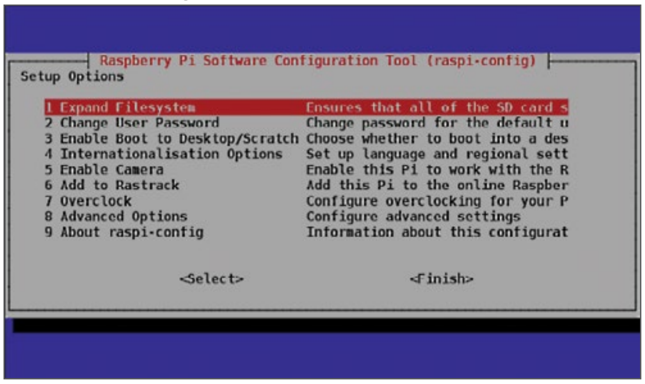
Please insert the SD card you want to use for the Raspberry Pi. It should be at least 2 GB (preferably 4 GB or more). It will be completely ERASED.

As a precaution, you may want to rename the card to "RASPBERRY" after inserting it, to minimize the chance of confusion with any other disks.

Insert the MicroSD card (32GB recommended, 16GB minimum) into the USB MicroSD Card Reader. Then insert the USB reader into the Mac or PC. You can rename the SD card to "Raspberry" to distinguish it from others. Click Continue. Make sure the SD card is empty. Pi Filler will *erase* all previous storage at runtime. If you need to back up the card, do so now. When you are ready to continue, the Raspbian OS will be written to the SD card. It should take between one to three minutes. Once the write is completed, eject the USB reader, remove the SD card, and insert it into the Raspberry Pi SD card slot. Give the Raspberry Pi power by plugging the power cord into the wall. It should start booting up. The Raspberry Pi default login is:

username: pi
password: raspberry

When the Raspberry Pi has completed booting for the first time, a configuration screen titled "Setup Options" should appear like the image below[2]:

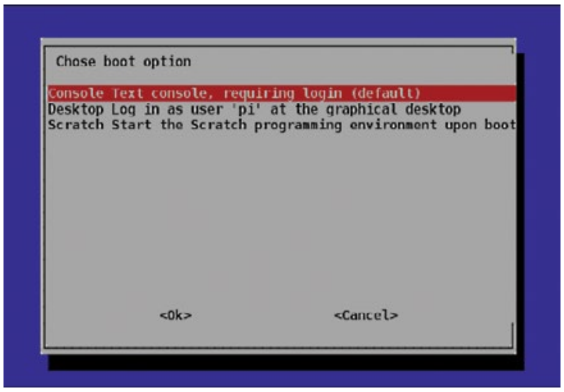


```
Raspberry Pi Software Configuration Tool (raspi-config)
Setup Options

 1 Expand Filesystem              Ensures that all of the SD card s
 2 Change User Password           Change password for the default u
 3 Enable Boot to Desktop/Scratch Choose whether to boot into a des
 4 Internationalisation Options   Set up language and regional sett
 5 Enable Camera                  Enable this Pi to work with the R
 6 Add to Rastrack                Add this Pi to the online Raspber
 7 Overclock                      Configure overclocking for your P
 8 Advanced Options               Configure advanced settings
 9 About raspi-config             Information about this configurat

              <Select>                       <Finish>
```

Select the "Expand Filesystem" option and hit the Enter key[3]. Also, I recommend selecting the second option, "Change User Password." It is important for security. It also personalizes your Raspberry Pi.

(Note: For an extra layer of security install fail2ban. Fail2Ban blocks suspicious requests coming from the internet. For example, if there are too many attempts to guess the password, it will block that IP address. It can be installed by typing into terminal: **$ sudo apt-get install fail2ban**)

Select the third option in the setup options list, "Enable Boot To Desktop/Scratch" and hit the Enter key. It will take you to another window titled "Choose boot option" as shown in the image below.



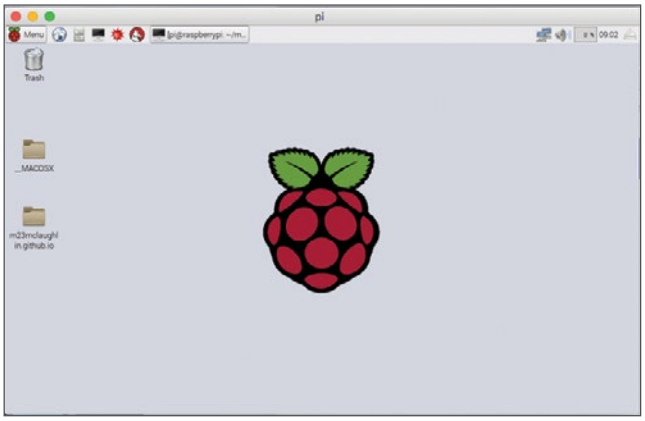
```
Chose boot option

Console Text console, requiring login (default)
Desktop Log in as user 'pi' at the graphical desktop
Scratch Start the Scratch programming environment upon boot

              <Ok>                    <Cancel>
```

In the "Choose boot option" window, select the second option, "Desktop log in as user 'pi' at the graphical desktop" and hit the Enter button[4]. Once this is done you will be taken back to the "Setup Options" page. If not, select the "OK" button at the bottom of this window and you will be taken back to the previous window.

Once both these steps are done, select the "Finish" button at the bottom of the page and it should reboot automatically. If it does not, then use the following command in the terminal to reboot.

```
$ sudo reboot
```

After the reboot from the previous step, if everything went well, you will end up on the desktop similar to the image below.



Once you are on the desktop, open a terminal and enter the following commands to update the firmware of the Raspberry Pi.

```
$ sudo apt-get update

$ sudo apt-get upgrade -y

$ sudo apt-get dist-upgrade -y

$ sudo rpi-update
```

This may take a few minutes. Now the Raspberry Pi is up-to-date and running.

Step 2: Configuring the Raspberry Pi

SSH, which stands for Secure Shell, is a cryptographic network protocol that lets you securely transfer data between your computer and your Raspberry Pi. You can control your Raspberry Pi from your Mac's command line without a monitor or keyboard.

To use SSH, first, you need your Pi's IP address. Open the terminal and type:

```
$ sudo ifconfig
```

If you are using Ethernet, look at the "eth0" section. If you are using Wi-Fi, look at the "wlan0" section.

Find "inet addr" followed by an IP address—something like 192.168.1.115, a common default IP I will use for the duration of this article.

With this address, open terminal and type:

```
$ ssh pi@192.168.1.115
```

For SSH on PC, see footnote[5].

Enter the default password "raspberry" when prompted, unless you changed it.

You are now logged in via SSH.

Remote desktop

Using a GUI (graphical user interface) is sometimes easier than a command line. On the Raspberry Pi's command line (using SSH) type:

```
$ sudo apt-get install xrdp
```

Xrdp supports the Microsoft Remote Desktop Client for Mac and PC.

On Mac, navigate to the app store and search for "Microsoft Remote Desktop." Download it. (For a PC, see footnote[6].)

After installation, search your Mac for a program called "Microsoft Remote Desktop." Open it. You should see this:

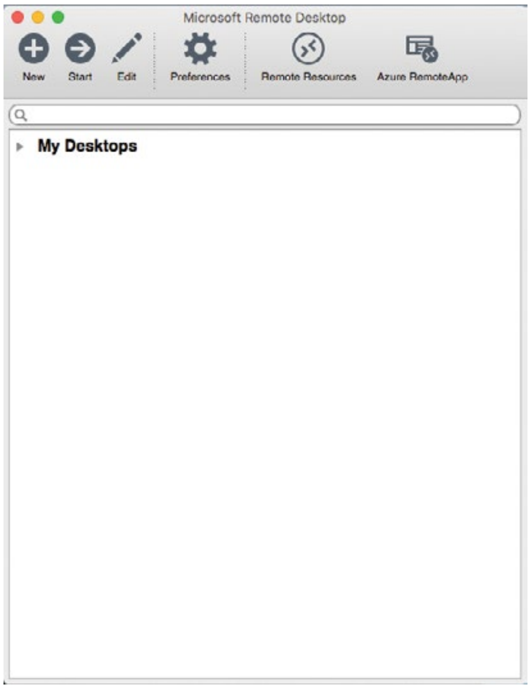
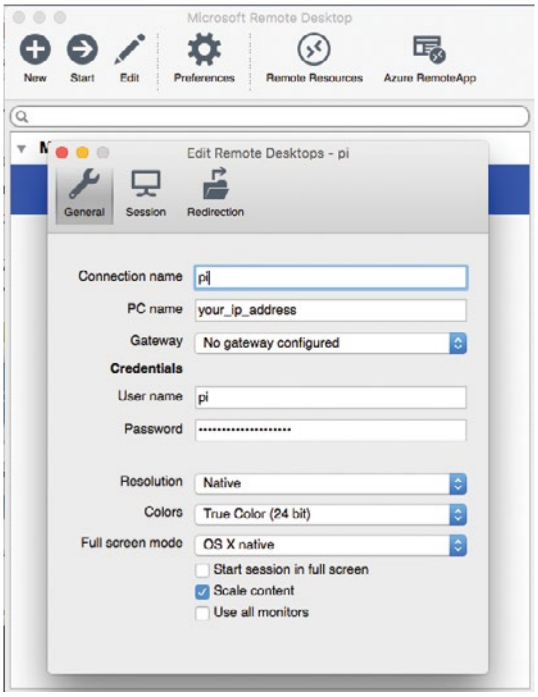


Image by Raspberry Pi Guy, CC BY-SA 4.0

Click "New" to set up a remote connection. Fill in the blanks as shown below.



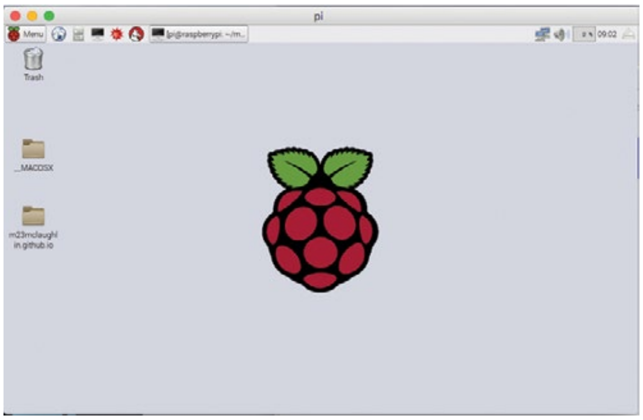
Connection name: pi
PC name: your_ip_address
Gateway: No gateway configured
Credentials
User name: pi
Password: ••••••••••••••••••••
Resolution: Native
Colors: True Color (24 bit)
Full screen mode: OS X native
☐ Start session in full screen
☑ Scale content
☐ Use all monitors

Image by Raspberry Pi Guy, CC BY-SA 4.0

Save it by exiting out of the "New" window.

You should now see the remote connection listed under "My Desktops." Double click it.

After briefly loading, you should see your Raspberry Pi desktop in a window on your screen, which looks like this:



Perfect. Now, you don't need a separate mouse, keyboard, or monitor to control the Pi. This is a much more lightweight setup.

Static local IP address

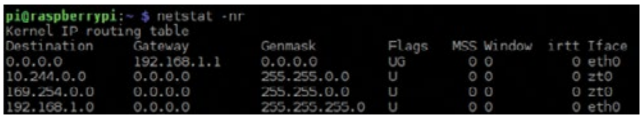
Sometimes the local IP address 192.168.1.115 will change. We need to make it static. Type:

```
$ sudo ifconfig
```

Write down from the "eth0" section or the "wlan0" section, the "inet addr" (Pi's current IP), the "bcast" (the broadcast IP range), and the "mask" (subnet mask address). Then, type:

```
$ netstat -nr
```

Write down the "destination" and the "gateway/network."



```
pi@raspberrypi:~ $ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
10.244.0.0      0.0.0.0         255.255.0.0     U         0 0          0 zt0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 zt0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

The cumulative records should look something like this:

```
net address 192.168.1.115
bcast 192.168.1.255
mask 255.255.255.0
gateway 192.168.1.1
network 192.168.1.1
destination 192.168.1.0
```

With this information, you can set a static internal IP easily. Type:

```
$ sudo nano /etc/dhcpcd.conf
```

Do not use **/etc/network/interfaces**.

Then all you need to do is append this to the bottom of the file, substituting the correct IP address you want.

```
interface eth0
static ip_address=192.168.1.115
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Once you have set the static internal IP address, reboot the Raspberry Pi with:

```
$ sudo reboot
```

After rebooting, from terminal type:

```
$ sudo ifconfig
```

Your new static settings should appear for your Raspberry Pi.

Static global IP address

If your ISP (internet service provider) has already given you a static external IP address, you can skip ahead to the port forwarding section. If not, continue reading.

You have set up SSH, a remote desktop, and a static internal IP address, so now computers inside the local network will know where to find the Pi. But you still can't access your Raspberry Pi from outside the local Wi-Fi network. You need your Raspberry Pi to be accessible publicly from anywhere on the Internet. This requires a static external IP address[7].

It can be a sensitive process initially. Call your ISP and request a static external (sometimes referred to as static global) IP address. The ISP holds the decision-making power, so I would be extremely careful dealing with them. They may refuse your static external IP address request. If they do, you can't fault the ISP because there is a legal and operational risk with this type of request. They particularly do not want customers running medium- or large-scale Internet services. They might explicitly ask why you need a static external IP address. It is probably best to be honest and tell them you plan on hosting a low-traffic personal website or a similar small not-for-profit internet service. If all goes well, they should open a ticket and call you in a week or two with an address.

Port forwarding

This newly obtained static global IP address your ISP assigned is for accessing the router. The Raspberry Pi is still unreachable. You need to set up port forwarding to access the Raspberry Pi specifically.

Ports are virtual pathways where information travels on the Internet. You sometimes need to forward a port in order to make a computer, like the Raspberry Pi, accessible to the Internet because it is behind a network router. A YouTube video titled What is TCP/IP, port, routing, intranet, firewall, Internet [5] by VollmilchTV helped me visually understand ports.

Port forwarding can be used for projects like a Raspberry Pi web server, or applications like VoIP or peer-to-peer downloading. There are 65,000+ ports [6] to choose from, so you can assign a different port for every Internet application you build.

The way to set up port forwarding can depend on your router. If you have a Linksys, a YouTube video titled *How to go online with your Apache Ubuntu server* [7] by Gabriel

Ramirez explains how to set it up. If you don't have a Linksys, read the documentation that comes with your router in order to customize and define ports to forward.

You will need to port forward for SSH as well as the remote desktop.

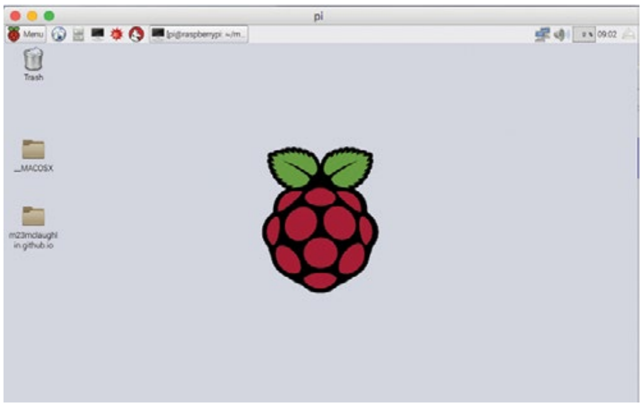
Once you believe you have port forwarding configured, check to see if it is working via SSH by typing:

```
$ ssh pi@your_global_ip_address
```

It should prompt you for the password.

Check to see if port forwarding is working for the remote desktop as well. Open Microsoft Remote Desktop. Your previous remote connection settings should be saved, but you need to update the "PC name" field with the static external IP address (for example, 195.198.227.116) instead of the static internal address (for example, 192.168.1.115).

Now, try connecting via remote desktop. It should briefly load and arrive at the Pi's desktop.



Good job. The Raspberry Pi is now accessible from the Internet and ready for advanced projects.

As a bonus option, you can maintain two remote connections to your Pi. One via the Internet and the other via the LAN (local area network). It's easy to set up. In Microsoft Remote Desktop, keep one remote connection called "Pi Internet" and another called "Pi Local." Configure Pi Internet's "PC name" to the static external IP address—for example, 195.198.227.116. Configure Pi Local's "PC name" to the static internal IP address—for example, 192.168.1.115. Now, you have the option to connect globally or locally.

If you have not seen it already, watch How to go online with your Apache Ubuntu server [7] by Gabriel Ramirez as a transition into Project 2. It will show you the technical architecture behind your project. In our case, you are using a Raspberry Pi instead of an Ubuntu server. The dynamic DNS sits between the domain company and your router, which Ramirez omits. Beside this subtlety, the video is spot on when explaining visually how the system works. You might notice this tutorial covers the Raspberry Pi setup and port forwarding, which is the server-side or back end. See the original source for more advanced projects covering the domain name, dynamic DNS, Jekyll (static HTML generator), and Apache (web hosting), which is the client-side or front end.

Footnotes

[1] I do not recommend starting with the NOOBS operating system. I prefer starting with the fully functional Raspbian Jessie operating system.

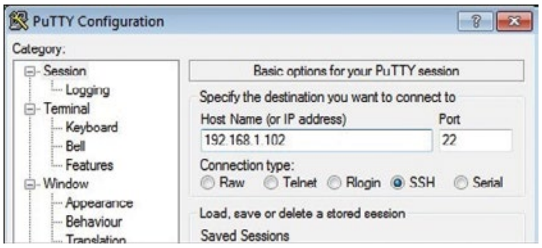
[2] If "Setup Options" does not pop up, you can always find it by opening Terminal and executing this command:

```
$ sudo-raspi-config
```

[3] We do this to make use of all the space present on the SD card as a full partition. All this does is expand the operating system to fit the entire space on the SD card, which can then be used as storage memory for the Raspberry Pi.

[4] We do this because we want to boot into a familiar desktop environment. If we do not do this step, the Raspberry Pi boots into a terminal each time with no GUI.

[5]



PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.102 Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Download and run PuTTY [8] or another SSH client for Windows. Enter your IP address in the field, as shown in the above screenshot. Keep the default port at 22. Hit Enter, and PuTTY will open a terminal window, which will prompt you for your username and password. Fill those in, and begin working remotely on your Pi.

[6] If it is not already installed, download Microsoft Remote Desktop [9]. Search your computer for Microsoft Remote Desktop. Run it. Input the IP address when prompted. Next, an xrdp window will pop up, prompting you for your username and password.

[7] The router has a dynamically assigned external IP address, so in theory, it can be reached from the Internet momentarily, but you'll need the help of your ISP to make it permanently accessible. If this was not the case, you would need to reconfigure the remote connection on each use.

For the original source, visit Mitchell McLaughlin's Full-Stack Computer Projects [10].

Links

[1] http://elinux.org/RPi_SD_cards
[2] https://www.raspberrypi.org/downloads/raspbian/
[3] http://ivanx.com/raspberrypi/
[4] http://ivanx.com/raspberrypi/files/PiFiller.zip
[5] https://www.youtube.com/watch?v=iskxw6T1Wb8
[6] https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
[7] https://www.youtube.com/watch?v=i1vB7JnPvuE#t=07m08s
[8] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[9] https://www.microsoft.com/en-us/store/apps/microsoft-remote-desktop/9wzdncrfj3ps
[10] https://mitchellmclaughlin.com/server.html

Author
RaspberryPiGuy

You can't have DevOps without open source

BY JEN KRIEGER

YOU PROBABLY THINK I'm going to talk about all the reasons why you should use open source tooling as the foundation for an effective DevOps culture in your organization, but that's not what this is about. Not to marginalize the complexity of the challenges faced by the team I work with [1], but I have confidence that the engineers are going to figure the tooling part out. Believe it or not, the daunting part is wrapped in cultural change.

I have spent a significant amount of time reading about cultural change [2], what you need to have an effective DevOps community [3], how you build high functioning teams [4], and asking the question, "How do I DevOp?" [5] The ideas I've read have given me a few new things to stick in my tool belt. However, nothing has resonated with me as much as this:

The open source way is...

> **Open exchange**
> We can learn more from each other when information is open. A free exchange of ideas is critical to creating an environment where people are allowed to learn and use existing information toward creating new ideas.
>
> **Participation**
> When we are free to collaborate, we create. We can solve problems that no one person may be able to solve on their own.
>
> **Rapid prototyping**
> Rapid prototypes can lead to rapid failures, but that leads to better solutions found faster. When you're free to experiment, you can look at problems in new ways and look for answers in new places. You can learn by doing.
>
> **Meritocracy**
> In a meritocracy, the best ideas win. In a meritocracy, everyone has access to the same information. Successful work determines which projects rise and gather effort from the community.
>
> **Community**
> Communities are formed around a common purpose. They bring together diverse ideas and share work. Together, a global community can create beyond the capabilities of any one individual. It multiplies effort and shares the work. Together, we can do more.

That is how you get an effective DevOps culture. **You embrace the open source way.**

If you didn't fist pump, remember a job you have had in the past where you felt like this guy, and then read that again.

Open exchange, participation, community

My career before Red Hat was filled with statements like "just do your job" and "that's just the way it is, you can't change it." I have viscerally felt the horrible feeling of having to tell someone that they couldn't have their idea, not because it wouldn't solve many problems, but because they didn't know the right people or know the best way to get their idea across.

When you have an ***open exchange*** and everyone is encouraged to ***participate***:

- People talk to one another and build off of each other's collective experience.
- People earn each other's respect while working together.
- People are less likely to say, "It's so and so's job, not mine," when they know and respect each other.

No team is going to believe they are in an optimal working environment if someone is saying, "Work together or else." Yes, they will work together, but wouldn't you prefer to give them a reason to want to work together? The reason can be as simple as helping two individuals connect on similar interests, or as hard as getting teams with a long-standing history of disliking one another to work together. But the key element here is a path to empathy and having respect for those you spend your week with. At the end of the day, you need to foster an environment where it is OK to have an opinion, OK to have an idea... OK to share.

Even on the team [6] I'm working on, where the values of the open source way are individually embraced, we have to work hard to continue to cultivate that global sense of community and collaboration. *Even at Red Hat it is hard.* I strive on a daily basis to provide as much visibility into what our team is working on regardless of how trivial it may seem. The result of this? People are sharing their ideas and helping to build something we can all be proud of and support.

Rapid prototyping and meritocracy

A brilliant thing happened very early on with the team of engineers I'm working with; they reminded me about the importance of just getting something done. We spent a good bit of time trying to muddle through the mountain of things we could work on and the output of that? Well, frankly—it was a lot of talking.

Being able to show someone what you are doing, and receive feedback on that thing, is so much more satisfying than the talking. **Rapid prototyping?** Getting to see the code [7], instead of an idea disappearing into a requirements hole and resurfacing at QA, is so satisfying I can't shout about that enough. In my past experience with closed source projects, I haven't seen code delivered, input received for changes, and subsequently modified very quickly. So, that continues to be transformational for me.

Don't get me wrong, this is not an easy thing to master. Early on the team made a technical decision to move forward with a custom application [8] that could provide certain system information [9] (like software versions running) without requiring root access to servers. I understand that there are tools that already do that, but the team wanted something *quick to ease the pain*. After we were done, we agreed we would start looking into longer-term solutions to help. We are still feeling the pain from that decision; **meritocracy** is in full-on mode here where this project is concerned, and no one is shy about sharing the details about the impact it has had on their servers. However, I can still say the work was successful because at the end of the day it delighted the people who needed it the most, and it certainly gets people talking to me.

Consider this

How hard do you think a DevOps transformation will be if employees don't feel comfortable enough to share ideas, are being told not to collaborate because it isn't their job, or it is implied that their financial well-being is wrapped around performing one function and there is no emphasis placed on whole systems thinking?

The open source way isn't an easy button for success. However, what it can do is provide a set of values for an individual and a group to follow that can set your organization on the path towards an effective DevOps community. Do me a favor and go back and read those values again. Are you and your organization open enough to embrace the open source way?

Links

[1] http://developerblog.redhat.com/tag/inception/
[2] http://www.amazon.com/Leading-Change-With-Preface-Author/dp/1422186431
[3] http://itrevolution.com/devops-culture-part-1/
[4] http://www.slideshare.net/reed2001/culture-1798664
[5] http://blog.mihasya.com/2013/06/11/how-do-i-devops.html
[6] https://twitter.com/mrry550/lists/team-inception
[7] https://github.com/RHInception
[8] https://github.com/RHInception/talook
[9] https://github.com/RHInception/jsonstats

Author

Jen Krieger - Jen Krieger is Chief Agile Architect at Red Hat. Most of her 20+ year career has been in software development representing many roles throughout the waterfall and agile lifecycles. At Red Hat, she led a department-wide DevOps movement focusing on CI/CD best practices. Most recently, she worked with with the Project Atomic & OpenShift teams. Now Jen is guiding teams across the company into agility in a way that respects and supports Red Hat's commitment to Open Source. At Red Hat, she led a department-wide DevOps movement focusing on CI/CD best practices. Most recently, she worked with with the Project Atomic & OpenShift teams. Now Jen is guiding teams across the company into agility in a way that respects and supports Red Hat's commitment to Open Source. Follow her at @mrry550

Why is Kubernetes so popular?

BY ANURAG GUPTA

The Google-developed container management system has quickly become one of the biggest success stories in open source history.

KUBERNETES, [1] AN OPEN SOURCE container management system, has surged in popularity in the past several years. Used by the largest enterprises in a wide range of industries for mission-critical tasks, it has become one of the biggest success stories in open source. How did that happen? And what is it about Kubernetes that explains its widespread adoption?

Kubernetes' backstory: Origins in Google's Borg system

As the computing world became more distributed, more network-based, and more about cloud computing, we saw large, monolithic apps slowly transform into multiple, agile microservices. These microservices allowed users to individually scale key functions of an application and handle millions and millions of customers. On top of this paradigm change, we saw technologies like Docker containers emerge in the enterprise, creating a consistent, portable, and easy way for users to quickly build these microservices.

While Docker continued to thrive, managing these microservices and containers became a paramount requirement. That's when Google, which had been running container-based infrastructure for many years, made the bold decision to open source an in-house project called Borg [2]. The Borg system was key to running Google's services, such as Google Search and Gmail. This decision by Google to open source its infrastructure has created a way for any company in the world to run its infrastructure like one of the top companies in the world.

One of the biggest open source communities

After its open source release, Kubernetes found itself competing with other container-management systems, namely Docker Swarm and Apache Mesos. One of the reasons Kubernetes surged past these other systems in recent months is the community and support behind the system: It's one of the largest open source communities (more than 27,000+ stars on GitHub); has contributions from thousands of organizations (1,409 contributors); and is housed within a large, neutral open source foundation, the Cloud Native Computing Foundation (CNCF) [3].

The CNCF, which is also part of the larger Linux Foundation, has some of the top enterprise companies as members, including Microsoft, Google, and Amazon Web Services. Additionally, the ranks of enterprise members in CNCF continue to grow, with SAP and Oracle joining as Platinum members within the past couple of months. These companies joining the CNCF, where the Kubernetes project is front and center, is a testament to how much these enterprises are betting on the community to deliver a portion of their cloud strategy.

The enterprise community around Kubernetes has also surged, with vendors providing enterprise versions with added security, manageability, and support. Red Hat, CoreOS, and Platform 9 are some of the few that have made Enterprise Kubernetes offerings key to their strategy going forward and have invested heavily in ensuring the open source project continues to be maintained.

Delivering the benefits of the hybrid cloud

Yet another reason why enterprises are adopting Kubernetes at such a breakneck pace is that Kubernetes can work in any cloud. With most enterprises sharing assets between their existing on-premises datacenters and the public cloud, the need for hybrid cloud technologies is critical.

Kubernetes can be deployed in a company's pre-existing datacenter on premises, in one of the many public cloud environments, and even run as a service. Because Kubernetes abstracts the underlying infrastructure layer, developers can focus on building applications, then deploy them to any of those environments. This helps accelerate a company's Kubernetes adoption, because it can run Kubernetes on-premises while continuing to build out its cloud strategy.

Real-world use cases

Another reason Kubernetes continues to surge is that major corporations are using the technology to tackle some of the industry's largest challenges. Capital One, Pearson Education, and Ancestry.com are just a few of the companies that have published Kubernetes use cases [4].

Pokemon Go [5] is one of the most-popular publicized use cases showing the power of Kubernetes. Before its release, the online multiplayer game was expected to be reasonably popular. But as soon as it launched, it took off like a rocket, garnering 50 times the expected traffic. By using Kubernetes as the infrastructure overlay on top of Google Cloud, Pokemon Go could scale massively to keep up with the unexpected demand.

What started out as an open source project from Google—backed by 15 years of experience running Google services and a heritage from Google Borg—Kubernetes is now open source software that is part of a big foundation (CNCF) with many enterprise members. It continues to grow in popularity and is being widely used with mission-critical apps in finance, in massive multiplayer online games like Pokemon Go, and by educational companies and traditional enterprise IT. Considered together, all signs point to Kubernetes continuing to grow in popularity and remaining one of the biggest success stories in open source.

Links

[1] https://kubernetes.io/
[2] http://queue.acm.org/detail.cfm?id=2898444
[3] https://www.cncf.io/
[4] https://kubernetes.io/case-studies/
[5] https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html

Author

Anurag Gupta is a Product Manager at Treasure Data driving the development of the unified logging layer, Fluentd Enterprise. Anurag has worked on large data technologies including Azure Log Analytics, and enterprise IT services such as Microsoft System Center. Follow him at @Anurag_Gup

Four Linux distros for kids

BY ASEEM SHARMA

I CAN SEE THE BRIGHTNESS of curiosity in my six year old niece Shuchi's eyes when she explores a mobile phone or manipulates the idiot box with its remote control or becomes creatively destructive with any other electronic device. She, like a lot of kids her age, love experimenting.

This curiosity reaches its peak when she sits in front of my laptop or her father's laptop. A lot of times, however, I observe that she is lost in complicated applications that are suitable only to adults. An operating system that an adult uses and the system running it can look like a beast to a lot of kids. These applications are beyond the comprehension of very young kids and do not provide an ideal (and playful) introduction to computers. Futher, adults' laptops and tablets do not serve as a good learning environment for any kid (younger or older) who is just onboarding into the world of computing. Besides, letting a kid run wild on a computer with an online connection can be daunting for the parents.

As a big kid myself, and an open source software enthusiast for over four years now, I like exploring and experimenting with different software solutions. Pertaining to the problem of finding and setting up an ideal system for my young niece, I found that the open source Linux community has created specialized operating systems and environments for kids. Plus, setting up these systems is a breeze.

Why should kids learn Linux

I have reached a conclusive opinion at this point in my life that children should be exposed to the power of Linux early on. Two of the reasons are...

For the future of computing

I recently read the article, *A year of Linux desktop at Westcliff High School* [1], which is an excellent piece by Stu Jarvis in which Malcolm Moore replies to a question by stating, "Here is a survey that reports in 2000, 97% of computing devices had Windows installed, but now with tablets and phones, etc., Windows is only on 20% of computing devices, and in the world of big iron, Linux reigns supreme. We specialize in science and engineering and want our students to go on to do great things like start the next Google or collapse the universe at CERN. In those environments, they will certainly need to know Linux."

Linux runs some of the most complex infrastructures in the world. For anyone even remotely interested in a career in technology, learning Linux will be a definite asset. Besides that, the adoption of Linux is massive and ubiquitous. Consider this:

- Linux powers international space stations
- Linux powers the technology in new cars like Tesla and Cadillac
- Linux powers air traffic control systems
- Google, Facebook, Twitter, all use Linux
- 9 out of 10 supercomputers in the world run on Linux

There is a rational reason that initiatives like One Laptop per Child [2], which in my opinion is one of the most powerful programs today that is working to bridge the digital divide, use Linux based systems.

For customization and variety

Learning at an early age can be best enhanced in an environment that encourages exploration. There is no other operating system that offers such variety and autonomy to customize the system based on specific needs like Linux. Like toys and clothes for kids, the Linux community has developed specific operating systems that can offer them a fun learning environment. I believe that to boost curiosity in kids, it is important to create a set up that gives them a feeling of wonder.

Programs to teach kids Linux

There are many different varieties of environments that the Linux community has designed for the children, and I haven't yet explored them all, but of the ones I did, you should be able to find a great solution for teaching a kid you know about Linux and computing.

Qimo

http://www.qimo4kids.com/

Qimo for kids is a Ubuntu-based distribution designed specifically for children. The operating system comes pre-installed with a lot of educational applications for children ages 3 years and older. It comes with GCompris, a perfect suite for children aged 3 to 10 years. It consists of over 100 educational games that teaches basic computer use, reading, art history, telling time, and drawing pictures, as well as Childs Play, a collection of memory-building games.

One of the things I like best about this distribution is that it uses XFCE desktop , which is a lightweight desktop that can be installed on older machines. The hardware requirements are low and it is absurdly easy to repurpose an old laptop or a desktop system. We had an old PC at home, and Qimo resurrected it. This operating system was my choice for my niece because of its simple child friendly cartoon desktop and assortment of educational applications.

Sugar

https://www.sugarlabs.org/

Sugar was designed for the One Laptop per Child program. It is an easy to use and kid-friendly operating system. Children who love exploring will figure out things quickly in this environment, even if they cannot read or write yet.

From Sugar Labs:

> Information is about nouns; learning is about verbs. The Sugar interface, in its departure from the desktop metaphor for computing, is the first serious attempt to create a user interface that is based on both cognitive and social constructivism: learners should engage in authentic exploration and collaboration. It is based on three very simple principles about what makes us human.

Ubermix

http://www.ubermix.org/

Ubermix is extensively used in schools. The system was designed to store user data and software in seperate partitions. So, in case the computer malfunctions, the user can wipe out the operating system and resotre fresh copies quickly. From Ubermix founder, Jim Klein, in an Opensource.com interview [3]:

> Ubermix comes pre-loaded with a number of applications for education, productivity, design, programming, Internet, and multimedia construction. Education oriented applications like Celestia, Stellarium, Scratch, VirtualLab Microscope, Geogebra, iGNUit, and Klavaro, as well as educational games like TuxMath, TuxTyping, gMult, and Numpty Physics all bring with them plenty of opportunities to learn.
>
> Internet applications we all know and love, like Firefox, Thunderbird, Chrome, Google Earth, and Skype are all there. Common productivity apps like LibreOffice, NitroTasks, Planner Project Management, VYM (View Your Mind), and Zim Desktop Wiki are too. Kids interested in design will find the GIMP, Inkscape, Scribus, Dia, Agave, and even TuxPaint for the younger ones. And apps like Audacity, Openshot, Pencil, and ffDiaporama help round out the media offerings. These, and many more, make Ubermix a powerful launchpad for student creativity and learning.

Edubuntu

http://www.edubuntu.org/

Formally the Ubuntu Education Edition, Edubuntu was developed in collaboration with educators and teachers. It embeds a variety of educational programs and a suitable learning environment. An advantage to it is access to the Ubuntu software repository. The education community has extensively used this operating system in schools and organizations to provide an enriched learning environment for their students. It's a great operating system to teach older children about Linux; it can have a steeper learning curve in comparison to Qimo and Sugar.

Links

[1] https://opensource.com/education/13/7/linux-westcliff-high-school
[2] http://one.laptop.org/
[3] https://opensource.com/education/13/3/ubermix-linux

Author

Aseem is a graduate of Conrad Business, Entrepreneurship and Technology Center, Faculty of Engineering, University of Waterloo, Canada. He also holds a masters in computers application from Guru Nanak Dev University, Punjab, India. On Opensource.com, he serves as an author. He also blogs at http://aseemsharma.info/. Follow him at @aseem_sharma

6 remarkable features of the new United Nations open source initiative

BY FRANK KARLITSCHE

What does it mean when the UN goes open source?

THREE MONTHS AGO the United Nations asked me to join a new advisory board to help them develop their open source strategy and policy. I'm honored to have the opportunity to work together with a group of established experts in open source licensing and policy areas.

The United Nations wants to make technology, software, and intellectual property available to everyone, including developing countries. Open source and free software are great tools to achieve this goal since open source is all about empowering people and global collaboration while protecting the personal data and privacy of users. So, the United Nations and the open source community share the same values.

This new open source strategy and policy is developed by the United Nations Technology Innovation Labs (UNTIL) [1]. Last month, we had our first in-person meeting in Helsinki in the UNTIL offices. I find this initiative remarkable for several reasons:

- **Sharing:** The United Nations wants to have a positive impact on everyone on this planet. For that goal, it is important that software, data, and services are available for everyone independent of their language, budget, education, or other factors. Open source is perfect to guarantee that result.
- **Contributing:** It should be possible that everyone can contribute to the software, data, and services of the United Nations. The goal is to not depend on a single software vendor alone, but instead, build a bigger ecosystem that drives innovation together.
- **Empowering:** Open source makes it possible for underdeveloped countries and regions to foster local companies and expertise by building on top of existing open source software—standing on the shoulders of giants.
- **Sustainability:** Open source guarantees more sustainable software, data, and services by not relying on a single entity to support, maintain, and develop it. Open source helps to avoid a single point of failure by creating an equal playing field for everyone.
- **Security:** Open source software is more secure than proprietary software because the code can be constantly reviewed and audited. This fact is especially important for security-sensitive applications that require transparency and openness [2].
- **Decentralization:** An open source strategy enables decentralized hosting of software and data. This fact makes it possible to be compliant with all data protection and privacy regulations and enables a more free and open internet.

We discussed that a fair business model like the one from Nextcloud should be encouraged and recommended. Specifically, we discussed that that 100% of the code should be placed under an OSI-approved open source license [3]. There should be no open core, proprietary extensions, dual licensing, or other limited-access components to ensure that everyone is on the same playing field.

I'm excited to have the opportunity to advise the United Nations in this matter, and I hope to have a positive influence on the future of IT, especially in developing countries.

Links

[1] https://until.un.org/
[2] https://until.un.org/content/governance
[3] https://opensource.org/licenses

Author

Frank Karlitschek is a long time open source contributor and former board member of the KDE e.V. He managed engineering teams for over 20 years and worked as head of unit and managing director at different internet companies. In 2001 he created the openDesktop.org social network as well as GTK-Apps.org, GNOME-Look.org, KDE-Apps.org and other 'AppStores' before AppStores existed. He founded ownCloud in 2010 and the successor Nextcloud in 2016 to create a fully open source and decentralized alternative to big centralized cloud companies. Frank was an invited expert at the W3C to help to create the ActivityPub standard. Frank has spoken at MIT, CERN, Harvard and ETH and keynoted LinuxCon, Latinoware, FOSSASIA, Campus Party and many other conferences. Frank is the founder and CEO of Nextcloud GmbH. He is also a fellow of Open Forum Europe and an advisor to the United Nations. Follow him at @fkarlitschek